Performance Analysis of Wormhole Switched Interconnection Networks with Virtual Channels and Finite Buffers

A Thesis Submitted

by

Nasser Alzeidi

for

The Degree of Doctor of Philosophy

to

The Faculty of Information and Mathematical Sciences
University of Glasgow

Abstract

An efficient interconnection network that provides high bandwidth and low latency inter-processor communication is critical to harness fully the computational power of large scale multicomputers. *K*-ary *n*-cube networks have been widely adopted in contemporary multicomputers due to their desirable properties. As such, the present study focuses on a performance analysis of *k*-ary *n*-cubes employing wormhole switching, virtual channels, and adaptive routing. The objective of this dissertation is twofold: to examine the performance of these networks, and to compare the performance merits of various topologies under different working conditions, by means of analytical modelling.

Most existing analytical models reported in the literature have used a method originally proposed by Dally to capture the effects of virtual channels on network performance. This method is based on a Markov chain and it has been shown that its prediction accuracy degrades as traffic increases. Moreover, these studies have also constrained the buffer capacity to a single flit per channel, a simplifying assumption that has often been invoked to ease the derivation of the analytical models. Motivated by these observations, the first part of this research proposes a new method for modelling virtual channels, based on an M/G/1 queue. Owing to the generality of this method, Dally's method is shown to be a special case when the message service time is exponentially distributed. The second part of this research uses theoretical results of queuing systems to relax the single-flit buffer assumption. New analytical models are then proposed to capture the effects of deploying arbitrary size buffers on the performance of deterministic and adaptive routing algorithms. Simulation experiments reveal that results from the proposed analytical models are in close agreement with those obtained through simulation.

Building on these new analytical models, the third part of this research compares the relative performance merits of k-ary n-cubes under different operating conditions, in the presence of finite size buffers and multiple virtual channels. Namely, the analysis first revisits the relative performance merits of the well-known 2D torus, 3D torus and hypercube under different implementation constraints. The analysis has then been extended to investigate the performance impact of arranging the total buffer space, allocated to a physical channel, into multiple virtual channels. Finally, the performance of adaptive routing has been compared to that of deterministic routing.

While previous similar studies have only taken account of channel and router costs, the present analysis incorporates different intra-router delays, as well, and thus generates more realistic results. In fact, the results of this research differ notably from those reported in previous studies, illustrating the sensitivity of such studies to the level of detail, degree of accuracy and the realism of the assumptions adopted.

to Fida
to my parents
to my wife and son
to my brothers and sisters

for all their endless love, support and encouragement

Acknowledgements

First and foremost, I must thank the almighty Allah (SWT) who has again helped me to surmount another important challenge in my life.

I cannot fully express my gratitude to my supervisory team: *Dr. M Ould-Khaoua* and *Dr. L. M Mackenzie* for their exceptional support and guidance. Their constructive comments and insightful reviews have enabled me to progress in this research. I've been fortunate to learn from their vast experience.

I am highly indebted to the Sultan Qaboos University, Sultanate of Oman, for the financial support and for granting me a scholarship leave to pursue my higher studies.

I must thank the Faculty of Information and Mathematical Sciences and the Department of Computing Science at University of Glasgow for creating such an excellent research environment. My thanks are also due to my caring colleagues and friends here in the UK and back home.

My words will fail to express my heartfelt thanks and appreciation to my parents for the unending love, protection and encouragement throughout my life. They have always been my greatest source of inspiration. My only hope is that this achievement is up to their expectations. Thanks also to my brothers and sisters for their unconditional support.

Finally, I would be remiss if I did not mention the woman who really touched my life; my beloved wife *Amna*. Without a doubt she is the most astonishingly talented, complacent, tolerant and encouraging woman. I also thank my little son, *Azzam*, for all the happiness he has given me, and for coping with the long fatherless evenings and weekends when I was doing my research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The performance of digital systems today is limited by communication or interconnection, not by logic or memory [42, 49]. Interconnection networks, the hardware fabric supporting communication among individual components of these systems, have evolved rapidly in response to this communication bottleneck. Originally developed for the demanding communication requirements of multicomputers, interconnection networks have started to replace buses as the standard system-level interconnection [42]. They are also replacing dedicated wiring in special-purpose systems as designers discover that routing messages via an interconnection network is faster and more economic than using dedicated wiring to interconnect different components in these systems [41, 99]. Today, interconnection networks can be found in systems ranging from large supercomputers [3, 6, 53, 105, 109, 110, 133, 140, 147] to small embedded system-on-chip [86, 87, 116, 142] architectures. The efficiency of the interconnection network is crucial to the overall performance of such systems [42, 49, 51, 116].

This chapter starts with a brief introduction to interconnection networks and the most important factors affecting their performance. This is then followed by a discussion about the performance evaluation tool that has been used during the course of this research, namely analytical modelling. Related work is then outlined and motivations explained. The thesis statement is then presented and the main contributions of this research work are summarised. Finally, an overview of the subsequent chapters in this PhD dissertation is presented.

1.1. Interconnection networks

An interconnection network can be viewed as a system that transports data between individual components so that they can coordinate operation to accomplish collectively a specified task. The network is composed of many components: buffers, channels, switches, and controls that work together to deliver data [42, 49]. Networks meeting this broad definition occur at many scales. On-chip networks may deliver data between memory arrays, registers, and arithmetic units within a single processor. Board-level and system-level networks tie processors to memories or input ports to output ports. Finally, local-area and wide-area networks connect disparate systems within an enterprise or around the globe [42].

Parallel computers with multiple processors are generally considered to be the only feasible way of achieving the ever-growing computational and storage requirements of many applications, especially in the fields of science and engineering [33, 49, 57, 58, 61]. A powerful interconnection network is the key to harness fully the computational power offered by parallel computers. Interconnection networks in these systems can distinctively be classified into two main categories: *direct interconnection networks* and *indirect interconnection networks*. In the former (also called point-to-point networks [49]) each node has a point-to-point, or direct, connection to some number of other nodes, called neighbours, allowing for direct communication between nodes. The torus, binary n-cube (or the hypercube) and mesh are common direct interconnection networks that have been implemented in commercial and experimental machines. Some examples of these machines include IBM Blue Gene/L [3], Alpha21364 router [105], J-machine [109], Cray XT3 [147], Cray T3D [68], Cray T3E [133], Cray X1 [110, 140] and SGI Origin [53]. Figure 1.1 shows some of the most commonly used direct interconnection networks.

In contrast to direct networks, indirect networks do not provide direct connections between processing (and also memory) elements [42, 49]. Instead, multiple intermediate stages of switching elements are responsible for moving data and control messages between the

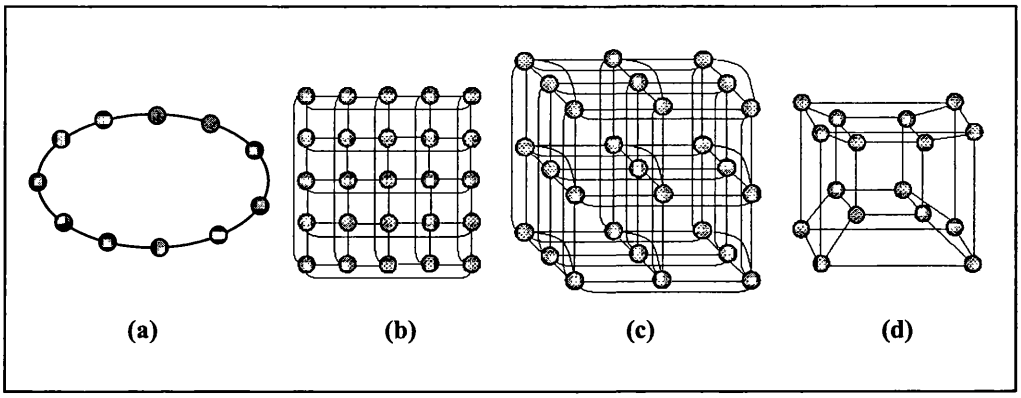


Figure 1.1: Some popular topologies of direct interconnection networks: (a) the ring (with 11 nodes), (b) the 2D torus (5x5 nodes), (c) the 3D torus (3x3x3 nodes), and (d) the 4D hypercube (or 4-cube).

processing elements. Early experimental and commercial machines have deployed indirect interconnection networks such as: Hitachi SR2201 [52], Cray X/Y-MP [144], DEC GIGA switch and Cenju-3, IBM RP3 [117], and SP2 [15], and Thinking Machine CM-5 [88].

In this research, we focus on direct networks not only because they are the most common types of networks found in recent multicomputers [3, 6, 23, 85, 105, 109, 110, 133, 140], but also because they are emerging as the preferred topology in the on-chip networks for system-on-chip architectures [86, 87, 116, 142]. It should be mentioned that according to the Top500 Supercomputers List [13] that appeared during the Supercomputing Conference (CS06) in Tampa, Florida on November 14th 2006, the top three supercomputers are based on direct interconnection networks. The terms *network*, *interconnection network* and *direct interconnection network* will be used interchangeably in this dissertation.

Figure 1.2 shows an example of a direct network (the torus) and its node structure. Each node consists of a processing element (PE), possibly with some IO, local memory and a router, which handles message transfer across the network. The PE is connected to its router via injection and ejection channels (sometimes called internal channels). The injection channel is used to inject messages to the network and the ejection channel is used to consume messages from the network. A node is linked to its neighbouring nodes using input

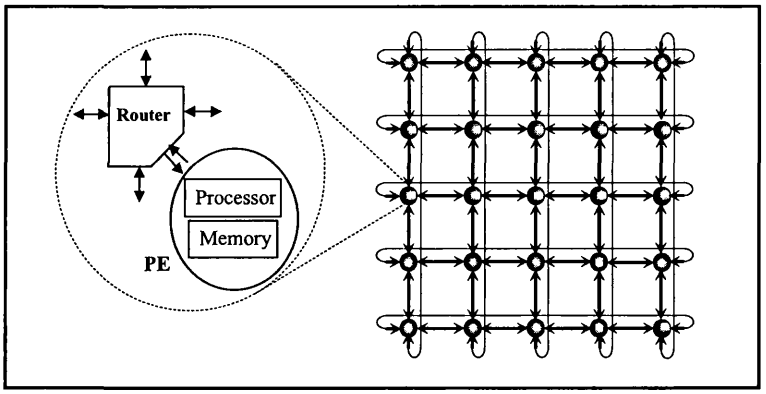


Figure 1.2: An example of direct network (5×5 torus) and its node structure.

and output channels, which can be unidirectional or bidirectional. The input and output channels (sometimes also called external channels) of a router are usually connected by a crossbar switch to allow all possible combinations *of input-output* connections. The topology and the router structure of direct interconnection networks are discussed in more detail in Chapter 2 of this dissertation.

The interconnection network is a critical component in determining the overall system performance. Network performance is usually quantified in terms of *latency* and *throughput*. Latency measures how fast the network can deliver a given message to its destination. It is defined as the time elapsed since a message is given to the local router at its source node for transmission until it is delivered to its destination. The throughput, on the other hand, is the number of messages delivered by the network per time unit [51, 58]. Networks with minimum latency and maximum throughput are often the objective for system designers. Among other factors, the performance of such networks is greatly influenced by its *topology, switching method, and routing algorithm* [42, 49]. The connection pattern of the network nodes defines the network's topology. The switching method dictates which message gets access to a particular network resource over time. The routing algorithm determines which path a message actually takes to advance from its source towards its destination. These factors are now considered in turn.

1.1.1 Network topology

The topology of direct interconnection networks can be efficiently described using a directed graph $G=(V,E)$ where the set of vertices of this graph, V, represents the network nodes and the set of edges of the graph, E, represents the physical channels that connect the nodes. *Network diameter*, *node degree* and *network degree* are often used to characterise a given topology [42, 49]. The network diameter is the maximum distance between any two nodes in the network. The number of channels connecting a node to its neighbours is called the node degree while the network degree is the maximum node degree in the network. A network is regular if all nodes have the same degree and it is symmetric if it is isomorphic to itself with any node labelled as origin [49].

K-ary n-cube networks, where k is referred to as the *radix* and n as the *dimension*, have many desirable topological properties including ease of implementation, low diameter, regularity, symmetry, scalability and recursive structure [42, 49, 56, 131]. They are suited for a variety of applications including matrix computation, image processing, and problems whose task graph can be embedded naturally into this class of topologies. Formally, each node in the k-ary n-cube can be identified by an n-digit radix-k address $(a_1, a_2, \ldots, a_n)$, the i^{th} digit of the address vector, a_i, represents the node position in the i^{th} dimension. Node A with address $(a_1, a_2, \ldots, a_n)$ and node B with address $(b_1, b_2, \ldots, b_n)$ are connected if and only if there exists $i\,(1 \le i \le n)$, such that $a_i = (b_i \pm 1) \bmod k$ and $a_j = b_j$ for $(1 \le j \le n)$ and $i \ne j$. Some topological properties of k-ary n-cubes include [42, 49, 56]

$$\textit{The number of nodes} = k^n$$

$$\textit{The diameter} = \begin{cases} 0 & \text{if } k = 1 \\ n\lfloor k/2 \rfloor & \text{if } k \ge 2 \end{cases}$$

$$\textit{The degree} = \begin{cases} 0 & \text{if } k = 1 \\ n & \text{if } k = 2 \\ 2n & \text{if } k > 2 \end{cases}$$

1.1.2 Switching method

The switching method in an interconnection network manages the allocation of network resources to messages as they travel inside the network. In other words, it determines when and how internal switches connect their inputs to outputs and the time at which message components may be transferred along these paths [49, 116]. The key resources in most interconnection networks are the physical channels and the buffers. While physical channels have the key role of transporting messages between nodes, buffers, on the other hand, are storage space implemented within the nodes and allow whole messages or part of a message to be temporarily held at the node.

Any switching strategy must avoid resource conflicts that can cause a channel to be unnecessarily idle. For example, it should not block a message that can use an idle channel because it is waiting on a buffer held by another message that is blocked on a busy channel [42]. The solution is to add bypass lanes to decouple the resource dependencies, allowing the blocked message to make progress without waiting [36, 42, 49]. *Packet switching*, *virtual cut-through,* and *wormhole switching* are examples of switching methods. Each of these switching methods is now briefly discussed.

In packet switching, a message is divided into fixed length packets each of which carry full routing information and whenever a router has a packet to be sent, it transmits it. The need for storing entire packets in intermediate routers makes buffer requirement high and complicates the router design. This is not a desirable property either in networks for parallel computers, where routers must be kept simple and fast, or in on-chip networks for system-on-chip architectures, where routers should not consume a large fraction of the valuable silicon area [18, 55, 116].

Virtual cut-through (VCT) switching [67] has been introduced as an enhancement to packet switching in order to reduce the transmission time. In this switching method, a message header (i.e. part of the message that contains routing information) is examined upon arrival

at an intermediate router, and if the next requested channel is busy, the message is consumed from the network and entirely stored at that node. Otherwise, the message is transmitted immediately (cut-through) to the next node without buffering. The communication latency, especially under low and moderate traffic loads, is noticeably reduced as blocked messages are removed from the network and the channels are utilised to transmit unblocked messages. However, nodes must provide sufficient buffer spaces for all blocked messages passing through. A large buffer space is required at each node, as multiple messages may become blocked simultaneously. Therefore, VCT switching might be costly to implement [49, 103] due to the high buffer requirement which also has an adverse effect on the router speed [26, 49, 96].

The drawback of VCT switching has encouraged the use of a variant called wormhole switching (WS) [39]. In WS, the message is divided into fixed length flow control units called *flits* (containing typically a few bytes) and the buffers are expected to store only few flits and not the entire message. The header flit (or flits), which contains the routing information, governs the route of the message in the network and subsequent data flits follow it in a pipelined fashion. As a result each data flit is simply forwarded along the same output channel as the preceding data flit. If the header flit faces a busy channel and is blocked, subsequent data flits also have to wait at their locations forming a chain of flits along the message path [42, 49].

The main difference between WS and VCT lies in the way messages are handled in the event of blocking. In VCT, if the header is blocked, the entire message is consumed from the network and temporarily stored at the point of blocking while in WS, the data flits stop advancing and remain spread across the routers that they have already acquired. Hence, as the message progresses flit by flit across the network, each node needs to store only a few flits when blocking occurs. As a result the buffer space requirement is minimised and thus the routers are more compact and less complicated, making WS a natural choice for

interconnection networks in parallel computers [29] and in system-on-chip architectures [18, 55].

One drawback of WS, however, is that the transmission of multiple messages cannot be interleaved over a physical channel [42, 49]. A message must cross the channel in its entirety before that channel can be used by another message. This decreases the channel utilisation, especially under high traffic when blocked messages form chains of flits across the network. The utilisation of the network resources can significantly be improved by introducing *virtual channels*, a flow control mechanism that decouples the allocation of buffers from the allocation of physical channels [36, 42, 49]. While each virtual channel has its own flit buffer and control logic, all virtual channels of a given physical channel share the same bandwidth in a time multiplexed fashion. It has been shown in various research studies [36, 119, 130, 134, 146] that this decoupling between the physical channel and the finite buffer space, by using virtual channels, significantly improves network performance; if a flit belonging to a particular message is blocked in one of the virtual channels, then flits of alternate messages may still use the other free virtual channels and, ultimately, the physical channel.

WS along with virtual channels has been widely employed in many practical parallel computers, such as Cray T3D [68] and Cray T3E [148]. More recently, it has also been deployed in high speed LANs such as Myrinet switches [16, 49] and suggested for network-on-chip architectures [18, 55, 94]. The functionality of WS and virtual channels will be discussed in more detail in Chapter 2

Switching methods are often described using *time-space* diagrams. Figure 1.3 shows such diagrams for (a) packet switching and (b) wormhole switching in a non-blocking scenario. Time is shown in the horizontal axis as cycles (i.e. time to transmit one flit from one router to the next), and space is shown in the vertical axis by listing the channels used to send the message. A flit, represented by a labelled box, is shown in the diagram during the cycle that

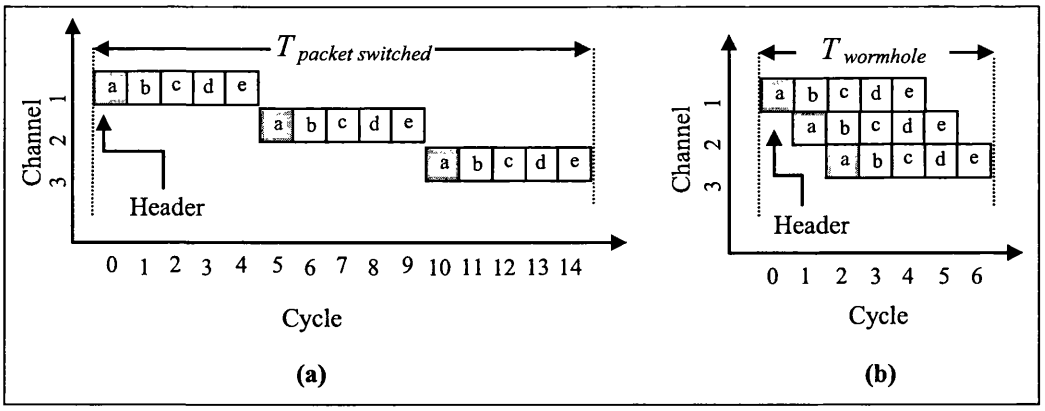


Figure 1.3: Time-space diagrams for a packet consisting of 5 flits for (a) packet switching and (b) wormhole switching. The vertical axis shows space (channels) and the horizontal axis shows time (cycles)

it uses the bandwidth of a specific channel [42]. The Figure illustrates that the choice of a switching method can significantly affects message transmission time. With packet switching, a message is completely transmitted across one channel before transmission across the next channel is started. Contrarily, with WS, messages are pipelined, with each flit being transmitted over the next channel as soon as it arrives.

1.1.3 Routing algorithm

The routing algorithm determines a network path that a message should take to advance from source to destination. A path (or a *route*) is an ordered set of channels $P=\{c_1, c_2, c_3, \ldots, c_k\}$ where the output from channel c_i forms the input to channel c_{i+1}. Messages from the source node form the input to channel c_1, and the destination node receives the messages as the output of channel c_k. *K*-ary *n*-cubes provide many possible paths for routing a message between any two nodes. This introduces the issue of selecting a best route among several alternatives. Routing algorithms can be divided into two classes according to their ability to modify routing paths; *deterministic* and *adaptive* [49, 103, 137].

In deterministic routing messages with the same source and destination addresses always take the same network path; intermediate nodes are unable to redirect messages to alternative paths. In adaptive routing, on the other hand, intermediate nodes can take the actual network

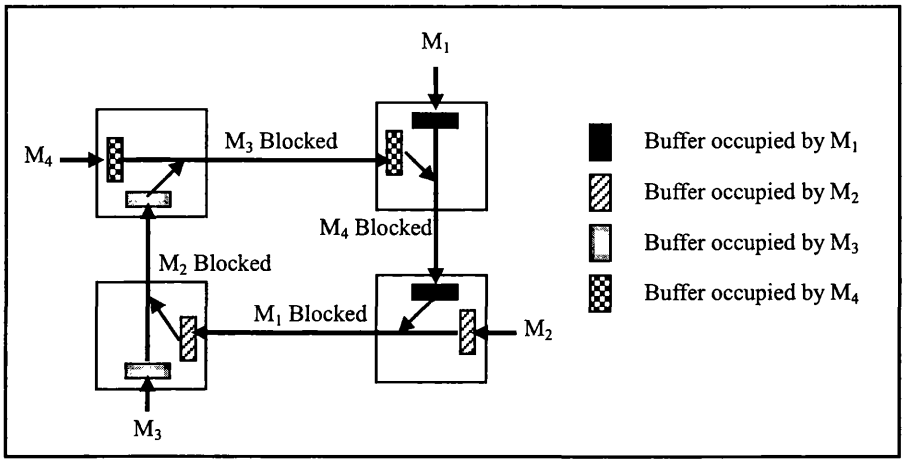


Figure 1.4: A typical deadlock situation caused by four messages.

conditions, such as the presence of congestions or failures, into account and determine accordingly to which node a message should be sent [137]. Examples of these two types of routing algorithms are discussed in more detail in Chapter 2.

A critical requirement for any routing algorithm is to ensure deadlock freedom. Deadlock is a situation where a set of messages are mutually blocked waiting for network resources to be released by each other, and consequently no message can advance any further [33, 49, 137]. Deadlock can typically occur in a network where cyclic dependencies could be formed [46]. Figure 1.4 shows such a case, where each of the four messages M_1, M_2, M_3 and M_4 holds a buffer required by another message and waits for the buffer in the next node in a circular fashion. Many routing algorithms have been proposed to prevent deadlock in wormhole-switched networks by incorporating routing restrictions and virtual channel allocations to avoid cyclic dependencies [20, 42, 46, 49].

1.1.4 Implementation constraints

When wormhole switching is used, messages are divided into fixed size flow control units known as flits. The flit is often composed of one or more *phits*, each of which is equal to the channel width C_w. Each phit takes one cycle to be transmitted from one router to the next. The topology and implementation technology forces some limitations on the channel width

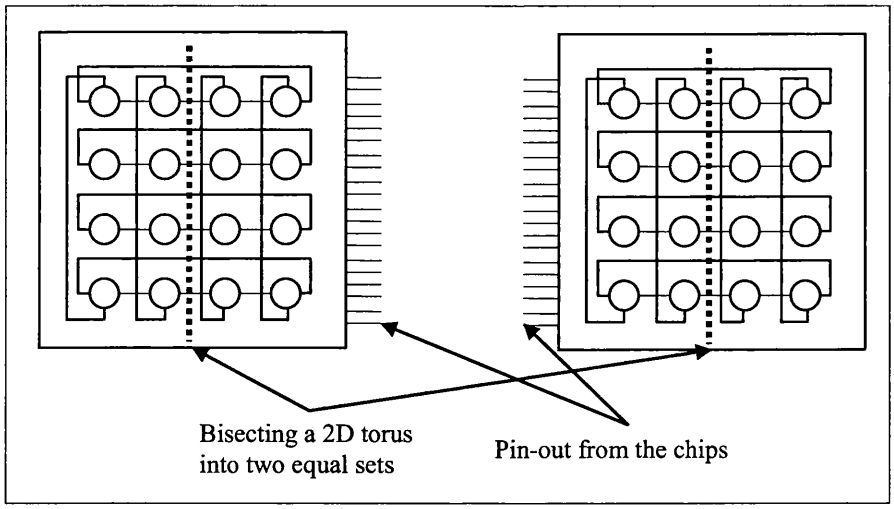


Figure 1.5: An illustration of the bisection width and the pin-out constraints when interconnecting two chips each of which implements a 2D torus.

which in turn affects the overall network performance [42, 49]. One such constraint is the bisection width which is the minimum number of wires that must be cut when the network is divided into two equal sets of nodes [42, 49]. The bisection width constraint is usually used to quantify the available wiring area of a given implementation. For example, in a 2D torus with $N=k^2$ nodes, if the network is divided into two equal sets then there will be a $2kC_w$ wires crossing the bisection where C_w is the channel width. A second implementation constraint on the channel width is the number of I/O pins available per router through which data must travel [5, 42, 49]. Figure 1.5 illustrates the bisection width and the pin-out constraints when implementing a 2D torus.

Constant *bisection* width [35, 42] and constant *pin-out* [2, 5] channel bandwidth constraints have been used by several researchers to analyse the performance merits of competing network topologies. Without such constraints, one cannot evaluate a topology or perform a fair comparison between different topologies [42].

It has been shown in [143] that when the network is implemented on a single VLSI chip, the wiring density of the network determines the overall system cost and performance. In [35], Dally has chosen constant bisection width as a bandwidth constraint when evaluating the

performance of different k-ary n-cube configurations. On the other hand, Abraham and Padmanabhan [2] and Agarwal [5] have argued that the wiring density constraint is applicable *only* when the entire network is implemented in a single VLSI chip, and thus it is not applicable to the situation where the network has to be partitioned over many chips. They have identified that the bandwidth constraint is imposed by the chip's I/O pins. The studies in [2, 5] analysed the performance of k-ary n-cubes under the pin-out constraint and reached different conclusions from those reported in [35]. Nevertheless, these studies have demonstrated that the network performance is highly dependent on the implementation constraint [92]. Having said the above, other design factors (switching method, routing algorithm, virtual channels, router complexity ...) also greatly influence network performance. All of these factors and implementation constraints have been considered when conducting the performance comparisons reported in this research.

1.2. Performance evaluation

Today's computers, and communication systems, are more complex, more rapidly evolving and more essential to the conduct of science, engineering and business than those of even a few years ago. The performance of the interconnection networks employed in these systems has become a key factor in the overall application and system performance. This will increasingly be the case as applications become more complex with high computation and communication requirements [43]. Hence there is always an increasing demand for tools that assist in understanding the behaviour and evaluating the performance of interconnection networks.

Analytical models, simulations and measurements all play roles in performance evaluation [51, 79, 90]. Analytical and simulation models are used to provide approximate performance measures with a minimum amount of effort and cost [42]. For instance, analytical models allow an entire family of networks with varying parameters to be evaluated at once by

deriving a set of equations that predicts the performance and gives insight into how different factors affect the performance of the entire family. Once a network is constructed (or at least prototyped) it can be instrumented and measurements can be taken for different configurations. Of course at this point, If it is not an experimental setup (i.e. the system is up and running), it is difficult, time consuming, and expensive to make changes if performance problems are encountered [42].

Simulations can be thought of as computer programs that mimic the operations of a system [95]. They are an invaluable tool in situations where developing analytical models to capture the performance measures is too complex due to the large number of factors affecting the performance. However, simulation is a double-edged sword. While it can mimic complex network designs, simulators can themselves be extremely complex, costly and time consuming [42]. Difficulties with simulations include the laborious effort often required to develop the program code in addition to the large amount of computation time and resources that are required to run the simulation [51, 79, 90]. Each simulation run can take considerable time and evaluates only a single network configuration, traffic pattern and load point. Another problem with simulation experiments is the difficulty in reproducing the published results for further comparisons and validations as researchers tend not to provide sufficient information about their simulation environments [9, 83].

An analytical model is an abstraction of a system: an attempt to distil from the mass of details that are in the system itself, exactly those aspects that are essential to the system's behaviour and performance. Once a model has been defined through this abstraction process, it can be parameterised to reflect any of the alternative cases under study [42]. It is then used to capture the crucial features of the system, display underlying trends, yield insights into the system performance and often provides a great reduction in the time and cost required for such investigations [51, 79, 90]. Therefore, analytical modelling is a viable and cost-effective alternative to simulation for investigating system performance. In particular, it can

be used to obtain performance results for large-scale systems, which may be infeasible to analyse through simulations due to the excessive computational demands. Moreover, analytical models can operate alongside simulations where, for example, the latter is used to verify results from the former. Analytical modelling based on queuing and probability theory has been a practical approach to the performance modelling and evaluation of computer and communication systems [76, 108]. Here a system is represented as a network of queues, whose performance measures, such as delay and throughput, are evaluated analytically [76, 108].

Beside the complexity involved in developing the analytical model, a drawback of analytical performance modelling, and also simulations, is that it usually involves making a number of approximations and assumptions that may affect the accuracy of the results. Hence, the validity of these models must be demonstrated before they can be used to analyse the performance. Validation is the process of checking the accuracy of a performance model against known good data. For example, if we have data collected from an actual network, we can validate a simulation model or an analytical model using this data by generating the same numbers via simulations and analytical models. If the numbers match, we have high confidence that the model accurately predicts the performance of the studied network and it is likely that the models will also accurately handle similar networks under similar operating conditions [42]. If we have high confidence in a simulation model, we can validate the analytical model against simulations [42, 51] (this approach is used to validate the models developed in this research).

1.3. Related work

Within the area of interconnection networks, this study is related to the work on analytical performance modelling and evaluation of wormhole-switched interconnection networks. More specifically, it focuses on modelling the effects on network performance of multiple

virtual channels and finite buffers at intermediate routers. This section will sheds some light on existing analytical models that have been developed for wormhole-switched networks. Work related to the performance comparisons of k-ary n-cubes under different operating condition will be discussed in detail in Chapter 6.

In [35], Dally has devised the first analytical model for calculating the average message latency in a unidirectional k-ary n-cube. He has assumed a wormhole-switched network without virtual channels and deterministic routing. The accuracy of the model degrades significantly when applied to networks with high traffic as it does not take into account the blocking behaviour in the network.

Draper and Gosh [45] presented an accurate model for unidirectional k-ary n-cubes using wormhole switching and deterministic routing. However, this study has not considered the effects of employing multiple virtual channels. It is worth mentioning that the authors have also proposed an approximation to estimate the variance of the service time distribution. This approximation significantly eases the derivation of analytical models that need to use the variance of the service time. Similarly, Adve and Vernon [4] have proposed a model for k-ary n-cubes employing virtual cut-through and wormhole switching techniques without the use of virtual channels. Their study considers deterministic routing and can model traffic with any arbitrary source-destination distribution.

Ciciani et al [28] have introduced a model for wormhole switched 3D torus (i.e. k-ary 3-cube) with deterministic routing. Like previous studies, this study has not considered the effects of multiple virtual channels. Another model has been proposed by Greenberg and Guan [60] for the 2D mesh. This model employs wormhole switching and deterministic routing but loses accuracy when approaching heavy traffic loads.

In [73], Kim and Das have introduced an analytical model for hypercubes using wormhole and virtual cut-through switching without virtual channels. Their model considers

deterministic and random routing algorithms. The model takes account of the blocking effect and displays good accuracy. Another model for hypercubes has been proposed by Hady and Menezes [62] using wormhole switching and deterministic routing. Results obtained from simulation experiments show close agreement to those predicted by the model.

Note that all the above mentioned models have considered only deterministic routing and employ no virtual channels. The first attempt to model adaptive routing has been done by Boura et al [22]. They have proposed a model for wormhole-switched hypercubes employing virtual channels and adaptive routing. The model is fairly accurate for all traffic loads. Loucif et al in [93] and Ould-Khaoua in [111] have introduced analytical modes for unidirectional k-ary n-cubes. Their models have considered wormhole switching with virtual channels, and adaptive routing.

Sarbazi-Azad et al [126] have presented an analytical model for k-ary n-cube considering wormhole switching with multiple virtual channels and adaptive routing algorithms. In addition to modelling uniform traffic, the authors have also considered important non-uniform traffic patterns such as hot-spot traffic [127], matrix-transpose traffic [129] and digit-reversal traffic [128]. More recently, Min and Ould-Khaoua [102] have developed an analytical model for wormhole-switched interconnection networks that employ adaptive routing algorithms and self-similar traffic.

All of the above models have used deadlock-free routing algorithms (either deterministic or adaptive). In [69, 70, 71], Khonsari et al have devised new analytical models for some deadlock-recovery routing algorithms. Deadlock recovery routing allows the deadlock to occur and tries to recover from the deadlock situation. Models for compressionless routing [74], software-based routing [97], and Disha routing [12], have been presented and results show close agreement with those obtained from simulation experiments.

It is worth mentioning that almost all of the above proposed models have assumed *single* flit buffers per network channel. Hu and Kleinrock [66] have proposed a model, based on M/G/1/K queuing system, for wormhole-switched networks employing finite buffers (i.e. releasing the single buffer constraint). The authors have assumed deterministic routing and no virtual channels. Their model has been developed for irregular topologies used in local area networks and hence assumes large buffer requirements due to relatively high propagation delays in such networks [16, 75]. The model accuracy degrades for small buffer sizes (i.e. less than the message size) that are usually found in multicomputer networks. Beside its high complexity, the model by Hu and Kleinrock [66] cannot be directly applied to the interconnection networks used in multicomputers due to the substantial topological differences and buffer requirements in these two types of interconnects.

Kouvatsos et al [81, 82] have proposed analytical models for hypercube and torus networks with finite buffers. In these models, the authors have used the same approach suggested in [66] but have relaxed the Poisson arrival assumption. Like the model in [66], these models assume deterministic routing and do not consider the effects of multiple virtual channels.

Nevertheless, it should be mentioned that there have been a few studies that attempted to model finite buffers in Multistage Interconnection Networks (see for example [64, 104, 149]). These models cannot directly be extended to k-ary n-cube networks (e.g., torus or hypercube) due to the inherently different topological properties and switching methods used in these two types of interconnects.

In [36], Dally has studied the performance of networks with multiple virtual channels using both analytical and simulation methods. He has presented a method based on a Markov chain to calculate the probability of busy virtual channels per physical channel. His method has been used in many performance studies (see for example: [22, 71, 93, 102, 111, 126, 127]) that have considered the effects of virtual channel multiplexing on network performance.

Table 1.1: A summary of the main features of some recent parallel computers listed in a chronological order

System	Year of introduction	Max. No. of nodes	Topology	No. of VC	Buffer	Routing
Cray XT3 [147]	2004	32K	3D torus	4	Cray SeaStar ♦	Adaptive
IBM Blue Gene/L [3]	2004	65536	3D torus	5	1KB/VC	Adaptive and Deterministic
Cray X1 [110, 140]	2003	4096	2D torus	4	Cray SeaStar ♦	Adaptive
Alpha 21364 [105]	2001	128	2D torus	3	20 KB	Adaptive
SGI Origin [42, 53]	1997	512	Hypercube	4	SGI Spider ♦	Deterministic
Cray T3E [133]	1995	2048	3D torus	5	22 and 12 flits	Adaptive
Cray T3D [68]	1993	2048	3D torus	4	1 flit	Deterministic
J-machine [42, 109]	1991	1024	3D mesh	1	1 flit	deterministic

1.4. Motivations

Most recent multicomputers and high speed switches such as the IBM Blue Gene/L [3], Alpha21364 router [105], J-machine [109], Cray XT3 [147], Cray T3D [68], Cray T3E [133], Cray X1 [110, 140] and SGI Origin [53] as well as many recent proposed on-chip networks [86, 87, 116, 142] have used k-ary n-cubes as their underlying topology. Wormhole switching, along with virtual channels [42, 49], is advantageous over other switching methods, like packet switching, as it is less sensitive to the message distance [49]. Also, due to its low buffer requirement, it promotes compact and faster router design, an especially significant factor in both networks for multicomputers, where routers must be kept simple and fast, and on-chip networks for system-on-chip architectures, where routers should not consume large fraction of the silicon area [18, 55, 116]. Table 1.1 summarises the main features of some recent practical and experimental parallel computers.

In order to harness fully the computational power offered by parallel computers, it is essential that the performance and behaviour of their underlying interconnection network be deeply and thoroughly investigated. As mentioned earlier, one approach to conduct such a performance investigation is to develop accurate, efficient, and realistic analytical models.

♦ Cray SeaStar and SGI Spider are communication and routing chips used in these parallel computers. There is no buffer space dedicated to each channel but channels use a shared memory (384KB) to buffer messages according to different allocation algorithms [42, 53, 110, 140, 147].

Analytical models are especially of great importance in analysis of large-scale systems, which may be infeasible to simulate due to excessive computational demands. Many analytical models (for instance [22, 28, 35, 45, 60, 62, 66, 70, 71, 73, 81, 82, 93, 102, 111, 127, 129]) have been suggested in the literature over the past years in order to assess the performance of interconnection networks under different assumptions and configurations. Table 1.2 lists the main characteristics of the analytical models related to this research.

It has been shown in existing research studies [36, 119, 130, 134, 146] that adding virtual channels noticeably improves the performance of wormhole-switched interconnection networks. This has encouraged the adoption of virtual channels in practical systems, for example, the IBM Blue Gene/L [3] and the Cray T3E [10] uses five virtual channels per physical channel and Cray T3D [68], XT3 [147] and X1 [110] uses four virtual channels per physical channel. Hence, it is of significant importance for any analytical model to accurately capture the performance impact of using multiple virtual channels.

Almost all of the analytical models (for example [22, 69, 82, 93, 102, 111, 126]) that have been developed so far have used the method proposed by Dally in [36] to calculate the probability of the number of busy virtual channels per physical channel (see column 5 of Table 1.1). This method is based on a Markov chain and although it is useful in some cases, for example under light traffic, it loses its accuracy as network traffic increases. Due to the blocking nature of wormhole switching, as the traffic increases, blocked messages at subsequent channels may interrupt the transmission of other messages, resulting in the Markovian assumption being no longer valid. This is reflected in a degradation of accuracy that can be noticed in the reported results of the models that have used this method to capture the effects of virtual channels under moderate and high traffic conditions (see for example [22, 69, 82, 93, 102, 111, 126]). This observation combined with the importance of dealing with virtual channels in modelling the performance of interconnection networks has motivated this study to reinvestigate the occupancy probabilities of busy virtual channels.

Table 1.2: Previous analytical models that have been suggested to assess the performance of wormhole-switched interconnection networks

Model	Year	Routing Algorithm	Traffic	VC model	Buffers model	Topology
Dally	1990	Deterministic	Uniform	No	Single flit	*k*-ary *n*-cube
Draper and Gosh	1994	Deterministic	Uniform	No	Single flit	*k*-ary *n*-cube
Adve and Vernon	1994	Deterministic	Any source-destination distribution	No	Single flit + Infinite	*k*-ary *n*-cube
Kim and Das	1994	Deterministic + Random	Uniform	No	Single flit	Hypercube
Boura and Das	1994	Adaptive	Uniform	Dally	Single flit	Hypercube
Hady and Menezes	1995	Deterministic	Uniform	No	Single flit	Hypercube
Ciciani et al	1997	Deterministic	Uniform	No	Single flit	Asymmetric 3D Torus
Greenberg and Guan	1997	Deterministic	Uniform	No	Single flit	2D Mesh
Hu and Kleinrock	1997	Deterministic	Uniform	No	Finite	Irregular
Ould-Khaoua	1999	Adaptive	Uniform	Dally	Single flit	*k*-ary *n*-cube
Sarbazi-Azad et al	2001	Adaptive	Uniform and Non – uniform	Dally	Single flit	*k*-ary *n*-cube
Kouvatsos et al	2003	Deterministic	Uniform	No	Finite	*k*-ary *n*-cube
Min et al	2004	Adaptive	Self-similar	Dally	Single flit	*k*-ary *n*-cube
Khonsari et al	2004	Deadlock avoidance	Uniform	Dally	Single flit	*k*-ary *n*-cube
Alzeidi (our aim)	2007	Adaptive and Deterministic	Uniform	General	Finite	*k*-ary *n*-cube

Although *pure* wormhole switching requires single-flit buffers, most practical systems have deployed deep finite buffers that hold more than one flit, to reduce performance degradation in the presence of high message blocking [3, 42, 49, 105, 133, 147]. Nevertheless, Table 1.1 (column 6) reveals that almost all of the proposed analytical models have assumed single flit buffers at intermediate nodes. This is a simplifying assumption that has been invoked to ease the derivation of the analytical models and to make them tractable and simple to calculate.

It has been pointed out in the previous section that the studies in [66, 81, 82] have proposed analytical models for networks where the single-flit buffers constraint is relaxed. These studies have been carried out for deterministic routing and have not considered the effects of multiple virtual channels. Although deterministic routing algorithms are simpler to implement [42, 49], they can not exploit network channels efficiently since messages cannot use alternative paths to avoid congested channels and thus reduce message communication latency. 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 including Cray T3E [133], IBM Blue Gene/L [3] and Alpha 21364 [105]. Moreover, for the sake of comparison with deterministic routing, it is necessary to develop analytical models to capture the effects of finite buffers on the performance of adaptive routing. Motivated by the above observations, this research proposes original analytical models for k-ary n-cubes that employ adaptive (and deterministic) routing and combine wormhole switching, multiple virtual channels and finite buffers (i.e. >1 flit).

Owning to the new analytical models developed below, that can capture the effects of finite buffers, this dissertation is the first to compare analytically the relative performance merits of k-ary n-cubes under different implementation constraints, operating conditions and routing algorithms in the presence of finite size buffers. In addition to its new insights and the conclusions reached, this study also acts as a proof for the usefulness of the newly developed models in studying the performance characteristics of interconnection networks.

1.5. Thesis statement

Most existing analytical models of wormhole-switched networks have partially relied on some simplifying and restrictive assumptions, notably the single-flit buffers constraint and the exponential service time for message transmission. The aims of this research have been driven by the motivations discussed in the above section and can be summarised into the following thesis statement. In this research, I assert that:

T1: The inherent blocking mechanism of wormhole switching creates blocking dependencies along the message path. Therefore, assuming an exponential service time distribution for message transmissions may generate inaccurate performance predictions, especially when the network operates in moderate and high traffic conditions. If this assumption is relaxed by adopting a general service time distribution using an M/G/1 queuing system, modelling the virtual channels occupancy probabilities as the number of customers in such a queue enables more accurate predictions of network performance.

T2: The single-flit buffers constraint, used in many network modelling studies, makes the existing models inadequate for assessing the network performance in the presence of deeper buffers. This constraint can be released by exploiting results from probability and queuing theory. As a result, new analytical models can be developed to capture the effects on the performance of deploying finite buffers.

T3: The modelling approaches, mentioned in T1 and T2, are versatile and cost effective tools that can be used to study, more realistically, the performance behaviour of k-ary n-cubes. To this end, they can be used to explore the relative performance merits of different topologies in the presence of finite buffers and under different implementation constraints. The models also allow for an extensive analytical investigation of different buffer arrangements into virtual channels and the performance of adaptive and deterministic routing algorithms under various operating conditions and system sizes.

1.6. Summary of the contributions

The first part of this dissertation proposes a new method to model the effects of virtual channels in wormhole switched networks. The method is based on an M/G/1 queuing system as opposed to the Markov chain proposed in [36]. More specifically this work models the probability of busy virtual channels as the probability of the number of customers in the queue. These probabilities are computed by inverting the *z-transform* of the number of customers in the queue. The number of arrivals at this queue is the average number of messages received by each physical channel, while the service time is equal to the channel holding time, the period from the moment a message is granted access to the physical channel till the last flit of that message is successfully transmitted. It is assumed that this service time follows a general distribution and is expressed in terms of its *Laplace transform*. It is this general service time that allows easy customisation of the model to fit different traffic conditions by assuming different service time distributions.

It should be mentioned that whenever the service time is known then different methods can be used to invert the *z-transform* and hence calculate the probability of busy virtual channels. However, when only the first two moments of the service time distribution are known, a two *moment-matching* to approximate the service time distribution is presented. Moreover, to demonstrate the customisability and the generality of the new method, it is shown that Dally's method can be deduced from it as a special case when the service time is exponentially distributed. The validity and the accuracy of the new general method are demonstrated by deploying it in an existing analytical latency model originally based on Dally's method [36] and comparing the results to an event-driven simulator.

The second part of this dissertation proposes the first analytical model to capture the effects of finite buffers on the performance of deterministic as well as adaptive routing algorithms and on scenarios where multiple virtual channels are used per physical channel. Beside its simplicity and superior accuracy, this model is the first that considers adaptive routing and

multiple virtual channels and can handle small buffer sizes (i.e. bigger than 1 flit and smaller than the message length) found in wormhole-switched networks.

Finite buffering complicates the model development into two ways. First, the previously used assumption that a message spans from the source to the destination is no longer valid. Messages may now occupy only a fraction of the channels along their paths due to the available buffer space. The proposed model tackles this challenge by defining and calculating the effective number of channels as a function of the network topology, message size and buffer size. Secondly, when routers are equipped with finite buffers, in addition to blocking due to contention over the virtual channels, a message may also be blocked inside a buffer due to there being insufficient space to accommodate it entirely. The new model determines these two types of blocking separately, which allows the handling of arbitrary message lengths and buffer sizes. The analytical model is validated by means of simulations and the results reveal that it exhibits a good degree of accuracy for various network configurations.

Building on the analytical models developed, the last part of this research work is devoted to studying and analysing the performance of interconnection networks with finite buffers and multiple virtual channels. The models are first used to compare various k-ary n-cube topologies under different implementation constraints when finite buffers are employed. Then the first comprehensive analytical study of the effects of the arrangement of virtual channels is conducted, i.e. given a finite amount of buffer per physical channel, the study attempts to determine the optimal arrangement of the virtual channels either as deep or parallel buffers. Moreover, the new models are used to assess analytically the performance of adaptive and deterministic routing when finite buffers and multiple virtual channels are deployed in the network. Extensive experiments have been conducted for a varying number of network parameters to analyse comprehensively the network performance. It is important to mention that, although some of these issues have already been tackled through simulation

in the past, the present studies are comprehensive and cover a wider spectrum of the network parameters using the new analytical models developed in this research.

1.7. Outline of the dissertation

The rest of this dissertation is organised as follows. Chapter 2 provides some preliminary background information that is necessary for a clear understanding of the analytical models and performance studies presented in the subsequent chapters. To be more specific, the chapter elaborates more on the torus network and its router structure. It then explains the functionality of wormhole switching and the role of virtual channels. Finally, two types of routing algorithms are discussed, namely the dimension-order routing, as an example of deterministic routing algorithms, and the Duato's adaptive routing, as an example of adaptive routing algorithms.

Chapter 3 is devoted to the development and derivation of a new general method to capture the effects of virtual channels on the performance of wormhole-switched networks.

Chapter 4 presents an original analytical model that captures the effect of finite buffers in wormhole-switched k-ary n-cubes with multiple virtual channels. The model is derived for the deterministic dimension order routing algorithm.

Chapter 5 presents a second new model for finite buffers based on Duato's adaptive routing algorithm. All the models presented in Chapters 3, 4, and 5 are validated through simulation experiments.

Chapter 6 analyses the performance of interconnection networks under different working conditions when finite buffers and virtual channels are used. This chapter address three issues. First, it compares different k-ary n-cube topologies under different implementation constraints when finite size buffers are employed. Second, it presents a comprehensive performance study of the optimal arrangement of finite buffers in deep or parallel virtual

channels. Finally, it analyses the effects of multiple virtual channels and finite size buffers on the performance of adaptive and deterministic routing.

Chapter 7 concludes this dissertation by summarising the main results made in this research and then outlines some potential directions for future work.

Chapter 2

Preliminaries

This chapter will provide in more detail some preliminary background information that is necessary for a clear understanding of the development of the new analytical models which form the core of this dissertation. First, the structure of the torus network and its router are outlined. Then, the operations of wormhole switching and virtual channels are presented. Finally, *dimension-order* routing (as an example of deterministic routing) and *Duato*'s adaptive routing (as an example of adaptive routing) algorithms are discussed.

2.1 The torus network

The torus network, as a member of k-ary n-cube networks, packs $N=k^n$ nodes in a regular n-dimensional grid with k nodes in each dimension. Being a direct network, each node in the torus serves simultaneously as an input terminal (i.e., that can generate messages), an output terminal (i.e., that can consume messages) and a switching node (i.e. that can forward messages to other nodes) of the network.

Nodes in the k-ary n-cube can be identified by an n-digit radix-k address $(a_1, a_2, \ldots, a_n)$, where the i^{th} digit of the address vector, a_i, represents the node position in the i^{th} dimension. Each node is connected to all other nodes that differ by $\pm 1 (mod\ k)$ in exactly one address digit [42, 49]. Formally, node A with address $(a_1, a_2, \ldots, a_n)$ and node B with address $(b_1, b_2, \ldots, b_n)$ are connected if and only if there exists $i (1 \le i \le n)$, such that $a_i = (b_i \pm 1) mod\ k$ and $a_j = b_j$ for $(1 \le j \le n)$ and $i \ne j$. The "modulo" operator is used to

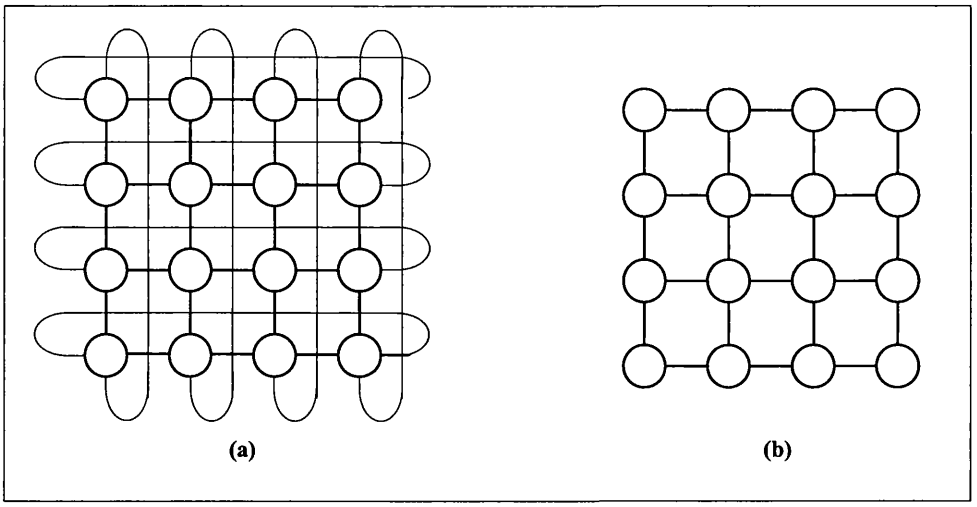


Figure 2.1: Torus and mesh networks: (a) a torus network (4-ary 2-cube) includes the wrap-around connections, but (b) a mesh network (4-ary 2-mesh) omits these connections

account for the *wrap-around* connections between the nodes at both edges of each dimension. A k-ary n-mesh [42, 49], however, is simply a torus without the wrap-around connections. Figure 2.1 shows an example of a 4-ary 2-cube (torus) and a 4-ary 2-mesh.

The torus may be *unidirectional* with channels in only one direction (from a_i to a_{i+1}) in each dimension, or *bidirectional* with channels in both directions between connected nodes [42, 49, 56]. The average and the maximum number of hops that a message may traverse along one dimension and across the network, in the bidirectional torus, are lower than their unidirectional counterparts. Moreover, messages in the bidirectional torus can take advantage of the greater *path diversity* due to the fact that messages can traverse both negative and positive directions of the network dimensions. Path diversity describes the number of available paths between any pair of nodes. It is a desirable property as it adds to the robustness of the network by balancing load over channels and allowing the network to tolerate faulty channels and nodes. This dissertation focuses on the bidirectional torus and the terms "torus" and "k-ary n-cube" are used interchangeably to mean the "bidirectional torus" unless otherwise mentioned.

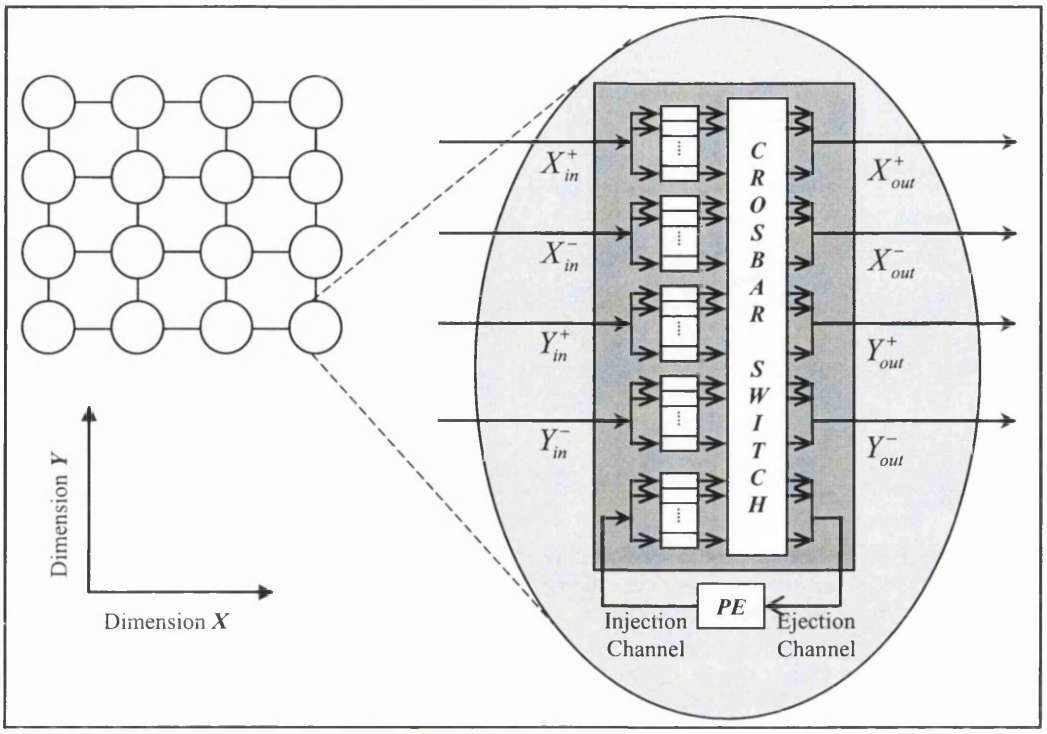


Figure 2.2: A 4 × 4 torus network (or 4-ary 2-cube) and its node structure. The wrap-around connections are omitted for clarity of presentation.

As illustrated in Figure 2.2, each node in the torus consists of a processing element (PE) and a router which are intra-connected through *injection* and *ejection* channels. The injection channels are used by the PE to inject messages to the network to be delivered to their destinations while, on the other hand, the ejection channels are used to consume messages, which are addressed to the destination PE, from the network. The router links a node to its neighbouring nodes via $2n$ input and $2n$ output channels, corresponding to two neighbours (one in each direction) in each of the n dimensions. The input and output channels are connected by a crossbar switch, which can connect simultaneously multiple inputs to multiple outputs in the absence of channel contention. Traffic generated by the local PE is placed in a local queue and waits there for transmission [42, 49]. Each physical channel is equipped with a finite amount of buffer space which can be arranged into multiple ($V \geq 1$) virtual channels[1], each of which has its own flits buffer of size $F>1$ [36].

[1] Virtual channels are explained in more detail in Section 2.3

2.2 Wormhole switching

The switching method determines when and how switches inside routers are set to connect input to output channels, and the time at which messages are transmitted along output channels to continue their network journey towards their destinations [42, 49]. Dally and Seitz have proposed wormhole switching, which was first implemented on the Torus Routing Chip [39]. In wormhole switching a message is fragmented into elementary units, called *flits* (each of a few bytes), for transmission and flow control [36, 49] and the buffers are typically large enough to store a few flits of a message. The *header* (one or more flits that contains routing information) governs the path through the network, and the remaining data flits follow it in a pipelined fashion.

When the header of a message arrives at a node, a routing decision has to be made and an output channel has to be allocated to the message. Rather than waiting for all data flits of the message to arrive fully, the router starts to forward the header and the following data flits as soon as they arrive. If the header is blocked, the data flits stop advancing and remain spread across the channels that they have already acquired. This is because buffers are typically too small to accommodate completely an entire message. The allocated output channel cannot be used by other messages until the last data flit of the previous message has been transmitted fully, and hence flits from different messages cannot be interleaved within the same output channel [42, 49]. Figure 2.3 illustrates a message being transmitted along a path that requires routers $R1$, $R2$, $R3$, and $R4$. At $R4$, message A requires an output channel that is already occupied by message B whose path also goes through $R4$ and $R5$. Hence, due to small buffers at each node (less than the message size), message A blocks in situ and remain spread over multiple routers, which may lead to blocking other messages.

However, since wormhole switching uses pipelining, it can perform well even in high diameter networks [42, 49, 60, 80, 103] as latency is less sensitive to the message distance than in packet switching methods. Moreover, as the message progresses flit by flit across the

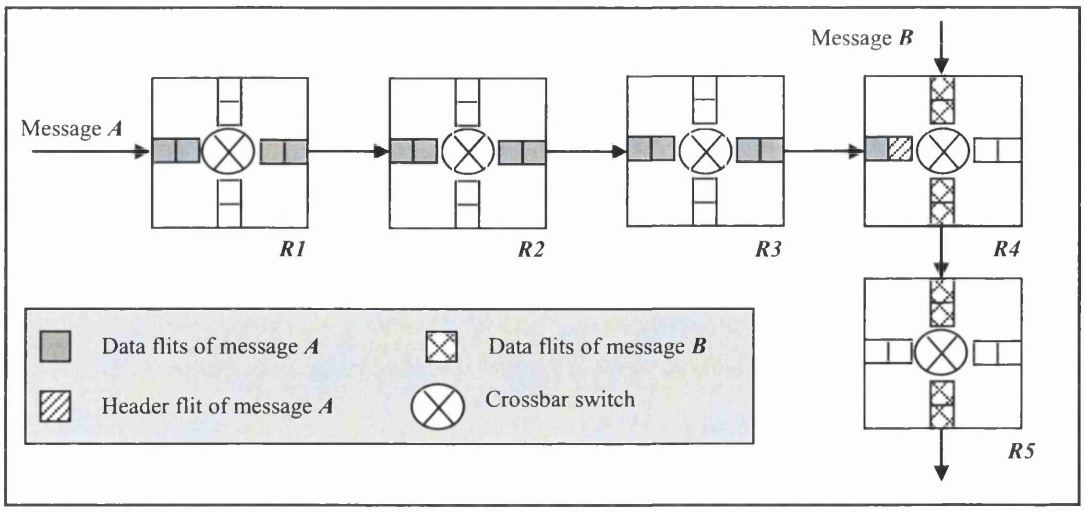


Figure 2.3: Two messages *A* and *B* transmitted using wormhole switching. Message *A* is blocked at router *R*4 because it requires an output channel that is being used by message *B*.

network, each node needs to store only a few flits in the event of messages blocking, minimizing the buffer requirements at each router and enabling the construction of small, compact and fast routers [42, 49, 103]. Wormhole switching has been employed widely in practical networks such as those used, for example, in Cray T3D [68], Cray T3E [133] and Myrinet switches [16]. Recent studies have also suggested it for on-chip networks [55, 94].

A major drawback of wormhole switching arises from the fact that blocked messages continue to occupy channels and buffers, which can therefore freeze up network resources, blocking other messages, and as a consequence increasing the likelihood of deadlock situations in the network. Moreover, it has been shown in previous research studies that the chained blocking property of wormhole switching can limit the network bandwidth to a fraction (20%-50%) of the total available physical network bandwidth [5, 35]. One solution to alleviate this problem is to use *virtual channels flow control*. Flow control concerns techniques for managing the available network resources, namely buffers and physical channels, and it is very dependent on the switching method in use. Virtual channels flow control has been commonly used in the implementations of wormhole switching [42].

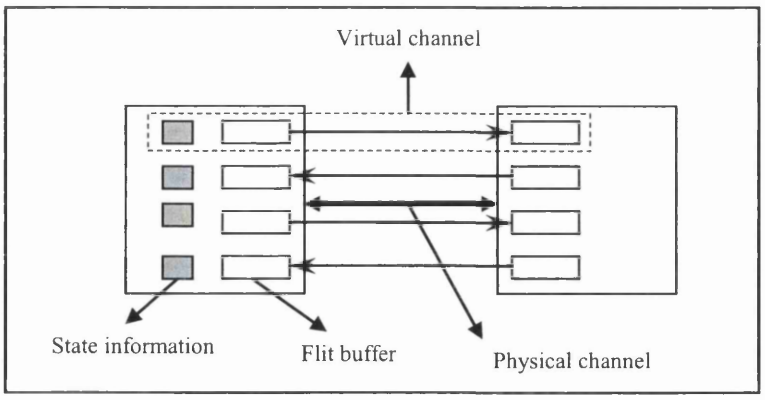


Figure 2.4: A virtual channel is realised by a flit buffer and some state information to help multiplexing flits from all virtual channels into the same physical channel

2.3 Virtual channels

Virtual channels were originally suggested to develop deadlock-free routing algorithms in wormhole switched networks [40]. With this approach, instead of having a unique buffer queue associated with each physical channel, the available buffer space is divided into several smaller size parallel queues, and some state information is introduced to enable multiplexing header and data flits from these queues (or virtual channels) into the same physical channel. Hence, each virtual channel is realized by an independently managed pair of flit buffers and associated control logic as illustrated in Figure 2.4. Virtual channels decouple the allocation of buffers from the allocation of physical channels, allowing active messages to bypass blocked messages [36, 49].

Virtual channels overcome the chain blocking problem of wormhole switching by allowing other messages to use the channel bandwidth that would otherwise be left idle when a message blocks, as illustrated in Figure 2.5. The figure shows 4 routers of a 2D torus where message $M1$ has entered router $R2$ from the top, acquired channel $C2$, and then gets blocked. A second message $M2$ has entered router $R1$, acquired channel $C1$, entered router $R2$ from the left and needs to be routed by $R3$ via channel $C3$ to continue its journey towards its destination.

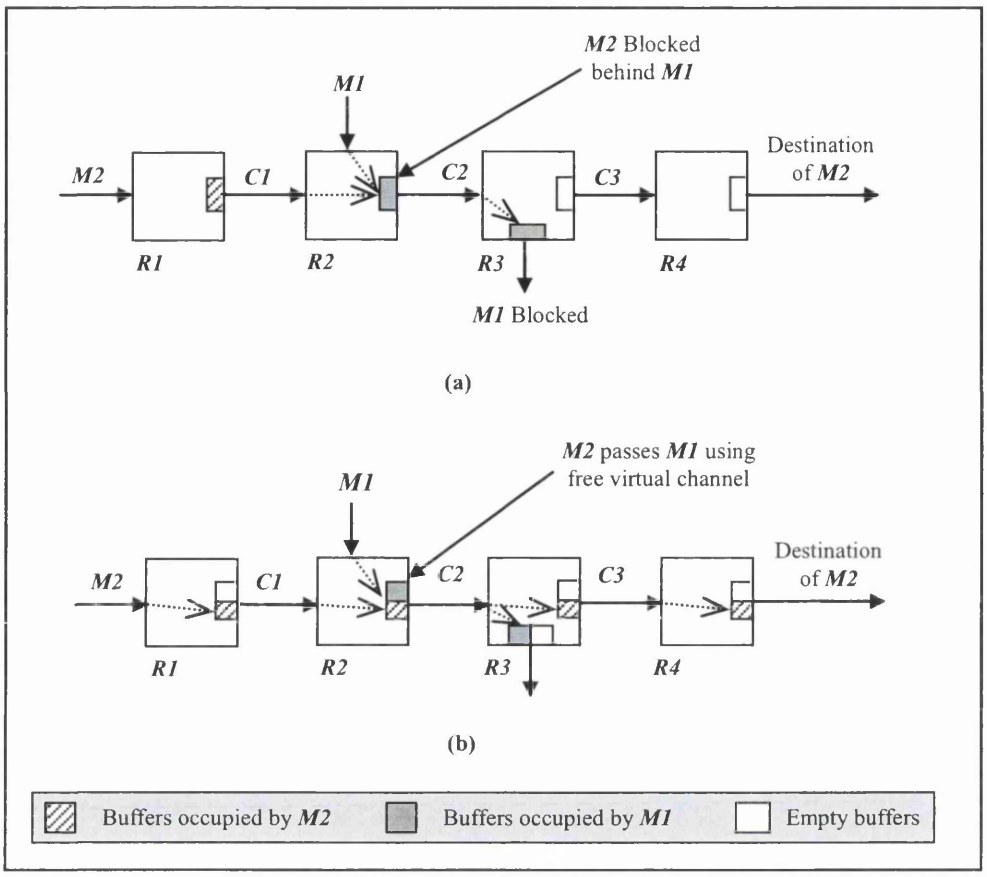


Figure 2.5: Performance improvement using virtual channels: (a) message *M*2 is blocked behind message *M*1; (b) virtual channels provide an additional buffer, allowing message *M*2 to bypass the blocked message *M*1.

Figure 2.5(a) shows the situation where just a single virtual channel per physical channel is used. In this case, channels *C*2 and *C*3 are both idle because message *M*1 is blocked. Therefore, message *M*2 is also blocked at router *R*2 and is unable to acquire channel *C*2. This is because message *M*1 holds the buffer in router *R*2 which is coupled to the physical channel *C*2. Figure 2.5(b) depicts the addition of virtual channels to the network illustrated in Figure 2.5(a). Each physical channel is split into two virtual channels. Although message *M*1 remains blocked and holds the upper buffer located in router *R*2, message *M*2 can now progress towards its destination because the lower buffer in router *R*2 is available, allowing it to access the physical channel *C*2.

Virtual channels dedicated to a physical channel may be organised in a number of different possible ways. Figure 2.6 shows several arrangements of a 12-flit buffer into different

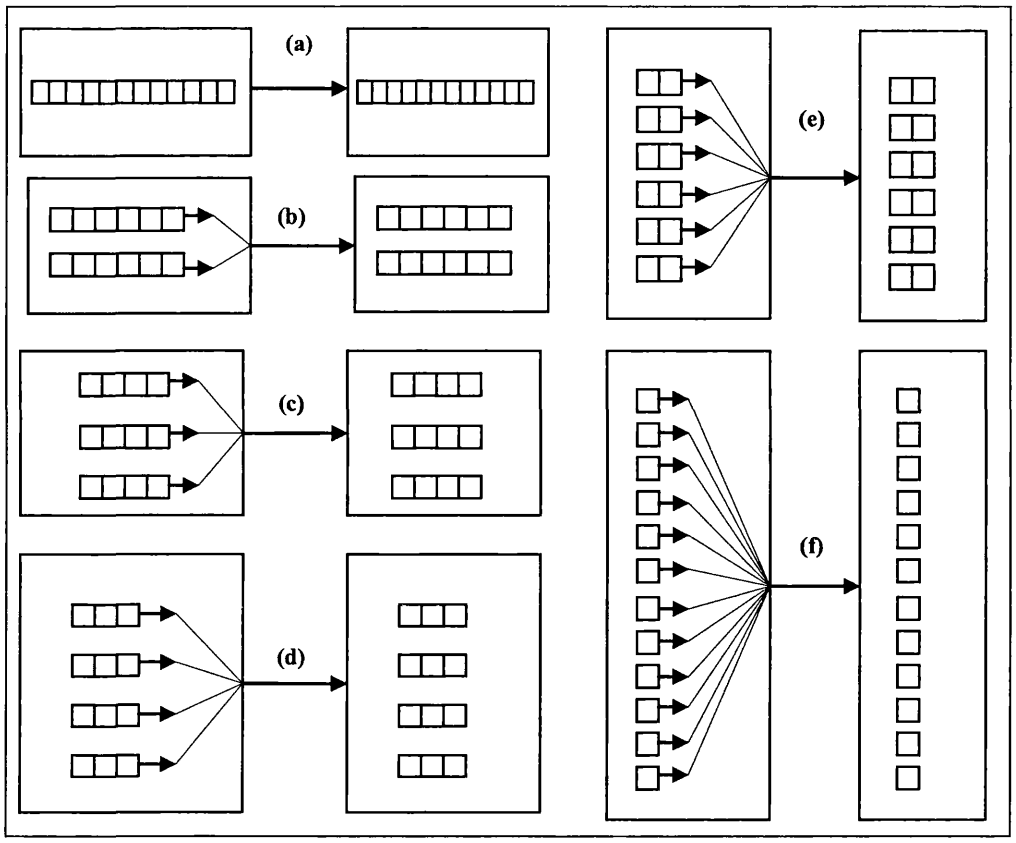


Figure 2.6: Arranging a 12-flit buffer in several ways; (a) when no virtual channels are used, the buffer is organised as one queue, while networks using virtual channels may organise it into several arrangements each with different queue size, namely (b) 2×6 flits, (c) 3×4 flits, (d) 4×3 flits, (e) 6×2 flits, and 12×1 flit.

number of virtual channels. The choice of an arrangement differs in terms of performance, hardware requirements and switch complexity [36, 42]. Although it is often observed that increasing the number of virtual channels leads to improvement in network performance, studies have demonstrated that there is an optimal number of virtual channels where the network performance, in term of latency and throughput, is maximised [36, 49].

Multiplexing Virtual channels over a single physical channel involves allocating the channel bandwidth among virtual channels. Several strategies can be used to allocate this resource including first-come-first served, round-robin, random, priority, or deadline scheduling [36]. For the sake of comparison with previous studies [22, 49, 103, 111, 127], this research work assumes a FIFO scheduling strategy for allocating the physical bandwidth among multiple virtual channels.

2.4 Dimension-order routing

Most interconnection networks, including k-ary n-cubes, provide multiple physical paths between two given nodes. This introduces the problem of choosing a best route between many possible alternatives. Routing is responsible for selecting a path for a message to use to cross from source to destination. Deterministic routing [49, 103] has been employed widely in early practical wormhole-switched networks [6, 21, 68, 89, 109] as it offers a simple way to avoid message deadlock. This is achieved by forcing messages to visit the channels in a strict order.

Dimension-ordered routing [40] is a typical example of deterministic routing where messages cross network dimensions in a pre-defined order, reducing to zero the *offset* (i.e. the number of hops that a message needs to traverse from its current node to its destination node) in one dimension before visiting the next. As a result, messages always take the same path between a given pair of nodes. Although this form of routing has the advantage of keeping the router design simple and compact, it cannot take advantage of the high path diversity provided by k-ary n-cubes.

As mentioned in Section 1.1.3, dealing with deadlock situations is a critical requirement for any routing algorithm. It has been shown in [46, 47] that Deadlock can be prevented by avoiding cyclic dependencies from being formed between the channels of the message path. When a message reserves a channel and later requests the use of another channel, possibly several hops further on, there is a waiting dependency between those channels, and if these dependencies form a cycle then we have a deadlock situation. In other words, messages may hold indefinitely some channels while waiting for other channels that are held by other messages, possibly in other nodes.

Dimension order routing is deadlock free when used in the hypercube as it prevents cyclic dependencies from being formed in this network topology [49]. However, dimension-order

Algorithm: dimension-order routing for the n-dimensional torus.
Input: Current node $C=(c_1, c_2, \ldots, c_n)$,
Destination node $D=(d_1, d_2, \ldots, d_n)$.
Output: The channel to route the message through.
begin
i =1;
while $c_i = d_i$ **do**
i = i+1;
loop
if i>n **return** $Chan_{(ejection)}$
$dim\_Offset = |c_i - d_i|$;
if $c_i > d_i$ **then**
if $dim\_Offset > k/2$ **then** $dim = i^+$;
else $dim = i^-$;
if $c_i < d_i$ **then**
if $dim\_Offset > k/2$ **then** $dim = i^-$;
else $dim = i^+$;
repeat
vc = **random** (VC_1);
if (vc **is free**) **then return** $Chan_{(dim,vc)}$
else $VC_1 = VC_1 - \{vc\}$;
until $|VC_1| = 0$;
if $c_i < d_i$ **then** $vc=v_1$
else $vc=v_2$

if (vc **is free**) **then return** $Chan_{(dim,vc)}$
else return (*blocked*);
end.

Figure 2.7: Dimension order routing algorithm for n-D torus with V virtual channels [40]

routing does not eliminate cyclic dependencies in other high radix k-ary n-cubes, such as the torus [40, 49]. This is because the wrap-around connections in the torus can result in deadlock conditions as cyclic dependencies can occur within a given dimension. The authors in [40] proposed using an additional virtual channel to transform the *cycles* into *spirals*. With this approach, at least two virtual channels are needed to implement a deadlock-free

dimension order routing in the torus. One virtual channel is used by messages that traverse wrap-around connections while the other virtual channel is used for all other messages that do not need to traverse any wrap-around connection.

Although two virtual channels per physical channel are sufficient to guarantee deadlock-free dimension-order routing in the torus, the use of more virtual channels improves network performance by increasing throughput and decreasing message blocking [36, 40]. Figure 2.7 depicts the dimension-order routing algorithm for the n-dimensional torus with V virtual channels where $V>2$. Channels are labelled as $Chan_{(dim,v)}$, where $dim \in \{1^+, 1^-, 2^+, 2^-, \ldots, n^+, n^-\}$ is the dimension and $v \in \{1, 2, \ldots, V\}$ is the virtual channel to be traversed by the message. The algorithm assumes that the message header is at a current node with address $C = (c_1, c_2, \ldots, c_n)$ and the destination node is at address $D = (d_1, d_2, \ldots, d_n)$. Using the methodology in [46], virtual channels can be split into two sets: $VC_1 = (v_3, v_4, \ldots, v_V)$ and $VC_2 = (v_1, v_2)$. At each routing step in dimension i, a message can choose any of the (V-2) virtual channels from VC_1. However, if all channels in VC_1 are busy, the message crosses v_1 if $c_i < d_i$; otherwise, it crosses v_2. It has been shown in [125] that this organisation of virtual channels yields the best performance compared to alternative organisations.

2.5 Duato's adaptive routing

Adaptive routing algorithms, which enable messages to explore alternative network paths, have been suggested to overcome the performance limitation of deterministic routing [42, 49, 103]. In adaptive routing, messages reaching a given router have typically several alternative channels to choose from, consequently improving the performance by balancing evenly the traffic in the network channels.

In many existing studies (see for example [20, 27, 37, 46, 91]) researchers have exploited the concept of virtual channels for designing deadlock-free adaptive routing algorithms in wormhole-switched networks. These algorithms display tradeoffs between their degree of adaptivity and the number of virtual channels that are required to prevent deadlock situations. Specifically, introducing more virtual channels usually results in a higher degree of adaptivity. However, a high number of virtual channels results in increased hardware complexity, which can have a negative impact on the router speed, causing substantial degradation in network performance.

The high cost of adaptivity has motivated some researchers to develop adaptive routing algorithms that require a moderate number of virtual channels. Amongst these, Duato's algorithm [46] has been studied extensively [22, 49, 103, 111, 127] and adopted widely in practical systems like the IBM Blue Gene/L [3], Cray T3E [133] and the Reliable Router [38]. Duato's routing allows for efficient router implementation as it requires a limited number of virtual channels to ensure deadlock freedom. In this algorithm, the virtual channels divide the network into two separate virtual networks. A message is routed adaptively without any restriction in the first virtual network. If the message is blocked, it switches to using virtual channels in the second virtual network, which is deadlock-free and therefore provides *escape routes* for messages to break any deadlock that may occur in the first virtual network. A routing *sub-function* is a restriction of a routing algorithm that supplies these escape channels needed by a deadlock-free adaptive routing algorithm. Such a routing sub-function can itself be deterministic [46].

Figure 2.8 illustrates Duato's adaptive routing algorithm for the k-ary n-cube (k>2). The algorithm requires $V, (V > 2)$, virtual channels per physical channel, which are split into two sets: $VC_1 = \{v_3, v_4, \ldots, v_V\}$ and $VC_2 = \{v_1, v_2\}$. The two virtual channels in VC_2 (also called deterministic virtual channels) are used to implement a deadlock-free routing sub-function (i.e. escape routes). The other virtual channels in VC_1 (also called adaptive virtual channels),

Algorithm: Adaptive routing for bidirectional *k*-ary *n*-cube

Input: $C=(c_n,c_{n-1},\ldots,c_1)$ is the address of the current node,

$D=(d_n,d_{n-1},\ldots,d_1)$ is the address of the destination node.

Output: The channel to route the message through.

begin

if *D* = *C* **then return** (*ejectionChannel*);

$P=\{j \mid 1 \le j \le n, c_j \ne d_j\}$;

repeat

i = **random**(*P*);

if ($c_i < d_i$) **then**

if ($|c_i - d_i| < k/2$) **then**

dim = *i*+ ;

else dim = *i*- ;

else

if ($|c_i - d_i| < k/2$) **then**

dim = *i*- ;

else dim = *i*+ ;

endif;

$V = VC_1^{\dim}$;

repeat

vc = **random** (*V*);

if *vc* **is free then return** (*dim,vc*);

else *V*=*V*-{*vc*};

until $|V| = 0$;

$P = P - \{i\}$;

until $|P| = 0$;

$d = max\{i \mid 1 \le i \le n, c_i \ne d_i\}$;

if ($c_d < d_d$) **then**

if ($|c_d - d_d| < k/2$) **then**

dim = *d*+ ; *vc* = *2*;

else dim = *d*- ; *vc* = *1*;

else

if ($|c_d - d_d| < k/2$) **then**

dim = *d*- ; *vc* = *2*;

else dim = *d*+ ; *vc* = *1*;

endif;

if *vc* **at dimension** dim **is free then return** (*dim, vc*);

else return (*blocked*);

end.

Figure 2.8: Duato's adaptive routing for bidirectional *k*-ary *n*-cubes [46]

can be visited adaptively in any order that brings the message closer to its destination. At any routing step, a message firstly checks the adaptive virtual channels (channels in VC_1) of the remaining dimensions to be visited. If more than one adaptive virtual channel are available, one of them is chosen randomly to route through. If all virtual channels in VC_1 are busy, the message is routed through the deterministic virtual channels (channels in VC_2) of the lowest (or alternatively highest) dimension to be visited. If the deterministic virtual channel is also busy, then the message is blocked and waits for that virtual channel to become free.

When $k = 2$, the *k*-ary *n*-cube network collapses to the well-known hypercube. According to Duato's methodology [46], the virtual channels are divided into two sets: $VC_2 = \{v_1\}$ and $VC_1 = \{v_2, v_3, v_4, \ldots, v_V\}$ when designing adaptive routing for the hypercube. This is because only one virtual channel is needed to implement a deadlock-free deterministic routing sub-function for the hypercube [49]. As described earlier, the remaining virtual channels in VC_1 are used adaptively by messages to get closer to their destinations. If all adaptive virtual channels are busy, then the message is routed through the deterministic virtual channel.

2.6 Summary

While Chapter 1 has provided the context and the motivation behind undertaking this research work, this chapter completes the presentation of the background information that is necessary for a clear understanding of the analytical models and comparisons presented in the subsequent chapters of this dissertation. The torus network and its router structure have been introduced. Then, the operation of wormhole switching and virtual channels has been presented. Finally, dimension-order and Duato's routing algorithms have been described.

It has been pointed out that virtual channels influence the performance of both deterministic and adaptive routing algorithms. In an attempt to capture accurately the effects of virtual channels on the network performance, Chapter 3 proposes a new method for virtual channels

occupancy probabilities, based on an M/G/1 queue. The new method can be customised for different traffic conditions and its accuracy has been demonstrated by comparing its predictions against those obtained from an event-driven simulator.

Chapter 3

A New Performance Model for Wormhole-Switched *k*-Ary *n*-Cubes with Virtual Channels

3.1 Introduction

Dividing a physical channel into several virtual channels has been introduced to (a) design adaptive deadlock-free routing algorithms, (b) overcome the chain blocking problem encountered in wormhole-switched networks, and consequently (c) improve the overall performance of the network [36, 40, 42, 49]. Previous studies (for example [36, 40, 119, 130, 146]) have revealed that adding virtual channels reduces greatly the blocking delay and thus improves considerably the performance of wormhole-switched networks. As a result, capturing accurately the effects of virtual channels on network performance is a crucial step towards the development of realistic analytical models for wormhole switched networks.

A number of research studies [22, 71, 93, 102, 111, 126, 127] have proposed analytical models in order to assess the performance of interconnection networks that employ virtual channels. Most of these models have relied on a method proposed by Dally in [36] for capturing the effects of virtual channels on network performance. Dally has modelled virtual channels using a Markov chain. Although his method exhibits good accuracy under light traffic conditions, it loses accuracy as traffic increases. This is mainly because the Markovian assumption does not hold for moderate and heavy traffic conditions due to the blocking nature of the flow control mechanism used in wormhole-switched networks [7].

This chapter develops an analytical model for wormhole-switched k-ary n-cubes with virtual channels. The proposed model is based on a new method for calculating the probability of busy virtual channels in order to capture the effects, on network performance, of virtual channels. In this new approach, the virtual channels are modelled as the customers in an M/G/1 queuing system. The probability of busy virtual channels is then computed by inverting the z-transform of the number of customers in the queue, where v busy virtual channels correspond to v customers in the queue.

One of the main advantages of this virtual channels model is its ability to be customised and adapted to different traffic conditions by using different service time distributions for the M/G/1 queue. Owing to its generality, Dally's method is shown to be a special case of the proposed method, when the message service time is exponentially distributed. The new model exhibits a good degree of accuracy under light, moderate and heavy traffic conditions. As a further validation step, it is demonstrated that the predictions made by the analytical model, based on the new method, match more closely the results obtained through simulations, compared to the model that uses Dally's method. The new M/G/1 model of virtual channels can be integrated easily into any analytical performance model for wormhole-switched networks that employ multiple virtual channels. This is because, it calculates the probability of busy virtual channels while making no assumptions about other parameters considered in the analytical model.

Several researchers have proposed analytical models to estimate the average message latency in k-ary n-cubes. In [111], Ould-Khaoua proposed a model for Duato's [46] adaptive routing and used Dally's method [36] to model the virtual channels. The model in [111] is simple to implement and has been shown to exhibit a good degree of accuracy under light traffic. For the sake of the present discussion, the new model of virtual channels will be incorporated (instead of Dally's) in Ould-Khaoua's model, which is summarised in Appendix A1. The interested reader is further referred to [111] for a more detailed derivation of the model.

The rest of this chapter is organised as follows. Section 3.2 provides the assumptions used in the analysis. Section 3.3 describes briefly Dally's model of virtual channels and then presents a new model for virtual channels. Section 3.4 derives Dally's method as a special case of the new method whereas Section 3.5 presents a two-moment matching approximation for the service time distribution. Section 3.6 validates the proposed method using simulation and compares it against Dally's method. Finally, Section 3.7 concludes the chapter.

3.2 Assumptions

The model uses the following assumptions that have been accepted widely in the literature [2, 5, 22, 35, 45, 62, 71, 93, 102, 111, 126, 127].

a) Nodes generate traffic independently of each other and according to a Poisson process with a mean rate of λ_g messages per cycle.

b) The message length is M flits, each of which requires one cycle to be transmitted from one router to the next.

c) The mean message service time is $\overline{S}$ cycles and consists usually of the message transmission time, routing delay and blocking delay. The calculation of $\overline{S}$ depends on the network topology, traffic pattern and routing algorithm.

d) The message arrival at a physical channel is approximated by an independent Poisson process with a mean rate of λ_c. This approximation has often been adopted in store-and-forward networks. However, previous studies [22, 65, 111] have shown that it still can be used to model message arrivals in wormhole-switched networks.

e) V ($V > 2$) virtual channel are used per physical channel and are organised, according to Duato's adaptive routing algorithm [46], into two sets; set VC_1 contains (V-2) virtual channels, which are crossed adaptively, and set VC_2 contains

two virtual channels, which are crossed deterministically. Detailed description of Duato's adaptive routing has been presented in Chapter 2.

In what follows, the probability of v busy virtual channels per physical channel, as calculated by Dally in [36], is referred to as $\{P_v^{Dally}; 0 \le v \le V\}$. Similarly let $\{P_v^{New}; 0 \le v \le V\}$ represents this probability, as computed using the new method. Also, the probability of i customers in an M/G/1 queue (including the one being serviced) at arbitrary time is referred to as $\{\pi_i^{M/G/1}; i = 0, 1, 2, \ldots\}$.

3.3 The virtual channels model

Due to the blocking nature of wormhole switching, as traffic increase, blocked messages at subsequent channels interrupt the message transmission time making the Markovian assumption in Dally's method no longer valid. This is reflected in a degradation of accuracy that can be noticed in the reported results predicted by the models that used this method to capture the effect of virtual channels under moderate and high traffic conditions (see for example [22, 71, 93, 102, 111, 126, 127]). In the remainder of this section Dally's method is first explained, and then a new method is proposed for computing the probability of busy virtual channels per physical channel.

3.3.1 Dally's method of modelling virtual channels

Dally has modelled virtual channels using a Markov chain [36], shown in Figure 3.1. In this figure, state $\Re_v$ corresponds to v virtual channels being busy (i.e. occupied by message flits). The transition rate out of state $\Re_v$ to state $\Re_{v+1}$ is λ_c, while the rate out of state $\Re_v$ to state $\Re_{v-1}$ is $1/\overline{S}$ (i.e. each message arrival requires a new virtual channel and each service completion releases a virtual channel). The transition rate out of the last state is reduced by λ_c to account for the arrival of messages while a channel is in this state. Solving the equations of this model yields the probability of busy virtual channels [36]

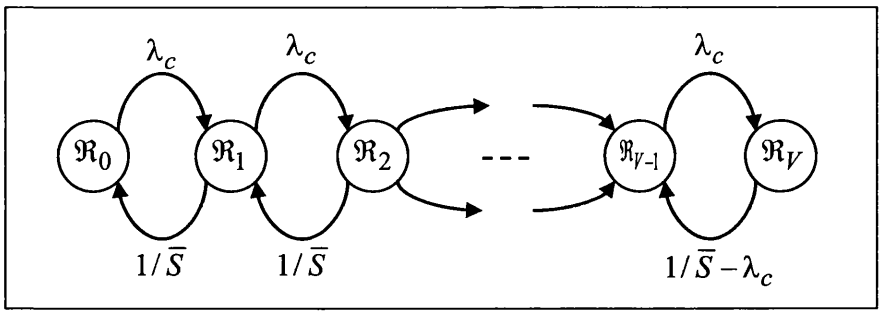


Figure 3.1: The Markov chain, proposed by Dally, for computing the probability of busy virtual channels per physical channel.

$$P_v^{Dally} = \begin{cases} \dfrac{1}{\sum_{i=0}^{V} Q_i} & v = 0 \\ P_{v-1}^{Dally} . \lambda_c \bar{S} & 0 < v < V \\ P_{v-1}^{Dally} . \dfrac{\lambda_c}{1/\bar{S} - \lambda_c} & v = V \end{cases} \tag{3.1}$$

In the above equation, Q_i is a temporary variable and is given by [36]

$$Q_i = \begin{cases} 1 & i = 0 \\ Q_{i-1} . \lambda_c \bar{S} & 0 < i < V \\ Q_{i-1} . \dfrac{\lambda_c}{1/\bar{S} - \lambda_c} & i = V \end{cases} \tag{3.2}$$

Let us now show that the probability of busy virtual channels per physical channel as calculated by Dally in [36] (i.e. $P_v^{Dally}; 0 \le v \le V$) is exactly the same as the probability of the number of customers in an M/M/1 queuing system. To do so, let us eliminate the use of the temporary variable by expanding the summation in the denominator of the probability of no busy virtual channel, P_0^{Dally}, in equation (3.1). Hence we have

$$P_0^{Dally} = \frac{1}{\sum_{i=0}^{V} Q_i} = \frac{1}{1 + (\lambda_c \bar{S}) + (\lambda_c \bar{S})^2 + \ldots + (\lambda_c \bar{S})^{V-1} + (\lambda_c \bar{S})^{V-1} \dfrac{\lambda_c}{1/\bar{S} - \lambda_c}} \tag{3.3}$$

After some manipulation of the above equation we obtain

$$\begin{aligned} P_0^{Dally} &= \left(1 + (\lambda_c \bar{S}) + (\lambda_c \bar{S})^2 + \ldots + (\lambda_c \bar{S})^{V-1} + \frac{(\lambda_c \bar{S})^V}{1 - \lambda_c \bar{S}} \right)^{-1} \\ &= \left(\frac{(1-\lambda_c \bar{S})\left\{1 + (\lambda_c \bar{S}) + (\lambda_c \bar{S})^2 + \cdots + (\lambda_c \bar{S})^{V-1}\right\} + (\lambda_c \bar{S})^V}{1 - \lambda_c \bar{S}} \right)^{-1} \\ &= \left(\frac{1}{1 - \lambda_c \bar{S}} \right)^{-1} = 1 - \lambda_c \bar{S} \end{aligned} \tag{3.4}$$

Equation 3.1 can now be rewritten in terms of P_0^{Dally} as

$$\begin{aligned} P_v^{Dally} &= \begin{cases} 1 - \lambda_c \bar{S} & v = 0 \\ P_{v-1}^{Dally} . \lambda_c \bar{S} & 0 < v < V \\ P_{v-1}^{Dally} . \dfrac{\lambda_c}{1/\bar{S} - \lambda_c} & v = V \end{cases} \\ &= \begin{cases} (1 - \lambda_c \bar{S})(\lambda_c \bar{S})^v & 0 \le v < V \\ (1 - \lambda_c \bar{S})(\lambda_c \bar{S})^{V-1} . \dfrac{\lambda_c}{1/\bar{S} - \lambda_c} & v = V \end{cases} \end{aligned} \tag{3.5}$$

After simplifying the above equations, $\{P_v^{Dally}; 0 \le v \le V\}$ can be expressed as

$$P_v^{Dally} = \begin{cases} (1 - \lambda_c \bar{S})(\lambda_c \bar{S})^v & 0 \le v < V \\ (\lambda_c \bar{S})^V & v = V \end{cases} \tag{3.6}$$

This is simply the probability of the number of customers in an M/M/1 queuing system; see for example [17, 77, 108].

By virtue of the above discussion, Dally's method for calculating the probability of busy virtual channels per physical channel, $\{P_v^{Dally}; 0 \le v \le V\}$, has now been reduced to the calculation of the probability of the number of customers in an M/M/1 queuing system. This approach is accurate under low traffic conditions where blocking delays have only minor

effects on the message transmission time. However, as traffic increase, the Markovian assumption does not hold due to high message blocking in the network [7].

3.3.2 The new method of modelling virtual channels

Motivated by the above observations, a new method for calculating the probability of busy virtual channels per physical channel, $\{P_v^{New}; 0 \le v \le V\}$, in wormhole-switched k-ary n-cubes has been developed. Recall that in wormhole switching and virtual channels flow control (see Chapter 2), when a message arrives at a given physical channel it requests a new virtual channel to be used for its transmission. The message is granted any available virtual channel, which remains engaged until the message is fully transmitted. Once a message is fully transmitted it releases the virtual channel which then can be allocated to other messages.

In the new method, each physical channel is treated as an M/G/1 queue with a mean arrival rate of λ_c messages per cycle and a mean message service time of $\overline{S}$ cycles. The mean arrival rate is the average amount of traffic received by each physical channel (Appendix A1, equation A1.10) whereas the mean service time is equal to the channel holding time (i.e. from the time a message is granted a virtual channel until the last bit of the message is transmitted) and is assumed to be generally distributed with mean $\overline{S}$ (Appendix A1, equation A1.3). The number of busy virtual channels can then be approximated as the number of customers in this M/G/1 queue. Hence, the probability of busy virtual channels, $\{P_v^{New}; 0 \le v \le V\}$, is expressed in terms of the probability of the number of customers in the queue, $\{\pi_i^{M/G/1}; i = 0,1,2,\ldots\}$.

As illustrated in Figure 3.2, when there are v busy virtual channels per physical channel, this corresponds to v customers in the M/G/1 queuing system. The probability that v $(0 \le v < V)$ virtual channels are busy is approximated as the probability of v customers in the system,

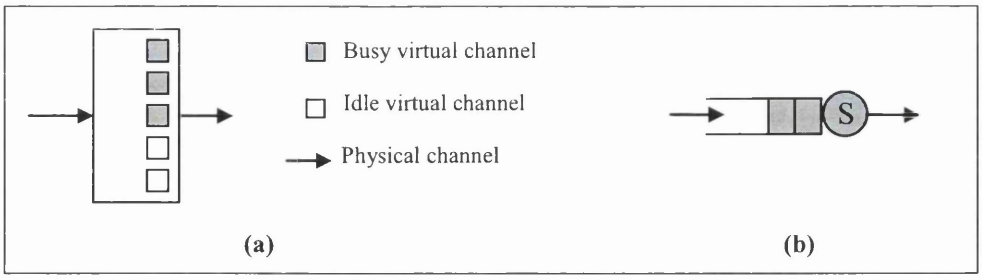


Figure 3.2: The new model for virtual channels. (a) Three busy virtual channels corresponds to (b) Three customers in the M/G/1 queue: two in the queue and one being serviced

including the one being serviced (i.e. $P_v^{New} = \pi_v^{M/G/1}$, $v = 0, 1, 2, \ldots, V-1$). However, in order to calculate the probability of all virtual channels being busy one should take into consideration the requests for new virtual channels while all of them are already occupied by other messages. Hence, the probability that there are more than V customers in the system also corresponds to all (i.e V) virtual channels being busy. The probability that all virtual channels are busy, P_V^{New}, is then approximated as the probability of V customers in the system plus the summation of the probabilities of i customers in the system where $V-1 \le i < \infty$ (i.e. $P_V^{New} = \sum_{i=V}^{\infty} \pi_i^{M/G/1}$). In other words, P_V^{New} is equal to the tail of the probability distribution of the number of customers in the M/G/1 queuing system. To summarise, we can write

$$P_v^{New} = \begin{cases} \pi_v^{M/G/1} & 0 \le v \le V-1 \\ \sum_{i=V}^{\infty} \pi_i^{M/G/1} & v = V \end{cases} \qquad (3.7)$$

The problem is now reduced to finding the probability of the number of customers in an M/G/1 queuing system which is well documented in the literature of queuing theory [17, 77, 108, 141, 145]. For a detailed derivation of the probability of the number of customers in an M/G/1 queuing system, the interested reader is referred to [76, 108, 141]. However, for the sake of completeness and for this dissertation to be self-contained, we present here the z-transform of the number of customers in the M/G/1 queue, which after inverting yields the

probability of the number of customers in the queue, $(\pi_i^{M/G/1}; i = 0, 1, 2, \ldots)$. The z-transform of the number of customers in an M/G/1 queue is given by [141]

$$G_N(z) = \frac{(1-\lambda_c \bar{S})(1-z)}{S^*(\lambda_c(1-z)) - z} S^*(\lambda_c(1-z)) \tag{3.8}$$

In the above equation, $S^*(.)$ represents the Laplace transform of the service time distribution of the M/G/1 queue. Inverting equation (3.8) yields an expression for $\{\pi_i^{M/G/1}; i = 0, 1, 2, \ldots\}$ which, by using equation (3.7), leads to the calculation of the probability of busy virtual channels per physical channel. Equation (3.8) can be determined in two ways [77, 108]. In the first, closed-form inversion methods (for example, table lookup, partial fraction expansion, etc [19]) are used. However, it is often difficult to invert the transform using these methods. In such cases, algorithmic approaches can be used to recursively invert the z-transform [77]. We address this issue in the next section.

Some important observations can be made from equation (3.8). First, using the initial value property of the z-transform [108] and by setting z=0, the probability that the system is idle is given by $1-\lambda_c \bar{S}$, which is the probability of no (or *zero*) busy virtual channels. This result is consistent with Little's Law [108] which implies that the server utilisation of the M/G/1 queue is given by $\lambda_c \bar{S}$. It is noteworthy to mention that both Dally's method and the new method produce the same value for the probability of no busy virtual channels.

Moreover, the complete summation property implies that $G_N(1) = 1$ [108]. This allows us to calculate the probability of all (i.e. V) busy virtual channels without the need to evaluate the infinite summation in equation (3.7). Hence, equation (3.7) can be rewritten as

$$P_v^{New} = \begin{cases} \pi_v^{M/G/1} & 0 \le v \le V-1 \\ 1 - \sum_{i=0}^{V-1} \pi_i^{M/G/1} & v = V \end{cases} \tag{3.9}$$

3.4 Deriving Dally's method

The aim of this section is twofold. First, it shows the generality and the customisability of the new method by deriving Dally's method as a special case, when the service time (i.e. the message transmission time) is exponentially distributed. Second, it demonstrates how the z-transform in equation (3.8) can be inverted.

Recall that the Laplace transform of the exponential service time distribution with a mean $\overline{S}$ is given by [77, 108]

$$S^*(s) = \frac{\frac{1}{\overline{S}}}{s + \frac{1}{\overline{S}}} \tag{3.10}$$

Hence,

$$S^*(\lambda_c(1-z)) = \frac{(1/\overline{S})}{\lambda_c(1-z) + (1/\overline{S})} = \frac{1}{\lambda_c \overline{S}(1-z) + 1} \tag{3.11}$$

Substituting the above Laplace transform into equation (3.8) yields

$$\begin{aligned} G_N(z) &= \frac{(1-\lambda_c\overline{S})(1-z)}{\frac{1}{\lambda_c\overline{S}(1-z)+1} - z\left(\frac{1}{\lambda_c\overline{S}(1-z)+1}\right)} \\ &= \frac{(1-\lambda_c\overline{S})(1-z)}{1-z\{\lambda_c\overline{S}(1-z)+1\}} \end{aligned} \tag{3.12}$$

It is worth mentioning that the common factor $(1-z)$ in the numerator and denominator of the above equation may result in ambiguity when $z=1$. After some algebraic manipulation of the above equation to eliminate $(1-z)$ from the denominator, the z-transform of the number of customers in the M/G/1 queue can be simplified as

$$G_N(z) = \frac{1-\lambda_c\overline{S}}{1-z\lambda_c\overline{S}} \tag{3.13}$$

This can be inverted easily since it corresponds to a geometric random variable with parameter $(1-\lambda_c\bar{S})$ and hence the probability of the number of customers in the queue is given by [108]

$$\pi_i^{M/G/1} = (1-\lambda_c\bar{S})(\lambda_c\bar{S})^i \qquad ; i = 0, 1, 2, \ldots \tag{3.14}$$

Substituting equation (3.14) into equation (3.9) and simplifying yields the probability of busy virtual channels per physical channel as

$$\begin{aligned} P_v^{New} &= \begin{cases} (1-\lambda_c\bar{S})(\lambda_c\bar{S})^v & 0 \le v \le V-1 \\ 1-\sum_{i=0}^{V-1}(1-\lambda_c\bar{S})(\lambda_c\bar{S})^i & v = V \end{cases} \\ &= \begin{cases} (1-\lambda_c\bar{S})(\lambda_c\bar{S})^v & 0 \le v \le V-1 \\ (\lambda_c\bar{S})^V & v = V \end{cases} \end{aligned} \tag{3.15}$$

This is exactly the equation that has been derived by Dally in [36] and here we have derived it as a special case of the new method by inverting equation (3.8).

Different closed-form methods for inverting the *z*-transform are also available in the literature [19]. In [108] the partial fractions and power series expansion methods have been used to invert equation (3.8) when various service time distributions are assumed. However, sometimes it is hard or even impossible to invert the resulting *z*-transform using table lookups or closed-form methods. Appendix A2 outlines a simple algorithmic approach for inverting the *z*-transform to be used in such cases.

3.5 Two moment approximation

It can be seen from equation (3.8) that the new method of calculating the probability of busy virtual channels requires explicitly the distribution of the message service time, which is often unknown. To overcome this limitation, this section presents a two moments-matching approximation [98] of the message service time distribution to be used when only the first

two moments [108] are known. Kleinrock [77] has shown that any distribution function can be approximated as closely as desired by a series-parallel stage-type of exponential distributions. The basic idea is that the parameters of the approximated distribution are calculated by equating the moments of the service time with those of the approximated distribution. This approach has been used widely to approximate the distribution of complex functions when knowledge of them is not available [65, 66, 98]. Hence, the following approximation, based on the squared coefficient of variation, C_S^2, can be used.

When $C_S^2 \geq 0.5$, it is often suggested [17, 145] that the service time can be approximated as a Coxian distribution of order 2 [108]. A random variable X has a Coxian distribution of order 2 if it has to go through up to at most 2 exponential phases with means μ_1 and μ_2, respectively. The random variable starts in the first phase and upon completion it comes to an end with probability β or it enters the second phase with probability $1-\beta$. For this distribution it holds that the squared coefficient of variation is greater than or equal to 0.5. The following parameters are suggested to fit the first two moments of the service time distribution [145]

$$\begin{cases} \beta = (1/2)C_S^2 \\ \mu_1 = 2\overline{S} \\ \mu_2 = \mu_1 \beta \end{cases} \tag{3.16}$$

On the other hand, if $C_S^2 < 0.5$, the service time distribution can be approximated as a mixed Erlang distribution $E_{r,r+1}$, where r is the number of exponential phases [108]. A random variable X has a mixed Erlang distribution of order r if it is the sum of r exponentials, with probability δ, and the sum of r+1 exponentials, with probability $1-\delta$, with the same mean μ [108]. For a mixed Erlang distribution it holds that the squared coefficient of variation is always less than 1. The following set of parameters can be used to fit the first two moments of the service time to a mixed Erlang distribution of order r [145]

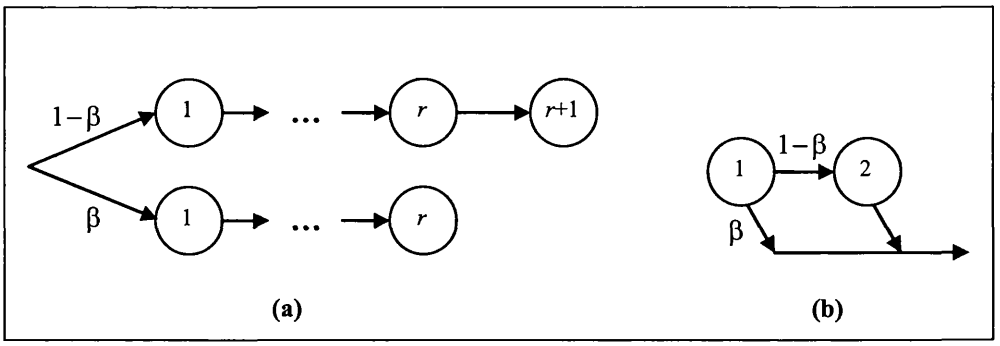


Figure 3.3: Phase diagram for (a) mixed Erlang distribution and (b) two-phase Coxian distribution.

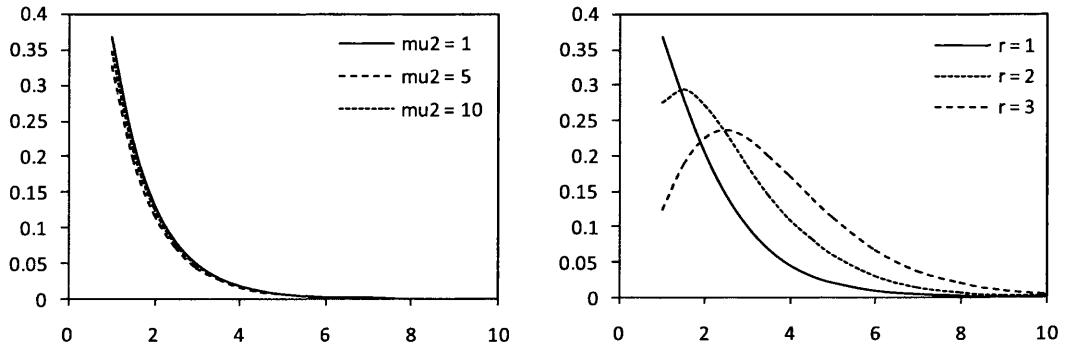


Figure 3.4: The probability density functions of the Coxian-2 and the mixed Erlang distributions; (a) Coxian-2 distribution with β=0.5, μ1=1 and different values for μ2, (b) mixed Erlang distribution with δ =0.5, μ=1 and different values for *r*.

$$\begin{cases} \delta = \dfrac{1}{1+C_S^2}\sqrt{rC_S^2 - [r(1+C_S^2) - r^2 C_S^2]} \\ \mu = \dfrac{r-\delta}{\bar{S}} \end{cases} \tag{3.17}$$

To minimise the number of stages in the mixed Erlang distribution, r is set to be equal to the ceiling of the inverse of the square coefficient of variation (i.e. $r = C_S^{-2}$). However when C_S^2 is extremely small, one can assume a deterministic ($C_S^2 = 0$) service time of $\bar{S}$. Figure 3.3 and Figure 3.4 illustrates the phase diagrams and the probability density functions for both distributions. The Laplace transform of the approximated service time distribution can then be written as

$$S^*(s) = \begin{cases} \dfrac{\beta\mu_1(\mu_2+s)+(1-\beta)\mu_1\mu_2}{(\mu_1+s)(\mu_2+s)} & C_S^2 \geq \dfrac{1}{2} \\ \dfrac{s\beta\mu^r+\mu^{r+1}}{(\mu+s)^{r+1}} & C_S^2 < \dfrac{1}{2} \end{cases} \tag{3.18}$$

Substituting this into equation (3.8) gives the z-transform for the number of customers in the queue. Now, $G_N(z)$ can be inverted to get an expression for the probability of the number of customers in the system (i.e. $\pi_i^{M/G/1}; i = 0, 1, 2, \ldots$). Finally the probability of busy virtual channels is calculated using equation (3.9).

It should be mentioned that the two moments matching approximation described above requires the knowledge of the variance of the service time distribution, σ_S^2. Finding the exact expression for σ_S^2 is a difficult and complex task since this depends on several parameters. An iterative approach has been used to approximate the variance of the service time distribution. Recall that the variance of a given random variable with mean μ is given by

$$\sigma^2 = \int_0^\infty (x-\mu)^2 p(x)dx \tag{3.19}$$

For each value of the random variable X (which is in this case the message service time), the iterative process starts at the minimum service time (i.e. M) and continues until an error bound of $\varepsilon = 0.0001$ is reached. However, given that we are driven by the requirement for analytical simplicity as well as the desire of versatility and practicality, we can use the simple approximation suggested by Draper and Ghosh [45] for computing the variance of the service time distribution. Since the minimum service time at a given channel is equal to the message length, M, the variance of the service time can be approximated as $\sigma_S^2 = (\bar{S} - M)^2$ and then the square coefficient of variation is calculated (i.e. $C_S^2 = \sigma_S^2 / \bar{S}^2$). Both

approximations have resulted in similar predictions of the mean message latency when used in the above analytical model. Therefore, we opt to use the latter as it requires less computation while maintaining good accuracy as can be seen from the figures Section3.6.

3.6 Validation and comparison

The analytical model has been validated against an event driven simulator operating at the flit level by comparing latency results obtained from the simulator to those predicted by the analytical model. The simulator has been developed by previous researchers in our group and has been used extensively since then for different research studies [70, 71, 93, 102, 111, 112, 126, 127, 130]. Furthermore, its output has been validated against that of another simulator developed by Dally [42].

The simulator mimics the behaviour of the k-ary n-cube with multiple virtual channels and supports both adaptive and deterministic routing algorithms. The network cycle time is defined as the time to transmit a single flit from one router to the next. Nodes generate messages (M flits each) according to exponentially distributed inter-arrival times. 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 in the source node until the last data flit reaches the destination node.

For the purpose of comparison, two versions of the analytical model have been plotted; one uses Dally's method and the other uses the new method for modelling virtual channels occupancy probabilities. The implementation of both models are identical except for the calculation of the probability of busy virtual channels, where we have approximated the service time distribution using the two moment matching method presented in Section 3.5.

The batch means method [42, 95] has been used to collect the simulation output where a long simulation run is divided into a set of fixed size batches and the mean is calculated for each

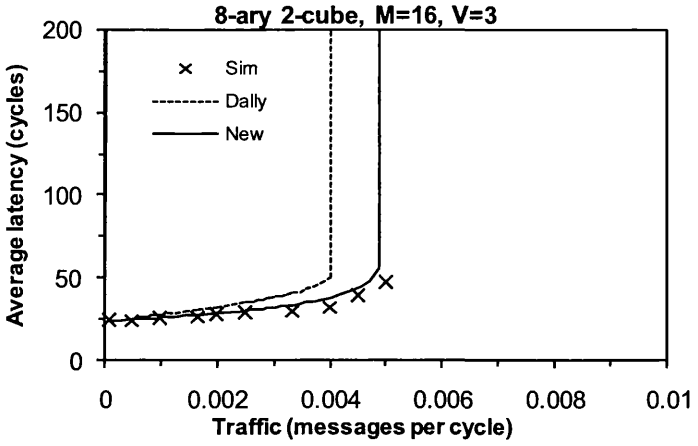
Table 3.1: The mean, variance and 95% confidence interval for some of the simulation points plotted in Figure 3.4(b): 8-ary 2-cube with 5 virtual channels and 16-flit messages.

λ_g	Mean	Variance	95% Confidence interval
0.0001	23.348	0.0022	[23.331 - 23.364]
0.0005	24.37	0.0068	[24.34 - 24.399]
0.001	25.819	0.0223	[25.765 - 25.872]
0.002	29.332	0.0598	[29.244 - 29.419]
0.0025	31.268	0.0942	[31.158 - 31.378]
0.004	37.365	3.1325	[36.732 - 37.999]
0.005	41.57	6.3254	[40.67 - 42.47]
0.006	44.32	12.2924	[43.065 - 45.575]

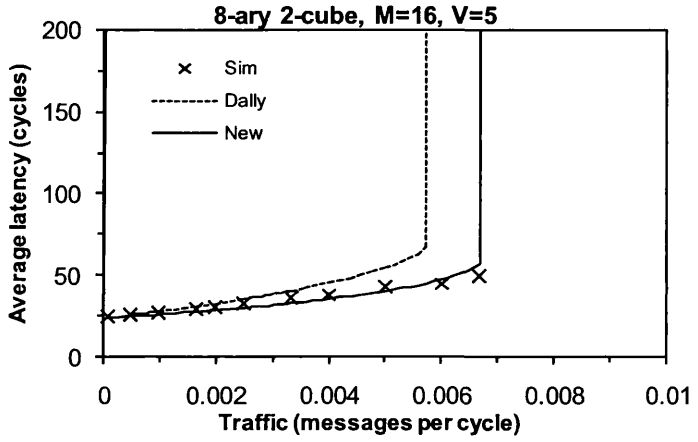
batch. These batch means are then used to calculate the average message latency along with 95% confidence intervals. To compute the confidence interval, it is assumed that the underlying process being measured (in this case the message latency) is stationary and normally distributed. To ensure that the process is stationary, the first two batches from each simulation run is discarded to avoid distortions due to start-up conditions (warm-up period). Discarding the first two batches from the simulation statistics is sufficient to reach the steady state. Discarding more than two batches makes little difference in the collected statistics. While we cannot assume that the message latency is normally distributed, the *central limit theorem* [42] states that the cumulative distribution of many observations (say N) of an arbitrary random process approaches the normal distribution as the number of these observations increases [42, 108] (i.e. N>25 [95]).

Table 3.1 outlines the mean, variance and 95% confidence interval for some of the simulation points plotted in Figure 3.4. However, like existing studies [2, 5, 22, 35, 45, 62, 71, 93, 102, 111, 126, 127] and due to the small confidence interval *half-widths* , only the mean message latency is shown in the presented figures.

(a)



(b)



(c)

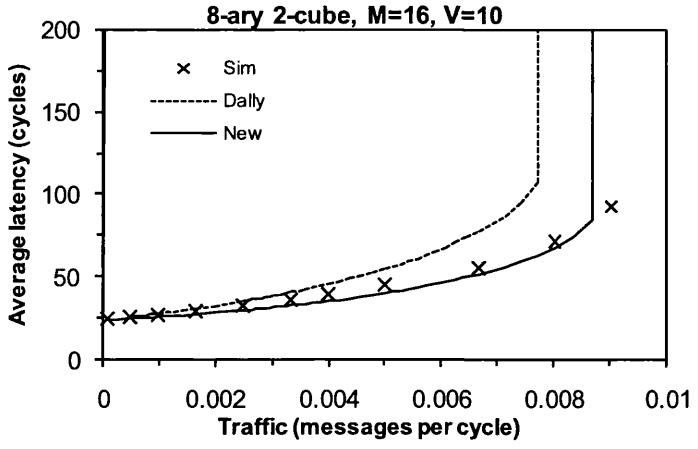
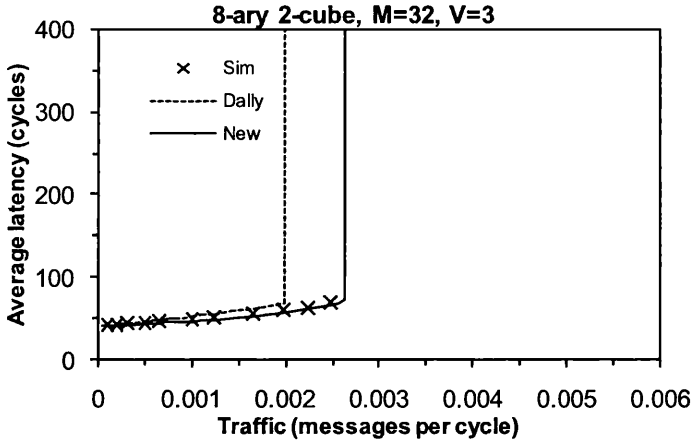
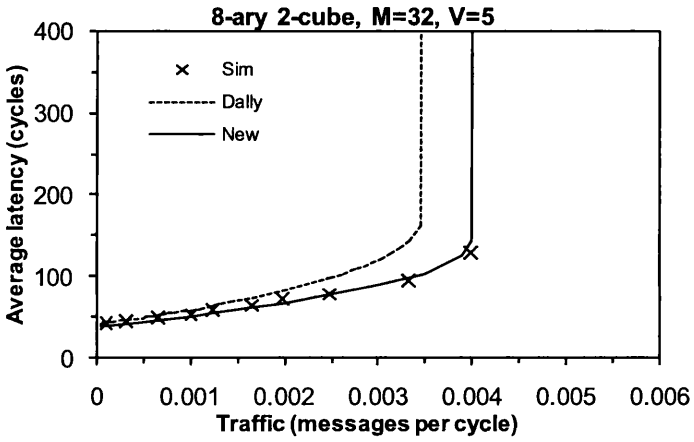


Figure 3.5: The mean message latency in the 8-ary 2-cube predicted by the simulator and analytical model when Dally's method and the new method are used for modelling virtual channels. The message size is 16 flits and the number of virtual channels per physical channel is (a) 3, (b) 5, and (c) 10.

(a)



(b)



(c)

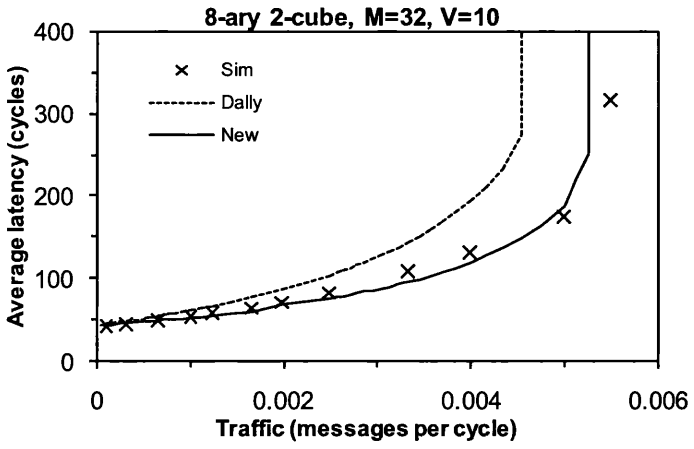
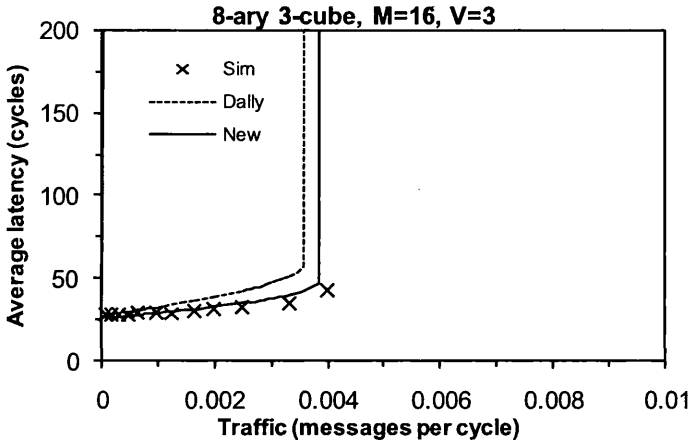
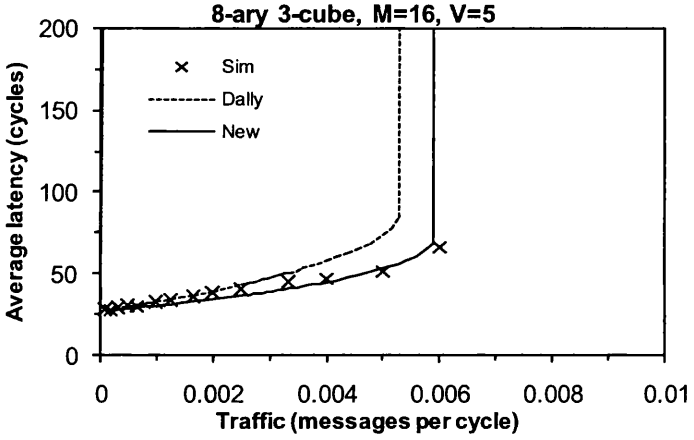


Figure 3.6: The mean message latency in the 8-ary 2-cube predicted by the simulator and analytical model when Dally's method and the new method are used for modelling virtual channels. The message size is 32 flits and the number of virtual channels per physical channel is (a) 3, (b) 5, and (c) 10.

(a)



(b)



(c)

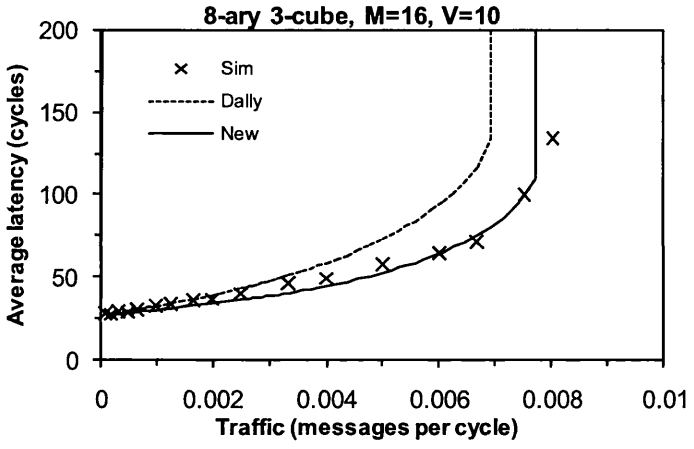


Figure 3.7: The mean message latency in the 8-ary 3-cube predicted by the simulator and analytical model when Dally's method and the new method are used for modelling virtual channels. The message size is 16 flits and the number of virtual channels per physical channel is (a) 3, (b) 5, and (c) 10.

Numerous validation experiments have been conducted for several combinations of network sizes, message lengths, and virtual channels to assess the accuracy of the proposed method. For illustration purposes, Figures 3.5, 3.6, and 3.7 depict latency results for 8-ary 2-cube and 16-ary 2-cube versus the traffic generation rate. The figures are plotted for different message lengths (M=8, 16 and 32 flits) and different numbers of virtual channels (V=3 and V=7). The horizontal axis in all figures shows the traffic generation rate, while the vertical axis shows the mean message latency from the simulator and from the analytical models.

For the purpose of comparing our results with previous studies [70, 71, 93, 102, 111, 112, 126, 127, 130], the traffic will be considered light, moderate and heavy if it is respectively less than 20% , between 20% and 50%, and more than 50% of the network capacity. The figures reveal that both models (i.e. the one that uses Dally's method and the one that uses the new method for modelling virtual channels) are in close agreement with simulation results under low traffic conditions. However, as the traffic rate increases, the model based on Dally's method starts to deviate from the simulation results. Meanwhile, the model that is based on the new method continues to match the simulation results as the network approaches the saturation point. However, some discrepancies near the saturation point are still apparent. These can be accounted for the use of the approximation of the service time distribution and the approximation of the variance. These approximations have been adopted because they simplify greatly the derivation of the model. Nevertheless, it can be concluded that the new model of virtual channels is, firstly, more accurate than Dally's method as it matches the simulation results more closely, and secondly, it can be customised to adopt with different traffic conditions by using different service time distributions.

3.7 Summary

Most existing analytical models that have been developed to assess the performance of wormhole switched networks with virtual channels have employed a method presented by

Dally [36] to calculate the probability of busy virtual channels per physical channel. Dally's method is based on a Markov chain and exhibits good performance predictions under light traffic conditions. However, its prediction accuracy degrades as the traffic rate increases. In this chapter, a new method to calculate the probability of busy virtual channels per physical channel has been proposed and validated. The new model is based on computing the probability of the number of customers in an M/G/1 queue. Furthermore, a two-moment matching approximation of the service time distribution has been described to be used when only the first two moments of the service time are available.

Beside the accuracy that it achieves under light, moderate and heavy traffic loads, one of the main advantages of the new method is the simplicity of tuning it to suit different traffic conditions by using different service time distributions. The generality of the method has been demonstrated by showing that Dally's method is a special case when the message service time follows an exponential distribution. The new method has been compared against that of Dally's and to an event-driven simulator in order to assess its accuracy. Results have revealed that the proposed method agrees with the simulator results more closely than Dally's method, especially under moderate and heavy traffic conditions. Moreover, the new method achieves a good degree of accuracy without compromising the simplicity of the analysis, making it a practical evaluation tool that can be used to gain more realistic insights into the effects of virtual channels on the performance of wormhole-switched networks.

Almost all existing analytical models that have been proposed to assess the performance of wormhole-switched networks have assumed single-flit buffers per channel (see for example [2, 5, 22, 35, 45, 62, 71, 93, 102, 111, 126, 127]). Therefore, performance evaluation studies have so far relied on software simulation [26, 36, 48, 96, 101, 119] to analyse the effects of finite size buffers on network performance. In the next chapters, new analytical models are proposed that are able to capture the effects of introducing finite size buffers on the performance of wormhole-switched k-ary n-cubes.

Chapter 4

Modelling Deterministic Routing with Finite Buffers

4.1 Introduction

Analytical models for deterministic routing in k-ary n-cubes have been reported widely in the literature over the past years (see for example [2, 4, 5, 29, 30, 45, 62, 73, 125]). However, all of these models have assumed unrealistically that network channels are equipped with single-flit buffers. This assumption has been invoked to ease the derivation of the analytical models despite that practical and experimental systems use finite size buffers (i.e. the depth of the buffer is usually more than one flit and is finite). For example, Cray T3E [133], SGI Origin 2000 [53], IBM Blue Gene/L [3] and IBM SP2 [138] supercomputers use deep buffers, leading to improvement in network throughput and message latency [96].

Nevertheless, there have been a few attempts recently to develop analytical models to capture the effects of finite buffers on the performance of wormhole-switched networks. Hu and Kleinrock [66], for example, have proposed a model based on an M/G/1/K queuing system. This model was intended primarily for irregular networks and incorporates a number of computation-demanding estimations. It also assumes large buffer sizes which are suitable for LAN environments where propagation delays are relatively high and large buffers are often employed in order to accommodate typically long transiting messages. In multicomputer environments, however, the propagation delay is low and the buffer requirement is relatively minimised, compared to LAN environments. It has been shown that

this model losses accuracy in its predictions when small buffer sizes (i.e. less than the message size) are used [66].

Similarly, Kouvatsos et al [81, 82] have used the same approach as that suggested in [66] to analyse the performance of the hypercube and 2D torus. The authors [81, 82] propose to replace the Poisson assumption for message arrivals by that of a compound Poisson. However, both studies [66] and [81, 82] have not considered the effects of virtual channels on network performance as they have assumed one virtual channel per physical channel.

This chapter suggests a new analytical performance model for computing the message latency in k-ary n-cubes combining finite size buffers, deterministic routing and multiple virtual channels per physical channel. In the new modelling approach, two types of blocking, that contributes to the message latency are identified and separately modelled. Moreover, unlike the previous models [66, 81, 82], the suggested model is capable of capturing the impact of virtual channels on network performance. The model is validated via simulation experiments and the results will reveal that it yields good latency predictions under various network operating conditions.

The rest of this chapter is organised as follows. Section 4.2 outlines the assumptions and notation used in the analysis while Section 4.3 discusses how finite buffers complicate the model development. Section 4.4 presents the model for the bidirectional torus and Section 4.5 validates it via simulation experiments. Section 4.6 shows briefly how the model can be adapted for the unidirectional torus and hypercube. This extension is important as the models for the unidirectional torus and hypercube will be used in Chapter 6 to compare different network topologies under different operating conditions. Furthermore, these topologies are used in existing parallel computers as can be seen from Table 1.1 in Chapter 1. Finally, the chapter is concluded in Section 4.7.

Table 4.1: Notation used in the model for deterministic routing with finite buffers

Symbol	Description
k	Network radix; number of nodes per dimension
n	Network dimension
N	Total number of nodes in the network ($N = k^n$)
M	Message size
F	Buffer size $1 \le F \le M$
V	Number of virtual channels per physical channel
$\overline{S}$	Mean network latency
$\overline{W}_s$	Mean waiting time at the source
$\overline{V}$	Degree of virtual channel multiplexing
$A_{h,i}^{j}$	Number of effective channels of the i^{th} step of an h-hop message
$A_{h,i}^{j,l}$	Effective channels at dimension l when the h-hop message is at its i^{th} step
S_h	Mean network latency of an h-hop message
W_s^h	Mean waiting time of an h-hop message
V_h	Degree of virtual channels multiplexing of an h-hop message
λ_g	Input traffic generated by each node per cycle
λ_c	Average traffic received by a physical channel per cycle
$\hat{\lambda}_c$	Traffic rate expressed in flits per cycle
λ_s	Traffic rate received by an injection channel
H	Maximum number of hops between any two nodes
H^+	Maximum number of hops at any dimension of an h-hop message
H^-	Minimum number of hops at any dimension of an h-hop message
Ω_h^n	Total number of way to distribute h hops over n dimensions
C_c^h	One combination of distributing h-hops over n dimensions ($1 \le c \le \Omega$)
P_h^{hop}	The probability of generating an h-hop message
n_h	The number of nodes that are h-hops a way from a given source
$B_{h,i}^{j}$	Blocking delay of an h-hop message at its i^{th} step at dimension j
$\theta_{h,i}^{j}$	Probability of blocking due to contention for virtual channels
$\Upsilon_{h,i}^{j}$	Probability of blocking due to the finite buffer
$\varphi_{h,i}^{j}$	Probability that an h-hop message gets blocked at its i^{th} step at dimension j
$W_{h,i}^{j}$	Mean waiting time of an h-hop message at its i^{th} step at dimension j
$T_{h,i}^{j}$	Channel holding time of the i^{th} step of an h-hop message at dimension j
$\hat{T}_{h,i}^{j}$	Channel holding time of a flit
$P_v^{T_{h,i}^{j}}$	Probability of v busy virtual channel when the channel holding time is $T_{h,i}^{j}$
$\pi_\kappa^{M/G/1/F+1}$	Probability that there are κ , customers in the M/G/1/F+1 queue

4.2 Assumptions and notation

The node and the router structure of the bidirectional k-ary n-cube used in the development of the analytical model have already been described in Chapter 2. Furthermore, the model presented in this chapter is based on the following assumptions, which are used commonly in similar studies [2, 4, 5, 29, 30, 45, 62, 66, 73, 81, 82, 125]. The notation used in the derivation of the model are also summarised in Table 4.1.

a) Each node generates traffic independently of all other nodes and follows a Poisson process with a mean rate of λ_g messages per network cycle; a cycle is the time to send a single flit from one router to the next.

b) The messages arrival process at a given physical channel is approximated by an independent Poisson process. This approximation has often been used in store-and-forward networks [77, 78], which differs substantially from wormhole-switched networks in various aspects (for example, packet storing and forwarding as opposed to flit buffering and pipelining). However, simulation experiments from previous studies [22, 65, 111, 122, 127] have shown that the Poisson process can still be used to model the messages arrival process at each physical channel of wormhole-switched networks.

c) The message length is fixed and is equal to M flits. Moreover, message destinations are distributed uniformly across the network nodes.

d) Messages are routed according to the deterministic *dimension-order* routing algorithm described in Section 2.4. Each router takes one network cycle to decide on the output channel that should be used by the message flits.

e) Each physical channel is divided into V $(V > 2)$ virtual channels, each of which is equipped with a finite buffer of size F flits $(1 < F \leq M)$ resulting in a total buffer size of FV flits per physical channel.

f) The local queue of the injection channel in the source node has infinite capacity. Moreover, messages are transferred to the local processing element (PE) as soon as they arrive at their destinations through the ejection channel (i.e. there is no blocking at ejection channels).

4.3 Modelling finite buffers

This section discusses as to why the presence of finite size buffers requires a different analysis from that of single flit buffers. The provision of finite buffering complicates the development of the analytical model in two ways. Firstly, the commonly used assumption that a message reaches its destination before its tail leaves its source node is no longer valid. In case of blocking, and due to the available finite buffer space, a blocked message might occupy only a fraction of the channels along its path. The channel holding time (i.e. the time from the moment a message is granted access to the virtual channel until all its flits are transmitted over that channel) is now affected by only a limited number of channels. As a consequence, unlike previous models which have assumed one flit buffers, the mean network latency (defined below) cannot be used as an approximation for the channel holding time and hence another approximation should be adopted.

When single-flit buffers constraint is assumed, messages get blocked only at the head of the queue, waiting for a virtual channel to become free. However, if this constraint is released, a message may also be blocked inside the buffer due to insufficient space to accommodate it. This challenge can be dealt with by forcing the message size to be exact multiples of the buffer size (i.e. blocking can happen only at the head of the buffer because messages can always be fully accommodated in the equipped channels). Nevertheless, the model presented in this chapter assumes no restrictions on the message size or the buffer size and hence both types of blocking must be captured and evaluated. Figure 4.1 illustrates these two types of blocking when finite size buffers are assumed.

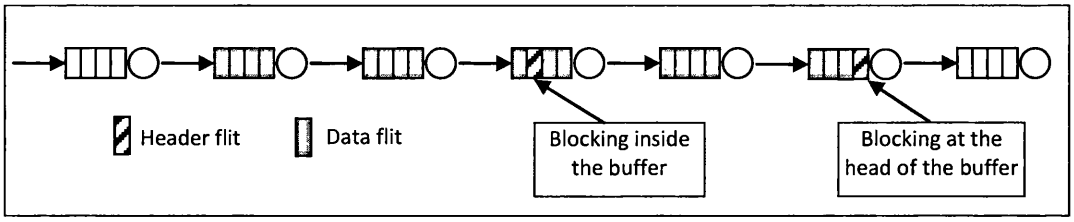


Figure 4.1: An illustration of two messages (10 flits each) blocked at the head of and inside a finite buffer.

4.4 The analytical model

The proposed analytical model calculates the mean message latency as the summation of the mean network latency, $\bar{S}$, and the mean waiting time at the source node, $\bar{W}_s$. The mean network latency is the time a message takes to cross the network from the point where it leaves the local queue at the source node until it arrives to its destination node. This includes the transmission delay, routing delay, and blocking delay. On the other hand, the mean waiting time at the source node is the time that a message waits in the local queue of the injection channel at the source node. However, because multiple virtual channels are multiplexed over the same physical channel, message latency has to be scaled by a factor, $\bar{V}$, in order to capture the effect of the multiplexing that takes place at each physical channel. Therefore the mean latency of a message crossing the network can be expressed as [114]

$$Mean\ Message\ Latency = \left(\bar{S} + \bar{W}_s\right)\bar{V} \tag{4.1}$$

In what follows, we will describe the calculation of $\bar{S}$, $\bar{W}_s$ and $\bar{V}$ but first we investigate how finite buffers affect the channel status in the event of message blocking.

4.4.1 The number of effective channels ($A_{h,i}^{j}$)

To estimate the blocking delay encountered at each hop it is important to first analyse how the existence of finite buffers affects the channel status and the message transmission time. Introducing finite buffers reduces the number of channels a message can occupy in the event

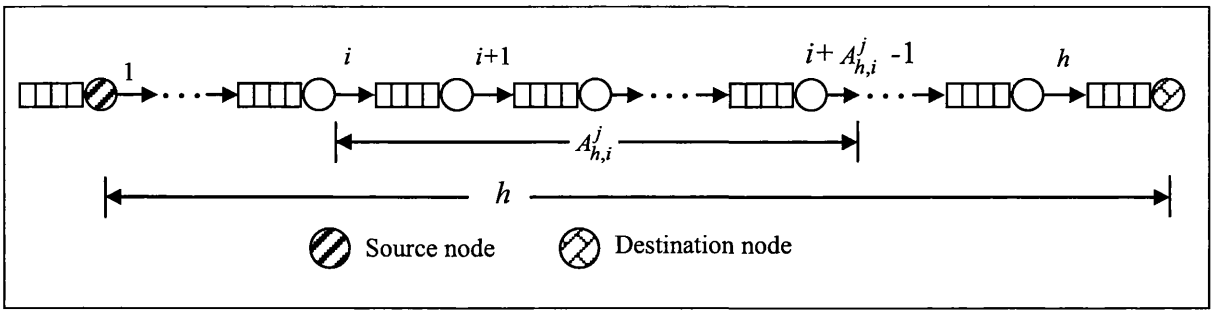


Figure 4.2: An illustration of the number of effective channels.

of blocking. The channel holding time is then affected by only a limited number of subsequent channels. That is, any blocking that occurs after a certain number of channels (i.e. effective channels) do not affect the channel holding time. This is because the message has already been accommodated fully in the finite buffers of the routers along the effective channels of its path. The number of effective channels is calculated as follows.

Definition: The number of effective channels for an h-hop message at its i^{th} step (which is in dimension j) is an integer $A^j_{h,i}$ such that any blocking that occurs in the next $A^j_{h,i}$ channels increases the channel holding time at step i and is given by

$$A^j_{h,i}=\begin{cases}\lceil M/F \rceil & h-i+1 \geq \lceil M/F \rceil \\ h-i+1 & otherwise\end{cases} \qquad (4.2)$$

In the above equation, $\lceil M/F \rceil$ denotes the number of buffers that are necessary to accommodate an entire message. The ceiling operator is to account for any partially filled buffer and is required because the message size may not be exact multiples of the buffer size.

The above equation can be explained as follows. A message that is about to make its i^{th} step has already passed i-1 channels of its h-hops journey and still has to traverse $h-i+1$ channels to arrive at its destination. It can be seen from Figure 4.2 that when the number of the remaining channels to be traversed (i.e. $h-i+1$) is less than the number of buffers that are necessary to accommodate the entire message, $\lceil M/F \rceil$, then any blocking that may

occur at the remaining channels to be traversed will increase the channel holding time of the i^{th} step. On the other hand, if the remaining channels to be traversed are more than the buffers that are required to fully accommodate the message, then the number of effective channels is simply $\lceil M/F \rceil$. Figure 4.2 demonstrates that any blocking that occurs after the next $A_{h,i}^{j}$ channels does not affect the channel holding time at step *i*.

4.4.2 Mean network latency ($\bar{s}$)

In bidirectional *k*-ary *n*-cubes, the maximum number of hops a message can make over one dimension is $\lfloor k/2 \rfloor$, and hence the maximum number of hops a message may make to cross the network is

$$H = n\left\lfloor \frac{k}{2} \right\rfloor \tag{4.3}$$

We proceed by calculating the mean network latency for messages that are *h*-hops ($1 \le h \le H$) away from the destination, S_h. An *h*-hop message is a message that needs to traverse *h* channels (hops) to reach its destination. The mean network latency $\bar{S}$ can be written as the weighted average of the mean network latency for all possible hops. If P_h^{hop} represents the probability of generating an *h*-hop message, we can write

$$\bar{S} = \sum_{h=1}^{H} P_h^{hop} S_h \tag{4.4}$$

To determine P_h^{hop}, let n_h be the number of nodes that are *h*-hops away from the source node. Since the network is symmetric and regular, this number is the same for any given source node. Because the traffic is distributed uniformly over the network nodes, the probability of generating an *h*-hop message can be written as

$$P_h^{hop} = \frac{n_h}{N-1} \tag{4.5}$$

To determine n_h, we utilise some results related to the topological properties of k-ary n-cubes presented in [131]. The number of nodes that are h-hops away from a given node is given by (a detailed proof is available in [131])

$$n_h = \begin{cases} \sum_{i=0}^{n} \sum_{j=0}^{i} (-1)^j 2^i \binom{n}{i} \binom{i}{j} \binom{h-kj/2-1}{i-1} & k \text{ is odd} \\ \sum_{t=0}^{n} \sum_{i=0}^{n-t} \sum_{j=0}^{i} (-1)^j 2^i \binom{n}{t} \binom{n-t}{i} \binom{i}{j} \binom{h-k(j+t)/2-1}{i-1} & k \text{ is even} \end{cases} \quad (4.6)$$

For the uniform traffic pattern, the statistical characteristics of a channel (for example, channel utilization) in a given dimension are identical to those of any other channel in the same dimension when single flit buffers are assumed. However, introducing finite buffers leads to differentiation among channels even in the same dimension as the number of effective channels changes from one hop to the next. Moreover, when dimension order routing is employed, messages at different dimensions may experience different channels characteristics. This is because a message that gets blocked when trying to traverse the j^{th} dimension may occupy some channels from higher (alternatively lower) dimensions but will not be using any channels from the lower (alternatively higher) dimensions. Thus, to compute the mean message latency for an h-hop message we should consider all possible ways to distribute these h hops over n dimensions. To do this, let us refer to the following result from combinatorial theory [108, 120].

Proposition: *The number of ways to distribute r like objects into m different cells such that no cell contains less than p objects and not more than p+q-1 objects is given by* [108, 120]

$$N_p^{p+q-1}(r,m) = \sum_{i=0}^{m} (-1)^i \binom{m}{i} \binom{r-mp-iq+m-1}{m-1} \quad (4.7)$$

Let Ω_h^n denote the number of ways to distribute h hops over n dimensions such that the maximum and the minimum number of hops per dimension are H^+ and H^-, respectively. By using the above preposition, Ω_h^n can be calculated as

$$\Omega_h^n = N_{H^-}^{H^- + H^+ - 1}(h,n) \tag{4.8}$$

Since h can be less than $k/2$, the maximum number of hops per dimension, H^+, is the minimum of h and $k/2$. Similarly, the minimum number of hops per dimension, H^-, is zero if h is less than H^+; otherwise it is the number of hops that are left after the message makes H^+ hops in the rest of the dimensions. Therefore, H^+ and H^- are given by

$$H^+ = \begin{cases} k/2 & h > k/2 \\ h & h \le k/2 \end{cases} \tag{4.9}$$

$$H^- = \begin{cases} h - \dfrac{k(n-1)}{2} & h > k(n-1)/2 \\ 0 & h \le k(n-1)/2 \end{cases} \tag{4.10}$$

Let each combination of the possible ways of arranging the h hops over the n dimensions be denoted as $C_c^h = (d_c^0, d_c^1, \ldots, d_c^{n-1});\ \ 1 \le c \le \Omega_h^n$, where $(d_c^j; 0 \le j \le n-1)$ is the number of hops that an h-hop message makes over dimension j such that $\sum_{j=0}^{n-1} d_i^j = h$.

The network latency of an h-hop message, S_h, is composed of the time to transmit the message M, the routing time h, and the blocking delays, $B_{h,i}^j$, encountered by the message and averaged over all possible ways of distributing h-hops over n dimensions. If n is equal to 2 (i.e. the network is 2D torus), then S_h can be written as

$$S_h = M + h + \frac{\sum_{c=1}^{\Omega_h^2} \left(\sum_{i=h-d_c^0+1}^{h} B_{h,i}^0 + \sum_{i=h-d_c^0-d_c^1+1}^{h-d_c^0} B_{h,i}^1 \right)}{\Omega_h^n} \tag{4.11}$$

In the above equation, a message makes d_c^0 hops in dimension 0 and d_c^1 in dimension 1 such that $d_c^0 + d_c^1 = h$ for all possible combinations. Similarly, when the network is a 3D torus, S_h can be written as

$$S_h = M + h + \frac{\sum_{c=1}^{\Omega_h^3} \left(\sum_{i=h-d_c^0+1}^{h} B_{h,i}^0 + \sum_{i=h-d_c^0-d_c^1+1}^{h-d_c^0} B_{h,i}^1 + \sum_{i=h-d_c^0-d_c^1-d_c^2+1}^{h-d_c^0-d_c^1} B_{h,i}^2 \right)}{\Omega_h^3} \tag{4.12}$$

Hence, for any $n>1$, the above equations can be generalised as

$$S_h = M + h + \frac{\sum_{c=1}^{\Omega_h^n} \left(\sum_{j=0}^{n-1} \left(\sum_{i=h+1-d_c^0-d_c^1-\cdots-d_c^j}^{h-d_c^0-d_c^1-\cdots-d_c^{j-1}} B_{h,i}^j \right) \right)}{\Omega_h^n} \tag{4.13}$$

The first and second terms of the above equation represent, respectively, the transmission time and the routing time of an h-hop message. The third term represents the average blocking delay encountered by the message and can be read as follows. For every possible combination of distributing h hops over n dimensions, C_c^h $(1 \le c \le \Omega_h^n)$, the blocking delay at the i^{th} step (which is at dimension j) is accumulated and then averaged over the number of all possible combinations.

4.4.2.1 Blocking delay ($B_{h,i}^j$)

Releasing the single-flit buffers constraint, by allowing arbitrary buffer size, causes messages to experience not only contention delay at the head of the buffer but also queuing delay inside the buffer. Upon its arrival at an intermediate router, the message requests a virtual channel and granted one if any is available. Otherwise it gets blocked and competes with other messages for the next available virtual channel. Once the message holds a virtual channel, its flits start to flow through that virtual channel until the header flit reaches the head of the finite buffer in the next router. This process is then repeated at each router along the network path until the message reaches its destination. However, if other messages are currently holding (entirely or partially) the buffer, the message is then blocked from reaching the head of the buffer until the other message is transmitted. Therefore, assuming that the above two events (i.e. the contention for virtual channels and the status of the finite buffer) are independent, the probability of blocking at each routing step i of an h-hop message at

dimension j, $\varphi_{h,i}^{j}$, can be written as the product of the probabilities of blocking due to contention for virtual channels, $\theta_{h,i}^{j}$, and due to insufficient buffer space, $\Upsilon_{h,i}^{j}$. That is

$$\varphi_{h,i}^{j}=\theta_{h,i}^{j}.\Upsilon_{h,i}^{j} \tag{4.14}$$

Hence, the mean blocking delay of an h-hop message at its i^{th} step can be approximated as

$$B_{h,i}^{j}=\varphi_{h,i}^{j}.W_{h,i}^{j} \tag{4.15}$$

In the above equation, $W_{h,i}^{j}$ represents the mean time that a message spends waiting in the event of blocking. It can be determined by modelling each router as an M/G/1 queue. If λ_c is the arrival rate, $E[S]$ and $E[S^2]$ are, respectively, the first and the second moments of the message service time, then the mean waiting time of the M/G/1 queue is given by [77]

$$W_{h,i}^{j}=\frac{\lambda_c E[S^2]}{2(1-\lambda_c E[S])} \tag{4.16}$$

Since the traffic is distributed uniformly over all nodes, the arrival rate (i.e. the mean traffic received by each physical channel) can be calculated as follows. Each PE generates λ_g messages per cycle, resulting in a total of $N\lambda_g$ messages per cycle in the network, which are distributed evenly over $2nN$ channels in the network (N nodes with $2n$ channels per node). This traffic can be destined to any node that is h-hops away with probability P_h^{hop}. Hence, the effective traffic rate received by each channel can be written as

$$\lambda_c=\sum_{h=1}^{H}\left(P_h^{hop}h\frac{N\lambda_g}{2nN}\right) = \frac{\lambda_g}{2n}\sum_{h=1}^{H}\left(P_h^{hop}h\right) \tag{4.17}$$

The first moment of the service time (or simply the mean service time) is exactly the channel holding time, $T_{h,i}^{j}$ (calculated below). In other words, a message holds the channel during the course of its transmission. Calculating the second moment of the service time, $E[S^2]$,

requires explicit knowledge of the service time distribution. Finding the exact distribution is a difficult and complex task. Furthermore, given that we are driven by the requirement of analytical simplicity, as well as the desire of versatility and practicality, and because the minimum service time at a given channel is equal to the message length, the variance of the service time can be approximated as $(T_{h,i}^{j} - M)^2$. This approximation has been suggested by Draper and Ghosh [45] and has been used in similar network modelling studies [22, 71, 93, 102, 111, 114, 127]. Hence the second moment of the service time is given by [108]

$$E[S^2] = (T_{h,i}^{j} - M)^2 + (T_{h,i}^{j})^2 \tag{4.18}$$

Substituting the above into equation (4.16) yields the mean waiting time as

$$W_{h,i}^{j} = \frac{\lambda_c \left[(T_{h,i}^{j} - M)^2 + (T_{h,i}^{j})^2 \right]}{2\left(1 - \lambda_c T_{h,i}^{j}\right)} \tag{4.19}$$

Let us assume that in dimension order routing messages are routed first in higher dimensions then in lower dimensions (i.e. messages first traverse dimension n-1 then dimension n-2 and so on until they reaches dimension *zero*). In the following three subsections we calculate the channel holding time, $T_{h,i}^{j}$, the probability of blocking due to contention for virtual channels, $\theta_{h,i}^{j}$ and the probability of blocking due to the finite buffer, $\Upsilon_{h,i}^{j}$.

4.4.2.2 Channel holding time ($T_{h,i}^{j}$)

When finite buffers are employed, a message occupies only a portion of the channels along its path. Therefore, the mean message latency cannot be used to approximate the service time (i.e. the channel holding time) of a message. In this section we calculate the channel holding time as a function of the number of effective channels to account for the deployment of finite size buffers.

To compute the channel holding times, the number of effective channels at each of the remaining dimensions to be visited must first be determined. A message at dimension j still needs to visit dimensions (j-1, j-2, ..., 1, 0) to reach its destination. Therefore, the number of effective channels at all higher dimensions (i.e. $j+1, j+2, \ldots, n-1$) is *zero*, as the message has already crossed these dimensions. However, the number of effective channels in the lower dimensions is the total effective number of channels minus the effective number of channels in the dimensions that have not yet been traversed. Therefore, in virtue of the above discussion, the effective channels of dimension l when an h-hop message makes its i^{th} step at dimension j, $A_{h,i}^{j,l}$, is given by the following three equations

$$A_{h,i}^{j,l} = 0 \qquad ; l > j \tag{4.20}$$

$$A_{h,i}^{j,l} = min\left\{ A_{h,i}^{j}, (h-i+1) - \sum_{m=0}^{j-1} d_c^m \right\} \qquad ; l = j \tag{4.21}$$

$$A_{h,i}^{j,l} = min\left\{ d_c^l \, , A_{h,i}^{j} - \sum_{m=l+1}^{j} A_{h,i}^{j,m} \right\} \qquad ; l < j \tag{4.22}$$

Note that for a given combination of distributing h hops over n dimensions, we can write

$$\sum_{l=0}^{j} A_{h,i}^{j,l} = A_{h,i}^{j} \tag{4.23}$$

The blocking nature of wormhole switching leads to a differentiation among the channels in different dimensions when deterministic routing is used. This is because the number of channels to contend for decreases as the message travels from one dimension to the next. Also, when finite buffers are employed, messages will have different holding times as the effective numbers of channels are different for each hop, leading to different blocking delays. As a result, different channel holding times for a given combination, $C_c^h = (d_c^0, d_c^1, \ldots, d_c^{n-1})$, of distributing the h-hops of a message over n dimensions should be considered. To achieve this, we need to trace back the path of the message starting from the ejection channel to the highest dimension.

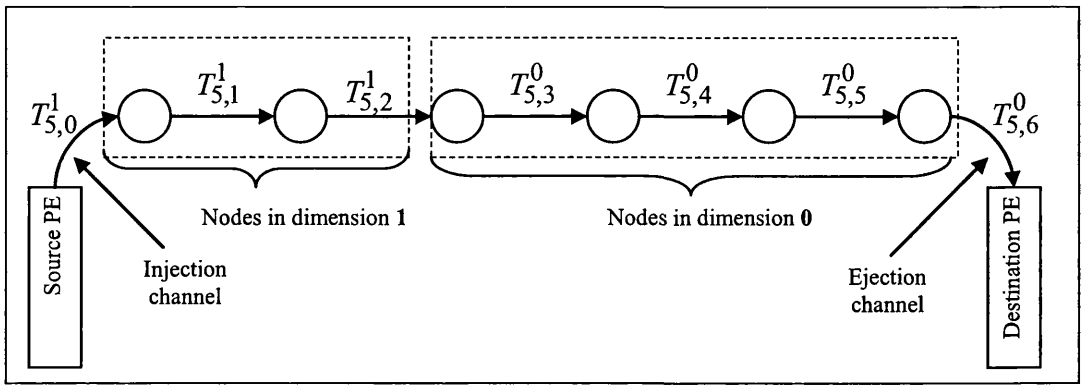


Figure 4.3: A 5-hop message being transmitted in a 2D torus where in this specific combination, $C_c^5 = (3,2)$, it traverses 3 channels from one dimension and 2 channels in the other dimension. The link holding times are labelled accordingly.

Since there is no blocking in the destination (assumption **f**), the holding time of the ejection channel, denoted by $T^0_{h,h+1}$, is simply the message length (i.e. $T^0_{h,h+1} = M$). A message in dimension *zero* has already crossed all higher dimensions (1, 2, ..., n-1) and does not have any channels to cross from these dimensions. Therefore, the channel holding time for an h-hop message at its i^{th} step in dimension *zero* consists of the message length M, the routing time $A^j_{h,i}$ and the blocking delay along the subsequent effective channels remaining in dimension *zero*. However, a message at dimension j ($j > 0$) may still need to cross some channels from dimensions (j-1, j-2, ..., 1, 0) to get to its destination. Hence, the channel holding time for an h-hop message at its i^{th} step in dimension j ($j > 0$) consists of the message length M, the routing time $A^j_{h,i}$, and the blocking delay along the subsequent effective channels in all dimensions that the message still needs to visit.

To clarify the present discussion, Figure 4.3 depicts a message being transmitted over a 2D torus. The channel holding times of an h-hop message is given by

$$T^0_{h,h+1} = M \tag{4.24}$$

$$T^0_{h,i} = M + A^0_{h,i} + \sum_{l=i}^{i+A^{0,0}_{h,i}-1} B^0_{h,l} \tag{4.25}$$

$$T_{h,i}^{1} = M + A_{h,i}^{1} + \sum_{l=i}^{i+A_{h,i}^{1,1}-1} B_{h,l}^{1} + \sum_{l=i+A_{h,i}^{1,1}}^{i+A_{h,i}^{1,0}+A_{h,i}^{1,1}-1} B_{h,l}^{0} \tag{4.26}$$

Similarly, if the message is transmitted over a 3D torus, the channel holding times of an h-hop message in the third dimension is given by

$$T_{h,i}^{2} = M + A_{h,i}^{2} + \sum_{l=i}^{i+A_{h,i}^{2,2}-1} B_{h,l}^{2} + \sum_{l=i+A_{h,i}^{2,2}}^{i+A_{h,i}^{2,1}+A_{h,i}^{2,2}-1} B_{h,l}^{1} + \sum_{l=i+A_{h,i}^{2,1}+A_{h,i}^{2,2}}^{i+A_{h,i}^{2,0}+A_{h,i}^{2,1}+A_{h,i}^{2,2}-1} B_{h,l}^{0} \tag{4.27}$$

The above equations can be generalised for any $n \geq 1$ as

$$T_{h,i}^{j} = M + A_{h,i}^{j} + \sum_{d=0}^{j} \left(\sum_{m=i+A_{h,i}^{j,d+1}+A_{h,i}^{j,d+2}+\cdots+A_{h,i}^{j,j}}^{i-1+A_{h,i}^{j,d}+A_{h,i}^{j,d+1}+\cdots+A_{h,i}^{j,j}} \left(B_{h,m}^{d} \right) \right) \tag{4.28}$$

4.4.2.3 Blocking due to contention for virtual channels ($\theta_{h,i}^{j}$)

To compute the probability of blocking due to busy virtual channels, $\theta_{h,i}^{j}$, two cases should be considered, **(a)** V virtual channels are busy, which mean all virtual channels in sets VC_1 and VC_2 are busy, and **(b)** V-1 virtual channels are busy, in which only one combination out of a total of V combinations results in all virtual channels in VC_1 being busy and the virtual channel to be used from VC_2 (the escape channel) is also busy [125]. Let $P_v^{T_{h,i}^{j}}$ denote the probability that v virtual channels are busy at the i^{th} step of an h-hop message (with channel holding time of $T_{h,i}^{j}$). Recall that Chapter 3 has proposed a new method for calculating $P_v^{T_{h,i}^{j}}$. Considering the above two cases, $\theta_{h,i}^{j}$ can be calculated as [125]

$$\theta_{h,i}^{j} = P_V^{T_{h,i}^{j}} + \frac{P_{V-1}^{T_{h,i}^{j}}}{V} \tag{4.29}$$

4.4.2.4 Blocking due to the finite buffer ($\Upsilon_{h,i}^{j}$)

As illustrated in Figure 4.1, a message may be blocked inside the buffer if the header flit can not reach the head of the buffer because it is occupied by other messages. This can happen if

there is at least one flit from the previous message in the buffer, other than the one being serviced. To calculate this probability, we utilise some results from a finite capacity queue, denoted as M/G/1/F+1 (F flits from the finite buffer and one flit being serviced). The probability of blocking inside the finite buffer resembles the probability of (2, 3, ..., F+1) customers in the M/G/1/F+1 queue. Therefore, the probability that an h-hop message gets blocked inside the buffer of its i^{th} step at dimension j, $\Upsilon_{h,i}^{j}$, can be approximated as

$$\Upsilon_{h,i}^{j} = \sum_{\kappa=2}^{F+1} \pi_{\kappa}^{M/G/1/F+1} \tag{4.30}$$

In the above equation, $\pi_{\kappa}^{M/G/1/F+1}$ $(1 < \kappa \leq F+1)$, is the probability that there are κ customers in the M/G/1/F+1 queue. The arrival rate to the queue is the effective traffic received at each physical channel and is given by equation (4.17). Likewise, the service rate of the queue is the holding time of the next channel (this is because the input buffer is in the next router) and is given by equation (4.28).

The calculation of the probability of the number of customers in an M/G/1/F+1 queuing system, $P_{\kappa}^{M/G/1/F+1}$, has been reported in [77, 141] and we refer interested readers to these references for a detailed explanation. Nevertheless, it should be mentioned that the capacity of the M/G/1/F+1 queue is expressed in terms of flits, namely the capacity is F+1 flits. However, the arrival rate (equation 4.17) and the service rate (equation 4.28) are expressed in terms of messages. That is, λ_c messages arrive during one network cycle and it takes $T_{h,i}^{j}$ network cycles to service a message. For the above approximation to work properly, the arrival rate and the service time should be expressed in terms of flits not messages. Since the message size is fixed, the flit arrival rate, $\hat{\lambda}_c$, and flit service rate, $\hat{T}_{h,i}^{j}$, that should be used in calculating $\pi_{\kappa}^{M/G/1/F+1}$ can be expressed in terms of λ_c and $T_{h,i}^{j}$ as

$$\hat{\lambda}_c = \lambda_c M \tag{4.31}$$

$$\hat{T}_{h,i}^{j} = T_{h,i}^{j} / M \tag{4.32}$$

4.4.3 Waiting time at the source ($\bar{W}_s$)

The mean waiting time at the source node, $\bar{W}_s$ is calculated by modelling the local queue at the source node as an *M/G/*1 queue. Each PE generates λ_g messages per cycle which can be injected into the network through any of the V virtual channels of the injection channel. Hence, the mean arrival rate at the source is given by

$$\lambda_s = \lambda_g / V \tag{4.33}$$

Previous analytical models (i.e. models that assume single-flit buffers) have used the mean network latency, S_h, as an approximation for the service rate of the queue at the source. However, when finite buffers are deployed, this approximation is not applicable. This is because the service time of the ejection channels (also all other channels as have been detailed above) is affected only by a given number of channels as the message does not necessarily span over all channels between the source and destination nodes. Therefore, the service rate is equal to the channel holding time of the injection channel, $T_{h,0}^{j}$, as illustrated in Figure 4.3, and is given by equation (4.28). The variance of the service time distribution can be approximated as $(T_{h,0}^{j} - M)^2$ [45], and, hence, the mean waiting time at the source node of an h-hop message, W_s^h, is given by

$$W_s^h = \frac{\lambda_s (T_{h,0}^{j})^2}{2\left(1 - \lambda_s T_{h,0}^{j}\right)} \left(1 + \frac{\left(T_{h,0}^{j} - M\right)^2}{(T_{h,0}^{j})^2} \right) \tag{4.34}$$

Averaging over all possible hops gives the mean waiting time at the source node as

$$\bar{W}_s = \sum_{h=1}^{H} P_h^{hop} . W_s^h \tag{4.35}$$

4.4.4 Virtual channel multiplexing ($\bar{V}$)

When multiple virtual channels are used per physical channel, they share the bandwidth in a time-multiplexed manner [36, 42]. The average degree of virtual channel multiplexing of an h-hop message that takes place at a given physical channel is given by [36]

$$V_h = \frac{\sum_{v=1}^{V} v^2 P_v^{S_h}}{\sum_{v=1}^{V} v P_v^{S_h}} \tag{4.36}$$

Averaging over all hops, the average degree of virtual channel multiplexing in the network is written as

$$\bar{V} = \sum_{h=1}^{H} P_h^{hop} V_h \tag{4.37}$$

4.4.5 Model Implementation

The model can be translated easily to a computer program using any programming language that permits double precision declarations like C++. However, it can be seen that there are several interdependencies between different variables of the model. For example, equation (4.28) reveals that the channel holding time, $T_{h,i}^{j}$, is a function of the blocking delay $B_{h,i}^{j}$, which is a function of the mean waiting time, $W_{h,i}^{j}$. The mean waiting time, on the other hand, is expressed in terms of the channel holding time in equation (4.19). Therefore, iterative techniques are used to evaluate different model variables as a closed-form solution is not feasible due to the interdependencies between the model equations [66, 77]. The following procedure is used to compute the mean message latency using the proposed model.

Step 1: Read M, F, λ_g, k and n.

Step 2: Calculate the probabilities of generating h-hop messages P_h^{hop} $(1 \le h \le H)$ using equations (4.5) and (4.6)

Step 3: Compute λ_c using equation (4.17)

Step 4: For every possible combination, $C_c^h = (d_c^0, d_c^1, \ldots, d_c^{n-1})$, do the following

Step 4.1: Initialise $T_{h,i}^{j} = M + A_{h,i}^{j}$

Step 4.2: Calculate $B_{h,i}^{j}$ using equation (4.15)

Step 4.3: Compute the new $T_{h,i}^{j}$ using equation (4.28)

Step 4.4: Repeat Steps 4.2 and 4.3 until the difference between the new and the previous $T_{h,i}^{j}$ is less than a given error bound, ε

Step 5: Compute S_h, W_s^h and V_h using equations (4.11) , (4.34) and (4.36)

Step 6: Repeat steps 2, 3, 4 and 5 for every possible value of h

Step 7: Calculate $\bar{S}$, $\bar{W}_s$ and $\bar{V}$ using equations (4.4) , (4.35) and 4.37)

Step 8: Compute the mean message latency using equation (4.1)

Running the above iterative procedure in any conventional computer takes only a few seconds to produce entire curves representing the model predictions in Figures 4.4 - 4.9. That time is far less than the time required to produce a single simulated point in the same graphs.

4.5 Model validation

To investigate the accuracy of the proposed analytical model, the discrete-event simulator that mimics the behaviour of dimension-order routing at the flit level in the wormhole-switched k-ary n-cube has been used. The network cycle time is defined as the transmission time of a single flit from one router to the next. Nodes generate fixed size messages (M flits each) randomly with exponentially distributed inter-arrival times. The destination node of an h-hop message is 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 time to cross the network and the mean time spent in the local queue of the source node.

The Batch mean method [42, 95] has been used to collect simulation output where in each simulation experiment, a total number of 150000 messages (divided into 30 batches) are delivered to their destinations. In order to avoid the distortions due to start-up conditions, the

first 10000 messages were ignored. Along with the mean message latency, 95% confidence intervals have also been obtained from the simulations. However, due to the commonly small *confidence interval half-widths* especially in low and moderate traffic regions, and for clarity of presentation, the figures presented, like previous studies (for example [2, 4, 5, 29, 30, 45, 62, 73, 125]), depict only the mean message latency.

Numerous validation experiments have been conducted, for several combinations of network sizes, message sizes, virtual channels and buffer sizes, to assess the accuracy of the proposed analytical model. However, validation results are presented for the following cases

- Network size is 8-ary 2-cube, 16-ary 2-cube and 8-ary 3-cube;
- Message length M=16, 32 and 64 flits;
- Number of virtual channels V=3 and 5;
- Buffer size F=2, 4, 8, and 16 flits.

Figures 4.4-4.9 depict latency results predicted by the analytical model (labelled as *Model*) plotted against those obtained from the simulator (labelled as *Sim*). The horizontal axis in all the figures shows the traffic generation rate at each node per network cycle (i.e. λ_g) while the vertical axis shows the mean message latency. The figures indicate that in all cases the simulation measurements and the values predicted by the analytical model are in close agreement for various network operating environments. 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 point (when the network is no longer able to deliver any message to its destination). From the modelling point of view, the saturation point occurs when the server utilisation reaches one, that is $\lambda_c T_{h,i}^j \geq 1$.

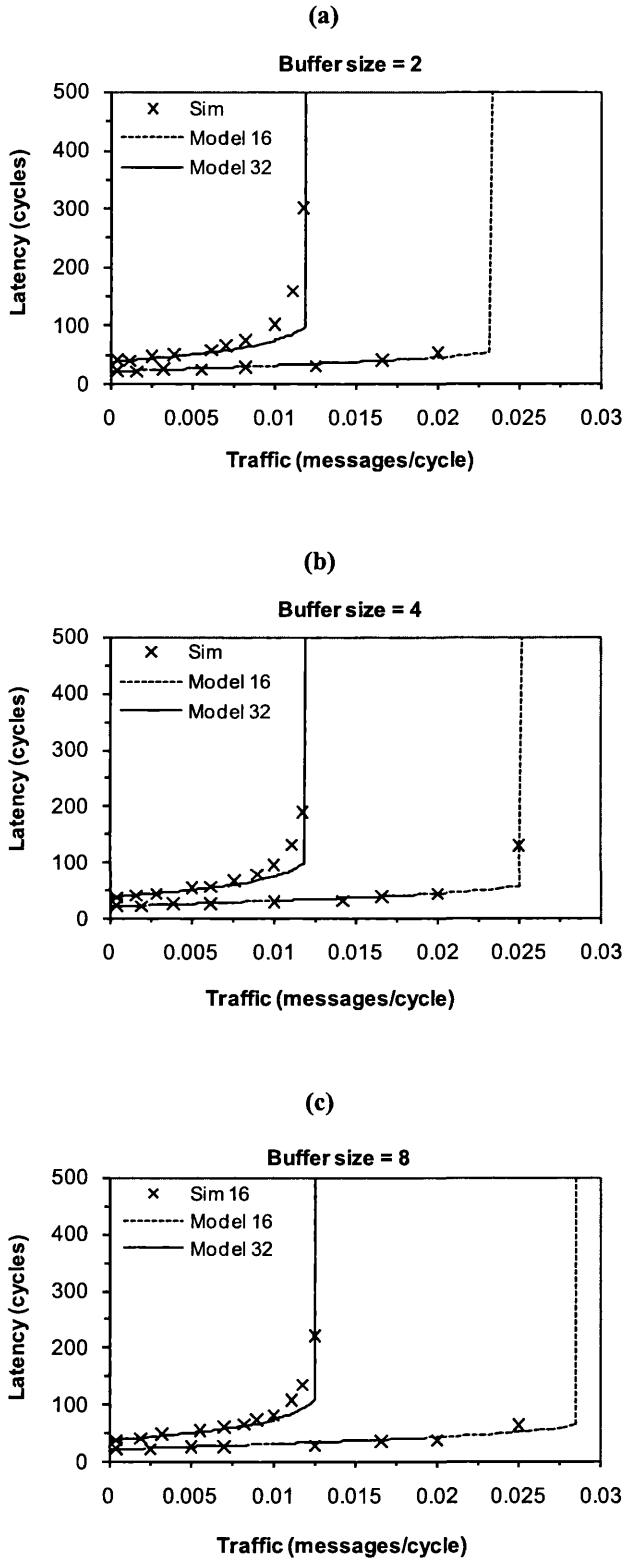


Figure 4.4: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 2-cube. Message length M=16 and 32 flits; number of virtual channels per physical channel V= 3. Buffer size (a) F=2 flits, (b) F=4 flits, and (c) F= 8 flits.

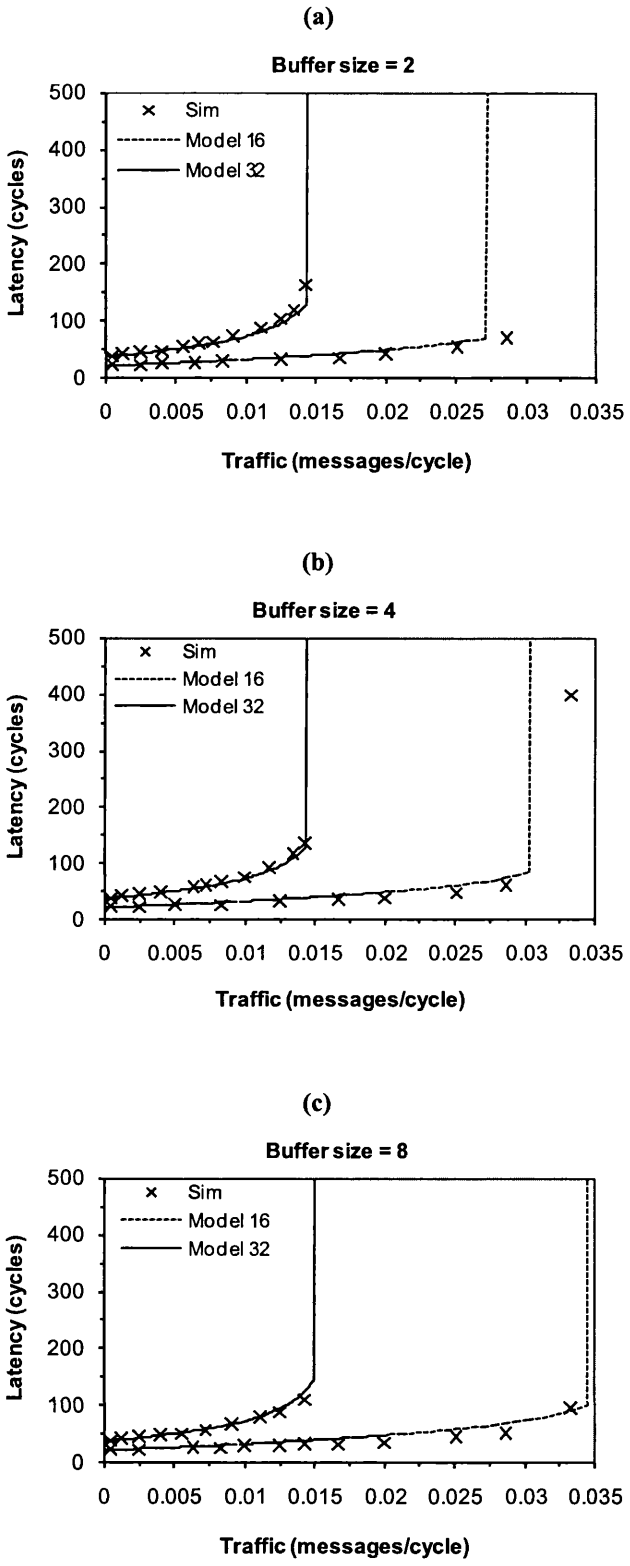


Figure 4.5: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 2-cube. Message length M=16 and 32 flits, number of virtual channels per physical channel V=5. Buffer size (a) F=2 flits, (b) F=4 flits, and (c) F= 8 flits.

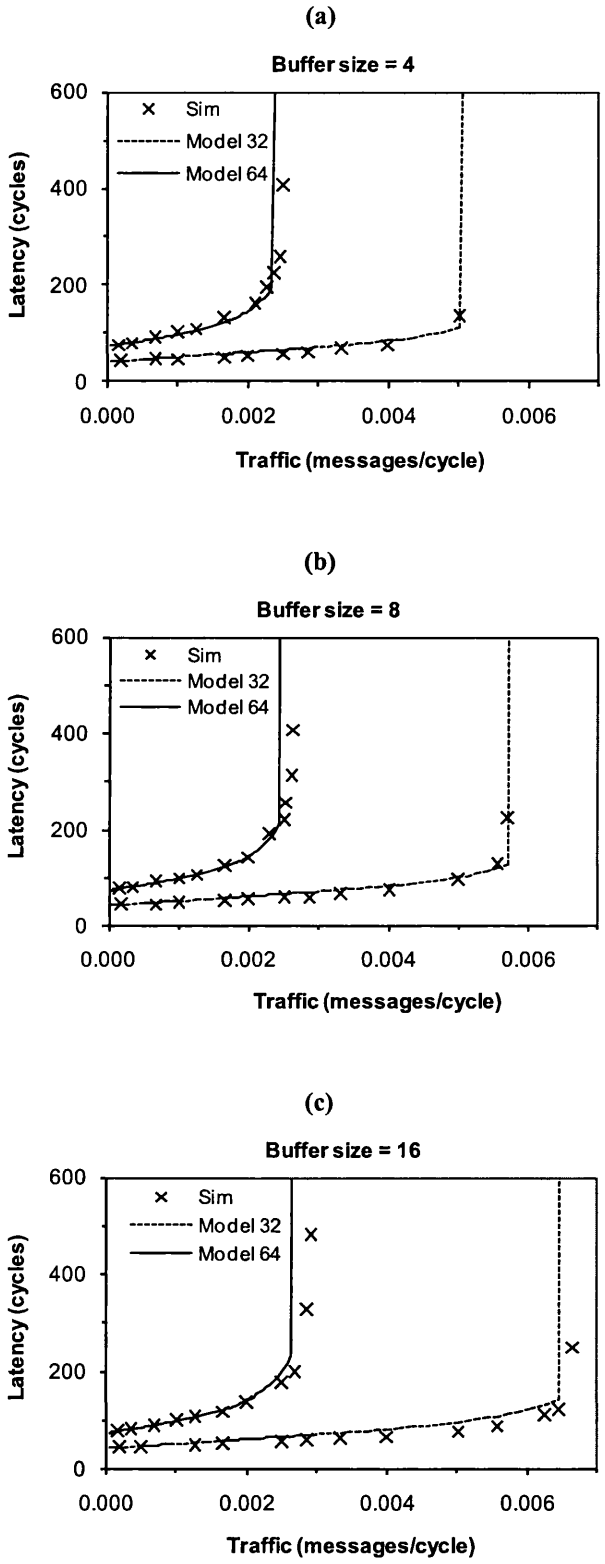


Figure 4.6: Mean message latency predicted by the model and simulator against the traffic generation rate in the 16-ary 2-cube. Message length M=32 and 64 flits, number of virtual channels per physical channel V= 3. Buffer size (a) F=4 flits, (b) F=8 flits, and (c) F= 16 flits.

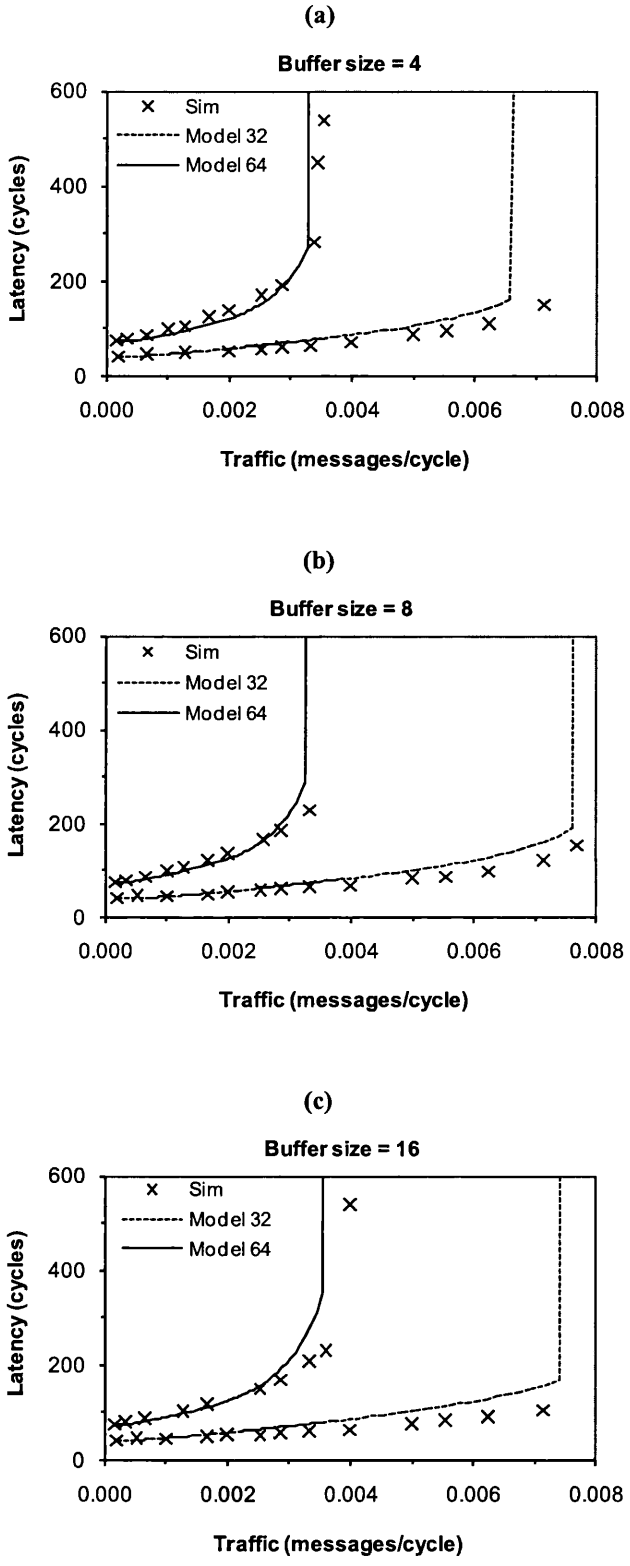
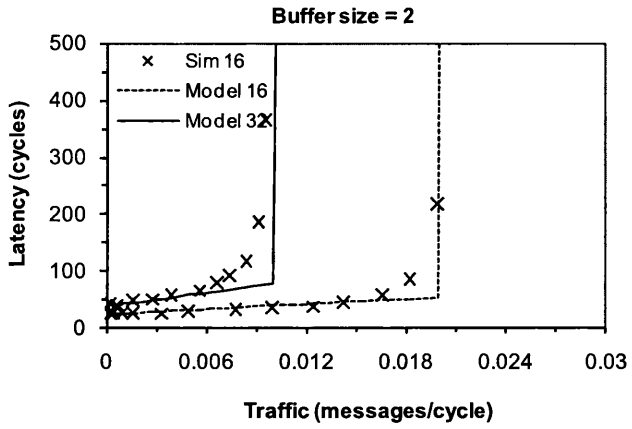
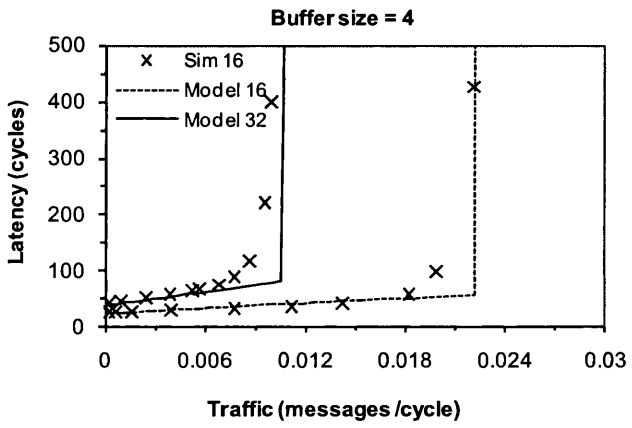


Figure 4.7: Mean message latency predicted by the model and simulator against the traffic generation rate in the 16-ary 2-cube. Message length M=32 and 64 flits, number of virtual channels per physical channel V= 5. Buffer size (a) F=4 flits, (b) F=8 flits, and (c) F=16 flits.

(a)



(b)



(c)

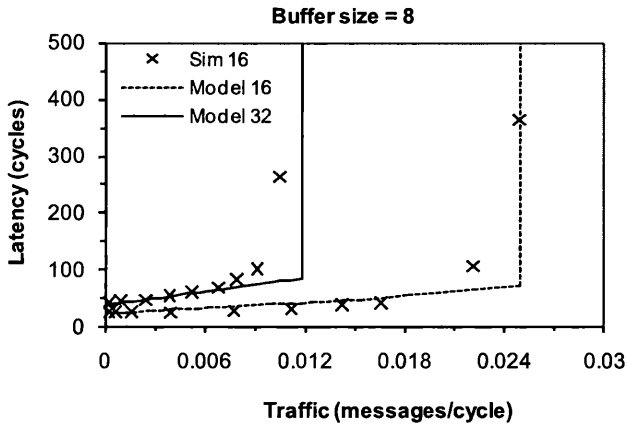


Figure 4.8: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 3-cube. Message length M=16 and 32 flits, number of virtual channels per physical channel V= 3. Buffer size (a) F=2 flits, (b) F=4 flits, and (c) F= 8 flits.

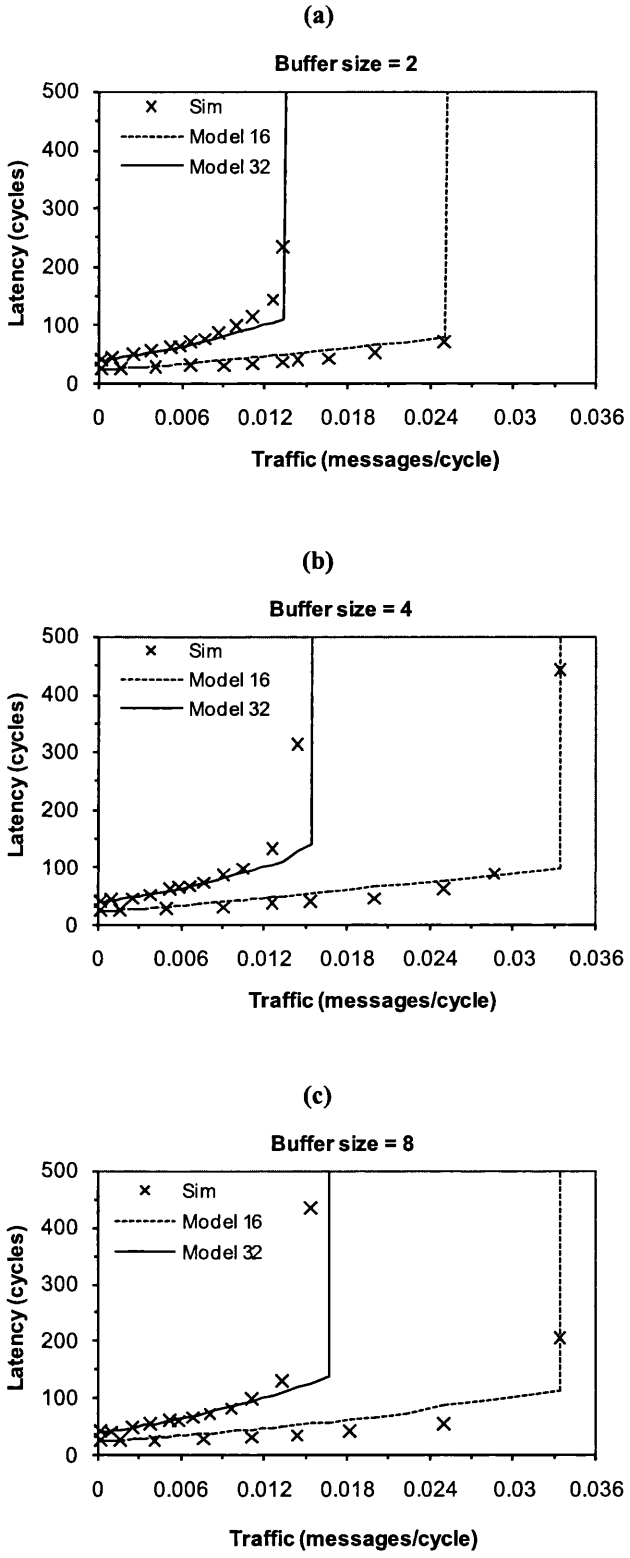


Figure 4.9: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 3-cube. Message length M=16 and 32 flits, number of virtual channels per physical channel V= 5. Buffer size (a) F=2 flits, (b) F=4 flits, and (c) F= 8 flits.

However, in some cases, some discrepancies around the saturation point are apparent. These can be accounted for, for instance, by the approximation made to estimate the variance of the message service time distribution. As mentioned above, this approximation simplifies greatly the model, as it allows us to avoid computing the exact distribution of the message service time; which is not a straightforward task due to the inter-dependencies between service times at successive channels as wormhole switching 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 of *k*-ary *n*-cubes.

4.6 Extension of the model

This section outlines briefly the modifications that have to be made to the model in order to adapt it to the unidirectional torus and hypercube.

4.6.1 The unidirectional torus

In the unidirectional torus, an *h*-hop message can make at most *k*-1 hops per dimension. Similarly, the minimum number of hops at any dimension is *zero* if *h* is less than *k*-1. Otherwise, the minimum number of hops at any dimension is equal to the number of hops that are left after making the maximum number of hops in (*n*-1) dimensions. Therefore, H^+ and H^- for the unidirectional torus are respectively given by

$$H^+ = \begin{cases} k-1 & h > k-1 \\ h & h \le k-1 \end{cases} \tag{4.38}$$

$$H^- = \begin{cases} h-(k-1)(n-1) & h > (k-1)(n-1) \\ 0 & h \le (k-1)(n-1) \end{cases} \tag{4.39}$$

Therefore, the maximum number of hops between any given pair of nodes in the network is given by

$$H = n(k-1) \tag{4.40}$$

Since the router in the unidirectional torus has n output channels (as opposed to $2n$ output channels in the bidirectional torus), then the mean traffic rate received by each channel is obtained by modifying equation (4.17) to reflect the above changes. Hence, λ_c of the unidirectional torus can be written as

$$\lambda_c = \sum_{h=1}^{H}\left(P_h^{hop} h \frac{N\lambda_g}{nN}\right) = \frac{\lambda_g}{n}\sum_{h=1}^{H}\left(P_h^{hop} h\right) \tag{4.41}$$

Finally, the number of nodes that are h-hops away from a given node, n_h, in the unidirectional torus is given by [131]

$$n_h = \sum_{i=0}^{n}(-1)^i \binom{n}{i}\binom{h-ik+n-1}{n-1} \tag{4.42}$$

4.6.2 The hypercube

When the number of nodes per dimension is 2, the k-ary n-cube collapses to the well known hypercube topology [49]. To model the hypercube, some equations derived above can be further simplified due to the fact that the maximum number of hops per dimension is 1 (i.e. each dimension contains only two nodes). The probability of generating an h-hop message in hypercubes is given by [1]

$$P_h^{hop} = \frac{\binom{n}{h}}{N-1} \tag{4.43}$$

Making one hop in the hypercube means crossing one dimension. Therefore, the maximum number of hops that a message needs to reach to any destination is n. This simplifies the calculation of the network latency and the channel holding time. The network latency can now be written as [8]

$$S_h = M + h + \sum_{i=1}^{h} B_{h,i}^{i} \tag{4.44}$$

Similarly, the channel holding time of an h-hop message at its i^{th} step (i.e. i^{th} dimension) is given by [8]

$$T_{h,i}^{i} = M + A_{h,i}^{i} + \sum_{l=i}^{i+A_{h,i}^{i}-1} B_{h,l}^{l} \tag{4.45}$$

Also, the mean waiting time at the source of an h-hop message is given by

$$W_{s}^{h} = \frac{\lambda_{s}\left(T_{h,0}^{0}\right)^{2}}{2\left(1-\lambda s T_{h,0}^{0}\right)}\left(1+\frac{\left(T_{h,0}^{0}-M\right)^{2}}{\left(T_{h,0}^{0}\right)^{2}}\right) \tag{4.46}$$

Because dimension-order routing in the hypercube is deadlock-free, the virtual channels ($V \geq 1$) are not partitioned into two sets like the torus (i.e. there is no need to dedicate any virtual channels for deadlock-prevention) [49]. Hence, the probability of blocking due to the contention for virtual channels is equal to the probability that all virtual channels are busy (i.e. $P_{V}^{T_{h,i}^{i}}$). Therefore, the mean blocking delay becomes

$$B_{h,i}^{i} = P_{V}^{T_{h,i}^{i}} \cdot \Upsilon_{h,i}^{i} \cdot W_{h,i}^{i} \tag{4.47}$$

4.7 Conclusions

This chapter proposed a new analytical model that captures the effects of finite buffers on the performance of wormhole-switched k-ary n-cubes when deterministic routing is used. This is the first model to be reported in the literature that considers the use of multiple virtual channels per physical channel. The derivation of the model has been described in detail for the bidirectional torus. However, modifications to the model in order to extend it to the unidirectional torus and hypercube have also been outlined. The model has been validated via simulation experiments and the results have demonstrated its reasonable accuracy under various network operating conditions. The simplicity and the accuracy of the proposed model make it a practical and cost-effective evaluation tool that can be used to study the performance impacts of introducing finite buffers in wormhole-switched k-ary n-cubes.

Next chapter presents an analytical model that can capture the effects of finite buffers when adaptive routing is used in wormhole switched networks. More precisely, a new model is proposed for Duato's adaptive routing algorithm [46], which has been widely studied and deployed in practical multicomputers [3, 22, 38, 49, 103, 111, 127, 133].

Chapter 5

Modelling Adaptive Routing with Finite Buffers

5.1 Introduction

A number of researchers have proposed analytical models to assess the performance of adaptive routing in k-ary n-cubes over the past few years [22, 71, 93, 102, 111, 126, 127, 128, 129]. However, like the models proposed for deterministic routing, these models have also constrained the buffer depth to a single flit. Consequently, most studies have resorted to simulations to capture the effects of finite buffers on the performance of adaptive routing in k-ary n-cubes [14, 26, 48, 101].

The present chapter suggests a new model to capture the effects of finite size buffers on network performance when adaptive routing is used in k-ary n-cubes with virtual channels. The routing algorithm that has been used in the development of the new model is Duato's adaptive routing algorithm [46] but the modelling approach can be applied to other adaptive routing algorithms in a straightforward manner [93, 111, 126].

The previous chapter, and also some existing studies [66, 81, 82], described analytical models for deterministic routing with finite buffers. However, the modelling approach adopted for deterministic routing cannot be applied directly to adaptive routing. This stems from the inherently different operations of these two types of routing algorithms. Moreover, developing such a model for adaptive routing is a more difficult undertaking. In adaptive routing, when a message reaches a given router, it is not clear how many

dimensions it still needs to traverse; as the message path is not predefined as in the deterministic routing. The problem is further exacerbated as the number of network dimensions increases. As a result, a different approach is required for computing the blocking probabilities in adaptive routing. The presence of the finite buffers complicates the derivation further as the statistical characteristics of each channel, even in the same dimension, could vary depending on the number of buffers that are necessary to accommodate an entire message.

It is shown below that the proposed model exhibits good accuracy under light, moderate and heavy traffic conditions. It achieves this because, on the one hand, it takes account of the two types of blocking that may affect the message transmission; one is due to the contention for virtual channels and the other is due to the existence of the finite buffers. On the other hand, the model determines different components that make up the average message latency including routing delay, blocking delay and source delay as a function of the number of effective channels. This is to account for the reduction in the number of channels that are occupied by a message due to the available buffer space.

The remainder of this chapter is organised as follows. Section 5.2 lists the assumptions and notation used in the analysis. Section 5.3 presents the analytical model for the bidirectional torus. Section 5.4 validates the model through simulation experiments while Section 5.5 shows how the model can be adapted to model the unidirectional torus and hypercube. Finally, Section 5.6 concludes this chapter.

5.2 Assumptions and notation

The notation used to describe different variables and quantities of the model are listed in Table 5.1. Moreover, the model presented in this chapter relies on the following assumptions that are commonly accepted in the literature [22, 66, 71, 81, 82, 93, 102, 111, 126].

a) The message length is M flits, each of which requires one network cycle to advance from one router to the next using wormhole switching. Moreover, message destinations are distributed uniformly across the network nodes.

b) Each node generates traffic independently of all other nodes according to a Poisson process with a mean generation rate of λ_g messages per cycle.

c) The arrival process at a given channel is approximated by an independent Poisson process with a mean arrival rate of λ_c messages per cycle.

d) Each physical channel is organised into V (V>2) virtual channels each of which is equipped with a finite buffer of size F ($1 < F \leq M$) flits.

e) Messages are routed according to Duato's adaptive routing algorithm [46], where the virtual channels are divided into two sets: VC_1 and VC_2. The set VC_1 contains (V-2) virtual channels that are crossed adaptively. When there is more than one virtual channel available, then a message chooses one at random. Meanwhile, the set VC_2 contains two virtual channels that are crossed deterministically. Only one of these two channels can be used in an increasing (or decreasing) order of dimension. Duato's adaptive routing is explained in more detail in Chapter 2.

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

Table 5.1: Notation used in the model for adaptive routing with finite buffers

Symbol	Description
s	Source node
d	Destination node
M	Message size in flits
F	Buffer size in flits
n	Number of dimensions of the network
k	Number of nodes per dimension

$\bar{k}$	Average number of hops to cross one dimension
$\bar{d}$	Average number of hops to across the entire network
λ_g	Message generation rate at each node of the network
λ_c	Rate of messages received at each physical channel
$T_i^{<s,d>}$	Channel holding time at the i^{th} hop of a message travelling from s to d
$A_i^{<s,d>}$	Number of effective channels at the i^{th} hop of a message travelling from s to d
$\Im_{<s,d>}$	A set where each element represents the number of hops that a message makes at each dimension when travelling from s to d
$\left\|\Im_{<s,d>}\right\|$	Total number of hops between nodes s and d
δ	Number of elements in the set $\Im_{<s,d>}$
$\Omega\left(0, \Im_{<s,d>}, i\right)$	Number of ways to distribute i hops over δ dimensions with at least zero and at most $h_j \in \Im_{<s,d>}$ hops in the j^{th} dimension
$\hat{\Im}_{<s,d>}$	Temporary set used to calculate $\Omega\left(0, \Im_{<s,d>}, i\right)$
$W_s^{<s,d>}$	Source delay seen by a message travelling from s to d
$B^{<s,d>}$	Blocking delay seen by a message travelling from s to d
$V^{<s,d>}$	Multiplexing degree seen by a message travelling from s to d
$\bar{W}_s$	Mean source delay
R	Mean routing delay
B	Mean blocking delay
$\bar{V}$	Average degree of multiplexing
$\bar{S}$	Mean network latency
$S^{<s,d>}$	Average network latency of a message crossing from s to d
$W_i^{<s,d>}$	Mean waiting time for a message crossing from s to d
$\theta_i^{<s,d>}$	Probability of blocking due to contention at the i^{th} hop of a message travelling from s to d
$\Upsilon_i^{<s,d>}$	Probability of blocking due to insufficient buffer space at the i^{th} hop of a message travelling from s to d
$P_{i,a}^{<s,d>}$	Probability that all adaptive virtual channels at the physical channel of the i^{th} hop of a message crossing from s to d are busy
$P_{i,d}^{<s,d>}$	Probability that all adaptive virtual channels and the deterministic virtual channel to be used at the physical channel of the i^{th} hop of a message crossing from s to d are busy
$Pass_{i,z}^{<s,d>}$	Probability that a message crossing from s to d has passed z dimensions on its i^{th} hop.
$\pi_\kappa^{M/G/1/F+1}$	Probability that there are κ customers in the M/G/1/F+1 queue
$P_v^{T_i^{<s,d>}}$	Probability that there are v busy virtual channels when the channel holding time is $T_i^{<s,d>}$ at the i^{th} hop of a message crossing from s to d

5.3 Derivation of the model

It has been highlighted in Section 4.3 how introducing finite buffers complicates the derivation of the analytical model. These complications arise because finite buffers reduce the number of channels that a message occupies, and hence blocking delays inside the buffer and at the head of the buffer should both be considered and expressed as a function of the number of effective channels.

The time to transmit a message from a source node to a destination node can be broken into four main components. The first component is the time that the message spends waiting at the local queue of the injection channel before it traverses the first network channel. The second component is the actual message transmission time and is simply equal to the message size as we have assumed that each flit of the message takes one cycle to be transmitted across a physical channel from one node to the next. Upon arriving at an intermediate node, the header flit is decoded and the router decides which output channel it shall forward the message to. This routing decision at each hop constitutes the third component that contributes to the overall message latency. Finally, the fourth component of the message latency is the blocking delay that the message may encounter at each hop.

The proposed analytical model calculates the mean message latency as the summation of the above four components (i.e. the source delay $\bar{W}_s$, the transmission time M, the routing delay R, and the blocking delay B). However, the latency has to be scaled by a factor, $\bar{V}$, to capture the effects of multiplexing multiple virtual channels over a physical channel [36, 114, 126]. The mean message latency can therefore be expressed as

$$\textit{Mean Message Latency} = (\bar{S} + \bar{W}_s)\bar{V} \tag{5.1}$$

In the above equation, and in what follows, we will denote by $\bar{S}$ the mean network latency (i.e. the time to cross the network) which is the summation of the transmission time, routing delay and blocking delay. That is

$$\bar{S} = M + R + B \tag{5.2}$$

Since the torus is symmetric, $\bar{S}$ and $\bar{W}_s$ can be calculated by averaging the latencies seen by a message generated at one source node and destined to all other nodes in the network. Hence without any lose of generality, and to simplify the analysis, we can assume that the source node is $s = (s_0, s_1, \ldots, s_{n-1})$ where $s_j = 0 \ (0 \le j \le n-1)$. Let the destination node be denoted by $d = (d_0, d_1, \ldots, d_{n-1})$ such that $d \in G - \{s\}$, where G is the set of all nodes in the network. Let us also define the set $\Im_{<s,d>} = \{h_0, h_1, \ldots, h_{n-1}\}$, where each element, h_j, represents the number of hops that a message makes in each dimension, j, along its path from the source node s to the destination node d. Hence the total number of hops is given by

$$\left| \Im_{<s,d>} \right| = \sum_{j=0}^{n-1} h_j \tag{5.3}$$

In bidirectional networks, messages can be routed in both the positive and negative directions of any dimension. Given that the maximum number of hops a message can make to cross any dimension is $k/2$ (this is due to the wrap-around connections), and since $s_j = 0 \ (0 \le j \le n-1)$, the set $\Im_{<s,d>}$ can be constructed as

$$h_j = \begin{cases} d_j & d_j \le \lceil k/2 \rceil \\ k - d_j & otherwise \end{cases} \qquad ; j = 0, 1, \ldots, n-1 \tag{5.4}$$

Let us further denote the source delay and the blocking delay between a given source node s and a given destination node d as $W_s^{<s,d>}$ and $B^{<s,d>}$, respectively. Averaging over all N-1 possible destinations, $\bar{W}_s$ and B can, be written as follows

$$\bar{W}_s = \frac{\sum_{d \in G-\{s\}} W_s^{<s,d>}}{N-1} \tag{5.5}$$

$$B = \frac{\sum_{d \in G-\{s\}} B^{<s,d>}}{N-1} \tag{5.6}$$

However, the routing delay, R, of a message originated at node s and destined to node d is simply equal to the length of the route, $\left|\Im_{<s,d>}\right|$ (i.e. the number of hops that a message needs to make to reach its destination). This is because when wormhole switching is used, the routing decision is taken only for the header flit while all remaining data flits follow the header in a pipelined fashion. By assuming that each routing decision takes one network cycle, R can be written as

$$R = \frac{\sum_{d \in G-\{s\}} \left|\Im_{<s,d>}\right|}{N-1} \tag{5.7}$$

5.3.1 The blocking delay ($B^{<s,d>}$)

Because the destination nodes are distributed uniformly in the network (assumption **a**), the average number of hops that a message makes along one dimension, $\bar{k}$, and across the network, $\bar{d}$, in bidirectional k-ary n-cubes is given by [5]

$$\bar{k} = \begin{cases} k/4 & k \bmod 2 = 0 \\ \frac{1}{4}(k - (1/k)) & k \bmod 2 \neq 0 \end{cases} \tag{5.8}$$

$$\bar{d} = n\bar{k} \tag{5.9}$$

Duato's adaptive routing allows a message to use any available channel that brings it closer to its destination. This results in an evenly distributed traffic rate over all network channels. As discussed in Chapter 2, the router of the bidirectional k-ary n-cube has two output channels, corresponding to the positive and negative directions, for every dimension, resulting in a total of $2n$ output channels per router. Given that under uniform traffic each message traverses on average $\bar{d}$ hops to travel from its source to its destination and because each node generates λ_g messages per cycle, the average traffic received by each channel in the network, λ_c, can be written as

$$\lambda_c = (\lambda_g \bar{d})/2n \tag{5.10}$$

At each hop i $\left(1 \le i \le \left|\Im_{<s,d>}\right|\right)$ of its journey from node s to node d, a message may suffer from two types of blockings. Firstly, it may get blocked at the head of the buffer with probability $\theta_i^{<s,d>}$ due to the contention for virtual channels. Secondly, a message may be blocked inside the buffer with probability $\Upsilon_i^{<s,d>}$ due to insufficient buffer space to accommodate the entire message. If a message gets blocked during the i^{th} hop of its path, it then spends a mean waiting time of $W_i^{<s,d>}$. Therefore, the blocking delay of a message consists of the summation of the blocking delays that the message may encounter at each of its $\left|\Im_{<s,d>}\right|$ hops during its journey from node s to node d and is expressed as

$$B^{<s,d>} = \sum_{i=1}^{\left|\Im_{<s,d>}\right|} \left(\theta_i^{<s,d>} . \Upsilon_i^{<s,d>} . W_i^{<s,d>}\right) \tag{5.11}$$

To determine $W_i^{<s,d>}$, each network channel is treated as an M/G/1 queuing system with a mean waiting time of [77]

$$E[W] = \frac{\lambda_c E[S]^2 \left(1 + \frac{\sigma_S^2}{E[S]^2}\right)}{2\left(1 - \lambda_c E[S]\right)} \tag{5.12}$$

In the above equation, λ_c is the arrival rate, $E[S]$ is the mean service time and σ_S^2 is the variance of the service time distribution. While the arrival rate is given by equation (5.10), the mean and the variance of the service time are still to be evaluated. The mean service time of the queue is exactly the channel holding time of the i^{th} hop, denoted as $T_i^{<s,d>}$, and will be calculated in the following subsection.

However, calculating the variance of the service time requires explicate knowledge of the service time distribution which is hard to obtain due to the inherited blocking nature of wormhole switching. Following a suggestion proposed in [45], and because the minimum service time of any message is equal to its length, M, the variance of the service time

distribution can be approximated as $(T_i^{<s,d>} - M)^2$. Substituting $T_i^{<s,d>}$ for the mean service time and $(T_i^{<s,d>} - M)^2$ for the variance of the service time into equation (5.12) yields an expression for the mean waiting time

$$W_i^{<s,d>} = \frac{\lambda_c \left(T_i^{<s,d>}\right)^2 + \lambda_c \left(T_i^{<s,d>} - M\right)^2}{2\left(1 - \lambda_c T_i^{<s,d>}\right)} \tag{5.13}$$

5.3.1.1 The channel holding time ($T_i^{<s,d>}$)

Due to the available buffering space, the different components that make up the mean message latency are affected by only a limited number of channels (i.e. the number of effective channels). Using the definition given in Chapter 4, the number of effective channels can be calculated as

$$A_i^{<s,d>} = \begin{cases} \lceil M/F \rceil & \left|\Im_{<s,d>}\right| - i + 1 \geq \lceil M/F \rceil \\ \left|\Im_{<s,d>}\right| - i + 1 & \textit{otherwise} \end{cases} \tag{5.14}$$

In the above equation, $\lceil M/F \rceil$ denote the number of buffers that are necessary to accommodate the entire message. The ceiling operator is required because the message size may not be exact multiples of the buffer size.

When adaptive routing is used in a uniform traffic model, we can assume safely that the service time (i.e. the channel holding time) is the same for all network channels, as adaptive routing distributes traffic evenly among all channels. However, when finite size buffers are introduced, channels (regardless of their dimension) will see different holding times because the numbers of effective channels vary from one hop to the next. As a result, we need to consider different channel holding times for every hop a message makes along its path from the source to the destination. Taking into consideration the effective number of channels, the channel holding time of the i^{th} hop consists of the actual message transmission time, M, the routing delay that occurs along the effective channels (we have assumed that routing

decision takes one cycle), and the blocking delays encountered along the subsequent effective channels. Hence, the channel holding time can be written as

$$T_i^{<s,d>} = M + A_i^{<s,d>} + \sum_{l=i}^{i+A_i^{<s,d>}-1} \left(\theta_l^{<s,d>} . \Upsilon_l^{<s,d>} . W_l^{<s,d>} \right) \tag{5.15}$$

5.3.1.2 Probability of blocking due to contention ($\theta_i^{<s,d>}$)

The calculation of the probability of blocking due to contention for virtual channels, $\theta_i^{<s,d>}$, has been reported widely in the literature (see for example [22, 71, 93, 102, 106, 111, 126]). The method proposed in [126] is based on some topological properties of *k*-ary *n*-cubes [131] and has been shown to preserve a good degree of accuracy. In the rest of this section, we adopt a similar approach to compute $\theta_i^{<s,d>}$.

Let δ denote the number of elements in the set $\Im_{<s,d>}$. Also, let $\Omega\left(0, \Im_{<s,d>}, i\right)$ denote the number of ways to distribute i hops over δ dimensions such that the number of hops in dimension j, is at least zero and is at most the j^{th} element of the set $\Im_{<s,d>}$. $\Omega\left(0, \Im_{<s,d>}, i\right)$ can be determined recursively using[1]

$$\Omega\left(0, \Im_{<s,d>}, i\right) = \begin{cases} 1 & i = 0 \\ 0 & i < 0 \; or \; \delta < 1 \\ \sum_{m=0}^{h_\delta} \Omega\left(0, \Im_{<s,d>} - \{h_\delta\}, i-m\right) & otherwise \end{cases} \tag{5.16}$$

The probability that a message has crossed entirely z (z=1, 2, ..., n-1) dimensions at its i^{th} hop is given by [126]

$$Pass_{i,z}^{<s,d>} = \frac{\sum_{l_1=0}^{n-1} \sum_{l_2=1}^{n-1} \cdots \sum_{l_z=z}^{n-1} \Omega\left(0, \hat{\Im}_{<s,d>}, i - \sum_{j=1}^{z} h_{l_j}\right)}{\Omega\left(0, \Im_{<s,d>}, i\right)} \tag{5.17}$$

[1] A detailed proof is presented in [131]

In the above equation, $\hat{\Im}_{<s,d>}$ is a temporary set and is constructed as follows [126]

$$\hat{\Im}_{<s,d>} = \{\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_{n-1}\}$$

$$\hat{h}_i = \begin{cases} 0 & i = l_1 \text{ or } i = l_2 \text{ or } \ldots \text{ or } i = l_z \\ h_1 - 1 & \text{otherwise} \end{cases} \tag{5.18}$$

Using the law of total probability [108], the probability that the message still has to visit all dimensions is given by

$$Pass_{i,0}^{<s,d>} = 1 - \sum_{z=0}^{n-1} Pass_{i,z}^{<s,d>} \tag{5.19}$$

When a message has crossed entirely z dimensions, it can select any of the available $(n\text{-}z)(V\text{-}2)$ adaptive virtual channels from VC_1 to make its next hop. If all virtual channels from VC_1 are busy, the message can use the predefined deterministic virtual channel from VC_2 to advance to the next router otherwise the message gets blocked. Let $P_{i,a}^{<s,d>}$ denotes the probability that a message cannot use the virtual channels from VC_1 (i.e. all adaptive virtual channels are busy) at the i^{th} hop of its journey from node s to node d. Similarly, let $P_{i,d}^{<s,d>}$ denotes the probability that a message cannot use the predefined deterministic virtual channel from VC_2 (i.e. all adaptive virtual channels and the deterministic virtual channel to be used are busy).

A message is blocked at its i^{th} hop, if all the adaptive virtual channels of the remaining dimensions are busy and the deterministic virtual channel to be used is busy. Hence, the probability of blocking at the head of the queue due to the contention for virtual channels can be written as [126]

$$\theta_i^{<s,d>} = \sum_{z=0}^{n-1} \left[Pass_{i,z}^{<s,d>} \left(P_{i,a}^{<s,d>} \right)^{n-z-1} P_{i,d}^{<s,d>} \right] \tag{5.20}$$

To compute $P_{i,a}^{<s,d>}$, the following three cases should be considered [111]

a) V virtual channels are busy, which means all adaptive and all deterministic virtual channels are busy.

b) (V-1) virtual channels are busy. There is a total of V combinations where (V-1) out of V virtual channels are busy. Among these combinations only two cases results in all adaptive virtual channels being busy. The first case is when all adaptive virtual channels are busy and the *first* virtual channel from VC_2 is busy. The second case is when all adaptive virtual channels are busy and the *second* virtual channel from VC_2 is busy.

c) (V-2) virtual channels are busy. The number of combinations where (V-2) out of V virtual channels are busy is $\binom{V}{V-2}$ of which only one case results in all virtual channels being busy. This case occurs when all virtual channels are busy, but the two deterministic virtual channels from VC_2 are not.

Likewise, to compute the probability that all adaptive virtual channels and the deterministic virtual channel to be used are busy, $P_{i,d}^{<s,d>}$, two cases should be considered [111]

a) V virtual channels are busy which means all adaptive and all deterministic virtual channels are busy.

b) (V-1) virtual channels are busy. In this case only two combinations results in all adaptive virtual channels being busy and the deterministic virtual channel to be used also busy.

The probability of busy virtual channels per physical channel has been discussed extensively in Chapter 3. Equation (3.9) gives the probability of v busy virtual channels, $P_v^{T_i^{<s,d>}}\left(v=0,1,\ldots,V\right)$, when the channel holding time is $T_i^{<s,d>}$ at the i^{th} hop of a message travelling from s to d. Taking into account the cases mentioned above, $P_{i,a}^{<s,d>}$ and $P_{i,d}^{<s,d>}$ are given, in terms of $P_v^{T_i^{<s,d>}}$, by [111, 126]

$$P_{i,a}^{<s,d>} = P_V^{T_i^{<s,d>}} + \frac{2P_{V-1}^{T_i^{<s,d>}}}{\binom{V}{V-1}} + \frac{P_{V-2}^{T_i^{<s,d>}}}{\binom{V}{V-2}} \tag{5.21}$$

$$P_{i,d}^{<s,d>} = P_V^{T_i^{<s,d>}} + \frac{2P_{V-1}^{T_i^{<s,d>}}}{\binom{V}{V-1}} \tag{5.22}$$

5.3.1.3 Probability of blocking due to finite buffers ($\Upsilon_i^{<s,d>}$)

The calculation of $\Upsilon_i^{<s,d>}$ has been explained in Chapter 4 and the same approach is used here. This is because, in both deterministic and adaptive routing algorithms, the routing decision is taken only for the header flit when it reaches the head of the queue. Therefore, the probability of blocking inside the finite buffer, of a message at its i^{th} hop is given by

$$\Upsilon_i^{<s,d>} = \sum_{\kappa=2}^{F+1} \pi_\kappa^{M/G/1/F+1} \tag{5.23}$$

5.3.2 The source delay ($W_s^{<A,B>}$)

A message that makes $\left|\Im_{<s,d>}\right|$ hops to travel between node s and node d, may experience queuing delay at the injection channel before it is delivered to the network for transmission. To calculate the mean waiting time at the source node, the injection channel can be modelled as an M/G/1 queuing system with mean arrival rate (λ_g / V) (i.e. a message at the source node may enter the network via any of the V virtual channels). The service time of the queue is the channel holding time of the injection channel, $T_0^{<s,d>}$, which can be calculated using equation (5.15). Approximating the variance of the service time, as suggested in [45], by $(T_0^{<s,d>} - M)^2$, yields the mean waiting time at the source [77]

$$W_s^{<s,d>} = \frac{\frac{\lambda_g}{V}\left(T_0^{<s,d>}\right)^2\left(1+\left(T_0^{<s,d>}-M\right)^2/\left(T_0^{<s,d>}\right)^2\right)}{2\left(1-\frac{\lambda_g}{V}T_0^{<s,d>}\right)} \tag{5.24}$$

5.3.3 The multiplexing factor ($\bar{V}$)

The network latency, $\bar{S}^{<s,d>}$, (i.e. the time to cross the network) of a message travelling between node s and d consists of the actual message transmission time, the blocking delay and the routing delay. Hence it can be written as

$$\bar{S}^{<s,d>} = M + \left|\Im_{<s,d>}\right| + B^{<s,d>} \tag{5.25}$$

When multiple virtual channels are used per physical channel, they share the same bandwidth in a time-multiplexed manner. The average multiplexing degree of virtual channels that takes place at a physical channel in the path between source node s and destination node d can be estimated by [36]

$$V^{<s,d>} = \frac{\sum_{v=1}^{V} v^2 P_v^{\bar{S}^{<s,d>}}}{\sum_{v=1}^{V} v P_v^{\bar{S}^{<s,d>}}} \tag{5.26}$$

In the above equation, $P_v^{\bar{S}^{<s,d>}}$ $(v = 0, 1, \ldots, V)$ represents the probability of busy virtual channels, as calculated in Chapter 3, when the mean service time is $\bar{S}^{<s,d>}$.

Finally, the average degree of virtual channel multiplexing is computed by averaging the multiplexing degree over all possible source-destination pairs. That is

$$\bar{V} = \frac{\sum_{s \in G-\{d\}} V^{<s,d>}}{N-1} \tag{5.27}$$

5.3.4 Model implementation

Examining the above equations reveals that there are some interdependencies between the deferent variables and equations of the analytical model. For instance, equation (5.15) reveals that the channel holding time, $T_i^{<s,d>}$, is a function of the mean waiting time, $W_i^{<s,d>}$, while the waiting time is expressed as a function of the channel holding time in

equation (5.13). Therefore, the model equations are solved iteratively, because a closed-form solution is very difficult to obtain [1, 5, 22, 31, 82, 111, 126]. The following simple iterative procedure is used to compute the mean message latency as described by the above model.

Step 1: Read M, F, λ_g, k and n

Step 2: Construct the set $\Im_{<s,d>}$ using equation (5.4)

Step 3: For every i $(1 \le i \le |\Im_{<s,d>}|)$ do the following:

Step 3.1: Compute $A_i^{<s,d>}$ using equation (5.14)

Step 3.2: Initialize $T_i^{<s,d>} = M + A_i^{<s,d>}$

Step 3.3: Compute $W_i^{<s,d>}$, $\theta_i^{<s,d>}$ and $\Upsilon_i^{<s,d>}$ using equations (5.13), (5.20) and (5.23), respectively

Step 3.4: Calculate the new $T_i^{<s,d>}$ using equation (5.15)

Step 3.5: Repeat Steps 3.3 and 3.4 until the difference between the new and old $T_i^{<s,d>}$ is less than a given error bound, ε

Step 4: Compute $B^{<s,d>}$, $W_s^{<s,d>}$ and $V^{<s,d>}$ using equations (5.11), (5.24) and (5.26), respectively

Step 5: Repeat steps 2, 3 and 4 for all possible destinations

Step 6: Compute $\bar{W}_s$, B, R, $\bar{V}$ and $\bar{S}$ using equations (5.5), (5.6), (5.7), (5.27) and (5.2), respectively

Step 7: Calculate the mean message latency using equation (5.1)

It is noteworthy to mention that we may avoid computing $B^{<s,d>}$, $W_s^{<s,d>}$ and $V^{<s,d>}$ for some source-destination pairs. This is because adaptive routing tends to balance the traffic over all network dimensions of the naturally symmetric k-ary n-cube. The following example illustrates this claim. In a 3-ary 3-cube network, if we assume that the source node s is equal to (0,0,0), then the destination node d can take any of the following values: (0,0,1), (0,0,2), (0,1,0), (0,1,1,), (0,1,2), (0,2,0), (0,2,1) (0,2,2) (1,0,0) (1,0,1) (1,0,2) (1,1,0), (1,1,1), (1,1,2), (1,2,0), (1,2,1), (1,2,2), (2,0,0), (2,0,1), (2,0,2), (2,1,0), (2,1,1), (2,1,2), (2,2,0), (2,2,1) and (2,2,2). However, due to the adaptive routing and the symmetry of the network, the following destinations will give the same latency results.

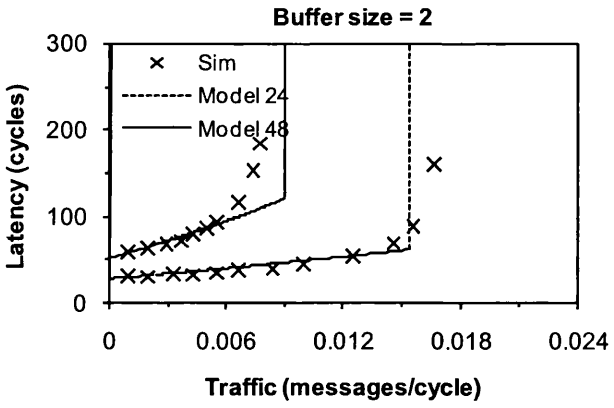
- (0,0,1) = (0,1,0) = (1,0,0),
- (0,0,2) = (0,2,0) = (2,0,0),
- (0,1,1) = (1,1,0) = (1,0,1),
- (0,2,2) = (2,0,2) = (2,2,0),
- (1,1,2) = (1,2,1) = (2,1,1),
- (1,2,2) = (2,1,2) = (2,2,1), and
- (0,1,2) = (1,2,0) = (2,0,1) = (1,0,2) = (0,2,1) = (2,1,0).

Therefore, calculating the latency of only one pair of the above 7 permutations (and also for (1,1,1) and (2,2,2)) will reduce substantially the run time of the model. The computation will be (26/9) times faster since we will consider only 9 permutations, instead of computing (3^3-1) permutations corresponding to the 26 possible destination nodes. Such a gain in computation time would improve even more, as the network size increases.

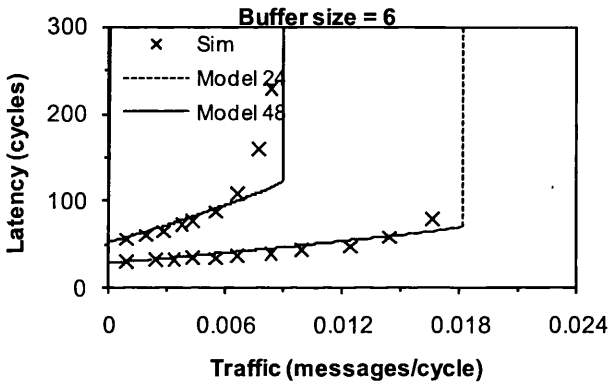
5.4 Validating the model

The above analytical model has been validated through the discrete-event simulator that has been used in previous chapters. However, the simulator has been augmented to deal with adaptive routing is used here. The simulator mimics the behaviour of adaptive routing in wormhole-switched bidirectional k-ary n-cubes with finite buffers and multiple virtual channels. The mean message latency is defined as the time from the generation of a message until the last flit reaches its destination. For each simulation experiment a total of 30 batches of messages were delivered to their destinations where each batch consisted of 5000 messages. Statistics for the first 2 batches have been discarded to avoid distortions due to warm-up conditions [42, 95]. Extensive experiments have been performed for several combinations of network parameters, in order to validate the proposed model. However, Figures 5.1 through 5.6 present validation results for the 8-ary 2-cube, 16-ary 2-cube and 8-ary 3-cube networks with message length M= 24, 48 and 96 flits. The number of virtual channels is set to V=3 and 5 with different buffer sizes (F=2, 3, 4, 6, 8, 12 and 16 flits).

(a)



(b)



(c)

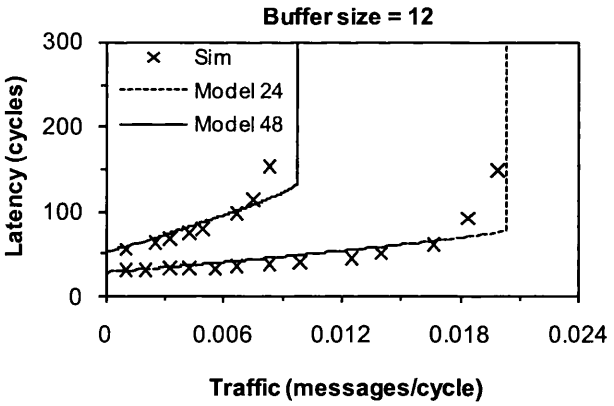
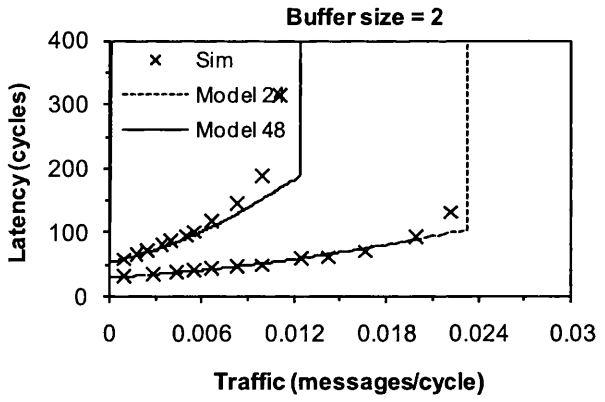
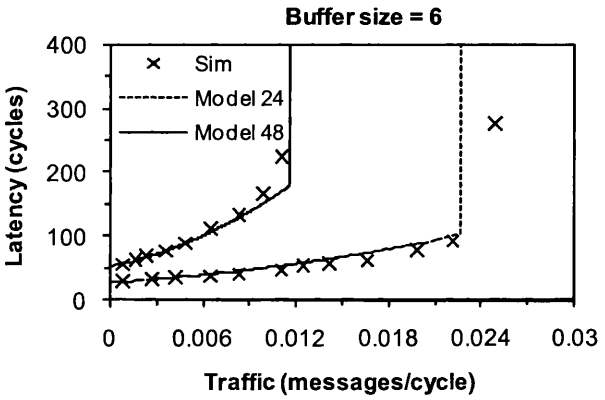


Figure 5.1: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 2-cube. Message length M=24 and 48 flits, number of virtual channels V=3 and buffer size (a) F=2, (b) F=6, and (c) F= 12 flits.

(a)



(b)



(c)

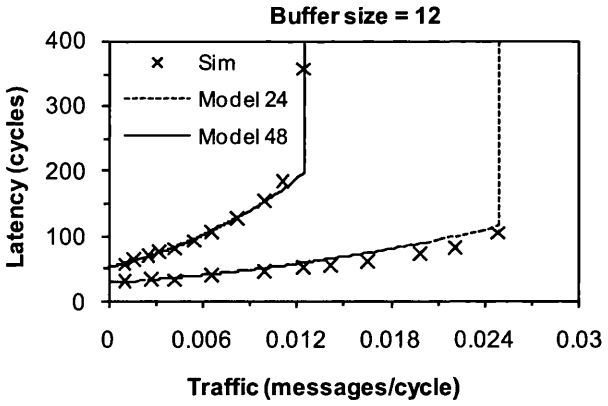
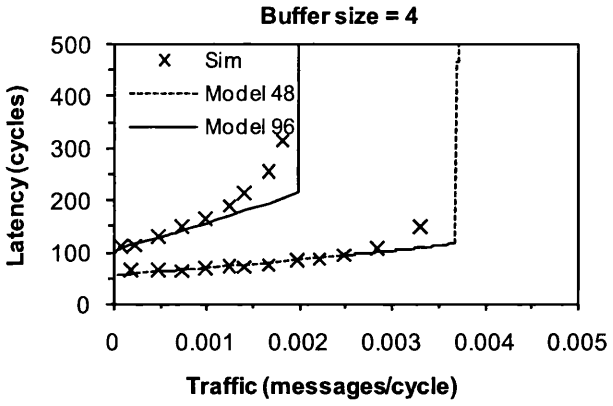
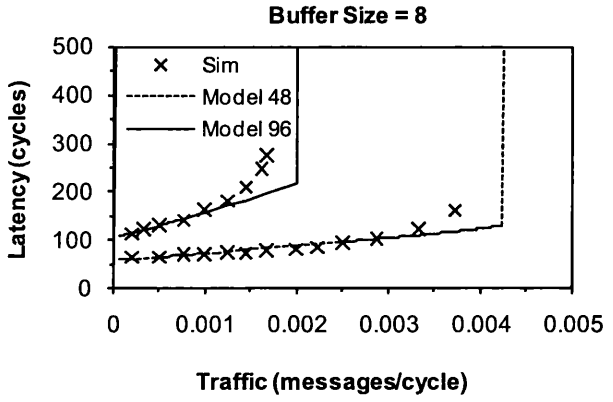


Figure 5.2: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 2-cube. Message length M=24 and 48 flits, number of virtual channels V=5 and buffer size (a) F=2, (b) F=6, and (c) F= 12 flits.

(a)



(b)



(c)

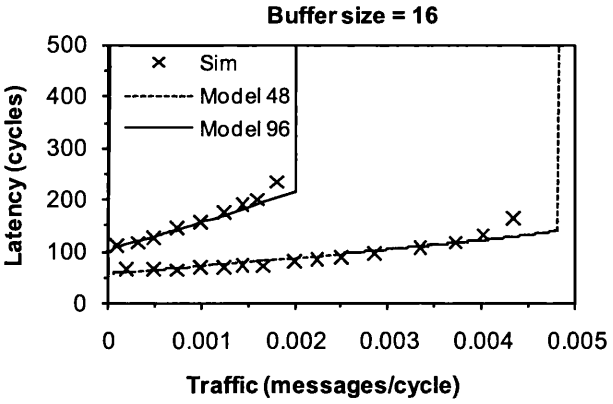
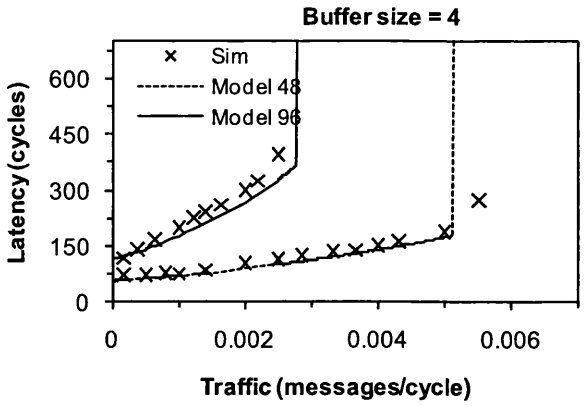
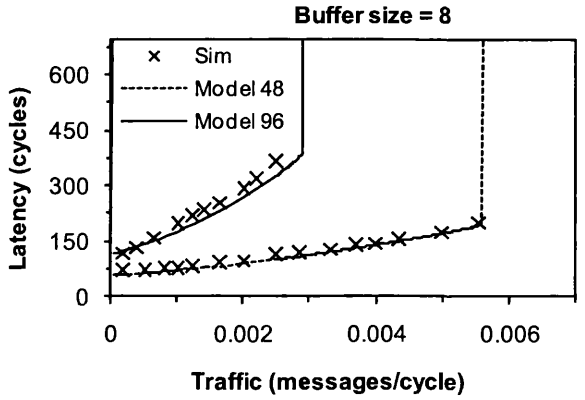


Figure 5.3: Mean message latency predicted by the model and simulator against the traffic generation rate in the 16-ary 2-cube. Message length M=48 and 96 flits, number of virtual channels V=3 and buffer size (a) F=4, (b) F=8, and (c) F= 16 flits.

(a)



(b)



(c)

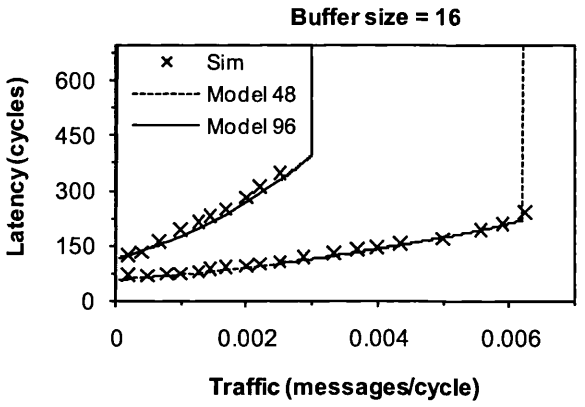
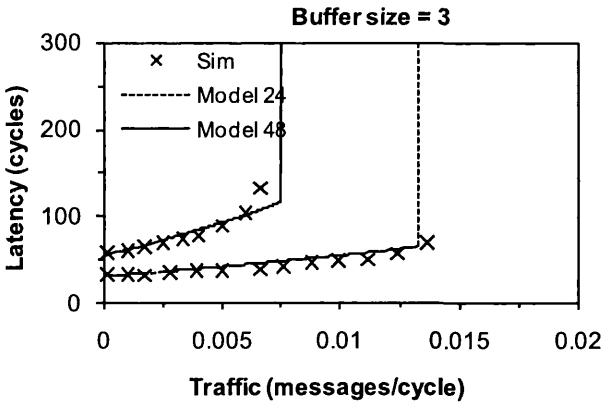
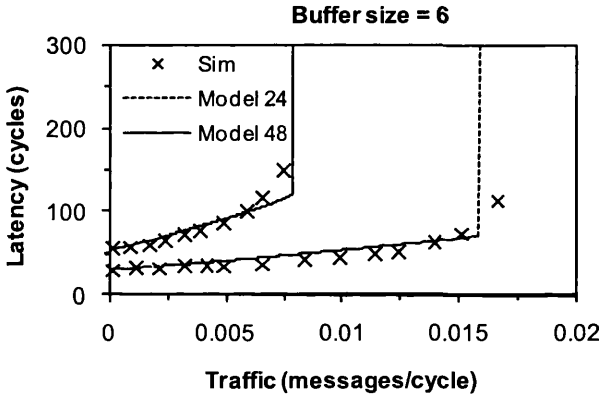


Figure 5.4: Mean message latency predicted by the model and simulator against the traffic generation rate in the 16-ary 2-cube. Message length M=48 and 96 flits, number of virtual channels V=5 and buffer size (a) F=4, (b) F=8, and (c) F= 16 flits.

(a)



(b)



(c)

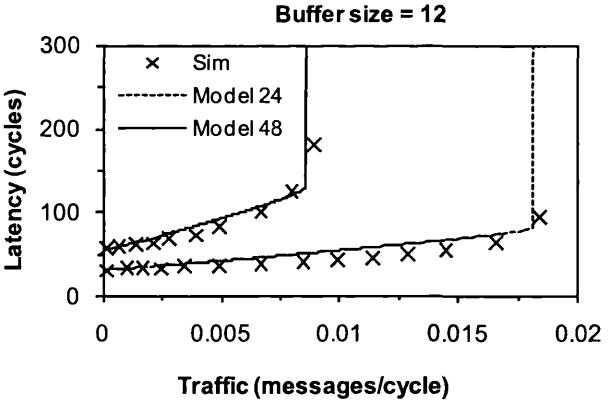
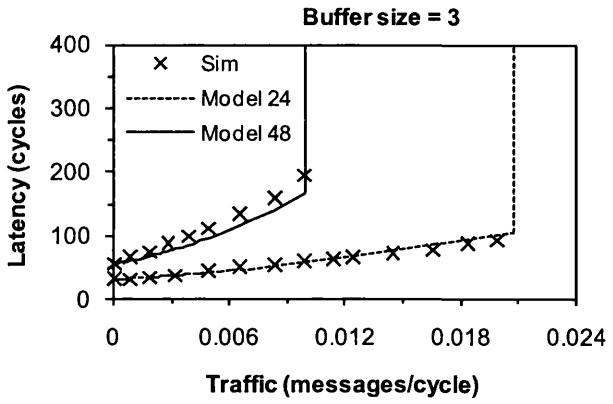
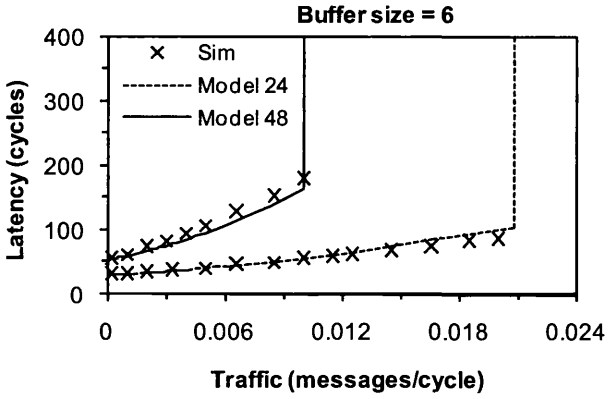


Figure 5.5: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 3-cube. Message length M=24 and 48 flits, number of virtual channels V=3 and buffer size (a) F=3, (b) F=6, and (c) F= 12 flits.

(a)



(b)



(c)

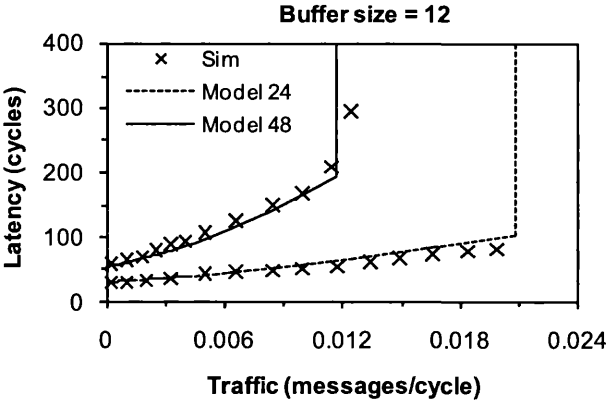


Figure 5.6: Mean message latency predicted by the model and simulator against the traffic generation rate in the 8-ary 3-cube. Message length M=24 and 48 flits, number of virtual channels V=5 and buffer size (a) F=3, (b) F=6, and (c) F= 12 flits.

The figures reveal that the analytical model predicts latency results with a high degree of accuracy under low, moderate and high traffic conditions. However, the accuracy of the model degrades, in some cases, around the saturation point. This is due mainly to the various approximations that have been made in the analysis; for example those made for calculating the variance of the service time distribution (equations 5.13 and 5.24) and for the assumption of independence between the probability of blocking due to the contention for virtual channels and the probability of blocking due to finite buffers. These assumptions simplify greatly the development of the model. Nevertheless, the high degree of accuracy along with the simplicity of the proposed model make it a practical and cost-effective evaluation tool to gain insights into the behaviour of adaptive routing when finite buffers and multiple virtual channels are used in wormhole-switched *k*-ary *n*-cubes.

5.5 Extension of the model

This section outlines briefly the modifications that have to be made to the model in order to adapt it to the unidirectional torus and hypercube.

5.5.1 The unidirectional torus

In unidirectional networks the maximum number of hops that a message makes to traverse one dimension is equal to (*k*-1). Hence the number of hops that a message makes at a given dimension, *j*, while crossing from the source node *s*=(0, 0, ..., 0) to any destination node $d=(d_0, d_1, \ldots, d_{n-1})$ is simply equal to d_j. That is

$$h_j = d_j \qquad ; j = 0, 1, \ldots, n-1 \tag{5.28}$$

The router in unidirectional *k*-ary *n*-cubes has *n* output channels (i.e. one channel per dimension either to the positive or to the negative direction). The average number of hops that a message makes across one dimension, $\bar{k}$, and also the rate of messages received by each physical channel, λ_c, for unidirectional networks are respectively given by [5, 111]

$$\bar{k} = \frac{k-1}{2} \tag{5.29}$$

$$\lambda_c = \frac{\lambda_g \bar{d}}{n} \tag{5.30}$$

5.5.2 The hypercube

When the network is a hypercube (i.e. a k-ary n-cube with k=2), some of the equations that are derived above are modified as follows. In the hypercube, a message makes at most one hop per dimension and therefore, the average number of hops that a message may traverse to cross the entire network can be calculated as [1]

$$\bar{d} = \sum_{i=1}^{n} i \frac{\binom{n}{i}}{N-1} = \frac{nN}{2N-2} \tag{5.31}$$

When a message makes one hop in the hypercube, then it consequently has passed one dimension. This simplifies the calculation of the blocking due to contention as there is no need to evaluate equation (5.20) above. Therefore, the probability of blocking due to contention for virtual channels can be written as [124]

$$\theta_i^{<s,d>} = (P_{i,a}^{<s,d>})^{|\Im_{<s,d>}|-i} P_{i,d}^{<s,d>} \tag{5.32}$$

Since there are only two nodes per dimension and the routing algorithm is minimal, there are no cyclic dependencies between channels in each dimension. Therefore, to implement Duato's adaptive routing algorithm in the hypercube with V virtual channels, the virtual channels are split into two sets: $VC_1 = \{v_2, v_3, \ldots, v_V\}$ and $VC_1 = \{v_1\}$. This is because only one virtual channel is needed to insure deadlock-freedom [49] in the hypercube. During any routing step, a message may select any available virtual channel that brings it closer to its destination. If all adaptive virtual channels are busy, then the message is routed via the deterministic virtual channel (i.e. v_1) associated to the highest (lowest) dimension to be visited. Therefore, the probability that all adaptive virtual channels are busy and the probability that the deterministic virtual channel is busy are expressed as [22, 124]

$$P_{i,a}^{<s,d>} = P_V^{T_i^{<s,d>}} + \frac{P_{V-1}^{T_i^{<s,d>}}}{\binom{V}{V-1}} \tag{5.32}$$

$$P_{i,d}^{<s,d>} = P_V^{T_i^{<s,d>}} \tag{5.33}$$

5.6 Conclusions

There have been many attempts to model the performance of adaptive routing in wormhole-switched k-ary n-cubes. However, most of the studies have assumed unrealistically single flit buffers per network channel. This chapter has presented the first analytical model to capture the effect of finite buffers on the performance of adaptive routing in wormhole-switched k-ary n-cubes with multiple virtual channels. Simulation experiments have revealed that the model predictions are in close agreement with results obtained from an event-driven simulator that mimics the behaviour of the studied network. The simplicity and the high accuracy of the model make it a versatile and cost-effective tool to gain insights into the performance metrics of interconnection networks with more relaxed assumptions with respect to the available buffer size at each network channel.

In the next chapter, the models proposed in Chapter 3, 4 and 5 are used to carry out three comparative performance evaluation studies of k-ary n-cubes. Namely, the relative performance merits of k-ary n-cube topologies will be revisited when their routers are equipped with finite buffer capacity. Next, the impact of different virtual channels arrangements on the performance of wormhole-switched k-ary n-cubes will be investigated. Finally, the relative performance merits of adaptive and deterministic routing algorithms will be analysed in the presence of finite buffers and multiple virtual channels.

Chapter 6

Performance Analysis of k-ary n-cubes with Finite Buffers and Virtual Channels

6.1 Introduction

It has been pointed out in Section 1.1.4 that the topology properties and the implementation technology force some limitations on the channel's bandwidth [5, 42, 49]. Constant *bisection* width [34, 35] and constant *pin-out* [2, 5] are two important channel bandwidth constraints that have been used by several researchers to analyse the performance merits of competing network topologies. Without such constraints, one cannot evaluate a topology or perform a fair comparison between the topologies. Comparative performance studies of k-ary n-cubes under different design constraints and operating conditions have been reported widely in the literature [2, 5, 35, 72, 112, 114, 130, 132].

When systems are implemented on a single VLSI chip, the wiring density of the network determines the overall system cost and performance [143]. Dally [35] has shown that under such constraint (i.e. with constant bisection width), the low-dimensional 2D torus outperforms the higher-dimensional hypercube [35]. This can be accounted for by the wider channels, and thus higher bandwidth, of the 2D torus that compensates for the lower diameter of the hypercube.

Abraham and Padmanabhan [2] and Agarwal [5] have argued that while the wiring density constraint is applicable where the entire network is implemented on a single VLSI chip, this is not the case in the situation where the network has to be partitioned over many chips. In

this scenario, the authors in [2, 5] have identified that the bandwidth constraints is imposed by the chip's I/O pins through which data must travel. Unlike the study in [35], they have concluded that the hypercube exhibits the best performance under the constant pin-out constraint.

It is worth mentioning that the routing algorithm used in the above studies [2, 5, 35] has been the deterministic dimension-order routing algorithm. Sarbazi-Azad et al [130] however, have compared the performance of adaptive routing in different topologies under both constant bisection width and constant pin-out constraints. They have arrived at the same conclusion as Dally [35] when the constant bisection width constraint is imposed (i.e. the 2D torus outperforms the hypercube). On the other hand, when the pin-out constraint is considered, the authors in [130] have concluded that for small, moderate, and large networks, the topology exhibiting the best relative performance is, respectively, the hypercube, 2D torus, and 3D torus.

All these existing studies [2, 5, 35, 72, 112, 114, 130, 132] have assumed single flit buffers per virtual channel. Deploying finite size buffers, however, may add extra complexity to the router design and consequently influence the network cost and performance [96]. Moreover, introducing finite buffers reduces the number of channels that a message occupies, and hence minimises the probability of blocking which affects considerably the overall network performance. In view of these observations, the first part of this chapter presents the first analytical comparison of the relative performance merits of the 2D torus, 3D torus and hypercube in the presence of finite buffers.

To the best of our knowledge, studying the issue of optimal arrangement of virtual channels (i.e. given a fixed amount of buffer per physical channel, what is the optimal way to arrange it into virtual channels) has so far resorted to simulation experiments [36, 119]. This can chiefly be accounted for by the lack of analytical models that captures the effects of finite size buffers on the performance of wormhole-switched networks.

Dally [36] has concluded that with the total amount of buffer per physical channel held constant, adding virtual channels to a network allows a more effective use of buffer space than adding depth to a single virtual channel. More recently, Rezazad and Sarbazi-Azad [119] have conducted a similar simulation study and arrived at similar conclusions. They have identified that increasing the buffer depth, and therefore decreasing the number of virtual channels, results in good performance for low traffic rates, while the converse (i.e. increasing the number of virtual channels can improve the performance of the network) is true under high traffic conditions. Both studies [36, 119] have suggested that increasing the number of virtual channels beyond a threshold value has adverse effect on network performance. For example, in a 16-ary 2-cube network with constant message size and constant buffer size of 32 flits, the author in [36] has observed that the best performance is achieved when the available buffer space is arranged into 4 virtual channels (each with 8 flits buffers), suggesting an optimal number of virtual channels for performance-cost tradeoffs.

It should be mentioned that the existing studies in [36, 119] have considered deterministic dimension-order routing only. Motivated by the above observations, the second part of this chapter investigates, by means of analytical models, the optimal arrangement of the available buffer space into virtual channels when adaptive routing is used in k-ary n-cubes.

Although it has been shown in several studies [26, 48, 96, 101, 136] that adaptive routing performs better than deterministic routing in most cases, the majority of these studies have assumed single flit buffers. Moreover, existing studies have resorted to simulation and either ignored the inter-router delay (i.e. the blocking and the actual transmission time) [26, 48] or considered different switching methods [101]. Making concrete comparison between adaptive and deterministic routing requires taking into account the finite buffers constraint. This is because increasing the buffer size, increases the router size and complexity which perhaps, reduces the router speed and, consequently, the overall network performance [96]. Therefore, as the cost of adaptivity has been questioned by several researchers [14, 101], the

last part of this chapter compares adaptive and deterministic routing algorithms when finite buffers are employed.

The rest of this chapter is organised as follows. Section 6.2 outlines the assumptions used in the analysis while Section 6.3 investigates how switching and routing delays affect network performance. Section 6.4 compares the relative performance merits of different topologies in the presence of finite size buffers under different implementation constraints. Section 6.5 is devoted to studying the optimal arrangement of virtual channels per physical channel and Section 6.6 sheds some light on the performance of adaptive and deterministic routing algorithms when finite buffers and multiple virtual channels are deployed. Finally, Section 6.7 concludes the chapter by summarising the findings of the conducted studies.

6.2 Assumptions

The present analysis uses the analytical models developed in Chapters 3, 4 and 5 and is further based on the following assumptions which have been used widely in existing studies [2, 5, 26, 35, 36, 48, 72, 96, 101, 112, 119, 130, 136]

a) Message length is fixed and is equal to M flits. Moreover, Message destinations are distributed uniformly across the network nodes.

b) Nodes generate traffic independently of each other and according to a Poisson process with a mean rate of λ_g messages per cycle.

c) The channel cycle time (i.e. the time to transmit one flit from one router to the next over a physical channel), the switching delay (i.e. the time to cross from the input port to the output port of the switch) and the routing decision time are assumed to be t_c, t_s and t_r cycles, respectively.

d) The local queue at the injection channel in the source node has infinite capacity. Messages are transferred to the local PE immediately after they arrive at their destinations.

e) V (V>1) virtual channels are used per physical channel. At a given routing step a message chooses a specific virtual channel according to the routing algorithm (routing algorithms are explained in detail in Chapter 2).

6.3 Intra-router delay

The intra-router delay (i.e. the time to cross the router) has been assumed to be a constant when developing the models in Chapters 3, 4 and 5, as the models accuracy depends mainly on predicting message blocking delay. However, to ensure a fair and concrete comparison, the intra router delay must be considered as the complexity of the router affects the performance [26, 130]. This is true especially for wormhole-switched networks, as their performance is sensitive to these delays [42, 49].

Two components contribute to the intra-router delay; routing delay and switching delay [26]. The routing delay involves address decoding, routing decision and updating the message header. According to [26, 48], the routing delay is given by

$$t_r = \frac{4.7 + 1.2\log_2(R)}{4.9} \tag{6.1}$$

In the above equation, R is the degree of freedom, or the number of alternative channels that can be used by the routing algorithm to route the message through.

Similarly, the switching delay is composed of the delay involved in the internal flow control, the delay to cross the crossbar switch and the time to set up the output channel. Hence, according to the studies in [26, 48, 101], if P denotes the number of ports in the crossbar switch, then the switching delay is given by

$$t_s = \frac{3.4 + 0.6\log_2(P)}{4.9} \tag{6.2}$$

It should be mentioned that the original equations presented in [26, 48, 101] compute these delays in terms of time units (namely nanoseconds) and are not divided by 4.9 as in the above equations. However, because our models measure different delays in network cycles (instead of nanoseconds), the equations have to be normalised over the network cycle time which was found to be 4.9 nanoseconds in [26, 48, 101].

6.4 Torus versus hypercube

The 2D torus, 3D torus and hypercube have been used widely in many practical and experimental systems such as Intel Paragon [24] and iWarp [21], Ncube [50, 115], MIT J-Machine [109] and Alewife [6], Cray T3E [133], T3D [68] and XT3 [147], IBM Blue Gene/L [3]. In this study we focus on the unidirectional versions of these networks in order to compare the torus against the hypercube which is topologically a unidirectional 2-ary *n*-cube whereas the bidirectional 2-ary *n*-cube is a hypercube with redundant links [130]. Previous studies [2, 5, 35, 130] have also taken this option. This section revisits the relative performance merits of these topologies in the presence of finite buffers.

Let the radix (i.e. the number of nodes per dimension) of the 2D torus and 3D torus be denoted by k_{2D} and k_{3D}, respectively. Let also n be the number of dimensions in the hypercube. To keep the number of nodes in the network, N, fixed for all three topologies, then k_{2D}, k_{3D} and n must be chosen such that

$$N = (k_{2D})^2 = (k_{3D})^3 = 2^n \tag{6.3}$$

To allow for a fair comparison, Section 6.4.1 highlights how different implementation constraints affect the channel cycle time of these topologies. Section 6.4.2 then sheds some light on the intra-router delay of the topologies under investigation. The comparison between the topologies in question are then presented and discussed in Section 6.4.3.

6.4.1 Implementation constraints

In wormhole switching a flit is often composed of one or more *phits*, each of which is equal to the channel width, C_w, and takes one cycle to be transmitted from one router to the next [5,42, 49]. This is because, in practice, different implementation technologies impose a limit on the channel bandwidth. Constant *bisection* width [34, 35] and constant *pin-out* [2, 5] have been two important bandwidth constraints that have been used by several researchers to study and compare the performance merits of different network topologies [130].

Let us consider the 2D torus as our base network for the comparison and calculate the desired parameters (e.g. channel bandwidth and intra-router delay) for the 3D torus and hypercube in terms of those in the 2D torus base network (i.e. the networks can for example be implemented in PCB 2D physical medium). We could have focused equally in a 3D physical medium (e.g. cabinets) as our base network by simply changing the equations below. However, as long as the relative performance merits are the goal, a 2D or 3D physical medium constraint make little change to the conclusions.

Let us assume that the flit size is equal to the channel width in the 2D torus. This means that the phit size (i.e. the channel width) is equal to the flit size (i.e. $t_c^{2D} = 1$). However, a flit in the hypercube and 3D torus requires more than one cycle to be transmitted from one router to the next, because the phit size is smaller than the flit size. In other words, the channel width of the 3D torus and hypercube is smaller than that of the 2D torus when bandwidth constraints are imposed [42, 49]. Therefore, when constant bisection width constraint is imposed, the channel cycle time for the 3D torus is given by [130]

$$t_c^{3D} = \sqrt[3]{N} \tag{6.4}$$

Similarly, the channel cycle time for the hypercube under the same constraint is [130]

$$t_c^{hyp} = \frac{N}{8} \tag{6.5}$$

In other words, if the bisection width is kept constant for the three topologies, one flit takes one cycle to be transmitted from one router to the next in the 2D torus, while it takes $\sqrt[3]{N}$ cycles in the 3D torus and $(N/8)$ cycles in the hypercube. Similarly, when the constant pin-out constraint is imposed, the channel cycle time for the 3D torus and hypercube is given by [130]

$$t_c^{3D} = 3\sqrt[6]{N} \tag{6.6}$$

$$t_c^{hyp} = n\sqrt{N} \tag{6.7}$$

Equations (6.4 – 6.7) allows us to calculate the channel cycle time for the 3D torus and hypercube with respect to that of the 2D torus. However, to make a fair comparison, the router complexity should also be taken into account. Because this study assumes finite buffers as opposed to single flit buffers, the total buffer size per router will be $(P.V.F)$, where P is the number of input channels to the router, V is the number of virtual channels per physical channel, and F is the buffer size per virtual channel. Therefore, the total buffer space allocated to each router in the 2D torus, 3D torus and hypercube should satisfy

$$2FV_{2D} = 3FV_{3D} = nFV_{hyp} \tag{6.8}$$

To keep the total buffer per router fixed for all three topologies the number of virtual channels for the 2D torus, 3D torus and hypercube must be normalised. Since the hypercube requires only 2 virtual channels per physical channel to ensure deadlock freedom (according to Duato's adaptive routing algorithm [46, 49]), the number of virtual channels for the above topologies can be normalised using equation (6.8) as

$$V_{hyp} = 2 \tag{6.9}$$

$$V_{2D} = n \tag{6.10}$$

$$V_{3D} = \frac{2}{3}n \tag{6.11}$$

It should be mentioned here that one could equally normalise the buffer size instead of the number of virtual channels, to keep the total buffer allocated to each router constant. In such a case, the router's complexity (i.e. the number of ports) of the topologies in question will differ from one topology to another. However, we have chosen to normalise the number of virtual channels for the following reasons. Firstly, this makes the cost of the hardware used inside a router in the 2D torus, 3D torus and hypercube comparable. Secondly, this normalisation makes the switching delay in these networks comparable as it is affected directly by the complexity of the router (i.e. the number of ports in the crossbar switch), which is kept fixed. Thirdly, in networks for parallel computers, there is abundant buffer space that can be utilised and the most important constraint is the wiring complexity (i.e. the number of channels) [41, 42, 49, 116] and so it is more natural to normalise the number of virtual channels instead of normalising the buffers. Finally, this option was selected in order to compare the results of this study with those from previous studies [119, 130].

6.4.2 Routing and switching delays

As can be seen from equation (6.1), to calculate the routing delay, the degree of freedom, R, should be found first. When Duato's adaptive routing algorithm is used [46, 49], a message may be routed to any adaptive virtual channel of the remaining dimensions to be visited or to one of the two deterministic virtual channels (i.e. when all adaptive virtual channels are busy). Moreover, a message may also be routed to the local PE (i.e. when the current node is the destination node). Therefore, for k-ary n-cubes, the degree of freedom of the adaptive routing algorithm can be written as

$$R = D_{rem}|VC_1| + |VC_2| + 1 \tag{6.12}$$

In the above equation, D_{rem} is the number of remaining dimensions to be visited, $|VC_1|$ and $|VC_2|$ are the number of adaptive and deterministic virtual channels, respectively. Substituting this into equation (6.1) gives the routing delay for three topologies is given by

$$t_r^{2D} = \frac{4.7 + 1.2\log_2\left(2(V-2)+2\right)}{4.9} \tag{6.13}$$

$$t_r^{3D} = \frac{4.7 + 1.2\log_2\left(3(V-2)+2\right)}{4.9} \tag{6.14}$$

$$t_r^{hyp} = \frac{4.7 + 1.2\log_2\left((n-1)(V-1)+2\right)}{4.9} \tag{6.15}$$

Moreover, the switching delay for the three topologies can be found using equation (6.2) which is only a function of P, the number of ports in the crossbar switch. However, we have kept P constant by using a different number of virtual channels per physical channel for the three topologies, as explained in the previous section. This results in an equal switching delay, as calculated by equation (6.2), for the 2D torus, 3D torus and hypercube. Nevertheless, it is worth mentioning that another way of normalising the switching delay is possible by using a same number of virtual channels per physical channel for the three topologies and hence different values of P. In such a scenario, equation (6.2) should be used to compute the switching delay for the 2D torus, 3D torus and hypercube.

6.4.3 Results and discussions

In this section, the performance merits of the three topologies under study are examined for the constant bisection width and constant pin-out constraints using the above cost-performance model. Numerous experiments have been conducted for various buffer sizes, message sizes and network sizes. However, for illustration purpose, we present the results for network topologies and sizes listed in Table 6.1.

Table 6.1: Network topologies and sizes used for the comparisons in Section 6.4.3

Network	N	2D torus	3D torus	Hypercube
Small	64	8 × 8	4 × 4 × 4	6-dimensional
Medium	256	16 × 16	6 × 6 × 6 ♦	8-dimensional
Moderately Large	1024	32 × 32	10 × 10 × 10 ♦	10-dimensional

♦ Approximated value

Table 6.2: The channel cycle time and the number of virtual channels per physical channel for the 2D torus, 3D torus and hypercube for N=64, 256 and 1024 nodes under constant bisection width (BS) and pin-out (PO) constraints

Size	N=64				N=256				N=1024			
Topology	V	t_r	t_c		V	t_r	t_c		V	t_r	t_c	
			BS	PO			BS	PO			BS	PO
2D torus	6	1.8	1	1	8	1.9	1	1	10	2	1	1
3D torus	4	1.7	4	3	5	1.8	6.3	3.8	7	2	10.1	4.8
Hypercube	2	1.6	8	6	2	1.7	32	16	2	1.8	128	40

The channel cycle time for the 2D torus is set to 1, indicating that the flit size is the same as the channel width. Equations (6.4-6.7) and (6.9-6.11) are used to compute the normalised channel cycle times and the number of virtual channels for the 2D torus, 3D torus and hypercube. Also the routing delay is calculated for all three topologies using equations (6.13-6.15). Table 6.2 summarises the normalised channel cycle times, routing delays and number of virtual channels for the topologies in question. Figures 6.1, 6.2 and 6.3 depict latency results predicted by the analytical models for the above mentioned topologies under the constant bisection width and constant pin-out constraints for messages of size 48[▽] flits and with different buffer sizes (namely F = 2, 4 and 10 flits). The horizontal axis represents the message generation rate, λ_g (messages/cycle), and the vertical axis represents the average message latency (in cycles) as predicted by the analytical models.

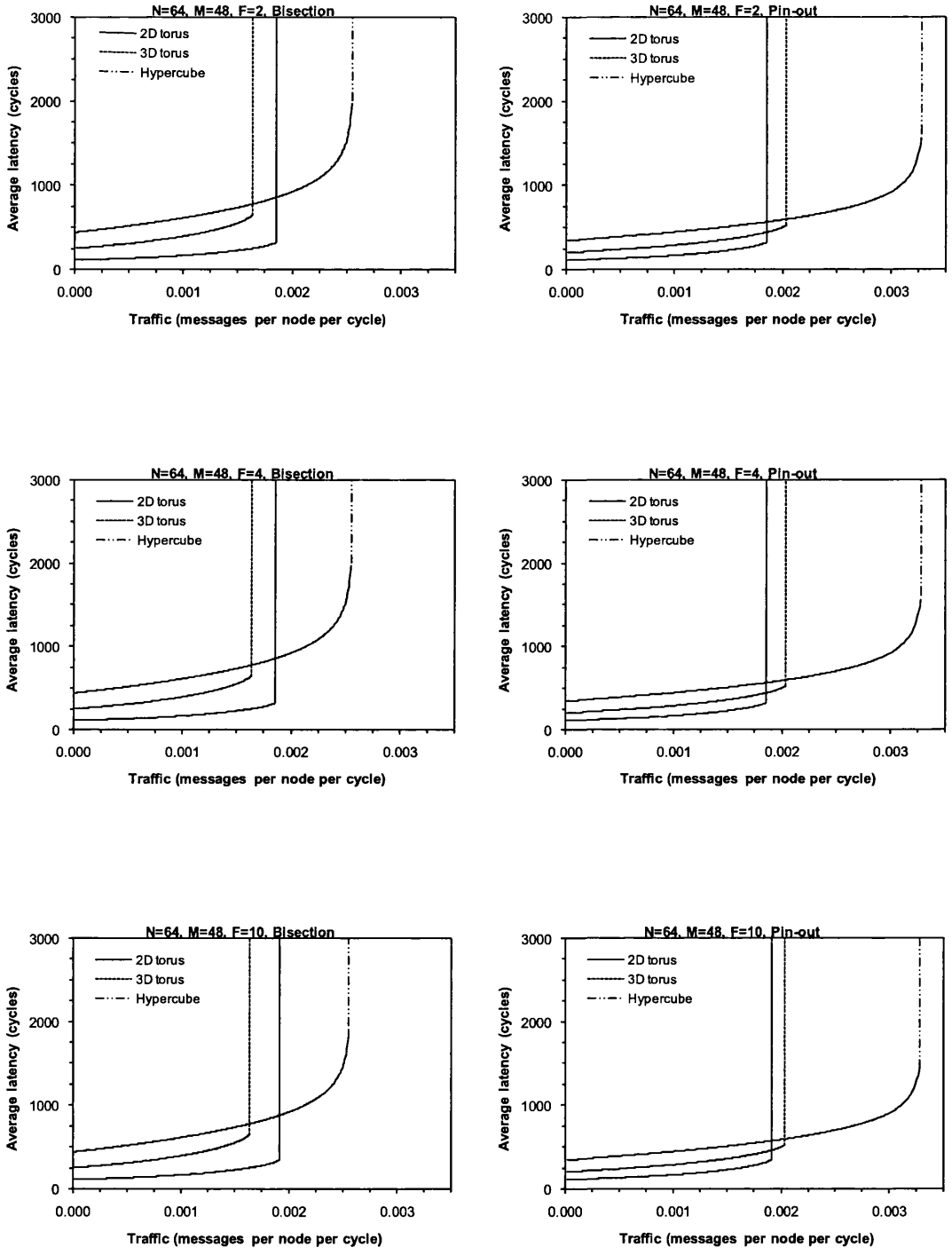
When the constant bisection width constraint is considered, Figures 6.1(a), 6.2(a) and 6.3(a) reveal that the 2D torus is able to exploit its wider channels to provide a lower latency than the 3D torus and hypercube, regardless of the buffer size and network size, when the traffic is low. However, as traffic increases, the relative performance merits of the three topologies vary from one network size to another. Figure 6.1(a) shows that when the network size is small (i.e. 64 nodes), then the performance of the 2D torus degrades substantially due to high message blocking, offsetting any advantage of having wider channels. In this case, and unlike the results that have been reported by other researchers [35, 130], the hypercube

[▽] Similar performance trends have been obtained when we experimented with M= 24 and 96 flits.

outperforms the 2D and 3D torus as traffic increases and the network approaches the saturation region. This can be accounted for by the small diameter of the hypercube (i.e. on average, messages make a smaller number of hops to attain to their destinations) and its rich connectivity that provides more alternative physical paths between communicating nodes and thus more adaptivity. Moreover, for small networks, the relative cycle time for the hypercube is not large compared to the 2D torus, as can be seen in Table 6.2.

With moderate and large networks under the same constraint, however, lower dimensional networks outperform their higher dimensional counterparts as can be seen from Figure 6.2(a) and Figure 6.3(a). This, on the one hand, can be accounted for the thinner channels of the hypercube compared to the 2D torus which results in a significantly higher number of cycles required to transmit one flit, as can be seen from Table 6.2. On the other hand, as the network size increases the average number of hops in the 2D torus increases more rapidly compared to the hypercube. For example, in 1024-nodes networks the average number of hops for the 2D torus is 31, while it is only 5 hops for the hypercube. This allows the 2D torus to exploit the available buffer size to reduce the number of channels that a message occupies, thus reducing the blocking delays. Whereas, in the hypercube, due to the smaller average number of hops, messages occupy almost all the channels between the source and destination nodes increasing the probability of blocking, even with large buffer sizes.

Nevertheless, it is worth mentioning that the 2D torus is able to exploit the increase in the buffer size more efficiently compared to the hypercube. We have noticed that, in 64-node systems, increasing the buffer size from 4 flits to 10 flits increases the maximum throughput in the 2D torus by 3.5 % compared to less than 0.25 % increase in the hypercube. This is mainly due to the higher average number of hops a message makes in the 2D torus (in this case 7 hops) compared to the hypercube (in this case 3 hops). Notice that, a 48-flit message requires 5 consecutive channels to be accommodated entirely when the buffer size is 10 flits. As shown in Figure 6.4 (a), this effect becomes more noticeable as the network size grows.



(a) (b)

Figure 6.1: The average message latency for three different network topologies (2D torus, 3D torus and hypercube) with N=64 nodes, for message length M=48 flits and different buffer sizes (F=2, 4 and 10 flits) when (a) constant bisection width constraint and (b) pin-out constraint, is imposed.

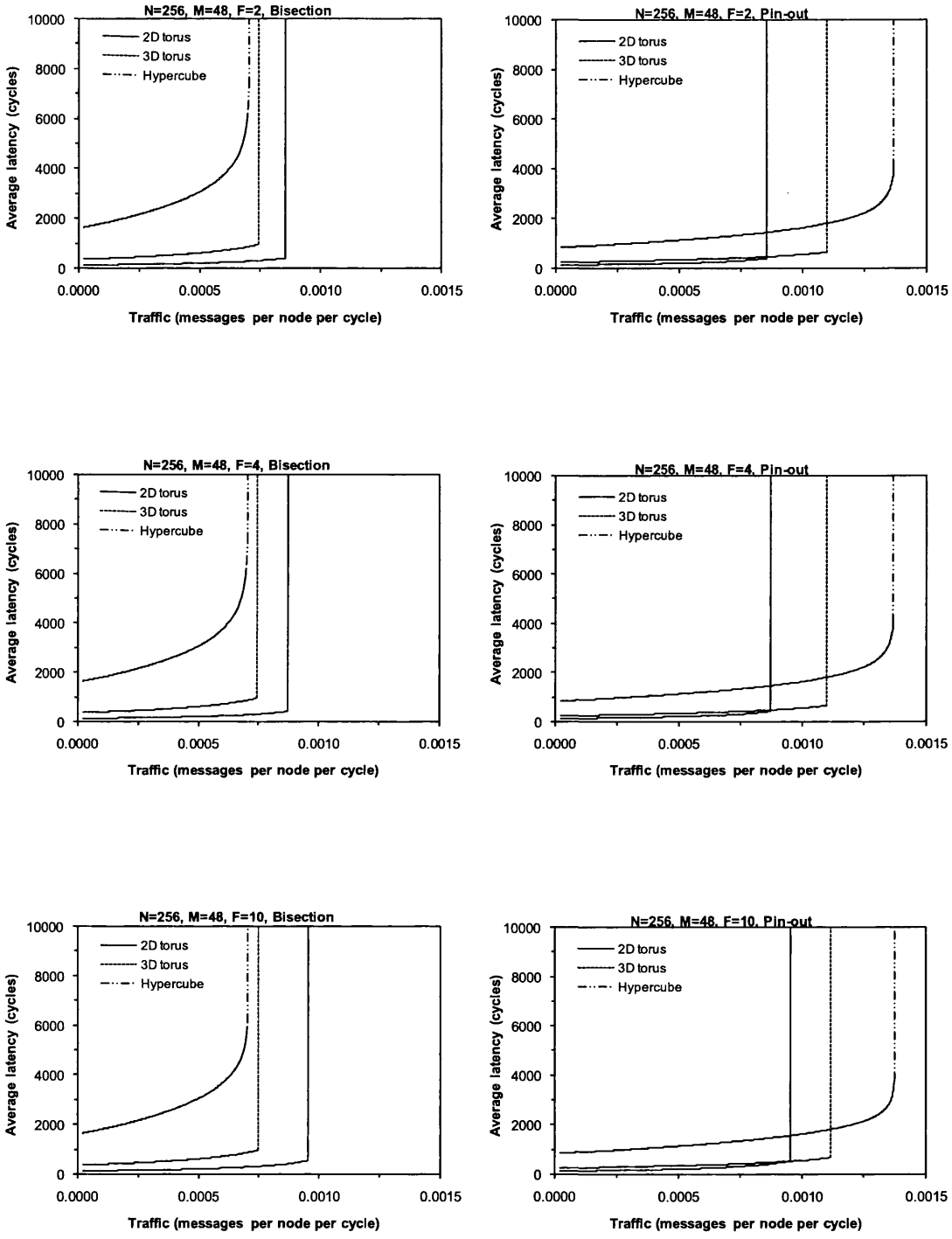
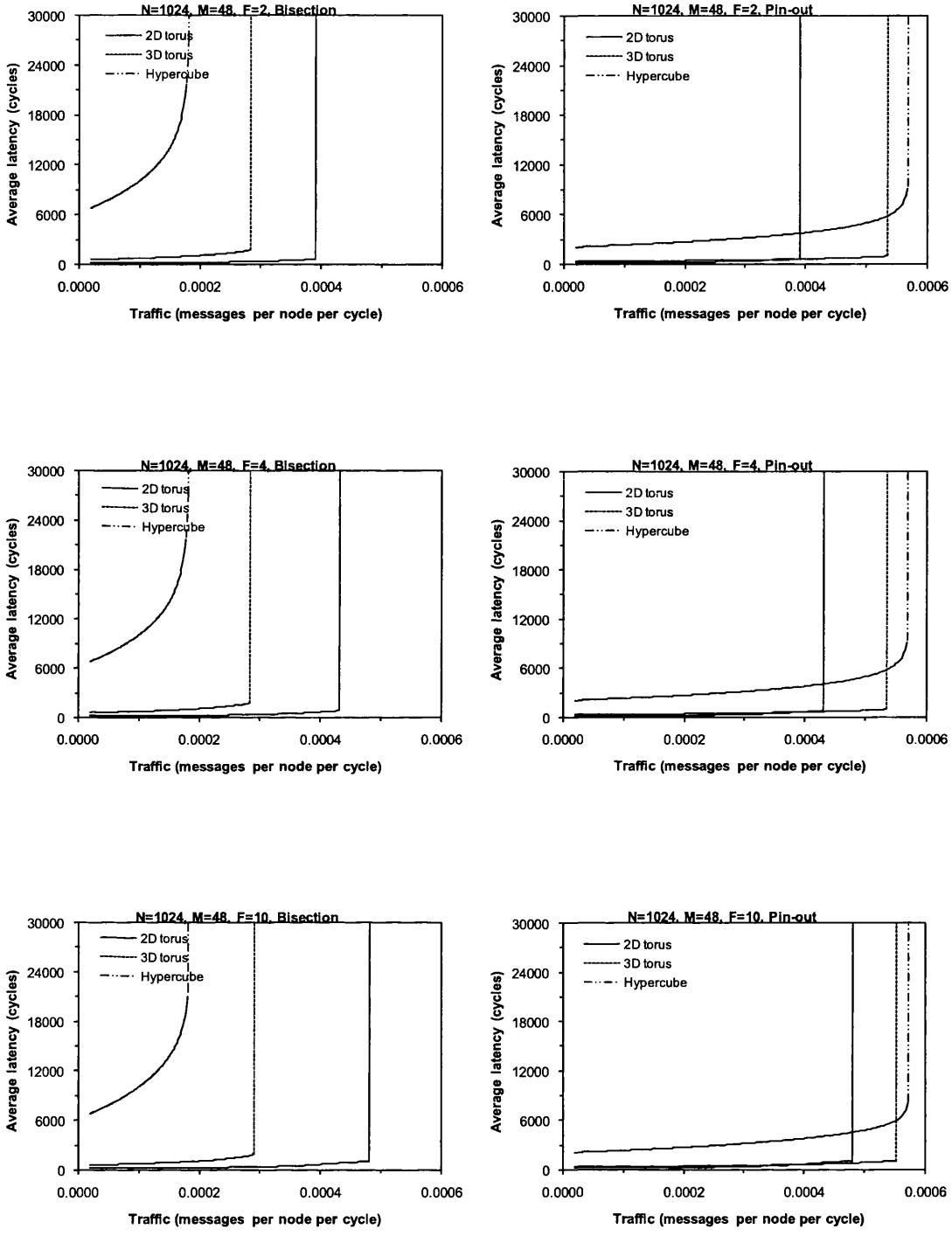
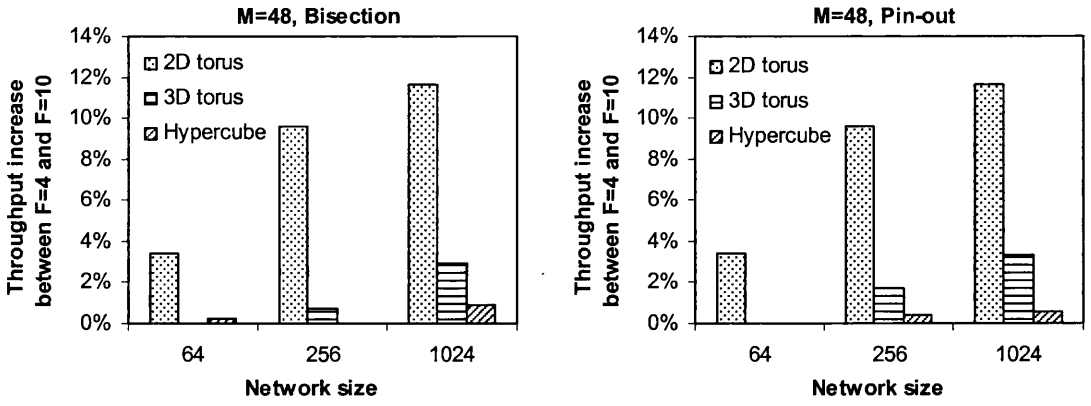


Figure 6.2: The average message latency for three different network topologies (2D torus, 3D torus and hypercube) with N=256 nodes, for message length M=48 flits and different buffer sizes (F=2, 4 and 10 flits) when (a) constant bisection width constraint and (b) pin-out constraint, is imposed.



(a) **(b)**

Figure 6.3: The average message latency for three different network topologies (2D torus, 3D torus and hypercube) with N=1024 nodes, for message length M=48 flits and different buffer sizes (F=2, 4 and 10 flits) when (a) constant bisection width constraint and (b) pin-out constraint, is imposed.



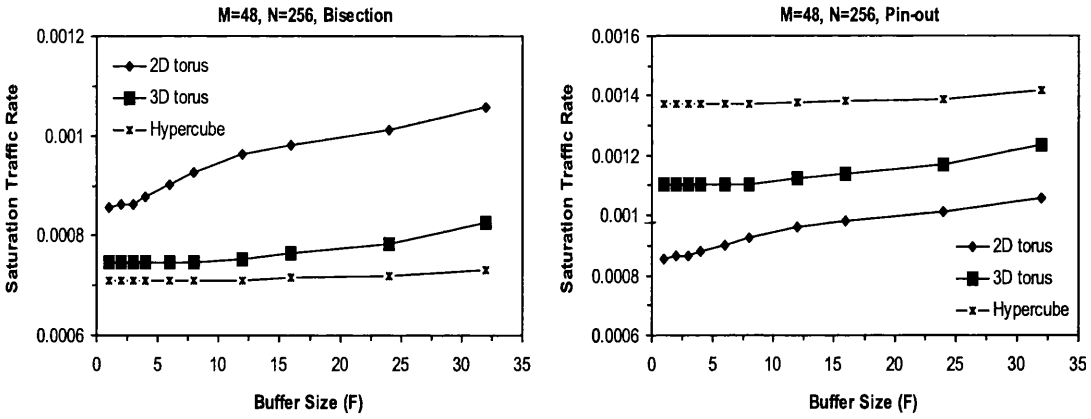
(a) (b)

Figure 6.4: The increase in maximum throughput for 2D torus, 3D torus and hypercube with networks of size 64, 256, and 1024 under (a) constant bisection width constraint and (b) pin-out constraint, with messages of size 48 flits.

Let us now consider the case when the constant pin-out constraint is imposed. Figures 6.1(b), 6.2(b) and 6.3(b) show that higher dimensional networks have superior performance over their lower dimensional counterparts under this constraint. This conclusion differs from that reported by Sarbazi-Azad et al in [130], where they claimed that for small, moderate, and large networks, the topology exhibiting the best relative performance is, respectively, the hypercube, 2D torus, and 3D torus. This discrepancy is due to the following two reasons.

First, the effects of intra-router delay have been accounted for in this study (i.e. routing and switching delays). This factor, as can be seen from Table 6.2, favours the hypercube because it exhibits a lower routing delay compared to the 2D torus and 3D torus. Second, under the constant pin-out constraint, the cycle time of the hypercube is relatively small compared to the constant bisection width constraint. This diminishes the advantages that the finite buffers lend to the lower dimensional networks.

Nonetheless, like the constant bisection width constraint, when the constant pin-out constraint is imposed, lower dimensional k-ary n-cubes tend to exploit the available buffer size more efficiently compared to their higher dimensional counterparts. For instance, when

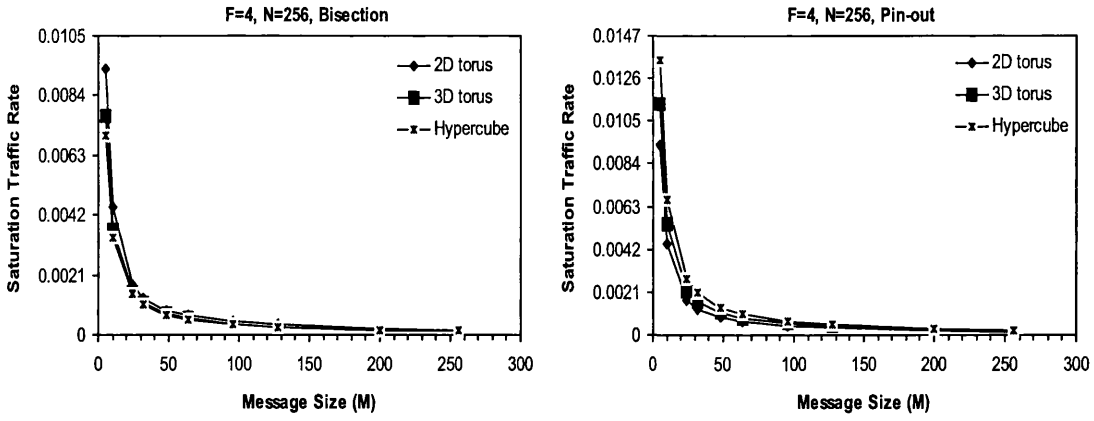


(a) (b)

Figure 6.5: The effect of buffer size, *F*, on the saturation traffic rate in 256-node 2D torus, 3D torus and hypercube under (a) constant bisection width constraint and (b) pin-out constraint, with messages of size 48 flits.

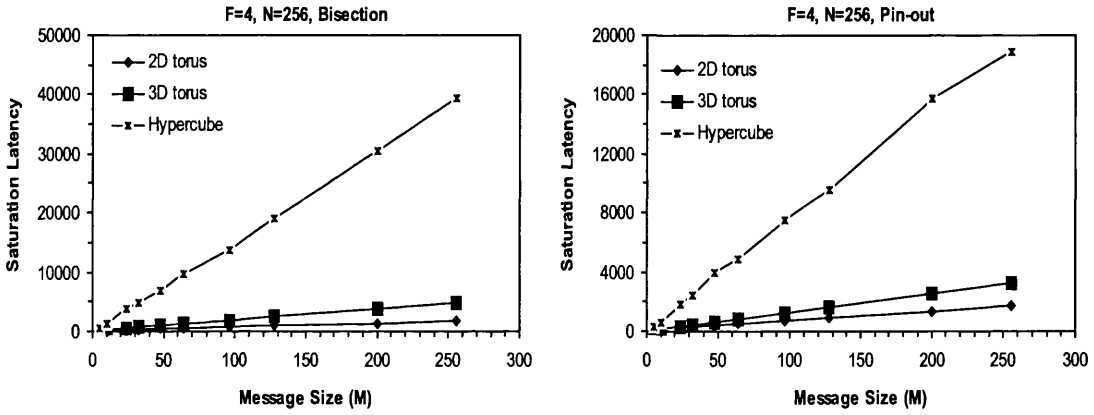
the network is of size 256 nodes, the increase in the maximum throughput of the 2D torus, 3D torus and hypercube is, respectively, 9.62%, 1.69%, and 0.41% when the buffer is increased from 4 to 10 flits. This can be seen clearly in Figure 6.4(b). To further visualise the effects of the buffer size on the maximum throughput of the 2D torus, 3D torus and hypercube, Figure 6.5 plots the saturation traffic rate versus the buffer size for 256-node networks and 48-flit messages. Two important observations can be deduced from this figure.

First, for both the constant bisection width and constant pin-out constraints, the maximum throughput of the hypercube remains almost unchanged as the buffer size increases. This goes side by side with the observation that was made earlier about the marginal increase in the throughput of the hypercube as the buffer size increases for different network sizes. Again this can be accounted for by the smaller average number of hops that a message needs to make to cross the hypercube, compared to the 2D torus and 3D torus. Second, one can deduce safely that under the constant bisection width constraint (Figure 6.5(a)), the 2D torus achieves the highest maximum throughput. Contrarily, under the constant pin-out constraint (Figure 6.5(b)), the hypercube outperforms the 2D torus and 3D torus regardless of the buffer size.



(a) (b)

Figure 6.6: The effect of message size, *M*, on the saturation traffic rate in 256-node 2D torus, 3D torus and hypercube under (a) constant bisection width constraint and (b) pin-out constraint, with buffers of size 4 flits.



(a) (b)

Figure 6.7: The effect of message size, *M*, on the average latency when the saturation traffic is applied to 256-nodes 2D torus, 3D torus and hypercube under (a) constant bisection width constraint and (b) pin-out constraint, with buffers of size 4 flits.

Figure 6.6 depicts the saturation traffic rate as a function of the message size for a 256-node 2D torus, 3D torus and hypercube when the buffer size is 4 flits. The figure reveals that increasing the message size diminishes the advantages of one topology over another in terms of the saturation traffic rate. One factor contributing to this behaviour is that as the message size increases the queuing delay at the source node dominates the overall message latency. Moreover, increasing the message size (while holding the buffer size fixed), increases the

number of channels that the message occupies, resulting in higher blocking probabilities and soon offsetting any advantages of wider channels or lower routing delays.

Finally, Figure 6.7 illustrates the message latency versus the message size when the network operates just below its saturation traffic rate for 256-node networks with 4-flit buffers. A near linear performance trend is observed for the hypercube under both the constant bisection width and constant pin-out constraints. As the message size increases, the queuing delay at the source node dominates the overall message latency creating this linear relation between the message size and the latency when the saturation traffic is applied to the (inherently small diameter) hypercube. Furthermore, for a given message size, the hypercube has a higher saturation latency than the 2D torus and 3D torus, and the difference becomes more noticeable as the message size increases. Moreover, the figure reveals that the message size is less influential in the saturation latency of the 2D torus and 3D torus than of the hypercube. For instance, in a 256-node network with 4-flit buffers and under the constant bisection width constraint, increasing the message size from 10 to 128 flits leads to a 4 times increase in the saturation latency of the 2D torus, whereas there is a 14 times increase in the saturation latency of the hypercube.

6.5 Deep versus parallel buffers

In this section, the analytical models described in Chapters 3, 4 and 5 are used to conduct the first performance comparison, by means of analytical models, of deep versus parallel buffers in adaptively routed wormhole-switched k-ary n-cubes when the total amount of buffer associated with each physical channel is kept constant. The assumption of keeping the physical buffer size constant is necessary to give a fair cost-performance comparison, as it is a limited resource in practical systems.

Since the buffer size allocated to each physical channel has been kept constant, increasing the number of virtual channels results inevitably in a decrease in the buffer size allocated to

each virtual channel. For instance, if a 24-flit buffer, per physical channel, is arranged into 3 virtual channels, then each virtual channel will have 8 flits buffer. This means that a 48-flit message will occupy the buffers of 6 consecutive channels, in the event of blocking. However, if the same amount of buffer (i.e. 24 flits) is arranged into 8 virtual channels, the buffer space allocated to each of them will be only 3 flits. Consequently, in this case, a 48-flit message requires 16 consecutive channels to be fully accommodated, when blocking occurs. The advantage of increasing the number of virtual channels is that the physical bandwidth is utilised more optimally [36]. However, decreasing the buffer depth of the virtual channels causes messages to be distributed over a greater number of routers, resulting in higher blocking probabilities.

Several combinations of experiments have been conducted for different network sizes, message sizes, buffer sizes, and number of virtual channels. For illustration purposes, Figures 6.8 and 6.9 depict latency results in the 2D torus and hypercube, respectively, when the total buffer associated with each physical channel is set to 24, 48 and 96 flits. It should be mentioned that the switching and routing delays have been calculated for all the presented arrangements, using equations (6.1) and (6.2) and (6.11). This is because these delays are expressed as functions of the number of ports in the crossbar switch, which changes from one arrangement of virtual channels to another.

In low dimensional networks (in this case the 2D torus), Figure 6.8 reveals that increasing the buffer size (and therefore decreasing the number of virtual channels) results in a better performance (i.e. a lower latency) for low and moderate traffic rates. This can be observed for almost all cases regardless of the total amount of buffer per physical channel and/or the message size. Similar results have also been reported in [36, 119] when deterministic routing is used. With low traffic loads there are sufficient virtual channels and it is the depth of the virtual channel buffers that have the major impact on performance, as deeper buffers reduce the number of channels occupied by messages and hence reduce the probability of blocking.

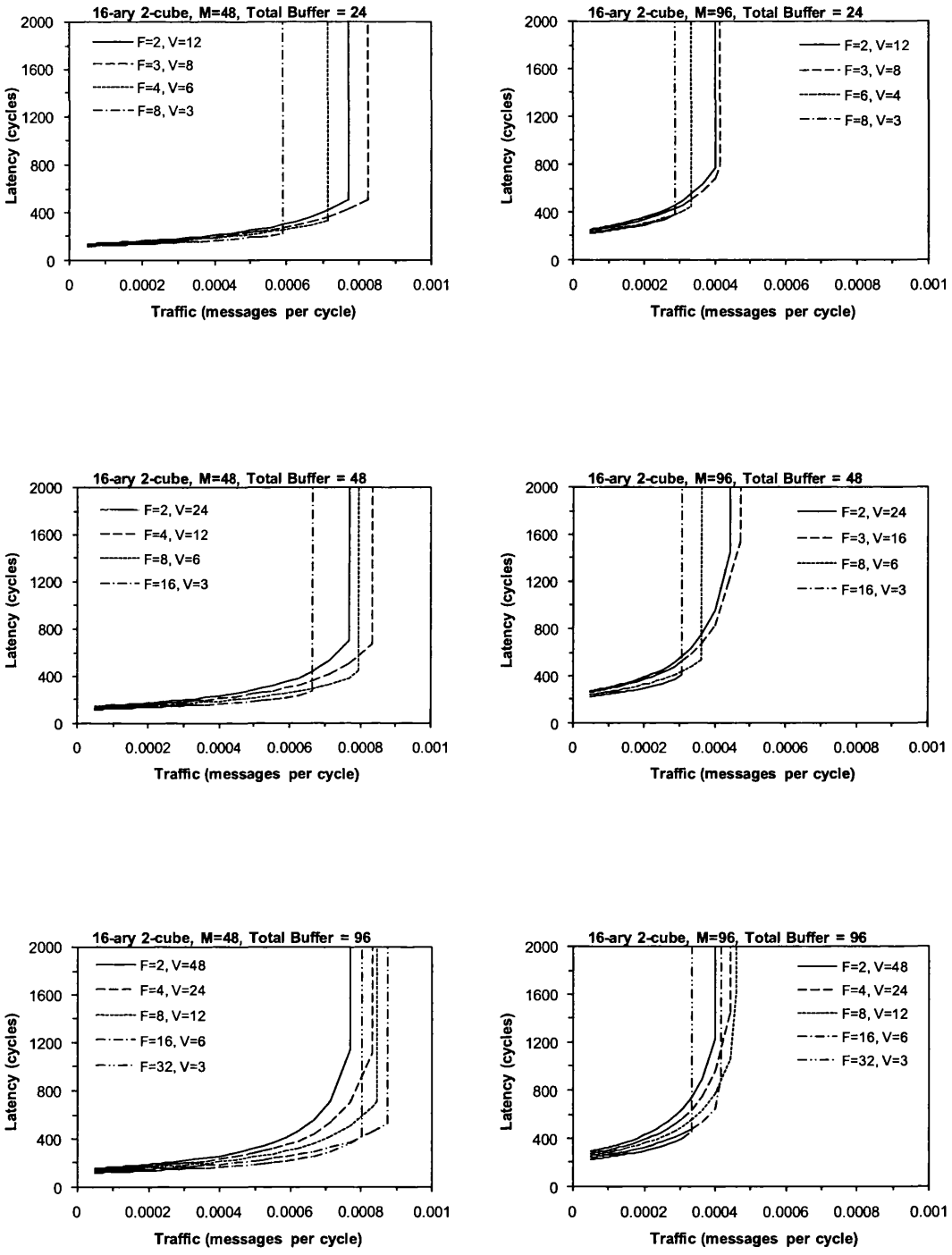


Figure 6.8: The average message latency in a 16-ary 2-cube 2D torus (N=256) for messages of size (a) M=48 flits and (b) M=96 flits when different amount (24, 48 and 96 flits) of total buffer space is associated with each physical channel of the network.

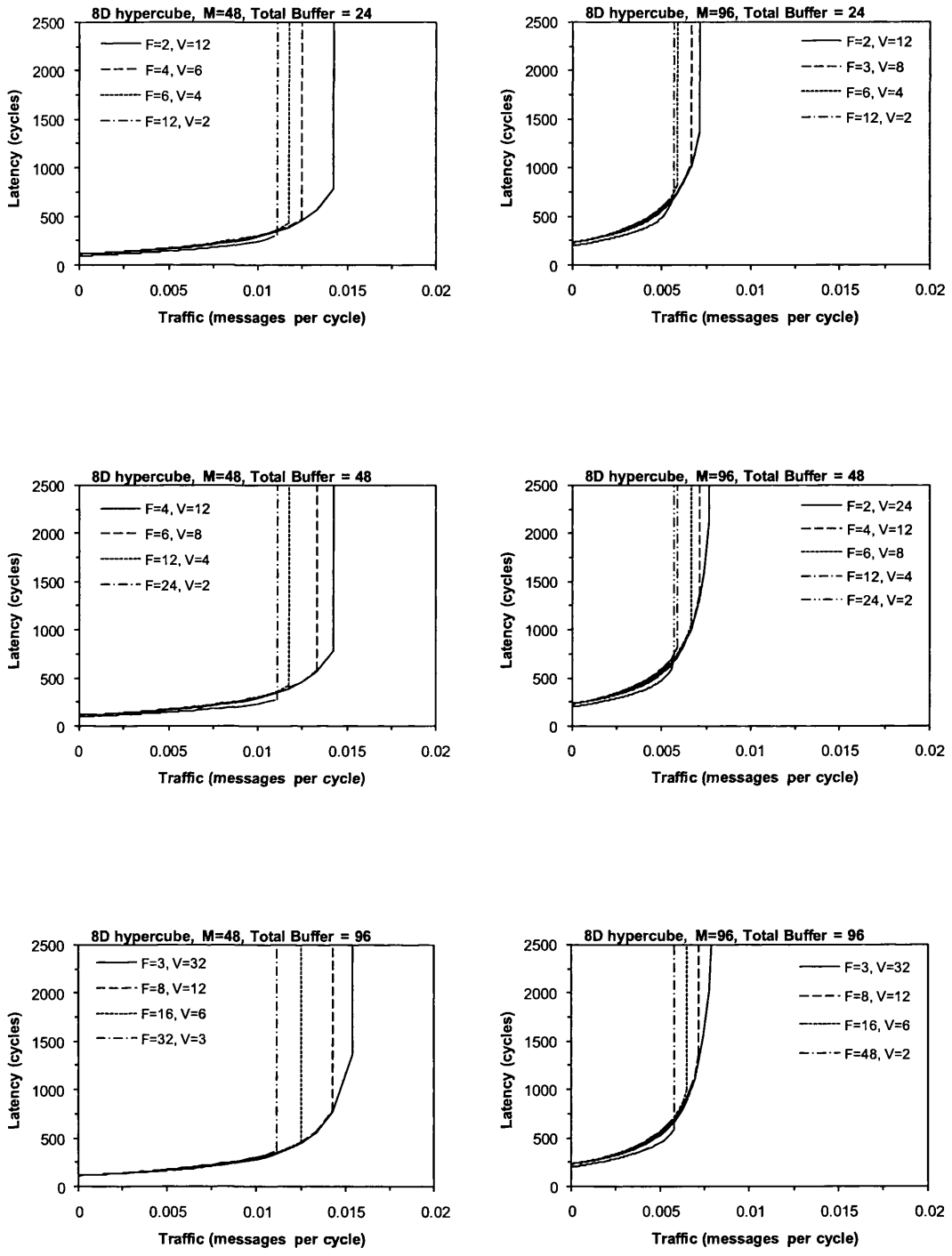


Figure 6.9: The average message latency in an 8-dimensional hypercube (N=256) for messages of size (a) M=48 flits and (b) M=96 flits when different amount (24, 48 and 96 flits) of total buffer space is associated with each physical channel of the network.

However, as traffic rate increases, increasing the number of virtual channels cause some enhancement in the network performance (i.e. the network saturates at higher traffic generation rates). Nevertheless, from a certain threshold, the increase in the number of virtual channels (and hence the decrease in the buffer size of each virtual channel) leads to adverse effects on the performance, as can be noticed from Figure 6.8. For example, for messages of size 48 flits, increasing the number of virtual channels beyond 8 (when the total buffer space is 24 flits) or beyond 6 (when the total buffer space is 96 flits) decreases substantially the saturation traffic rate. Two factors contributing to this effect are apparent. First, the increase in the switching and routing delays diminishes the advantages of having a larger number of virtual channels. This is because as the number of virtual channels increases, the complexity of the router increases too and hence the routing and switching delays increase (this can be seen from equations (6.1) and 6.2)). The second factor is that increasing the number of virtual channels, decreases the buffer sizes, which results in messages occupying larger numbers of channels and, therefore, higher blocking delays.

Figure 6.9 shows that the hypercube (as an example of high dimensional networks) favours the increase in the number of virtual channels as opposed to the increase in the buffer size, especially under moderate and high traffic conditions. This behaviour is expected, as we have already shown in Section 6.4.3 that the hypercube is less sensitive to the increase in the buffer size (see Figure 6.5). This is basically because the low diameter (and hence the smaller average number of hops that messages traverse in the hypercube) offsets the role of moderate size buffers in reducing the number of occupied channels, in the event of blocking. This can be noticed clearly in Figure 6.9 (b) due to the relatively larger message size.

Figure 6.9 reveals that even increasing the buffer size to a point where the message can be accommodated entirely in a small number of channels (for example by using only 2 virtual channels) does not give any substantial performance advantage, especially under moderate and high traffic conditions. This is because in high traffic loads the number of virtual

channels (which is minimised here) becomes the important factor in enhancing performance, as a higher number of virtual channels allows for optimal utilisation of the available physical bandwidth. Finally, it should be noticed that even though lower latencies are achieved with larger buffer sizes, this decrease in the average latency is quite marginal as can be observed from the curves in Figure 6.9. For instance, the average latency of 48-flit messages in the 8D hypercube, with 24-flit buffers per virtual channel, is less than that of the same network with, 4-flit buffers per virtual channel, by less than 20%.

6.6 Adaptive versus deterministic routing

This section is devoted to the comparison of adaptive and deterministic routing algorithms when multiple virtual channels and finite size buffers are used in k-ary n-cubes.

Even though deterministic routing requires fewer virtual channels for deadlock prevention, several studies (see for example [36, 40]) have shown that the use of more virtual channels improves the network performance. Therefore, to conduct a fair comparison it is assumed that both adaptive and deterministic routing algorithms use the same number of virtual channels. This assumption is necessary because it gives both algorithms the same opportunity to utilise an equal number of virtual channels. Moreover, it ensures almost comparable router complexity for both algorithms (i.e. the hardware cost of the routers of both algorithms is kept almost constant).

The switching and routing delays for adaptive routing have already been determined, using equations (6.2) and (6.13 - 6.15). For deterministic routing, however, these delays are different as messages cross the network in a predefined order. This allows the design of simpler deterministic routers by cascading a single crossbar switch for each dimension [26, 48, 101], where each switch either forwards messages in the same dimension or to the next dimension. To compare with the adaptive router design presented in Chapter 2, Figure 6.10 illustrates the cascaded deterministic router proposed in [26, 49].

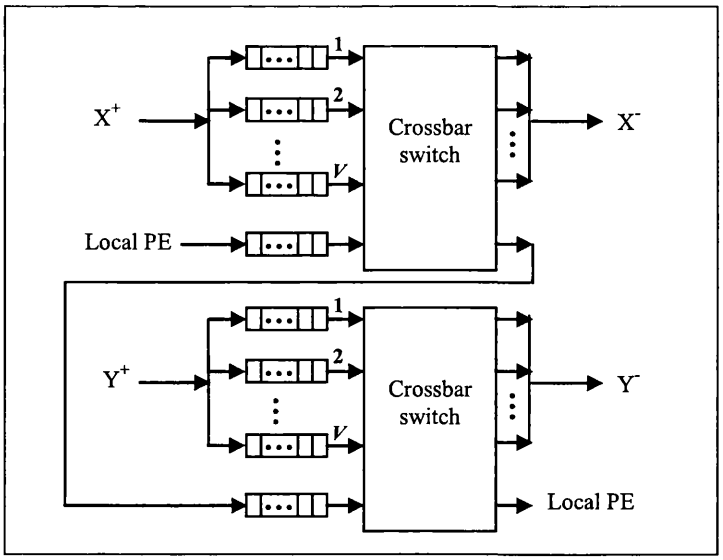


Figure 6.10: A router design for the dimension-order routing algorithm in the 2D torus [26].

Taking into account that dimensions are crossed in order, most messages will continue in the same dimension with at most one switch to the next dimension in the case of the 2D torus. Thus the number of crossbar switches traversed will be one most of the time, and hence, the number of ports in the switch will be $P=(V+1)$. Moreover, because at each routing step the router can forward messages to any virtual channel in the same dimension or to the next dimension, the router has $(V+1)$ alternative channels to choose from (i.e. $R=V+1$). Substituting R and P in equations (6.1) and (6.2) gives the switching and the routing delays of the cascaded deterministic router as

$$t_s = \frac{3.4 + 0.6\log_2(V+1)}{4.9} \tag{6.15}$$

$$t_r = \frac{4.7 + 1.2\log_2(V+1)}{4.9} \tag{6.16}$$

A comparison between the two routing algorithms have been conducted for several combinations of network sizes, message sizes, buffer sizes, number of virtual channels in both the 2D torus (i.e. low dimensional network) and the hypercube (i.e. high dimensional network). For the sake of illustration, Figures 6.11 and 6.12 depict latency results for the following configurations

- Network size N=256 nodes;
- Message size M=48 and 96 flits;
- Number of virtual channels V=3 and 7
- Buffer size F=2, 4 and 10 flits per virtual channel.

Figure 6.11 reveals that under light traffic rates there is no significant difference in the performance of deterministic and adaptive routing in the 2D torus. For instance, the average message latency of the deterministic routing is less than that of the adaptive routing by not more than 35 network cycles (10 %) in a 256-node 2D torus with 48-flit messages, 4-flit buffers and 7 virtual channels. This observation is expected as the blocking is minimised under light traffic conditions. However, deterministic routing achieves slightly lower latency results, apparently, due to the relatively lower switching and routing delays of the deterministic router, especially when the number of virtual channels is relatively large.

As traffic rate increases, however, some differences between deterministic and adaptive routing algorithms are noticed, in the 2D torus. When the buffer size per virtual channel is small, adaptive routing performs better than deterministic routing (i.e. adaptive routing saturates at higher traffic rates), especially with small message sizes. This can be seen in Figure 6.11, which also shows that increasing the buffer size, while keeping the message size fixed, gives an advantage to the deterministic routing algorithm, as can be seen when the buffer size is 10 flits.

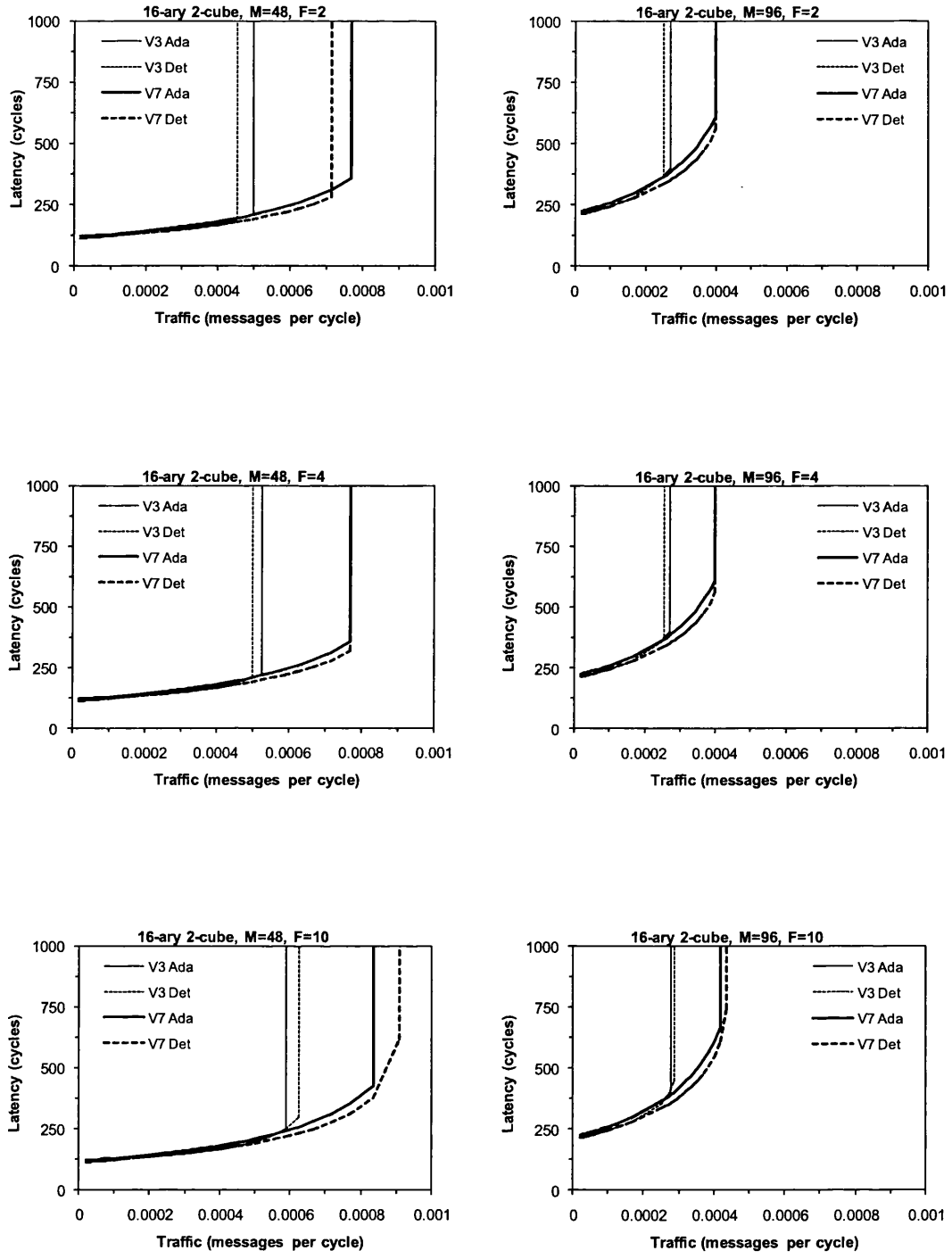
By investigating closely the figures for different message and buffer sizes, it has been noticed that there is a cut-off point after which deterministic routing outperforms (i.e. achieves higher saturation traffic rates) adaptive routing. This point occurs when the number of channels that are necessary to accommodate a full message (i.e. M/F) is less than or equal to the average number of hops that a message needs to make to cross the entire network, $\overline{d}$, (which in the case of 256-node 2D torus is equal to 15 hops). More precisely, when $(M/F) < \overline{d}$, deterministic routing outperforms adaptive routing while adaptive

routing outperforms deterministic routing when $(M/F) > \bar{d}$. More interestingly, when (M/F) is approximately equal to $\bar{d}$, both algorithms achieves comparable performance. From this observation, one can deduce that as the buffer size increases, the number of channels occupied by a message decreases, and hence the probability of blocking decreases too. This diminishes the advantage of adaptive routing of being able to route messages more efficiently when contention is high, and creates a situation similar to that of low traffic conditions. This scenario favours deterministic routing which already has smaller switching and routing delays. It is worth mentioning that previous studies [14, 26, 48, 136] have made little mention of the relationship between buffer size, message size and network size.

For the hypercube, the same observations can be made under light traffic (i.e. the performance of the two algorithms are comparable) as can be seen in Figure 6.12. Under high traffic, however, adaptive routing always has superior performance over deterministic routing, regardless of the number of virtual channels or buffer size, as can be seen in Figure 6.12. This can be chiefly accounted for by the rich connectivity of the hypercube which can be utilised more efficiently by adaptive routing.

It can also be noticed that the buffer size has hardly any effect on the performance of the adaptive and deterministic routing algorithms in the hypercube. This is expected as we have already shown (see Figure 6.5) that the hypercube is less sensitive to the buffer size due to its smaller diameter, and hence smaller average number of hops to traverse the network.

It is important to mention at this point that, unlike the 2D torus, using cascaded routers in the hypercube is not feasible due to scalability problems, i.e. the number of crossbar switches grow linearly as the number of dimensions increase. This is because, in the hypercube, a message makes at most one hop per dimension. Hence, a complete crossbar switch (like the adaptive router) has been assumed when conducting the above comparison which results in an equal switching delay for both routing algorithms.



(a) (b)

Figure 6.11: The average message latency in a 2D torus (N=256) with 3 and 7 virtual channels per a physical channel for messages of size (a) M=48 flits and (b) M=96 flits when the buffer size per a virtual channel is F=2, 4 and 10 flits.

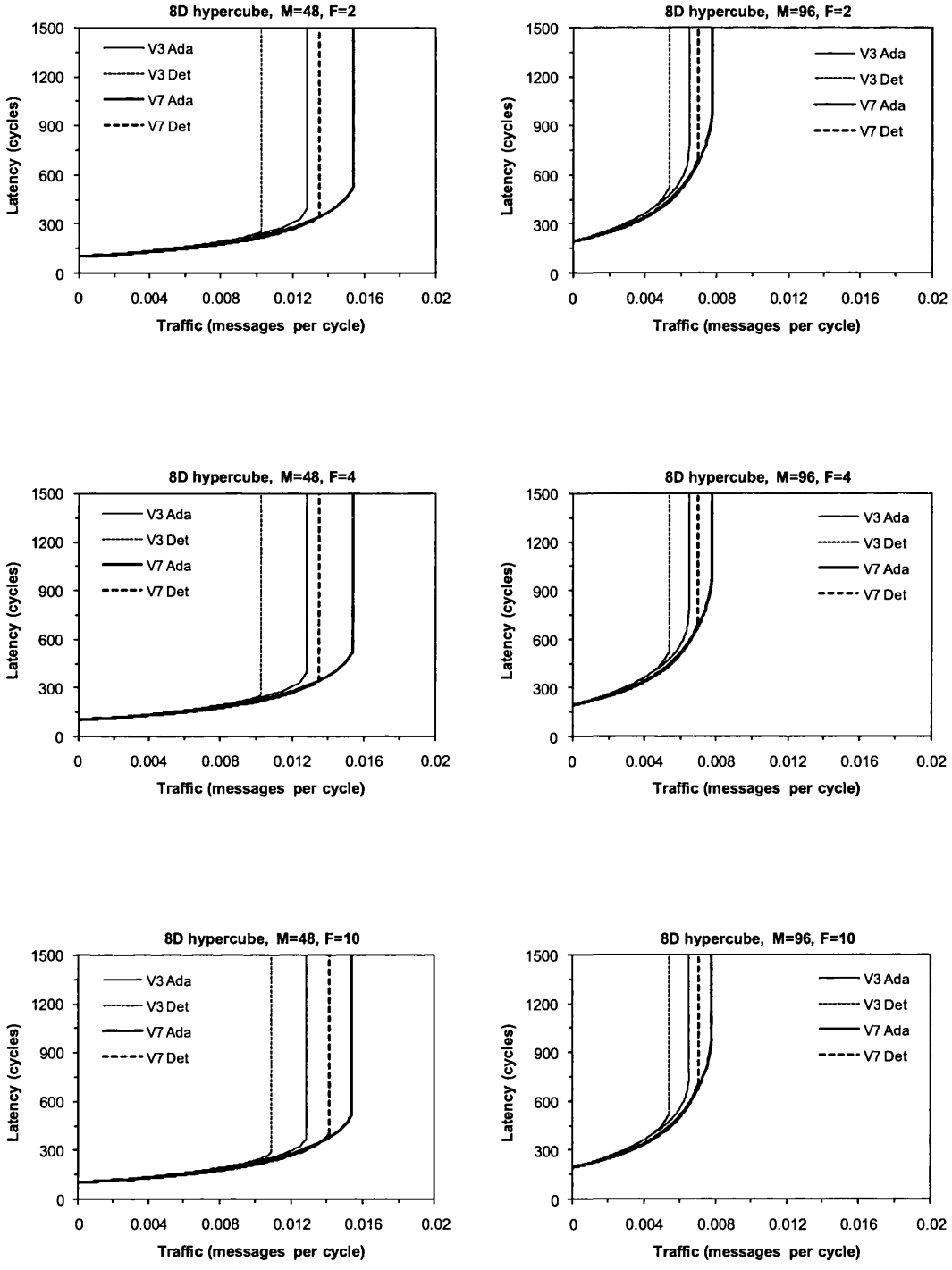


Figure 6.12: The average message latency in an 8-dimensional hypercube (N=256) with 3 and 7 virtual channels per a physical channel for messages of size (a) M=48 flits and (b) M=96 flits when the buffer size per a virtual channel is F=2, 4 and 10 flits.

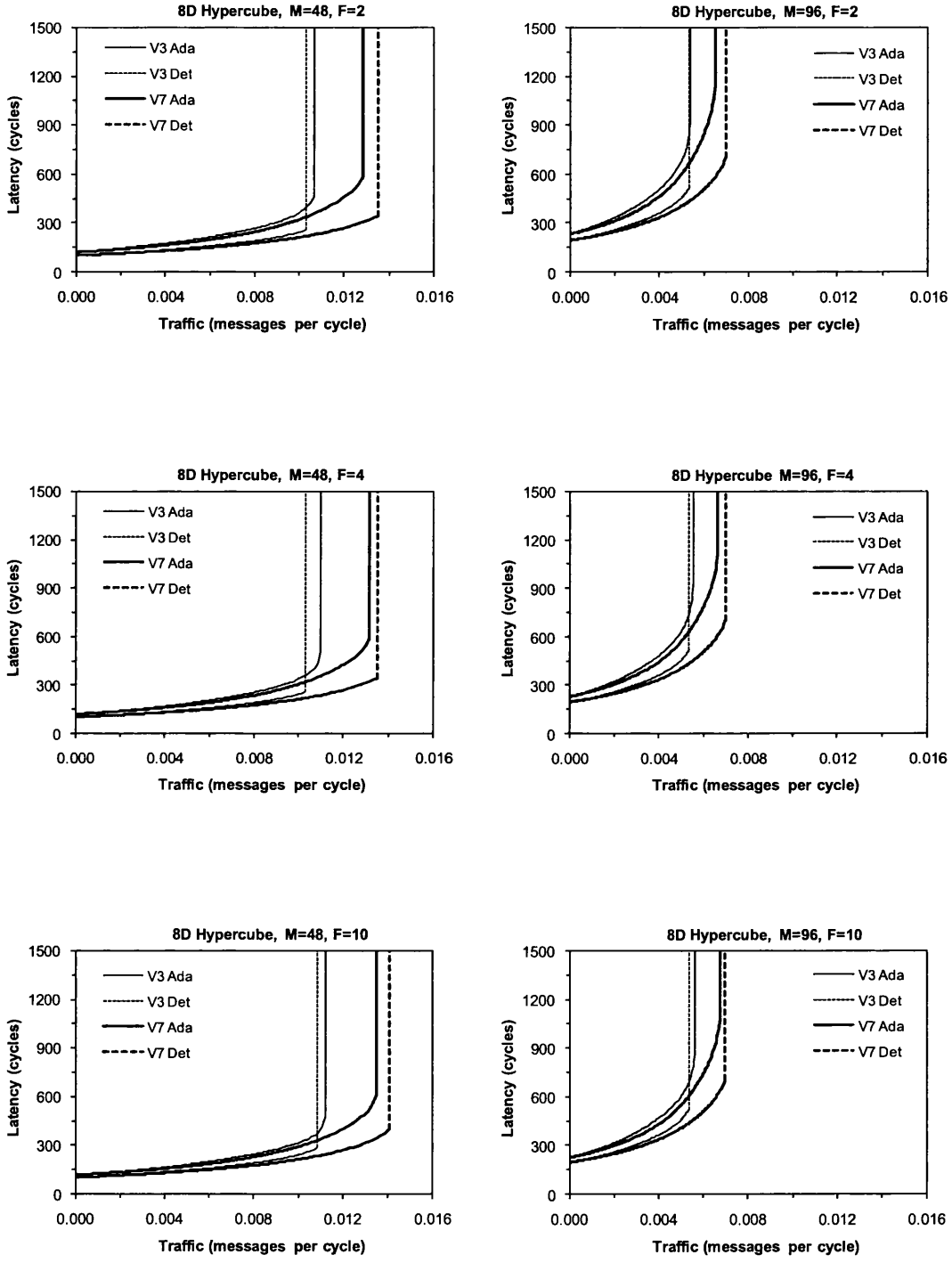


Figure 6.13: The average message latency in an 8-dimensional hypercube with different buffer sizes when a dedicated crossbar switch is used per dimension. Message size is (a) M=48 flits and (b) M=96 flits, virtual channels V= 3 and 7.

Nevertheless, some experiments have been conducted where cascaded routers are used in the hypercube. In such a case, it has been found that deterministic routing achieves better performance when the number of virtual channels is moderately large, as can be seen in Figure 6.13. The relatively higher switching delay of the adaptive router offsets the advantage of the rich connectivity of the hypercube topology, especially when a large number of virtual channels is deployed. For example, the switching delay in the hypercube with 7 virtual channels is higher by 33% than the switching delay in the deterministic routing, while this difference is only 10% for the 2D torus. Another factor contributing to this behaviour could be the role of virtual channels in reducing message blocking inside the network. Note that increasing the buffer size in the 2D torus and increasing the number of virtual channels in the hypercube reduces blocking in the network, a situation which is ideal for deterministic routing. It can also be noticed from the figure that the difference between adaptive and deterministic routing under light and moderate traffic is more apparent compared to the case of the non-cascaded routers. This is due to a higher switching delay in the hypercube when its router is constructed by cascading dedicated per-dimension crossbar switches like the one illustrated in Figure 6.10.

6.7 Conclusions

Many studies have been conducted in the past to analyse the performance of k-ary n-cube topologies, such as the torus and the hypercube, under various working conditions and implementation constraints [2, 5, 26, 35, 36, 48, 72, 96, 101, 112, 119, 130, 136]. However, these studies have resorted mostly to simulation experiments and have either been restricted to deterministic routing or have not considered the effects of finite size buffers. The analysis, presented in this chapter, has relied on the analytical models proposed in Chapters 3, 4 and 5 to assess the performance merits of k-ary n-cubes under different working conditions and implementation constraints when finite size buffers are deployed. Different intra-router delays have been accounted for, in an attempt to make a concrete and fair comparison.

The first part of this chapter has revisited the relative performance merits of the 2D torus, 3D torus and hypercube in the presence of finite size buffers. The analyses demonstrated that under light traffic conditions, the 2D torus is able to exploit its wider channels to provide a lower latency than the 3D torus and hypercube, regardless of the buffer size, network size, and any implementation constraint imposed. This conclusion is in agreement with those reported in previous studies of [2, 5, 35, 130]. However, as traffic increases, the topology exhibiting the best performance depends on the network size and the imposed implementation constraint. When moderate and large size networks are subject to the constant bisection width constraint, lower dimensional networks have been found to exhibit superior performance over their higher dimensional counterparts. However, for small networks, under the same constraint, the hypercube outperforms the 2D and 3D torus. This conclusion is different from those of previous studies [35, 130] which have claimed that lower dimensional networks always outperforms their higher dimensional counterparts under constant bisection width constraint irrespective of network size.

Under the constant pin-out constraint, the hypercube has been found to achieve a better performance than the 2D and 3D torus for all examined network sizes. This conclusion is different from that obtained by the authors in [130], where they have identified that the network exhibiting the best performance depends on the network size. They, however, unlike the present study, have assumed single-flit buffers per virtual channel.

Although it has been noticed that the message size has less influence in the saturation latency of the 2D and 3D than that of the hypercube, it also has been equally observed that for a given message size, the hypercube has higher saturation latency than the 2D and 3D torus. Furthermore, it has been shown the lower dimensional torus is able to exploit the buffer size more efficiently, compared to the higher dimensional hypercube. A near linear performance trend between the message size and the saturation latency has been observed for the hypercube under both constant bisection width and constant pin-out constraints.

The second part of this chapter has investigated the performance of different arrangements of the available buffer space into multiple virtual channels. In low dimensional networks, the results have revealed that increasing the buffer size (and hence decreasing the number of virtual channels) results in better performance under low traffic rates. However, as traffic rates increases, the analysis shows that increasing the number of virtual channels leads to performance enhancement. Nevertheless, these findings agree with previous studies [36, 119] that from a certain threshold, the increase in the number of virtual channels causes performance degradation. Higher dimensional networks, however, tend to favour the increase in the virtual channels as opposed to the increase in the buffer size, in all cases.

The final part of this chapter has compared the performance of adaptive and deterministic routing when finite size buffers are deployed in wormhole-switched k-ary n-cubes. The results have revealed that under light traffic there is no significant difference in the performance of the deterministic and adaptive routing algorithms. As traffic rate increases, however, it has been found that the routing algorithm exhibiting the best performance in the 2D torus depends on the network size, message size and buffer size. When the number of channels required to accommodate an entire message is less than the average number of hops, deterministic routing outperforms adaptive routing while the converse is true when the number of channels required to accommodate an entire message is larger than the average number of hops. In other words, if the buffer size is kept fixed then parallel applications that generate large messages will perform better under adaptive routing. Similarly applications that create small messages will achieve better performance when deterministic routing is used. This is true because small messages require a small number of routers to be fully accommodated which minimises blocking and gives advantage to deterministic routing. Nonetheless, the rich connectivity of the high dimensional hypercube gives advantage to adaptive routing over deterministic routing. Moreover, it has been noticed that the buffer size has little influence on the performance of both adaptive and deterministic routing algorithms in the hypercube, due to its small diameter.

Chapter 7

Conclusions and Future Directions

7.1 Introduction

Parallel computers are generally considered to be the most promising way of achieving the ever-growing computational and storage requirements of many applications, especially in the fields of science and engineering [33, 49, 56, 57, 58, 61, 107]. Today, the performance of parallel computers is limited by their underlying interconnection, not by their logic or memory [42, 49]. A powerful interconnection network offers an attractive solution to this communication bottleneck and is the key to harness fully the computational power offered by parallel computers.

Although many network topologies have been studied extensively, and in some cases even deployed, none has proved clearly superior under all circumstances [42, 49]. This is due mainly to the wide variety of communication requirements exhibited by real-life parallel applications. Nonetheless, k-ary n-cubes have proven to be the most popular interconnection network in experimental and practical multicomputers [3, 6, 53, 68, 84, 89, 105, 109, 110, 133, 140, 147]. They have also emerged as the preferred topology for system-on-chip interconnects [86, 87, 116, 142] due to their desirable properties, including low diameter, ease of implementation, scalability, regularity and recursive structure [42, 49].

Analytical and simulation performance studies of k-ary n-cubes have been reported widely in the literature. This dissertation has extended earlier work in the area by developing new

analytical models to assess the performance behaviour of k-ary n-cubes under different working conditions and system sizes. New performance models have been developed for k-ary n-cubes combining wormhole switching, multiple virtual channels, and finite size buffers. These new models have been validated extensively via simulation experiments.

Even though many analytical models have been reported in the literature (for example, [4, 22, 28, 35, 45, 60, 62, 66, 69, 70, 71, 73, 81, 82, 93, 102, 111, 126, 127, 129]), the models developed in this research have relaxed some of the restrictive assumptions that have been used in the past. Specifically, the commonly used exponential message service time has been relaxed by allowing a general distribution. Furthermore, while most previous models have assumed unrealistically single flit buffers per channel, the models proposed in this dissertation released this constraint and are able to capture the effects of arbitrary buffer size on the performance of k-ary n-cubes.

Previous performance evaluation studies which have considered finite size buffers have been based on simulation experiments [36, 48, 96, 101, 119]. However, to simulate large network sizes requires enormous computational resources in order to model the interaction between various network parameters and their effects on performance. Analytical models are powerful alternatives to such a compute-intensive task and can indeed save considerable time and expense for network designers. In view of these observations, this research has shown that cost-effective analytical models can be constructed with a sufficient level of detail and accuracy to be a practical evaluation tool for wormhole switched k-ary n-cubes with virtual channels and finite size buffers.

Building on the analytical models developed in this research, an extensive comparative study of k-ary n-cubes under different working conditions and implementation constraints have been performed. The relative performance merits of the 2D torus, 3D torus and hypercube when equipped with finite buffering capacity have been revisited. Then, the performance impact of arranging the buffers allocated to a physical channel into multiple virtual channels

has been investigated. Finally, the performance of adaptive routing has been compared to that of deterministic routing in the presence of finite buffers and multiple virtual channels. The analysis has considered router complexity as well as network implementation cost. This study has revealed that capturing the buffering capacity in network routers can lead to significantly different conclusions from those reported by previous research studies. This illustrates the sensitivity of such studies to the level of detail, degree of accuracy, and the realism of the assumptions adopted.

7.2 Summary of the results

The main goal of this dissertation has been to develop new analytical models for wormhole switched k-ary n-cubes with finite size buffers. The major achievements made during this research are summarised below.

- A new general method to determine the virtual channel occupancy probabilities in wormhole-switched k-ary n-cubes has been developed. Existing analytical models have relied on a method proposed by Dally [36], based on Markov chains, to calculate these probabilities. The new general method is based on inverting the z-transform of the number of customers in an M/G/1 queuing system. Dally's method can be derived as a special case of the new method when the service time is assumed to be distributed exponentially. Furthermore, a two-moment matching method has been adopted to approximate the service time distribution when only the first two moments of the service time are available. Together with its accuracy under different traffic conditions, the new method is general enough that it can be integrated easily into almost any analytical model that needs to model virtual channel multiplexing.

- A new analytical model that can capture the effects of finite buffers on the performance of wormhole-switched k-ary n-cubes with deterministic routing has been presented. This is the first model to be reported in the literature that considers

the use of multiple virtual channels per physical channel. Capturing the effects of finite buffers complicates the derivation of the analytical model due to the fact that messages no longer necessarily span the entire path between the source and destination nodes. Moreover, messages may get blocked not only at the head of a given buffer when contending for virtual channels but also inside the buffer.

The derivation of the model has been described in detail for the bidirectional torus. However, modifications to the model, in order to extend it to the unidirectional torus and hypercube, have also been outlined. The model has been validated via simulation experiments and the results have demonstrated its reasonable accuracy under various network operating conditions.

- The first analytical model for computing message latency of adaptive routing in wormhole-switched k-ary n-cubes with finite buffers and multiple virtual channels has been developed. The routing algorithm used in the development of the model is Duato's adaptive routing algorithm [46], which has been studied widely and deployed in experimental and commercial multicomputers [3, 22, 38, 49, 103, 111, 127, 133]. However, the proposed modelling approach can be applied to other routing algorithms in a straightforward manner. The model exhibits good accuracy because, firstly, it takes account of the two types of blocking that contribute to message latency; one due to contention for virtual channels and the other due to lack of sufficient buffer space. Secondly, the model computes different components that make up the average message latency, including routing delay, blocking delay and source delay, as a function of the number of effective channels. This is because only a limited number of channels are affected by message blocking.

  The results predicted by the model have been found to be in close agreement with those obtained from an event-driven simulator. The simplicity and the accuracy of

the model make it a practical evaluation tool for studying the performance of adaptive routing in wormhole-switched k-ary n-cubes with finite buffering.

- The analytical models developed have been used to conduct an extensive investigation of the relative performance merits of k-ary n-cubes under different working conditions and implementation constraints when finite size buffers are used. Unlike previous studies, different intra-router delays (i.e. switching delay and routing delay) have been taken into consideration in an attempt to make a concrete and fair comparison. The relative performance merits of different k-ary n-cube topologies (2D torus, 3D torus and hypercube) with different sizes were compared under the constant bisection width and constant pin-out constraints.

  The analysis demonstrated that under light traffic conditions the 2D torus is able to exploit its wider channels to provide a lower latency than the 3D torus and hypercube, regardless of the buffer size, network size and any implementation constraint imposed. This conclusion is in agreement with those reached previously in [2, 5, 35] for deterministic routing and in [130] for adaptive routing. However, as traffic increases, the topology exhibiting the best performance depends on the network size and implementation constraint. When moderate and large size networks are subject to the constant bisection width constraint, lower dimensional networks have been found to exhibit superior performance over their higher dimensional counterparts. However, for small networks, under the same constraint, the hypercube outperforms the 2D and 3D torus. This conclusion is different from that of previous studies [35, 130] which have claimed that lower dimensional networks always outperforms their higher dimensional counterparts under constant bisection width constraint irrespective of the network size. This demonstrates clearly that the presence of finite buffers can alter the outcome of investigating the relative performance merits of competing network topologies.

- Under the constant pin-out constraint, the hypercube has been found to achieve better performance than the 2D torus and 3D torus for all examined network sizes. This result differs from that obtained by the authors of [130] who, by assuming single-flit buffers, have concluded that the topology exhibiting the best performance depends on network size. This discrepancy again shows the importance of properly modelling the effect of node buffering.

- Although it has been noticed that the message size has less influence on the saturation latency of the 2D and 3D torus than on that of the hypercube, it has also observed equally that, for a given message size, the hypercube has higher saturation latency than the 2D and 3D torus. A near linear performance trend between the message size and the saturation latency has been observed for the hypercube. Similar results have also been noticed when the constant pin-out constraint is imposed. The analysis has also identified that the lower dimensional k-ary n-cubes are able to exploit buffer size more efficiently compared to their higher dimensional counterparts. Such conclusions have not been reported in previous similar studies [2, 5, 35, 130] of k-ary n-cubes because chiefly they have constrained the buffer depth to a single flit.

- The analysis has been extended to investigate the optimal arrangement of the total buffer space allocated to each physical channel when multiple virtual channels are used. In low dimensional networks (2D and 3D torus) the results have revealed that increasing the buffer size (and hence decreasing the number of virtual channels) results in better performance under low and moderate traffic rates. In such cases, there are sufficient virtual channels and it is the depth of the virtual channel buffers that have the major impact on the performance, as deeper buffers reduces the number of channels occupied by messages and hence reduces the probability of blocking. However, as traffic rates increase, it has been observed that increasing the number of

virtual channels leads to performance enhancement. Nevertheless, the present analyses agree with previous studies [36, 119] that from a certain threshold, the increase in the number of virtual channels causes adverse effects on performance.

- On the other hand, higher dimensional networks (e.g. the hypercube) favour the increase in the number of virtual channels as opposed to the increase in the buffer size. This is because the low diameter (and hence the smaller average number of hops that messages traverse in the hypercube) offsets the role of moderate size buffers in reducing the number of occupied channels between the source and destination nodes. It should be mentioned that although adaptive routing was considered, the results, obtained using the new analytical models, are in line with previous results [36, 119], obtained via simulation for the dimension-order routing. This conclusion demonstrates that even though adaptive routing requires a higher number of virtual channels for deadlock prevention, as in deterministic routing, virtual channels should not exceed a specific threshold due to increased implementation cost with minor performance returns.

- The performance of adaptive routing has been compared against that of deterministic routing when an equal number of virtual channels and finite size buffers are employed in k-ary n-cubes. For lower dimensional networks, under light traffic there is no significant difference between the performances of deterministic and adaptive routing with deterministic routing achieving marginally lower latency results. As the traffic rate increases, however, the routing algorithm exhibiting the best performance depends on the number of consecutive channels required to accommodate an entire message and on the average number of hops to cross the network (i.e., it depends on the buffer size, message size, and network size). When the number of channels required to accommodate an entire message is less than the average number of hops to cross the network, deterministic routing outperforms adaptive routing and the

converse is true when the number of channels required to accommodate a full message is larger than the average number of hops to cross the network. It is worth mentioning here that previous network performance studies [14, 26, 48] have made little mention of the relationship between buffer size, message size and network size.

- Finally, the results have shown that the buffer size has little effect on the performance of either adaptive or deterministic routing in higher dimensional networks (e.g. the hypercube). It has also been identified that the rich connectivity of the hypercube gives an advantage to adaptive over deterministic routing as the former outperforms the latter regardless of the buffer size and the number of virtual channels.

  These conclusions are different from those reported in the past [14, 26, 48] where a higher number of virtual channels has been allocated in the case of adaptive routing, giving it the advantage in network resource utilisation over deterministic routing (a bias which gives the former superior performance in all cases). In this study, however, it has been assumed that both approaches organise the available buffer space into the same number of virtual channels to ensure a fair comparison.

7.3 Directions for future work

As a result of this research, a number of interesting issues and open problems have been identified that could be pursued in future investigations. These can be grouped into two broad categories: (1) those which make the proposed models more realistic, and (2) those which tackle other general issues in interconnection networks.

7.3.1 Developing more realistic models

There are a number of suggestions as how the proposed models might be modified to capture more realistic situations. Some of these are outlined below.

- The analytical models presented in this dissertation have assumed a uniform traffic pattern (i.e. all nodes receive messages with equal probability). This is due mainly to the added complexity of considering finite buffers and also due to time limitations. However, there are real parallel applications that exhibit non-uniform traffic patterns (i.e. some nodes are likely to receive more messages than others). Typical examples of non-uniform traffic are generated by the presence of hotspots in the network, but also naturally by certain scientific computations such as matrix transpose and digit traversal [59, 118]. Although there have been a number of studies [113, 127, 128, 129] that have attempted to address this issue, they have assumed single flit buffers. A possible line of research would be to extend the proposed analytical modelling approaches to deal with common non-uniform traffic patterns.

- A number of research studies have revealed that the Poisson model cannot properly emulate the traffic characteristics of some practical applications, especially those involving multimedia data such video and audio. Self-similar and pseudo self-similar traffic models have been shown to be a more realistic alternative [32, 54, 78, 123]. A useful extension to this work would be to devise new analytical models that can capture the effects of finite size buffers in the presence of self-similar traffic.

- Incorporating fault-tolerant techniques is of great importance to many parallel machines [49, 63, 91]. Apart from a few attempts [121, 122], most existing analytical models have been discussed in the context of fault-free networks. Another direction of research along the broad lines of this thesis would be to develop analytical models to investigate the effects of using finite size buffers in fault-tolerant routing.

- Several recent studies have revealed that deadlocks occur very infrequently in the network, and this has motivated researches to devise deadlock recovery routing algorithms (e.g. compression-less [74], software-based [97] and Dish [11] routing algorithms). There have been attempts to develop analytical models for some of these

deadlock recovery routing algorithms [69, 70, 71] but these studies have assumed single flit buffers. Modelling finite buffers in deadlock recovery routing algorithms may lead to substantially different insights. Another research direction would be to devise new analytical models for deadlock recovery routing algorithms with finite size buffers.

- In this research a FIFO allocation strategy has been assumed when modelling the effects of virtual channels multiplexing. However, several other strategies can be used to allocate the channel bandwidth to multiple virtual channels including round-robin, random, priority, or deadline scheduling [36]. Another research direction would be to extend the proposed models to capture the effects of some of these allocation strategies.

7.3.2 Future research in interconnection networks

Moving beyond the core of the present work, there remain many interesting issues in the field which would benefit from the same analytical approach adopted in this research. A selection of such issues is listed below as an illustration of the potential of this line of research.

- In this work, the focus has been on modelling the performance of interconnection networks for parallel computers. However, interconnection networks have been suggested as an alternative to buses in system-level interconnection [42]. In this role they would replace dedicated wiring in special-purpose systems, exploiting the fact that routing messages via an interconnection network is faster and more economic than dedicated wiring [41, 99]. K-ary n-cubes are emerging as the preferred topology in such network-interconnected system-on-chip architectures [86, 87, 116, 142]. An interesting research avenue would be to devise new analytical models for these architectures and assess their performance behaviour.

- Virtual cut-through switching [67] has been introduced as an enhancement to packet switching in order to reduce the transmission time. Communication latency, especially under low and moderate traffic loads, is reduced noticeably in virtual cut-through. This is due to the fact the blocked messages are accommodated entirely at the point of blocking, freeing the channels to be utilised by other non-blocked messages [49, 103]. A natural extension to this research work would be to devise new analytical models to assess the performance of virtual cut-through switching, which differs substantially from wormhole switching in terms of buffer requirements.

- Unlike point-to-point communications (i.e. single sender and single receiver), in collective communications, such as multicast (one-to-many) and broadcast (one-to-all), a group of nodes is involved in exchanging messages. The importance of this class of traffic patterns is derived from the fact that many parallel applications such as sorting, process control, data migration, searching and matrix applications require collective communication operations [25, 44, 100, 135]. It would be interesting to develop analytical models for multicast and broadcast routing algorithms in wormhole-switched networks with finite buffers and virtual channels.

Appendix A1

A Latency Model for Adaptive Routing in k-Ary n-Cubes

In [111] Ould-Khaoua developed an analytical model to compute the average message latency in wormhole-switched *k*-ary *n*-cubes with adaptive routing, based on Duato's adaptive routing algorithm [46]. As discussed in Chapter 3, any latency model should adopt a method to model virtual channels, if it needs to consider their effects on network performance. The model in [111] has relied on a method proposed by Dally [36] to model virtual channels. Chapter 3 described a new model for virtual channels and argued that the predictions of the latency model could be enhanced by using the new virtual channels model. For this dissertation to be self-contained, this appendix summarises the main equations of the latency model presented in [111]. The interested reader is referred to [111] for a detailed derivation of the model equations.

The model starts by calculating the average number of hops that a message makes to cross one dimension, $\bar{k}$, and to cross the entire network, $\bar{d}$, as

$$\bar{k} = (k-1)/2 \qquad \text{(A1.1)}$$

$$\bar{d} = n\bar{k} \qquad \text{(A1.2)}$$

The mean network latency (i.e. time to cross the network) is then computed as

$$S = M + \bar{d} + \sum_{i=1}^{\bar{d}} \left(P_{bi} W_b \right) \qquad \text{(A1.3)}$$

In the above equation, P_{bi} is the probability of blocking at the i^{th} hop and W_b is the mean waiting time. The probability of blocking at the i^{th} hop is given by

$$P_{bi} = \begin{cases} P_d (P_a)^{n-1} & 1 \le i \le \bar{k} \\ \sum_{l=0}^{j} P_i^j P_d (P_a)^{n-1-l} & j\bar{k}+1 \le i \le (j+1)\bar{k},\ 1 \le j \le n-1 \end{cases} \quad \text{(A1.4)}$$

P_a, in the above equation, is the probability that all adaptive virtual channels (according to Duato's routing algorithm) are busy and is given by

$$P_a = P_V + (2P_{V-1}) / \binom{V}{V-1} + (P_{V-2}) / \binom{V}{V-2} \quad \text{(A1.5)}$$

Likewise, P_d is the probability that all adaptive and the usable deterministic virtual channel are busy and is given by

$$P_d = P_V + (2P_{V-1}) / \binom{V}{V-1} \quad \text{(A1.6)}$$

The probability that a message has crossed j dimensions at its i^{th} hop is given by

$$P_i^j = \begin{cases} \dfrac{\binom{n}{j} N_0^{\bar{k}-1}(i-1-j\bar{k}, n-j)}{\sum_{l=0}^{j} \binom{n}{l} N_0^{\bar{k}-1}(i-1-l\bar{k}, n-l)} & ,\ 1 \le j \le n-1 \\ 1 - \sum_{l=1}^{n-1} P_i^l & ,\ j = 0 \end{cases} \quad \text{(A1.7)}$$

The number of ways to distribute r hops over m dimensions such that the number of hops in each dimension is at least p and is at most $p+q-1$, $N_p^{p+q-1}(r,m)$, is given by

$$N_p^{p+q-1}(r,m) = \sum_{l=0}^{m} (-1)^l \binom{m}{l} \binom{r-mp-lq+m-1}{m-1} \quad \text{(A1.8)}$$

The mean waiting time when a message is blocked is given by

$$W_b = \frac{\lambda_c S^2 \left(1 + (S - M)^2 / S^2\right)}{2(1 - \lambda_c S)} \tag{A1.9}$$

where the rate of messages received by each channels, λ_c, is given by

$$\lambda_c = \left(\lambda_g \bar{d}\right) / n \tag{A1.10}$$

The mean waiting time at the source is approximated as

$$W_s = \frac{\left(\lambda_g / V\right) S^2 \left(1 + (S - M)^2 / S^2\right)}{2\left(1 - \left(\lambda_g / V\right) S\right)} \tag{A1.11}$$

The average degree of virtual channel multiplexing is computed as

$$\bar{V} = \frac{\sum_{i=0}^{V} i^2 P_i}{\sum_{i=0}^{V} i P_i} \tag{A1.12}$$

Notice again hat, P_i, in equations (A1.6), (A1.7) and (A1.12), represents the probability of i busy virtual channels. These probabilities are calculated in [111] using the method proposed by Dally [36] whereas this work proposes to compute them using the new method proposed in Chapter 3.

Finally, the summation of the mean network latency and the mean waiting time at the source node, is scaled by the average degree of virtual channel multiplexing to give the average message latency

$$Latency = (S + W_s)\bar{V} \tag{A1.13}$$

Appendix A2

Inverting the *z*-Transform

Different methods of inverting the *z*-transform in equation (3.8, Chapter 3) are available in the literature [19]. For example, Partial fractions and power series expansion methods are used in [108] to invert the transform when various service time distributions are assumed. However, sometimes it is hard or even impossible to invert the resulting *z*-transform using these methods. This appendix outlines a simple algorithmic approach to invert the *z*-transform of the number of customers in the M/G/1 queuing system.

A2.1 The embedded chain

Consider the embedded Markov chain, N_k (i.e. the queue length immediately after the departure epochs). The number of customers left in the queue immediately after the departure of the m^{th} customer from the system is given by [77]

$$N_{m+1} = \begin{cases} L & N_m = 0 \\ L + N_m - 1 & N_m \geq 1 \end{cases} \qquad \text{(A2.1)}$$

In the above equation, L represents the number of new arrivals during the service time of customer $m+1$. Let α_i be the probability that there are $i\,(0 \leq i < \infty)$ new service requests during the service time of a customer and $f_s(x)$ be the probability density function of this service time. According to the law of total probability, we can write [77]

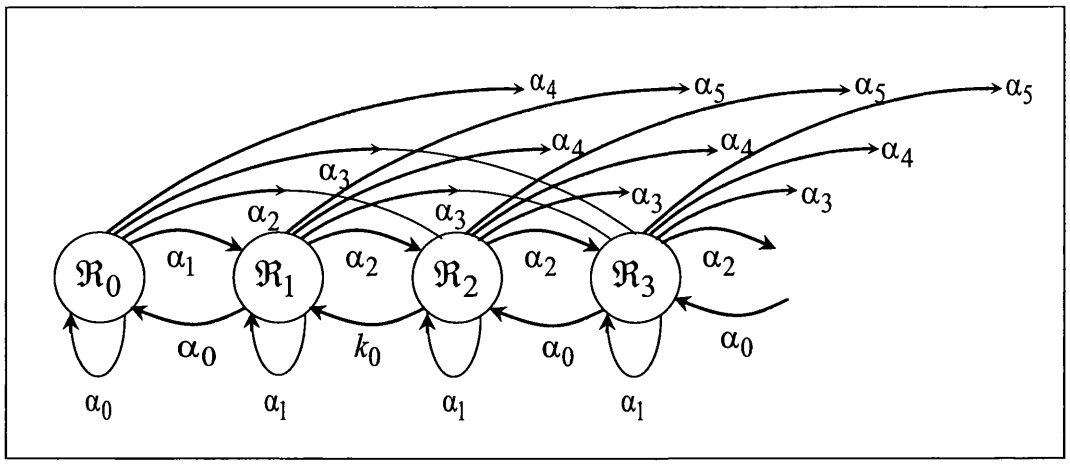


Figure A2.1: The state diagram of the number of requested virtual channel

$$\begin{aligned}\alpha_i &= \Pr[L=i] \\ &= \int_0^\infty \Pr[L=i \mid S=x] f_s(x)dx \\ &= \int_0^\infty \frac{(\lambda_c x)^i}{i!} e^{-\lambda_c x} f_s(x)dx\end{aligned} \tag{A2.2}$$

A2.2 Transition probabilities

Figure A2.1 shows the state diagram of the number of busy virtual channels at a given physical channel where $\Re_v$ correspond to $v\,(0 \le v \le V)$ busy virtual channels. The transition probability from state $\Re_i$ (i customers in the system) to state $\Re_j$ (j customers in the system) is defined as the probability that there are j customers in the system after the departure of the $(m+1)^{th}$ customer, given that the number of customers in the system after the departure of the m^{th} customer was i. Using the evolution equation of the chain, N_k, (equation A2.1), one can identify the transition probabilities as [77]

$$\begin{aligned}P_{i,j} &= \Pr[N_{m+1}=j \mid N_m=i] \\ &= \begin{cases}\Pr[L=j-i+1]=\alpha_{j-i+1} & i \ge 1, j \ge 0 \\ \Pr[L=j-i]=\alpha_{j-i} & i=0, j \ge 0\end{cases}\end{aligned} \tag{A2.3}$$

Hence the transition matrix for all values of i and j can be written as

$$P=\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots \\ 0 & \alpha_0 & \alpha_1 & \alpha_2 & \cdots \\ 0 & 0 & \alpha_0 & \alpha_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{A2.4}$$

The global balance equations (when the system is in the steady state) yield the following recursive equations [77]

$$\begin{aligned} \alpha_0\pi_1 &= (\alpha_1+\alpha_2+\cdots)\pi_0 \\ \alpha_0\pi_2 &= (\alpha_2+\alpha_3+\cdots)\pi_0+(\alpha_2+\alpha_3+\cdots)\pi_1 \\ \alpha_0\pi_3 &= (\alpha_3+\alpha_4+\cdots)\pi_0+(\alpha_3+\alpha_4+\cdots)\pi_1+(\alpha_2+\alpha_3+\cdots)\pi_2 \\ &\vdots \\ \alpha_0\pi_i &= (\alpha_i+\alpha_{i+1}+\cdots)\pi_0+(\alpha_i+\alpha_{i+1}+\cdots)\pi_1+\ldots+(\alpha_2+\alpha_3+\cdots)\pi_{i-1} \end{aligned} \tag{A2.5}$$

A2.3 Recursion

Now equation (A2.5) can be solved recursively for $\pi_i\ (0\le i\le V)$ starting from the known value $\pi_0=1-\rho$, where ρ is the server utilisation. Let a_i be the probability that the number of new service requests is grater than *i*. Then a_i is determined by

$$\begin{aligned} a_i &= \Pr[L>i] \\ &= \sum_{j=i+1}^{\infty}\alpha_j \end{aligned} \tag{A2.6}$$

Since $\alpha_0+\alpha_1+\alpha_3+\ldots=1$, we have

$$\begin{aligned} \alpha_0 &= 1-(\alpha_1+\alpha_2+\alpha_3+\ldots) \\ &= 1-a_0 \end{aligned} \tag{A2.7}$$

Hence, the probability of the number of customers in the M/G/1 queue can be computed recursively. The recursion begins from $\pi_0=1-\rho$ and the general recursion step is given by

$$\pi_i=\frac{1}{1-\rho}\left(a_{i-1}(1-\rho)+\sum_{j=1}^{i-1}a_{i-j}\pi_j\right) \tag{A2.8}$$

References

[1] S. Abraham and K. Padmanabhan, "Performance of the direct binary n-cube network for multiprocessors", IEEE Transactions on Computers, vol. 38, no.7, pp. 1000-1011, 1989.

[2] S. Abraham and K. Padmanabhan, "Performance of multicomputer networks under pin-out constraints", Journal of Parallel and Distributed Computing, vol. 12, no.3, pp. 237-248, 1991.

[3] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, et al., "Blue Gene/L torus interconnection network", IBM Journal of Research and Development, vol. 49, no.2-3, pp. 265-276, 2005.

[4] V. S. Adve and M. K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no.3, pp. 225-246, 1994.

[5] A. Agarwal, "Limits on interconnection network performance", IEEE Transactions on Parallel and Distributed Systems, vol. 2, no.4, pp. 398-412, 1991.

[6] A. Agarwal, R. Bianchini, D. Chaiken, F. T. Chong, K. L. Johnson, D. Kranz, et al., "Mit alewife machine", Proceedings of the IEEE, vol. 87, no.3, pp. 430-444, 1999.

[7] N. Alzeidi, A. Khonsari, M. Ould-Khaoua, and L. M. Mackenzie, "On the probability distribution of busy virtual channels," Proceedings of the 20th International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, 2006.

[8] N. Alzeidi, M. Ould-Khaoua, and A. Khonsari, "Worm latency in hypercube with finite buffers," Proceedings of the 20th UK Performance Engineering Workshop, pp.94-103, Bradford, UK, 2004.

[9] T. R. Andel and A. Yasinsac, "On the credibility of MANET simulations", Computer, vol. 39, no.7, pp. 48-54, 2006.

[10] E. Anderson, J. Brooks, C. Grassl, and S. Scott, "Performance of the Cray T3E multiprocessor," Proceedings of the ACM/IEEE Supercomputing Conference, pp.39-55, San Jose, CA USA, 1997.

[11] K. V. Anjan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," Proceedings of the 22nd annual international symposium on computer architecture, pp.201-210, Santa Margherita Ligure, Italy, 1995.

[12] K. V. Anjan, T. M. Pinkston, and J. Duato, "Generalized theory for deadlock-free adaptive wormhole routing and its application to DISHA concurrent," Proceedings of the 10th International Parallel Processing Symposium, pp.815-821, Honolulu, HI, USA, 1996.

[13] Top 500 supercomputers, http://www.top500.org, Accessed at: 05/03/2007.

[14] K. Aoyama and A. A. Chien, "The cost of adaptivity and virtual lanes in a wormhole router", VLSI Design, vol. 2, no.4, pp. 315-333, 1995.

[15] M. Banikazemi, V. Moorthy, L. Herger, D. K. Panda, and B. Abali, "Efficient virtual interface architecture (VIA) support for the IBM SP switch-connected NT clusters," Proceedings of the International Parallel and Distributed Processing Symposium, pp.33-42, Cancun, Mexico, 2000.

[16] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, et al., "Myrinet. A gigabit-per-second local area network", IEEE Micro, vol. 15, no.1, pp. 29-36, 1995.

[17] G. Bolch, S. Greiner, H. D. Meer, and K. S. Trivedi, Queueing networks and Markov chains: Modeling and performance evaluation with computer science applications. New York, NY: Wiley-Interscience, 1998.

[18] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip", Integration, the VLSI Journal, vol. 38, no.1, pp. 19-42, 2004.

[19] W. Bolton, Laplace and z-transforms: Longman Publishing Group 1997.

[20] R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms", IEEE Transactions on Parallel and Distributed Systems, vol. 7, no.2, pp. 169-183, 1996.

[21] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, et al., "iWARP: An integrated solution to high-speed parallel computing," Proceedings of the Supercomputing, pp.330-339, Orlando, FL, USA, 1988.

[22] Y. Boura, C. R. Das, and T. M. Jacob, "A performance model for adaptive routing in hypercubes," Proceedings of the International Workshop on Parallel Processing, pp.11-16, 1994.

[23] R. Brightwell, "A comparison of three MPI implementations for Red Storm," Proceedings of the 12 European PVM/MPI Users' Group Meeting, Sorrento, Italy, pp. 425-432, 2005.

[24] G. Cabillic, T. Priol, and I. Puaut, "MYOAN: An implementation of the KOAN shared virtual memory on the Intel Paragon", IRISA, Research Report, vol. 812, 1994.

[25] C. M. Chiang and L. M. Ni, "Efficient software multicast in wormhole-routed unidirectional multistage networks," Proceedings of the IEEE Symposium on Parallel and Distributed Processing, pp.106-113, San Antonio, TX, USA, 1995.

[26] A. A. Chien, "Cost and speed model for k-ary n-cube wormhole routers", IEEE Transactions on Parallel and Distributed Systems, vol. 9, no.2, pp. 150-162, 1998.

[27] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors", Journal of the ACM, vol. 42, no.1, pp. 91-123, 1995.

[28] B. Ciciani, M. Colajanni, and C. Paolucci, "An accurate model for the performance analysis of deterministic wormhole routing," Proceedings of the 11th International Parallel Processing Symposium, pp.353-359, Geneva, Switzerland, 1997.

[29] B. Ciciani, M. Colajanni, and C. Paolucci, "Performance evaluation of deterministic wormhole routing in k-ary n-cubes", Parallel Computing, vol. 24, no.14, pp. 2053-2075, 1998.

[30] M. Colajanni, B. Ciciani, and S. Tucci, "Performance analysis of circuit-switching interconnection networks with deterministic and adaptive routing", Performance Evaluation, vol. 34, no.1, pp. 1-26, 1998.

[31] M. Colajanni, A. Dell'arte, and B. Ciciani, "Performance evaluation of message passing strategies and routing policies in multicomputers", Simulation Practice and Theory, vol. 6, no.4, pp. 369-385, 1998.

[32] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes", IEEE/ACM Transactions on Networking, vol. 5, no.6, pp. 835-846, 1997.

[33] D. Culler, J. P. Singh, and A. Gupta, Parallel computer architecture: A hardware/software approach. San Francisco, CA: Morgan Kaufmann, 1998.

[34] W. Dally, "Network and processor architecture for message-driven computers", VLSI and parallel computation table of contents, pp. 140-222, 1990.

[35] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks", IEEE Transactions on Computers, vol. 39, no.6, pp. 775-785, 1990.

[36] W. J. Dally, "Virtual channel flow control", IEEE Transactions on Parallel and Distributed Systems, vol. 3, no.2, pp. 194-205, 1992.

[37] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels", IEEE Transactions On Parallel And Distributed Systems, vol. 4, no.4, pp. 466-475, 1993.

[38] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "The reliable router: A reliable and high-performance communication substrate for parallel computers," Proceedings of the First International Workshop on Parallel Computer Routing and Communication, pp.241-255, Washington,Seattle, USA, 1994.

[39] W. J. Dally and C. L. Seitz, "Torus routing chip", Distributed Computing, vol. 1, no.4, pp. 187-196, 1986.

[40] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", IEEE Transactions on Computers, vol. 36, no.5, pp. 547-553, 1987.

[41] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," Proceedings of the Design Automation Conference, pp.684-689, Las Vegas, NV, USA, 2001.

[42] W. J. Dally and B. Towles, Principles and practices of interconnection networks Amsterdam; London, Elsevier/Morgan Kaufmann, 2004.

[43] L. Dickman, "Beyond hero numbers: Factors affecting interconnect performance," QLOGIC System Interconnect Group, White paper, 2005.

[44] S. Dobrev and I. Vrt'o, "Optimal broadcasting in even tori with dynamic faults", Lecture Notes in Computer Science, vol. 1900/2000, pp. 927-930, 2002.

[45] J. T. Draper and J. Ghosh, "A comprehensive analytical model for wormhole routing in multicomputer systems", Journal of Parallel and Distributed Computing, vol. 23, no.2, pp. 202-214, 1994.

[46] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no.12, pp. 1320-1331, 1993.

[47] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", IEEE Transactions on Parallel and Distributed Systems, vol. 6, no.10, pp. 1055-1067, 1995.

[48] J. Duato and P. Lopez, "Performance evaluation of adaptive routing algorithms for k-ary-n-cubes," Proceedings of the First International Workshop on Parallel Computer Routing and Communication, pp.45-59, Seattle, Washington, USA, 1994.

[49] J. Duato, S. Yalamanchili, and L. M. Ni, Interconnection networks: An engineering approach. San Francisco: Morgan Kaufmann Publishers Inc., 2003.

[50] B. Duzett and R. Buck, "An overview of the nCUBE 3 supercomputer," Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, pp.458-464, McLean, VA, USA, 1992.

[51] P. Fortier and H. Michel, Computer systems performance evaluation and prediction. Boston ; London: Digital Press, 2003.

[52] H. Fujii, Y. Yasuda, H. Akashi, Y. Inagami, M. Koga, O. Ishihara, et al., "Architecture and performance of the Hitachi SR2201 massively parallel processor system," Proceedings of the 11th International Parallel Processing Symposium, pp.233-241, Genva, 1997.

[53] M. Galles, "Spider: A high-speed network interconnect", IEEE Micro, vol. 17, no.1, pp. 34-39, 1997.

[54] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic", ACM SIGCOMM Computer Communication Review, vol. 24, no.4, pp. 269-280, 1994.

[55] J. Ge, "Wormhole routers for network-on-chip," Proceedings of the The IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, MA, USA, 2004.

[56] S. A. Ghozati and H. C. Wasserman, "K-ary n-cube network: Modeling, topological properties and routing strategies", Computers and Electrical Engineering, vol. 25, no.3, pp. 155-168, 1999.

[57] A. Grama, A. Gupta, G. Karypis, and V. Kumar, Introduction to parallel computing Harlow ; London; New York: Pearson/Addison-Wesley, 2003.

[58] M. Grammatikakis, D. Hsu, and M. Kraetzl, Parallel system interconnections and communications. Boca Raton ; London ; New York: CRC Press, 2001.

[59] M. D. Grammatikakis, D. F. Hsu, and J. F. Sibeyn, "Packet routing in fixed-connection networks: A survey", Journal of Parallel and Distributed Computing, vol. 54, no.2, pp. 77-132, 1998.

[60] R. Greenberg and L. Guan, "Modelling and comparison of wormhole-routed mesh and torus networks," Proceedings of the 9th International Conference on Parallel and Distributed Computing and Systems, Washington, D.C., USA, 1997.

[61] J. A. Gregorio, R. Beivide, and F. Vallejo, "Modeling of interconnection subsystems for massively parallel computers", Performance Evaluation, vol. 47, no.2/3, pp. 105-129, 2002.

[62] F. T. Hady and B. L. Menezes, "The performance of crossbar-based binary hypercubes", IEEE Transactions on Computers, vol. 44, no.10, pp. 1208-1215, 1995.

[63] V. Halwan, F. Ozguner, and A. Dogan, "Routing in wormhole-switched clustered networks with applications to fault tolerance", IEEE Transactions on Parallel and Distributed Systems, vol. 10, no.10, pp. 1001-1011, 1999.

[64] S. H. Hsiao, C. Y. R. Chen, K. Nwosu, and D. Meliksetian, "Performance analysis of finite-buffered multistage interconnection networks," Proceedings of the IEEE International Conference on Communications, pp.53-57, Geneva, 1993.

[65] P. C. Hu and L. Kleinrock, "A queueing model for wormhole routing with timeout," Proceedings of the International Conference on Computer Communications and Networks, pp.584-593, Las Vegas, NV, USA, 1995.

[66] P. Hu and L. Kleinrock, "An analytical model for wormhole routing with finite size input buffers," Proceedings of the 15th International Telegraphic Congress, pp.549-560, Washington D. C., 1997.

[67] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique", Computer networks, vol. 3, no.4, 1979.

[68] R. E. Kessler and J. L. Schwarzmeier, "Cray t3d: A new dimension for cray research," Proceedings of the IEEE Compcon Spring, pp.176-182, Francisco, CA, USA, 1993.

[69] A. Khonsari, A. Farahani, and M. Ould-Khaoua, "DISHA: A performance model of a true fully adaptive routing algorithm in k-ary n-cubes," Proceedings of the 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS), pp.183-190, Texas, USA, 2002.

[70] A. Khonsari and M. Ould-Khaoua, "Compressionless wormhole routing: An analysis for hypercube with virtual channels", Computers and Electrical Engineering, vol. 30, no.1, pp. 45-60, 2004.

[71] A. Khonsari, H. Sarbazi-Azad, and M. Ould-Khaoua, "Analysis of true fully adaptive routing with software-based deadlock recovery", Journal of Systems and Software, vol. 71, no.3, pp. 259-270, 2004.

[72] A. Khonsari, A. Shahrabi, M. Ould-Khaoua, and H. Sarbazi-Azad, "Performance comparison of deadlock recovery and deadlock avoidance routing algorithms in wormhole-switched networks", IEE Proceedings: Computers and Digital Techniques, vol. 150, no.2, pp. 97-106, 2003.

[73] J. Kim and C. R. Das, "Hypercube communication delay with wormhole routing", IEEE Transactions on Computers, vol. 43, no.7, pp. 806-814, 1994.

[74] J. H. Kim, L. Ziqiang, A. A. Chien, J. H. Kim, L. Ziqiang, and A. A. Chien, "Compressionless routing: A framework for adaptive and fault-tolerant routing", IEEE Transactions on Parallel and Distributed Systems, vol. 8, no.3, pp. 229-244, 1997.

[75] S. C. Kim and S. Lee, "Measurement and prediction of communication delays in Myrinet networks", Journal of Parallel and Distributed Computing, vol. 61, no.11, pp. 1692-1704, 2001.

[76] L. Kleinrock, Queueing systems: Applications, vol. 2. New York: John Wiley, 1975.

[77] L. Kleinrock, Queueing systems: Theory, vol. 1. New York: John Wiley, 1975.

[78] L. Kleinrock, "On the modeling and analysis of computer networks", Proceedings of the IEEE, vol. 81, no.8, pp. 1179-1191, 1993.

[79] H. Kobayashi, Modeling and analysis : An introduction to system performance evaluation methodology. Reading, Mass. ; London: Addison-Wesley, 1978.

[80] K. Kotapati and S. P. Dandamudi, "Buffer management in wormhole-routed torus multicomputer networks", Future Generation Computer Systems, vol. 16, no.5, pp. 483-491, 2000.

[81] D. Kouvatsos, S. Assi, and M. Ould-Khaoua, "Performance modelling of hypercubes with deterministic wormhole routing," Proceedings of the Performance Modelling and Evaluation of Heterogeneous Networks, pp.77/71-77/10, Ilkley, UK, 2003.

[82] D. Kouvatsos, S. Assi, and M. Ould-Khaoua, "Performance modelling of wormhole switching in two-dimensional torus with finite buffers," Proceedings of the Performance Modelling and Evaluation of Heterogenous Networks, pp.P31/31 - P31/11, Ilkley, UK, 2004.

[83] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET simulation studies: The incredibles", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 9, no.4, pp. 50-61, 2005.

[84] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, et al., "The Stanford FLASH multiprocessor," Proceedings of the 21st Annual International Symposium on Computer Architecture, pp.302-313, Chicago, IL, USA, 1994.

[85] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA highly scalable server," Proceedings of the 24th Annual International Symposium on Computer Architecture, ISCA, pp.241-251, Denver, Colorado, USA, 1997.

[86] K. Lee, S. J. Lee, and H. J. Yoo, "Low-power network-on-chip for high-performance SoC design", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no.2, pp. 148-160, 2006.

[87] S. J. Lee, K. Lee, S. J. Song, and H. J. Yoo, "Packet-switched on-chip interconnection network for system-on-chip applications", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 52, no.6, pp. 308-312, 2005.

[88] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, et al., "The network architecture of the connection machine CM-5", Journal of Parallel and Distributed Computing, vol. 33, no.2, pp. 145-158, 1996.

[89] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, et al., "The Stanford DASH multiprocessor", Computer, vol. 25, no.3, pp. 63-79, 1992.

[90] D. Lilja, Measuring computer performance: A practitioner's guide, Cambridge, Cambridge University Press, 2000.

[91] 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, pp. 2-12, 1991.

[92] K. J. Liszka, J. K. Antonio, and H. J. Siegel, "Is an alligator better than an armadillo? [Interconnection Networks]", IEEE Concurrency, vol. 5, no.4, pp. 18, 20-28, 1997.

[93] S. Loucif, M. Ould-Khaoua, and L. M. Mackenzie, "Analysis of fully adaptive wormhole routing in tori", Parallel Computing, vol. 25, no.12, pp. 1477-1487, 1999.

[94] Z. Lu and A. Jantsch, "Flit admission in on-chip wormhole-switched networks with virtual channels," Proceedings of the 2004 International Symposium on System-on-Chip, pp.21-24, Tampere, Finland, 2004.

[95] M. H. Macdougall, Simulating computer systems: Techniques and tools. London: The MIT Press, 1987.

[96] J. M. Martinez, P. Lopez, and J. Duato, "Impact of buffer size on the efficiency of deadlock detection," Proceedings of the Fifth International Symposium On High-Performance Computer Architecture, pp.315-318, Orlando, FL, USA, 1999.

[97] J. M. Martinez, P. Lopez, J. Duato, and T. M. Pinkston, "Software-based deadlock recovery technique for true fully adaptive routing in wormhole networks," Proceedings of the International Conference on Parallel Processing, pp.182-189, Bloomington, IL, USA, 1997.

[98] A. J. Mary and R. T. Michael, "An investigation of phase-distribution moment-matching algorithms for use in queueing models", Queueing Systems, vol. V8, no.1, pp. 129-147, 1991.

[99] G. Mas and P. Martin, "Network-on-chip: The intelligence is in the wire," Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp.174-177, San Jose, California, USA, 2004.

[100] P. K. Mckinley, Y.-J. Tsai, and D. F. Robinson, "Collective communication in wormhole-routed massively parallel computers", Computer, vol. 28, no.12, pp. 39-50, 1995.

[101] D. Miller and W. A. Najjar, "Preliminary evaluation of a hybrid deterministic/adaptive router," Proceedings of the Parallel Computing, Routing and Communication Workshop, pp.21–32, Atlanta, Georgia, USA, 1997.

[102] G. Min and M. Ould-Khaoua, "A performance model for wormhole-switched interconnection networks under self-similar traffic", IEEE Transactions on Computers, vol. 53, no.5, pp. 601-613, 2004.

[103] P. Mohapatra, "Wormhole routing techniques for directly connected multicomputer systems", ACM Computing Surveys, vol. 30, no.3, pp. 374-410, 1998.

[104] P. Mohapatra and C. R. Das, "Performance analysis of finite-buffered asynchronous multistage interconnection networks", IEEE Transactions on Parallel and Distributed Systems, vol. 7, no.1, pp. 18-25, 1996.

[105] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture", IEEE Micro, vol. 22, no.1, pp. 26-35, 2002.

[106] H. H. Najaf-Abadi and H. Sarbazi-Azad, "An accurate combinatorial model for performance prediction of deterministic wormhole routing in torus multicomputer systems," Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp.548-553, San Jose, California, USA, 2004.

[107] K. Nakazawa, H. Nakamura, T. Boku, I. Nakata, and Y. Yamashita, "CP-PACS: A massively parallel processor at the University of Tsukuba", Parallel Computing, vol. 25, no.13, pp. 1635-1661, 1999.

[108] R. Nelson, Probability, stochastic processes, and queueing theory: The mathematics of computer performance modeling. New York: Springer-Verlag, 1995.

[109] M. D. Noakes, D. A. Wallach, and W. J. Dally, "J-machine multicomputer. An architectural evaluation", Proceedings of the Annual Symposium on Computer Architecture, pp. 224, 1993.

[110] L. Oliker, R. Biswas, J. Borrill, A. Canning, J. Carter, M. J. Djomehri, et al., "A performance evaluation of the cray x1 for scientific applications," Proceedings of the Lecture Notes in Computer Science, pp.51-65, 2005.

[111] M. Ould-Khaoua, "A performance model for duato's fully adaptive routing algorithm in k-ary n-cubes", IEEE Transactions on Computers, vol. 48, no.12, pp. 1297-1304, 1999.

[112] M. Ould-Khaoua, L. M. Mackenzie, and R. Sutherland, "Alleviating channel bandwidth constraints in multicomputer networks", Journal of Systems Architecture, vol. 44, no.11, pp. 837-848, 1998.

[113] M. Ould-Khaoua and H. Sarbazi-Azad, "An analytical model of adaptive wormhole routing in hypercubes in the presence of hot spot traffic", IEEE Transactions on Parallel and Distributed Systems, vol. 12, no.3, pp. 283, 2001.

[114] M. Ould-Khaoua, H. Sarbazi-Azad, and M. S. Obaidat, "Performance modeling and evaluation of high-performance parallel and distributed systems", Performance Evaluation, vol. 60, no.1-4, pp. 1, 2005.

[115] J. F. Palmer, "The nCUBE family of high-performance parallel computer systems " Proceedings of the 3ed conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues pp.847-851, Pasadena, California, USA, 1998.

[116] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE Transactions on Computers, vol. 54, no.8, pp. 1025-1040, 2005.

[117] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. Mcauliffe, et al., "IBM research parallel processor prototype (RP3): Introduction and architecture," Proceedings of the International Conference on Parallel Processing, pp.764-771, PA, USA, 1985.

[118] G. F. Pfister and V. A. Norton, "hot spot" contention and combining in multistage interconnection networks. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994.

[119] M. Rezazad and H. Sarbazi-Azad, "The effect of virtual channel organization on the performance of interconnection networks," Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, USA, 2005.

[120] J. Riordan, An introduction to combanitorial analysis: John Wiley & Sons, 1958.

[121] F. Safaei, A. Khonsari, M. Fathy, N. Alzeidi, and M. Ould-Khaoua, "Performance modeling of fault-tolerant circuit-switched communication networks," Proceedings of the International Symposium on Parallel Computing in Electrical Engineering, pp.239-244, Bialystok, Poland, 2006.

[122] F. Safaei, M. Rezazad, A. Khonsari, M. Fathy, M. Ould-Khaoua, and N. Alzeidi, "Software-based fault-tolerant routing algorithm in multidimensional networks," Proceedings of the 20th International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, 2006.

[123] J. Sahuquillo, T. Nachiondo, J. C. Cano, J. A. Gil, and A. Pont, "Self-similarity in SPLASH-2 workloads on shared memory multiprocessors systems," Proceedings of the 8th Euromicro Workshop on Parallel and Distributed Processing, pp.293-300, Rhodes, Greece, 2000.

[124] H. Sarbazi-Azad, "A mathematical model of deterministic wormhole routing in hypercube multicomputers using virtual channels", Applied Mathematical Modelling, vol. 27, no.12, pp. 943-953, 2003.

[125] H. Sarbazi-Azad, A. Khonsari, and M. Ould-Khaoua, "Analysis of deterministic routing in k-ary n-cubes with virtual channels ", Journal of Interconnection Networks, vol. 3, no.1&2, pp. 67-83, 2002.

[126] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "An accurate analytical model of adaptive wormhole routing in k-ary n-cubes interconnection networks", Performance Evaluation, vol. 43, no.2-3, pp. 165-179, 2001.

[127] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "Analytical modeling of wormhole-routed k-ary n-cubes in the presence of hot-spot traffic", IEEE Transactions on Computers, vol. 50, no.7, pp. 623-634, 2001.

[128] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "A performance model of adaptive wormhole routing in k-ary n-cubes in the presence of digit-reversal traffic", Journal of Supercomputing, vol. 22, no.2, pp. 139-159, 2002.

[129] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "Analytical modelling of wormhole-routed k-ary n-cubes in the presence of matrix-transpose traffic", Journal of Parallel and Distributed Computing, vol. 63, no.4, pp. 396-409, 2003.

[130] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "Towards a more realistic comparative analysis of multicomputer networks", Concurrency and Computation: Practice & Experience, vol. 16, no.13, pp. 1271-1289, 2004.

[131] H. Sarbazi-Azad, M. Ould-Khaoua, L. M. Mackenzie, and S. G. Akl, "On some properties of k-ary n-cubes," Proceedings of the 8th International Conference on Parallel and Distributed Systems, pp.517-524, KyongJu City, Korea, 2001.

[132] S. L. Scott and J. R. Goodman, "Impact of pipelined channels on k-ary n-cube networks", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no.1, pp. 2-16, 1994.

[133] S. L. Scott and G. M. Thorson, "The Cray T3E network: Adaptive routing in a high performance 3D torus," Proceedings of the Symposium on High Performance Interconnects, pp.147-156, Stanford, CA, USA, 1996.

[134] A. Shahrabi and M. Ould-Khaoua, "On the performance of routing algorithms in wormhole-switched multicomputer networks," Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS, pp.515-519, Fukuoka, Japan, 2005.

[135] A. Shahrabi, M. Ould-Khaoua, and L. M. Mackenzie, "Analytical modelling of broadcast in adaptive wormhole-routed hypercubes", Microprocessors and Microsystems, vol. 25, no.8, pp. 389-398, 2001.

[136] A. Sharhabi, "Performance comparison of routing algorithms in wormhole-switched networks", Parallel Computing, vol. 32, pp. 870-885, 2006.

[137] D. Sima, T. Fountain, and P. Kacsuk, Advanced computer architectures: A design space approach: Addison Wesley, 1997.

[138] C. B. Stunkel, D. G. Shea, B. Abali, M. G. Atkins, C. A. Bender, D. G. Grice, et al., "SP2 high-performance switch", IBM Systems Journal, vol. 34, no.2, pp. 185-204, 1995.

[139] C. B. Stunkel, D. G. Shea, D. G. Grice, P. H. Hochschild, and M. Tsao, "The SP1 high-performance switch," Proceedings of the Scalable High-Performance Computing Conference, pp.150-157, Knoxville, TN, USA, 1994.

[140] C. S. Sunder, G. Baskar, V. Babu, and D. Strenski, "A detailed performance analysis of the interpolation supplemented Lattice Boltzmann method on the Cray T3E and Cray X1", International Journal of High Performance Computing Applications, vol. 20, no.4, pp. 557-570, 2006.

[141] H. Takagi, Queueing analysis - a foundation of performance evaluation, vol. 2. Amesterdam: North-Holand, 1993.

[142] T. Theocharides, G. M. Link, N. Vijaykrishnan, and M. J. Irwin, "Networks on chip (NoC): Interconnects of next generation systems on chip", Advances in Computers, vol. 63, pp. 35-92, 2005.

[143] C. D. Thompson, "Area-time complexity for VLSI," Proceedings of the eleventh annual ACM symposium on Theory of computing, pp.81-88, 1979.

[144] S. R. Thompson, A. R. Hainline, and L. L. Halcomb, "Vector performance estimation for Cray X-MP/Y-MP supercomputers, part 2", Journal of Supercomputing, vol. 7, no.4, pp. 437-467, 1993.

[145] H. C. Tijms, A first course in stochastic models. Chichester: John Wiley & Sons, 2003.

[146] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, "Impact of virtual channels and adaptive routing on application performance", IEEE Transactions on Parallel and Distributed Systems, vol. 12, no.2, pp. 223-237, 2001.

[147] J. S. Vetter, S. R. Alam, T. H. Dunigan, Jr., M. R. Fahey, P. C. Roth, and P. H. Worley, "Early evaluation of the Cray XT3," Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes Island, Greece, 2006.

[148] G. Zanghirati, F. Cocco, G. Paruolo, and F. Taddei, "A Cray T3E implementation of a parallel stochastic dynamic assets and liabilities management model", Parallel Computing, vol. 26, no.5, pp. 539-567, 2000.

[149] B. Zhou and M. Atiquzzaman, "Efficient analysis of multistage interconnection networks using finite output-buffered switching elements", Computer Networks and ISDN Systems, vol. 28, no.13, pp. 1809-1829, 1996.

Selected Publications

- N. Alzeidi, M. Ould-Khaoua, A. Khonsari. A New Modelling Approach of Wormhole-Switched Networks with Finite Buffers, to appear in International Journal of Parallel, Distributed, and Emergent Systems, 2008.

- N. Alzeidi, M. Ould-Khaoua and L.M. Mackenzie, Performance Modelling and Analysis of k-Ary n-Cubes with Virtual Channels and Finite Buffers. Submitted to IEEE Transactions on Parallel and Distributed Systems.

- N. Alzeidi, M. Ould-Khaoua, L.M. Mackenzie, A. Khonsari, Performance Analysis of Adaptively Wormhole-Routed Networks with Finite Buffers, proceedings of the IEEE International Conference on Communications (ICC 2007), June 2007, Glasgow, UK.

- N. Alzeidi, M. Ould-Khaoua, A. Khonsari, Performance Modelling of Torus Interconnection Networks with Finite Buffers, to appear in the International Journal of Computers and Applications.

- N. Alzeidi, M. Ould-Khaoua, L.M. Mackenzie, A. Khonsari. A New General Method to Compute Virtual Channels Occupancy Probabilities in Wormhole Networks, to appear in the Journal of Computer and System Sciences, doi:10.1016/j.jcss.2007.07.006

- N. Alzeidi, A. Khonsari, M. Ould-Khaoua and L.M. Mackenzie, A New Approach to Modelling virtual Channels in Interconnection Networks. To appear in Journal of Computer and System Sciences, doi:10.1016/j.jcss.2007.02.002

- N. Alzeidi, A. Khonsari, M. Ould-Khaoua, L.M. Mackenzie. A New General Model for Virtual Channel Multiplexing in Interconnection Networks. Accepted for publication in the International Conference on Network and Parallel Computing (NPC 2006), Japan, October 2006.

- N. Alzeidi, A. Khonsari, M. Ould-Khaoua, L.M. Mackenzie. On the Probability Distribution of Busy Virtual Channels. 5th International Workshop on Performance Modelling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2006) in conjunction with (IPDPS 2006). Greece, April 2006.

- N. Alzeidi, A. Khonsari, M. Ould-Khaoua, L.M. Mackenzie. A New Approach to Model Virtual Channels in Interconnection Networks. 11th International CSI Computer Conference (CSICC06), Iran, January 2006. (Best student paper award)

- N. Alzeidi, M. Ould-Khaoua, A. Khonsari. Worm Latency in Hypercube with Finite Buffers. 20th Annual UK Performance Engineering Workshop (UKPEW' 2004), Bradford, UK, July 2004.