Performance Evaluation of Distributed Crossbar Switch Hypermesh

Samia Loucif

Dissertation Submitted for the Degree of Doctor of Philosophy
to the Faculty of Science, Glasgow University.

Abstract

The interconnection network is one of the most crucial components in any multicomputer as it greatly influences the overall system performance. Several recent studies have suggested that *hypergraph* networks, such as the *Distributed Crossbar Switch Hypermesh* (DCSH), exhibit superior topological and performance characteristics over many traditional *graph* networks, e.g. *k*-ary *n*-cubes.

Previous work on the DCSH has focused on issues related to implementation and performance comparisons with existing networks. These comparisons have so far been confined to deterministic routing and unicast (one-to-one) communication. Using analytical models validated through simulation experiments, this thesis extends that analysis to include adaptive routing and broadcast communication. The study concentrates on wormhole switching, which has been widely adopted in practical multicomputers, thanks to its low buffering requirement and the reduced dependence of latency on distance under low traffic.

Adaptive routing has recently been proposed as a means of improving network performance, but while the comparative evaluation of adaptive and deterministic routing has been widely reported in the literature, the focus has been on graph networks. The first part of this thesis deals with adaptive routing, developing an analytical model to measure latency in the DCSH, and which is used throughout the rest of the work for performance comparisons. Also, an investigation of different routing algorithms in this network is

presented.

Conventional k-ary n-cubes have been the underlying topology of contemporary multicomputers, but it is only recently that adaptive routing has been incorporated into such systems. The thesis studies the relative performance merits of the DCSH and k-ary n-cubes under adaptive routing strategy. The analysis takes into consideration real-world factors, such as router complexity and bandwidth constraints imposed by implementation technology.

However, in any network, the routing of unicast messages is not the only factor in traffic control. In many situations (for example, parallel iterative algorithms, memory update and invalidation procedures in shared memory systems, global notification of network errors), there is a significant requirement for broadcast traffic. The DCSH, by virtue of its use of hypergraph links, can implement broadcast operations particularly efficiently. The second part of the thesis examines how the DCSH and k-ary n-cube performance is affected by the presence of a broadcast traffic component.

In general, these studies demonstrate that because of their relatively high diameter, k-ary n-cubes perform poorly when message lengths are short. This is consistent with earlier more simplistic analyses which led to the proposal for the *express-cube*, an enhancement of the basic k-ary n-cube structure, which provides additional *express channels*, allowing messages to bypass groups of nodes along their paths. The final part of the thesis investigates whether this "partial bypassing" can compete with the "total bypassing" capability provided inherently by the DCSH topology.

Dedication

There is no doubt, in my mind, that this work would not have Been completed without the help of Allah (SWT) and memory of my father. My only regret is that he is not with us to see that one of his greatest wish has come true.

This thesis is dedicated to my father whose encouragement, during his life time and continuous reminder, after his death enabled me to pull through the 'low' times during the course of this research.

Acknowledgement

I would like to express my grateful thanks to Dr. Lewis Mackenzie for his great help throughout the course of this work. My great thanks are also due to Dr. Mohamed Ould-Khaoua. They both provided me with continuous guidance, encouragement, and valuable feedback since the early stages of this thesis. I would also like to thank Professor D.C. Gilles and Dr. J. Jeacocke for providing help and advice any time I asked.

I am particularly grateful to my mother, sisters, sister-in-law, brothers, brothers-in-law, and especially adorable nephews, for their unconditional love and moral support. I was fortunate enough to make the acquaintances of Mrs. Nasreen Ahmad and her family, who have become and will remain my second family in Glasgow. I thank them for their support.

Finally, I am indebted to the Algerian Government and British Council for the financial support and the opportunity they gave me to study for this postgraduate degree.

Contents

List of Figures

List of Symbols

B_h^C	mean message blocking delay at a physical channel, h hops away from destination
B_{DCSH}	bisection width in the DCSH
B_{Torus}	bisection width in the torus
$B_{Express\text{-}Cube}$	bisection width in the express-cube
B_i^m	mean blocking delay at the input multiplexer of dimension i
Ch_j^B	j^{th} local channel at the first stage within a dimension in the express-cube
Ch_j^E	j^{th} local channel at the last stage within a dimension in the express-cube
C_j^n	number of j combinations among n distinguishable items
d_{avg}	mean message distance in the network
D_t	router delay
Exp_j	j^{th} express channel within a dimension
fe	mean message rate traveling on express channels within a dimension
fl	mean message rate continuing on local channels within a dimension
i, j, h	indices
k	dimension width (or cluster size)
k_{avg}	mean message distance within a dimension in the torus and express-cube

k_n	group size in the express-cube
k_{avg}^{B}	mean number of local channels crossed by a message before reaching the first interchange
k_{avg}^{E}	mean number of express channels crossed by a message within a dimension
k_{avg}^{F}	mean number of local channels crossed by a message after the last interchange
L_b	mean length of broadcast messages
L_m	mean length of unicast messages
L_i	mean latency of an i-hops message, excluding waiting time at the source and multiplexing effects
L	mean network latency, excluding waiting time at the source and multiplexing effects
$Latency$	mean message latency
l_1, l_2	indices
M_{avg}	mean multiplexing degree at input multiplexers
M_{avg}^{i}	mean multiplexing degree at input multiplexer of dimension i
M	message length in bits
m_b	mean rate of broadcast messages generated by a PE
m_g	mean rate of unicast messages generated by a PE
m_c	mean message rate on a physical channel
m'	fraction of the generated traffic taking a given path in the network
m	number of express channels connecting each pair of interchanges
$m^{L/F/E}$	total message rate on a local or express channel
N	network size in nodes
N_j	number of nodes distant j hops away from any node in the DCSH
n	network dimensionality

P_h^C	probability of message blocking at a router h hops far from destination
P^C	probability that a required virtual channel is occupied
P_j^C	probability that j virtual channels are busy at a physical channel
P_j^m	probability that j virtual channels at the input multiplexer are busy
$P_{i,j}^C$	probability that j virtual channels are busy at the i^{th} physical channel
P_{c_h}	probability that there remains one dimension for the message to cross
P_a	probability that all adaptive virtual channels at a dimension are busy
P_d	probability that adaptive and deterministic virtual channels at a dimension are busy
P_{DCSH}	node pin-out in the DCSH
P_{Torus}	node pin-out in the torus
$P_{Express-Cube}$	node pin-out in the express-cube
$P_{i,j,v}^{B/F/E}$	Probability that v virtual channels of the j^{th} physical (local or express) channel in dimension i are busy
p_j	probability that a generated message crosses j hops
pcl	probability that a message arriving at an interchanges crosses a local channel in its next hop
p_s	probability that a message skips a dimension
q_j^C	variable used in the calculation of probability of virtual channels occupancy
q_j^m	intermediate variable used in the calculation of P_j^m
$q_{i,j,v}^{B/F/E}$	intermediate variable used in the calculation of $P_{i,j,v}^{B/F/E}$
R_i^L	mean message rate entering the network through dimension i
R_i^P	mean message rate entering dimension i from lowest dimensions
Rc	total message rate circulating within a dimension in the express-cube

S_i	latency seen by a message entering the network through dimension i
$T_p(j)$	factor reflecting traffic distribution
T_i	latency seen by a message exiting dimension i
T_i^c	mean message blocking delay at dimension i in deterministic routing
T_i^b	mean service time at i^{th} physical channel
$T_{e,\alpha}$	mean service time seen by messages of the flow α when entering a channel
$T_{q,\alpha}$	mean service time seen by messages of the flow α after exiting a channel
$T_{q,\beta}$	mean service time seen by message rate β exiting a channel
$T_{i,j}^B$	mean service time seen by a message entering j^{th} local channel of the first stage within dimension i
$T_{i,j}^F$	mean service time seen by a message entering j^{th} local channel of the last stage within dimension i
$T_{i,j}^E$	mean service time seen by a message entering j^{th} express channel within dimension i
V	number of virtual channels per physical channel
V_{avg}	mean multiplexing degree of virtual channels
V_{avg}^i	mean multiplexing degree of virtual channels at dimension i
$V_{i,j}^{BIFIE}$	mean multiplexing degree of virtual channels at the j^{th} physical (local or express) channel of dimension i
W_{qs}	mean waiting time of a message at the source
W_{qd}	mean waiting time of a message at the destination
w	channel width in bits
w_{DCSH}	channel width in the DCSH
w_{Torus}	channel width in the torus

$w_{Express\text{-}Cube}$	channel width in the express-cube
Φ	total message rate taking the same path in the network and entering dimension i
α, β	variables used in the calculation of $T_{e,\alpha}$

Chapter 1

Introduction

Much recent research and development has been directed at increasing processing power by incorporating concurrent features such as pipelining and multiple function units into uni-processor design [65]. However, there is still a limit to the performance gain achievable in this way. Scientific and engineering applications, such as finite element analysis, partial differential equations solving, and linear algebra require more powerful machines to reduce their execution time [26, 27].

Parallel processing is the ultimate means for increasing performance beyond that achievable from a single processor. In such systems, parallel tasks inherent in an application are distributed over a set of processors to run simultaneously. Those processors are physically interconnected by an *interconnection network*, and co-ordinate their activities to solve a common problem by exchanging information. Depending on the way the communication is achieved between processors, two types of parallel systems

can be distinguished: *multiprocessors*, where processors share a common memory through which they communicate [58, 81, 121, 129], and *multicomputers*, where each processor has its own local memory, communicating with the other processors by passing messages [104, 111, 118]. In order to allow processors to concentrate on computational tasks and permit the overlapping of communication with computation, a *router*, responsible for handling message communication among processors, is associated with each processing element (PE). The assembly of processor and router is called a *node*. Because of their scalability, multicomputers have gained popularity over multiprocessors [7, 91, 130].

The performance of parallel architectures can be affected by considerations of different system levels, from the effectiveness of load balancing [69, 73] (the way tasks are distributed over processors; an uneven distribution can lead to poor performance) to the efficiency of the underlying network. The latter has been an active research area in recent years since any interaction between processors is highly dependent on the efficiency of the underlying network.

Network performance is usually measured in terms of *latency* and *throughput*. The former metric measures the speed of the network, and it is defined as the time taken by a message to cross the network. The latter is the number of messages delivered by the network per unit of time, and represents the load handling capacity of the network. Ideally, a network should provide low latency and high throughput. Nevertheless, there are many parameters which can affect network design as will be discussed in the next section.

1.1 Interconnection Networks and their Features

Network performance can be affected by several factors of which the most important are *topology*, *switching method*, *routing algorithm*, *traffic distribution*, and *implementation technology*.

1.1.1 Topology

Topology describes the way system nodes are connected. *Diameter*, *degree*, and *regularity* are often used to characterise this. The diameter is the maximum shortest path between any pair of nodes. The node degree is the number of channels connecting the node to its neighbours, and the degree of a topology is the maximum degree of any node. Finally, a topology is said to be regular if all its nodes have the same degree.

Common multicomputer networks, such as the hypercube and k-ary n-cubes (including the torus and mesh) [134], are *graph* topologies[1]. A graph is of the form $G(E,V)$ and is defined over a set of vertices V and a set of edges E. Each vertex typically represents a node, and each edge between two vertices represents a physical channel connecting nodes. A fundamental constraint of the graph model is that each edge joins *exactly* two vertices. However, if a network channel can connect any number of nodes, a new class of topologies emerges. Using a graph-theoretical framework, members of this class can be modelled as *hypergraphs* [11], which are generalisations of conventional graphs in which individual "hyper-edges" are able to join an arbitrary number of vertices.

[1] There is also another class of networks, namely multi-stage networks, that cannot be classified as graphs or hypergraphs because PEs are indirectly connected by multiple stages of banks of routers. These networks have originally been proposed for early shared-memory multiprocessors [30, 58, 90], and have not been gained wide use in multicomputers, and therefore they are excluded from further discussion in this thesis.

One interesting class of *regular multi-dimensional* hypergraphs whose basic characteristic layout is the *hypermesh* [135, 137], is shown in Figure 1.1. The $N(=k^n)$ nodes of the system are organised as a collection of orthogonal groups, called *clusters*, of k nodes directly connected, and arranged in an n-dimensional space. These networks have desirable topological properties. Firstly, compared to graph networks, their diameter grows more slowly as the system size scales up. Secondly, they can support applications that map naturally onto the hypercube, torus, and binary tree due to their richer connectivity [78, 135]. Finally, they can realise all SIMD (Single Instruction Single Data) permutations of some multi-stage networks including the Omega and inverse Omega networks [136-138].

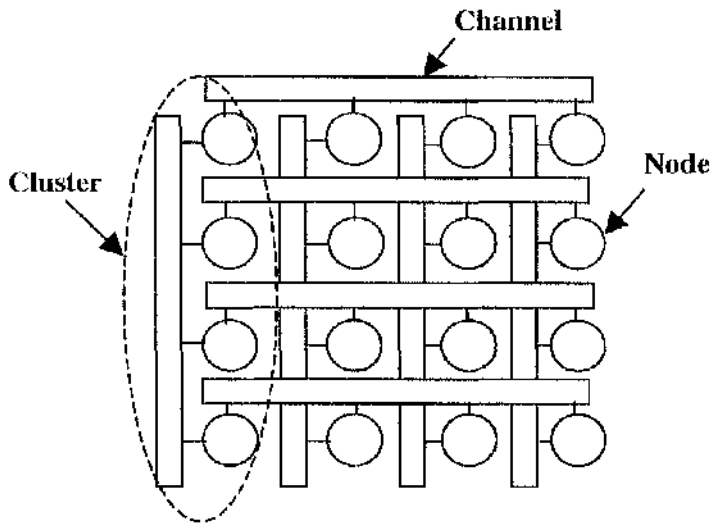


Figure 1.1: A 2-dimensional hypermesh (k=4).

1.1.2 Switching Method

The switching method used in a network determines the way messages are handled as they travel through intermediate routers. Messages, in general, are divided into packets to reduce buffer storage and transmission delay [77]. A packet, in turn, can be divided into *flits*, the smallest data unit on which flow control can be performed. The first flit, called the *header*, contains routing information.

In circuit switching [139], the path between the source and destination must be set up before the initiation of message transmission. This eliminates the contention delay for messages when traversing the network. However, its primary disadvantage is the significant overhead to establish the path under heavy traffic, notably with short messages, which has limited its acceptance for use in multicomputers.

Store-and-Forward switching (S&F) [101] was popular in the first generation of multicomputers [68, 127]. In this method, a packet must be completely received by the router before it is forwarded to the next node. This has two shortcomings: the large amount of memory needed to buffer the whole packet and the waste of channel bandwidth due to unnecessary buffering. *Virtual-cut-through* switching (VCT) [72] has been introduced as an enhancement to S&F under low and moderate traffic loads. The header flit determines the route and the data flits follow on through the channels in a pipelined fashion. If the header is blocked because the required channel is busy, the data flits are temporarily buffered. Therefore, this technique behaves as S&F under high traffic loads, requiring a large amount of memory.

This drawback has encouraged the use of a blocking variant of VCT called *wormhole*

switching [109] in the latest generation of multicomputers [6, 110, 120]. The main difference between VCT and wormhole switching lies in the way messages are handled when they are blocked. In wormhole switching, if the header is blocked, the data flits are also blocked in situ. In addition to the fact that latency becomes insensitive to the message distance under light traffic loads in both VCT and wormhole switching, wormhole routers are fast and simple to design [109] due to their low buffering requirement.

One disadvantage of wormhole routing, however, is that when messages are blocked, they are spread-out along the path, holding channels and buffers. Dally [34] has proposed the *virtual channel* concept to reduce the effects of the chained channel blocking. Instead of having a unique buffer queue for each physical channel, the buffer queue is split into several parallel queues, called virtual channels, which are time multiplexed and share the bandwidth of the same physical channel. The virtual channel concept de-couples buffer allocation from physical channel allocation, allowing non-blocked messages to bypass blocked ones. Figure 1.2 illustrates how messages progress simultaneously across a physical channel. Each physical channel is split into two virtual channels. At node $N1$, messages $M1$ and $M2$ share the bandwidth of the physical channel $C1$ on a flit by flit basis. If message $M1$ gets blocked at node $N2$, message $M2$ still can progress. At node $N2$, message $M2$ uses the whole bandwidth of the physical channel $C2$ because the second virtual channel buffer is empty.

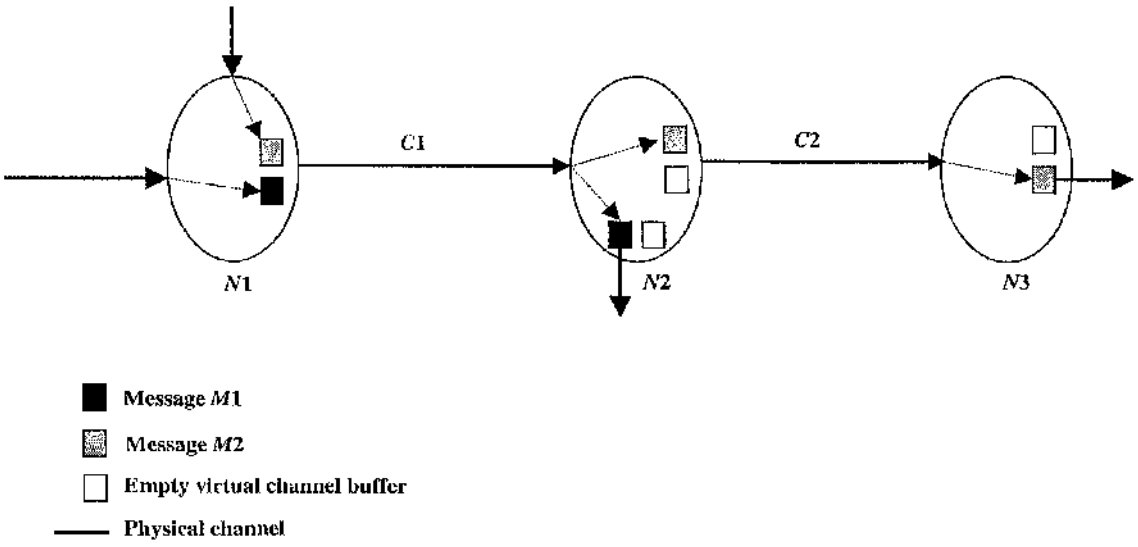


Figure 1.2: Example of messages sharing the physical channel bandwidth.

1.1.3 Routing Algorithms

A routing algorithm establishes the path for messages to cross the network. It is broadly classified as *deterministic* or *adaptive*. The former offers a predetermined path to each message, fixed by its source and destination addresses. The latter gives flexibility to messages to choose their paths to avoid congested regions.

A critical requirement for any routing algorithm is to ensure *deadlock freedom*; deadlock occurs when messages cannot advance in the network because of *cyclic* waiting for resources (i.e., buffers and channels). Figure 1.3 shows an example of a deadlock, where each of the four messages ($M1$, $M2$, $M3$ and $M4$) holds a flit buffer and requests the flit buffer of the next node in a circular fashion. A simple idea to avoid deadlock is to

prohibit messages to use some turns which may cause cycles in the network. This often results in providing messages a unique path, predetermined by the source and destination addresses. A typical way of implementing this is seen in the classic deterministic routing, widely known as *dimension-ordered* routing algorithm (also known as *e-cube* [32, 128]) used in the hypercube. In this form of routing, messages are restricted to change dimensions in a predefined (increasing or decreasing) order to ensure deadlock avoidance.

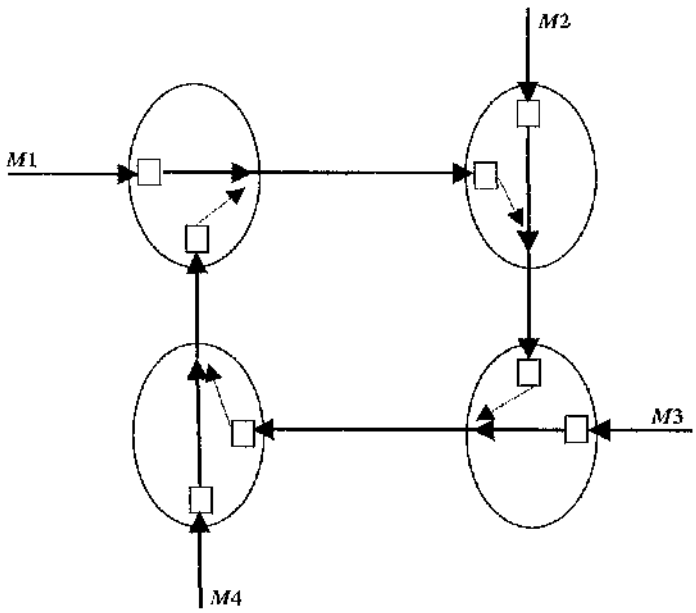


Figure 1.3: Example of a deadlock situation caused by four messages.

In some networks, like k-ary n-cubes, forcing messages to cross dimensions in a strict order is not sufficient to avoid deadlock. Seitz & Dally [32] have suggested the use of virtual channels as a means of breaking cycles caused by wrap-around connections within dimensions, and transforming these cycles into "spirals". Their idea is to assign each virtual channel a unique number to force messages to visit virtual channels in a strict order. Deterministic routing has been widely used in existing multicomputers because it offers a simple way to ensure deadlock-freedom, resulting in simple and fast routers [24, 103]. However, since messages always take the same route between a pair of nodes, this form of routing cannot respond to the onset of congestion. Adaptive routing has recently been proposed to overcome the performance limitation of deterministic routing [10, 18, 47, 70, 89].

Several authors have used the concept of virtual channels for designing adaptive routing algorithms, requiring different numbers of virtual channels. Linder & Harden's algorithm [89] is one typical example, which uses a large number of virtual channels. For n-dimensional k-ary n-cubes, this needs up to $(n+1)2^{n-1}$ virtual channels per physical channel to avoid deadlock. Recent studies have revealed that this large number of virtual channels often translates in practice into additional control logic in the router circuitry, increasing router complexity and delays [5, 24, 108]. The high cost of adaptivity has motivated researchers to develop adaptive routing algorithms that require a moderate number of virtual channels.

Duato [48] has recently introduced a simple and efficient method for designing adaptive routing algorithms. His method consists of splitting the network into a set of virtual networks. Cycles are allowed in all the virtual networks but one. The virtual network, that

does not contain cycles, provides "escape" routes for messages to break deadlock, which may occur in the other networks. His method has gained popularity because it is general; it can be applied to any topology. Furthermore, it requires only one extra virtual channel to ensure deadlock-freedom compared to the deterministic routing algorithm. Duato's approach is extensively used in this work through an adaptive routing algorithm, which will be described more fully in the next chapter.

1.1.4 Traffic Distribution

The communication pattern, the way traffic generated by the system nodes is distributed across the network, strongly affects the performance of the latter. Two types of messages may be identified crossing the network, namely *unicast* and *broadcast*[2].

A unicast message is sent to exactly one target destination node. Uniform traffic distribution has been used in many research studies [20, 33, 34, 76]. In this traffic pattern, unicast messages generated by each node are sent to the other network nodes with equal probability. In addition to the fact that uniform traffic makes network analysis tractable, there are actually some important real situations where this type of traffic is generated. For instance, with the implementation of the shared-memory model on multicomputers [29, 74], practices such as memory interleaving tend to spread access uniformly over all nodes [2]. Moreover, iterative methods found in solving many scientific and engineering applications [94, 125] often use the spare matrix dense vector multiply algorithm. The latter creates almost uniform traffic distribution [63]. Other forms of unicast traffic can be non-uniform where each node communicates with one or a set of specific nodes. For

[2] Unicast and broadcast communications can be considered as special cases of *multicast* communication [107], where a message is sent to an arbitrary number of destinations.

example, permutation operations, such as matrix transpose, bit reversal and perfect shuffle, frequently used in numerical computations [38], generate such a traffic pattern.

A broadcast message is sent to all system nodes. Broadcast messages increase the traffic in the network and consume more network bandwidth than unicast messages (there is an intensive research on this topic to efficiently implement such a communication scheme [86, 107, 116]). Many scientific applications exhibit the need for broadcast communication. For instance, a variety of linear algebra algorithms such as matrix-vector computations [43], and parallel iterative algorithms [51, 99] rely on broadcast communication. Broadcast communication is also useful for global synchronisation [143] and cache invalidation and updating in distributed-shared memory systems [31].

1.1.5 Implementation Constraints

In any real comparison of networks it is important to take account of the implementation technology because this will, in general, impose limitations on the channel bandwidth which varies across different topologies.

Early studies of network performance ignored these limitations and assumed constant channel bandwidth regardless of the topology. The results suggested that the hypercube is the best topology due to its high connectivity, despite its high wiring complexity and high node degree. Consequently, the hypercube architecture was widely used in the first generation of multicomputers [104, 111, 127]. However, as researchers included implementation constraints in their analyses [1, 2, 33, 66, 115], these views have changed.

Investigating VLSI implementation of multicomputers, Dally [33] has introduced wiring density as a cost measure in his analysis of wormhole-routed k-ary n-cube performance. In a VLSI implementation, the whole network is assumed to be laid-out on a single chip. The wiring complexity of the system is a major problem, since the silicon area is limited and in general interconnection networks are wiring intensive [8, 54, 124]. *Bisection width* [141] has often been the metric used to approximate the wiring density of networks [2, 4, 33, 60]. It is defined as the number of wires cut, when the network is divided into equal halves. Under such constraint, Dally has shown that k-ary n-cubes exhibit superior performance over the hypercube due to their wider channels.

The wiring density argument is certainly applicable where an entire network can be implemented on a single VLSI chip. However, in some situations where the system cannot fit on a single chip and it has to be partitioned across several chips, here the node *pin-out* is a better measure of network complexity [1]. Assuming that each node is implemented on a single chip, the node pin-out is the degree times the channel width (in wires). The assumption used in the implementation of networks, when analysing their performance, can have a great impact on the results. For instance, when Abraham & Padmanabhan [1] have investigated the performance of the hypercube and k-ary n-cubes under multiple-chip technology, contrarily to Dally's conclusion, they have found that the hypercube provides superior performance than k-ary n-cubes.

Two other important implementation issues that have been considered when comparing network performance are *router delay* and *wire length* [2, 4, 33, 126]. Router delay, the time taken by a switch to decide which output a message should take, and also to establish the connection to this output, depends, in general, on the complexity of the

router circuitry. This, in turn, is mainly affected by the routing algorithm being used [24]. Router delay is significant in current technologies [2, 37], and it can have a detrimental effect on the performance of high-diameter networks. With the advent of adaptive routing, the effect of router delay on system performance is expected to be more severe due to the additional logic complexity to implement adaptivity and to prevent deadlock [5, 24].

Several authors [2, 33] have shown that long wires can have considerable degrading effect on network performance. Scott & Goodman [126] have recently proposed the use of pipelined channels as a way to reduce this effect by allowing multiple bits to be in flight on the wire simultaneously. The latest Cray T3E [29] is an example of a practical machine which employs pipelined channels.

1.2 Distributed Crossbar Switch Hypermesh

A simple way to implement the "hyper-edge" in the hypermesh is to use a common bus, shared by a cluster of nodes, such as the case of the *spanning bus hypercube* [41, 53, 142]. The problem with shared-buses in high-speed applications is the difficulty of arranging frequent changes of mastership without incurring significant performance overhead [62]. Other implementations of the hypermesh are based on either crossbar switches [14, 55, 56, 140, 144] or complete connection [13]. The latter is well known as the *generalised hypercube* where nodes in a given dimension are connected by a complete connection. Although a channel in the complete connection network interconnects exactly two nodes, the set of channels emanating from all nodes within a dimension is seen as the implementation of a hyper-edge. All these implementations,

however, suffer from bandwidth limitation as the system size increases [112].

An interesting implementation of the hypermesh, called the *Distributed Crossbar Switch Hypermesh* (DCSH) [95, 112], has been proposed as a part of the COBRA project at Glasgow university. The 1-dimensional DCSH, referred to as a *cluster*, depicted in Figure 1.4, forms the basic topology upon which higher-dimensional DCSH structures are formed. Every node possesses a uniquely owned unidirectional channel that connects it to the other k-1 nodes in the cluster. A distributed crossbar switch connects the k nodes along each dimension; the multiplexing/demultiplexing functions of the conventional "centralised" crossbar switch are performed in the router. A $(k-1)-\text{to}-1$ multiplexer with buffered inputs is provided at the input of the router for each cluster to arbitrate between the k-1 senders.

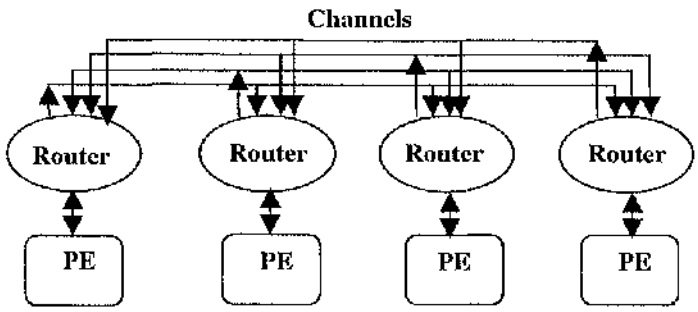


Figure 1.4: Structure of a cluster in the DCSH ($k = 4$).

A n-dimensional DCSH is a regular hypergraph with $N = k^n$ nodes, formed by taking the *Cartesian* n-product of the basic cluster topology. Let dimensions in an n-dimensional DCSH be numbered 0, 1,..., $n-1$. A node, a, can then be labeled by an $n \times 1$ address vector with a_i being the node position in its dimension (cluster) i. Each node is connected to $n(k-1)$ other nodes with which it differs in only one address digit, i.e., $a = a_0 ... a_{i-1} a_i a_{i+1} ... a_{n-1}$ is connected to $a' = a_0 ... a_{i-1} a_i' a_{i+1} ... a_{n-1}$ for all $0 \le i < n$, $0 \le a_i, a_i' < k$ and $a_i \neq a_i'$. When the size of the cluster is reduced to $k = 2$, the DCSH collapses to the hypercube.

The problem of mastership changes usually required by shared buses is alleviated in the DCSH because each node of the cluster drives its own channel. Furthermore, simultaneous broadcast operations within a DCSH cluster are possible. Moreover, a study in [112] has revealed that the DCSH, by its nature, can benefit from an optimal partitioning of the network to escape the channel bandwidth limitation imposed by VLSI and multiple-chip implementation technologies, from which other networks suffer. Rather than flattening the network on a single VLSI chip or placing each node on a separate chip, the switching and processing functions of the nodes are separated using a layered implementation scheme [113]. This allows the DCSH to overcome the constraint imposed by wiring density while minimising the constraint imposed by the limited node pin-out. The two and three-dimensional DCSH are the most interesting cases that can exploit this layered implementation [112]. This is not only because they have very small diameter as the size of the system scales up, but also because they map naturally onto physical implementation technologies.

The comparative performance analysis of the two-dimensional DCSH and other

hypermesh implementations assuming equal implementation cost has been reported in [112]. The results have revealed that the DCSH provides superior performance characteristics. When compared to graph networks [114], namely k-ary n-cubes, the DCSH also outperforms those topologies under the same conditions. Although k-ary n-cubes possess wider channels than the DCSH, the smaller diameter of the DCSH enables it to provide better performance, especially when router delay is taken into consideration.

1.3 Motivations

Most commercial and experimental multicomputers are based on graph topologies, such as k-ary n-cubes. Recent studies have shown that the DCSH is a potential alternative to graph networks due to its attractive topological properties. In particular, the two-dimensional DCSH along with deterministic routing and unicast communication (one-to-one) has shown promising features when compared to common graph networks, e.g. k-ary n-cubes and hypercubes. The main objective of the thesis is to extend the work on the DCSH by investigating important issues concerning traffic management in the network, namely adaptive routing and broadcast communication. Moreover, it assesses the merits of the "total" bypassing inherent in the DCSH topology over "partial" bypassing strategy proposed for some graph networks like k-ary n-cubes. Wormhole switching is the technique used throughout the thesis. Moreover, the study is based on analytical models, which are developed whenever it is possible, alongside simulation.

Adaptive routing has recently been suggested as a way to overcome the performance limitations of deterministic routing. Numerous previous studies investigating the benefits of adaptive routing have focused on graph networks only [17, 44, 123] and there is no

comparable work on hypergraphs. To fill this gap, this thesis presents some results of an investigation of different routing algorithms in the DCSH.

The comparative performance analysis of the DCSH and k-ary n-cubes reported in [112, 114] has assumed the use of deterministic routing. More recently, adaptive routing has been incorporated in some multicomputers based on k-ary n-cube [29]. However, before adaptive routing can be widely adopted in practical systems, it is necessary to determine which of the competing topologies are able to fully exploit its performance benefits. To this end, this thesis re-assesses the relative performance merits of the DCSH and k-ary n-cubes in the context of adaptive routing.

Any general-purpose network must be able to handle broadcast communication as it is an important operation required by many parallel applications. Research on broadcast communication has always been focused on the design of efficient broadcast algorithms. Up to now, there has been little work which evaluates how well existing networks can support this type of communication. Among the desirable features of the DCSH is its ability to perform simultaneous broadcast operations within a cluster. The thesis assesses the performance merits of the DCSH and k-ary n-cubes in the presence of broadcast traffic.

Messages in k-ary n-cubes often visit a large number of intermediate nodes to cross the network. As a result, high router delays can cause considerable degradation in k-ary n-cubes performance, especially at short message lengths. To overcome this problem, Dally [37] has proposed an enhanced version of k-ary n-cubes, called *express-cubes*, where messages use "*express*" channels to *partially* bypass groups of nodes within a dimension. Examining the structure of the DCSH cluster, depicted in Figure 1.4, reveals that the

DCSH also shares the concept of "bypass channels" with express-cubes since it allows messages to "totally", rather than partially, bypass all the nodes in the cluster. The last part of the thesis examines whether this total bypassing strategy inherent in the DCSH topology yields better performance than the partial bypassing strategy provided by express-cubes.

1.4 Outline of the Thesis

Chapter 2 presents an analytical model to compute message latency in the DCSH with adaptive routing, based on Duato's algorithm [45]. The model with two other models developed in chapter 3 are used to evaluate the performance of deterministic and other adaptive routing strategies in the DCSH.

In chapter 4, the relative performance merits of the DCSH and k-ary n-cubes are investigated when adaptive routing is employed. The analysis takes into consideration router delay overheads and equal implementation costs in VLSI and multiple-chip technologies. The performance comparison of the two networks is extended in chapter 5 to include broadcast communication.

Chapter 6 analyses whether the total bypassing provided by the DCSH outperforms the partial bypassing provided by express-cubes. The study assumes equal implementation costs and takes into account router delays.

Finally, chapter 7 summarizes the results presented in the thesis, and discusses some possible directions for future research work.

Chapter 2

A Performance Model of Adaptive Routing in the DCSH

2.1 Introduction

Deterministic routing has been widely adopted in several practical systems [35, 74, 84, 118] because it offers a simple way to ensure deadlock-freedom. However, it does not allow messages to avoid congested parts of the network to reduce their communication latency. Adaptive routing has often been proposed to improve network performance by giving messages flexibility in choosing their paths taking into consideration the dynamic state of the network.

Performance evaluation studies can be achieved through either simulation or analytical models. The latter is often preferred over simulation whenever it is possible. This is because simulation requires laborious effort to develop the program code and costly

computing resources to run large systems. Nonetheless, given that analytical models resort in most cases to simplifying assumptions, simulation is often needed to validate the accuracy of the model. The validation is typically carried out for cases which require reasonable computing time and resources.

Models of deterministic routing have been widely reported in the literature [2, 33, 42, 59, 75, 76, 114]. Except from the work in [20], which has proposed a model of adaptive routing in the hypercube, there has been no study that has considered adaptive routing in other networks, including hypergraphs (e.g. DCSH) and other graph networks (e.g. k-ary n-cubes). This chapter proposes an analytical model of adaptive routing in the DCSH. The modelling approach adopted here will be also used to derive other models for different adaptive routing algorithms in the following chapters.

Before presenting the model derivation and its validation, this chapter gives first an overview of some adaptive routing algorithms proposed in the literature along with a brief description of adaptive routers. The application of Duato's adaptive routing algorithm in the DCSH is then discussed in more detail, as it will be used in the analysis.

2.2 Adaptive Routing Algorithms and Router Design

This section gives an overview of the different approaches suggested to design wormhole-routed deadlock-free adaptive routing algorithms, and presents a general description of adaptive routers.

2.2.1 Adaptive Routing Algorithms

Adaptive routing algorithms[3] use the concept of virtual channels [34] for deadlock avoidance. Recall that a virtual channel represents a "logical" channel with its own buffer and flow control. They are arranged so that a blocking in one does not affect another, sharing the bandwidth of the same physical channel. Studies on the complexity of routers [5, 24, 40, 49, 50] have shown that, in addition to adaptivity, a high number of virtual channels translates into high hardware complexity, which can significantly reduce router speed, decreasing the overall network performance.

Some researchers have proposed *partially* adaptive routing algorithms [22, 25, 57, 71] to reduce router complexity, by restricting the freedom given to messages in choosing their path. Although this approach improves performance compared to deterministic routing, messages cannot exploit all available paths to reduce their blocking delays [44]. Numerous routing algorithms have been proposed, enabling messages to exploit all the possible shortest paths provided by the network topology, resulting in *fully* adaptive routing algorithms [10, 18, 36, 70, 89].

The main difference between adaptive routing algorithms proposed in the literature is in the number of virtual channels required for deadlock avoidance and the way these channels are assigned to messages. For instance, Linder & Harden's algorithm [89] and the Jesshope *et al* algorithm [70] are representative cases of adaptive routing algorithms, which require a high number of virtual channels. The algorithms can use up to $(n+1)2^{n-1}$ virtual channels per physical channel. They are based on partitioning the network into a

[3] The thesis focuses on *minimal* routing algorithms only (messages are routed using the shortest paths in the network). Non-minimal routing algorithms [106] require extra hardware router circuitry to guarantee the delivery of messages to their destination, and increase message latency and traffic in the network.

number of virtual networks. A message is restricted to use a particular virtual network depending upon its source and destination addresses. In addition to the high number of virtual channels, the allocation of virtual channels to messages by these routing algorithms can lead to an unbalanced traffic in the network [92].

Boppana & Chalasani [18] have proposed another approach to design deadlock-free adaptive routing algorithms, based on the idea of the *structured buffer pool* method, traditionally used in S&F networks [61]. Each physical channel is split into D virtual channels, where D is the diameter of the network. To guarantee deadlock-freedom, messages cross virtual channels according to the number of hops they have made in the network. Upon reaching an intermediate node, a message uses the h^{th} virtual channel to complete its h^{th} hop. Again, the high number of virtual channels required in this routing algorithm makes it impractical in large diameter networks, e.g. k-ary n-cubes.

Duato [45, 48] has developed a method to design fully adaptive routing algorithms, which allow efficient router implementation, as they require a small number of virtual channels; only one extra virtual channel is required compared to deterministic routing. The idea here is that virtual channels divide the network into two classes of virtual networks: one adaptive and one deterministic. A message is routed adaptively without any restriction on the usage of virtual channels in the first network. If it is blocked, it switches to using virtual channels in the deterministic network. The deterministic network is deadlock-free, and therefore constitutes an "escape" virtual network to break deadlock that may occur in the adaptive network. Numerous adaptive routing algorithms have been proposed, based on Duato's method [10, 36, 47, 88, 92, 93, 133]. The success of that method in reducing the hardware cost in terms of virtual channels has led to the

implementation of adaptive routing in the MIT Reliable router [39], and also in the Cray T3E, which is the first commercial multicomputer that has adopted adaptive routing.

The adaptive routing algorithm used in the analysis, presented below, is based on Duato's method. The routing algorithm can be stated as follows. Each physical channel is divided into two classes of virtual channels: a and b. At each routing step, a message can adaptively visit at any dimension any available virtual channel from class a. If all virtual channels belonging to the class a are busy, it crosses a virtual channel from class b using deterministic routing. The virtual channels of class b define a complete virtual deadlock-free sub-network, which acts like a "drain" for the sub-network built from the class a virtual channels.

2.2.2 Router Design

In this section, a brief description of routers is presented (more details can be found in [5]). The main components of a router are *input buffers*, a *switch*, and a *routing control unit*. At each input buffer is an associated logic control unit which performs address decoding of the received message header, and flow control operations across the switch and between routers. The switch is usually implemented by a crossbar because it allows simultaneous paths between inputs and outputs, provided that there are no inputs requiring the same output. The routing control unit decides on the connection between router inputs and outputs.

A message header arriving at a router is buffered at its appropriate virtual channel buffer. Based on the routing algorithm, the logic control unit associated to that input decodes the header to generate requests for permissible outputs, and sends them to the routing control

unit. At the same time, it computes all possible new header addresses. The routing control unit, using the routing algorithm, combines output channel status information and the requests received from the router inputs, to decide on the assignment of inputs to outputs. It sends signals to the crossbar switch to establish the connection between the inputs and their corresponding selected outputs, and sends the decision made to each requesting input in order to select the appropriate new header.

Given that virtual channels share the bandwidth of the same physical channel, a virtual channel controller is required for each output physical channel to multiplex fairly between input virtual channels which are ready to transmit; inputs which have data to transmit and the buffer at the receiver node is available. Assuming a physical channel is split into V virtual channels, a 1-to-V demultiplexer is also needed at the other end of each physical channel to store the received flit in its appropriate buffer.

The routing control unit, logic control unit at router inputs, and virtual channel controller units affect the complexity of the router. As the degree of adaptivity and the number of virtual channels increase, so do the complexity and the delay at those units [5, 24].

2.3 Duato's Algorithm in the DCSH

Before discussing the implementation of Duato's algorithm in the DCSH, let us first recall the router structure and the way input multiplexers operate under deterministic routing, as suggested in [112]. Figure 2.1 shows the router structure for the two-dimensional DCSH; the discussion can be easily extended to higher dimensional cases. A router is connected to its neighbouring routers by external channels and it is connected to its local PE by internal channels, known as *injection/ejection* channels. Messages

generated by the local PE are injected into the network through the injection channel, while they are removed from the network and transferred to the local PE through the ejection channel. Messages arriving on a dimension from different nodes of the same cluster are buffered at the input of a (k-1)-to-1 multiplexer. Under deterministic routing, the arbitration of these multiplexers has been proposed to be at the *message level* [112]. Once the multiplexer selects one of the (k-1) inputs corresponding to the (k-1) senders in a given cluster, a connection is made through the router's crossbar to the output channel. This connection is maintained until the tail of the message leaves the router.

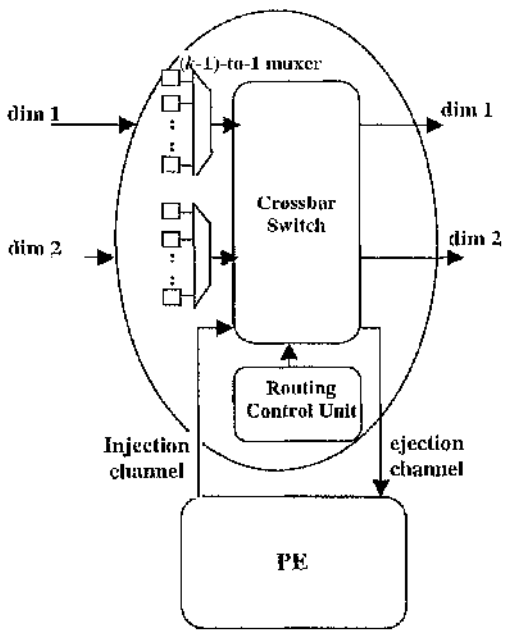


Figure 2.1: DCSH router structure with deterministic routing.

When each physical channel is split into V virtual channels, the size of input multiplexers scales by V. Moreover, adaptive and deterministic virtual channels of physical channels belonging to the same cluster are all connected to the corresponding input multiplexer. Duato's algorithm has been designed assuming non-blocking crossbars, and deterministic virtual channels in the algorithm represent escape routes for blocked messages holding adaptive channels. In the DCSH, however, due to the presence of input multiplexers and if these are servicing inputs at the message level, messages holding adaptive virtual channels can block those buffered at deterministic channels, leading to a deadlock problem. An example of a deadlocked configuration in two-dimensional DCSH ($k = 2$) is shown in Figure 2.2.

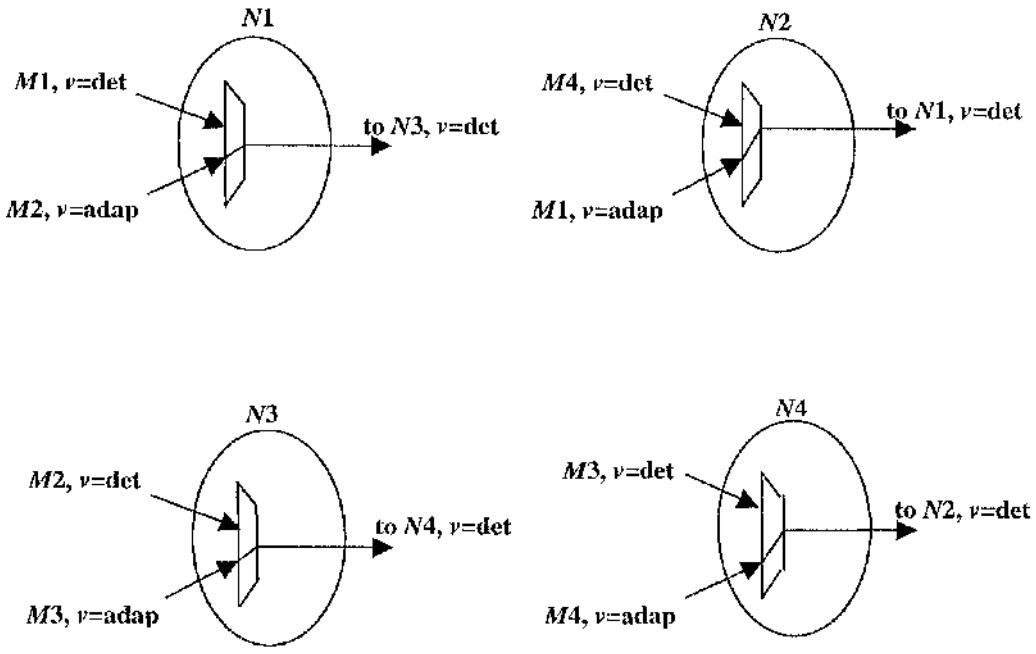


Figure 2.2: An example of a deadlocked configuration in the DCSH.

Since in the DCSH a message crosses at most one channel within a dimension, in what follows channels and dimensions are used interchangeably. As can be seen in Figure 2.2, message $M2$ arrives at node $N1$ on an adaptive virtual channel. It is then serviced by the input multiplexer. Because adaptive virtual channels at the dimensions still to be crossed are occupied, it acquires a deterministic channel in order to progress to node $N3$. $M2$ is blocked at node $N3$ because the input multiplexer at that node is servicing message $M3$. The latter is also holding a deterministic channel and is blocked at node $N4$ by message $M4$. $M4$, on its turn, is prevented to use any output channel at the node $N2$ because the input multiplexer is servicing message $M1$.

When a multiplexer selects an input, and the header of the message buffered at that input is blocked at a subsequent stage, it prevents messages of deterministic virtual channels from using the crossbar input. Although the output channel required by a message on a deterministic virtual channel becomes free, the crossbar at the input is unavailable, and therefore deterministic virtual channels no longer behave as escape channels, leading to possible deadlock. One solution to such a problem is to use "non-blocking" multiplexers at the input of the crossbar. Messages that can progress through their required output channels are never blocked, and they are time-multiplexed at the *flit level* over the crossbar inputs.

2.4 Analysis

This section describes the model proposed along with its validation through simulation. The number of virtual channels that provide the optimal performance of the DCSH is also examined. The router structure used in the analysis is shown in Figure 2.3. It is

similar to the one described in Figure 2.1. However, because adaptive routing requires the use of virtual channels and to keep a small crossbar size, instead of having V input multiplexers, each one dedicated to one class of virtual channels, one input multiplexer is used and connects virtual channels of all physical channels belonging to the same cluster.

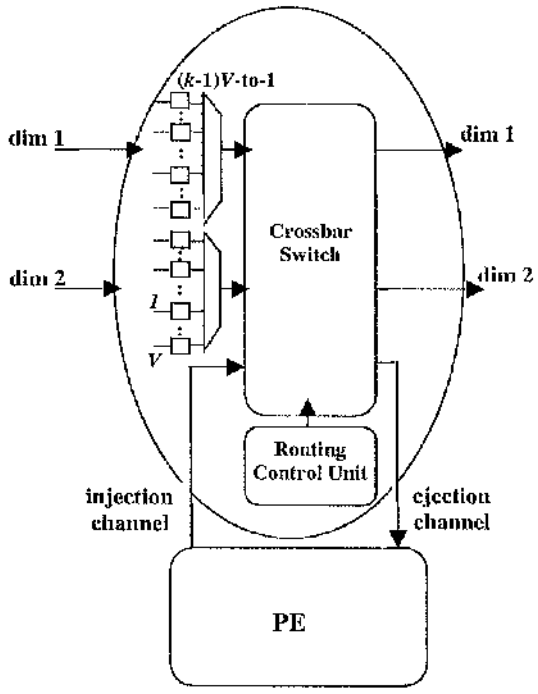


Figure 2.3: DCSH router structure with adaptive routing.

2.4.1 The Model

The model computes the mean message latency in a $k^n (= N)$ DCSH. It is based on the following assumptions, which are widely used in the literature [33, 34, 46, 64, 75, 79].

a) Message destinations are uniformly distributed across the network nodes.

b) Nodes generate traffic independently of each other, and follow a Poisson process, with a mean rate of m_g messages/cycle.

c) Message length is exponentially distributed with a mean of L_m flits, each requiring one-cycle transmission time across a physical channel.

d) Messages generated at the source node are put in a local injection queue of infinite capacity.

e) Messages at the injection are serviced in a FIFO discipline, and requests to multiplexers are serviced following a Round Robin arbitration scheme.

f) When a header flit reaches a router, a routing decision takes place to select the next output channel. The router's decision time takes D_t cycles.

g) V virtual channels are used per physical channel, each of one flit buffer capacity. According to Duato's algorithm, class a contains $(V-1)$ virtual channels and are crossed adaptively. Class b contains one virtual channel, crossed in a deterministic way (e.g. in increasing order of dimensions). If there is more than one adaptive virtual channel available at a given routing step, a message chooses randomly any one of them. Let the virtual channels belonging to class a and b be called adaptive and deterministic virtual channels respectively. To simplify the model derivation, no distinction is made between deterministic and adaptive virtual channels when computing the different virtual channel occupancy

probabilities.

The mean message latency is computed using the following stages. First, the mean network latency, which is the mean time to cross the network, is determined excluding the mean waiting time at the source, and the effects of multiplexing of virtual channels and input multiplexers. Second, the mean waiting time at the source node is evaluated. Third, the average multiplexing degrees that take place at physical channels and at input multiplexers are computed. Finally, the mean message latency is scaled by the multiplexing degrees to obtain the overall message latency.

Let $(a_0...a_{i-1}a_ia_{i+1}...a_{n-1})$ be the node address in a k^n DCSH. The number of nodes, N_j, which are distant j hops away from the source node, is all the combinations of j digits among n and each digit taking any value in the range [0...k-1]. Therefore, N_j is given by

$$N_j = (k-1)^j C_j^n \tag{2.1}$$

where C_j^n is the number of j combinations among the n address digits. Let us define a factor, $T_p(j)$, which reflects the communication pattern. Under uniform traffic, it is equal to

$$T_p(j) = \frac{1}{N-1} \qquad (1 \le j \le n) \tag{2.2}$$

The probability, p_j, that a generated message is sent j hops away from a given source node can be written as

$$p_j = T_p(j) N_j \tag{2.3}$$

The average distance crossed by a message is given by [12]

$$d_{avg} = \sum_{j=1}^{n} jp_j = \frac{\sum_{j=1}^{n} j(k-1)^j C_j^n}{N-1}$$

$$= n\frac{k-1}{k}\frac{N}{N-1} \tag{2.4}$$

Fully adaptive routing allows a message to choose any channel to advance towards its destination, resulting in an equal and balanced traffic load on all channels. Since a router in the DCSH has n output channels and the PE generates, on average, m_g messages in a cycle, which cross, on average, d_{avg} channels, the rate of messages received by each channel, m_c, can be written as [75]

$$m_c = \frac{m_g d_{avg}}{n} \tag{2.5}$$

Following a similar analysis as in [20], we compute first the network latency of $i-hop$ messages, L_i $(1 \le i \le n)$. This latency consists of two parts: one is the delay due to the actual message transmission time, and the other is due to blocking in the network. Given that a message makes i hops to reach its destination, L_i can be written as

$$L_i = i(D_i + 1) + L_m + \sum_{h=i}^{i} B_h^C + W_{qd} \qquad (1 \le i \le n) \tag{2.6}$$

Where L_m is the message length, B_h^C is the mean blocking delay when a message is h hops away from its destination, and W_{qd} is the mean waiting time at the destination.

Let us start by computing the mean blocking delay B_h^C. This delay depends on the probability of blocking, P_h^C, and the mean waiting time when blocking occurs. A message is blocked at a given router if all adaptive virtual channels of the $(h-1)$ dimensions to be visited, and both adaptive and deterministic virtual channels of the lowest dimension to be visited are busy. The case where all adaptive virtual channels of a dimension are busy occurs when either all adaptive and deterministic channels are busy or $(V-1)$ virtual channels are busy. In the latter case, only one case out of the C_{V-1}^V combinations, the $(V-1)$ virtual channels are adaptive. Let P_v^C be the probability that v virtual channels at a dimension are busy (P_v^C is computed below). The probability of blocking at a router when a message is h hops from its destination is given by

$$P_h^C = P_V^C \left(P_V^C + \frac{P_{V-1}^C}{C_{V-1}^V} \right)^{h-1} \tag{2.7}$$

Given that blocking has occurred, a message has to wait for the deterministic virtual channel at the lowest dimension still to be visited. Since adaptive routing distributes traffic evenly among network channel, the waiting time for a deterministic virtual channel is the same across the network, and is approximated by the mean network latency, L. Taking into account Equation (2.7), the mean blocking delay can be written as

$$B_h^C = P_h^C L = P_V^C \left(P_V^C + \frac{P_{V-1}^C}{C_{V-1}^V} \right)^{h-1} L \tag{2.8}$$

The probability, P_v^C, that v virtual channels at a physical channel are busy is calculated

using a Markovian model [34], as shown in Figure 2.4. The state S_v corresponds to v virtual channels at a physical channel are busy. This state passes to S_{v+1} at rate m_c, and passes to the state S_{v-1} at rate $1/L$. However, if the system is in the state S_V (i.e., all virtual channels are busy), when a message releases one virtual channel, a blocked message may occupy it. So, the transition rate from the last stage to the previous one is reduced to account for the arrival of messages while the system is at the last state S_V.

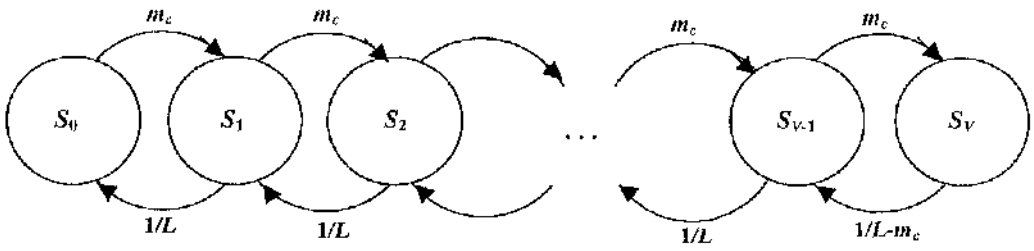


Figure 2.4: Transition diagram to compute the occupancy probabilities of virtual channels in the DCSH.

At steady state, the solution of the model is given by the set of recurrence relations [34]

$$P_j^C = \begin{cases} \dfrac{1}{\sum_{l=0}^{V} q_l^C} & j = 0 \\ P_{j-1}^C m_c \mathrm{L} & 0 < j < V \\ P_{j-1}^C \dfrac{m_c}{1/L - m_c} & j = V \end{cases} \qquad (2.9)$$

Where q_j^C is an intermediate variable used in the calculation of P_j^C and it is given as

$$q_j^C = \begin{cases} 1 & j = 0 \\ q_{j-1}^C m_c L & 0 < j < V \\ q_{j-1}^C \dfrac{m_c}{1/L - m_c} & j = V \end{cases} \tag{2.10}$$

The mean waiting time, W_{qd}, seen by a message when it arrives at the destination node, is the mean waiting time to acquire the ejection channel, and is computed as follows. At the steady state, the traffic generated by each node is equal to the traffic received. Therefore, W_{qd} can be written as [80]

$$W_{qd} = m_g L_m^2 \tag{2.11}$$

Once the network latency, L_i, of an $i-hop$ message is calculated, the mean network latency, L, is obtained by simply averaging L_i over all hops $(1 \le i \le n)$. So, L can be given by

$$L = \sum_{i=1}^{n} p_i L_i \tag{2.12}$$

It can be noticed that L_i, through Equation (2.8), is a function of L, while Equation (2.12) shows that L is a function of L_i. Given that a closed-form solution to this inter-dependency is very difficult to determine, L_i and L are calculated using iterative techniques for solving equations [20].

The effects of waiting, which occur at the source node, must also be included. A message at the source can enter the network using any of the V virtual channels; through either the deterministic virtual channel or one of the $(V-1)$ adaptive ones. The arrival rate of messages to a single virtual channel can be approximated by m_g / V. The mean service

time seen by a message entering the network is equal to the mean network latency, L. Therefore, modelling each virtual channel at the source as an M/M/1 queue, yields the mean waiting time at the source, W_{qs} as [80]

$$W_{qs} = \frac{m_g}{V} L^2 \tag{2.13}$$

Including multiplexing effects at physical channels and input multiplexers, the average multiplexing degree of virtual channels, that takes place at a given physical channel, is found to be [34]

$$V_{avg} = \frac{\sum_{j=0}^{V} j^2 P_j^c}{\sum_{j=0}^{V} j P_j^c} \tag{2.14}$$

Inspecting Figure 2.3 reveals that messages in the DCSH also encounter multiplexing delay at the input multiplexers when crossing a given router. The average multiplexing degree of input multiplexers can be found using a similar Markovian chain as in Figure 2.4. The number of states in this case is $(k-1)V$ and the arrival rate is m_c. While the departure rate is $1/L$ for all stages S_j $(0 \le j < (k-1)V)$ and $1/L - m_c$ from the state $S_{(k-1)V}$. In steady state, the solution is given by

$$P_j^m = \begin{cases} \dfrac{1}{\sum_{l=0}^{V} q_l^m} & j = 0 \\ P_{j-1}^m m_c L & 0 < j < (k-1)V \\ P_{j-1}^m \dfrac{m_c}{1/L - m_c} & j = (k-1)V \end{cases} \tag{2.15}$$

$$q_j^m = \begin{cases} 1 & j = 0 \\ q_{j-1}^m m_c L & 0 < j < (k-1)V \\ q_{j-1}^m \dfrac{m_c}{1/L - m_c} & j = (k-1)V \end{cases} \tag{2.16}$$

The average multiplexing degree at each input multiplexer is therefore given as

$$M_{avg} = \frac{\sum_{j=0}^{V(k-1)} j^2 P_j^m}{\sum_{j=0}^{V(k-1)} j P_j^m} \tag{2.17}$$

Finally, including the mean waiting delay at the source, and the multiplexing effects at input multiplexers and physical channels, overall message latency in the network can be written as

$$Latency = [W_{qs} + M_{avg} L] V_{avg} \tag{2.18}$$

2.4.2 Validation

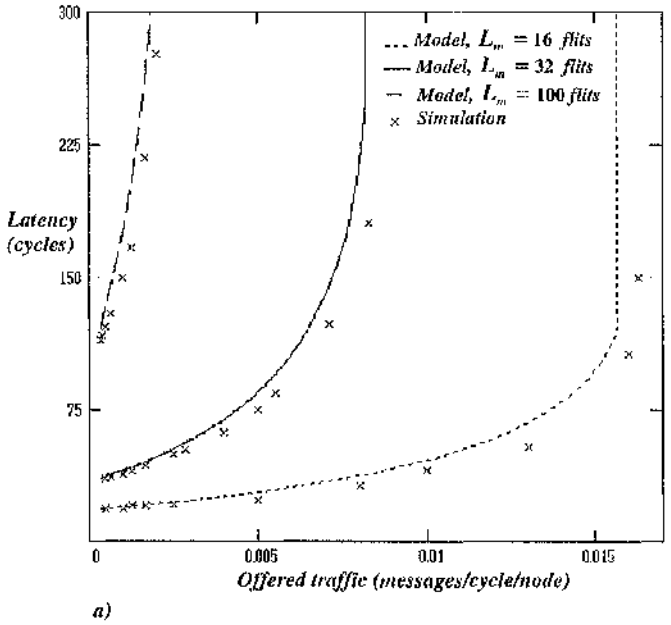
The above model has been validated by means of a discrete-event simulator, written in C code, and operating at the flit level. The simulation uses the same assumptions as the analysis. Some of these assumptions are detailed here with a view to make the network operation clearer. The network cycle represents the transfer time of a single flit through a physical channel. Messages are generated at each node following Poisson inter-arrival time distribution of an average m_g messages/cycle. Message lengths are exponentially distributed with an average of L_m flits. Destination nodes are determined using a uniform random number generator.

Each simulation experiment was run until the network reaches its steady state, that is until a further increase in simulated network cycles does not change the collected statistics appreciably. The average message latency is calculated as the mean amount of time from the time a message is generated until the last data flit is removed from the network. The other measures include the mean network latency, and the mean waiting time at the source and destination nodes.

Extensive simulation experiments have been conducted to validate the model for several combinations of network sizes, message lengths and virtual channels. Here, for the sake of illustration, latency results are presented for the following parameters on a two-dimensional DCSH of 256 nodes.

- The average message length L_m has been varied between 16, 32 and 100 flits, representing short and long messages respectively.
- The router delay, D_r, has been set to zero and then to two cycles, reflecting the cases when router delay is ignored and when it is taken into account.
- The number of virtual channels V is set to 2 and 4. In both cases one virtual channel crossed in a deterministic manner and the rest traversed adaptively.

Figures 2.5 and 2.6 show the mean message latency predicted by the model against simulation results as a function of the message rate generated by each node. The x-axis in the figures (and the figures of the following chapters, unless otherwise stated) represents the traffic rate i.e., the rate at which a node injects messages into the network in a cycle. The y-axis shows the mean message latency in crossing from source to destination. The results generally show a close agreement between the model and simulation.



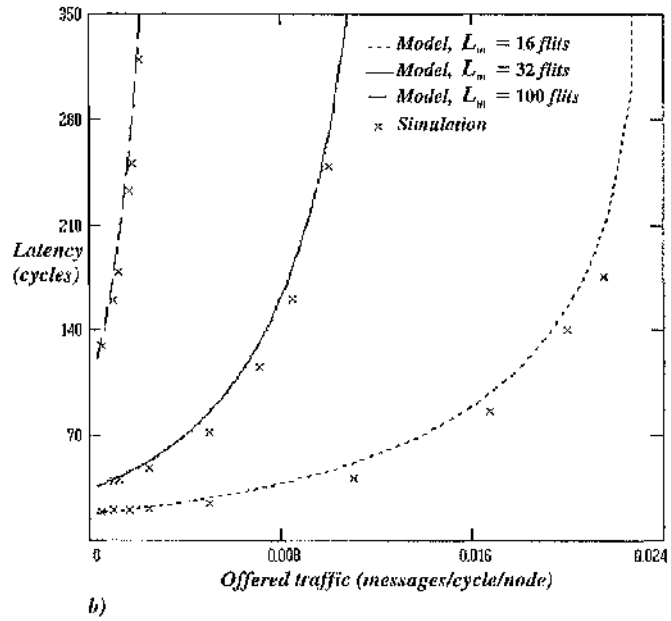
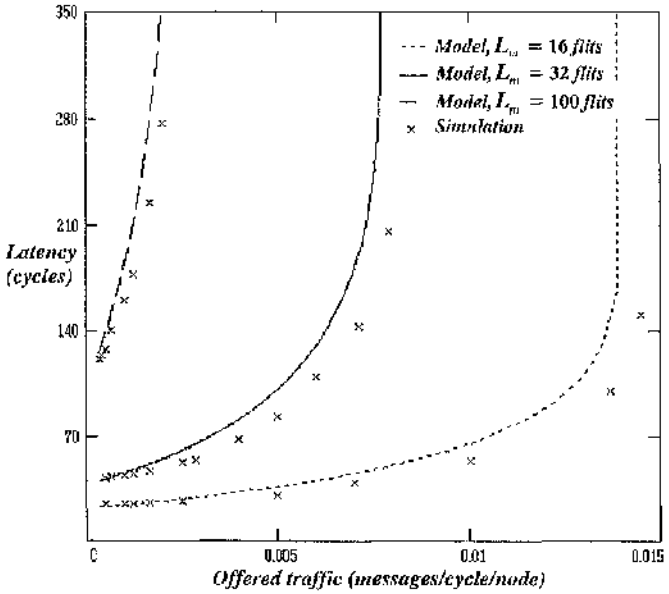
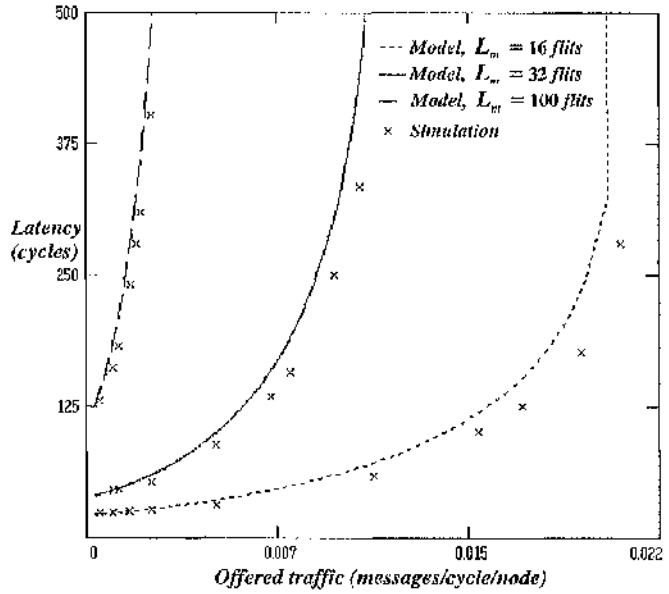


Figure 2.5: Latency predicted by the model against simulation results in 2D-DCSH, D_f=0. a) V=2, b) V=4.



a)



b)

Figure 2.6: Latency predicted by the model against simulation results in 2D-DCSH, D_f=2. a) V=2, b) V=4.

2.4.3 Impact of Virtual Channels on the DCSH Performance

Dally [34] has shown that the use of extra virtual channels as *"bypass lanes"* reduces message blocking, and thus improves network performance. This section uses the model developed in the previous section to assess to what extent the DCSH can benefit from extra virtual channels to improve its performance. For illustrative purposes, the results are shown for a two-dimensional configured medium system of 256 nodes, with an average message length equal to 32 flits. The number of virtual channels has been varied between two and five, keeping one virtual channel deterministic and the rest adaptive to obtain maximum performance, as suggested in [47].

Figure 2.7 reveals that at light traffic load, the increase in the number of virtual channels does not bring any improvement to the network performance. This is because under low traffic intensity, messages do not experience contention in the network. Under moderate to high traffic loads, there is an improvement in the performance when the number of virtual channels is increased from two to three. Indeed, the addition of the third virtual channel reduces the load on the two previous channels, thereby reducing the message blocking delay and latency.

The fact of adding a fourth and a fifth virtual channel does not improve appreciably the DCSH performance. As reported in [34], the use of virtual channels has two opposite effects. At heavy loads, a high number of virtual channels reduce message blocking at physical channels, and therefore reduces message latency. On the other hand, because messages share the bandwidth of the same physical channel, the multiplexing effect increases message latency. Hence, we can conclude that three virtual channels are sufficient to enable the DCSH to achieve its best performance.

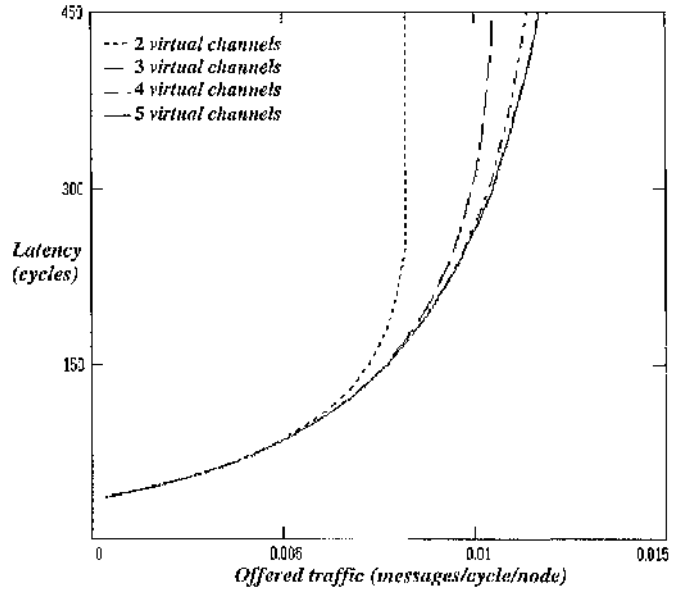


Figure 2.7: Impact of virtual channels on the DCSH performance.

2.5 Conclusion

This chapter has proposed an analytical model to compute message latency in the DCSH, using Duato's adaptive routing algorithm. The model has been validated through simulation. Latency results predicted by the model have shown close agreement with those provided by simulation. The model has been used to investigate the effects of virtual channels on the performance of the DCSH. The results have revealed that using three virtual channels allows the DCSH to achieve its best performance. Adding more virtual channels after that does not yield any further performance gains.

The modelling approach adopted here will be used in the next two chapters to develop analytical models for other routing algorithms and for comparing performance of the DCSH with k-ary n-cubes.

Chapter 3

Evaluation of Routing Algorithms in the DCSH

3.1 Introduction

This chapter investigates the impact of routing algorithms on DCSH performance. Three routing algorithms are considered in this analysis. The first is deterministic and is the well-known e-cube. The two other algorithms are fully adaptive. The first one is based on Duato's method and has already been discussed in the previous chapter. The second is an algorithm recently proposed by Chalasani and Boppana [18], and will be known throughout this chapter as the *hop-based* algorithm. The three algorithms differ in the degree of flexibility given to messages to cross dimensions and the way virtual channels are allocated to messages.

Several studies have been conducted to analyse the performance of routing algorithms [17, 44, 52, 119, 123], focusing mainly on the two commonest variants of the k-ary n-cubes: the torus and mesh. Most of these studies have given each routing algorithm as many virtual channels as required for deadlock prevention and different conclusions have been drawn, depending on the topology. For example in the mesh, Pertel [119] has found that under uniform traffic, deterministic routing performs better than adaptive routing. The latter concentrates the traffic in the central part of the mesh when looking for available paths and, consequently, it creates congestion in that region, and increases message latency. In [123], it has been shown that with higher number of virtual channels for both deterministic and adaptive routing algorithms, the latter improves the performance of the mesh. Concerning the torus, however, there has been a consensus between researchers [17, 44], where they have shown that adaptive routing improves the performance of the torus.

For fairness, the present study of routing in the DCSH takes into consideration equal buffer storage of physical channels for deterministic and adaptive routing algorithms. The results are presented for uniform and non-uniform traffic patterns. Under the former pattern, the comparison is undertaken mathematically. However, for the latter scheme the results are obtained from simulation experiments, since modelling such a communication scheme is a complicated undertaking.

3.2 Outline of the Models

In this section, analytical models for the hop-based and deterministic routing algorithms are derived and validated through simulation.

3.2.1 The Hop-based Algorithm

Deadlock prevention in the hop-based routing algorithm is based on the concept of a structured buffer pool, the method used to avoid the deadlock problem in S&F networks. There is no restriction on traversing dimensions, since the algorithm is fully adaptive. Nevertheless, an ordering on the usage of virtual channels is established to guarantee deadlock-freedom for messages. If we assume that virtual channels are numbered in a decreasing order, a message arriving at a router needing i hops to reach its destination can only use the i^{th} virtual channel of any of the i remaining dimensions to cross. This removes the cyclic dependency between channels and the algorithm is deadlock-free. The advantage of this algorithm is that it can be applied to any topology. However, its implementation requires a high number of virtual channels per physical channel. This number must be at least equal to the network diameter, which makes it impractical in high-diameter networks, such as k-ary n-cubes.

The approach followed to derive the mean message latency for the hop-based algorithm is similar to that presented in chapter 2. Additionally, the same assumptions as in chapter 2 are used here, and are briefly recapped here for sake of clarity. Messages are generated following Poisson inter-arrival time distribution. Message lengths are exponentially distributed with a mean of L_m flits. System nodes have equal probability to receive any generated message. However, the assumption concerning the allocation of virtual channels to messages is stated as follows. Each physical channel is split into V virtual channels, which is equal to the DCSH diameter. At any step of its journey, a message can use only one virtual channel at any of the remaining dimensions to visit.

The model developed for Duato's algorithm is still valid for the hop-based algorithm.

However, the calculation of the mean message blocking delay at a given dimension, B_h^C, differs. A message h hops away from its destination gets blocked at a router, if the h^{th} virtual channel at each of the h remaining dimensions is busy. Recall that, in chapter 2, it has been assumed that there is no distinction between virtual channels when the virtual channel occupancy probabilities are calculated. However, the probability that a specific virtual channel is busy is needed here. Holding the above assumption, similar results are yielded, since the number of combinations is the same for any virtual channel taken as the one in question.

In general, when j virtual channels among the V virtual channels at a given physical channel are busy, there are C_j^V combinations. Only C_{j-1}^{V-1} cases out of C_j^V, the virtual channel required that is busy. That event occurs when there is at least one virtual channel, at a physical channel, is busy. Thus, the probability that the required virtual channel at a dimension is busy can be written as

$$P^C = \sum_{j=1}^{V} \frac{P_j^C}{C_j^V} C_{j-1}^{V-1} \tag{3.1}$$

Once P^C is calculated, the probability that the required virtual channels at the h dimensions are busy is $(P^C)^h$. If this event happens, the message has to wait on average L cycles. Therefore, the mean message blocking delay can be written as

$$B_h^C = \left(P^C\right)^h L \tag{3.2}$$

3.2.2 The Deterministic Algorithm

In deterministic routing, messages cross dimensions in a predefined order. Although doing this is sufficient to avoid the deadlock problem in the DCSH, Dally [34] has shown that virtual channels improve the deterministic routing performance over single deep queues provided at the physical channels. The allocation of the physical channel and buffers then is decoupled, which damps the effect of chained blocking caused by wormhole routing. Therefore, to make a fair comparison, this study considers the deterministic routing implemented with the virtual channel concept.

The model is based on the same assumptions used in the derivation of the previous models. Moreover, it is assumed that messages cross dimensions in an increasing order (dimensions are numbered from 0 to n-1). At a given dimension, a message can choose any of the V virtual channels to progress towards its destination. Figure 3.1 shows the diagram used to derive the mean message latency.

A message generated at the source sees a mean latency, *Latency*, to reach its destination. Furthermore, a message can enter the network through any of the n dimensions, depending on its source and destination addresses. So, when a message enters the network through dimension i, it sees a network latency S_i to reach the destination. Similarly, after exiting dimension i, the message requires a mean service time T_i to reach its destination.

Let $p_s = 1/k$ be the probability that a message skips a dimension, and let m' be the mean message rate crossing a given path in the network. In the DCSH case, m' is simply equal to m_g because channels are uni-directional. The mean message rate generated by a given node and entering the network through dimension i $(0 \le i < n)$, R_i^L, skipping the

j $(0 \le j < i)$ lowest dimensions is given by

$$R_i^L = (1 - p_s)\, p_s^i m' \tag{3.3}$$

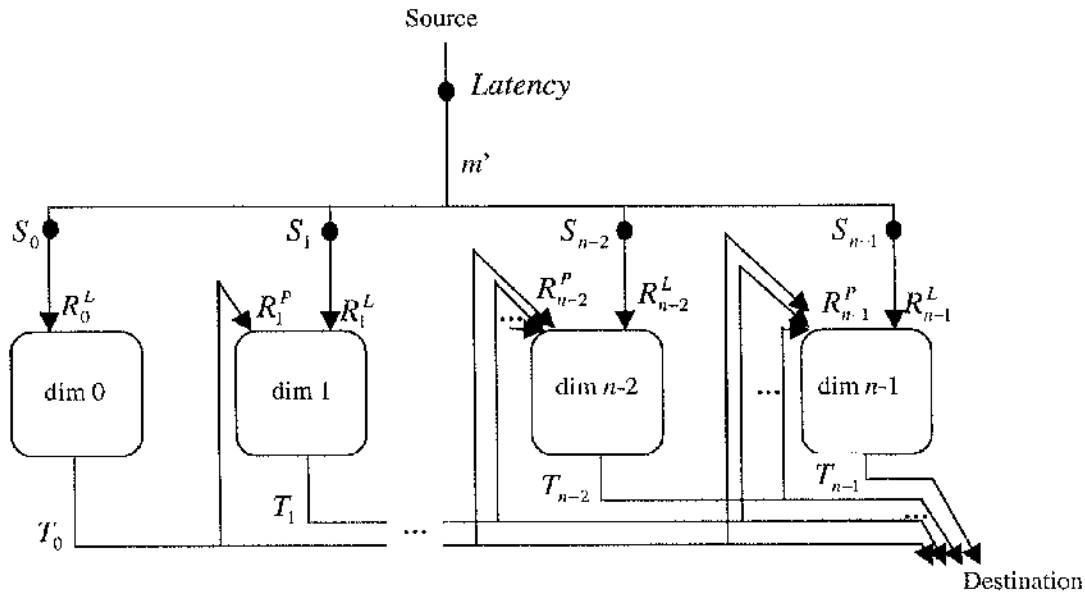


Figure 3.1: Diagram of a message path in the network.

The probability that a message arriving on dimension j $(0 \le j < n)$ and crosses at its next step dimension i $(j < i < n)$ is $(1 - p_s)^2\, p_s^{i-j-1}$, where $(1 - p_s)^2$ represents the probability that the message crosses dimensions j and i, and p_s^{i-j-1} represents the probability that the message skips dimensions between j and i. So, the total message rate entering dimension i from the lowest dimensions j $(0 \le j < i)$, R_i^P, can be written as

$$R_i^P = \begin{cases} 0 & i = 0 \\ \sum_{j=0}^{i-1} (1-p_s)^2 \, p_s^{i-j-1} m' & 0 < i < n \end{cases} \tag{3.4}$$

The sum of the rates R_i^L and R_i^P at any dimension i is found to be $\Phi = (1-p_s)m'$. The network latency, S_i, of a message entering the network through dimension i is the mean service time seen when exiting that dimension, T_i, increased by the mean blocking delay, T_i^c, at that dimension. Hence, S_i can be given by

$$S_i = T_i^c + T_i \tag{3.5}$$

A message gets blocked at dimension i if all the virtual channels, V, at that dimension are occupied, this happens with the probability $P_{i,V}^C$. Given that blocking has occurred, the message waits for an average time T_i^b to acquire one virtual channel. T_i^c is given then as

$$T_i^c = P_{i,V}^C T_i^b \tag{3.6}$$

Two cases need to be considered when calculating T_i^b [19]. The first case happens when at least one of the blocking messages will terminate after crossing dimension i. The probability that this event happens is $1-(1-p_s^{n-i-1})^V$, where p_s^{n-i-1} is the probability that a message exits the network through dimension i. In this case, the blocked message waits for the blocking message to acquire the ejection channel and its transfer time through that channel (i.e., $W_{qd} + L_m$). The second case happens when all the blocking messages need to cross subsequent dimensions to progress in the network, and the probability that this event occurs is $(1-p_s^{n-i-1})^V$. In this case, the blocked message waits an average time equal to the time required by the blocking message to release the virtual channel. This time includes

the mean blocking delays at subsequent dimensions j $(i < j < n)$ (each with the probability p_s^{j-i-1}; the probability that the blocking message will cross dimension j and having already crossed dimension i), and at the ejection channel, increased by the actual transfer time of the message through the physical channel. Taking into account all these possibilities, T_i^b can therefore be written as

$$T_i^b = \left[1-(1-p_s^{n-i-1})^V\right](W_{qd}+L_m)+(1-p_s^{n-i-1})^V\left[\sum_{j=i+1}^{n-1} p_s^{j-i-1}(D_i+T_j^c)+W_{qd}+L_m\right] \qquad (3.7)$$

Once the message has acquired a virtual channel at dimension i, T_i, the mean service time to reach its destination, is the mean blocking delays at the subsequent dimensions j $(i < j < n)$ and at the ejection channel, increased by the transmission time through the ejection channel. Thus, T_i is given by

$$T_i = \sum_{j=i+1}^{n-1} p_s^{j-i-1}(D_i+T_j^c)+W_{qd}+L_m \qquad (3.8)$$

To calculate $P_{i,V}^C$ a Markov chain similar to that described in chapter 2 (Figure 2.4) is used. The transition rate from a stage to the next one is the total message rate crossing dimension i, which is Φ. While the transition rate exiting a stage to the previous one is $1/T_i$, except the last one, where the transition rate from that stage is $1/T_i - \Phi$.

Taking into account all the possible ways that a message can take to enter the network, the mean network latency L, excluding the effects of multiplexing at physical channels and input multiplexers, and the mean waiting time at the source, can be written as

$$L = d_{avg} + \sum_{i=0}^{n-1}(1-p_s)p_s^i S_i \qquad (3.9)$$

The average multiplexing degree at dimension i is given by

$$V_{avg}^{i} = \frac{\sum_{j=0}^{V} j^2 P_{i,j}^{C}}{\sum_{j=0}^{V} j P_{i,j}^{C}} \tag{3.10}$$

The average multiplexing degree of virtual channels in the network can be obtained as the average of multiplexing degrees at all dimensions. It is given, thus, as follows

$$V_{avg} = \frac{\sum_{i=0}^{n-1} V_{avg}^{i}}{n} \tag{3.11}$$

Similarly, the average multiplexing degree at input multiplexers in the network can be written as

$$M_{avg} = \frac{\sum_{i=0}^{n-1} M_{avg}^{i}}{n} \tag{3.12}$$

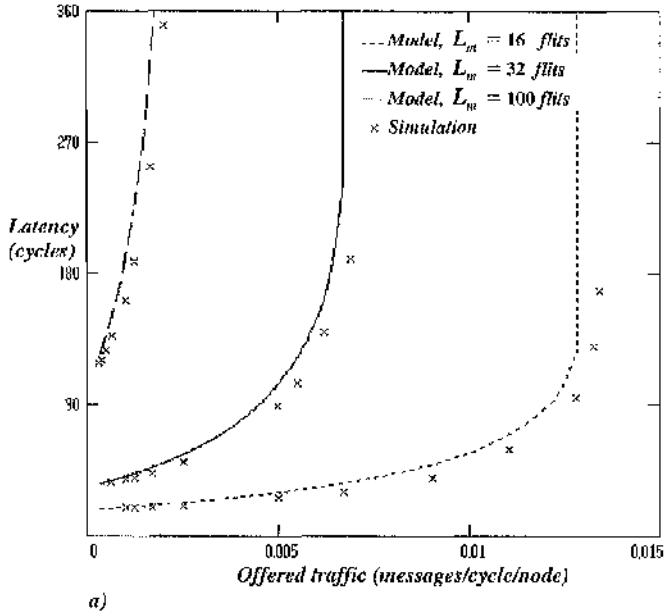
Where M_{avg}^{i} is the multiplexing degree at the input multiplexer at dimension i. Finally, the mean message latency in the network, including the mean waiting time at the source node and the multiplexing effects at physical channels and input multiplexers, and taking into consideration the average time crossed by the header, is found to be

$$Latency = [W_{qs} + M_{avg} L]V_{avg} \tag{3.13}$$

Where W_{qs} is the mean waiting time at the source and is calculated the same way as with adaptive models (Equation (2.13)).

3.2.3 Validation of the Models

Latency results provided by the models and simulation experiments are shown in Figures 3.2 through 3.4 for two-dimensional DCSH of 256-node size. The simulators use the same assumptions as those used by the models. Also, the results shown here are based on the same parameters as those set with Duato's model validation (chapter 2). In general, the figures show a close agreement between the message latency predicted by the models and simulation results.



a)

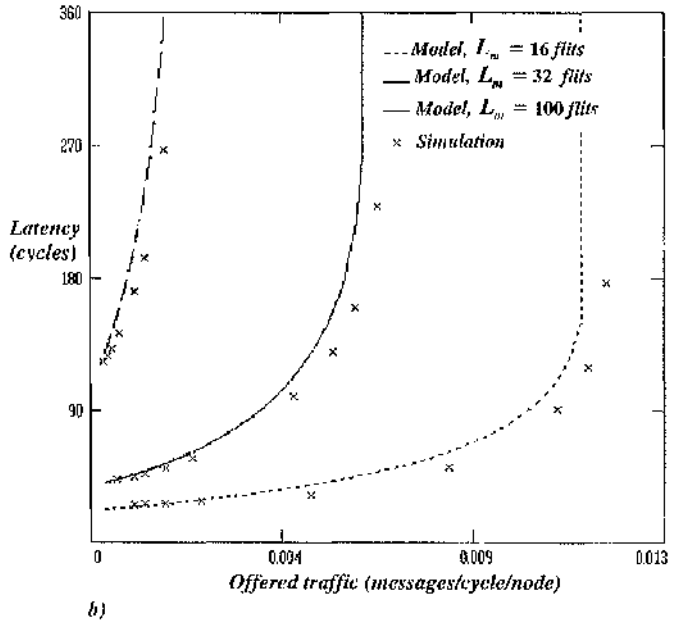
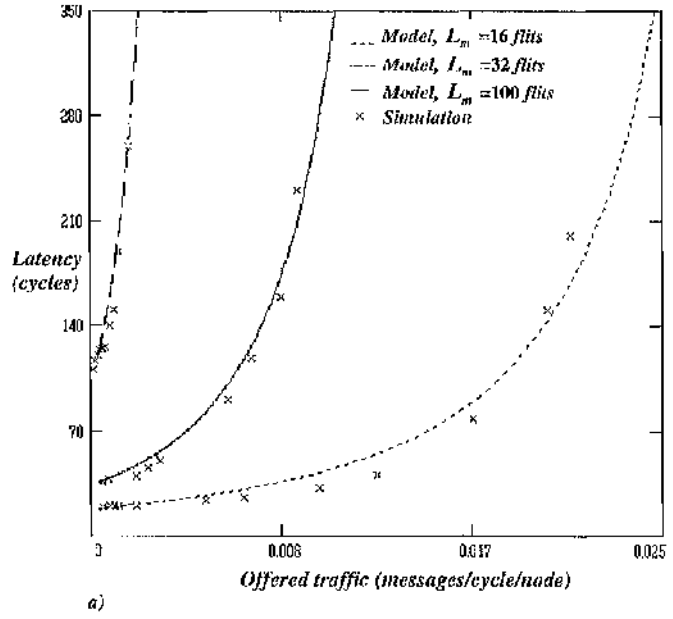


Figure 3.2: Validation results of the hop-based algorithm model, V=2.

a) $D_t = 0$, b) $D_t = 2$.



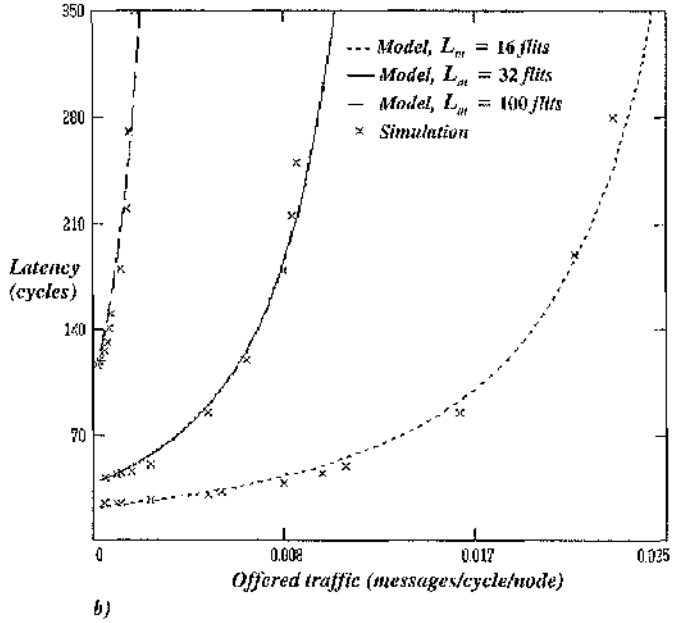
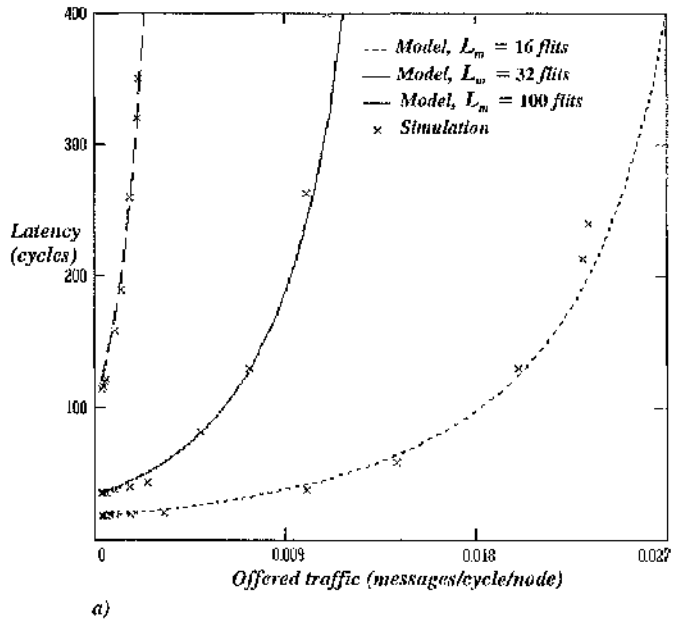


Figure 3.3: Validation results of deterministic algorithm model, V=2.

a) $D_t = 0$, b) $D_t = 2$.



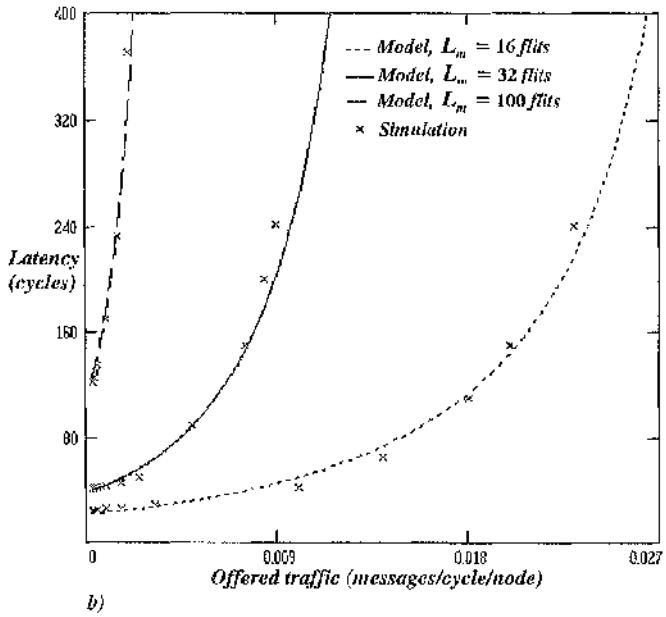


Figure 3.4: Validation results of deterministic algorithm model, V=4.
a) $D_t = 0$, b) $D_t = 2$.

3.3 Performance under Uniform Traffic

This section presents the performance of the DCSH with the three routing algorithms: deterministic routing (*det*), Duato's algorithm (*Duato's*), and the hop-based algorithm (*hop-b*), under uniform traffic distribution. Measurements of the mean message latency are obtained from queuing models developed in the previous section and in chapter 2.

For illustrative purposes, latency results shown in this section and in the next section are obtained for the DCSH size of 256 nodes. The results for larger system sizes do not differ qualitatively from those presented here. The three algorithms use equal buffering capacity

at physical channels. The number of virtual channels has been fixed at two, the minimal number required by the two adaptive routing algorithms to guarantee message deadlock-freedom.

Figure 3.5 depicts latency results for adaptive and deterministic routing algorithms as a function of the message rate injected into the network, for short and long messages of 32 and 128 flits, respectively. Such figures have been widely considered in the literature [1, 2, 19, 119]. Moreover, router delay has been set to zero for the three routing algorithms. Although, somewhat unrealistic even in the deterministic case (and is even more so when the complexity of adaptive router hardware is taken into account), this does not affect the conclusion drawn.

Under light traffic loads, the figure shows that the deterministic algorithm performs similarly to the other two. This is because when the traffic intensity in the network is low, the probability that messages get blocked is negligible, and therefore all approaches behave almost identically.

As the traffic increases, however, the three algorithms behave differently due to the way message contention is handled by each algorithm. The figure reveals that the deterministic algorithm performs better than the hop-based algorithm under moderate traffic load, and better than both adaptive algorithms under heavy traffic loads. Interestingly this holds even when the extra hardware complexity of adaptive routers is ignored. The reason is that the number of paths between any pair of nodes in the two-dimensional DCSH is small and the improvement in blocking delay of message headers when using adaptive routing is modest when the traffic is uniformly distributed in the network. However, since adaptive algorithms allow messages to cross virtual channels of any dimension in order to progress

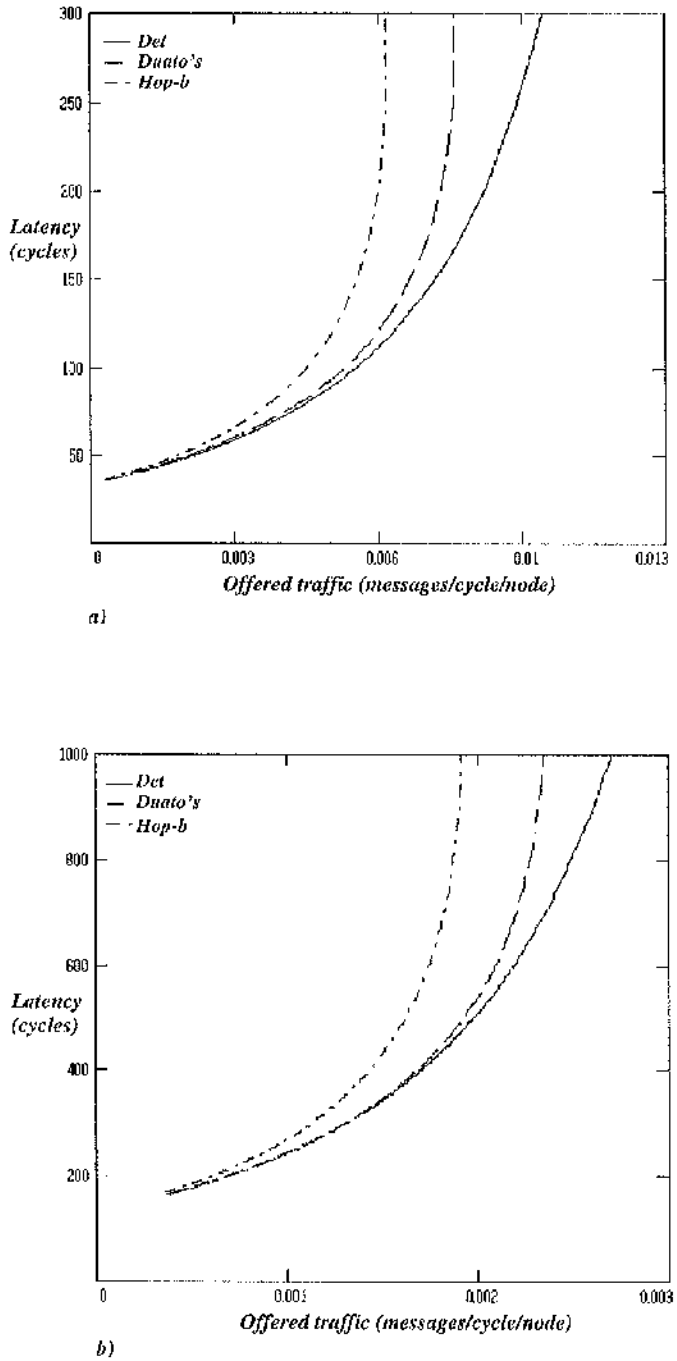


Figure 3.5: Comparison results of deterministic and adaptive routing algorithms under uniform traffic. a) $L_m = 32$ flits, b) $L_m = 128$ flits.

in the network, this results in higher utilisation of physical channels, which is normally a prime advantage of adaptive routing algorithms. However, here higher utilisation of physical channels translates into longer service time of messages due to the multiplexing effects at both physical channels and input multiplexers, resulting in higher message latency.

Moreover, when comparing the performance of Duato's algorithm to that of the hop-based algorithm, the results show that the latter saturates more quickly. There are two reasons for this. Firstly, the number of virtual paths offered to messages in the hop-based algorithm is smaller than in Duato's algorithm. Secondly, the disturbance of the traffic caused by adaptivity in Duato's algorithm is less than in the hop-based algorithm. In the former, one virtual channel is crossed in a deterministic way, which preserves the uniformity of a fraction of the traffic, while in the latter, all virtual channels are crossed adaptively.

3.4 Performance under Non-uniform Traffic

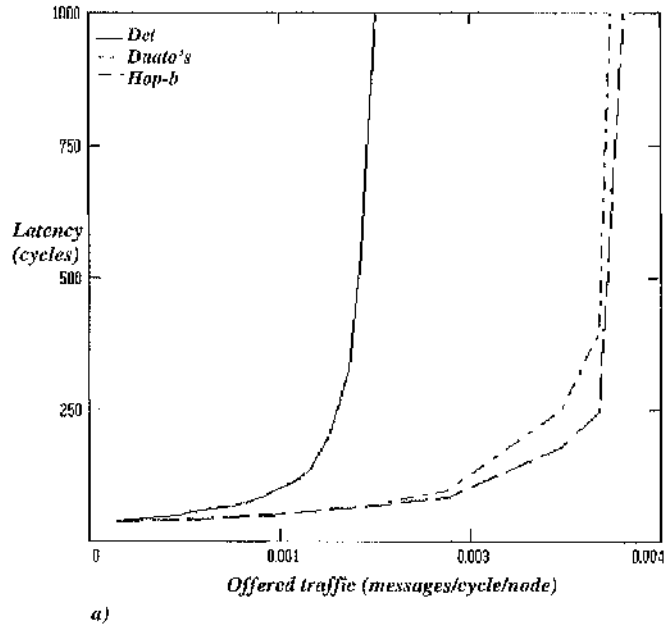
This section extends the performance analysis of the three routing algorithms to non-uniform traffic. Although there are several ways to generate non-uniform traffic [38, 43], the results presented in this section are based on matrix transpose communication scheme (in the general case, a node whose address is $a_0a_1....a_{n-2}a_{n-1}$ communicates with the node whose address is $a_{n/2}...a_{n-1}a_0...a_{(n/2)-1}$). This communication scheme has been selected because it represents the highly non-uniform traffic pattern. Note that other permutation schemes such as bit reversal and perfect-shuffle yield similar conclusions.

Figure 3.6 shows the performance of deterministic and adaptive routing algorithms in

terms of the mean message latency, for 32 and 128 flits, respectively. The number of virtual channels is fixed at two, and the router delay in the three algorithms is set to zero in order to compare the results to those obtained for uniform traffic. The effects of increase in router delay due to adaptivity will also be examined later.

The figure reveals that under light traffic loads, deterministic routing provides similar performance to that of adaptive routing algorithms and this for short and long messages. However, if the traffic in the network is increased the deterministic algorithm performs poorly from moderate loads while Duato's algorithm gives the best performance followed by the hop-based algorithm. With deterministic routing, messages generated by a source are sent to the same destination, following the same path. In addition, messages generated by nodes of the same cluster compete at the input multiplexer of the same intermediate router. At heavy loads, this increases the multiplexing delay for messages at input multiplexers and their waiting time at the source. Adaptive routing algorithms allow messages to reduce their blocking delays by circumventing heavily loaded channels, and spreading the load in the network.

Again, Duato's algorithm exhibits better performance than the hop-based alternative. Both algorithms use equal numbers of virtual channels and messages have flexibility to choose physical channels of any non-corrected dimension to progress in the network. The main difference remains in the allocation of virtual channels to messages. When a message arrives at a router, it can use only one virtual channel at any dimension in the hop-based algorithm, while Duato's allows a message to use any virtual channel at any dimension, giving it more chance to reduce blocking delays, and therefore latency, still further.



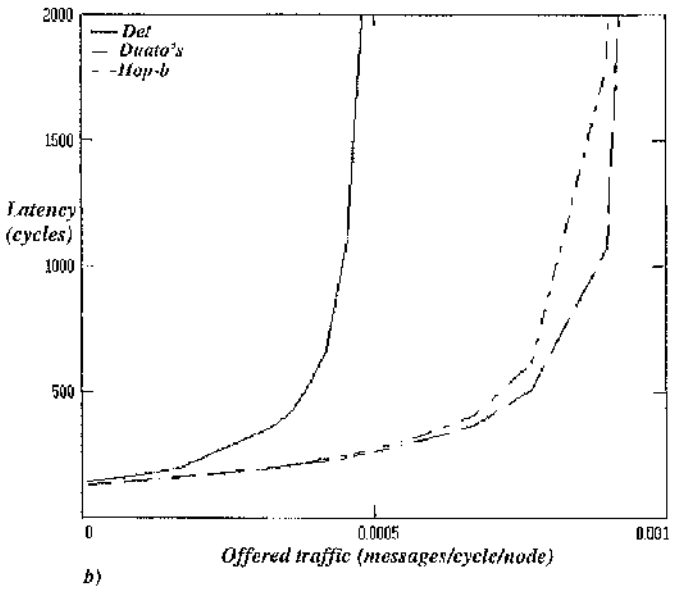


Figure 3.6: Comparison results of deterministic and adaptive routing algorithms under non-uniform traffic. a) $L_m = 32$ flits, b) $L_m = 128$ flits.

Figure 3.7 shows the impact of router delay on the performance of the three algorithms. The results are shown for 32 flits but the conclusion has been verified for long messages as well. The figure shows that increasing router delay (two cycles) for the adaptive algorithms, while continuing to ignore it in the deterministic case, does not affect the performance of the former. They still provide lower message latency than the deterministic algorithm, because the blocking delays encountered by messages when looking for an available path is more important than the waiting delay to decide their route.

The analysis has also investigated whether the deterministic algorithm performance can approach that of the hop-based and Duato's algorithms, when equipped with more virtual channels. The number of these in the deterministic algorithm is increased to four with router delay again ignored, while the adaptive algorithms get only two virtual channels and a router delay set to two cycles. Once again, latency results in Figure 3.8 show that even with fewer virtual channels than in the deterministic algorithm, adaptive algorithms greatly outperform the deterministic routing. As pointed in [52], the virtual channel concept alone is not able to reduce message contention when the traffic is very heavy, notably with highly non-uniform traffic patterns. Adaptivity is the only means whereby such contention can be reduced by allowing messages to circumvent heavily loaded regions of the network.

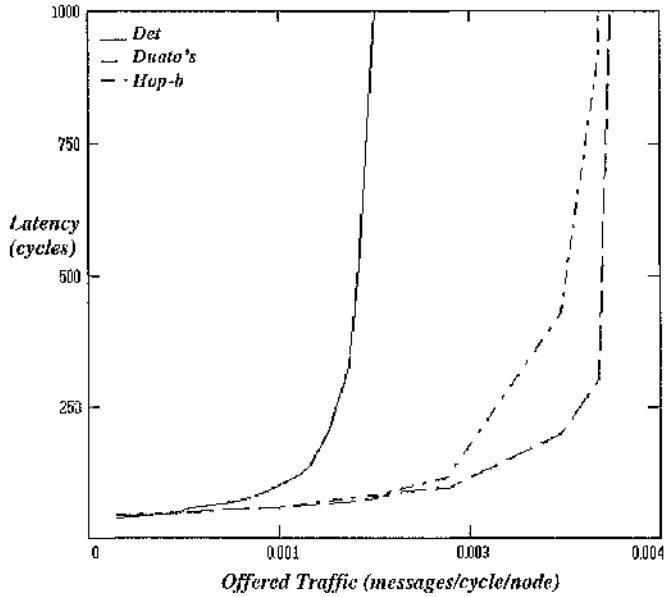


Figure 3.7: Effects of router delay on the performance of adaptive algorithms with non-uniform traffic, $L_m = 32$ flits.

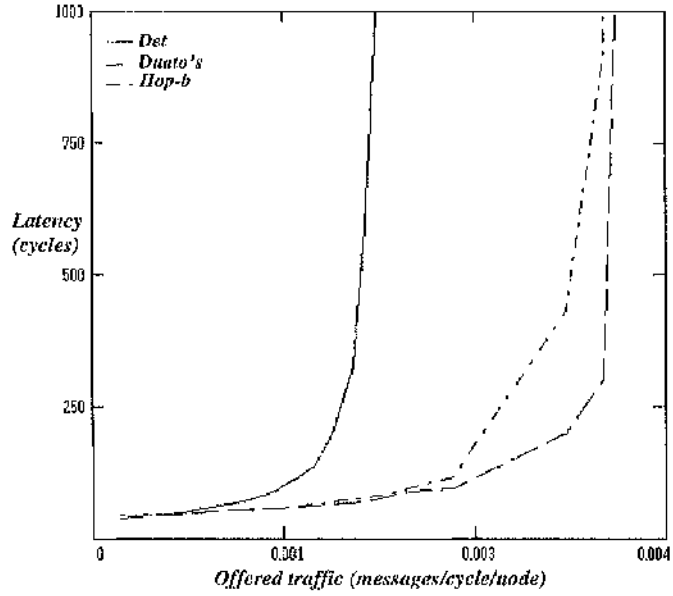


Figure 3.8: Effects of virtual channels on the performance of the deterministic algorithm with non-uniform traffic, $L_m = 32$ flits.

Another interesting finding is that an increase in the number of virtual channels makes little difference to the performance of the deterministic routing. Unlike uniform traffic, where messages generated by a source may have different destinations, in matrix transpose communication, messages are sent to the same destination. At high traffic intensity, increasing the number of virtual channels reduces message waiting time at the source, but, on the other hand, the message rate arriving at the input multiplexer of the intermediate router is also increased. These messages require the same outgoing channel at that router, so increasing message contention.

3.5 Conclusion

In this chapter, three different routing algorithms have been investigated for the DCSH. Two of these are adaptive, the hop-based algorithm and Duato's algorithm. The third is the deterministic routing with the virtual channel concept. The analysis has shown that under uniform traffic, the deterministic algorithm performs better than the two adaptive algorithms at heavy traffic loads. Adaptive routing algorithms disturb the uniformity of the traffic, which overwhelms the modest benefit of adaptivity in decreasing blocking delays of message headers.

Under non-uniform traffic, however, the blocking delays of messages when crossing channels is more considerable than the effects of virtual channel multiplexing and adaptive mechanisms are the only way to reduce such delays. These provide better performance for the DCSH than simple deterministic routing, even when the latter is equipped with more virtual channels.

Duato's algorithm is demonstrably superior to the hop-based algorithm under all the conditions examined. While it benefits from the deterministic virtual channel to keep the uniformity of a traffic fraction under a uniform traffic distribution, it uses virtual channels more efficiently to decrease message blocking delays under non-uniform traffic.

Chapter 4

Performance of the DCSH and *k*-ary *n*-cubes with Adaptive Routing

4.1 Introduction

K-ary *n*-cubes are the most popular networks used in the implementation of current generation of multicomputers [29, 74, 118, 120]. The move towards this type of network is mainly due to the analysis presented in [33], where the results have shown that under the bisection width constraint, *k*-ary *n*-cubes outperform hypercubes due to their wider channels. The other reason is due to their perceived *modularity* [131]; they can be expanded simply by adding nodes and channels without changes in the node structure. Unfortunately, as pointed in [112], this modularity is at the expense of performance, for a fixed-degree network, as the size grows the channels must be increased in bandwidth to maintain the same latency [112, 124]. The number of pins on a torus node chip must

therefore be increased with system size, a fact which is obvious in a topology like the DCSH whose degree increases as the number of nodes grows, but less apparent for the torus.

Most practical multicomputers employ deterministic message routing to ensure simple hardware implementation of routers. A performance study of the DCSH and k-ary n-cubes using this form of routing has been reported in [114], where the results have revealed that the DCSH exhibits superior performance over k-ary n-cubes. K-ary n-cubes, low-dimensional versions in particular, continue to be favoured topologies even in more recent systems, which have incorporated adaptive routing to reduce communication latency. This chapter evaluates the relative performance merits of the DCSH and k-ary n-cubes in the context of adaptive routing. It assesses to what extent the DCSH and k-ary n-cubes can exploit adaptivity to provide fast communication. To this end, an analytical model of adaptive routing, based on Duato's algorithm, in k-ary n-cubes is presented. This model along with the one for the DCSH proposed in Chapter 2 are used to compare the performance of the two networks, taking into account router delays and implementation costs for various technologies.

4.2 Analysis

Two and three-dimensional k-ary n-cubes have been the most popular multicomputer networks. They have been used in iWARP [120], J-Machine [110], Cray T3D [74], and Cray T3E [29]. These low-versions of k-ary n-cubes are widely known as tori. The present discussion focuses on the two-dimensional torus only, but the general conclusions reached in this chapter are equally applicable to three-dimensional case.

The torus has always been preferred over meshes, a variation without wrap-around connections, due to its symmetrical topology. Moreover, a study in [119] has revealed that using adaptive routing in the mesh can result in a significant performance degradation, because of the load unbalance on the network channels; the middle of the network in the mesh receives more traffic than the edges. Therefore, in this study and in the following chapters, the torus will be considered as the representative network of k-ary n-cubes.

4.2.1 Outline of the Model

This section proposes an analytical model for the two-dimensional torus. The validity of the model is achieved by comparing analytical results with those obtained through simulation.

The two-dimensional torus of $N(=k^2)$ nodes has k nodes arranged along each dimension. Each node is connected to its nearest neighbors in each dimension; a node at position (i, j), with $(0 \le i, j < k)$, is connected to nodes $(i \pm 1, j)$ and $(i, j \pm 1)$ modulo k. Furthermore, the torus can be implemented with uni-directional or bi-directional channels. This study, however, focuses on the bi-directional case because message distance is greatly reduced, compared to the uni-directional case.

Figure 4.1 shows the router structure of the torus. It has $(2n+1)$ inputs and $(2n+1)$ output physical channels. A node is connected to its neighboring nodes through $2n$ inputs and $2n$ output channels. The remaining channels are used by the local PE to inject/eject messages to/from the network respectively. Assuming that each physical channel is split into V virtual channels, the router contains one-flit buffer for each incoming virtual channel. The input and outputs of the router are connected by a crossbar switch, which is capable of

simultaneously connecting multiple input to multiple output channels given there is no contention over the output channels.

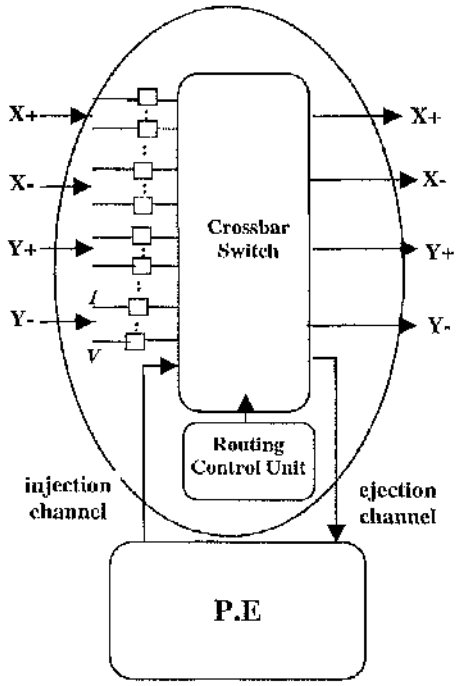


Figure 4.1: Router structure in 2D-torus.

Developing a mathematical model for high radix k-ary n-cube under adaptive routing strategy is more complicated than in networks with one channel per dimension, such as the DCSH and hypercube. Recall that the computation of the mean blocking delay at a given channel is determined as a function of the number of dimensions left to reach its destination. The difficulty here stems from the fact that when a message reaches a given

router; it is difficult to know how many hops are left to cross along a given dimension. The hops already made by a given message can belong to the same dimension or to different dimensions. The problem is further exacerbated as the number of dimensions increases. This problem does not exist for the DCSH or the hypercube. In these networks, a message crosses at most one channel along each dimension and therefore, it is easy to work out how many dimensions are left for a message to reach its destination, based on the number of hops already achieved.

The model proposed here follows the same approach as in chapter 2. A detailed study on the calculation of the probability of message blocking at intermediate routers is presented. The model is based on the same assumptions (*a* to *f*) as those of chapter 2. In addition, the following assumptions are made.

g) V $(V > 2)$ virtual channels are used per physical channel. In the torus, wrap-around channels form cycles within a dimension and therefore, at least two virtual channels are necessary to ensure deadlock-freedom within a dimension when these are crossed deterministically. If these channels are called *high* and *low*, a message is routed on the high channel if it is at a node which address is less than the destination address. Otherwise, it is routed on the low channels [32]. The $(V - 2)$ other virtual channels are crossed adaptively.

h) Physical channels are bit-transmission pipelined to reduce the effects of long wires, as suggested in [126]. This assumption is mainly used in this and following chapters when comparing the performance of the DCSH and torus.

The average number of channels that a message visits within a dimension and across the network is given by [2]

$$k_{avg} = \frac{k}{4} \tag{4.1}$$

$$d_{avg} = nk_{avg} \tag{4.2}$$

Since a router in the torus has $2n$ outgoing channels, the message rate received by each channel is

$$m_c = \frac{m_g d_{avg}}{2n} \tag{4.3}$$

The mean network latency, excluding multiplexing effects of virtual channels and the mean waiting time at the source, is given by

$$L = d_{avg}(D_t + 1) + L_m + \sum_{h=1}^{d_{avg}} B_h^C + W_{qd} \tag{4.4}$$

B_h^C is the mean blocking delay when the message is h $(1 \le h \le d_{avg})$ hops away from its destination, and W_{qd} is the mean waiting time at the destination. The latter is calculated using Equation (2.11). While B_h^C is given as

$$B_h^C = P_h^C L \tag{4.5}$$

where P_h^C is the probability of blocking when the message is h $(1 \le h \le d_{avg})$ hops away from its destination. This probability is a function of the number of virtual channels available for routing the message, and the number of dimensions that the message has still to cross. Hence, to calculate P_h^C, we need first to find how many dimensions remain for the message to reach its destination.

The number of alternative routes that a message can select, at its next hop, to advance

towards its destination depends on the number of hops already made in both dimensions. When a message is h $(1 \le h \le d_{avg})$ hops far from its destination, this means that it has already made $(d_{avg} - h)$ hops. These hops can be a combination of (l_1, l_2) hops, with l_1 and l_2 being the number of hops achieved in the first and second dimensions respectively, so that $(l_1 + l_2 = d_{avg} - h)$ and $(0 \le l_1, l_2 \le k_{avg})$.

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

a) When $(0 \le d_{avg} - h < k_{avg})$, the number of (l_1, l_2) combinations is $(d_{avg} - h + 1)$. In this case, a message still has to cross channels in both dimensions and therefore, can choose among adaptive virtual channels of both dimensions.

b) When $(k_{avg} \le d_{avg} - h < d_{avg})$, the number of (l_1, l_2) combinations is $(h+1)$. In only two cases, $(k_{avg}, d_{avg} - h - k_{avg})$ and $(d_{avg} - h - k_{avg}, k_{avg})$ out of these combinations, a message has crossed all channels of one dimension and thus, for the remaining hops, the message only crosses channels of the other dimension.

So, the probability that there remains only one dimension to cross by a message h hops away from its destination, P_{c_h}, can be written as

$$P_{c_h} = \frac{2}{h+1} \tag{4.6}$$

Hence, the probability that a message in its next hop can choose any adaptive virtual channel of the two dimensions is $(1 - P_{c_h})$.

Let P_a be the probability that all adaptive virtual channels at a dimension are busy, and

P_d be the probability that all virtual channels (including adaptive and deterministic) at a dimension are busy. As pointed in chapter 2, these two probabilities (P_a and P_d) are needed in the calculation of P_h^C. To compute P_a, three cases are considered.

a) V virtual channels are busy. This implies that all adaptive virtual channels are busy.

b) $(V-1)$ virtual channels are busy. The number of combinations where $(V-1)$ out of V virtual channels are busy is C_{V-1}^V. Only two combinations out of C_{V-1}^V result in all adaptive virtual channels being busy.

c) $(V-2)$ virtual channels are busy. The number of combinations where $(V-2)$ out of V virtual channels are busy is C_{V-2}^V. Only one combination out of these results in all adaptive virtual channels being busy.

Similarly, to obtain the second probability, P_d, two cases are considered.

a) V virtual channels are busy. This means that all adaptive and the required deterministic virtual channels are busy.

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

Keeping the same definition of P_v^C as in the previous chapters i.e., the probability that v virtual channels are busy, and taking into account the different cases mentioned above, P_a and P_d are found to be, respectively

$$P_a = P_V^C + \frac{2P_{V-1}^C}{C_{V-1}^V} + \frac{P_{V-2}^C}{C_{V-2}^V} \tag{4.7}$$

$$P_d = P_V^C + \frac{2P_{V-1}^C}{C_{V-1}^V} \tag{4.8}$$

Using Equations (4.6), (4.7) and (4.8), yield the probability of blocking P_h^C as

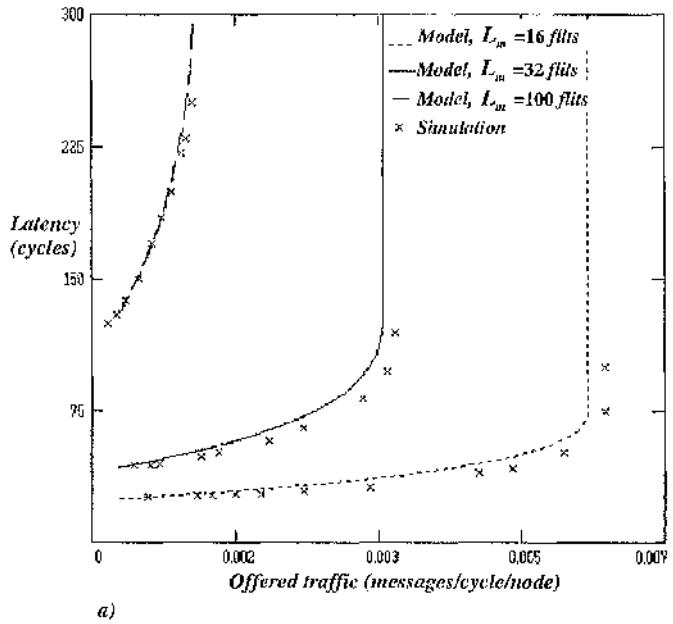
$$P_h^C = \begin{cases} P_a P_d & 0 \le d_{avg} - h < k_{avg} \\ (1 - P_{c_h}) P_a P_d + P_{c_h} P_d & k_{avg} \le d_{avg} - h < d_{avg} \end{cases} \tag{4.9}$$

Finally, taking into account the mean waiting time at the source and multiplexing effects of virtual channels, which are calculated in a similar way as in chapter 2, the overall message latency in the network can be written as

$$Latency = [W_{qs} + L]V_{avg} \tag{4.10}$$

4.2.2 Model Validation

Figures 4.2 and 4.3 show message latency predicted by the model against experimental results provided by simulation experiments with the following parameters. The network size was set to $N = 256$ nodes. The message length varied between 16, 32, and 100 flits. The number of virtual channels per physical channel V has been set to 3 then 5, in both cases two virtual channels are crossed in a deterministic way, and the remaining virtual channels are traversed adaptively. The router delay has also been varied between 0 and 2 cycles. The figures show that the above analytical model predicts message latency reasonably accurately under light and moderate traffic loads, with a slight overestimation as the traffic becomes heavier.



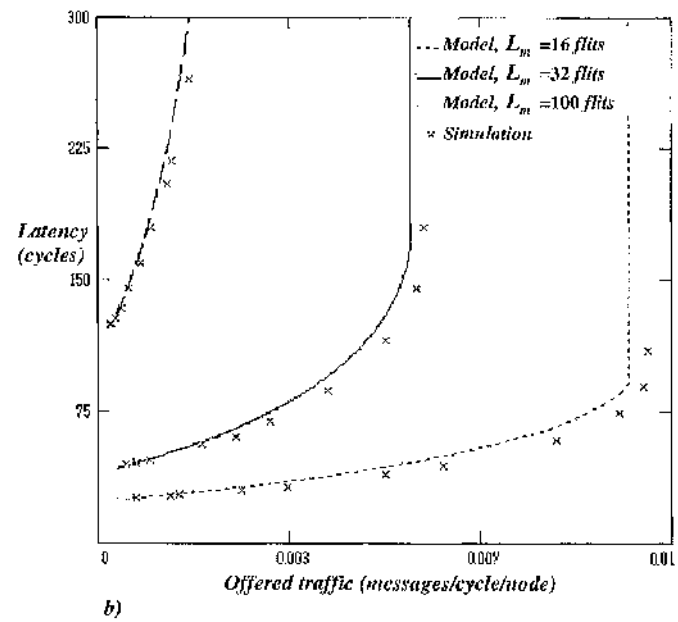
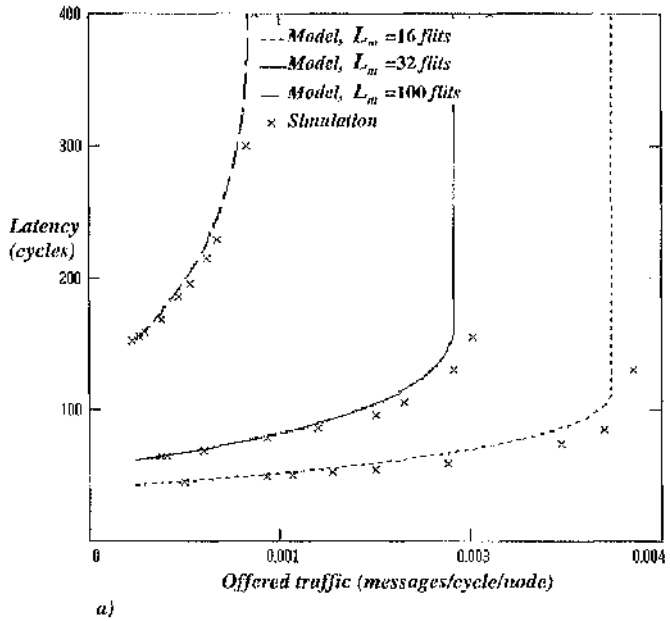
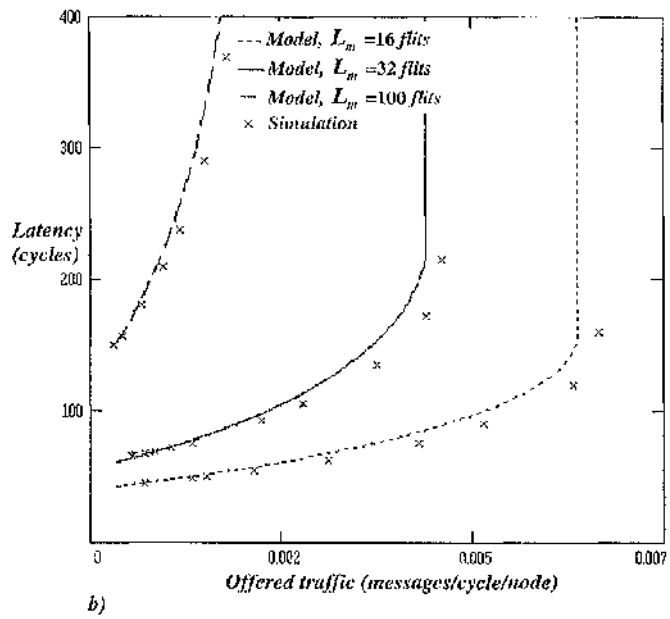


Figure 4.2: Latency predicted by the model against simulation results, $D_t = 0$. a) V=3, b) V=5.



a)



b)

Figure 4.3: Latency predicted by the model against simulation results, $D_t = 2$. a) V=3, b) V=5.

4.3 Comparison Results

If the channel width, in practice, is w bits, a message length of M bits is broken up into $L_m = M / w$ *phits* (or physical transfer unit), each of which contains w bits; L_m is referred to as the *message aspect ratio*. In practice, a flit in wormhole routing may be composed of one or more phits, but in the results presented below we assume that a flit is equal to one phit, since the general conclusions do not change for the other cases.

This section compares the performance of the two-dimensional DCSH to its torus counterpart with a fixed network size $N(=k^n)$ and equal implementation costs in VLSI and multiple-chip technologies. In a pure VLSI implementation, since wiring density is constrained, the *bisection width* i.e., the number of wires that cross the middle of the network is used to measure network implementation cost. Assuming that a network is implemented on the two-dimensional physical space with $\sqrt{N}$ nodes along a given dimension [38], the bisection width (B) of the DCSH and the torus, with a channel width of w_{DCSH} and w_{Torus} respectively, are found to be

$$B_{DCSH} = Nw_{DCSH} \tag{4.11}$$

$$B_{Torus} = \frac{4N}{k} w_{Torus} \tag{4.12}$$

Holding the bisection width constant, the channel width of the torus in terms of that of the DCSH is given by

$$w_{Torus} = \left\lceil \frac{k}{4} \right\rceil w_{DCSH} \tag{4.13}$$

When the network is laid out over multiple chips, each node on a single chip, the node

pin-out is a suitable cost metric. The node pin-out (P) of the DCSH and torus are given by

$$P_{DCSH} = nkw_{DCSH} \tag{4.14}$$

$$P_{Torus} = 4nw_{Torus} \tag{4.15}$$

Assuming constant node pin-out, the same channel width relationship, given by Equation (4.13), is obtained. Therefore, under both constant bisection width and node pin-out constraints, the torus has $k/4$ wider channels than the DCSH.

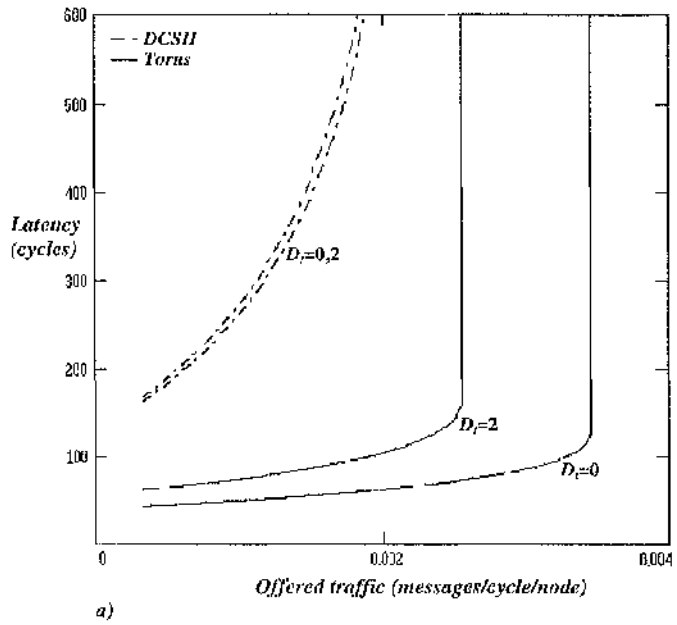
For illustration, let the channel width in the DCSH be one bit (the channel width in the torus is normalised to that of the DCSH). Two network sizes are used in this analysis: N=256 and N=1024 nodes, representing medium and moderately large systems, respectively. Two message lengths are considered: long messages of 128 (DCSH) phits and short messages of 32 (DCSH) phits, typical figures that have been widely considered by several authors [1, 2, 19, 21, 77, 119]. In this and what follows, all message lengths are quoted in terms of (DCSH) phits. Moreover, since a flit is assumed to be equal to one phit, the words flit and phit are interchangeably used.

The number of virtual channels per physical channel has been fixed to the minimal number required in each network to ensure deadlock-free routing. Two virtual channels are used in the DCSH, one crossed deterministically and the other one adaptively. Three virtual channels are used in the torus, two are crossed deterministically, and the remaining one crossed adaptively. The router delay has been set to 0 then to 2 cycles. Although zero-cycle router delay is unrealistic, this case has been included to assess its impact on performance. A figure of 2 cycles for the router delay has been chosen because

it has been suggested by other authors [92, 103], and reflects the increase in router delay due to the use of adaptive routing and virtual channels.

Figure 4.4 compares latency results in the DCSH and torus for a network size of 256 nodes. For equal implementation costs, channels in the torus are 4 times wider than those in the DCSH. Figure 4.4-(a), which shows the results for long message lengths of 128 flits, reveals that under the extreme (and unrealistic) situation when the router delay is ignored, the torus outperforms the DCSH under all traffic conditions. However, when router delay is taken into consideration, the torus still gives better performance characteristics than the DCSH. This is due, firstly, to the wider channels in the torus which reduce the message aspect ratio, a component of the message latency. Secondly, the number of paths offered to messages in the torus is greater than that in the DCSH, and hence message blocking delays are further reduced in the torus. (It is worth noting that comparing these results to those found in [114], where the two networks have been compared in the context of deterministic routing, it can be said that the torus greatly improves its performance with adaptive routing at long messages.)

Figure 4.4-(b) reveals that when the message length is decreased to 32 flits and the router delay is taken into consideration, the torus manages to take advantage of its wider channels under light traffic only. The DCSH provides lower message latency than the torus as the traffic increases. This is because short messages result in even shorter message aspect ratios, making message latency in the torus more sensitive to the distance and the effects of higher router delays. The increase in router delay along with short messages make the distance component dominates latency. Moreover, both Figures 4.4-(a) and 4.4-(b) show that the DCSH is almost insensitive to the effects of router delays because



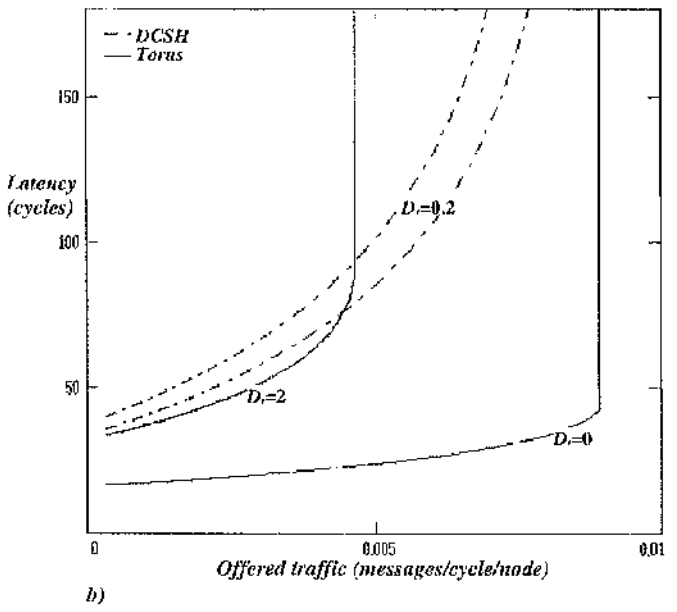


Figure 4.4: Latency in the DCSH and torus, N=256.

a) L_m = 128 flits, b) L_m = 32 flits.

messages traverse a fewer number of routers during their network journey. The torus, on the other hand, is very sensitive to the effects of router delays because messages cross, on average, a larger number of routers, and therefore require a longer service time to reach their destination.

Figure 4.5 shows performance results in the two networks as their size scales to 1024 nodes, and keeping the message length to 128 flits. As the network size increases, channels in the torus become much wider than those in the DCSH, and therefore the message aspect ratio in the torus becomes much smaller. However, the average message

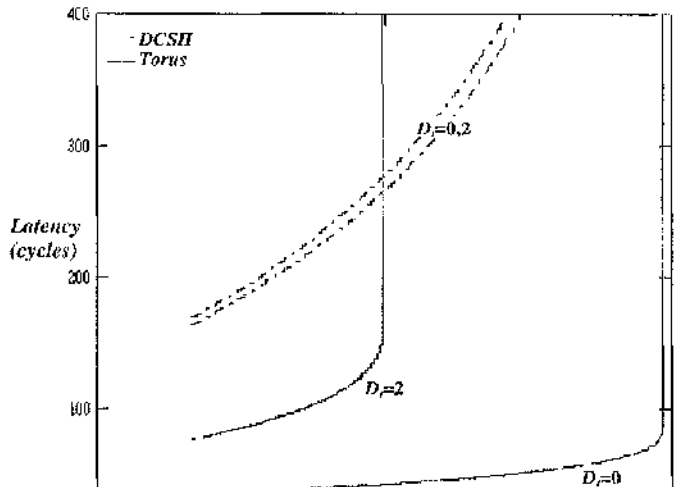


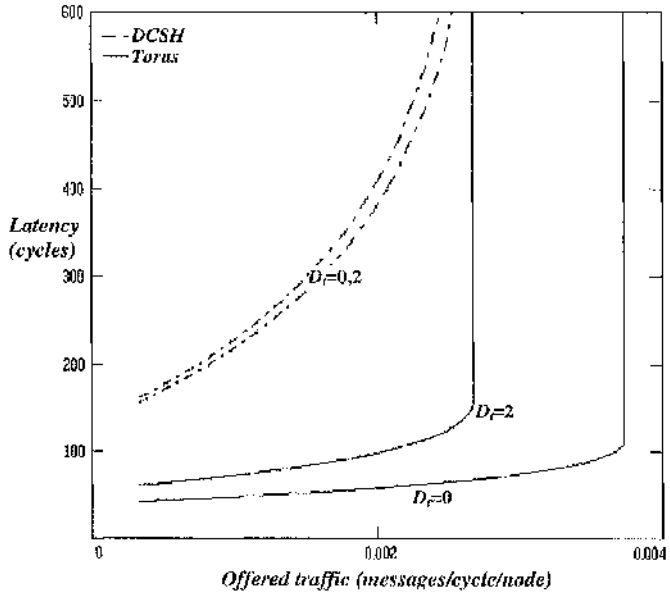
Figure 4.5: Latency in the DCSH and torus. N=1024 and L_m = 128 flits.

distance in the torus increases. So once again, when router delay is considered, the torus outperforms the DCSH at light traffic intensity only. At moderate and high traffic loads, the DCSH achieves lower message latency than the torus. As mentioned before, messages in the latter must cross larger distances, experiencing higher blocking delays under moderate and higher traffic loads.

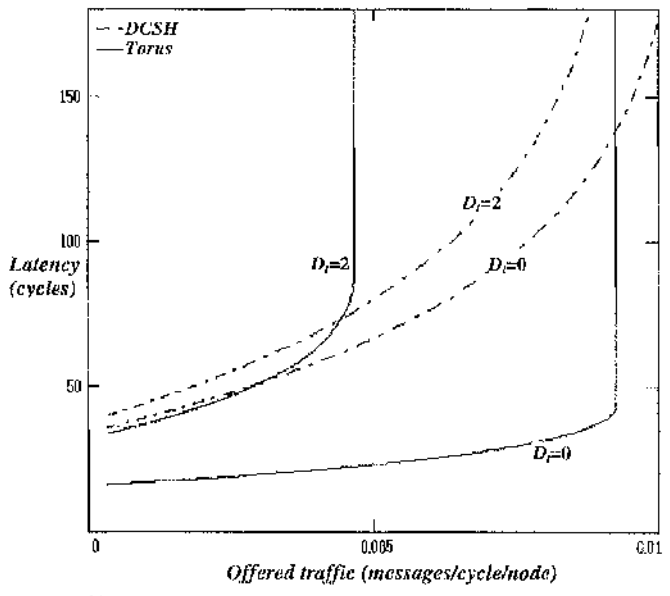
The above results have been obtained assuming that there is a unique ejection channel servicing messages, when arriving at their destination. The use of multiple ejection channels[4] has been proposed in [9] to improve the performance of networks; multiple ejection channels decrease message blocking delays due to message contention at their destination node, and as a result the effects of chained blocking across network channels caused by wormhole routing are reduced. The iWARP [120] also is an example of a practical multicomputer that uses multiple ejection channels to connect the router to its local PE. In the remainder of this section, the impact of router/PE interface on the performance of the DCSH and torus is also evaluated. With multiple ejection channels, messages are immediately consumed when they arrive at their destination.

For illustrative purpose, Figure 4.6 shows the performance of the two networks for a system size of 256 nodes. The performance improvement gained from using multiple ejection channels is more noticeable in the DCSH (by comparing Figures 4.6 and 4.4). This is because messages in the DCSH make at most two hops to cross the network. As a result, message headers arrive rapidly at their destination while their tails are still at the source. With one ejection channel, blocked messages are spread along their path, preventing messages behind them to use the channels which they occupy, increasing their

[4] Also multiple ejection channels have been proposed in the context of broadcast communication to avoid message deadlock [16].



a)



b)

Figure 4.6: Latency in the DCSH and torus with multiple ejection channels and N=256. a) $L_m = 128$ flits, b) $L_m = 32$ flits.

waiting delays at the source node. Multiple ejection channels reduce message blocking delays at the source and channels in the DCSH. In the torus, however, the results show that messages suffer more delays when traversing the network than at the consumption phase at the destination node because they have to cross a larger number of channels.

4.4 Conclusion

More recent multicomputers have dropped deterministic routing and opted for adaptive routing to improve performance. They still use, however, *k*-ary *n*-cubes, especially the torus. This chapter has examined the relative performance merits of the DCSH and torus when adaptive routing is used. To this end, an analytical model of adaptive routing for the torus has been presented, and validated through simulation. The comparative analysis has taken into account channel bandwidth constraints imposed by implementation technology, such as VLSI and multiple-chip technology.

The results have revealed that when router delay is taken into account, the torus can take advantage of its wider channels to provide a lower latency than the DCSH for long messages and small systems only. The DCSH, however, exhibits superior performance for short messages and large system sizes. This last finding reveals that the DCSH may well prove suitable for future fine-grain and distributed shared-memory multicomputers.

The performance degradation of the torus increases proportionally with the system size. Furthermore, the torus has shown its extreme sensitivity to the increase in router delays. Even though the wider channels in the torus reduce the message aspect ratio, messages still have to cross a larger number of intermediate routers, and consequently experience higher blocking delays under heavy traffic loads.

Chapter 5

Performance of the DCSH and *k*-ary *n*-cubes in the Presence of Broadcast Traffic

5.1 Introduction

This chapter examines the ability of the DCSH and *k*-ary *n*-cubes in handling broadcast communication. As in the previous analysis conducted in chapter 4, router delays and implementation costs for various technologies are taken into consideration in this study.

Broadcast communication, which refers to the delivery of the same message originated from a source to all other system nodes, is an important operation required in many parallel applications [31, 51, 85, 99, 143]. For instance, broadcast communication is often needed in scientific computations to distribute large data arrays over system nodes, and to

perform data manipulation operations, such as replication [51, 99]. Furthermore, it is required in control operations such as global synchronisation [143] and to signal changes in network conditions, e.g., faults. In the distributed shared-memory paradigm, broadcast communication is used to support shared data invalidation and updating procedures required for cache coherence protocols [31, 85].

The present analysis evaluates how the performance of the DCSH and k-ary n-cubes is affected by the presence of broadcast traffic. In other words, it assesses how much broadcast traffic can influence the performance of background traffic (composed of unicast messages). Due to the difficulty of developing analytical models for broadcast communication in wormhole-routed networks, this study is carried out using simulation models. The remainder of the chapter is organised as follows. Section 5.2 presents the different approaches proposed to implement broadcast communication. Section 5.3 describes the simulation model used in our comparative analysis. Section 5.4 compares the performance of the two networks. Finally section 5.5 concludes this study.

5.2 Broadcast Implementation Schemes

This section gives an overview of the different approaches suggested for implementing broadcast operations. The main problem with broadcast messages is they consume a large amount of network bandwidth. Furthermore, they tend to replicate messages, resulting in increased traffic in the network. This has motivated researchers to look for efficient ways to implement such operations in multicomputers [100, 107, 116]. Most existing multicomputers support only unicast communication in hardware, apart from few systems such as the nCUBE-2 [105], for instance, in which broadcast communication is

implemented in hardware. One straightforward way to implement broadcast operations is in software. A broadcast message is replicated into several unicast messages, one for each destination. This is often known as the *multiple unicast-based* (or *separate addressing*) approach. Examples of systems which adopt this scheme are the Intel Paragon [118], MIT Alewife [3] and Stanford FLASH [82]. This implementation approach has several advantages. It is simple, and does not require special hardware to process messages. Moreover, routing algorithms designed for unicast communication can still be used for routing broadcast messages. However, its main disadvantages are firstly, the increase in the *startup* latency[5]; the preparation time of messages before entering the network. Secondly, the waste of network bandwidth due to additional traffic caused by excessive message replication.

To overcome the limitations of the software implementation, hardware implementation for broadcast communication has been suggested. Two implementation schemes have been proposed in the literature, namely *tree-based* [96-98] and *path-based* models [16, 23, 87, 117]. These two schemes require changes in the router circuitry [43] to perform message replications in which case a flit at a given router input channel can be forwarded to more than one output channel.

In the tree-based approach, the idea consists of finding a spanning tree with the source node taken as the root. In the first step, the source node sends a copy of the message to a subset of destinations, which are its direct neighbours. Each node holding a copy of that message forwards it to another subset of destinations, which have not yet received a copy.

[5] We have so far considered network latency only. Communication latency includes both network latency and startup latency. The latter is related to the time taken in the message preparation phase, and does not depend on the network topology. In general, this latency is high due to the high software overhead [43, 102].

The process continues until all nodes of the system have a copy of the message. The nCUBE-2, for instance, applies this approach.

The path-based approach attempts to find one or a few paths that can be shared by a set of destinations. As a broadcast message propagates along its path, a copy is delivered to each node constituting that path. Various algorithms have been designed based on that concept; some use a unique path like the Hamilton path [86, 87], and others try to reduce the path length by finding a compromise solution between the multiple unicast-based and path-based schemes. Instead of sending a unique message covering the whole system nodes, destination nodes are divided into subsets and a few separate copies of the broadcast message are created at the source node, each of which traverses a path covering a given subset of nodes [16, 117].

Path-based routing algorithms have been suggested as an alternative to the tree-based algorithms for wormhole switching to reduce blocking in the network; if a branch of the tree is blocked because of the unavailable space at a receiver node, the whole tree is blocked, increasing network contention. However, path-based algorithms suffer from several inefficiencies [96], affecting mainly the startup latency and message distance. When multiple paths are used, several copies of the broadcast message have to be generated. When a unique path is used to cover all system nodes, the size of the message header increases with the system size, and the message has to cross a large distance. Moreover, node addresses must be ordered according to the order in which the message visits the nodes.

5.3 Simulation Model

The two-dimensional torus with bi-directional channels is considered in this study, as a representative network of k-ary n-cubes. To evaluate the performance of the DCSH and torus, flit-level simulators, based on a discrete event model, have been developed.

Nodes in the two networks can generate a mixture of traffic composed of unicast and broadcast messages, and the two types of messages compete for network resources. To reflect such a scenario, the simulators have been built atop those already developed for validation purposes in chapter 2 and chapter 4 for the DCSH and torus, respectively. Hence, Duato's algorithm is still used to route unicast messages towards their destination.

Broadcast messages are routed using the tree-based approach, requiring only one communication startup for each broadcast message, and reducing the number of broadcasting steps, compared to path-based routing algorithms. Also, due to the high simulation running time, we have considered broadcast messages of short length only. However, this still covers most of situations listed above. In addition, since messages are short, channels can be equipped with a few flit buffers to buffer broadcast messages when they are blocked, and therefore the blocking from which the tree-based approach suffers is alleviated. When a broadcast message arrives at a router on dimension i $(0 \le i < n)$, copies of each flit are forwarded to the local PE for consumption and to all the outgoing channels (or dimensions) j $(i < j < n)$. In the case of the torus, a message might continue using the same dimension on which it arrives to the router. Furthermore, a message at an input of a router gets blocked if at least one of the outgoing channels required is unavailable.

For unicast communication, the assumptions described in chapter 2 are used, and are briefly recalled here. Messages are generated at each node following a Poisson distribution for inter-arrival time of a mean rate m_g messages/cycle, message length is exponentially distributed with a mean of L_m flits, and the traffic is uniformly distributed over system nodes. Broadcast communication, however, is based on the following assumptions.

a') Nodes generate broadcast messages independently of each other, and which follow a Poisson distribution with a mean rate of m_b messages/cycle.

b') Broadcast message length is exponentially distributed with a mean of L_b flits. Also, a flit requires one transmission cycle time across a physical channel.

c') As mentioned above, broadcast messages hold important information, such as global notification of network errors or memory update and invalidation procedures in shared-memory systems, and therefore they need an efficient network. Often a separate network is dedicated to support broadcast communication, either virtual [132] or physical [15, 67, 83], of course, with extra implementation cost. This analysis assumes that a separate virtual network is used for broadcast communication, sharing the physical network bandwidth with the other virtual networks used by unicast traffic in a time-multiplexed manner. So, each physical channel is split into V virtual channels. In the DCSH, V is set to three, where two virtual channels are used by unicast messages, and the third is used by broadcast messages. Similarly, the torus uses four virtual channels among which three are used for routing unicast messages.

d') Broadcast messages are given higher priority than unicast messages when crossing the network, because they often rely on low communication latency for

good performance.

e') Each processor is connected to the network through two separate injection channels for unicast and broadcast messages respectively. Providing two injection channels alleviates the starvation problem for unicast messages since higher priority is given to broadcast messages. Similarly, two ejection channels at the processor/router interface of each node are used for removing messages from the network.

Throughout the analysis, the DCSH and torus performance investigation is achieved on small and medium system sizes of 64 and 256 nodes respectively. The mean message length for unicast operations is fixed to short of 32 flits. Broadcast message lengths considered in this analysis are 2 and 8 flits. These figures for the system size and message length have been chosen because we have found that simulation requires very excessive running times when the message length and system size are increased. The router delay has been set to two cycles to process on unicast and broadcast message headers. Although the time taken by a router to process on broadcast messages should be higher because of the time to establish the connection between an input and the set of the required outputs, this has not a great influence on the results since broadcast messages are buffered when they get blocked.

5.4 Performance Results

The purpose of this study is to see how the background traffic, composed of unicast operations, is affected by broadcast communication in both the DCSH and torus. To this end, several simulation runs have been conducted, where the metric of interest is the

mean latency of unicast messages. During these experiments, unicast messages are injected into the network at a constant rate (this is chosen in a manner that keeps the traffic load in the torus light to moderate in the absence of broadcast communication), while varying the rate at which broadcast messages are injected into the network. This gives us an insight into how each network can respond to different loads of broadcast traffic.

Figure 5.1 shows the mean message latency of the background traffic as a function of broadcast traffic rate m_b in the DCSH and torus, both of 64 node size. Background traffic rate injected in the two networks is fixed at $m_g = 0.0083$. Broadcast messages have a mean length of 8 flits. The results reveal that with very low broadcast traffic rates, the torus provides lower latency for unicast messages than the DCSH. This is because under such conditions broadcast traffic is almost absent. More importantly, the torus has smaller message aspect ratio due to its wide channels, as mentioned in the previous chapter, and therefore the background traffic rate chosen here represents a moderate load in the DCSH. However, a small increase in broadcast traffic sharply degrades the performance of the torus, yielding a quick saturation. The DCSH responds better to the increase in broadcast traffic loads. For instance here, with more than 40% additional broadcast traffic of that at which the torus saturates, the DCSH still provides lower message latency. The DCSH benefits from its hypergraph links, used to interconnect cluster nodes. These links enable it to send a broadcast message to all cluster nodes in one step and therefore, it builds broadcast trees in 2 steps. The torus, however, slowly builds broadcast trees in several steps. Therefore, broadcast messages consume more network bandwidth by crossing several channels. It has been shown in chapter 4 that unicast messages, notably when they are short, suffer from higher blocking delays encountered due to the distance

they cross. Even though unicast traffic load is low, and consequently contention between unicast messages on acquiring channels is low also, unicast messages experience blocking delays caused by broadcast messages. Since the latter has higher priority to use channel bandwidth, this results in higher blocking delay and higher latency for unicast messages.

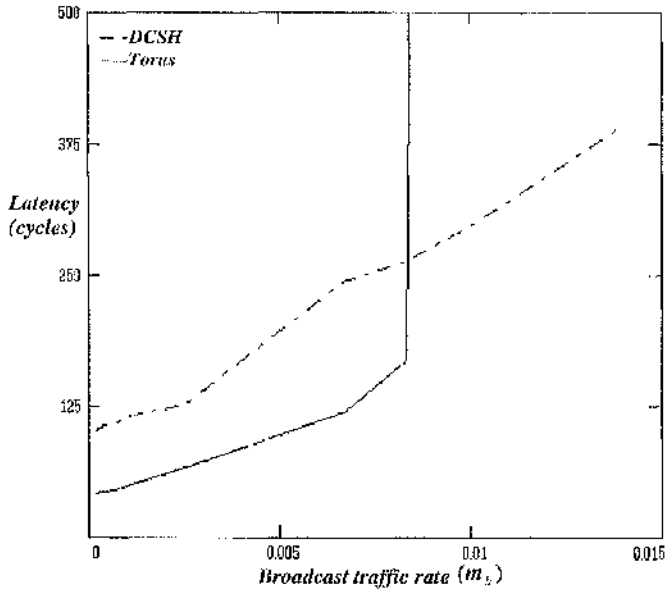


Figure 5.1: Latency of background traffic versus broadcast traffic in the DCSH and torus. $N = 64$ nodes and $L_b = 8$ flits.

Figure 5.2 shows the performance results of the DCSH and torus when their system size scales to 256 nodes. The background traffic rate injected into the two networks is set to

$m_g = 0.0083$ messages/cycle, and broadcast message length is held to 8 flits. Again, the torus offers better background traffic performance only when low broadcast traffic rates are injected in the two networks. The torus cannot support the increase in broadcast loads. Moreover, the DCSH performs even much better than the torus, under moderate and higher loads of broadcast traffic when system size is scaled up, by comparing the results with those of Figure 5.1. The increase in system size means an increase in message distance in the torus and accordingly, higher blocking delays for unicast messages. As can be seen in the figure, the DCSH saturates at around 70% additional broadcast traffic of that at which the torus is saturated.

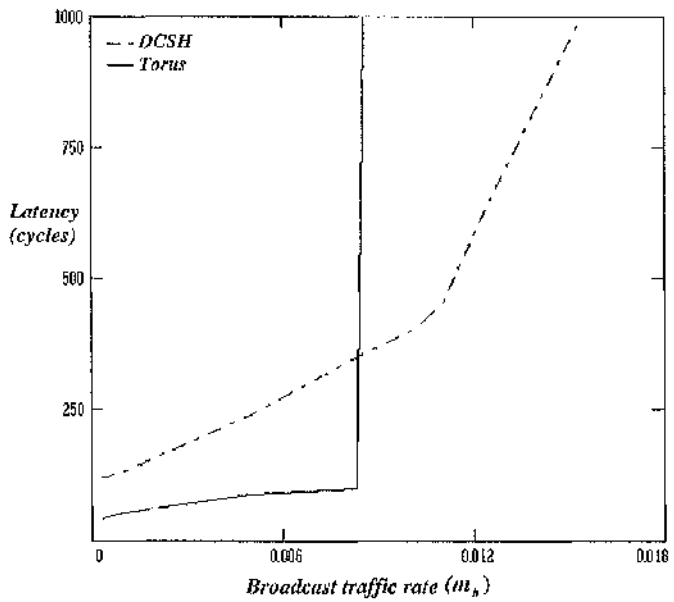


Figure 5.2: Latency of background traffic versus broadcast traffic in the DCSH and torus. $N = 256$ nodes and $L_b = 8$ flits.

Figures 5.3 shows the performance of the two networks when the length of broadcast messages is reduced to 2 flits. This is a reasonable length for broadcast messages in some situations such as the case of invalidation operations in the cache coherence procedure. One data flit is found to be a typical data size for invalidation messages [96]. The unicast messages are generated at a fixed rate $m_g = 0.01$ messages/cycle. It can be seen from the figure that similar trends are also observed as in the previous figures; unicast messages still exhibit higher latency in the torus when broadcast traffic is increased.

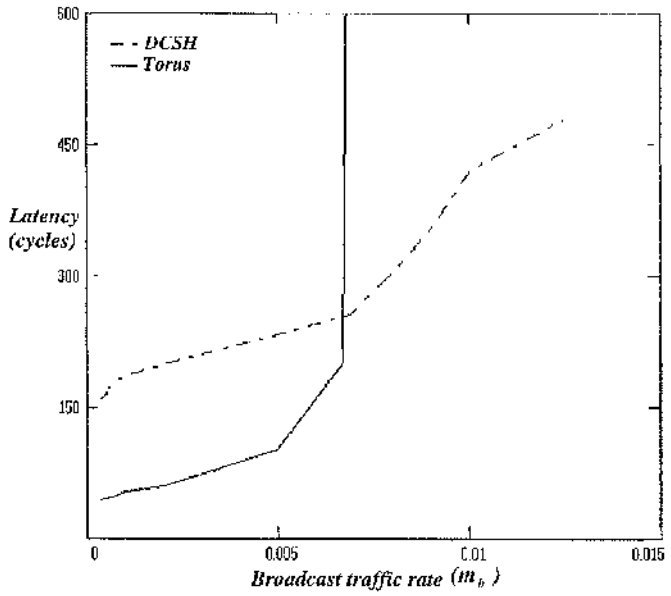


Figure 5.3: Latency of background traffic versus broadcast traffic in the DCSH and torus. $N = 256$ nodes and $L_b = 2$ flits.

So far, the above results have been obtained assuming that all system nodes are broadcasting. We have also investigated the behavior of the background traffic in the two networks when only a subset of system nodes perform broadcast. In the simulation, a uniform distribution, using a random number generator, is used to choose the set of the source nodes, which perform broadcast. For illustrative purpose, Figure 5.4 shows the performance results with only 12% of the system nodes generate broadcast traffic in a system size of 256 nodes. As in the previous experiments, broadcast message length is fixed at 8 flits, and the background traffic rate injected by each system node is held constant at 0.011 messages/cycle. The torus shows its sensitivity to the increase in broadcast traffic loads, while the DCSH conserves its stability at higher loads. The DCSH

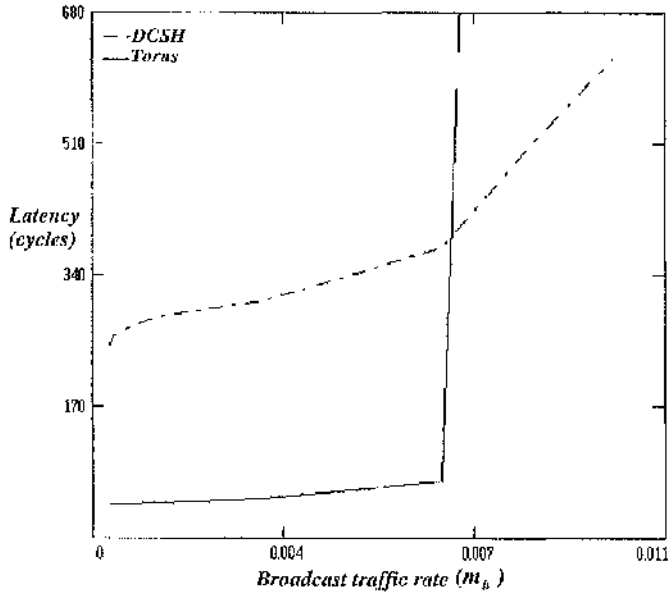


Figure 5.4: Latency of background traffic versus broadcast traffic.
$N = 256$ nodes, $L_b = 8$ flits, and 12% broadcasting nodes.

manages to support as much as 40% additional traffic loads of that at which the torus is saturated. Comparing these results to those in Figure 5.2, it can be concluded that as the number of broadcasting nodes increases, the performance degradation of the torus also increases.

5.5 Conclusion

This chapter has compared the performance of the DCSH and torus in the presence of broadcast communication. The results presented here have shown that the DCSH performs better than the torus and sustains higher loads of broadcast traffic when the number of broadcasting nodes increases.

The background traffic in the torus is significantly affected by broadcast traffic, even when broadcast messages are very short. Both unicast and broadcast messages in the torus traverse a larger distance. This increases the interference probability of the two types of message in the network, increasing blocking delays of unicast messages in the torus. The DCSH with its hypergraph links can manage to build broadcast trees more rapidly than the torus. Unicast messages travel much shorter distance, and therefore encounter less blocking delays caused by broadcast traffic. The DCSH has shown its higher ability to support significant amounts of broadcast traffic with negligible degradation in the performance of background traffic.

Chapter 6

The "Express" Channel Concept in the DCSH and *k*-ary *n*-cubes

6.1 Introduction

Previous chapters have shown that *k*-ary *n*-cubes perform poorly, especially when message lengths are short because of their high diameter. Moreover, chapter 4 has shown that higher router delays have detrimental effects on performance in the presence of blocking due to the large number of routers that messages have to visit to reach their destination; the distance component of the message dominates latency in these networks.

These results are in conformity with earlier more simplistic studies, which motivated Dally [37] to propose an enhanced version of *k*-ary *n*-cubes, called *express k-ary n-cubes*

(or *express-cubes* for short), which provides additional express channels[6], allowing messages to partially bypass groups of nodes within a dimension.

Having compared the performance of the DCSH to that of k-ary n-cubes in chapter 4, this chapter carries on with the performance analysis of the DCSH and express-cubes. The latter can be considered as a halfway topology between k-ary n-cubes and the DCSH. Moreover, by examining the structure of the two networks, the express-cube shares the concept of "bypass" channels with the DCSH since with express channels, it allows messages to "partially" rather than totally bypass nodes within a dimension. The present analysis investigates, through mathematical models validated through simulation, whether the total bypassing inherent in the DCSH topology yields better performance over partial bypassing provided by the express-cube. It assumes equal implementation costs and takes into account router delays. It is important to note, however, that the present study is limited to deterministic routing due to the complexity of modeling adaptive routing in express-cubes.

6.2 Express-cubes

Express-cube is obtained by inserting periodically in a conventional k-ary n-cube an *interchange* after each group of k_n nodes, to provide a short path to non-local messages. Figure 6.1 shows an example of a two-dimensional express-cube (with $k = 6$). An interchange, which is a simple router, is connected to its neighbouring interchanges by one or more, say m, express channels. The routing function of the interchanges is limited to forwarding incoming messages on local or express channels depending upon their

[6] More recently, Potlapalli & Agrawal [122] have used the concept of "express channels" to improve the performance of hierarchical multi-stage interconnection networks.

destination. Only nodes have the capability to consume messages, if they are destination, or to decide to switch messages to another dimension. When a message is generated at a node belonging to a given group, it traverses local channels only if its destination is in that group. Otherwise, it crosses some local channels to reach the last node leading up to the level of interchanges. At that level, if the destination is in the neighbour group, the interchange forwards the message down through the local channel. Otherwise, the message keeps on traversing interchanges using express channels until it reaches the last interchange connected to the group of nodes containing the destination.

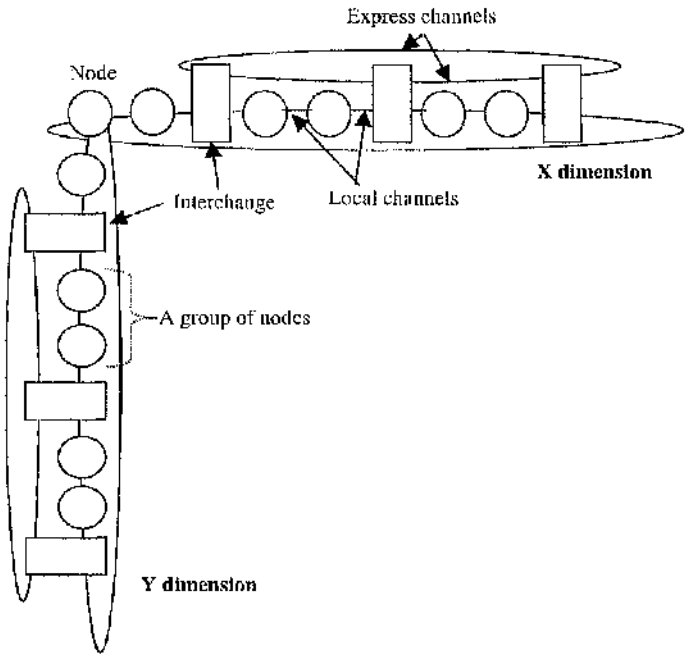


Figure 6.1: Express channels along X and Y dimensions in the express 6-ary 2-cube.

6.3 Analytical Models

The present analysis, as the previous ones, considers bi-directional express-cubes with $m\,(m \geq 1)$ express channels connecting each pair of interchanges. The structure of routers is identical to that in the torus, while the detailed structure of interchanges can be found in [37]. Identically to the previous chapters, the models developed here focus on the computation of the mean message latency, which includes message blocking delays at channels and the mean waiting time at the source and destination nodes. In addition to the assumptions (*a* to *f*) stated in chapter 2, the models assume that

g) Messages are routed in a deterministic way when traversing the network i.e., messages visit network dimensions in an increasing order. While in the DCSH, restricting messages to visit dimensions in an order is sufficient to avoid deadlock. In the express-cube, however, due to wrap-around channels, $(V =)2$ virtual channels per physical channel are needed to break deadlock within dimensions [32].

h) Each group in the express-cube is of k_n node size.

6.3.1 The Express-cube Model

The average distance crossed by messages in the express-cube is given by

$$d_{avg} = nk_{avg} \tag{6.1}$$

Where k_{avg} represents the average number of hops achieved by messages within a dimension. It is equal to the sum of the average number of local channels crossed before the message reaches the first interchange, k_{avg}^{B}, the average number of express channels,

k_{avg}^{E}, and the average number of local channels crossed after the last interchange, k_{avg}^{F}. Their expressions are given by [37]

$$k_{avg}^{B} = \frac{k_n + 1}{2} \tag{6.2}$$

$$k_{avg}^{E} = \left\lfloor \frac{h'}{k_n} - \frac{1}{2k_n} - \frac{1}{2} \right\rfloor \tag{6.3}$$

$$k_{avg}^{B} = 1 + \left(h' - \frac{k_n}{2} - \frac{1}{2} \right) mod\ k_n \tag{6.4}$$

Where $h' = k/4$ represents the average number of channels crossed by a message within a dimension in the absence of express channels.

Messages generated by a given node at a rate m_g can follow any of the 2^n possible paths in an n-dimensional express-cube if we consider the combination of directions (left or right) in all the n dimensions. So, the message rate on each path is $m' = m_g / 2^n$. Without loss of generality, the analysis focuses on one path, taking into account the interaction of messages traveling on different paths and sharing one or more dimensions. The results are similar for other paths due to the network symmetry. Recall that the total message rate arriving on dimension i from the previous dimensions and from the local PE has found in chapter 3 to be $\Phi = (1 - p_s)m'$, where p_s is the probability that a message skips a dimension.

The derivation of the mean message latency in the express-cube follows the same basic idea outlined in the diagram of Figure 3.1. However, to compute S_i, the network latency of a message entering the network through dimension i $(0 \le i < n)$, we need to expand each branch of that diagram as depicted in Figure 6.2. This figure shows the path taken

by a typical message within dimension i $(0 \le i < n)$. Let us also divide the traversal of that path into three stages. The first stage represents local channels before arriving at the first interchange, Ch_j^B $(1 \le j \le k_{avg}^B)$. The second one represents the set of express channels that a message crosses, Exp_j $(1 \le j \le k_{avg}^E)$, at a given interchange a message can choose any of the m express channels with equal probability. The third stage denotes the set of local channels leading the message to its destination, Ch_j^F $(1 \le j \le k_{avg}^F)$.

Recall that we are interested in determining latency seen by messages of the flow R_i^L. These messages compete at the first local channel with messages sharing the same path and having crossed lowest dimensions j $(0 \le j < i)$ R_i^P, messages which take opposite directions in the other dimensions and share the same direction of dimension i (their rate is equal to $(2^{n-1}-1)\Phi$), and messages that have already traversed some channels of dimension i and require the local channel to progress in that dimension, fl (this rate will be calculated later). The calculation of S_i is achieved in two steps. First the entering latency $T_{i,1}^B$ seen by the total message rate of the same path, Φ, at the first local channel is calculated. Then, the entering latency S_i seen by messages of the flow R_i^L is derived, taking into account the blocking delay caused by messages of the flow R_i^P.

The total message rate newly entering dimension i through a local channel is $2^{n-1}\Phi$. The continuing traffic on local and express channels within each dimension, Rc, is found to be [28]

$$Rc = 2^{n-1}\Phi k_{avg} \tag{6.5}$$

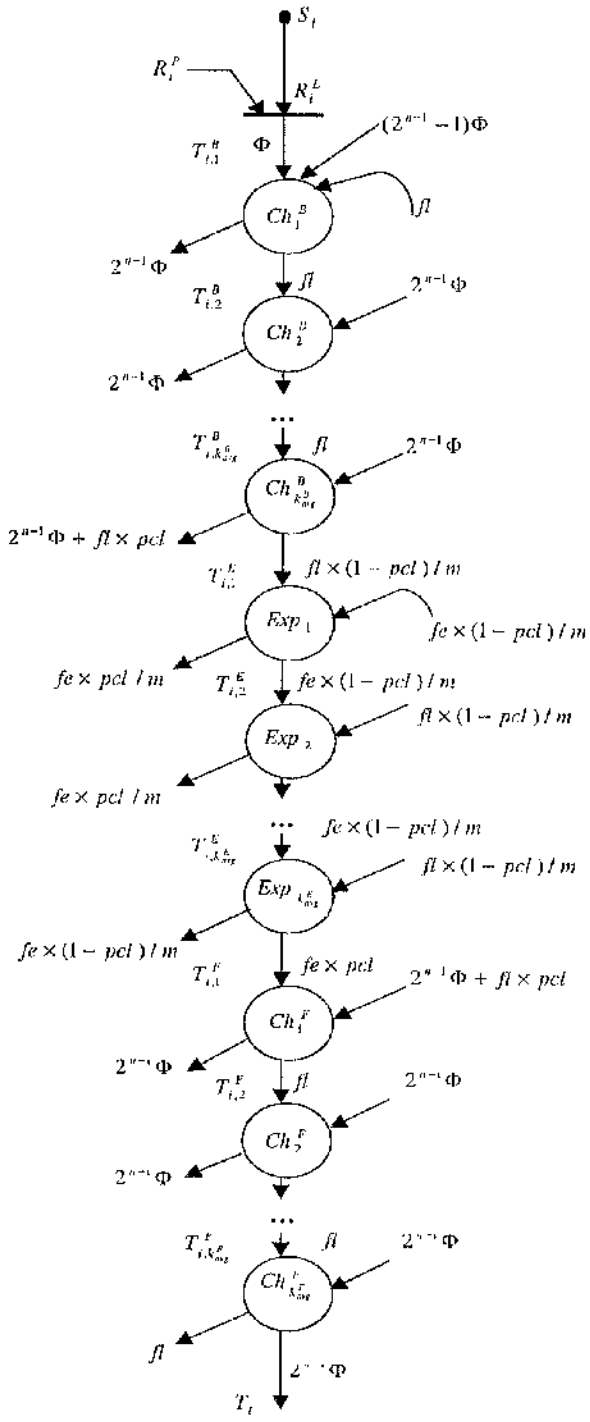


Figure 6.2: A message path within a dimension.

The above rate is divided into two fractions. A fraction, fl, concerns messages which travel on local channels, with the probability $pcl = 1/k_n$. The other fraction, fe, concerns messages traveling on express channels, with the probability $(1 - pcl)$. The rates fl and fe are given as follows

$$\begin{aligned} fl &= Rc \times pcl \\ fe &= Rc \times (1 - pcl) \end{aligned} \tag{6.6}$$

The total message rate entering the first local channel of the first stage is the sum of the message rate $2^{n-1}\Phi$ and the rate of messages continuing on that dimension, fl. Using the flow conservation law, the same rate exits the local channel; the rate fl continues travelling within the dimension, and therefore the rate $2^{n-1}\Phi$ exits the dimension. This is applicable for all local channels of the first stage except the last one. The latter is connected to an interchange, which forwards on its outgoing local channel the newly entering message rate $2^{n-1}\Phi$ with a fraction of the continuing traffic $fl \times pcl$. The other fraction of the continuing traffic $fl \times (1 - pcl)$ is sent on each of the m express channels with the probability $1/m$.

Each express channel in the path receives in addition to $fl(1 - pcl)/m$, coming from a local channel, a fraction $fe \times (1 - pcl)/m$, which represents the message rate arriving on an express channel and continues on that express channel. At the exit of any of the express channels except the last one, the fraction $(fe \times pcl)/m$ of the total traffic crossing that channel is forwarded by the interchange to its outgoing local channel, while an other fraction $fe \times (1 - pcl)/m$ continues traversing express channels. At the exit of the last express channel, the interchange forwards the message fraction $fe \times (1 - pcl)/m$ on one of its outgoing express channels, and the total fraction $fe \times pcl$ from all of its

incoming express channels on its outgoing local channel. This fraction joins the other fraction of traffic $2^{n-1}\Phi + fl \times pcl$ that the interchange forwards from its incoming local channel. Again, the total traffic rate received by each local channel of the last stage is $2^{n-1}\Phi + fl$, where the rate fl continues traversing channels of dimension i, and $2^{n-1}\Phi$ exits the dimension.

To calculate the mean message latency, *Latency*, we adopt a backward flow analysis, as outlined in chapter 3, starting from the destination and moving backward to the source. This analysis is based on the law that correlates latency seen by a message before entering a given dimension to the latency seen by the message at the exit of that dimension. The network latency, S_i $(0 \le i < n)$ is the latency seen by a message after exiting dimension i, T_i, increased by the mean blocking delay experienced by the message when crossing channels of dimension i. The same argument is applied to messages at the entrance and exit of channels within dimensions.

Let us consider a channel having two incoming message rates α and β, which see their latency after crossing the channel as $T_{q,\alpha}$ and $T_{q,\beta}$, respectively. Let us also assume that the message rate of interest is α and the latency seen by messages of that rate at the entrance of the channel is $T_{e,\alpha}$. The calculation of $T_{e,\alpha}$ is given by the following equation [28, 38]

$$T_{e,\alpha} = T_{q,\alpha} + \beta T_{q,\beta}^2 \tag{6.7}$$

Where $\beta T_{q,\beta}$ is the probability that there is a collision between messages of the two flows α and β, and if this does occur, messages of the flow α wait an average time $T_{q,\beta}$.

Let us apply Equation (6.7) to calculate the entering latency seen by messages of the flow of interest at local channels Ch_j^F. Messages of that flow compete with messages of the flow $2^{n-1}\Phi$, newly entering the dimension at any local channel (except at the first one, where those messages compete with messages of the flow $2^{n-1}\Phi + fl \times pcl$), which see an average latency $T_{i,2}^B$ at the exit of the channel. So, $T_{i,j}^F$ can be written as

$$T_{i,j}^F = \begin{cases} D_t + T_i + 2^{n-1}\Phi(T_{i,2}^B)^2 & (j = k_{avg}^F) \\ D_t + T_{i,j+1}^F + 2^{n-1}\Phi(T_{i,2}^B)^2 & (2 \le j < k_{avg}^F) \\ D_t + T_{i,j+1}^F + \left[2^{n-1}\Phi + fl \times pcl\right](T_{i,2}^B)^2 & (j = 1) \end{cases} \tag{6.8}$$

By proceeding in an analogous way for express channels, the following set of equations is obtained

$$T_{i,j}^E = \begin{cases} D_t + T_{i,1}^F + \dfrac{fl \times (1 - pcl)}{m}(T_{i,2}^E)^2 & (j = k_{avg}^E) \\ D_t + T_{i,j+1}^E + \dfrac{fl \times (1 - pcl)}{m}(T_{i,2}^E)^2 & (2 \le j < k_{avg}^E) \\ D_t + T_{i,2}^E + \dfrac{fe \times (1 - pcl)}{m}(T_{i,2}^E)^2 & (j = 1) \end{cases} \tag{6.9}$$

When j is stated to 2 in Equation (6.9) and using Equation (6.8), we obtain the following equation

$$T_{i,2}^{E} = (k_{avg}^{E} - 1)\frac{fl \times (1 - pcl)}{m}(T_{i,2}^{E})^2 + \left[k_{avg}^{F} \times 2^{n-1}\Phi + fl \times pcl\right](T_{i,2}^{B})^2 + T_i + (k_{avg}^{E} + k_{avg}^{F} - 1)D_t \tag{6.10}$$

It is an equation of the second degree with the unknown value $T_{i,2}^{E}$. The solution of this equation can be obtained as a function of $T_{i,2}^{B}$ and T_i, whose expressions will be calculated later. By choosing the sign minus because of consistency reasons, $T_{i,2}^{E}$ can be written as

$$T_{i,2}^{E} = \left(1 - \sqrt{1 - 4 \times ct1 \times ct2}\right)/(2 \times ct2) \tag{6.11}$$

Where $ct1$ and $ct2$ are constants given by

$$ct1 = T_i + (k_{avg}^{E} + k_{avg}^{F} - 1)D_t + \left[k_{avg}^{F} 2^{n-1}\Phi + fl \times pcl\right](T_{i,2}^{B})^2 \tag{6.12}$$

$$ct2 = \frac{(k_{avg}^{E} - 1)fl \times (1 - pcl)}{m} \tag{6.13}$$

A similar analysis is followed to calculate the entering latency, $T_{i,j}^{B}$, seen by messages of the flow of interest at local channels of the first stage Ch_j^{B}. $T_{i,j}^{B}$ is obtained as follows

$$T_{i,j}^{B} = \begin{cases} D_t + T_{i,1}^{E} + 2^{n-1}\Phi(T_{i,2}^{B})^2 & (j = k_{avg}^{B}) \\ D_t + T_{i,j+1}^{B} + 2^{n-1}\Phi(T_{i,2}^{B})^2 & (2 \le j < k_{avg}^{B}) \\ D_t + T_{i,2}^{B} + [fl + (2^{n-1} - 1)\Phi](T_{i,2}^{B})^2 & (j = 1) \end{cases} \tag{6.14}$$

When j is stated to 2, we obtain the following equation

$$T_{i,2}^{B} = 2^{n-1}\Phi(k_{avg}^{B}-1)(T_{i,2}^{B})^2 + (k_{avg}^{B}-1)D_i + T_{i,1}^{E} \tag{6.15}$$

This equation of the second degree, where the unknown value is $T_{i,2}^{B}$, reveals that its solution is a function of $T_{i,1}^{E}$. On the other hand, Equations (6.9) and (6.10) show that $T_{i,1}^{E}$ is a function of $T_{i,2}^{B}$. To solve such a problem, $T_{i,1}^{E}$ and $T_{i,2}^{B}$ are calculated using iterative techniques for solving equations. Once their values are determined, the latency seen by messages of the flow Φ at the entrance of the first channel of dimension i, $T_{i,1}^{B}$, is then calculated using Equation (6.14).

Recall that T_i has been defined as the latency seen by messages after exiting dimension i. At that point, a message can be either consumed with the probability p_s^{n-i-1}, and therefore skips all the subsequent dimensions j $(i < j < n)$, or it crosses subsequent dimensions j $(i < j < n)$ with the probability $(1-p_s)p_s^{j-i-1}$. In the first case, T_i is equal to the mean waiting time to acquire the ejection channel, W_{qd} (given by Equation (2.11)), and its transfer time through that channel. In the second case, the message competes to access each dimension j $(i < j < n)$ with messages of two flows. The first flow is composed of messages entering that dimension from the local PE. The second one is composed of messages entering that dimension from the previous dimensions other than i. Taking into account these two possibilities, T_i can be expressed as follows

$$T_i = \begin{cases} W_{qd} + L_m & (i = n-1) \\ p_s^{n-i-1}(W_{qd}+L_m) + \sum_{j=i+1}^{n-1}(1-p_s)p_s^{j-i-1}\left[T_{j,1}^{B} + \left(\sum_{\substack{k=0 \\ (k\neq i)}}^{j-1}(1-p_s)^2 p_s^{j-k-1} m'\right)(T_{j,1}^{B})^2\right] & (0 \le i < n-1) \end{cases} \tag{6.16}$$

Messages which enter the network through dimension i compete at the first channel with messages which share the same path, and arrive on the lowest dimensions j $(0 \leq j < i)$. Therefore, the network latency of messages entering the network through dimension i can be given as

$$S_i = T_{i,1}^B + R_i^P (T_{i,1}^B)^2 \tag{6.17}$$

The mean network latency, L, is obtained by considering all possible ways that a message can enter the network. It is given by

$$L = d_{avg} + \sum_{i=0}^{n-1} (1 - p_s) p_s^i S_i \tag{6.18}$$

To include the multiplexing effects of virtual channels, we proceed in a similar way as in chapter 2. Using the diagram of Figure 6.3, state S_v passes to state S_{v+1} at rate m^{BIFIE} (the total message rate entering a local channel of the first or last stage, or express channel, respectively), and to previous state S_{v-1} at rate $1/T_{i,j}^{BIFIE}$.

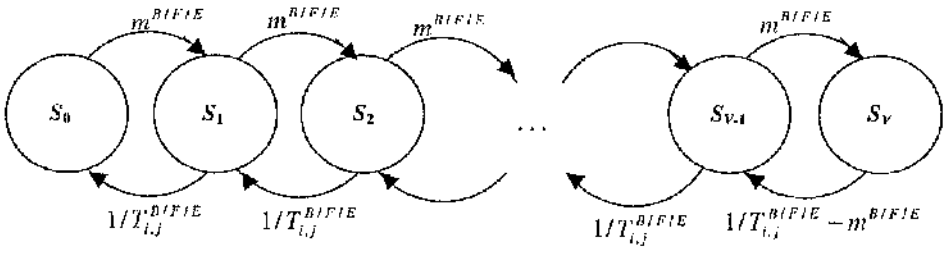


Figure 6.3: Transition diagram to compute the occupancy probabilities of the virtual channels in the express-cube.

The occupancy probabilities of virtual channels, $P_v^{B/F/E}$, are found to be [34]

$$P_{i,j,v}^{B/F/E} = \begin{cases} \dfrac{1}{\sum_{l=0}^{V} q_{i,j,l}^{B/F/E}} & v = 0 \\ P_{i,j,v-1}^{B/F/E} m^{B/F/E} T_{i,j}^{B/F/E} & 0 < v < V \\ P_{i,j,v-1}^{B/F/E} \dfrac{m^{B/F/E}}{1/T_{i,j}^{B/F/E} - m^{B/F/E}} & v = V \end{cases} \tag{6.19}$$

With $q_{i,j,v}^{B/F/E}$ being an intermediate variable used in the calculation of $P_{i,j,v}^{B/F/E}$.

$$q_{i,j,v}^{B/F/E} = \begin{cases} 1 & v = 0 \\ q_{i,j,v-1}^{B/F/E} m^{B/F/E} T_{i,j}^{B/F/E} & 0 < v < V \\ q_{i,j,v-1}^{B/F/E} \dfrac{m^{B/F/E}}{1/T_{i,j}^{B/F/E} - m^{B/F/E}} & v = V \end{cases} \tag{6.20}$$

The multiplexing delay of virtual channels at the j^{th} local (of the first or final stage) or express channel in dimension i is then

$$V_{i,j}^{B/F/E} = \frac{\sum_{v=0}^{V} v^2 P_{i,j,v}^{B/F/E}}{\sum_{v=0}^{V} v \; P_{i,j,v}^{B/F/E}} \tag{6.21}$$

Taking the average multiplexing delay over all physical channels along the message path, gives V_{avg} as

$$V_{avg} = \frac{\sum_{i=0}^{n-1}\left(\sum_{j=1}^{k_{avg}^{B}} V_{i,j}^{B} + \sum_{j=1}^{k_{avg}^{E}} V_{i,j}^{E} + \sum_{j=1}^{k_{avg}^{F}} V_{i,j}^{F}\right)}{nk_{avg}} \tag{6.22}$$

Finally, the mean message latency, including the mean waiting time at the source (given in chapter 2 by Equation (2.13)) and the effects of virtual channel multiplexing, is given as follows

$$Latency = [W_{qs} + L]V_{avg} \tag{6.23}$$

6.3.2 The DCSH Model

A similar approach to the above is followed to obtain the mean message latency in the DCSH. It is worth mentioning that a model has already been developed for the DCSH in [112]. Nevertheless, the simplicity of the present model has allowed us to extend it to derive the model for deterministic routing with the virtual channel concept, presented in chapter 3. The model for the DCSH is more simplified, since a message can cross at most one channel within a dimension. Channels are uni-directional, and therefore $m' = m_g$.

Recall that T_i^c has been defined in chapter 3 as the mean blocking delay experienced by messages within dimension i. This blocking delay is the sum of two blocking delays. The first one is due to the contention with messages arriving from the previous dimensions to enter that dimension (channel). The second one is caused by message contention at the input multiplexer of that dimension, B_i^m. Since Φ is the message rate entering dimension i, the traffic rate received by a multiplexer entry is $\Phi/(k-1)$. Therefore, a message at an entry of the multiplexer competes with messages of

the $(k-2)$ other entries to pass through the multiplexer. Hence, B_i^m can be written as

$$B_i^m = \frac{k-2}{k-1}\Phi(T_i)^2 \tag{6.24}$$

Adding the two delays together yields T_i^c as

$$T_i^c = R_i^p (T_i)^2 + B_i^m \tag{6.25}$$

Taking into account the blocking delays at the channel and multiplexer of each subsequent dimension j $(i < j < n)$, T_i is given as follows

$$T_i = \begin{cases} D_i + W_{qd} + L_m & (i = n-1) \\ D_i + p_s^{n-i-1}(W_{qd} + L_m) + \sum_{j=i+1}^{n-1} (1-p_s) p_s^{j-i-1} \left[B_j^m + T_j + \left(R_i^L + \sum_{\substack{k=0 \\ (k \neq 1)}}^{j-3} (1-p_s)^2 p_s^{j-k-1} m' \right) (T_j)^2 \right] & (0 \le i < n-1) \end{cases} \tag{6.26}$$

As mentioned before, the ordering traversal of dimensions in the DCSH is sufficient to guarantee deadlock-free routing. So, the mean waiting time at the source is found to be

$$W_{qs} = m_g L^2 \tag{6.27}$$

Finally, the mean message latency including the mean waiting time at the source is given by

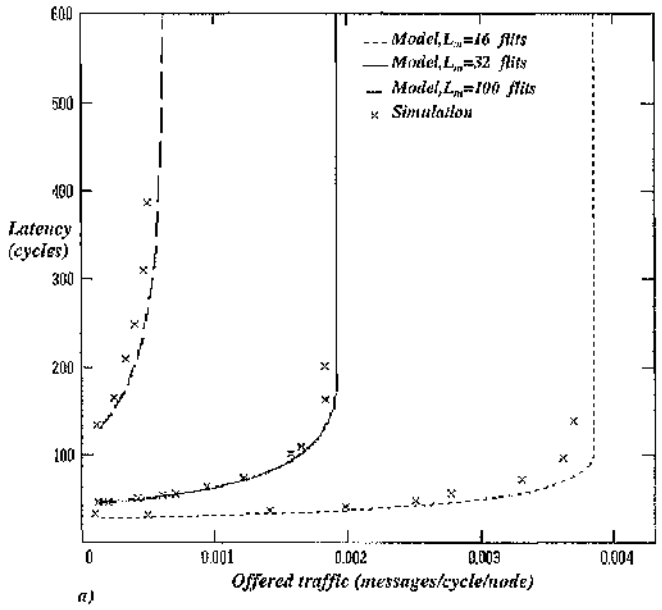
$$Latency = W_{qs} + L \tag{6.28}$$

Where L is the mean network latency, and it is calculated using Equations (3.5) and (3.9).

6.3.3 Validation of the Models

To validate the above models, discrete-event simulators, operating at the flit level, have been developed for the DCSH and express-cube. Simulation results have been collected by considering similar assumptions to those taken by the models, i.e., message generation at each node follows a Poisson distribution. Message length is exponentially distributed with an average of L_m flits. Any system node can be the destination of any generated message, reflecting uniform traffic distribution. The arbitration between messages at outgoing channels of routers is based on Round-Robin scheme.

Figures 6.4 to 6.6 show validation results for the express-cube and DCSH of 1024 nodes, respectively. The message length has been varied between 16, 32 and 100 flits and router delay between 0 and 1 cycle. The simulator has been run by considering different numbers of express channels. In Figure 6.4, the results are obtained when the number of express channels is set to the minimum, i.e., one channel, while Figure 6.5 shows the results when the number of express channels is increased to 4. The figures reveal, in general, that the results from the models are in close agreement with those from simulations for all cases under study. We can therefore conclude that the models are sufficiently accurate to be used as a basis for comparing the performance of the DCSH and express-cube.



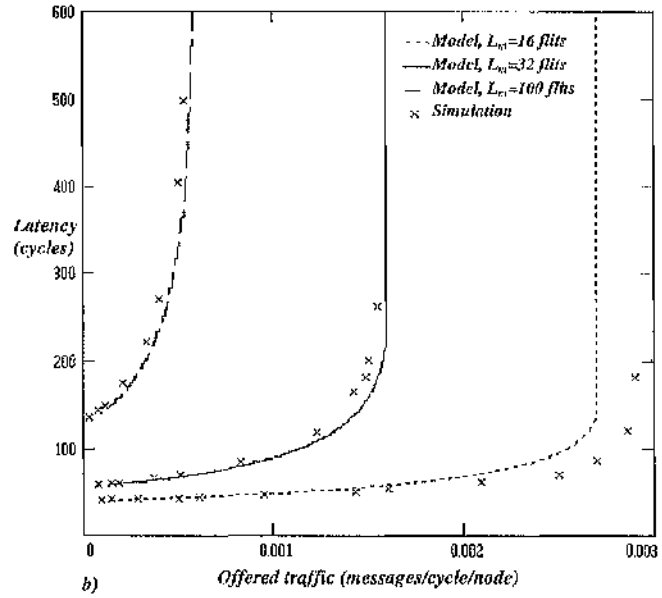
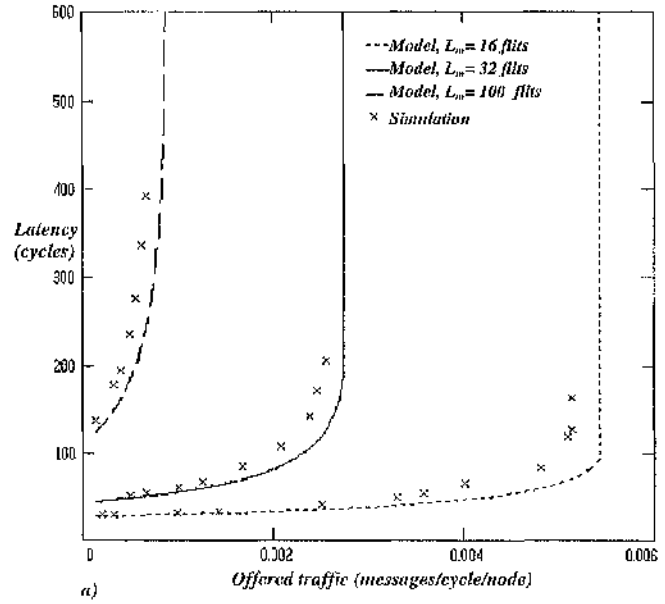


Figure 6.4: Model validation of the express-cube, $m = 1$.

a) $D_t = 0$, b) $D_t = 1$.



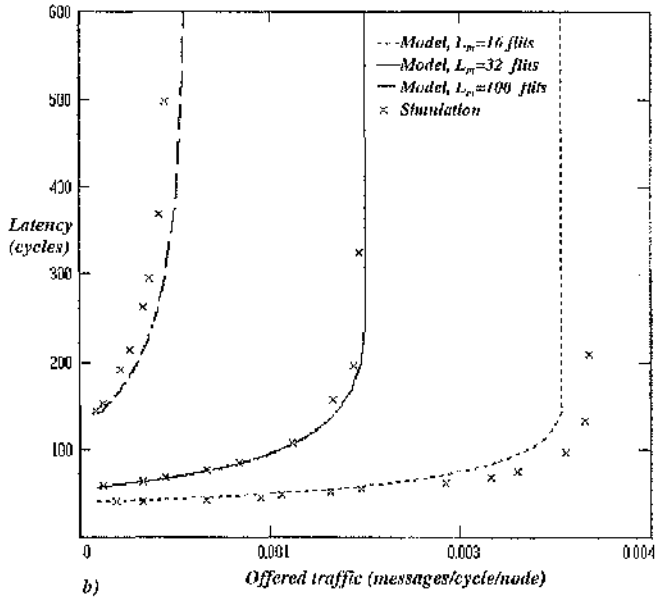
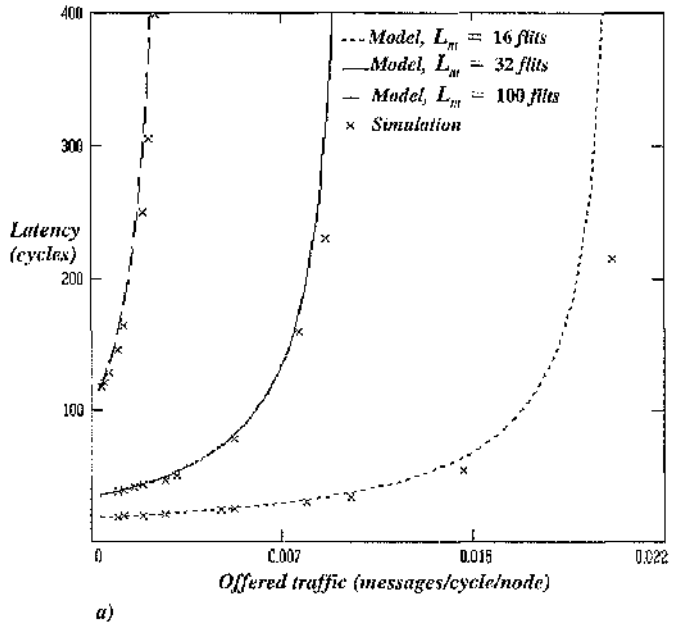


Figure 6.5: Model validation of the express-cube, $m = 4$.

a) $D_i = 0$, b) $D_i = 1$.



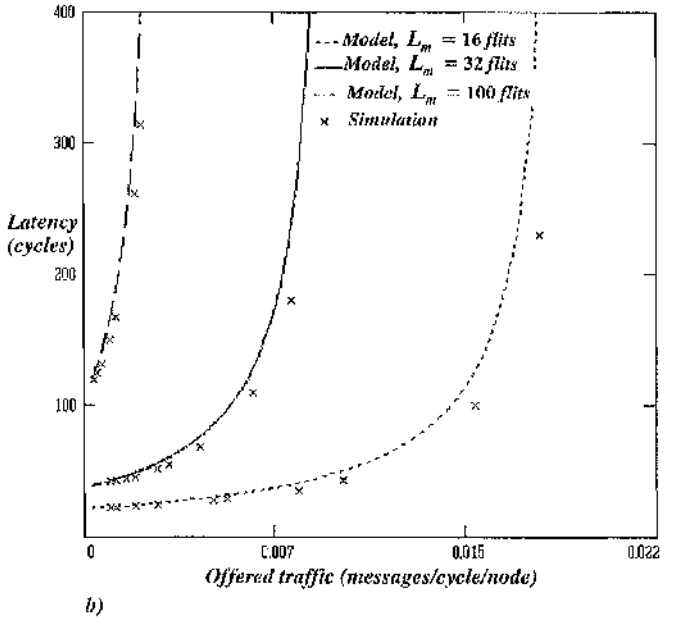


Figure 6.6: Model validation of the DCSH.

a) $D_t = 0$, b) $D_t = 1$.

6.4 Performance Comparison

The following discussion compares the two-dimensional DCSH to its express-cube counterpart with a fixed network size N and equal implementation cost in VLSI and multiple-chip technologies. The comparative analysis assumes that express and local channels in the express-cube have the same width. The bisection width (B) of the DCSH has already been calculated in chapter 4, and the bisection width of the express-cube, with a channel width of $w_{Express\ Cube}$, is found to be

$$B_{Express\text{-}Cube} = 4(m+1)k^{n-1}w_{Express\text{-}Cube} \tag{6.29}$$

Under constant bisection width, the channel width of the express-cube in terms of that of the DCSH is given by

$$w_{Express-Cube} = \left\lceil \frac{k}{4(m+1)} w_{DCSH} \right\rceil \tag{6.30}$$

The node pin-out (P) of the express-cube is given by the following set of equations

$$\begin{aligned} P_{Express-Cube} &= 4nw_{Express\ Cube} && \text{for a node} \\ P_{Express\text{-}Cube} &= 4(m+1)w_{Express\text{-}Cube} && \text{for an interchange} \end{aligned} \tag{6.31}$$

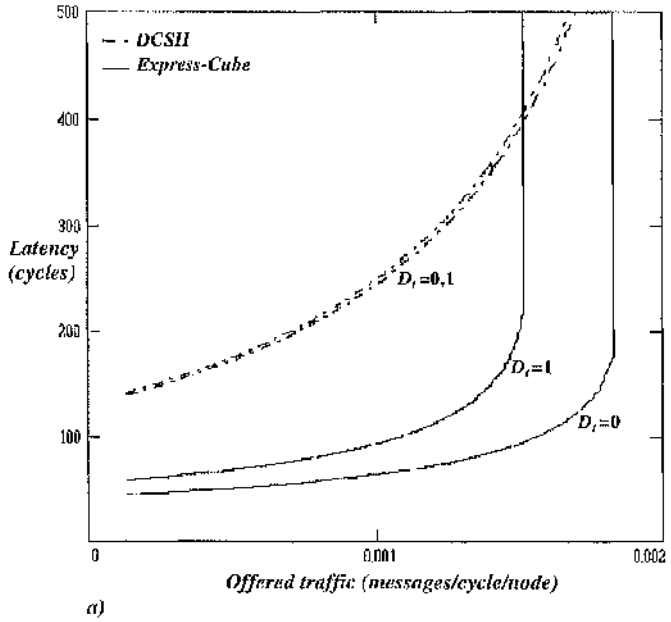
Since the minimum number of express channels that can be used in the express-cube must be at least equal to one ($m \geq 1$), the interchange pin-out dominates the cost. Under constant node pin-out, the channel width relationship of the two networks is given as follows

$$w_{Express\text{-}Cube} = \left\lceil \frac{nk}{4(m+1)} w_{DCSH} \right\rceil \tag{6.32}$$

Express-cubes have been suggested to reduce the message distance in large systems. So, for illustration, two large network sizes are considered: 1024 and 4096 nodes. k_n is set to 4 for both system sizes, this figure has been found through experiments to yield the optimal message distance along a dimension. Dally [37] has suggested the use of multiple express channels to reduce message contention. This has the advantage of making better use of the express channels since a message can choose any one of them. This flexibility, however, is achieved at the expense of higher node pin-out and increased wiring density requirements. Hence, the use of more than one express channel has been taken into consideration. Two values have been assigned to m: m=1 and m=3. The former represents the case where the minimum number of express channels is used, while the latter is used as a representative figure to assess the effects of increasing the number of express channels on the performance of express-cubes.

6.4.1 Results for VLSI Implementation

Figure 6.7 shows the mean message latency as a function of the traffic rate in the DCSH and express-cube of 1024 nodes, for long and short messages respectively. Different figures for router delays are considered. The number of express channels in the express-cube is set to m=1 in this set of results. Under constant bisection width, a channel in the express-cube is 4 times wider than that in the DCSH. The results reveal that with zero-cycle router delay the express-cube outperforms the DCSH under low and moderate traffic loads at long and short message lengths. This is because the wide channels in the express-cube reduce the message aspect ratio component of latency. As the traffic increases, however, the express-cube loses edge to the DCSH at short message lengths, as depicted in Figure 6.7-(b). Even though the message aspect ratio in the express-cube is smaller than that in the DCSH, the express-cube offsets that advantage by the higher



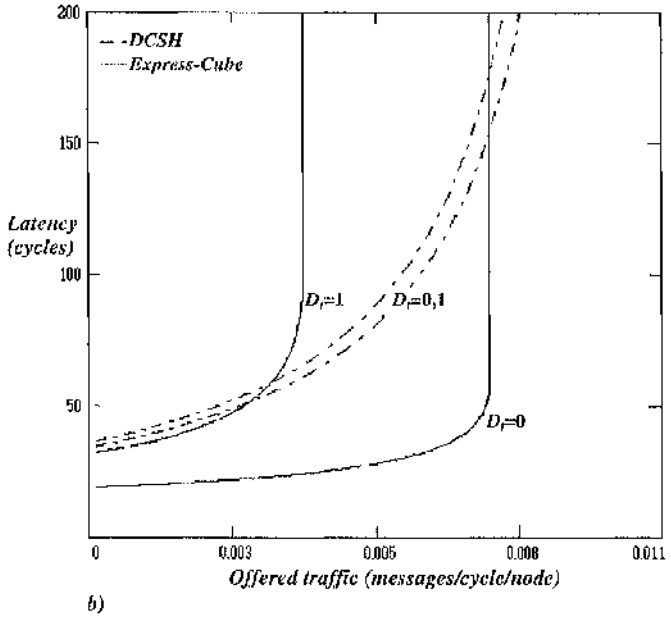
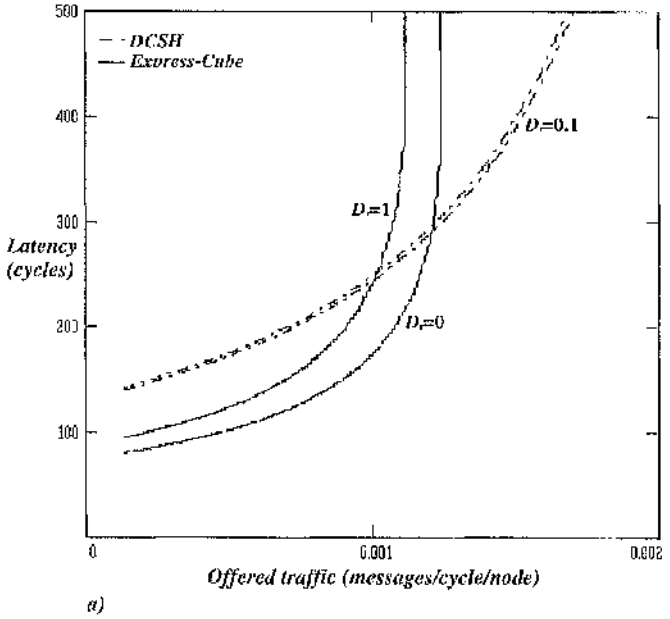


Figure 6.7: The performance of the DCSH and express-cube in VLSI for 1024 nodes and m =1. a) L_m=128 flits, *b*) L_m=32 flits.

blocking delay that messages experience in the network due the large distance that they cross to reach their destination. When the router delay is increased to one cycle, which is an optimistic figure given current technologies [2, 37], the DCSH provides a lower latency than the express-cube under moderate and heavy traffic for both long and short messages.

Keeping the same system size (N=1024 nodes) while increasing the number of express channels to m=3, Figure 6.8 shows that the DCSH performs better than the express-cube at moderate and heavy traffic loads. This is because, firstly, the ratio between message



a)

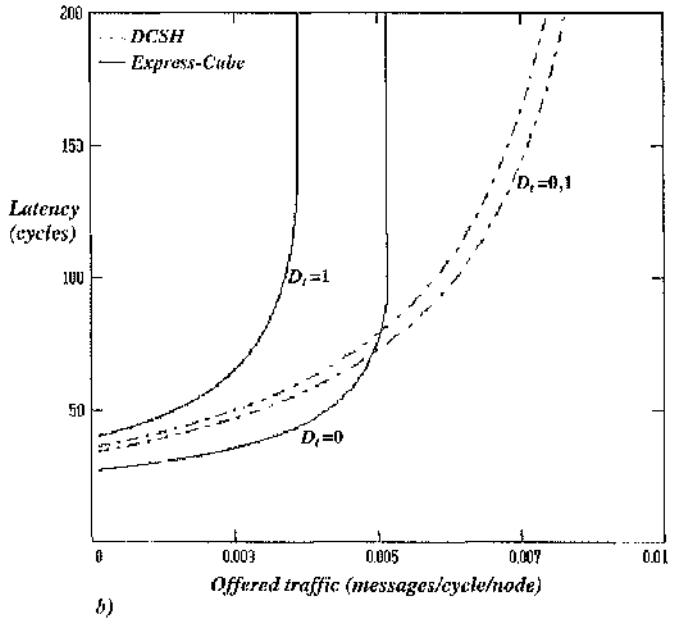
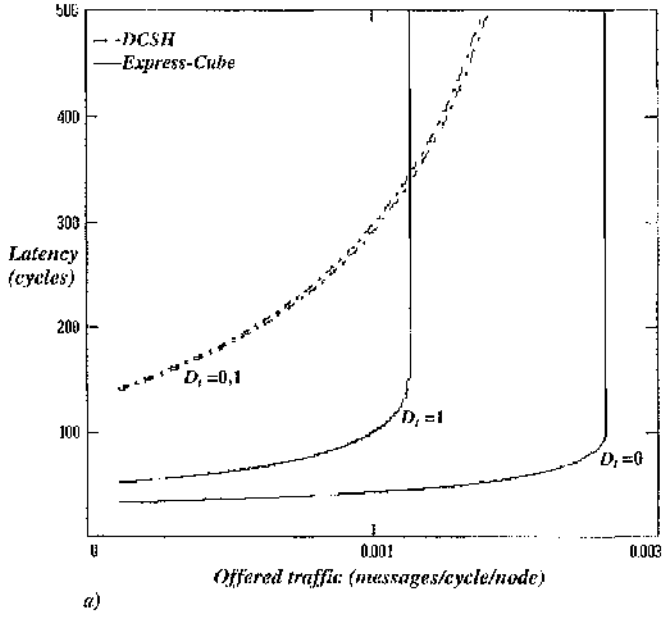


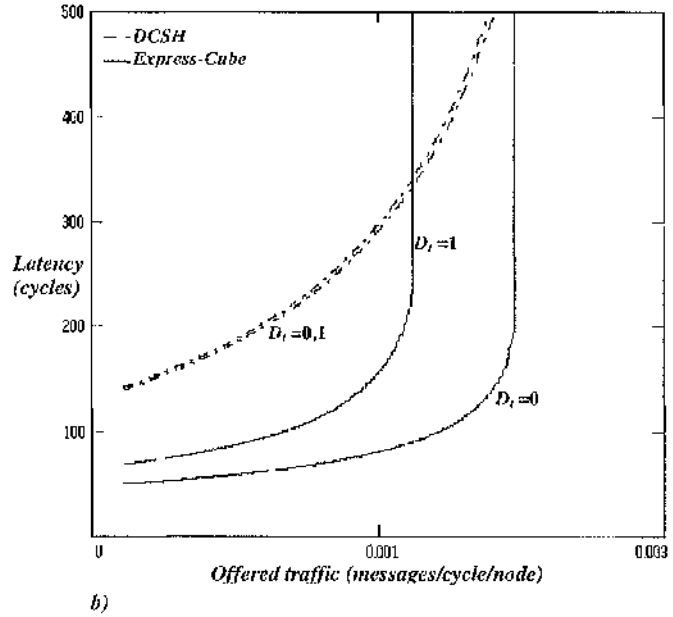
Figure 6.8: The performance of the DCSH and express-cube in VLSI for 1024 nodes and m =3. a) L_m=128 flits, *b*) L_m=32 flits.

aspect ratios of the two networks is reduced to half. Equation (6.30) reveals that the channel width in the express-cube is reversibly proportional to the number of express channels, m. Therefore, when m is increased to 3, channels tend to be narrower, which increases the message aspect ratio in the express-cube. Secondly, messages cross relatively a large distance in the express-cube, encountering higher message blocking.

Figure 6.9-(a) evaluates the performance of the two networks when their size is scaled to 4096 nodes while holding the number of express channels to one. A larger system size implies wider channels in the express-cube, but also a longer distance, compared to the DCSH. The message aspect ratio in the DCSH in this case is 8 times larger than that in the express-cube. The results reveal that zero-cycle router delay favours the express-cube



a)



b)

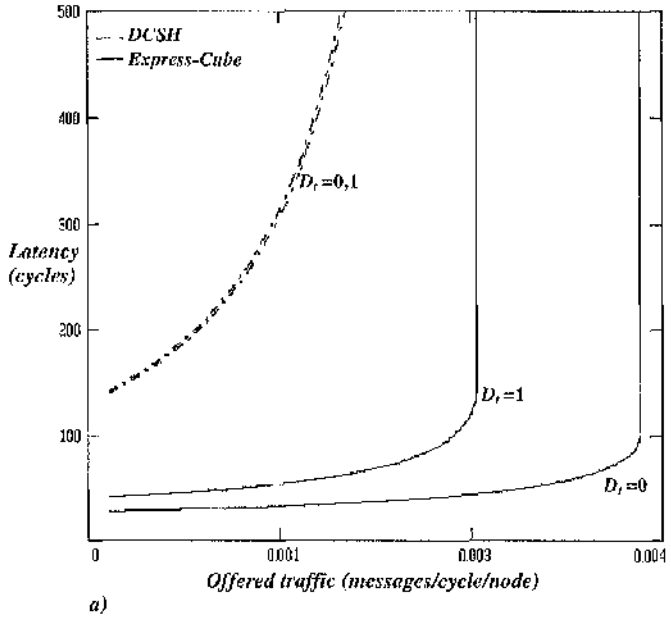
Figure 6.9: The performance of the DCSH and express-cube in VLSI for 4096 nodes and L_m=128 flits. a) m =1, b) m =3.

at all traffic conditions. Including the effects of router delays, however, even when it is only one cycle, causes performance degradation in the express-cube at heavy loads.

When the number of express channels is increased to m=3, Figure 6.9-(b) also shows that the express-cube provides a higher message latency at heavy traffic intensity. It can be noticed that the increase in m has less degradation effects on the performance of the express-cube compared to the 1024 node case. This can be explained by inspecting Equation (6.30). The equation reveals that increasing the radix k reduces the effects on the channel width in the express-cube caused by an increase in m.

6.4.2 Results for Multiple-chip Implementation

Under constant node pin-out, the ratio in channel width of the express-cube and DCSH is greater than that found under the bisection width constraint. For example, a system size of 1024 nodes and one express channel, the channel width in the express-cube is 8 times wider than that in the DCSH. Figures 6.10 shows the performance results for a 1024 node system. Similar conclusions are drawn when the size is increased to 4096; the large difference in the message aspect ratios in the two networks favours the express-cube. The latter outperforms the DCSH under all traffic conditions. Increasing the router delay to 1 cycle or the number of express channels to m=3 leads to similar conclusions.



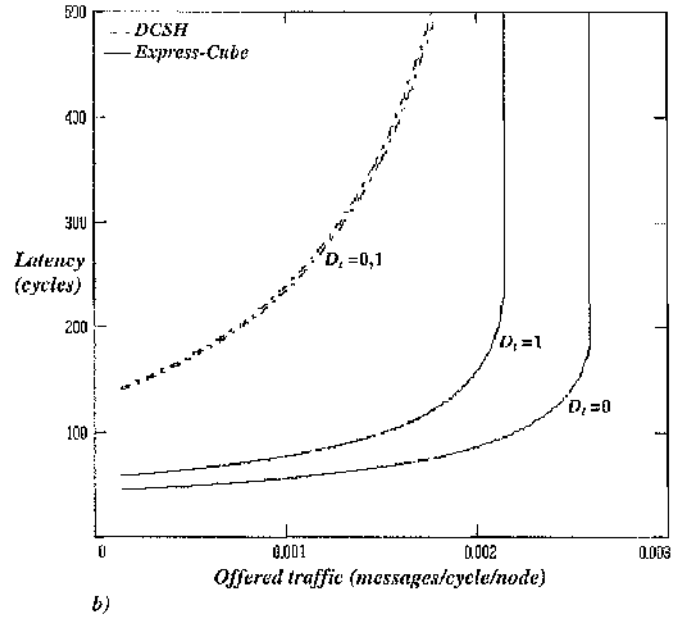


Figure 6.10: The performance of the DCSH and express-cube in multiple-chip for 1024 nodes and L_m=128 flits. a) m =1, *b*) m =3.

6.5 Conclusion

Express-cubes have been introduced to alleviate the problem of large message distance in k-ary n-cubes by allowing non-local messages to partially bypass groups of nodes within a dimension. The DCSH can be considered as the extreme case of the express-cube where one level of a total bypass path is provided along a dimension. This total bypassing reduces considerably the diameter and message average distance at the expense of increased wiring density, compared to express-cubes. This study has investigated whether the total bypass strategy of the DCSH can compensate for its narrower channels and provides superior performance to the partial bypassing of the express-cube.

The results of this comparative analysis reveal that the express-cube can outperform the DCSH only when the two networks are implemented in multiple-chip technology. Under VLSI implementation, the DCSH delivers lower message latency than the express-cube under moderate and heavy traffic loads, especially with short messages. The advantage of the total bypassing offered by the DCSH becomes more important when the network size is large and when the effects of router delays are taken into account. Moreover, the analysis shows that the use of multiple express channels has degrading effects on the performance of express-cubes. While they manage to reduce delays due message blocking, they increase the wiring density of the express-cube, and consequently they decrease their channel bandwidth.

Chapter 7

Conclusions and Future Directions

The interconnection network is a crucial component in any parallel computer since any interaction between the processing elements ultimately depends on its effectiveness. Although many network architectures have been studied, and indeed deployed, none has proved clearly superior in all roles, since the communication requirements of different applications vary widely. Certain applications map most naturally onto specific topologies (for example, finite element calculations map onto mesh-like structures), but while universal optimality is unattainable, it remains a desirable goal to identify physical interconnections that have the ability to support general-purpose multicomputer architectures.

Recent work on regular hypergraph topologies has produced a number of strongly-connected networks of which the Distributed Crossbar-Switch Hypermesh (DCSH), is one of the most promising. This thesis has extended earlier work to examine the

behaviour of this network by, for the first time, studying its performance under complex traffic patterns. The work has, in particular, focussed on adaptive routing strategies and broadcast traffic using both mathematical models and simulations based on discrete-event model.

Throughout this study, we have attempted to use realistic assumptions such as real-world implementation constraints and router delays. The k-ary n-cube has been selected as a benchmark topology when assessing the performance of the DCSH due to the fact that it is currently the dominant network employed in experimental and commercial parallel systems.

Considering first the DCSH alone, results reveal that, under uniform traffic distribution at heavy loads, the DCSH has superior message latency when deterministic routing is used. Due to the fact that messages in the DCSH make few hops to cross the network, the decrease in the blocking delays of message headers provided by adaptive routing is overwhelmed by the multiplexing delays at physical channels and node inputs. However, when the traffic is highly non-uniform, in applications such as matrix transpose communication, adaptive routing does provide significant benefits. Even when the deterministic routing strategy is equipped with more virtual channels, this does not significantly improve its performance.

Comparing the DCSH and k-ary n-cubes when adaptive routing is used, the latter has superior performance with long messages. However, the DCSH, with its small average message distance, is less sensitive to router delay effects and it quickly pulls ahead when messages are short and traffic load is moderate. Furthermore, when systems are scaled up, the k-ary n-cube performance degrades as the message aspect ratio decreases and the

rapidly increasing average message distance affects latency which becomes dominated by the distance.

When broadcast operations are included and their impact on the performance of background traffic composed of unicast messages is studied, the basic operation found in any application, the DCSH is seen to be capable of sustaining a heavier broadcast load than the k-ary n-cube. Again, because of the k-ary n-cube longer message distance, unicast messages are more severely affected by the blocking delays caused by broadcast traffic, leading to early saturation. This performance degradation increases with the number of broadcasting nodes or with system scaling.

In terms of Dally's express-cube proposal, when compared with conventional k-ary n-cube with no bypass channels at all, the DCSH benefits from what can be viewed as a total bypassing strategy. However, the question arises as to whether the partial bypassing proposed by Dally (the express-cube) might provide equivalent or greater benefit. It transpires that this is indeed sometimes the case, with the express-cube exhibiting superior performance under the node pin-out constraint. However, the bisection width constraint favours the total bypassing strategy of the DCSH. Even though the DCSH possesses narrower channels, its smaller average message distance compensates for this and provides lower message latency. This becomes more significant at short message lengths, or when the network size increases. Moreover, the increase in the number of express channels has shown to degrade the express-cube performance, comparatively to the DCSH.

Overall, for scalable systems, guaranteeing a low average distance for communication has proved more important than providing wide channels. This is true especially under the

dominant wormhole switching technique when router hardware incorporates advanced but complex features such as adaptivity and when messages are short. Since short messages characterise the traffic pattern in multicomputers based on the distributed shared-memory (DSM) model, which is gaining a growing interest in the design of current and future multicomputers, the DCSH is a strong candidate topology for the support of such systems.

There are several other interesting issues which have emerged from this work and which might merit further study. The first is related to router design in the DCSH. Multiple channels within a cluster eliminate message contention on channels, but they shift this contention to router inputs, so that messages experience a long service time at input multiplexers. If these are operating at the flit level, a large number of multiplexing inputs increases message delay (especially with adaptive routing as shown in chapter 3); if they are operating at the message level, there can be a waste of channel bandwidth. When the multiplexer is servicing one message, others may be blocked behind it even though the outputs required are idle. It seems possible that an enhancement to router design by introducing new multiplexing scheme at the input might further improve performance.

Secondly, an important feature of the DCSH is its small diameter, which allows it to outperform many other topologies. However, with wormhole routing under heavy traffic conditions, when a message header is blocked at the destination, waiting for the ejection channel, the whole message is spread out along its path, resulting in a considerable delay for messages behind it. The improvement of DCSH performance through the increase in the number of ejection channels has been examined in chapter 4. Since multiple ejection channels are costly to implement, it would be interesting to determine the optimal

number.

In the assessment of the DCSH performance and its comparison to other common networks, the analysis has been limited to uniform traffic distribution. However, there are many applications which generate a non-uniform traffic pattern. In order to get a more complete view of the performance that this type of network might offer to future scalable parallel systems, it would be useful to extend the analysis to include non-uniform communication schemes. Furthermore, channels and nodes are subject to failures. Up to now, the investigation of the DCSH has been conducted assuming perfect reliability. The analysis should now be extended to examine the impact of network failures.

Application mapping is another important area which requires further investigation. Typically, interconnection networks match a certain class of application. One of the most important characteristics of the DCSH is its high connectivity, which enables it to support a wide range of applications that might, at first sight, appear to map more naturally onto other topologies. A comprehensive study to investigate explicitly to what extent this ability extends, and to compare that to other graph and hypermesh implementations taking into account traffic contention, would be extremely useful.

One last issue which is worth further investigation is to relate the performance profiles which have emerged from this and other similar work to the higher level models familiar to programmers coding real parallel applications. A better understanding is required of the traffic patterns generated by such applications and more generally by styles of parallel programming such as those emerging from models like BSP. With this knowledge, it would be much easier for application developers to evaluate claims such as those made

here for the performance benefits of particular architectures, but also easier for those simulating the machines to select realistic traffic patterns.

References

[1] S. Abraham & K. Padmanabhan, Performance of Multicomputer Networks under Pin-out Constraints, *Journal of Par. & Dist. Syst.* 1991, pp. 237-248.

[2] A. Agrawal, Limits on Interconnection Network Performance, *IEEE Trans. Par. & Dist. Syst.*, Vol. 2, 1991, pp. 398-412.

[3] A. Agarwal *et al*, The MIT Alewife Machine: Architecture and Performance, *Proc. 22nd Ann. Int. Symp. Comp. Archit.*, 1995.

[4] J.R. Anderson & S. Abraham, Multidimensional Network Performance with Unidirectional Links, *Proc. Int. Conf. Par. Proc., IEEE Comp. Soc.*, 1997, pp. 26-33.

[5] K. Aoyama, Design Issues in Implementing an Adaptive Router, M. Sc. Thesis, Comp. Sci. Dept., University of Illinois, 1993.

[6] R. Arlanskas, iPSC/2 system: A Second Generation Hypercube, *Proc. 3rd ACM Conf. Hypercube Concurrent Computers & Applications*, Vol. 1, 1988.

[7] W.C. Athas & C.L. Seitz, Multicomputers: Message Passing Concurrent Computers, *IEEE Computer*, 21(8), 1988, pp. 9-24.

[8] Bakoglu, Circuits, Interconnections, Packaging for VLSI, Addison-Wesley 1990.

[9] D. Basak & D. Panda, Alleviating Consumption Channel Bottleneck in Wormhole-Routed *k*-ary *n*-cube Systems, *IEEE Trans. Par. & Dist. Syst.*, 9(5), 1998, pp. 481-496.

[10] P.E. Berman *et al*, Adaptive Deadlock and Livelock-free Routing with all Minimal paths in Torus Networks, *Proc. 4th Symp. Par. Algorithms & Archit.*, 1992, pp. 3-12.

[11] C. Berge, *Graphs and Hypergraphs*, North-Holland, 1977.

[12] L.N. Bhuyan *et al*, A General Class of Processor Interconnection Strategies, *Proc. 9th Int. Symp. Comp. Archit.*, 1982.

[13] L.N. Bhuyan & D. P. Agrawal, Generalized Hypercube and Hyperbus Structures for a Computer Network, *IEEE Trans. Comp.*, C-33(4), 1984, pp. 323-333.

[14] T. Boku *et al*, CP-CAPS: A Massively Parallel Processor for Large Scale Scientific Calculations, *Proc. ACM Supercomputing*, 1997, pp. 108-115.

[15] K. Bolding & W. Yost, The Express Broadcast Network: A Network for Low Cost Broadcast of Control Messages, *Proc. Int. Conf. on Algorithms & Architectures for Parallel Processing*, 1995.

[16] R.V. Boppana *et al*, Resource Deadlocks and Performance of Wormhole Multicast Routing Algorithms, *IEEE Trans. Par. Dist. Syst.*, 9(6), 1998, pp. 535-549.

[17] R.V. Boppana & S. Chalasani, A Comparison of Adaptive Wormhole Routing Algorithms, *Proc. 20th Ann. Int. Symp. on Comp. Archit.*, 1993.

[18] R.V. Boppana & S. Chalasani, A Framework for Designing Deadlock-free Wormhole Routing Algorithms, *IEEE Trans. Par. Dist. Syst.*, 7(2), 1996, pp. 169-

183.

[19] Y. Boura & C.R. Das, Modeling Virtual Channel Flow Control in *n*-dimensional Hypercubes, *Proc. Int. Symp. High Perform. Comp. Archit.*, 1995, pp. 166-175.

[20] Y. Boura *et al*, A Performance Model for Adaptive Routing in Hypercubes, *Proc. Int. Workshop Par. Proc.* , 1994, pp. 11-16.

[21] Y.M. Boura & C.R. Das, Performance Analysis of Buffering Schemes in the Routers, *IEEE Trans. Comp.*, 46(6), 1997, pp. 687-695.

[22] Y.M. Boura & C.R. Das, A Class of Partially Adaptive Routing Algorithms for n-dimensional Meshes, *Proc. 22nd Int. Conf. Par. Proc.*, Vol. 3, 1993, pp. 175-182.

[23] C. Chiang & L.M. Ni, Multicast-address Encoding for Multicast, *Proc. Par. Comp. Routing & Communication Workshop*, 1994, pp. 146-160.

[24] A.A. Chien, A Cost and Speed Model for *k*-ary *n*-cube Wormhole Routers, *IEEE Trans. Par. & Dist. Syst.*, 9(2), 1998, pp. 150-162.

[25] A.A. Chien & J.K. Kim, Planar Adaptive Routing: Low Cost Adaptive Networks for Multiprocessors, *Proc. 19th Ann. Int. Symp. Comp. Archit.*, 1992, pp. 268-277.

[26] J. Choi *et al*, Parallel Matrix Transpose Algorithms on Distributed Memory Concurrent Computers. Tec-Rep. ORNL/TM-12309, Oak Ridge National Laboratory, 1993.

[27] J. Choi *et al*, ScaLAPACK: A Scalable Linear Algebra Library for Distributed Memory Concurrent Computers. *Proc. 4th Symp. on the Frontiers of Massively Par. Comp., IEEE Computer Soc.*, 1992, pp.120-127.

[28] B. Ciciani *et al*, An Accurate Model for the Performance Analysis of

Deterministic Wormhole Routing, *Proc. 11th Int. Par. Proc. Symp.*, 1997, pp. 353-359.

[29] Cray Research Inc., The Cray T3E Scalable Parallel Processing System, on Cray's Web Page at http://www.cray.com/PUBLIC/product-info/T3E/CRAY_T3E.html.

[30] W. Crowther *et al*, Performance Measurements on a 128 Node Butterfly Parallel Processor, *Proc. Int. Conf. Par. Proc. (ICPP)*, 1985, pp. 531-540.

[31] D. Culler *et al*, Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann (Ed.) 1997.

[32] W.J. Dally & C.L. Seitz, Deadlock-free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. Comp.*, 36(5), 1987, pp. 547-553.

[33] W.J. Dally, Performance Analysis of *k*-ary *n*-cubes Interconnection Networks, *IEEE Trans. Comp.*, C-39(6), 1990, pp. 775-785.

[34] W.J. Dally, Virtual Channel Flow Control, *IEEE Trans. Par. & Dist. Syst.*, 3(2), 1992, pp. 194-205.

[35] W.J. Dally & C.L. Seitz, The Torus Routing Chip, *Journal of Dist. Comp.*, 1(3), 1986, pp. 187-196.

[36] W.J. Dally & H. Aoki, Deadlock-free Adaptive Routing in Multicomputer Networks Using Virtual Channels, *IEEE Trans. Par. & Dist. Syst.* 4(4), 1993, pp. 66-74.

[37] W.J. Dally, Express-cubes: Improving the Performance of *k*-ary *n*-cube Interconnection Networks, *IEEE Trans. Comp.* C-40 (9), 1991, pp. 1016-1023.

[38] W.J. Dally *et al*, VLSI and Parallel Computation, R. Suaya & G. Birtwistle (Eds.)

1990.

[39] W.J. Dally *et al*, The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers, *Proc. 1st Workshop on Par. Comp. Routing & Communication*, K. Bolding and L. Snyder (Eds.) *LNCS*, 1994, pp. 241-255.

[40] P. W. Dowd, High Speed Routing in a Distributed Memory Parallel System: A Simulation Study. IICS, 1(2), 1991.

[41] P.W. Dowd & K. Djabbor, Spanning Multi-access Channel Hypercube Computer Interconnection, *IEEE Trans. Comp.*, C-37(9), 1988, pp. 283-290.

[42] J.T. Draper & J. Ghosh, A Comprehensive Analytical Model for Wormhole Routing in Multicomputer Systems, *Journal of Par. & Dist. Comp.*, Vol 23, 1994, pp. 202-214.

[43] J. Duato *et al*, Interconnection Networks: An Engineering Approach, Matt Loeb (Ed.) 1997.

[44] J. Duato & P. Lopez, Performance Evaluation of Adaptive Routing Algorithms for *k*-ary *n*-cubes, *Par. Comp. Routing and Communication, K. Bolding & L. Snyder (Eds.) LNCS*, 1994.

[45] J. Duato, On the Design of Deadlock-free Adaptive Routing Algorithms for Multicomputers: Design Methodologies, *PARLE'91 Par. Archit. & languages Europe, K. Bolding & L. Snyder (Eds.) LNCS*, Vol. 1, 1991, pp. 390-405.

[46] J. Duato & P. Lopez, Bandwidth Requirements for Wormhole Switches: A Simple and Efficient Design, *Proc. Euromicro Workshop on Par. Dist. Proc.*, 1994, pp. 377-384.

[47] J. Duato, Deadlock-free Adaptive Routing Algorithms for Multicomputers: Evaluation of A New Algorithm, *Proc. 3rd IEEE Int. Symp. Par. & Dist. Proc.*, 1991.

[48] J. Duato, A New Theory of Deadlock-free Adaptive Routing in Wormhole Networks, *IEEE Trans. Par. Dist. Syst.*, 4(12), 1993, pp. 1320-1331.

[49] J. Duato, Improving the Efficiency of Virtual Channels with Time-dependent Selection Functions, *PARLE'92 Parallel Archit. & Languages*, Europe (605), *K. Bolding & L. Snyder (Eds.) LNCS*, 1992, pp. 635-650.

[50] J. Duato, Why Commercial Multicomputers Do Not Use Adaptive Routing, *IEEE Technical Committee on Computer Architecture Newsletter*, 1994, pp. 20-22.

[51] G.C. Fax *et al*, Solving Problems on Concurrent Processors. Volume I: General Techniques and Regular Problems. Prentice Hall 1988.

[52] W. Feng & K. Shin, The Effect of Virtual Channels on the Performance of Wormhole Algorithms in Multicomputer Networks, UM Directed Study Report, May 1994.

[53] A. Ferreira *et al*, Bus Based Parallel Computers: A Viable Way for Massive Parallelism, *PARLE'94 Parallel Archit. & Languages*, Europe (817), *K. Bolding & L. Snyder (Eds.) LNCS*, 1994, pp. 553-564.

[54] M.A. Franklin, Pin Limitation and Partitioning of VLSI Interconnection Network, *IEEE Trans. Comp.*, C-36(5), 1987, pp. 547-553.

[55] H. Fujii *et al*, Architecture and Performance of the Hitachi SR 2201 Massively Parallel Processor System, *Proc. 11th Int. Par. Proc. Symp.*, 1997, pp. 233-241.

[56] W.K. Giloi & S. Montenegro, Choosing the Interconnect of Distributed-memory

Systems by Cost and Blocking Behaviour, *IEEE Par. Proc. Symp.*, 1991, pp. 438-444.

[57] C.J. Glass & L.M. Ni, The Turn Model for Adaptive Routing, *Proc. 19th Ann. Int. Symp. Comp. Archit.*, 1992, pp. 278-287.

[58] A. Gottlieb *et al*, The NYU Ultracompuetr- Designing a MIMD Shared Memory Parallel Computer, *IEEE Trans. Comp.*, C-32, 1983, pp.175-189.

[59] R.I. Greenberg & L. Guan, Modelling and Comparison of Wormhole Routed Mesh and Torus Networks, *Proc. 5th IASRED Int. Conf. Par. & Dist. Comp. Syst.*, 1997.

[60] L. Guan, Message Routing and Problem Solving in Multiprocessor Networks, Ph. D. Thesis, Comp. Sci. Dept, Maryland Univ., 1997.

[61] I.S. Gopal, Prevention of Store-and-Forward Deadlock in Computer Networks, *IEEE Trans. on Communications*, COM-33(12), 1985, pp. 1258-1264.

[62] D.B. Gustavson & D.B. Theus, Wire-Or Logic on Transmission Lines, *IEEE Micro*, 1983, pp. 51-55.

[63] F.T. Hady, A Performance Study of Wormhole Routed Networks Through Analytical Modeling and Experimentation, Ph. D. Thesis, Elect. Eng. Dept., Maryland Univ., 1993.

[64] N.C. Hock, Queueing Modelling Fundamentals, J. Wiley & Sons (Eds.) 1996.

[65] K. Hwang, Advanced Computer Architecture: Parallelism, Scalability and Programmability. McGraww-Hill (Ed.) 1993.

[66] W.T. Hsu, P.C. Yew, The Impact of Wiring Constratints on Hierarchical Network

Performance. *IEEE Symp. Par. & Dist. Proc.*, 1992, pp. 580-588.

[67] H. Ishihata *et al*, Third Generation Message Passing Computer AP1000, *Proc. Int. Symp. Supercomputing*, 1991.

[68] Intel iPSC/1 Reference Manual, 1986.

[69] J. Jaja, Load Balancing and Routing in the Hypercube and Related Networks. *Journal of Par. & Dist. Comp.*, 1992, pp. 431-435.

[70] C.R. Jesshope *et al*, High Performance Communication in Processor Networks, *Proc. 16th Int. Symp. Comp. Archit.*, 1989, pp. 150-157.

[71] J.H. Kai & A.A. Chien, An Evaluation of Planar Adaptive Routing (PAR), *Symp. Par. & Dist. Proc.*, 1992.

[72] P. Kermani & L. Kleinrock, Virtual Cut-through: A New Computer Communication Switching Technique, *Comp. Networks*, Vol. 3, 1979, pp. 267-286.

[73] S.W. Keckler, The Importance of Locality and Load Balancing for Multiprocessors, MIT Laboratory, 1994.

[74] R.E. Kessler & J.L Swarszmeier, Cray T3D: A New Dimension for Cray Research, in Compcon, Spring 1993, pp. 176-182.

[75] J. Kim & C.R. Das, Hypercube Communication Delay with Wormhole Routing, *IEEE Trans. Comp.* , 43(7), 1994, pp. 806-814.

[76] J.H. Kim & A.A. Chien, Network Performance under Bimodal Traffic Loads, *Journal of Par. & Dist. Comp.*, Vol. 28, 1995, pp. 43-64.

[77] J.H. Kim & A.A. Chien, The Impact of Packetization in Wormhole-Routed

Networks, *Proc. Par. Archit. & Languages Europe (PARLE)*, 1993.

[78] S.Y. Kim & K.Y. Chwa, Optimal Embeddings of Multiple Graphs into a Hypermesh, *Int. Conf. Par. & Dist. Syst. (ICPADS)*, 1997, pp. 436-443.

[79] L. Kleinrock, On the Modeling and Analysis of Computer Networks, *Proc. of IEEE*, 81(8), 1993, pp. 1179-1191.

[80] L. Kleinrock, Queueing Systems, Volume 1, John Wiley (Ed.), New York, 1975.

[81] J. Konicek *et al*, The Organization of the Cedar System, *Int. Conf. Par. Proc.*, 1991, pp. 49-56.

[82] J. Kuskin *et al*, The Stanford Flash Multiprocessor, *Proc 21st Ann. Int. Symp. Comp. Archit.*, 1994, pp. 302-313.

[83] C. Leiserson *et al*, The Network Architecture of the Connection Machine CM-5, *Proc. Symp. Par. Algorithms & Archit.*, 1992, pp. 272-282.

[84] D. Lenoski *et al*, The Stanford DASH Multicomputer, *IEEE Comp.*, 25(3), 1992, pp. 63-79.

[85] K. Li & R. Shaefer, A Hypercube Shared Virtual Memory, *Proc. Int. Conf. Par. Proc.*, Vol. I, 1989, pp. 125-132.

[86] X. Lin & L.M. Ni, Deadlock-free Multicast Wormhole Routing in Multicomputer Networks, *Proc. 18th Ann. Int. Symp. Comp. Archit.*, 1991, pp. 116-125.

[87] X. Lin *et al*, Deadlock-free Multicast Wormhole Routing in 2D-Mesh Multicomputers, *IEEE Trans. Par. & Dist. Syst.*, Vol. 5, 1994, pp. 793-804.

[88] X. Lin *et al*, The Message Flow Model for Routing in Wormhole-Routed Networks, *Proc. Int. Par. Proc.*, Vol. 1, 1993, pp. 294-297.

[89] D.H. Linder and J.C. Harden, An Adaptive and Fault Tolerant Wormhole Routing Strategy for *k*-ary *n*-cubes, *IEEE Trans. Comp.*, 40(1), 1991, pp. 2-12.

[90] G.J. Lipovsky, Parallel Computing: Theory and Practice. John Wisley & Sons (Eds.) 1989.

[91] K.J. Liszka *et al*, Problems with Comparing Interconnection Networks: Is an Alligator Better than an Armadillo?, *IEEE Concurrency*, 5(4), 1997, pp. 18-28.

[92] Z. Liu & A.A. Chien, Hierarchical Adaptive Routing: A Framework for Fully Adaptive and Deadlock-Free Wormhole Routing, *Symp. Par. & Dist. Proc.*, 1994.

[93] P. Lopez & J. Duato, Deadlock-free Adaptive Routing Algorithms for the 3D-Torus: Limitations and Solutions, *Proc. Par. Archit. & Languages Europe* 93, 1993, pp. 684-687.

[94] R. Lucas, Solving Planar Systems of Equations on Distributed Memory Multiprocessors, Ph. D. Thesis, Elect. Eng. Dept., Stanford Univ., 1987.

[95] L.M. Mackenzie *et al*, The COBRA Project: Alleviating the Bandwidth Constraints in Large Multicomputer Networks, *Proc. Symp. Supercomputing*, Canada, 1992.

[96] M. Malumbres *et al*, An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors, *Proc. 8th IEEE Int. Symp. Par. & Dist. Proc.*, 1996.

[97] P.K. Mckinley & C. Trefftz, Efficient Broadcast in All-Port Wormhole-Routed Hypercubes, *Proc. Int. Conf. Par. Proc.*, 1993, pp. 288-291.

[98] P.K. Mckinley *et al*, Unicast-based Multicast Communication in Wormhole-Routed Networks, *IEEE Trans. Par. & Dist. Syst.*, 5(12), 1994, pp. 1254-1265.

[99] P.K. Mckinley *et al*, ComPaSS: Efficient Communication Services for Scalable Architectures, *Proc. Symp. Supercomputing*, 1992, pp. 478-487.

[100] P.K. Mckinley *et al*, Collective Communication in Wormhole-Routed Massively Parallel Computers, *IEEE Comp.*, 1995, pp. 39-50.

[101] P.M. Merlin & P.J. Shweitzer, Deadlock Avoidance in Store and Forward Networks. *IEEE Trans. Communication*, 28(3), 1980, pp. 345-354.

[102] P. Mohapatra & V. Varavithya, A Low Latency Hardware Multicast Routing Algorithm for Two-Dimensional Meshes, Tec-Rep. TR-ACAR-96-01, Dept. Elect. Comp. Eng., Iowa State Univ., 1996.

[103] P. Mohapatra, Wormhole Routing Techniques in Multicomputer Systems, to appear in the *ACM Computing Surveys*, Sept. 1998.

[104] nCUBE Systems, *N-cube Handbook*, nCUBE, 1986.

[105] nCUBE-2: nCUBE Company, nCUBE 6400 Processor Manual, 1990.

[106] nCUBE-3: http//www.ncube.com.

[107] L.M. Ni, Should Scalable Parallel Computers Support Efficient Hardware Multicast?, *Proc. ICPP Workshop on Challenges for Par. Proc.*, 1995, pp. 2-7.

[108] L.M. Ni & D.K. Panda, Sea of Interconnection Networks: What's Your Choice, A report of *the Int. Conf. Par. Proc. (ICPP)*, 1994.

[109] L.M. Ni & P.K. McKinley, A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Comp.*, Vol. 26, 1993, pp. 62-76.

[110] M. Noakes *et al*, The J-machine Multicomputer: An Architectural Evaluation, *Proc. 20th Int. Symp. Comp. Archit.*, 1993.

[111] S.F. Nugent, The iPSC/2 Direct-Connect Communication Technology, *Proc. Conf. Hypercube Concurrent Computers & Applications*, Vol. 1, 1988, pp. 51-60.

[112] M. Ould-Khaoua, Hypergraph-based Interconnection Networks for Large Multicomputers, Ph. D. Thesis, Comp. Sci. Dept., Glasgow Univ., 1994.

[113] M. Ould-Khaoua *et al*, Alleviating Channel Bandwidth Constraints through Layered Implementation, *Journal of Systems Architecture*, 44(12), Dec. 1998, pp. 837-848.

[114] M. Ould-Khaoua *et al*, Constraint-Based Evaluation of Hypergraph and Graph Networks, *Journal of Simulation Practice & Theory*, Vol. 4, 1996, pp.119-140.

[115] K. Padmanabhan, On the Tradeoff between Node Degree and Communication Channel within Shuffle Exchange Networks, *Proc. 3rd IEEE Symp. Par. & Dist. Proc.*, 1991.

[116] D. Panda, Issues in Designing Efficient and Practical Algorithms for Collective Communication on Wormhole-Routed Systems, *Proc. Int. Conf. Par. Proc. (ICPP) Workshop on Challenges for Par. Proc.*, 1995, pp. 8-15.

[117] D. Panda *et al*, Multidestination Message Passing in Wormhole *k*-ary *n*-cube Networks with Base Routing Conformed Paths, *IEEE Trans. Par. & Dist. Syst.* In Press.

[118] Paragon XP/S Product Overview. Intel Corporation, Supercomputer Systems

Division, Beaverton, Or, 1991.

[119] M.J. Pertel, A Critique of Adaptive Routing, Tec-Rep. CalTech-CS-TR-92-06, CalTech, 1992.

[120] C. Peterson *et al*, iWARP: A 100-MPOS LIW Microprocessor for Multicomputers, *IEEE Micro*, 11(13), 1991.

[121] G.F. Pfister *et al*, The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture, *Int. Conf. Par. Proc.*, 1985, pp. 764-771.

[122] Y.R. Potlapalli & D.P. Agrawal, Enhancing the Performance of HMINs Using Express Links, *Proc. Int. Conf. Par. Proc.*, 1995, pp. 56-59.

[123] S. Ramany & D. Eager, The Interaction between Virtual Channel Flow Control and Adaptive Routing in Wormhole Networks, *8th ACM Int. Conf. Supercomputing*, Manchester 1994.

[124] D.A. Reed, Cost-Performance Bounds for Multicomputer Networks, *IEEE Trans. Comp.*, C-32(1), 1984, pp. 1183-1196.

[125] J. Saltz *et al*, Performance Effects of Irregular Communication Patterns on Massively parallel Multiprocessors, *Journal of Par. & Dist. Comp.* Vol. 13, 1991, pp. 202-212.

[126] S.L. Scott & J.R. Goodman, The Impact of Pipelined Channels on *k*-ary *n*-cube Networks, *IEEE Trans. Par. & Dist. Syst.*, 5 (1), 1994, pp. 2-16.

[127] C.L. Seitz, The Cosmic Cube, *Communication of the ACM*, 28(1), 1985, pp. 22-33.

[128] C.L. Seitz, The Hypercube Communication Chip, Dep. Comp. Sci., CalTech,

Display File 5182:DF:85, 1985.

[129] H.J. Siegel *et al*, PASM: A Patitionable SIMD/MIMD System for Image Processing and Pattern Recognition, *IEEE Trans. Comp.*, C-30, 1981, pp. 934-947.

[130] H.J. Siegel & C.B. Stunkel, Trends in Parallel Machine Interconnection Networks, *IEEE Comput. Sci. & Eng.*, 1996, pp. 69-71.

[131] C.B. Stunkel, Commercially Viable MPP Networks, IBM Research report RC 20444 (4-29-96), IBM Research Division, T.J. Watson Research Center, New York, 1996.

[132] C.B. Stunkel *et al*, Architecture and Implementation of Vulcan, *Proc. Int. Par. Proc. Symp.*, 1994, pp. 268-274.

[133] C. Su and K.G. Shin, Adaptive Deadlock-free Routing in Multicomputers using One Extra Channel, *Proc. Int. Conf. Par. Proc. (ICPP)*, 1993, pp. 175-182.

[134] H. Sullivian & T.R. Bashkow, A Large Scale Homogeneous Fully Distributed Parallel Machine. *Proc. 4th Ann. Symp. Comp. Archit.*, 1977, pp. 105-117.

[135] T. Szymanski, Hyper-meshes: Optical Interconnection Networks for Parallel Processing, *Journal of Par. & Dist. Comp.* Vol. 26, 1995, pp. 1-23.

[136] T. Szymanski, A Fiber-Optic Hypermesh for SIMD/MIMD Machines, *Int. Conf. Par. Proc. (ICPP)*, 1992, pp. 710-718.

[137] T. Szymanski, O(LogN/LogLogN) Randomized Routing in Degree-logN Hypermeshes, *Int. Conf. Par. Proc. (ICPP)*, 1991, pp. 443-450.

[138] T. Szymanski, The Complexity of FFT and Related Butterfly Algorithms on

Meshes and Hypermeshes, *Int. Conf. Par. Proc. (ICPP)*, 1992.

[139] A.S. Tanenbaum, Computer Networks, Prentice-Hall Int., Inc. (2nd Ed.) 1989.

[140] N. Tanabe *et al*, Base-*m* *n*-cube: High Performance Interconnection Networks for Highly Parallel Computer PRODIGY, *Proc. Int. Conf. Par. Proc. (ICPP)*, 1991, pp. 509-516.

[141] C.D. Thompson, A Complexity Theory for VLSI. Ph. D. Thesis, Comp. Sci. Dept., Carnegie-Mellon Univ., 1980.

[142] L.D. Wittie, Communication Structures for Large Networks of Microcomputers, *IEEE Trans. Comp.*, C-30 (4), 1981, pp. 264-273.

[143] H. Xu *et al*, Efficient Implementation of Barrier Synchronisation in Wormhole-Routed Hypercube Multicomputers, *Journal of Par. & Dist. Comp.*, Vol.16, 1992, pp. 172-184.

[144] Y. Yasuda *et al*, Deadlock-free Fault Tolerant Routing in the Multidimensional Crossbar Network and its Implementation for the Hitachi SR 2201, *Proc. 11th Int. Par. Proc. Symp.*, 1997, pp. 346-352.

Publications During this work

1- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, to appear in *Computer journal* 1999.

2- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, A Performance Analysis of the Torus and Hypermesh with Fully-Adaptive Wormhole Routing, *Proc. 14th Ann. UK Perform. Eng. Workshop (PEW)*, Edinbrough (Scotland), July 1998.

3- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Modeling Adaptive Routing in Hypercubes, to appear in *Journal of Telecommunication Systems, special issue on High Performance Computing and Interconnection Networks*, June 1998.

4- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Analytical Comparison of Routing Algorithms in the Hypercube, extended abstract in *Proc. 6th Int. Conf. Telecommunication Systems: Modeling and Analysis*, Nashville (USA), Mar. 1998.

5- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, The "Express Channel" Concept in Hypermeshes and k-ary n-cubes, *Proc. 8th IEEE Symp. Par. & Dist. Proc., IEEE Computer Society Press*, New Orleans, Louisiana (USA), Oct. 1996.

6- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Adaptivity in Wormhole-

Routed Hypergraph Networks, *Proc. 10th Int. Symp. High Perform. Comp. Syst. (HPCS), IEEE Computer Society Press*, Ottawa (Canada), June 1996.

7- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Evaluation of Routing Algorithms in Wormhole-Routed Hypermeshes, *Proc. 2nd int. Conf. Massively Par. Comp. Syst., IEEE Computer Society Press*, Ischia (Italy), May 1996.

8- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Evaluation of Routing Algorithms in Wormhole-Routed Hypermeshes, Tec-Rep. TR-1996-03, Dept. Comp. Sci., Glasgow Univ., Feb. 1996.

9- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, Performance Analysis of Hypermeshes and Express k-ary n-cubes, Tec-Rep. TR-1996-13, Dept. Comp. Sci., Glasgow Univ., Apr. 1996.

10- S. Loucif, M. Ould-Khaoua and L.M. Mackenzie, An Analytical Model for Adaptive Routing in Wormhole-Routed Tori, Tec-Rep. TR-1996-14, Dept. Comp. Sci., Glasgow Univ., Apr. 1996.