The Effect of Hot Spots on the Performance of Mesh-Based Networks

Yazan M. Al-Issa

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://repository.lsu.edu/gradschool_disstheses

Recommended Citation
https://repository.lsu.edu/gradschool_disstheses/8345

This Thesis is brought to you for free and open access by the Graduate School at LSU Scholarly Repository. It has been accepted for inclusion in LSU Historical Dissertations and Theses by an authorized administrator of LSU Scholarly Repository. For more information, please contact gradetd@lsu.edu.
THE EFFECT OF HOT SPOTS ON THE PERFORMANCE
OF MESH-BASED NETWORKS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by

Yazan M. Al-Issa
B.S., United Arab Emirates University, 1996
August 1999
MANUSCRIPT THESSES

Unpublished theses submitted for the Master's and Doctor's Degrees and deposited in the Louisiana State University Libraries are available for inspection. Use of any thesis is limited by the rights of the author. Bibliographical references may be noted, but passages may not be copied unless the author has given permission. Credit must be given in subsequent written or published work.

A library which borrows this thesis for use by its clientele is expected to make sure that the borrower is aware of the above restrictions.

LOUISIANA STATE UNIVERSITY LIBRARIES
I would like to thank my advisor, Dr. Ahmed El-Amawy for his valuable comments and constant encouragement, help and guidance throughout my research work. I would like to thank Dr. El-Amawy for chairing my committee and Dr. Alexander Skavantzos and Dr. Vaidynathan for agreeing to be on my committee. I would also like to extend a special thanks to my Mom and Dad for encouraging me to follow my dreams, their efforts has been crucial to all my endeavours, academic or otherwise. I would not have been able to complete my graduate work without them. A special thanks to my friends Hamood Al-Busaidi, Naser Al-Siyabi, Alberto Fernandez, and Zaher Al-Busaidi for their continuous support during my studies.
Contents

ACKNOWLEDGEMENTS ........................................................................................................ ii

LIST OF FIGURES .............................................................................................................. v

ABSTRACT .......................................................................................................................... viii

CHAPTER
1 INTRODUCTION .............................................................................................................. 1
  1.1 Multicomputer Systems ............................................................................................. 2
  1.2 Direct Network Topologies ...................................................................................... 4
    1.2.1 Network Characteristics .................................................................................. 5
  1.3 Network Routing ....................................................................................................... 10
    1.3.1 Oblivious Routing ......................................................................................... 10
    1.3.1.1 Oblivious Path Selection .......................................................................... 10
    1.3.1.2 Oblivious Deadlock Prevention ............................................................... 11
    1.3.2 Minimal Adaptive Routing ........................................................................... 11
    1.3.3 Nonminimal Adaptive Routing .................................................................... 13
    1.3.3.1 Derouting .................................................................................................. 14
    1.3.3.2 Deadlock Prevention ............................................................................... 14
    1.3.3.3 Livelock Prevention ............................................................................... 15
  1.4 Flow Control Techniques ......................................................................................... 16
  1.5 Performance Evaluation ............................................................................................ 18
  1.6 Thesis Organization .................................................................................................. 22

2 PRELIMINARIES .............................................................................................................. 23
  2.1 The Network Model ................................................................................................. 23
  2.2 Network Routers ...................................................................................................... 24
    2.2.1 Oblivious Router .......................................................................................... 24
    2.2.2 Chaos Router ................................................................................................ 26
      2.2.2.1 Fault Tolerance ..................................................................................... 28
  2.3 Workload Models ...................................................................................................... 28
    2.3.1 Uniform Random Traffic .............................................................................. 29
    2.3.2 Hot Spot Traffic ............................................................................................ 29
    2.3.3 Other Traffic Patterns .................................................................................. 30
  2.4 The Approach ............................................................................................................ 31

iii
3 SIMULATION RESULTS .................................................................................................. 33
  3.1 Result Formats ............................................................................................................. 34
     3.1.1 Chaos Normal Form (CNF) ..................................................................................... 34
        3.1.1.1 General format of first graph ............................................................................. 35
        3.1.1.2 General format of second graph ........................................................................ 35
  3.2 Simulation Results ....................................................................................................... 35
     3.2.1 Uniform Random Traffic ....................................................................................... 35
     3.2.2 Effect of Hot Spot Number ..................................................................................... 49
     3.2.3 Effect of Hot Spot Factor ....................................................................................... 51
     3.2.4 Effect of Hot Spot location ..................................................................................... 63
     3.2.5 Effect of Network Size ............................................................................................ 69
     3.2.6 Effect of Network Topology .................................................................................... 79

4 CONCLUSION .................................................................................................................. 85

BIBLIOGRAPHY .................................................................................................................. 89

VITA ....................................................................................................................................... 93
List of Figures

1.1 A shared bus system ................................................................. 3
1.2 A typical computational node .................................................. 3
1.3 A star connected network ........................................................ 6
1.4 A four processor linear array .................................................. 6
1.5 A 2-dimensional mesh ............................................................ 7
1.6 A 3-dimensional hypercube .................................................... 7
2.1 Generic router model (Reproduced from [3]) ............................. 25
3.1 Hotspots located at the exact center of the mesh ......................... 37
3.2 Hotspots located at the exact perimeter of the mesh .................... 38
3.3 Average injection delay for a 256-node mesh using chaotic routing (Reproduced from [44]) .......................................................... 39
3.4 Throughput and latency for a 16 × 16 mesh under uniform traffic and hotspot factor of 20 located at the mesh center (fig. cont’d.) ......................... 40
3.5 Throughput and latency for a 16 × 16 mesh under uniform traffic and hotspot factor of 20 located at the mesh perimeter. ............................... 42
3.6 Throughput and latency for a 16 × 16 mesh under uniform traffic and hotspot factor of 20 located at the mesh center. ............................... 43
3.7 Throughput and latency for a 16 × 16 mesh under uniform traffic and hotspot factor of 20 located at the mesh perimeter. ............................... 44
3.8 Throughput and latency for a 16 × 16 mesh for a uniform distribution of message destinations. .......................................................... 45
3.9 Throughput and latency for a 16 × 16 mesh for a uniform distribution of message destinations. .......................................................... 46
3.10 Throughput and latency for a 16 × 16 mesh for a uniform distribution of message destinations.

3.11 Throughput and latency for a 16 × 16 mesh for a uniform distribution of message destinations.

3.12 Average message latency vs. applied load on a 16 × 16 mesh for hotspot factor of 5.

3.13 Throughput vs. applied load on a 16 × 16 mesh for hotspot factor of 5.

3.14 Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.15 Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.16 Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.17 Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.18 Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.19 Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.20 Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.21 Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.

3.22 Throughput and latency for a 16 × 16 mesh under hotspot factor of 5.

3.23 Throughput and latency for a 16 × 16 mesh under hotspot factor of 5.

3.24 Throughput and latency for a 16 × 16 mesh under hotspot factor of 10.

3.25 Throughput and latency for a 16 × 16 mesh under hotspot factor of 10.

3.26 Throughput and latency for a 16 × 16 mesh under hotspot factor of 20.

3.27 Throughput and latency for a 16 × 16 mesh under hotspot factor of 20.

3.28 Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic (fig. cont’d.).

3.29 Throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.
Abstract

Direct network performance is affected by different design parameters which include number of virtual channels, number of ports, routing algorithm, switching technique, deadlock handling technique, packet size, and buffer size. Another factor that affects network performance is the traffic pattern. In this thesis, we study the effect of hotspot traffic on system performance. Specifically, we study the effect of hotspot factor, hotspot number, and hotspot location on the performance of mesh-based networks. Simulations are run on two network topologies, both the mesh and torus. We pay more attention to meshes because they are widely used in commercial machines. Comparisons between oblivious wormhole switching and chaotic packet switching are reported. Overall packet switching proved to be more efficient in terms of throughput when compared to wormhole switching. In the case of uniform random traffic, it is shown that the differences between chaotic and oblivious routing are indistinguishable. Networks with low number of hotspots show better performance. As the number of hotspots increases network latency tends to increase. It is shown that when the hotspot factor increases, performance of packet switching is better than that of wormhole switching. It is also shown that the location of hotspots affects network performance particularly with the oblivious routers since their achieved latencies proved to be more vulnerable to changes in the hotspot location. It is also shown that the smaller the size of the network the earlier network saturation occurs. Further, it is shown that the chaos router's adaptivity is useful in this case. Finally, for tori, performance is not greatly affected by hotspot presence. This is mostly due to the symmetric nature of tori.
Chapter 1

Introduction

In the never-ending search for high computing power, parallel computers have become necessary. As the speed of logic devices reaches a limit, the maximum speed of a uniprocessor can only be in the \(10^7 - 10^8\) computations per second. One approach to increasing system throughput is to use many processors working in parallel. This is essential for many applications in areas ranging from space missions, computer vision, to real-time signal processing, to name a few. Many such applications require computational speeds in the range of \(10^{11} - 10^{15}\) computations/sec. The demand for higher performance includes more computing power, higher network and input/output bandwidth, and more memory and storage capacity. Even for applications requiring a lower computing power, parallel computers can be a cost-effective solution. Designing custom processors that boost the performance is not cost-effective. The alternative choice consists of designing parallel computers from commodity components. In these parallel computers, several processors cooperate to solve a large problem.

A challenging problem in designing large-scale parallel processing systems is how to interconnect the processors, memory and peripherals. The problem becomes more difficult as the number of components increases. Scalability is an important issue in designing multiprocessor systems, and the interconnection network must provide performance at reasonable cost [1].
The interconnection network provides the primary method of communication between the processing nodes. Known interconnection networks can be categorized into four major classes based primarily on network topology: shared-medium networks, direct networks, indirect networks, and hybrid networks. The shared medium networks are the simplest and least expensive method for interconnecting processors and memory. They also offer excellent resource scalability [2]. However, the single shared bus itself becomes the limiting factor on performance since its bandwidth is often exceeded as the number of processors is increased beyond a certain (small) integer (see Figure 1.1). Notice also that the single bus does not allow simultaneous data communications among processors. To allow all processors to send data simultaneously, each node could be directly connected to a (usually small) subset of other nodes in the network. These networks are known as direct networks. Instead of directly connecting the communicating devices, indirect networks connect those devices by means of one or more switches. Finally, hybrid approaches are possible.

1.1 Multicomputer Systems

A multicomputer consists of many nodes connected together with the help of an interconnection network. A typical node is nothing but a high performance CPU, local memory, graphics processor, an I/O server, and other supporting devices. The computational node is connected to the interconnection network through a communication co-processor. Figure 1.2 shows a typical computational node, with memory and a CPU connected via a local bus to a communication co-processor. Communication with other networked components is the communication co-processor’s job. The interconnection network consists of routers and channels. The channels connect adjacent routers, and the routers simply route messages from their sources to their destinations.

Network performance and the way messages flow through multicomputer interconnection networks is determined by many factors that can be classified into three basic categories:
Figure 1.1: A shared bus system

Figure 1.2: A typical computational node
network topology, flow control, and routing. Network topology defines how nodes are interconnected by channels. Flow control determines how and when the router switch is set. The routing algorithm determines the path selected by a packet to reach its destination. The above factors are not independent of each other but are closely related.

1.2 Direct Network Topologies

Following is a description of a representative sample of static interconnection network topologies and their properties. By no means the set described here is exhaustive:

*Star Connected Network:* Every processor in a star-connected network has a communication link connecting it to a unique central processor. The star connected network is similar to bus-based networks. Communication between any pair of processors must pass through the central processor, just as the shared bus forms the shared medium for all communication in a bus-based network. The drawback in the star topology is that the central processor becomes a communication bottleneck (see Figure 1.3).

*Linear Array and Ring:* The interconnection network where each processor in the network (except the processors at the ends) has a direct communication link to two other processors is referred to as a linear array. A linear array with a wraparound connection is called a ring (see Figure 1.4).

*Tree Network:* A tree network topology has a root node connected to a certain number of descendent nodes. Every node in a tree has a single parent node. Therefore, trees contain no cycles. Both linear arrays and star-connected networks are extensions of tree networks. The drawbacks of trees is that higher level nodes become a communication bottleneck. The bottleneck can be removed by allocating a higher channel bandwidth to channels located close to the root. This network is called a fat tree.

*Mesh Network:* The two-dimensional mesh is an extension of the linear array to two dimensions. In a two-dimensional mesh, each processor is directly connected to four other processors. Common extensions of the two-dimensional mesh include the multi-dimensional
mesh and the wraparound mesh (or torus) which is a mesh with processors at the periphery connected by wraparound connections. The majority of new generation, commercially available parallel computers are based on the mesh network (see Figure 1.5).

**Hypercube Network:** A hypercube is a special case of both n-dimensional meshes and k-ary n-cubes. A n-dimensional hypercube contains \( p = 2^n \) processors. In general, a (n+1)-dimensional hypercube can be recursively constructed by connecting the corresponding processors of two n-dimensional hypercubes. Notice that a zero-dimensional hypercube is a single processor (see Figure 1.6).

**K-ary n-cube Networks:** Here, \( k \) is the radix, which is defined as the number of processors in each dimension and \( n \) is the network dimension. In this class of networks, the network contains \( p = k^n \) processors. A ring is a p-ary 1-cube. By connecting the processors that occupy identical positions in \( k \) k-ary \((n-1)\) cubes into rings, a \( k \)-ary \( n \)-cube can be constructed.

In the remainder of this thesis, the focus will be on mesh, and torus networks.

### 1.2.1 Network Characteristics

There are several characteristics that help in evaluating networks and their performance. Of special interest when dealing with routing performance are the network connectivity, network diameter, the bisection bandwidth of the network, and the average distance traveled. There is no single network that is superior on the basis of all criteria. We must select a network on the basis of both the system’s cost and its intended applications.

*The connectivity of a network* is a measure of the multiplicity of paths between any two processors. One measure of connectivity is the minimum number of arcs that must be removed from the network to break it into two disconnected networks. This is called arc connectivity. It should be noticed that a network with high connectivity is desirable because it lowers contention for resources.
Figure 1.3: A star connected network

Figure 1.4: A four processor linear array
Figure 1.5: A 2-dimensional mesh

Figure 1.6: A 3-dimensional hypercube
Network's diameter is the maximum distance between any two processors in the network. The distance between two processors is defined as the length of the shortest path between them. Since distance can affect communication time, networks with smaller diameters are desirable. The diameter for a complete binary tree with V nodes is:

\[ \text{Diameter}_{\text{tree}} = 2 \log \left( \frac{V + 1}{2} \right) \]  \hfill (1.1)

For the 2-D mesh without wraparound connections

\[ \text{Diameter}_{\text{mesh}} = 2\sqrt{V} - 1 \]  \hfill (1.2)

For the 2-D mesh with wraparound connections

\[ \text{Diameter}_{\text{mesh}} = 2\lfloor \sqrt{V}/2 \rfloor \]  \hfill (1.3)

For the hypercube

\[ \text{Diameter}_{\text{hyp}} = \log V \]  \hfill (1.4)

and, for a 2-D torus:

\[ \text{Diameter}_{\text{torus}} = 2\left\lfloor \frac{\sqrt{V}}{2} \right\rfloor \approx \sqrt{V} \]  \hfill (1.5)

The bisection width of a network is defined as the minimum number of communication links that have to be removed to partition the network into two equal halves. The peak rate at which data can be communicated between the ends of a communication link is called channel bandwidth. Channel bandwidth is the product of channel rate and channel width. The bisection bandwidth of a network gives an idea of the network's ability to move data from one side of the network to the other. The bisection bandwidth is the product of the bisection width and the channel bandwidth.
Let $r$ be the maximum transmission rate on a single wire. Assuming that the channels are implemented as bi-directional buses of $d$ wires, the bisection bandwidth in terms of $V$, for a 2-D mesh is:

$$B_{\text{mesh}} = rd\sqrt{V} \quad \text{bits/unit time} \quad (1.6)$$

For a 2-D torus

$$B_{\text{torus}} = 2rd\sqrt{V} \quad \text{bits/unit time} \quad (1.7)$$

and for a hypercube:

$$B_{\text{hyp}} = rd\frac{V}{2} \quad \text{bits/unit time} \quad (1.8)$$

Cost is defined as the number of wires or the number of communication links required by the network. The bisection bandwidth of a network can also be used as a measure of its cost, as it provides a lower bound on the area in a two-dimensional packaging or the volume in a three-dimensional packaging.

The next metric, average distance, depends on network traffic. Random traffic, where sources and destinations are selected randomly over the entire network, is a commonly analyzed pattern. Thus, this metric is usually given in terms of this traffic pattern. The average distance, $D$, simply gives the number of network links an average message would take to cross the path from node $a$ to node $b$. When traffic is random, the average delay for messages can be computed based on the average distance traveled in the network. In general, $D$ in terms of $V$ for 2-D networks is given by:

$$D_{\text{mesh}} \approx \frac{2}{3} \sqrt{V} \quad (1.9)$$
\[ D_{torus} = \frac{1}{2} \sqrt{V} \]  

For the hypercube

\[ D_{hyp} = \frac{1}{2} \log V \]  

1.3 Network Routing

Efficient algorithms for routing a message to its destination are critical to the performance of parallel computers. Routing algorithms establish the path followed by each message or packet to travel from a source node to a destination node. Many properties of the interconnection network are determined by routing algorithms. These properties include: connectivity, adaptivity, deadlock and livelock freedom, and fault tolerance. In the following we present some general routing schemes. Two of these will be used in this thesis.

1.3.1 Oblivious Routing

In oblivious routing, the path taken by a message is a function of the destination address. In other words, the routing decision is independent of the network state. This means that, at any time, each packet has only one permissible path through a router that it has reached. This path is usually chosen to be a topologically shortest path in the network.

1.3.1.1 Oblivious Path Selection

Oblivious routing is among the most popular routing schemes because it is simple. Some topologies can be decomposed into several orthogonal dimensions. This is the case for hypercubes, meshes, and tori. In these topologies, it is easy to compute the topologically shortest path between source and destination nodes as the sum of the offsets in all the dimensions.
The oblivious router uses dimension order routing. This algorithm mandates that a message traverses the network dimensions in strictly increasing or decreasing order. For instance in a 2-D mesh, each packet is first routed in the x dimension until it reaches a router with the same x coordinates as the destination, then it is routed along the y dimension until it reaches the destination. At any node, if the needed out channel is busy, the packet must wait.

1.3.1.2 Oblivious Deadlock Prevention

In meshes, oblivious routing is known to be deadlock free [3]. For torus networks, it requires some additional resources to guarantee deadlock freedom.

Bolding states that [34] "Dimension order routing can also be shown to be deadlock-free for open-ended k-ary n-cube (those without wrap-around edges) using simple resource-ordering arguments [4]. In order to provide deadlock-freedom for dimension order routing in networks with wraparound links (torus), additional mechanisms must be provided to alleviate deadlock within a single dimension. Dally and Seitz provide a solution to this using virtual channels which employs extra buffering to remove the possibility of cyclic dependencies within a single dimension [5]. This mechanism perturbs the network's balance, though, resulting in non-uniform traffic flow through the network and degraded performance [44] [7]."

1.3.2 Minimal Adaptive Routing

A minimal adaptive router always selects one of the shortest paths between the source and the destination based on local or temporal conditions. Minimal adaptive routing schemes only supply channels that bring the packet closer to its destination with each hop taken, but the scheme can lead to congestion in parts of the network. Another advantage of minimal adaptive routing is that livelocks are avoided.

When a packet arrives at the input buffer of a minimal adaptive router, the router calculates the set of output links on shortest paths to the packet destination, called profitable
links. Then, the router checks the profitable links to see if they are available (since some links might be faulty, or congested). Faulty links are removed from the set. Following the determination of availability, the packet is sent on one of the available outgoing links that belong to the refined set. If profitable links are not available, the packet simply waits till one becomes available.

Bolding states that [34] "When more than one profitable link is available, the router must somehow choose from the alternatives. There are several alternative methods: random, dimension order, zigzag, and no-turn [8]. The first two methods are obvious in implementation. The zigzag method attempts to maximize the number of minimal paths still available at any time by choosing the dimension in which the packet has the largest displacement from its destination. Although this helps to preserve the largest set of profitable hops as the packet progresses towards its destination, its usefulness is marginal at best [9]. No-turn is based on the observation that when a packet turns or changes dimensions, it blocks packets traveling in two directions, while it would block packets only in one direction otherwise. Thus, the no-turn model chooses the link which minimizes the number of turns when possible.

A problem that arises in minimal adaptive routing is deadlock. The constraints that prevented deadlock in dimension-order routing are no longer present in adaptive routing, so other techniques must be added to ensure freedom from deadlock. Routers based on the turn model [10] restrict routing by forbidding certain turns. The Zenith router [11] and planar-adaptive router [12] require extra buffering as well as extra constraints on packet paths in order to prevent deadlock. Other routers are fully adaptive minimal, in that all minimal paths are allowed, although some amount of extra buffering is required. One disadvantage of some of these algorithms, is that different buffer capacity for different nodes in the network are required, creating non-uniformities, which disturb the network's performance.
However the main problem with minimal adaptive routers is that, in general, the number of paths available decreases as the distance to the destination decreases. Packets, which have a non-zero displacement in only one dimension, have only one path to choose from and can no longer avoid congestion or faults. Even when there is flexibility, packets near their destinations lose their ability to maneuver around congestion. This is especially evident in a result by Chinn, Leighton, and Tompa where the worst case routing time of a permutation on an N-node mesh for a class of minimal adaptive packet routing algorithms is shown to be $\Omega(N/k^2)$ when $k$ is the number of packets that can be buffered in a single node.”

1.3.3 Nonminimal Adaptive Routing

Nonminimal routers assume that taking an unprofitable link is likely to bring the packet to another set of profitable links that will allow further progress to the destination. The use of nonminimal paths helps in avoiding congestion and allows for fault-tolerance.

When minimal paths for a packet are congested, and longer paths are uncongested, it may be possible for a packet to arrive faster by taking the longer path. Also, when packets avoid a congested area, they don’t add to and worsen the congestion. Moreover, misrouting algorithms are usually proposed for fault tolerant routing because they are able to find alternative paths when all minimal paths are faulty. By allowing nonminimal paths, faults can be bypassed and communication is still possible.

Nonminimal routers might suffer from different problems such as deadlock, livelock, and starvation. Those problems are the result of the finite number of resources. The probability of reaching those situations increases with network traffic and decreases with the amount of buffer storage. Although nonminimal algorithms are flexible, they exhibit lower performance when combined with pipelined switching techniques. Degradation of performance is in the form of time lost in taking longer paths and delay caused by routing decisions complexity.

Message size affects performance considerably. For long messages that are not split into small packets, nonminial routing algorithms are not interesting because bandwidth is
wasted every time a long message reserves a nonminimal path. Note that this is not the case for short packets, as is the case in distributed shared memory multiprocessors, because misrouting is an alternative to waiting for a free minimal path and channels are reserved for a short period of time.

1.3.3.1 Derouting

Derouting is the decision to route a packet away from its destination. There are three classes of nonminimal adaptive routers: deflection routers, queuing routers, and wormhole routers.

The number of input channels is equal to the number of output channels in deflection routers or hot potato routers, [13] [14] [15] [16]. Thus, an incoming packet will always find a free output channel, and packets are always moving about. Deflection routing has two limitations. First, it requires storing the packet into the current node. Thus, it cannot be applied to wormhole switching. Second, misrouting packets increases packet latency and bandwidth consumption, and may produce livelock. Deflection routing can be used in any topology.

In queuing routers [17] [18] [19] packets move into a central buffer and wait there until a minimal path becomes available. If the central buffer becomes full, a packet is misrouted. Wormhole routers do not require the message to be broken into fixed size packets; and allow arbitrary size messages in the network making deadlock prevention more complex.

1.3.3.2 Deadlock Prevention

In a deadlock, all packets involved are blocked forever. A deadlock occurs when some packets cannot advance toward their destination because they are waiting for resources that will never be granted because they are held by other packets. In fully adaptive nonminimal routers, deadlock cannot occur through path dependencies because there are no explicit paths. However, deadlocks caused by link dependencies are still possible.
Routing algorithms based on deadlock recovery can be designed to use nonminimal paths. For networks using wormhole switching, nonminimal routing algorithms usually degrade performance because packets consume more network resources. In particular, blocked packets occupy more channels on average, reducing the bandwidth available to the remaining packets. However, blocked packets are removed from the network when VCT switching is used. In this case nonminimal adaptive routing is useful. Deadlock in VCT can be avoided by using deflection routing.

1.3.3.3 Livelock Prevention

A packet may be traveling around its destination without ever reaching it, because the channels required to do so are occupied by other packets. If there is no limit on the number of misrouting operations it is possible that livelock will occur. The simplest way to avoid livelock is to ignore it. The argument is that livelock rarely occurs, so why worry about it. Another way consists of using only minimal paths. This restriction usually increases performance in networks using wormhole switching because packets do not consume more channel bandwidth than the minimum amount required. Minimal routing algorithms are a special case of limited misrouting in which no misrouting is allowed. If there is no deadlock, the packet is guaranteed to be delivered to its destination.

Deflection routing is the only case in which misrouting is not limited without inducing deadlock. This routing technique depends on misrouting to avoid deadlocks. Limiting misrouting may produce deadlocks if the only available channel at a given node cannot be used because misrouting is limited. It has been shown that when deflection routing is used, routing is probabilistically livelock free. The probability of finding all the minimal paths busy decreases as the number of tries increases. Thus, when a sufficiently long period of time is considered, the probability of not reaching the destination approaches zero for all the packets. In practice, deflection routing requires very few misrouting operations.
An important reason to use misrouting is routing around faulty components. If only minimal paths are allowed, and the next link on the path is faulty the packet cannot advance. Given a maximum number of faults that do not disconnect the network, it has been shown that limited misrouting is enough to reach all destinations. By limiting misrouting, there is also an upper bound on the number of channels reserved by a packet, thus avoiding deadlock.

Misrouting can be limited by two basic methods: priorities, and randomization. Priority can be implemented using time stamps or battle scars. As it's name implies, time stamps [9] require that each packet be stamped with the time it was injected into the network. Whenever a deroute decision is made, the oldest packet is not chosen. Battle scar methods [13] add a field to the packet header to keep the misrouting count. Whenever a deroute decision is made, packets, which have the highest misrouting count, are less likely to be derouted. Overall, priority methods suffer from two problems: overhead and routing decision complexity.

Randomization is another solution used by chaotic routers [18]. Whenever a deroute decision is made, the router randomly chooses among queued packets. Livelock freedom is provided only in a probabilistic sense. Since livelock rarely occurs, this solution is practical.

1.4 Flow Control Techniques

The evolution of flow control techniques was naturally influenced by the need for better performance. Flow control is a synchronization protocol for transmitting and receiving a unit of information between routers and through routers in forwarding messages through the network. There are many differing forms of flow control present in network routers. Network communication can be accomplished in two basic manners: circuit switched and packet switched communication. In circuit-switching, the entire path from source to destination is reserved before communication proceeds, the path is relinquished only after the complete transmission of the message. Besides simplicity of transmission once the
path is set up, and low overhead for routing information, circuit switching is advantageous when messages are infrequent and long. Its disadvantage stems from the need to acquire the entire path for the duration of message transmission. On the other hand, in packet-switched (store-and-forward) communication, messages are partitioned and transmitted as fixed-length packets, each packet is individually routed from source to destination. A packet is completely buffered at each intermediate node before it is forwarded to the next node. Packet switching is advantageous when messages are short and frequent. It has the advantage of allowing changes in paths during transmission if necessary. Its disadvantages can be summarized in the need to receive and retransmit packets at each hop, the overhead time for packetization and reassembly, and the need to buffer packets at each local node.

In virtual cut-through (VCT) switching, rather than waiting for the entire packet to be received, and buffered at the node before it is forwarded towards its destination, the router can start forwarding the header and following data bytes as soon as the routing decision has been made and the output channel is free. In VCT, the packet is the unit of message flow control. In fact, the message does not even have to be buffered at the output and can cut through to the input of the next router before the complete packet has been received at the current router.

In wormhole switching, message packets are pipelined through the network. A packet is broken down into flits (the flit is the unit of message flow control) and input and output buffers at a router are typically large enough to store a few flits, reducing buffer sizes within the nodes. Thus, at any instant in time a blocked message occupies buffers in several routers and of course the channels in between.

The primary difference between virtual cut-through and wormhole switching is that in the latter the unit of message flow control is a single flit and, as a consequence, small buffers could be used. An entire message cannot be buffered at a router. Virtual cut-through overhead cost is relatively high as it requires enough storage at local nodes to hold
the entire packet. Virtual cut-through, however, retains the advantages of using fixed-size packets.

1.5 Performance Evaluation

The performance of an interconnection network is affected by different design parameters. In [3] the authors extensively discuss the effect of several design parameters. These parameters include: software messaging layer overhead, software support for collective communication, number of virtual channels, number of ports, routing algorithm, switching technique, deadlock handling technique, packet size, and buffer size.

- Software messaging layer. The overhead in the software messaging layer is responsible for the high percentage of communication latency in many multicomputers. Reducing or hiding this overhead is likely to have a higher impact on performance than the remaining design parameters, especially when messages are relatively short. The use of virtual circuit caching (VCC) can eliminate or hide most of the software overhead by overlapping path setup with computation, and caching and retaining virtual circuits for use by multiple messages. VCC complements the use of techniques to reduce the software overhead, like active messages. It should be noted that when software overhead is high it makes no sense to attempt improving other design parameters [20] [21] [22] [23].

- Software support for collective communication. Collective communication operations considerably benefit from using specific algorithms. When separate addressing is used, latency increases linearly with the number of participating nodes. However, when algorithms for collective communication are implemented in software, latency is considerably reduced, increasing logarithmically with the number of participating nodes [24] [25] [26].
• Number of virtual channels. Splitting each physical channel into several virtual channels increases the routing options. In wormhole switching, when no virtual channels are used, blocked messages do not allow other messages to use the bandwidth of the physical channels they are occupying. Adding the first additional virtual channel usually increases throughput considerably at the expense of a small increase in latency. On the other hand, adding more virtual channels produces much smaller increment in throughput while increasing hardware delays considerably. For deterministic routing in meshes, two virtual channels provide a good tradeoff. For tori, the partially adaptive algorithm with two virtual channels also provides a good tradeoff, achieving the advantages of channel multiplexing without increasing the number of virtual channels with respect to the deterministic algorithm. If fully adaptive routing is preferred, the minimum number of virtual channels should be used. Fully adaptive routing requires a minimum of two (three) virtual channels to avoid deadlock in meshes (tori) [29].

• Number of ports. The number of ports has a considerable influence on performance, especially when messages are sent locally. If the number of ports is too small, the network interface is likely to be a bottleneck for the network.

• Routing algorithm. The relative behavior of deterministic and adaptive routing algorithms for regular topologies and uniform traffic is similar. However, for switch-based networks with irregular topologies and uniform traffic, adaptive routing algorithms considerably improve performance over deterministic routing because of misrouting. Moreover, adaptive routing is especially interesting with nonuniform traffic, regardless of network topology. On the other hand, adaptive routing does not reduce latency when traffic is low to moderate because contention is small and base latency is the same for deterministic and fully adaptive routing. In case of using adaptive routing, the additional cost of implementing fully adaptive routing should be kept small.
Therefore, routing algorithms that require few resources to avoid deadlock or to recover from it, should be preferred [30].

- **Switching technique.** VCT and wormhole switching achieve similar latency for low traffic. For packet switching, latency is much higher. On the other hand, VCT and packet switching achieve similar throughput. This throughput is higher than that achieved by wormhole switching. Although VCT switching achieves a higher throughput, the performance difference is small when virtual channels are added to wormhole switching. Additionally, if messages are shorter than buffer size, VCT switching performs slightly better than wormhole switching. When messages are longer than buffer size, wormhole switching should be preferred. However, VCT switching is also preferable when it is not easy to avoid deadlock in wormhole switching (i.e., multicast routing in multistage networks) [27] [3].

- **Deadlock handling technique.** Current deadlock avoidance techniques allow fully adaptive routing across physical channels. However, some buffer resources (usually some virtual channels) must be dedicated to avoid deadlock by providing escape paths to messages blocking cyclically. Deadlock recovery techniques do not restrict routing at all, therefore allowing the use of the virtual channels to increase routing freedom, achieving the highest performance when packets are short. However, when packets are long and the network approaches the saturation point, the small bandwidth offered by the recovery hardware may saturate. In this case, some deadlocked packets may have to wait for long, thus degrading performance and making latency less predictable. Also, recovery techniques require efficient deadlock detection mechanisms. The poor behavior of current deadlock detection mechanisms (especially when all the packets are short and have a similar length) considerably limits the practical applicability of deadlock recovery techniques [3].
• Injection limitation mechanism. When fully adaptive routing is used, network interfaces should include some mechanism to limit the injection of new messages when the network is heavily loaded. Otherwise, increasing the applied load above the saturation point may degrade performance severely. When the start-up latency does not effectively limits the injection rate, simple mechanisms like restricting injected messages to use some predetermined virtual channels [30] or waiting until the number of free output virtual channels at a node is higher than a threshold are enough [31]. Injection limitation mechanisms are especially recommended when using routing algorithms that allow cyclic dependencies between channels. They can be used to limit the frequency of deadlock when using deadlock recovery mechanisms.

• Packet size. For pipelined switching techniques, routing a header usually takes longer than transmitting a data flit across a switch. Also, splitting and reassembling messages produces some overhead. These overheads can be amortized if packets are long enough. However, increasing packet size even more is not convenient because blocking time for some packets will be high. On the other hand, switching techniques like VCT may limit packet size, especially when packet buffers are implemented in hardware. Finally, it should be noted that this parameter only makes sense when messages are long [3].

• Buffer size. For wormhole switching and short messages, increasing buffer size above a certain threshold does not improve performance significantly. For long messages, buffer size has a more noticable impact on performance, because blocked messages occupy fewer channels. However, when messages are very long increasing buffer size only helps if buffers are deep enough to allow blocked messages to leave the source node and release some channels. In most cases the average message latency decreases when buffer size increases. However, the effect of buffer size on performance is small. For VCT switching, throughput increases considerably when moving from one to two buffers [3].
1.6 Thesis Organization

The remainder of this thesis is organized as follows, Chapter two describes the network model used, technical details concerning network routers, different traffic models, and the methodology used to obtain simulation results. This is necessary for the reader to understand and appreciate the results of our simulations. Chapter three is the heart of the thesis. It presents the simulation results which show the effect of hotspot factor, hotspots location, hotspots number, network size, and network topology on network performance. Chapter four presents the conclusions and results that can be drawn from the work presented in this thesis.
Chapter 2

Preliminaries

Before we present our simulation results, we must present the network model and routing techniques used in our study. This is necessary for the reader to gain full understanding and appreciation of the results we report in Chapter 3.

The evaluation of interconnection networks requires the definition of a network model. This is a difficult task because the behavior of the network may differ considerably from one architecture to another, and from one application to another. Moreover, in general, performance is more heavily affected by traffic conditions than by design parameters. Up to now, there has been no agreement on a set of standard traces that could be used for network evaluation. In what follows, we describe the most frequently used network model. In addition, the technical details of network routers, both the oblivious router and the chaos router are presented. This chapter also briefly discusses the different traffic models used in our results. Finally, the methodology used to obtain our results is presented.

2.1 The Network Model

In this thesis, the following network model is adopted. The network is a regular two-dimensional network. Between each pair of adjacent nodes in the network there is a channel consisting of control signals and a single bidirectional data channel. Messages are partitioned into fixed-size packets. Packets in turn consist of flow control units or flits. A flit is the
smallest unit of information that can be transmitted across a physical channel in one cycle. The first flit in a message is the header flit.

Simulations were run using the following assumptions. It takes a single clock cycle to make the routing decision, or to transfer one flit from an input buffer to an output buffer. The flit size is assumed to be equivalent to the width of the physical data channel of C bits. The header flit is assumed to be 1 flit, thus the size of an L-bit message is L+C. Also the router’s internal data paths are assumed to be equal to the channel width. Message length is kept constant and equal to 16 flits, which is consistent with many existing multicomputer designs. We consider both the two-dimensional mesh and the two-dimensional torus in our investigation. To judge changes in performance with network size, we compare networks of size 12 x 12, and 16 x 16 nodes.

2.2 Network Routers

Figure 2.1 show the structure of a generic router. We review two routers in this thesis: an oblivious router, and the chaos router. The routers have bidirectional channels. Although the channels are bidirectional, they are capable of transmitting in only one direction at a time (i.e., half duplex). The direction may change only at the end of a packet. The routers are virtual cut-through [32] routers. Therefore once the header of a packet arrives at a node, it can be routed immediately (cut-through) to a free channel before the packet has completely arrived. Packets can also cut-through the chaos multiqueue frames.

2.2.1 Oblivious Router

Oblivious routers are widely used in commercial multicomputers. Typically, oblivious routers consist of a set of input buffers connected to a set of output buffers using a crossbar switch. Each input buffer and output buffer is capable of holding exactly one fixed-size packet. Connected also to the routing and arbitration unit is the injection and ejection buffers. The oblivious router uses dimension order routing to send packets towards their
Figure 2.1: Generic router model (Reproduced from [3])
destination. If the channel for the lowest dimension needed is unavailable, the packet must wait. Virtual channels are used to prevent deadlocks [32]. In this thesis, we only consider 1 virtual channel per physical channel. The oblivious router operates in virtual cut-through fashion: instead of waiting for the complete packet to be received before any routing decision can be made, the packet header is examined as soon as it is received and the packet can cut-through to the input of the next router as soon as routing decision has been made and the output buffer is free. If the header is blocked because the output buffer is not available, the complete packet is buffered at the node. Thus, at high network loads, VCT switching behaves like packet switching. Note that only the header experiences routing delay.

2.2.2 Chaos Router

Chaotic routing is implemented in some packet-switched communication networks. Packet switching efficiently deals with computer networks in which messages of varying lengths are passed and in which some nodes are very active while others are idle or less active.

The chaos router is composed of several blocks: an input frame, an output frame, a crossbar, and a multiqueue. Each frame and multiqueue slot is capable of holding a packet. The basic operation of the chaos router is similar to a typical oblivious cut-through packet router: packets enter the router into an input frame, are connected through the crossbar to an output frame. In a chaos router, packets are routed to any output frame that brings them closer to their destination. Virtual cut-through allows packets to proceed through the router as soon as their header is received and decoded.

The chaos router differs from the oblivious router in two things. First, the chaotic router allows the routing decision to be adaptive by specifying a set of equally profitable channels. Second, it employs a multiqueue which holds packets for which no profitable channels are immediately available. By moving packets out of input frames to the multiqueue, critical channel resources are freed up so other packets can use them.
In chaos routing a packet is moved into the multiqueue if it is potentially in a deadlock loop or if it has stalled at an input frame. In order to guarantee deadlock freedom, the chaos router’s packet exchange deadlock prevention protocol defined in [33] guarantees that if routers on either side of shared link have packets to send to each other, both packets must be sent. The second situation on which a packet is moved into the queue is when it has stalled for a long time at an input frame while waiting for a profitable output frame.

The Chaos router processes packets according to the current channel dimension, which changes in a round-robin fashion to the next interesting dimension. A dimension, i, is interesting if output channel i is empty and a packet in either the multiqueue or on one of the input channels needs dimension i. When competing for free channels, packets in the multiqueue have precedence over packets in input frames. If more than one packet in the multiqueue desires the channel, FIFO priority is used. Whenever the multiqueue is full, a packet is randomly selected from the queue to be derouted to the next free output channel, making room for the new packet. Derouting of packets is the only mechanism for routing packets on nonminimal paths in the network.

Chaotic routing is a form of nonminimal adaptive routing which uses randomization to provide probabilistic protection from livelock without requiring complex protection hardware [33]. It has been shown that for all finite-sized networks the probability a packet remains undelivered after t seconds goes to zero as t increases [34] [33]. However, using randomization incurs only a small routing time penalty with respect to the oblivious router. The second use of randomization is to select one of the set of input frames that contain packets that need the current output dimension which is thought to be effective for distributing the load.
2.2.2.1 Fault Tolerance

A basic level of fault tolerance was a natural result of the chaos router design. Both network redundancy together with the non-minimal routing algorithm provide for a natural degree of fault tolerance in the chaos routing network.

Network nonuniformity caused by a failed component can create an island of congestion that affects neighboring routers. The presence of a faulty channel or router in a chaotic network, will not allow the routers to empty additional packets toward the faulty components, since packets move into an output frame only when there is space available. A router will bypass such a component, and packets will be automatically routed out on alternative minimal path. Packets which have no other profitable channels will be moved into the multiqueue. When the multiqueue fills to capacity, packets will be derouted on a random basis to the next available output channel.

Eventually, packets desiring failed channels will be derouted away from the profitable path and out on a non-profitable channel. From its new location, the packet routing decision process will be reinitiated. Packets that are within the congested area will be delayed as they work their way out. Thus, even though the minimal paths may be blocked by faults, chaotic routing allows communication to continue.

2.3 Workload Models

Network load strongly influences the relative performance of different routing algorithms. In order to evaluate the performance of a network, researchers use synthetic workloads since it is difficult to model the behavior of real applications. The choice of the workload is extremely important when trying to evaluate network performance since the performance of networks differ from one workload model to the other. The workload model is defined by three parameters: message length, distribution of destinations, and the injection rate. In
the following, we describe the most frequently used workloads: uniform random, hotspot traffic, and other traffic patterns presented in literature.

2.3.1 Uniform Random Traffic

This is the most frequently used model in the study of interconnection networks. It can be used as a benchmark to evaluate performance in the absence of more detailed information about the applications. In this model, all destinations including the source are equally likely to be chosen as destinations. In other words, the probability of a source node $a$ to send a message to a destination node $b$ is the same for all $a$ and $b$. Unlike other workload models, the uniform random traffic makes no calculations before injecting a message into the network. It is considered as an upper bound on the mean internode distance.

2.3.2 Hot Spot Traffic

Although uniform random traffic is widely used as a workload model, it is often criticized as not being representative of real workloads. Most real life applications exhibit some degree of communication locality whether spatial, temporal, or both. In hotspot traffic, certain nodes receive considerably more traffic than other nodes. Those nodes are called hotspots. Hot destinations are chosen randomly such that the amount of bias towards hot nodes is more than that oriented towards the rest of the network. In general, hotspot traffic can be used to represent cases of accessing shared variables (such as semaphores or locks) in multicomputer application. Such applications causes traffic bias towards the nodes containing shared variables.

In this thesis, whenever we refer to hotspot traffic, we mean to model a synthetic load which contains four “hot” nodes arrangement chosen at the beginning of a simulation to be located at either the network center or network perimeter. In our simulations, hotspots will be 1, 5, 10, and 20 times as likely as the other nodes to be the destination of a message.
2.3.3 Other Traffic Patterns

In their quest to model the behavior of real applications, researchers have considered several communication patterns between pairs of nodes. These patterns have been used previously in the literature. In these patterns, all nodes generate packets at the rate specified by the presented load and the destination nodes for the generated messages are always the same. These patterns can be defined as follows:

- Complement, is a permutation where each source node with binary coordinates $a_{n-1}a_{n-2}...a_1a_0$ communicates with the node whose label is the complement of the source's.

- Transpose, is a permutation where each with binary coordinates $a_{n-1}a_{n-2}...a_1a_0$ sends packets to $a_{n/2-1}a_{n/2-2}...a_0a_{n-1}a_{n-2}a_{n/2}$.

- Bit Reversal, the node with binary coordinates $a_{n-1}a_{n-2}...a_1a_0$ sends packets to the node $a_0a_1...a_{n-2}a_{n-1}$.

- Perfect shuffle, the node with binary coordinates $a_{n-1}a_{n-2}...a_1a_0$ sends packets to the node $a_{n-2}a_{n-3}...a_1a_0a_{n-1}$

- Butterfly, is a permutation where a node with binary coordinates $a_{n-1}a_{n-2}...a_1a_0$ sends packets to $a_0a_{n-2}...a_1a_{n-1}$.

The previously discussed patterns exhibit different features. The complement pattern causes packets to cross the network bisection in mesh-based topologies. Transpose and bit reversal are important because they occur in practical computations and can cause worst case behavior in hypercubic oblivious routers [39]. The shuffle converts row major indexing to shuffled row major indexing on the mesh or torus topologies. This indexing scheme can be used for efficient sorting [40].
2.4 The Approach

The performance of the networks and routers studied in this thesis are studied using a simulator written in C [35] [18] [43]. The chaos router simulator is a flit-based simulator that simulates a detailed model of $k$-ary $n$-cube networks. These include such common topologies as mesh, torus, and hypercube networks. The simulator is capable of emulating both packet and wormhole switched networks, and routes packets using either chaotic adaptive routing or dimension order oblivious routing using a variety of traffic patterns. It also handles networks of arbitrary dimensions. The sources of randomness are provided by the Learmonth-Lewis prime-modulus, multiplicative congruential generator [36] that is considered highly reliable for simulation studies [37].

It is assumed that the system satisfies the function central limit theorem. This allows the use of the batch means method [38] for computing 95% confidence intervals. The simulator runs until the statistics collected converge. Several trace files are generated during and after the simulation. Statistics are reported on input and compile time parameters which include: latency, throughput, saturation, and many other interesting aspects of the simulation.

All nodes generate packets with destinations specified by the simulated traffic pattern. The applied load is the rate at which packets are generated. All loads in this study are represented as a fraction of the network capacity for a uniform distribution of destinations, assuming that the most heavily loaded channels are located in the network bisection.

In the following we define the most important performance metrics:

- Latency, the time elapsed between the injection of the message header into the network by the source node and the reception of the last unit of information at the destination node.

- Source queuing time, the time a packet waits from its creation until it is injected into the injection frame (or queue).
• Delay, source queuing plus latency.

• Throughput, the maximum amount of information delivered per unit of time.

• Saturation, the smallest load at which the network cannot keep up with network traffic.

• Hops, number of channels traversed by a packet to reach its destination.

• Applied load, is the average load applied to the network per cycle. We use normalized applied load. The load is normalized with respect to the load that saturates the network's bisection under uniform random traffic. This load is given by $2B/N$ where $B$ is the bisection bandwidth and $N$ is the total number of nodes [3]

• Normalized throughput, network throughput divided by the applied load that saturates the network's bisection assuming uniform random traffic. The normalization is again with respect to $2B/N$ as in the applied load case above.
Chapter 3

Simulation Results

The design space of multiprocessor interconnection networks is vast with large number of possible parameters. Many researchers evaluated the performance of interconnection networks with different combinations of design parameters which include, software messaging layer overhead, software support for collective communication, number of virtual channels, number of ports, routing algorithm, switching technique, deadlock handling technique, packet size, and buffer size. Several evaluation studies compared the relative performance of different routing algorithms. Most recent studies focused on wormhole switching. Other studies reported results for packet switching, VCT switching, and scouting switching. Although most results were obtained for direct networks, indirect networks were also considered. Some studies analyzed the impact of the network topology on performance, considering implementation constraints. Furthermore, many studies analyzed the impact of virtual channels, message length, selection functions, injection limitation techniques, and deadlock avoidance techniques on performance.

The number of topologies and routing algorithms proposed in the literature is so large that it is impractical to evaluate all of them in our study. Therefore, we do not intend to present an exhaustive evaluation. Instead, we will focus on mesh-based topologies and routing algorithms. In this chapter we study the effect of hot spots on the performance of direct networks for both packet and wormhole switching. The issue of hotspots was investigated in several studies. However, none of the earlier studies was deep or broad enough to fully
cover the effects of hotspots on performance. None of them discussed the issue of changing the hotspot factor, hotspot location, or even hotspot number. Most researchers tackled this issue, only partially. Boppana and Chalasani [41] discussed the effect of a single hotspot at node (15,15) in a 16 x 16 torus on different routing algorithms. Bolding, Snyder, and Fulgham [42] discussed the effect of six and eight random arrangements of hotspots (with factors of 4 and 8) on torus and hypercube networks. Bolding [44] discussed the effect of the presence of 10 4xhotspots distributed randomly on a 256 node hypercube, mesh, and torus. This work is the most relevant to our investigation but does not address all the issues we address and is not as broad as the work reported here.

This thesis sheds more light, and studies at more depth, the effect of hotspots, their locations and the degree of reference bias to the hotspots on mesh and torus networks of moderate size. The results are compared to those of uniform random traffic. In the following, we briefly discuss the formats used to present simulation results. We discuss the advantages and disadvantages of both the Burton Normal Form (BNF) and the Chaos Normal Form (CNF).

### 3.1 Result Formats

Two standard formats are used to represent performance results. The Chaos Normal Form (CNF) requires paired accepted traffic versus applied load and latency versus applied load graphs. The other format, called BNF, uses a single latency versus accepted traffic graph. Use of only latency (including source queuing) versus applied load is discouraged because it is impossible to gain any data about performance above saturation using such graphs. In this thesis we will use the CNF format, exclusively.

#### 3.1.1 Chaos Normal Form (CNF)

CNF graphs display network latency on one graph and accepted traffic on a second graph. In both graphs, the X-axis corresponds to the applied load. By using two graphs, both
accepted traffic and latency are shown both below and after saturation, and the accepted traffic above saturation is visible. Although BNF graphs show the same data, CNF graphs are easy to comprehend.

3.1.1.1 General format of first graph

- Y-axis: Latency (not including source queuing).

3.1.1.2 General format of second graph

- Y-axis: normalized accepted traffic.

3.2 Simulation Results

The performance of a router cannot be measured without employing the router in real traffic. Barring building a multicomputer with the actual router is impractical, simulation is the best way to gauge performance. To measure performance, we concentrate on medium load latencies and high load throughput. For high loads, latency becomes less important, since the network cannot keep up with the load applied. However, latency is a more critical parameter below saturation. At low loads, all routing techniques are able to deliver the entire applied load easily. The point at which saturation occurs is the interesting point. Also, the shape of the throughput curve about saturation is important.

3.2.1 Uniform Random Traffic

To compare uniform and nonuniform traffic we use hotspot pattern with hsf of 20 as a representative nonuniform pattern. The results for $16 \times 16$ node mesh network with uniform random traffic are shown in Figures 3.4 through 3.11. In the Figures we use hsf to refer to the hotspot factor, which reflects the amount of bias towards hotspot nodes compared
to non hotspot nodes. A case with hsf=1 corresponds to uniform random traffic. We use four hotspots unless otherwise stated. Also the terms “mesh center” and “mesh perimeter” found in the figures refer to the position of the four hotspots. When we say “mesh center” we actually mean that the hotspots nodes are at the exact center of the network as shown in Figure 3.1. When we say “mesh perimeter” we mean that they are placed at the exact corners of the mesh (see Figure 3.2). Several of the figures will serve multiple purposes and will be referred to in different contexts. Unless otherwise stated, we use chaotic routing with packet switching and use oblivious routing with wormhole switching.

Results reported in Figures 3.10 and 3.11 show that when a 16×16 node mesh network is simulated, though, the differences between chaotic and oblivious routing are less apparent, both oblivious and chaotic routers saturate at nearly the same load. The throughput curves show that chaotic routing achieves slightly better throughput over the load range. Chaotic routing achieves nearly 90% throughput at full load. On the other hand, oblivious routing achieves around 78% throughput at full load. Oblivious routing achieves better latency over the load range. The performance of the chaotic router is actually lower than would be expected. The additional hardware gives no benefit under uniform random loads. The reason for the chaotic routing less than expected performance on the mesh network can be explained as follows.

In a mesh-connected network, under uniform random traffic, a large number of packets will have shortest paths which lead them through the network center. This leads to the formation of a very large permanent hot area in the network center. Figure 3.3 shows the average injection delay for a 256-node mesh with chaotic routing [44]. This diagram clearly shows that the network’s central nodes are much more congested than the nodes at the mesh periphery. Thus, any packets traveling through the network center will be delayed much more than packets that travel on the mesh periphery.

With chaotic routing, packets that travel through the central hot area will repeatedly bounce off of the wall created by the hot area, increasing their latency, and wasting band
Figure 3.1: Hotspots located at the exact center of the mesh
Figure 3.2: Hotspots located at the exact perimeter of the mesh
Figure 3.3: Average injection delay for a 256-node mesh using chaotic routing (Reproduced from [44])
width. On the other hand, with oblivious routing, packets tend to travel on non-diagonal paths which means that they will be less likely to travel through the mesh center and contribute to the large hot area seen with chaotic routing. Because of this congestion is reduced in the mesh center [45].

Compared to uniform traffic (Figures 3.4, 3.5, 3.6, and 3.7), hotspot traffic causes early saturation. Further, latencies after saturation are much higher compared to the uniform traffic case. It is clearly seen that the throughput achieved in the case of uniform traffic is higher than that achieved by nonuniform traffic. The bottom line is: with uniform random traffic, chaotic routing performance is not much different from that of oblivious routing.

![Performance of wormhole switching under different traffic patterns](image)

Figure 3.4: Throughput and latency for a 16 x 16 mesh under uniform traffic and hotspot factor of 20 located at the mesh center (fig. cont’d.)
Performance of wormhole switching under different traffic patterns

Throughput (%)

Applied load

Uniform traffic
Non uniform traffic
Figure 3.5: Throughput and latency for a $16 \times 16$ mesh under uniform traffic and hotspot factor of 20 located at the mesh perimeter.
Figure 3.6: Throughput and latency for a $16 \times 16$ mesh under uniform traffic and hotspot factor of 20 located at the mesh center.
Figure 3.7: Throughput and latency for a $16 \times 16$ mesh under uniform traffic and hotspot factor of 20 located at the mesh perimeter.
Figure 3.8: Throughput and latency for a $16 \times 16$ mesh for a uniform distribution of message destinations.
Figure 3.9: Throughput and latency for a $16 \times 16$ mesh for a uniform distribution of message destinations.
Figure 3.10: Throughput and latency for a $16 \times 16$ mesh for a uniform distribution of message destinations.
Figure 3.11: Throughput and latency for a $16 \times 16$ mesh for a uniform distribution of message destinations.
3.2.2 Effect of Hot Spot Number

Figures 3.12, and 3.13 show the effect of varying the number of hotspots on performance on a $16 \times 16$ mesh using wormhole switching. When the mesh network is simulated using hotspot traffic, as the number of hotspots increases, network performance degrades. In other words, results swing in favor of low number of hotspots.

Saturation figures are as expected. The higher the number of hotspots, the lower the load at which saturation occurs. Compared to uniform traffic, the increase in the number of hotspots causes early saturation. Latencies after saturation are much higher compared to the uniform traffic case.

Figure 3.12 shows that as the number of hotspots increases, network latency tends to increase. The presence of a large number of hotspots tends to increase the congestion, and since wormhole switching is unable to route messages around congested areas, the result is a very long path from source to destination and thus high latency. The difference in latency in this case is indistinguishable due to the network's small size and the small number of hotspots used.

Figure 3.13 shows that at low loads, throughput curves follow the same shape up to the saturation point where throughput curves start to differ depending on the number of hotspots assumed. However, at high loads the figures clearly show that the throughput doesn't differ much; actually all networks tend to achieve nearly 60% throughput after saturation. It can also be observed that in all cases the throughput doesn't degrade after saturation.
Performance of wormhole switching under hot spot factor of 5 randomly located

![Latency vs. Applied Load](image)

Figure 3.12: Average message latency vs. applied load on a 16 x 16 mesh for hotspot factor of 5.

Performance of wormhole switching under hot spot factor of 5 randomly located

![Throughput vs. Applied Load](image)

Figure 3.13: Throughput vs. applied load on a 16 x 16 mesh for hotspot factor of 5.
3.2.3 Effect of Hot Spot Factor

In this section, we study the effect of hotspot factor (hsf) on network performance. The hotspot factor is a measure of the amount of bias to the hot spot a random request would have. For that, we analyze the performance of networks using hotspot factors of 1, 5, 10, and 20. We will focus on the performance under packet and wormhole switching for different hotspot factors.

In the beginning, the saturation points for the networks under different hotspot factors are reported. Measuring the load at which the system saturates is an important issue since there is no way to predict message delivery time after saturation. The saturation point is the first applied load, using intervals of 0.1, at which the network saturates. Certain system statistics such as network delay are no longer valid beyond saturation.

Saturation figures are reported in Figures 3.14, 3.16, 3.18, and 3.20. It can be observed that the higher the hotspot factor the earlier the network saturates. This is because as the hotspot factor increases the unbalance in traffic increases creating areas of congestion around the hotspots that eventually result in premature network saturation. By examining Figures 3.18, and 3.20 we can see that saturation does not occur at the knee of the latency curves because saturation can change system behavior at any point during the load range depending on network conditions.

Figures 3.22, 3.23, 3.24, 3.25, 3.26, and 3.27 show the average packet latency versus normalized applied load for packet switching and for wormhole switching under different hotspot factors (5, 10, and 20 respectively) on a 16 x 16 mesh topology. As expected, at low loads the network achieves the same latency for low traffic and up to the saturation point. However, when traffic increases latency increases as the hotspot factor increases. In general, latencies follow the same general trends. It can be seen that oblivious routing achieves higher latency at medium load and that chaotic routing achieves higher latency at high loads and this is caused by the packet routers attempt to deroute messages around
congested areas which lead messages to wonder around the network for a while, resulting in very long paths from source to destination.

Figures 3.15, 3.17, 3.19, and 3.21 show the normalized throughput versus normalized applied load for packet switching and for wormhole switching under different hotspot factors (1, 5, 10, and 20 respectively). It can be seen that chaotic routing achieves higher throughput throughout the loading range. This is due to chaos routing’s good ability to route around local congestion, which makes up for its difficulty in handling the central network-induced mesh “hot area”. Wormhole switching saturates with a normalized throughput of 78% for hsf of 1, 53% for hsf of 5, 37% for hsf of 10, and 29% for hsf of 20. On the other hand packet switching saturates with a normalized throughput of 91% for hsf of 1 and 5, 75% for hsf of 10, and 37% for hsf of 20. After saturation, the throughput in all cases doesn’t degrade much. It can also be observed that as the hotspot factor increases the throughput decreases.

Comparisons presented earlier show that wormhole switching saturates at a lower applied load. In addition, wormhole switching achieves lower latency when compared to packet switching for all loads. On the other hand, with packet switching throughput is more than one and a half times that achieved with wormhole switching.

Performance of wormhole switching under different hot spot factors located at the mesh center

![Performance of wormhole switching under different hot spot factors](image)

Figure 3.14: Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.
Performance of wormhole switching under different hot spot factors located at the mesh center.

Figure 3.15: Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.
Performance of wormhole switching under different hot spot factors located at the mesh perimeter

Figure 3.16: Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

Performance of wormhole switching under different hot spot factors located at the mesh perimeter

Figure 3.17: Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.
Performance of packet switching under different hot spot factors located at the mesh center.

Figure 3.18: Average message latency vs. applied load on a 16 x 16 mesh under hotspot traffic.

Figure 3.19: Normalized throughput vs. applied load on a 16 x 16 mesh under hotspot traffic.
Performance of packet switching under different hot spot factors located at the mesh perimeter

![Latency vs. Applied Load](image)

Figure 3.20: Average message latency vs. applied load on a 16 × 16 mesh under hotspot traffic.

Performance of packet switching under different hot spot factors located at the mesh perimeter

![Throughput vs. Applied Load](image)

Figure 3.21: Normalized throughput vs. applied load on a 16 × 16 mesh under hotspot traffic.
Figure 3.22: Throughput and latency for a 16 × 16 mesh under hotspot factor 0f 5.
Figure 3.23: Throughput and latency for a $16 \times 16$ mesh under hotspot factor of 5.
Figure 3.24: Throughput and latency for a 16 × 16 mesh under hotspot factor of 10.
Figure 3.25: Throughput and latency for a $16 \times 16$ mesh under hotspot factor of 10.
Figure 3.26: Throughput and latency for a $16 \times 16$ mesh under hotspot factor of 20.
Figure 3.27: Throughput and latency for a $16 \times 16$ mesh under hotspot factor of 20.
3.2.4 Effect of Hot Spot location

Figures 3.8, 3.9, 3.28, 3.29, 3.30, and 3.31 compare a packet switched chaos router to an oblivious router with wormhole switching on a $16 \times 16$ mesh using two hotspot distributions: at the mesh center and at mesh corners. The hotspot patterns are chosen to study the hotspot traffic variation and are not necessarily used in practice. For hotspot traffic, the hotspot location affects performance, particularly with the oblivious router’s lack of adaptivity. See Figures 3.1, and 3.2 for the location of the hotspots and for the arrangement of the hotspots on the mesh.

Saturation figures are as expected. As can be seen, the more close the hot spot arrangement to the mesh center, the lower the load at which saturation occurs. In all cases when the hotspots are located at the mesh center, both routers tend to saturate early. This is caused by the fact that applying the hotspots near the mesh center tends to enlarge the inherently congested area at the center of mesh-connected networks which in turn creates a barrier to network traffic leading to premature saturation. It can be noticed also that saturation occurs at the knee of the latency curves.

Figures 3.10, 3.11, 3.22, 3.23, 3.24, 3.25, 3.26, and 3.27 show that unlike with uniform traffic, hotspot location affects the network saturation. It also shows that a packet switched chaos router performs better than a wormhole switched oblivious router. The chaos router saturates at higher loads than the oblivious router for all the hotspot cases. However, in all cases the difference is minor. The oblivious router’s inferior performance is the result of it’s lack of adaptivity.

Figures 3.15, 3.17, 3.19, and 3.21 show the normalized throughput versus normalized accepted traffic for packet switching and for wormhole switching under different hotspot factors (1, 5, 10, and 20). It can be seen that when the hotspot pattern is located at the mesh center wormhole switching saturates with a normalized throughput that tends to be higher for hsf of 10, doesn’t differ for hsf’s of 1, 5 and 20. On the other hand, packet
switching saturates with a normalized throughput which is nearly the same for hsf's of 1, 5, and 10, and a little bit lower for hsf of 20. After saturation, the throughput in all cases doesn't degrade as it reaches 100%.

We now revisit Figures 3.22, 3.23, 3.24, 3.25, 3.26, and 3.27. These Figures show the average packet latency versus normalized accepted traffic for packet switching and for wormhole switching under different hotspot factors (5, 10, and 20 respectively). As expected, when the hotspot pattern is located at the mesh center, network latency tends to be the same or lower. However, latency achieved by wormhole switching tends to be more vulnerable to changes in the hotspot location. Packet switching latency changes only when the traffic bias is extremely high (hsf of 20). In general, for both routers, latency follows the following trend: at low loads the oblivious router achieves lower latency up to the saturation point where packet switching starts to have a lower latency. However, when traffic increases, latency achieved by wormhole switching tends to be lower.

Previous comparisons show that when the hotspot pattern is located at the mesh center, the network tends to saturates early. On the other hand, for all hotspot pattern locations packet switching latency is higher than that of wormhole switching.

![Latency graph](image)

Figure 3.28: Average message latency vs. applied load on a 16 x 16 mesh under hotspot traffic (fig. cont’d.)
Hot spot factor = 10
Wormhole Switching

Latency (cycles)

Applied load

Hot spot factor = 20
Wormhole Switching

Latency (cycles)

Applied load
Figure 3.29: Throughput vs. applied load on a $16 \times 16$ mesh under hotspot traffic.
Figure 3.30: Average message latency vs. applied load on a $16 \times 16$ mesh under hotspot traffic.
Figure 3.31: Throughput vs. applied load on a 16 x 16 mesh under hotspot traffic.
3.2.5 Effect of Network Size

Figures 3.14, 3.20, 3.32, and 3.33, show the average packet latency versus normalized applied load for packet switching and for wormhole switching using two mesh sizes (16 x 16 and 12 x 12). The figures show that as the network size decreases, the network tends to saturate earlier. Figures 3.34, and 3.36 show that the chaos router performs better than the oblivious router. The chaos router saturates at higher loads than the oblivious router for both network sizes. This is caused by the fact that for small networks, traveling packets do not have many route options and the network gets congested quickly which creates a barrier to "cross-network" traffic leading to saturation. Comparing Figures 3.22, 3.23, 3.26, 3.27 and Figures 3.34, 3.36 and 3.38, it can be seen that latency in smaller networks is affected by the decrease in the average distance from source to destination. The increase in throughput achieved by the chaotic router is at the expense of higher latency. In general, for both routers latencies follow the three-part trend discussed earlier.

Figures 3.17, 3.19, 3.32, and 3.33 show the normalized throughput versus normalized accepted traffic for packet switching and for wormhole switching using two mesh sizes. It can be seen that when the network size decreases oblivious routing with wormhole switching saturates with a throughput that is very low; it can hardly reach 76% in case of uniform random traffic compared to 90% for chaotic routing with packet switching. Overall, chaotic routing with packet switching is not as vulnerable to changes in the network size.

For small network sizes, chaos router's adaptivity proves to be useful. For small networks, chaos routing becomes clearly better than oblivious routing. This can be seen as the oblivious router's throughput is highly affected by the decrease in network size while the chaos router's throughput remains stable (around 90%). For large networks the oblivious router's throughput is higher than that of smaller networks. For small networks the four hotspots influence the traffic greatly and the chaos router performs better, but as the network grows, the central hot area influence dominates the traffic flow and the oblivious
router's performance gets better. In other words, the influence of the inherently hot area at the mesh center becomes more profound in larger networks.

The throughput achieved by chaotic routing is a bit lower when increasing network size to $16 \times 16$ nodes. The reason for this is the fact that the influence of the large inherently hot area at the mesh center exceeds that of the four smaller hotspot arrangement. Another reason is that the effect of four hot spots on a $12 \times 12$ node network is larger than that in a $16 \times 16$ node network. For small networks under hotspot traffic, chaos routing becomes distinctly better than oblivious routing.

Previous comparisons show that for small size networks, chaotic routing performance is better than that of oblivious routing in terms of the point at which the network saturates, as well as achieved throughput.

Figure 3.32: Throughput and latency for a $12 \times 12$ mesh under hotspot traffic (fig. cont’d.)
Performance of wormhole switching under different hot spot factors located at the mesh center

Throughput (%)

Applied load
Performance of packet switching under different hot spot factors located at the mesh perimeter.

Figure 3.33: Throughput and latency for a $12 \times 12$ mesh under hotspot traffic.
Figure 3.34: Average message latency vs. applied load on a 12 × 12 mesh for hotspot factor of 5.
Figure 3.35: Throughput vs. applied load on a $12 \times 12$ mesh for hotspot factor of 5.
Figure 3.36: Average message latency vs. applied load on a $12 \times 12$ mesh for hotspot factor of 10.
Figure 3.37: Throughput vs. applied load on a $12 \times 12$ mesh for hotspot factor of 10.
Figure 3.38: Average message latency vs. applied load on a $12 \times 12$ mesh for hotspot factor of 20.
Figure 3.39: Throughput vs. applied load on a $12 \times 12$ mesh for hotspot factor of 20.
3.2.6 Effect of Network Topology

It is interesting to compare the performance of a mesh network to that of a torus network to see if introducing different hotspot arrangements on the torus affects the performance of oblivious and chaotic routing. On torus networks, chaos routers perform distinctly better than the oblivious routers in all aspects. Unlike chaotic routing, the introduction of four hot spots has no effect on oblivious routing, see Figures 3.41, 3.42, and 3.43.

Unlike the mesh, wraparound connections make the torus node-symmetric. A torus network appears the same to every node, and traffic is evenly distributed throughout the network. This uniformity in traffic distribution helps in reducing the effect of the induced hot area at the mesh center, since it allows messages to use the entire network without the constraints introduced by routing techniques. As can be seen in Figures 3.42, and 3.43, chaotic routing performs much better than oblivious routing on a torus network. At the same time, when compared to the mesh network, the oblivious router’s performance is degraded.

Oblivious routing’s poor performance is caused by non-uniformities introduced by the deadlock-prevention mechanism. Because different nodes on the network appear to have different amounts of buffering under this scheme, the performance of each oblivious node is dependent upon its location in the network. Because of this, drawbacks similar to the mesh’s “hotspot” are seen when using a torus oblivious router [44].

For the chaos and oblivious routers on torus networks, there is distinguishable differences in the points at which saturation occurs. In a torus, oblivious routers saturate earlier than chaotic routers. Compared to the mesh, packet router’s saturation points do not differ much, whereas oblivious routers tend to saturate a bit earlier in a torus.

Figures 3.41, 3.42, and 3.43 show that the chaos router can achieve high throughput (58%-96%) under hotspot traffic for all torus networks studied. On the other hand, oblivious routers can only achieve 40% throughput. Compared to the mesh, oblivious router’s
throughput degrades by 15% to 30%, whereas chaotic routers tend to have a higher throughput (up to 96%). The chaotic router's throughput degrades in case of the torus only if the hotspot factor is significantly increased. Again, the latencies remain low for low to medium loads, indicating very good performance. The behavior of oblivious routing on a torus can be explained as stated in [44] "A disturbing property of the torus oblivious router is that the maximum throughput is achieved at less than the maximum load. This is due to disturbance of the vertex-transitivity of the network introduced by the addition of deadlock prevention. Since the virtual-channel deadlock prevention scheme applied [30] distinguishes certain nodes as "special" in order to break cycles, the uniformity of the network is broken and hotspots are introduced at high loads. This results in the degradation of throughput as load is increased. The chaos router preserves the uniformity of the network and does not exhibit this behavior."

Figure 3.40: Average message latency vs. applied load on a 16 x 16 torus under hotspot traffic (fig. cont’d.)
Torus perimeter

Applied load

Latency (cycles)

Applied load

packet ———
wormhole ————

HSF=1
Figure 3.41: Normalized throughput vs. applied load on a $16 \times 16$ torus under hotspot traffic.
Figure 3.42: Throughput and latency for a $16 \times 16$ torus under hotspot traffic.
Figure 3.43: Throughput and latency for a 16 × 16 torus under hotspot traffic.
Chapter 4

Conclusion

We have reported the most comprehensive study on the effect of hotspots on network performance. This far, we have studied, thoroughly, the effects of hotspot factor, location, and number on the performance of mesh-based systems. We have done so far both wormhole switched networks with oblivious routing and packet switched networks with chaotic routing.

In case of uniform random traffic, both oblivious and chaotic routers saturate at nearly the same load. Oblivious routing achieves better latency over the load range. The throughput curves show that oblivious routing achieves slightly better throughput over the load range. It is shown that the differences between chaotic and oblivious routing are indistinguishable. Actually, chaotic router performance is lower than expected taking into consideration the additional hardware which seems to be of no benefit under uniform random loads because of the inherently hot area at the mesh center, and because of the small size of the network we are simulating. Compared to nonuniform traffic, hotspot traffic saturates the network earlier. With hotspots, latencies after saturation are much higher compared to the uniform traffic case. It is also clearly shown that the throughput achieved in the case of uniform random traffic is higher than that achieved by nonuniform traffic. Overall, with uniform random traffic, there is no difference between the performance of chaotic routing when compared to that of oblivious routing.

When simulating the mesh network using hotspot traffic, networks with low number of hotspots have better performance. In other words, results swing in favor of low number
of hotspots. The higher the number of hotspots, the lower the load at which saturation occurs. Compared to uniform traffic, the increase in the number of hotspots causes early saturation. Further, latencies and delays after saturation are higher compared to those in the case of uniform random traffic. As the number of hotspots increases, network latency tends to increase as well. The difference in latency in this case is indistinguishable due to the network small size and the small number of hotspots used. Throughput is the same up to the saturation point where the throughput achieved starts to vary depending on the number of hotspots assumed. At high loads the throughput doesn’t differ much.

The higher the hotspot factor, the earlier the network saturates. As the hotspot factor increases, traffic moving toward the hotspots increases creating areas of congestion that eventually result in premature saturation of the network. Performance of packet switching is better than that of wormhole switching. Packet switching saturates at a higher loads than wormhole switching. It can be seen that wormhole switching achieves better latency at medium loads and that packet switching achieves higher latency at high loads. This is due to the ability of packet routers to deroute packets which makes up for its difficulty in handling the inherently congested area at the mesh center. For all loads simulated, packet switching achieves higher latency at high loads than wormhole switching. On the other hand, packet switching’s throughput is higher than that achieved by wormhole switching.

The location of hotspots affects network performance, particularly with the oblivious routers which cannot deroute messages around congested areas. Unlike uniform traffic, hotspot location affects network saturation. The closer the hotspot nodes to the mesh center, the earlier saturation occurs. For all the hotspot cases a packet switched chaos router performs better than a wormhole switched oblivious router. The chaos router saturates at higher loads than the oblivious router. Latency achieved by wormhole switching tends to be more vulnerable to changes in the hotspot location. Packet switching latency changes only when it is under the influence of extremely high traffic. On the other hand, for all hotspot pattern locations packet switching latency is higher than that of wormhole
switching. Overall, throughput of both switching techniques doesn't show much change due to change in hotspot location.

The smaller the size of the network, the earlier network saturation occurs. For small size networks, the chaos router's adaptivity proves to be useful. Under hot spot traffic, chaos routing becomes distinctly better than oblivious routing for smaller networks. The chaos router with packet switching saturates after the oblivious router with wormhole switching for all network sizes. This is due to the fact that as the network becomes smaller, the network gets congested earlier leading to saturation. The oblivious router's throughput is highly affected by decreases in network size while the chaos router's throughput doesn't show much change. When the network size decreases oblivious routing with wormhole switching saturates with a normalized throughput that is very low compared to that of larger networks. It is also worth mentioning that latency of both routers is influenced by the decrease in the distance from source to destination in smaller networks.

For the chaos and oblivious routers on torus networks, oblivious routers tend to saturate early. Compared to the mesh, packet router's saturation points do not differ much, whereas oblivious routers tend to saturate a bit earlier. On a torus, chaotic routing performs distinctly better than oblivious routing. The chaos router achieves higher throughput when compared to that achieved by oblivious routers. At the same time, when compared to the mesh network, oblivious router's throughput degrades, whereas chaotic routers tend to have a higher throughput. The chaotic router's throughput degrades in case of the torus only if the hotspot factor is significantly high. Unlike chaotic routing, the introduction of hotspots has no effect on oblivious routing. Latencies remain almost constant up to medium loads, indicating very good performance.

It has been shown throughout the thesis that the number of hotspots, hotspots location, and hotspot factor have significant impact on the performance of direct networks. For that reason they need to be taken into consideration when scheduling tasks. We recommend the use of switching techniques that are adaptive in nature and have built-in deadlock and
livelock prevention mechanisms. Attention should be paid when scheduling tasks so as not to congest the processors at the middle of the network. This can be done by taking this point into consideration while writing the scheduling algorithms. Specifically, shared variables and synchronization mechanisms should deliberately be assigned to nodes near the periphery of the topology.
Bibliography


Yazan Alissa was born in Irbid, Jordan, in December of 1971, and grew up in the United Arab Emirates. He received his bachelor of science degree in Electrical and Computer Engineering from United Arab Emirates University in June 1996. During his undergraduate studies, he became interested in the field of parallel processing. Following graduation, he began pursuing a master of science degree in the Department of Electrical and Computer Engineering at Louisiana State University in the area of parallel processing, with emphasis on the performance of direct networks under different traffic patterns. Presently, he is a candidate for the degree of Master of Science in Electrical and Computer Engineering.
Candidate: Yazan Moh'd Al-Issa

Major Field: Electrical Engineering

Title of Thesis: The Effect of Hot Spots on the Performance of Mesh-Based Networks

Approved:

[Signatures]

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

[Signatures]

Date of Examination:

July 6, 1999