MIT/LCS/TR-382

CONGESTION CONTROL IN ROUTING NETWORKS

ANDREW ANDAI CHIEN

NOVEMBER 1986

MIT/LCS/TR-382 CHIEN CONGESTION IN ROUTING NETWORKS

This blank page was inserted to preserve pagination.

CONGESTION CONTROL IN ROUTING NETWORKS

by

Andrew Andai Chien

S.B., Massachusetts Institute of Technology
(1984)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS OF THE
DEGREE OF

MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
October 1986

©Andrew Andai Chien 1986

The author hereby grants to M.I.T. permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author Andrew A. Chien
Department of Electrical Engineering and Computer Science
October 21, 1986

Certified by Arvind 10/21/86
Dr. Arvind
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by _____
Dr. Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

CONGESTION CONTROL IN ROUTING NETWORKS

by

Andrew Andai Chien

Submitted to the
Department of Electrical Engineering and Computer Science
on October 21, 1986 in partial fulfillment of
the requirements for the Degree of Master of Science in
Electrical Engineering and Computer Science

Abstract

Multistage routing networks present an attractive cost-effective method of interconnection for medium to large scale multiprocessors. Recent results concerning performance degradation in the presence of "hot spots" have raised serious questions about the robustness of previous performance estimates for these routing networks. Research to date has focused on a limited class of hot spots – those in which all the hot spot traffic is destined for the same memory address.

We consider a more general kind of traffic imbalance – the hot spot traffic may be of arbitrary composition. By taking this more general view, we hope to understand a wider class of traffic imbalances. In this thesis, we define an analytic framework in which to study the problem of performance degradation due to "hot spots." We characterize the performance degradation due to these hot spots. This degradation is very severe. We then employ approximate methods to estimate the time to congest the network, and the time to dissipate that congestion. These approximations are validated by extensive simulation of the model.

We subsequently propose a solution to prevent performance degradation due to "hot spot" traffic imbalances. The effectiveness of this solution depends crucially on the traffic model one considers. In our studies, we assume that the traffic load is completely inelastic. This assumption is not verified. The proposed solution involves two primary mechanisms – misrouting and throttling the congesting traffic. Simulations results show that the proposed scheme is very effective in maintaining good network performance in the presence of "hot spots." Network throughput and delay are maintained at near balanced load levels. The simplicity of the scheme and a few key simulation statistics indicate that implementation of the proposed scheme should require only minimal hardware and limited run time communication.

Thesis Supervisor: Dr. Arvind
Title: Associate Professor of Electrical Engineering and Computer Science

Keywords: Multistage Routing Networks, Hot Spots, Crossbar, Dataflow, Multiprocessors, Tagged Token Dataflow Architecture, Von Neumann Architecture, Throughput, Delay.

Acknowledgements

I am grateful to many people who have made the completion of this thesis possible. I thank my thesis advisor, Arvind, whose patience and thoughtful criticism have vastly improved this thesis.

I would also like to thank Gino Maa, whose patience in listening to many half baked ideas helped the development of my ideas immeasurably. My thanks to Ken Traub, whose good natured prodding often prevented my lethargy from getting the better of me. Andy Boughton helped me to get started on this problem, and waded through an early version of this thesis.

I would like to thank Greg Papadopoulos and Bob Iannucci whose insightful comments were invaluable aid to the formulation of my ideas. Thanks also to Staffan Truve, Serge Plotkin, Stephen Brobst, and Behrouz Vafa for making the long hours in Tech Square enjoyable. My thanks to Mike Mack, Natalie Tarbet, Susan Hardy, and all of the members of CSG and PPG that make the second floor such a lively and interesting place to work.

I owe a great debt of thanks to my family. My parents have encouraged and supported me from the start. I am especially grateful to my father, who has always been an inspiration to me. I am thankful that he has imparted to me a little of the energy and excitement that he brought to his research. My mother has always given me what I needed. Encouragement when I was discouraged, motivation when I was indolent, and support through it all. I thank her for her unwavering faith in me. My thanks to my brothers and sister, Emily, Tony and Steve for their love, support, and kindly distraction.

My special friend, Ellen, I thank for her support and love. Her encouragement and good humor kept my spirits up through it all.

Most of all, I give thanks to God for blessing me with the ability, perseverance, and opportunity to the complete this thesis.

This report describes research done at the Laboratory for Computer Science of the Massachusetts Institute of Technology. Funding for this project is provided in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-84-K-0099. The author was supported in part by the Burroughs Corporation.

Contents

1 The Hot Spot Problem 1
- 1.1 Introduction . . . 1
- 1.2 Routing Networks and Hot Spots . . . 2
- 1.3 How Hot Spots Arise . . . 4
  - 1.3.1 Synchronization Hot Spots . . . 6
  - 1.3.2 Run Time Constant Hot Spots . . . 7
  - 1.3.3 Random Hot Spots . . . 8
  - 1.3.4 Locality Hot Spots . . . 8
- 1.4 Summary of the Problem . . . 9
- 1.5 Overview of the Thesis . . . 9

2 The Analytic Framework 12
- 2.1 The Network Model . . . 13
  - 2.1.1 Definition of Symbols . . . 14
- 2.2 Performance of Routing Networks . . . 17
  - 2.2.1 Traffic Terminology . . . 18
- 2.3 Steady State Network Behavior in the Presence of Hot Spots . . . 21
- 2.4 Simulation of Hot Spots . . . 23
  - 2.4.1 Simulation Framework . . . 23
  - 2.4.2 Simulation Results . . . 24
- 2.5 Delay Degradation . . . 25
  - 2.5.1 Simulation Results . . . 27
- 2.6 Summary . . . 28

3 Transient Behavior of Networks in the Presence of Hot Spots 30
- 3.1 The Number of Hot Spot Packets . . . 31
  - 3.1.1 The Fraction of Hot Packets . . . 31
  - 3.1.2 The Number of Hot Spot Packets . . . 34
- 3.2 Approximations to the Congestion Time . . . 35
  - 3.2.1 Overall Congestion Bound . . . 36
  - 3.2.2 Stage by Stage Congestion Bound . . . 37
  - 3.2.3 Comparison to Simulation Results . . . 39

3.3 Dissipation Bounds 42
3.3.1 A Simple Approximation for the Dissipation Time 42
3.3.2 Stage by Stage Dissipation Bound 44
3.4 Comparison to Simulation Results 45
3.5 Summary 46

4 Controlling Hot Spot Degradation **49**
4.1 The Basic Problem 49
4.2 Flow Control in Data Networks 50
4.3 Traffic in a Routing Network 51
4.4 Excess Traffic Assumption 51
4.4.1 A Scenario for No Excess Traffic 51
4.4.2 Two Scenarios for Excess Traffic 52
4.5 Traffic Elasticity 52
4.5.1 Elastic Traffic 53
4.5.2 Inelastic Traffic 54
4.6 A Traffic Model 55
4.6.1 Inelastic Traffic Model 55
4.7 A Mechanism for Controlling Hot Spot Degradation 56
4.7.1 Long Term Rate Matching 56
4.7.2 Short Term Unblocking 57
4.8 Summary 58

5 Simulation Results **60**
5.1 A Detection Mechanism 61
5.2 Steady State Performance of the System 62
5.3 A Case Study in Controlling Hot Spot Degradation 64
5.3.1 System Performance without Flow Control 64
5.3.2 System Performance with Flow Control 69
5.4 Overall System Performance Experiments 76
5.4.1 Motivation and Structure of Experiments 76
5.4.2 Results 77
5.5 Hot Spot Sink Throughput 79
5.6 Summary of Results 82

6 Conclusions **85**
6.1 Summary 85
6.2 Applications of Results 86
6.3 Future Research 88

A Simulation Data **90**

List of Figures

1.1 An 8-input, 8-output Butterfly Network. 3
1.2 Tree Buffering in an 8-port Butterfly Network. 5

2.1 A 2-by-2 Router. 14
2.2 An 8 Port Indirect-N-Cube Network. 16

3.1 Three Congestion Bounds along with Simulation Results. 41
3.2 Two Dissipation Bounds along with Simulation Results. 47

5.1 Steady State Throughput with a 16% Hot Spot 63
5.2 Steady State Delay with a 16% Hot Spot 65
5.3 Throughput of an Uncontrolled System with a 16% Hot Spot 67
5.4 Delay in an Uncontrolled System with a 16% Hot Spot 68
5.5 Throughput of a Flow Controlled System with a 16% Hot Spot 70
5.6 Delay in a Flow Controlled System with a 16% Hot Spot 72
5.7 Misrouted Packets in a Flow Controlled System with a 16% Hot Spot . . . 73
5.8 Warnings in a 64-port System with a 16% Hot Spot 74
5.9 Throughputs of Different Packet Sinks 75
5.10 System Throughput vs. Hot Spot Duration, a 64-port Network 78
5.11 Average Delay vs. Hot Spot Duration, a 64-port Network 80

List of Tables

2.1 Sustainable Average Throughputs for Various Networks (normalized) 18
2.2 Throughput Degradation with 16% Hot Spot 25
2.3 Average Delay with 16% Hot Spot 28

3.1 Congestion Times with 16% Hot Spot 40
3.2 Dissipation Bounds with 16% Hot Spot 46

5.1 Hot Spot Sink Utilizations with Short Hot Spots 83

A.1 Steady State Throughput and Delay in Controlled System – 16% Hot Spot. 90
A.2 Normalized System Throughput with Short Duration Hot Spots 91
A.3 System Average Delay with Short Duration Hot Spots 92

Chapter 1

The Hot Spot Problem

1.1 Introduction

In high speed routing networks, slight imbalances in the traffic distribution can have dramatic effects on network performance [18]. One particularly important type of imbalance – a "hot spot" – was discovered recently by Pfister and Norton [17]. Variations of this type of imbalance were further studied by Lee [13]. When a "hot spot" occurs, a particular communication link experiences a much greater number of requests than the rest of the links – many more than it can service. In a remarkably short period of time, the entire network may become congested. When congestion occurs, the network throughput decreases precipitously. In addition, the average transit time for packets increases dramatically. Because network performance is a key factor in limiting multiprocessor performance, such network performance degradation is of fundamental interest. Hot spots are particularly insidious because they may result from the cumulative effects of very small source imbalances. Such imbalances can be caused by a variety of mechanisms – a popular data structure, a synchronization lock, or any other type of fluctuation in network traffic.

1.2 Routing Networks and Hot Spots

If the processors in a multiprocessor are to cooperate efficiently in the solution of a wide variety of problems, they must be able to communicate and share data with little overhead. It is clear that central bus based systems cannot provide the scalable communication bandwidth required by large scale multiprocessors. At the other end of the spectrum, full crossbar interconnection is not a viable alternative because of the intolerably large hardware cost, $O(n^2)$ components, and serious questions about fault tolerance. In view of these considerations, an alternative interconnection scheme – the multistage routing network – has received considerable attention. These networks may be able to provide the scalable bandwidth and fault tolerance properties essential in large scale multiprocessor systems.

Routing networks reduce the hardware cost of interconnection (requiring only $O(nlog\ n)$ components) by sharing buffers and communication links within the network. For example, consider the well known butterfly routing network shown in Figure 1.1. In this picture, we see that all of the internal network links (in bold) are shared by several paths. The same is true for all of the network's internal buffers. This sharing causes little contention in the networks if the traffic load is balanced [15]. However, if the traffic distribution is uneven, radically lower performance can result [13,17,18]. The performance degradation is a direct result of the sharing of links and buffers within the communication network. A consequence of this widespread sharing is that when a hot spot occurs, all of the network inputs may experience throughput degradation and increased delay.

When a "hot spot" occurs, the entry rate of traffic requiring the use of a particular link is elevated beyond the link's maximum capacity. The popular link or network output port is generally referred to as a "hot spot". When the demand on this link exceeds its bandwidth, the traffic begins to back up into the network. In buffered networks, a set of buffers will fill. This set forms a tree of all of the buffers that can send packets to the saturated link. The buffer filling effect has been termed "tree saturation" [17] or "tree buffering" [3]. This effect is illustrated in Figure 1.2. Here we see that effects of a hot spot at Output 0. A tree

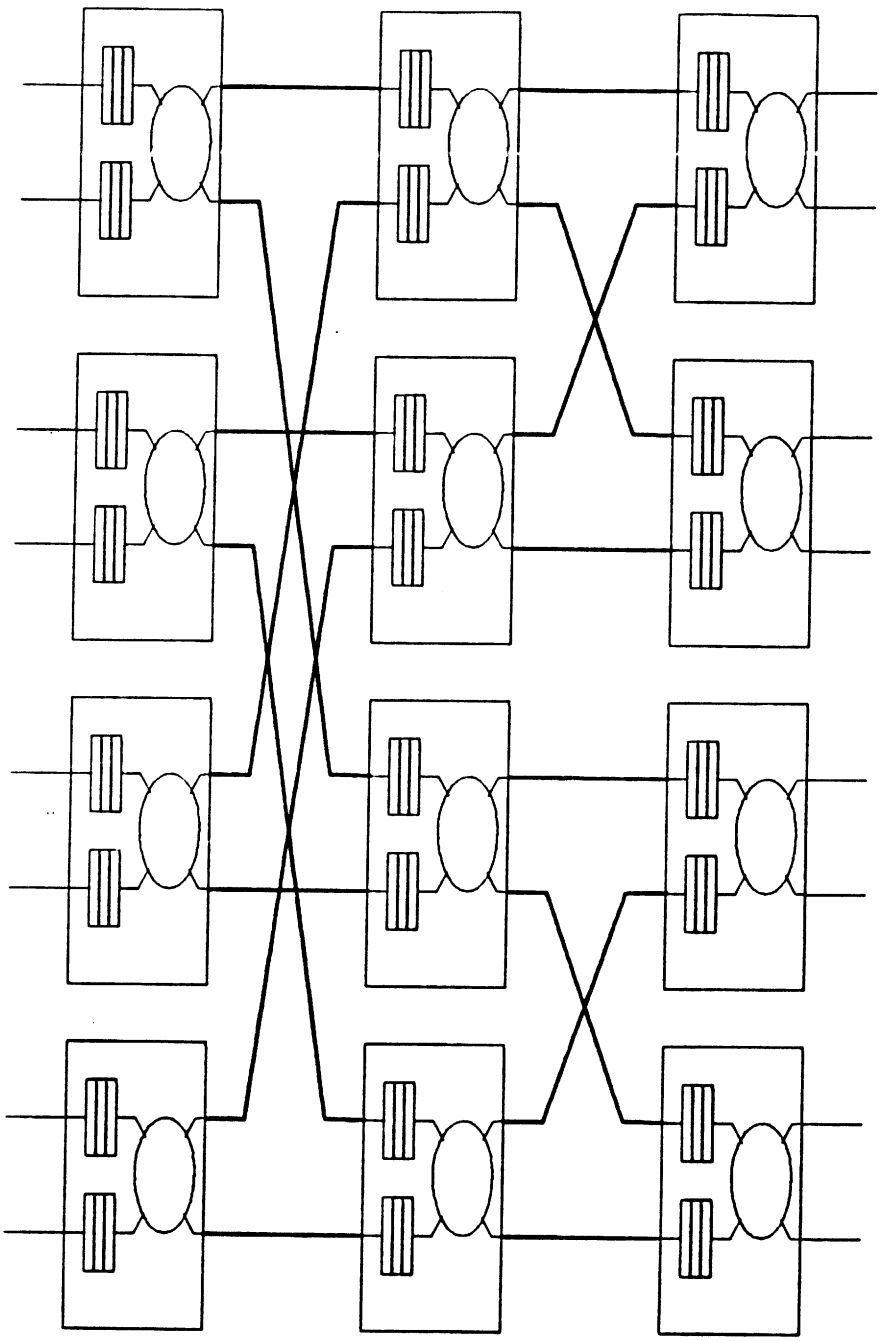

Figure 1.1: An 8-input, 8-output Butterfly Network.

of buffers has been filled leading back to all of the network inputs. These saturated buffers are shown shaded.

We make several observations about tree buffering due to hot spots. First, the tree of saturated buffers will lead back to all sources contributing to the hot spot.[1] Second, because the internal network buffers are shared, traffic not related to the hot spot is blocked by the slow moving hot spot traffic. Thus all traffic that needs to cross the tree of saturating buffers will be slowed. This results in a dramatic increase in network transit time. Third, the rate at which traffic moves in the saturating tree is determined by the maximum rate at which the hot spot can accept packets. Since most of the network traffic will have to cross the tree, the overall network throughput will also depend on the maximum acceptance rate at the hot spot. This is disastrous because the throughput of our entire system is dependent on the bandwidth of a single link – the speed of the underlying technology. In the presence of a hot spot, we have lost all scalability properties for the entire multiprocessor system.[2]

1.3 How Hot Spots Arise

The remarkable thing about the hot spot problem is the small magnitude of imbalances required to cause significant degradation in network throughput. The capacity of a popular output need only be slightly exceeded before such effects begin to occur. In a multistage routing network constructed from buffered routers, this could be as little as two times the average load.[3] In other words, if all of the nodes sent $\frac{2}{n}$ of their traffic to the popular port instead of the uniform $\frac{1}{n}$, we would see "hot spot" degradation. This kind of small traffic imbalance can occur often because of the bursty nature of program communication and data requirements.

[1] Any source exceeding its $\frac{1}{n}$ share of the sink bandwidth.

[2] It is also interesting to note that any decrease in network throughput will cause the effective delay experienced by network traffic to further increase. This is due to the fact that packets will have to wait before they are able to gain access to the network.

[3] This scenario assumes a network of 2-by-2 routers with 5 packets of buffering at each stage operating at full capacity. Arbitration for links is assumed to be instantaneous.

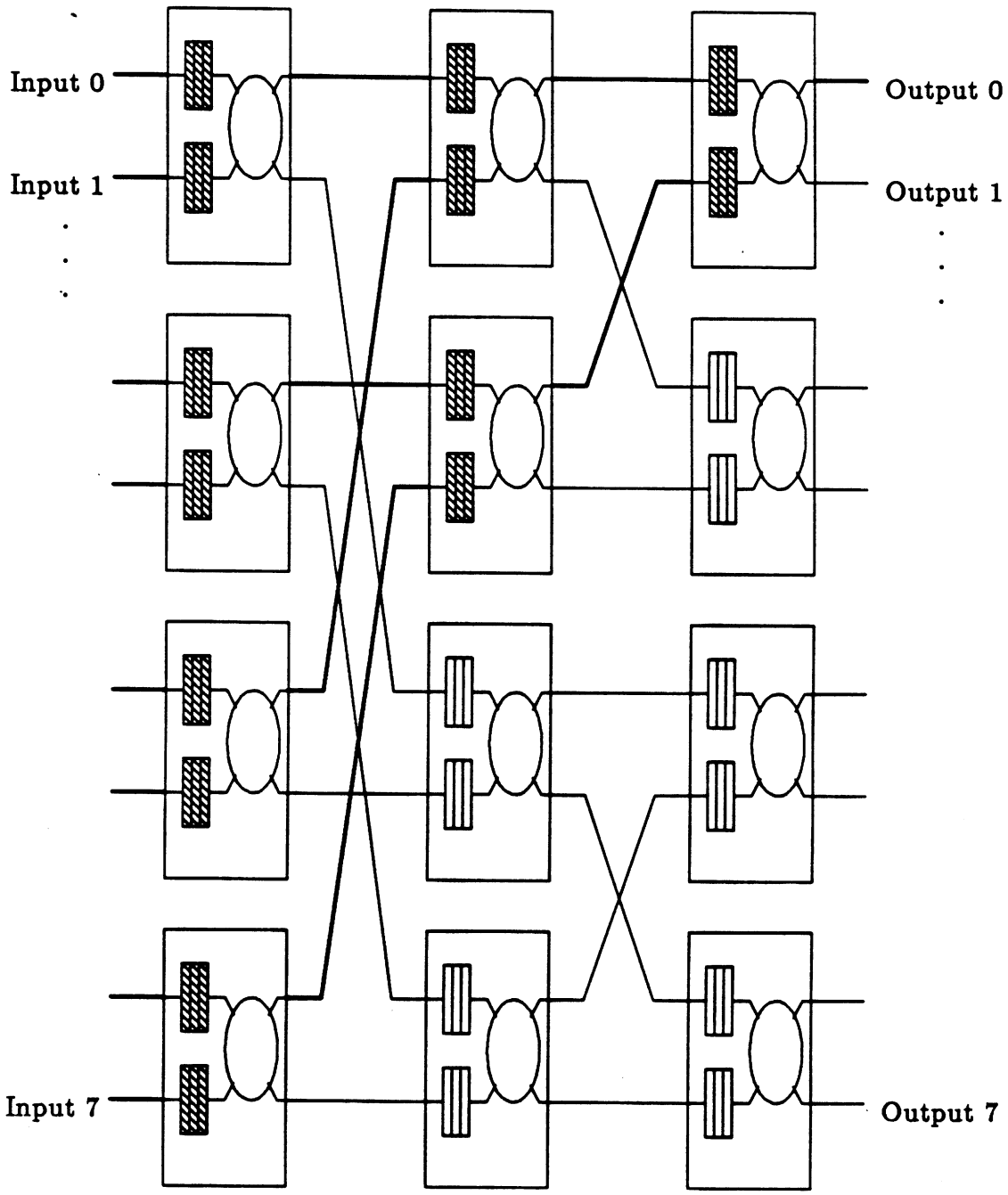

Figure 1.2: Tree Buffering in an 8-port Butterfly Network.

Pfister and Norton have examined one type of hot spot. In their model, hot spots result from synchronization references to a single memory location. The particular structure of these hot spots (homogeneous address mix) makes them amenable to solution by "Combining." The basic idea behind "Combining" is to combine several messages locally to decrease the load on the message destination.

When a response is received from the message destination, appropriate responses can be constructed locally and sent to the requesters. "Combining" was originally proposed as a mechanism for approximating the "Paracomputer" model of computation [9]. Combining reduces the number of requests, and hence could theoretically increase the effective bandwidth of any single memory location. When Pfister and Norton subsequently drew our attention to hot spot imbalances, they observed that combining is very effective in controlling the degradation due to these synchronization hot spots [17]. It is important to observe that combining will not be effective if the hot spot traffic does not have a homogeneous address mix.

In this thesis, we consider a more general class of traffic imbalances. We consider hot spots that arise by a variety of mechanisms. Consequently, these hot spots are likely differ widely in duration and intensity as well as composition. Some phenomena that may give rise to hot spots are system synchronization or work distribution (system management functions), run time constants, attempts at enhancing locality, or statistical fluctuations in traffic. In each of these cases, the resulting hot spots have different characteristics. We describe some of the different ways that hot spots may arise and detail their salient characteristics.

1.3.1 Synchronization Hot Spots

In the NYU Ultracomputer and the IBM RP3 [9,16], hot spots can be produced by global synchronization. These systems use "Fetch-and-Add" style operations on a shared memory location in order to synchronize and distribute work throughout the machine. If many pro-

cessors perform the synchronization at the same time, the resulting traffic may cause a hot spot. We will refer to this type of hot spot as a "synchronization" hot spot. Synchronization hot spots are characterized by combinable traffic and traffic demand proportional to the size of the machine. Synchronization hot spots may be of long or short duration, depending on how much skew is allowed to develop between processors.

A similar variety of hot spot is due to all kinds of centralized system management. In many systems, these management functions are implemented using "Fetch-and-Add" or "Test-and-Set" style operations. In that case, system management hot spots and synchronization hot spots are synonymous. If the system management functions are implemented in some other way, say a message passing style, the resulting traffic may no longer be combinable.

1.3.2 Run Time Constant Hot Spots

Another possible scenario for hot spots is undetected run time constants. If these run time constants are widely shared, then the faster we execute a program, the "hotter" the hot spot will become. If the compiler is unable to detect all of these constants, then they will eventually cause hot spots. If the compiler can detect these constants, then hot spots may be averted by copying the constants. However, good judgement in this area implies some information about the run time environment (i.e. machine size, speed, load, etc.). This implies a run time decision, perhaps made by the system manager. Because of their real time requirements, run time decisions rarely benefit from complete information and analysis. Thus, some sub-optimal decisions are likely to be made – resulting in hot spots. We will refer to these hot spots as "run time constant hot spots". The use of such constants will depend only on the program (and problem size). Hence the hot spot traffic would also only depend on the program and problem size. References to any scalar constant are 100% combinable. However, if the hot spot is caused by a constant structure, then the traffic may not be combinable. Hot spots due to run time constants are likely to be of long duration.

1.3.3 Random Hot Spots

Random traffic fluctuations may result in hot spot traffic imbalances. The dynamic behavior of programs is sufficiently unpredictable that analysis cannot guarantee an even mapping of traffic for all times. For reasons of efficiency, such mappings must be static or very slowly varying. Therefore if program communication requirements interact with the mapping scheme, hot spots may result. It is important to note that intense hot spots of very short duration can degrade system throughput for a period many times longer than the hot spot duration. Thus random hot spots, though they may be short in duration, may have a very significant impact on network performance.

We will refer to hot spots arising from random traffic fluctuations as random hot spots. Such hot spots will consist of diverse traffic. Thus the traffic in such hot spots is not likely to be combinable. Also, imbalances due to random fluctuations of longer duration are less likely, so we expect most random hot spots to be of short duration.

1.3.4 Locality Hot Spots

Perhaps the most likely scenario is hot spots arising from attempts to enhance spatial locality. High performance systems that do not attempt to do so will require very high communication bandwidth. Such high bandwidth interconnection networks may in fact be prohibitively expensive or completely infeasible. To reduce this required bandwidth, a common technique is to use a local memory to hold structures that are primarily locally accessed. Such attempts to enhance locality, while reducing communication traffic, are likely to increase the amount of contention in the network. If any structure that is used locally later becomes global (this is very likely in systems that make use of heap managed storage), that structure may cause a hot spot. In fact, no clever allocation policy will be able to determine for sure which of the local structures will become global and hence should not be allocated locally. Hot spots resulting from such attempts to enhance locality will be referred to as locality hot spots. They are likely to consist of largely non-combinable traffic

and to range in duration from short to long.

1.4 Summary of the Problem

Hot spots are a serious problem because they degrade the performance of the network to all traffic, not just that traffic destined for the hot spot. There are several plausible scenarios for the occurrence of hot spots in routing networks. The hot spots produced in these varied scenarios have significantly different characteristics. Several of these hot spot scenarios have already been realized – synchronization, run time constant, and enhanced locality hot spots have already been detected in multiprocessor simulations [6,8]. These results lead us to believe that hot spots are an example of a more general class of multiprocessor contention problems. Such problems are very complicated and are unlikely to be solved completely. This means that routing networks, if they are to be a viable alternative to crossbar interconnection, must be somewhat tolerant of hot spots. For this reason, we study the problem of performance degradation due to hot spots in routing networks.

1.5 Overview of the Thesis

Our primary interest is to solve the hot spot problem as described so far. To that end, we have broken the problem into several parts.

Rather than experiment blindly, we wish to have some insight into the network dynamics when hot spots are present. So, the next step is to define a framework (really an abstract model of the network) which allows us to analyze the problem. This model forms the basis for analysis and simulation throughout this thesis. In this framework, we then divide network behavior in the presence of hot spots into three parts – steady state hot spot congestion, the onset of hot spot congestion, and the dissipation of hot spot congestion. We first analyze the steady state network behavior. The resulting formulas are then verified with simulations of the abstract model. Information concerning the steady state network

behavior will tell us how severely long duration hot spots will degrade network performance. Definition of the abstract network model and study of steady state network behavior in the presence of hot spots can be found in Chapter 2.

Subsequently, we examine the transient behavior of the network – the onset and dissipation of hot spot congestion. We define the period of the onset of congestion as the time from the introduction of the hot spot imbalance at the sources to the time the network stage nearest the sources becomes congested. At the end of this period, the tree of buffers should be fully saturated. Similarly, we define the dissipation of congestion as the time from the removal of the hot spot imbalance at the sources to the time the switch nearest the hot spot becomes uncongested. At the end of this time, all network congestion due to the hot spot should be dissipated. Analysis of the transient behavior gives us information about the impact of brief hot spots. For example, if the network congests very rapidly, and dissipates congestion very slowly, then the impact of brief hot spots may be quite lengthy and severe. Conversely, if the network congests slowly, and dissipates congestion rapidly, then only hot spots of at least medium duration will congest the entire network. The shorter ones will be over before the network can congest!

Analysis of transients yields several bounds for the time to congest the network. These bounds are shown to be in quite good agreement with the simulation results. Analysis of the time for hot spot congestion to subside is also presented. The resulting bounds are not very tight, but seem to capture the basic characteristics of the network behavior during dissipation of hot spots. The predicted times to dissipate congestion are compared to the simulated values. Analysis of the transient network behavior is presented and compared to simulations of the abstract model in Chapter 3.

Armed with a working knowledge of network behavior in the presence of hot spots, we are now in a position to intelligently construct solutions to the problem. Towards solving the hot spot problem, we distinguish the crucial aspects of the problem. Solutions to the problem depend on some basic assumptions about the network traffic. These assumptions are presented and discussed in the framework of contemporary multiprocessor architectures.

A solution is proposed and described in detail. The solution is described in Chapter 4.

The proposed solution was implemented and simulated in the framework of the abstract model. The results of these simulations are presented graphically in Chapter 5. These results show the proposed solution to be effective over a variety of network sizes, hot spot intensities, and hot spot durations.

Finally, we summarize the results presented in this thesis. We briefly outline the contributions of the work presented. We then attempt to describe its implications in the general framework of multiprocessor architecture. As a concluding note, some open research issues still remaining are briefly outlined.

Chapter 2

The Analytic Framework

When a hot spot occurs in a multistage routing network, severe degradation of performance can result. Shortly after the hot spot begins, a tree of buffers saturates, blocking the network. The performance degradation is a direct consequence of this network blockage. All traffic from sources participating in the hot spot will experience degradation, as some of their traffic must cross the tree of saturated buffers.

In this chapter, we describe a terminology that allows us to analyze hot spot congestion succinctly. We divide the network behavior into three parts – steady state hot spot congestion, the onset of hot spot congestion, and the dissipation of hot spot congestion. Analysis of these three parts will tell us the impact that hot spots of various durations have on network performance. In this chapter, we analyze one of these parts – steady state hot spot congestion. We characterize the degradation of steady state network performance in the presence of a hot spot. This analysis shows the degradation due to hot spots to be severe. This severity makes it clear that hot spots of long duration will cause serious performance degradation in routing networks. The quantitative results of this analysis are then compared to simulations of the network model and found to be in very close agreement.

2.1 The Network Model

In this thesis, we study a particular type of *Delta Network*, the Indirect-n-cube built from 2-by-2 routers. This network topology was chosen for the simplicity of the routing – at each successive stage we simply shift the address left one bit and use the highest order bit to route. 2-by-2 routers were chosen because they are the smallest possible routers. The low arity of the router allows us to more easily consider quantities such as the local imbalance of traffic. This quantity turns out to be helpful in detecting hot spots. Although we only consider 2-by-2 routers, the analysis in this thesis generalizes readily to networks built from larger routers. The techniques proposed to control "hot spot" congestion are simplest in the context of networks built from 2-by-2 routers, but they are applicable to networks constructed from larger routers.

Simulation results by Kumar and Jump [11] have shown that some increase in throughput can achieved by having "inside" buffers. However, the position of these buffers does not dramatically affect the behavior of the network in the presence of hot spots [5]. For the purpose of simplicity of analysis and simulation, we used routers with input link buffers because such routers have only half as many queues as routers with "inside" buffers. An example of our 2-by-2 router model is displayed in Figure 2.1. Other results [11] have shown that network performance can be improved by replication of either the entire network, or replication of links inside the network. Such replication does not affect the fundamental nature of the "hot spot" problem, so we only consider unreplicated networks.

In characterizing the behavior of routing networks, we chose to use approximate methods. These methods are employed because several characteristics of the hot spot problem make exact analysis of the network behavior difficult. One reason that exact analysis is difficult is the fact that we are considering systems with large numbers of queues. With such large systems, contemporary queueing theory methods enable us to find analytic solutions for only the equilibrium (or steady state) behavior. However, we can only find such solutions if we make use of very simple queueing models. Two features of our system that are not

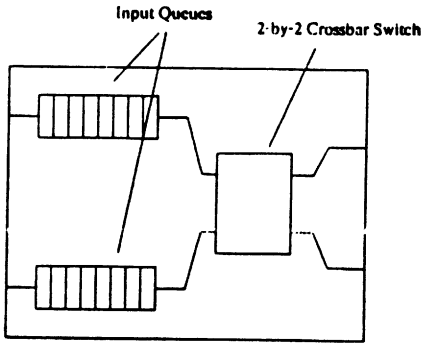

Figure 2.1: A 2-by-2 Router.

easy to incorporate in these models are finite queue length and blocking. As these features are essential to the hot spot problem, we find that we cannot even solve analytically for the equilibrium state of our system. Thus the fact that we are studying a finite queue blocking effect also makes exact analysis difficult. As we implied earlier, analysis of transients is even more difficult than the analysis of steady state. In systems of few queues with limited interaction, such analysis may be feasible. In the networks we considered, analysis of transient behavior was out of the question. The complexity of analyzing such a large stochastic system was too great. Here we see that our interest in the transient behavior of the system also makes exact analysis complicated. Thus, for a variety of reasons, exact analysis is infeasible. Consequently we resorted to approximate methods and extensive simulation of the model to confirm the validity of the formulae derived. In this case, we feel that the use of approximate methods may even yield one benefit – more intuitive derivations.

2.1.1 Definition of Symbols

In order to analyze multistage routing networks, it is helpful to first define the following symbols:

$n =$ # of network output ports = # of input ports

$b =$ The amount of buffering at each stage of switches

k = The stage number (0 at the outputs, $log\ n$ at the inputs)

r = The basic rate at which links can transmit packets

a = The average input loading as a fraction of r

The first three symbols n, b, and k are shown in Figure 2.2. The symbol n refers to both the number of inputs and the number of outputs in the network (always equal for this topology). The symbol b is the number of packets that can be buffered at a router on each input link. The buffers in a given router are not shared between different input links. In general, n and b are considered to be system parameters, and thus implicit arguments to all expressions derived in this thesis. This convention is just to simplify our notation and should not be the source of confusion.

The stage number is defined to be 0 at the destination side of the network. Thus, the maximum value it can attain in these multistage (single path) networks is $(log\ n)$. The buffers within the switch and the input links of the switch receive the same stage number as the switch. Thus the links have stage numbers from 0 to $(log_2\ n)$ and the input buffers at the destinations are not considered to be in the network. The switches have stage numbers from 1 to $(log_2\ n)$. This convention is also illustrated in Figure 2.2.

We define r to be the basic rate at which links can carry packets. It is assumed to be the same for all links in the system. The average network load, a, is equivalent to the background traffic intensity. It is the average load presented at the inputs to the network as a fraction of the physical bandwidth of the links. Under a uniform traffic load, this is the load carried by every link in the network. By multiplying this load by the basic traffic rate, we get the average traffic rate[1], ra. The symbol, a, represents the normalized average loading of the network and should not be confused with the maximum sustainable throughput of a given network configuration.

[1]As a convention, we will indicate multiplication by simple juxtaposition of the terms. In cases where it is ambiguous, we will place an asterisk to denote the multiplication operator. All function names are in uppercase. All quantities other than functions are in lower case.

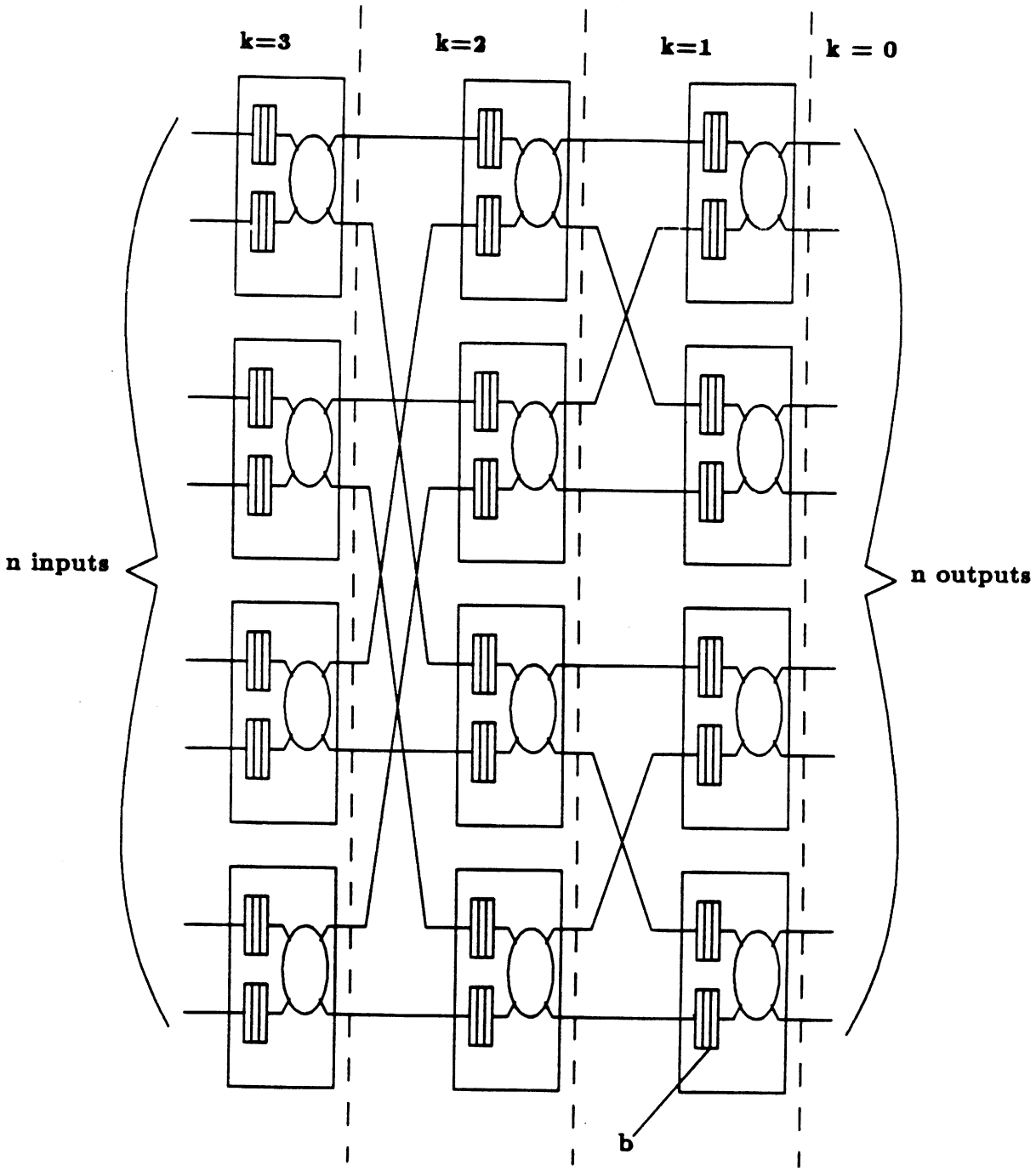

Figure 2.2: An 8 Port Indirect-N-Cube Network.

2.2 Performance of Routing Networks

As a preface to our discussion of hot spot network performance, it is instructive to first consider network performance under statistically balanced traffic. Even with a balanced load, no interconnection network will be able to fully utilize the link bandwidth of r packets/time unit. This is a consequence of messages contending for network outputs. As long as we only allow messages to pass through the network outputs at rate r, even with a full crossbar interconnection, destination contention reduces the achievable throughput to $\approx (1 - \frac{1}{e^a})r$ per input link (see [15] for a more detailed discussion)[2]. If the traffic load applied to a crossbar is uneven, the achievable throughput will be even lower.

Unbuffered multistage routing networks, due to the sharing of internal communication links, can sustain even less throughput than crossbars. Fortunately, Dias and Jump have shown that the addition of a very small amount of buffering to each switch improves the sustainable throughput in these networks dramatically [7]. To allow for a clearer discussion of throughput issues, we will define some terminology. The normalized sustainable throughput (or bandwidth) is some fraction of the maximum link rate r. This is the throughput fraction that the network is able to sustain for each of its inputs. We will denote this fraction by t_{max}, the normalized sustainable throughput for a routing network. Thus, we see that multiplying t_{max} by the basic link speed, r, gives the sustainable input rate for the network inputs.

For realistic networks, if the maximum sustainable throughput for a network is t_{max}, then the average traffic arrival rate for which the network is designed, a will probably be less than t_{max}. Some values for t_{max}, in the case of an *Indirect-n-cube* built from 2-by-2 routers, are given in Table 2.1. For reference, the achievable throughput for a full crossbar network is also shown[3]. In *Delta Networks*, squeezing out the last bit of throughput generally extracts a heavy penalty in terms of average delay. Thus, if we push a too close to t_{max}, then the

[2] In the expression for the achievable throughput, e is the base of the natural logarithm.

[3] The numbers given in Table 2.1 assume 5 packets of buffering per network stage. They also assume "outside" buffers

# of Ports	Full Crossbar	Indirect N-cube
16	0.644	0.63
32	0.638	0.61
64	0.635	0.59
128	0.634	0.58
256	0.633	0.58
512	0.632	0.58

Table 2.1: Sustainable Average Throughputs for Various Networks (normalized)

average delay experienced by packets will increase dramatically. If we push a very close to or even past t_{max}, then a significant fraction of the traffic will be rejected at network inputs. Thus in practice, we think of a as being a fraction significantly less than t_{max}. We see that when a routing network is designed, the designer chooses an operating point that will achieve the desired throughput and network delay. This is possible if we consider only the balanced traffic case.

If the traffic is unbalanced, then the sustainable throughput for the network is less than t_{max}. Traffic imbalances cause increased contention in the network, and if the imbalance is severe enough, blockage of network buffers may occur. This blockage mechanism is how hot spot imbalances cause severe degradation. If the hot spot is severe, the sustainable throughput will drop below the average arrival rate a. In addition, the average delay through the network will increase dramatically. We refer to these effects as "hot spot degradation." This degradation is the topic of analysis in this chapter.

2.2.1 Traffic Terminology

In order to discuss hot spot imbalances, we must first clarify some assumptions. We consider a hot spot to be a point elevation – an elevation of the traffic rate to a single output port. The fraction of the traffic from each source that gives rise to the elevation is denoted by h.

The remaining fraction of the traffic, $(1-h)$, is assumed to be evenly distributed over all of the network outputs, including the hot output. Here we observe that there is no compelling reason to believe that all the inputs will contribute equally to the hot spot. In fact, Lee has conducted some studies of hot spots in which only a few of the inputs participate [13]. Mechanisms that give rise to hot spots as well as multiprocessor traffic statistics in general are still open topics of research. In this thesis, we make a constraining assumption for the purposes of study. We only consider hot spots in which all the sources participate.

We define h as the additional fraction of packets destined for the "hot" output. This is assumed to be the same for all network inputs – all sources are contributing to the the hot spot. In addition, that hot output gets a fair share (the same as all the other outputs) of the remaining traffic. So h actually denotes only the traffic elevation. So, if there is a hot spot at Output 0, then each input sends the fraction $[h + \frac{(1-h)}{n}]$ of its traffic to Output 0. Each of the other outputs (excluding Output 0) receives $\frac{(1-h)}{n}$ of each input's traffic.

For the purpose of our discussion it is useful to introduce some notation for the traffic. In order to make the derivations more intuitive and lucid, we discuss several different partitions of the network traffic. The traffic component names reflect the partition which gave rise to them.

Let $T(h,k)$ denote the total traffic load on a given link of the network. In the presence of traffic imbalances, $T(h,k)$ will be different for different links in the network. The first partition that we consider is between the homogeneous traffic and the traffic due to the hot spot elevation. Since we are specifically interested in hot spot imbalances, we divide the traffic into two components: that due to the homogeneous background load, $B(h)$, and that due to the hot spot elevation, $E(h,k)$. $B(h)$, the background load represents the balanced component of the traffic load and consists of traffic destined for all of the output ports. Thus $B(h) = ra(1-h)$ for all links in the network. $E(h,k)$, the hot spot elevation consists entirely of packets destined for the hot spot. As a result, $E(h,k)$ is zero for all links not in the saturating tree, and non-zero for all links in the saturating tree. $B(h)$ and $E(h,k)$

are disjoint quantities, but together they encompass all traffic in the network. Therefore, Equation 2.1 holds for all links in the network.

$$T(h,k) = B(h) + E(h,k) \tag{2.1}$$

Another useful decomposition of total network traffic uses the destination of the traffic as the basis of discrimination. In this scheme, we distinguish the traffic destined for the hot spot, $HS(h,k)$ from the rest of the traffic in the system – the non hot spot traffic, $NHS(h,k)$. From this definition, it is clear that Equation 2.2 holds for all links in the network.

$$T(h,k) = HS(h,k) + NHS(h,k) \tag{2.2}$$

We further divide the hot spot traffic into that due to the background load, $HSB(h,k)$, and that due to the hot spot elevation, $HSE(h,k)$. It turns out that this division will make the expression of these terms particularly simple at a later point. At this point it is clear that,

$$HS(h,k) = HSE(h,k) + HSB(h,k) \tag{2.3}$$

and of course,

$$T(h,k) = HSE(h,k) + HSB(h,k) + NHS(h,k) \tag{2.4}$$

We point out that $HSE(h,k)$ is the same as $E(h,k)$. Also, $HSB(h,k)$ is the portion of traffic counted in $B(h)$ that is destined for the hot spot.

We refer to all those sources that are sending packets to the hot spot as "contributing" or "participating" sources. In addition, we refer to the tree buffers and links that congest during a hot spot as "saturating links" and "saturating buffers." Packets destined for the hot spot may be referred to as hot spot packets or simply hot packets.

2.3 Steady State Network Behavior in the Presence of Hot Spots

When a hot spot occurs, the aggregate demand for the bandwidth of the hot spot exceeds that link's physical bandwidth. We will consider traffic imbalances which cause the demand to exceed 100%. For less extreme imbalances, the degradation is significantly less severe, and hence of less interest. When a hot spot occurs in an uncontrolled network, the popular link will achieve very close to 100% utilization for the duration of the hot spot. However, even operating at it's maximum rate, the hot spot will not remove hot packets rapidly enough to prevent congestion. This means that tree buffering will result.

When the tree of buffers has fully saturated, network performance is determined by two factors. One factor is the rate at which hot spot packets are removed from the network. This rate is the basic link speed characteristic of the network – the maximum bandwidth of the hot spot link. The other factor affecting network performance is the proportion of hot spot packets in the input traffic, h.

One can see that the steady state network performance depends solely on these two factors by considering the following argument. If all the sources are contributing to the hot spot (as we assumed to be the case), then the tree of saturating buffers extends into each of their output queues. The proportion of hot spot packets to other packets in these queues is determined by the hot spot intensity. However, for any hot spot intensity and network size, the rate at which these hot packets leave the network is fixed by the basic link speed. In the steady state, the rate at which hot packets leave the sources' output queues must be the same as the rate at which the hot packets leave the network. Again, this rate is determined by the basic speed of links in the network. Therefore, the rate at which traffic can leave the source output queues depends directly on the proportion of hot packets in the traffic load and the basic link speed of the network.

Now consider a fully saturated system. Most of the packets have to cross the tree of

saturated buffers to reach their destination. Unfortunately, traffic in these trees moves very slowly. This slow rate limits the rate at which a source can introduce packets into the network (the network only contains a finite amount of buffering). Consequently, the rate at which packets can cross the saturated tree of buffers determines the network performance.

The throughput degradation due to hot spots can be characterized by examining the source and sink rates for "hot spot" packets. If strict FIFO ordering is used in the network buffers, computing the throughput degradation is straightforward. If the hot spot has intensity h, we recall that the fraction of hot packets in the source traffic is $[h + \frac{1-h}{n}]$. Thus if possible, each source would like to sustain rate $[ra(h + \frac{1-h}{n})]$ throughput to the hot spot. However, this will not be possible. We know that the hot spot output is saturated (100% utilized) or we would not consider the traffic imbalance a hot spot. The hot output can only accept packets at rate r. As that bandwidth is shared by all the sources, each source can only submit hot packets at rate $\frac{r}{n}$.

As a result, the source rate for hot packets (and thus for all packets) will be constrained to some rate lower than the desired rate. Applying the rate matching constraint on hot spot packets yields:

$$TH(h)r[h + \frac{1-h}{n}] = \frac{r}{n} \tag{2.5}$$

The expression $TH(h)$ represents the sustainable traffic rate as a fraction of the basic link speed r. Clearly, if we are going to experience degradation, the sustainable traffic rate must be less than the arrival rate, i.e. $TH(h)$ must be less than a. Equation 2.6 tells us that the degradation, $TH(h)$, is inversely proportional to the traffic demand at the hot spot (the denominator on the right side). If we double that demand, the degradation will be exactly twice as bad.

$$TH(h) = \frac{1}{1 + h(n-1)} \tag{2.6}$$

This expression was presented in [17] by Pfister and Norton. In the above expression, n denotes the number of network ports and h denotes the fraction of packets destined for the hot spot from each source. As we claimed earlier, the degraded system performance depends only on r (which we will multiply $TH(h)$ by to get the unnormalized degradation) and h.

2.4 Simulation of Hot Spots

2.4.1 Simulation Framework

Exact analysis of large complex queueing systems such as multistage routing networks is difficult. Consequently, we have resorted to approximate methods in characterizing steady state and transient network behavior. To confirm that our approximations are indeed accurate, we have performed a variety of simulation experiments. These experiments, presented at various points in this thesis, share a common framework. We describe this framework below.

The simulator used to perform the experiments is based on a synchronous network model. The network topology simulated is an Indirect-n-Cube, as previously discussed. For simplicity, all packets are assumed to be of equal length. In each time step, all enabled packets (those that are both at the head of a queue and not losing in the arbitration for a link) are transported one stage toward their destination. In the simulation, buffers in each switch can contain up to five packets. When a buffer is full, it blocks its corresponding input link. The transmission time for a packet (a packet time) is the basic time unit used in graphs and tabular data throughout this thesis. All simulation results presented reflect the averaging of many $(50 - 100)$ simulation runs. The random nature of the traffic generation

made such averaging essential to acquiring reproducible results.

2.4.2 Simulation Results

A simulation study was undertaken to determine the extent of throughput degradation due to hot spots and the accuracy of the derived expressions for the throughput degradation. Because a complete study would require an unreasonably large amount of computation, we studied only one hot spot intensity. We simulated a 16% hot spot in networks with 16, 32, 64, 128, 256, and 512 ports. The results are presented in Table 2.2. The first column denotes the network size for the simulation run described in that row. The second column contains the unnormalized sustainable throughput. This number represents the average number of packets that came out of the network per time step. The next column contains the normalized throughput – the number of packets out divided by the number of network ports. The fourth column contains the normalized throughput values predicted by our approximation for the degraded throughput. And, the fifth column displays the normalized sustainable throughput for the given network size with a statistically balanced load. The last column contains the percentage degradation of the sustainable balanced traffic throughput. Thus, if the network throughput were degraded to zero, this quantity would be 100%. If no degradation occurred, this quantity would be 0%. We observe several things about the results. First, we see that the derived expression for the degraded throughput is very close to the simulation results. This means that our approximation is very good for the cases we simulated. Second, we see that the simulation results show that the hot spot degraded throughput is dramatically lower than the balanced load throughput. This point is even more obvious if we examine the percentage degradation column. Finally, we see that the maximum sustainable throughput (unnormalized) does not seem to depend strongly on the network size. We held the hot spot intensity constant and the unnormalized throughput peaked at about 6.1 packets/time step.

# of Nodes	Average Throughput	Normalized Throughput	Predicted Throughput	Balanced Traffic	Percentage Degradation
16	4.7	0.293	0.29	0.397	26.2
32	5.4	0.168	0.17	0.397	57.7
64	5.9	0.092	0.090	0.396	76.8
128	6.3	0.049	0.047	0.396	87.6
256	6.1	0.024	0.024	0.394	93.9
512	6.1	0.012	0.012	0.392	96.9

Table 2.2: Throughput Degradation with 16% Hot Spot

2.5 Delay Degradation

In the presence of a hot spot, both hot spot traffic and non hot spot traffic experience increased delay. After the system has fully saturated, many non hot spot packets need to cross the tree of saturated buffers in order to reach their destination. For example, we know that all traffic from sources contributing to the hot spot must traverse at least one level of the tree. This is because the output buffers of the contributing sources will be part of the tree of saturated buffers. Some packets may only traverse one layer of the tree, but many of them will have to cross several layers. Crossing layers of the tree of buffers can take a very long time.

In a network of 2-by-2 routers, the saturated tree is $log_2 n$ stages deep. Depending on the packet's destination, it may have to cross from 1 to $log_2 n$ levels of the tree. By examining the rates of packet flow in the saturated buffers, we describe the time $L(h, k)$ to pass through a buffer of the tree at stage k. This tells us how the traffic for non hot spot destinations will be affected.

We approximate the delay in a hot spot congested network by examining two components of the delay: the time to cross the saturating tree, and the time to reach the destination after leaving the saturating tree. If the network throughput is significantly degraded, then packets are unlikely to experience contention after leaving the saturated tree. We assume

the queueing time is zero in this component of the travel and consider only the transit time from stage to stage. While crossing the saturating tree, the delay a packet experiences is dominated by the queueing time. Therefore, in this component of the travel, we will only consider the queueing time. Using, p to denote the number of congested stages that a packet must cross, we can approximate the delay experienced by that packet as follows:

$$DELAY(h,p) = (log_2 n - p)l_o + \sum_{k=log_2 n-(p-1)}^{log_2 n} L(h,k) \tag{2.7}$$

The symbol $DELAY(h,p)$ denotes the delay a packet will experience in the presence of a hot spot of intensity h, given that the traffic from a to b must cross p congested stages. The symbol $L(h,k)$ is the queueing time in stage k of the saturating tree, and $(log_2 n - p)l_o$ is the flight time of the packet through the uncongested portion of the network. We take the time to send a packet from one stage to the next (assuming no contention) to be some constant, l_o.

Now, to compute the average delay, we need only determine $L(h,k)$, and compute a weighted average of $DELAY(h,p)$. This average should be weighted by the traffic distribution. The symbol $TR(s,d)$ denotes the fraction of the entire network traffic that originates from s and is destined for d (the traffic distribution). We also denote the number of congested stages that traffic from s to d must cross by $STAGES(s,d)$. Thus the average delay, $AVGDELAY(h)$, can be written:

$$AVGDELAY(h,n) = \sum_{s\in S}\sum_{d\in D} TR(s,d) * DELAY(h, STAGES(s,d)) \tag{2.8}$$

Naturally, S is the set of all sources, and D is the set of all destinations. We know how to determine $STAGES(s,d)$. It is strictly an attribute of the network topology and routing function. And $TR(s,d)$ comes directly from the traffic model. The only thing that we must

yet determine is the function $DELAY(h,p)$.

To approximate $DELAY(h,p)$, we assume that the hot spot packets are the rate determiners in the saturated buffers. With this assumption, we see that the time for a packet to pass through a congested buffer can be approximated by the time to remove the hot spot packets in the buffer. Since we know that hot packets leave the network at rate r, then we see that at stage k, they leave the buffer at rate $\frac{r}{2^k}$. We use the expression $HSP(h,k)b$ to denote the number of hot spot packets, in a buffer[4]. Then, we can write $L(h,k)$ as:

$$L(h,k) = HSP(h,k)\frac{2^k}{r}b \tag{2.9}$$

Consequently, we can write the following:

$$DELAY(h,p) = (log_2 n - p)l_o + \sum_{k=log_2 n-(p-1)}^{log_2 n} b * HSP(h,k)\frac{2^k}{r} \tag{2.10}$$

Now we have all of the expressions necessary to calculate the average delay. We simply substitute Equation 2.10 into Equation 2.8. This gives us an approximation for the average delay. We could have determined this quantity by simply applying Little's Law (if we knew the number of packets in the system). However, this derivation has been instructive because it shows us that traffic for different source and destination pairs may experience radically different network delay.

2.5.1 Simulation Results

The average delay is a very important metric of performance of a routing network. For applications in which there is little tolerance for latency, this number may be the single most important metric for network performance. To check the accuracy of our characterization,

[4]The quantity $HSP(h,k)$ is derived in Chapter 3

Number of Nodes	Simulated Average Delay	Predicted Average Delay	Balanced Traffic	Factor of Increase
16	30.4	31.0	6.1	5.0
32	63.5	57.7	7.4	8.6
64	119.3	109.9	8.8	13.6
128	232.1	213.3	10.0	23.2
256	421.5	418.9	11.4	37.0
512	908.5	829.4	12.7	71.5

Table 2.3: Average Delay with 16% Hot Spot

we simulated a 16% hot spot in various size networks (ranging from 16 to 512 ports) and compared the results to those predicted by $AVGDELAY(h)$. In the simulation, buffers in each switch could contain up to five packets ($b = 5$). After they were full, they blocked their input link. The values of $AVGDELAY(h)$ are scaled appropriately to take the buffer size into account. The simulated and predicted values are summarized in Table 2.3. For reference, the simulated average delay in the case of balanced traffic is displayed in the fourth column. The dramatic increase in the delay is highlighted in the last column. The values in the last column can be computed by dividing the the values in column 2 by those in column 4. These values reflect the sharp increase in the simulated average delay. We make two important observations about the simulation results. First, the simulation results show that the derived expression for the average delay is indeed an accurate one. Second, the simulation results show that the increase in delay when a hot spot occurs is very severe.

2.6 Summary

In Chapter 2, we have defined terminology and an abstract network model for analysis and simulation. On this basis, we characterized the degradation of network performance in the presence of a hot spot. We were able to characterize the steady state network throughput and delay in the presence of a hot spot. Simulations were carried out in order to validate

our results. These simulations showed that the expressions for degradation of network performance (both the average delay and the system throughput) to be in close agreement with the values predicted by the analysis. Furthermore, the simulation results showed that degradation – increased delay and decreased throughput – due to hot spots can be very severe.

Chapter 3

Transient Behavior of Networks in the Presence of Hot Spots

To determine how to deal with the hot spot problem in a high speed routing network, it is useful to know how rapidly a "hot spot" can affect the network and how long that effect lingers after the "hot spot" is removed. In this chapter, we attempt to approximate the average time for a network to congest and the time to dissipate that congestion. In order to do so, we first examine the distribution and the total number of hot spot packets in a fully saturated system[1]. This information allows us to analyze and derive several bounds for the congestion time – the time from when a hot spot is introduced to the time the system is fully saturated. We also derive bounds for the time to dissipate congestion after a hot spot is removed from the system. The accuracy of these bounds is checked against simulation results. When discussing bounds for the time for tree saturation to occur and the time for it to dissipate, we will refer to the former as Congestion Bounds and the latter as Dissipation Bounds.

[1] A tree of buffers has filled completely and extends from the hot spot back to all participating sources.

3.1 The Number of Hot Spot Packets

In order to analyze of the network behavior in the presence of a hot spot, it is helpful first to examine the number of hot spot packets that will be in the system when the tree of saturating buffers is completely full. This number depends only on the number of ports in the communication network, the hot spot intensity, and size of the buffers at each stage. In this section, we derive an expression for the "steady state" number of hot spot packets in a congested network.

3.1.1 The Fraction of Hot Packets

When a "hot spot" is introduced in the system, for a short period of time, hot spot packets enter the network at a higher rate than they exit it. Consequently, hot spot packets accumulate in the network. Due to the routing property of the network, all of the hot spot packets in the network can be found in the tree of buffers leading from the contributing sources to the hot output. These buffers gradually fill up with hot packets and other traffic. When the buffers are full, a tree of full buffers leading "backwards" to the sources has been constructed. At this point, we say that the network is fully saturated.

In a fully saturated network, the number of hot spot packets in the network has stabilized. The average rates at which hot spot packets flow in and out of the network are once again balanced. In fact the average rates of flow in and out of each saturated buffer are balanced. We can determine the time average distribution of hot spot packets in the saturated buffers because it is just a function of the hot spot intensity and the buffer position in the network. We characterize the distribution of hot packets in saturated buffers by first considering the background load and then superimposing the hot spot load.

First consider the homogeneous traffic, $B(h)$ throughout the tree of saturating buffers. We know that the background load is the same on all links in the network, so we can write $B(h)$ as follows:

$$B(h) = TH(h)r(1-h) \tag{3.1}$$

We write $TH(h)$ instead of a to denote the sustainable load. If degradation occurs, $TH(h)$ will be less than a, the offered load. Due to the nature of routing, the closer we move to the destination side of the network, the fewer outputs this traffic can have as a destination. In fact, the number of possible destinations decreases by a factor of 2 at each stage[2]. The total amount of traffic is constant, so we see that the amount of traffic for each possible destination must increase by a factor of 2 for each stage. Thus the hot spot traffic due to the background load, $HSB(h,k)$, on the links of the saturating tree may be written as follows:

$$HSB(h,k) = B(h)2^{-k} = TH(h)(1-h)2^{-k}r \tag{3.2}$$

The exponent is $-k$ because we chose to number the stages from the destination side of the network back towards the sources. Checking this equation at the boundaries, we see that we get $TH(h)(1-h)r$ for $\mathbf{k = 0}$, as we should. We also get $TH(h)\frac{(1-h)}{n}r$ for $\mathbf{k = (\log_2 n)}$, which is also correct.

Now we consider the hot spot load due to the hot spot elevation. We denote this quantity $HSE(h,k)$. This load also increases exponentially as we move towards the sinks. It can be written as follows:

$$HSE(h,k) = h2^{log_2 n - k}TH(h)r = (nh)2^{-k}TH(h)r \tag{3.3}$$

Again this meets the boundary conditions at both edges of the network. We see that $HSE(h,0) = TH(h)rnh$ for $k = 0$ and $HSE(h, log\ n) = TH(h)rh$ for $k = log\ n$ as expected. Given $HSB(h,k)$, $HSE(h,k)$, and $B(h)$, we can calculate directly the distribution for hot

[2]More generally, it decreases by a factor of the switch arity.

packets on the links of the saturating tree. To do so, let's first define two quantities. We calculate the total load on the links of the saturating tree and denote it $T(h,k)$. We will also calculate the load due to hot packets and denote it $HS(h,k)$. Directly form Equation 2.1, we see that total load can be expressed as follows:

$$T(h,k) = B(h) + HSE(h,k) = [(1-h) + (nh)2^{-k}]TH(h)r \tag{3.4}$$

Referring to Equation 2.3 the load on the saturating links due to all traffic destined for the hot spot is simply:

$$HS(h,k) = HSE(h,k) + HSB(h,k) = [(nh)2^{-k} + (1-h)2^{-k}]TH(h)r \tag{3.5}$$

Of course the load due to hot packets on all other links in the network is zero.

Having calculated both the overall load and the hot spot load for the links of the saturating tree, now we can deduce the hot packet distribution on those links. This quantity, denoted by $HS^{FR}(h,k)$, is presented below. $HS^{FR}(h,k)$ is shorthand for the hot spot fraction of the overall traffic in each buffer.

$$HS^{FR}(h,k) = \frac{HS(h,k)}{T(h,k)} = \frac{(nh)2^{-k} + (1-h)2^{-k}}{(1-h) + (nh)2^{-k}} \tag{3.6}$$

Simplifying gives:

$$HS^{FR}(h,k) = \frac{nh + (1-h)}{nh + (1-h)2^{k}} \tag{3.7}$$

As a check on our results, we examine this equation at the boundaries of the network. At the lower boundary, $k = 0$,

$$HS^{FR}(h,0) = \frac{(nh + (1-h))}{(nh) + 1(1-h)} = 1 \tag{3.8}$$

And at the upper boundary, $k = log\ n$,

$$HS^{FR}(h, log\ n) = h + \frac{(1-h)}{n} \tag{3.9}$$

This is just as we expected. At the destination side, one more step would give us all hot packets. At the source side of the network, we get the expected fraction of traffic destined for the hot output.

3.1.2 The Number of Hot Spot Packets

Having characterized the fraction of packets which are hot spot packets on the links of each network stage, we make the following observation. The distribution of the hot packets on the links is exactly the same as that on the input buffers of the following stage. As a result, when the tree of buffers has fully saturated, we can calculate the number of hot spot packets in the buffers. We do so by scaling our expression for $HS^{FR}(h,k)$ by the amount of buffering in the saturating tree for each stage of the network. Doing so, we can write the number of "hot spot" packets in a saturating buffer in stage k as $HSP(h,k)$.

$$HSP(h,k) = HS^{FR}(h,k)b = \frac{nh + (1-h)}{nh + 2^k(1-h)}b \tag{3.10}$$

Multiplying by the appropriate number of buffers at each stage and summing over all of the stages gives us the number hot spot packets in the system. This result is presented below.

$$HSP(h) = \sum_{k=1}^{log_2 n} \frac{nh + (1-h)}{nh + 2^k(1-h)} 2^k b \tag{3.11}$$

This quantity is linear in the switch buffer size, and many of the quantities derived will also be linear in the buffer size. In order to present results independent of buffer size, unless otherwise specified, bounds for congestion and dissipation are normalized by the number of buffers per stage.

3.2 Approximations to the Congestion Time

When a network is fully saturated, the rate at which all packets pass through the network is determined by the rate at which the hot sink can absorb packets. These packets are the "rate determiners" in the tree of saturating buffers. Furthermore, since this tree of buffers extends back to all of the sources, the rate at which the hot spot packets can move determines the throughput of the entire system. The degree of this degradation has been characterized in Section 2.3.

We define the Congestion Time as the time from when the input traffic distribution becomes unbalanced to the time when a tree of buffers in the network is fully saturated. We actually would like to measure the time until the system reaches steady state. This condition is difficult to detect, so we use the buffer saturation criterion. In an attempt to characterize the congestion time, we will present three approximations for the time required for a hot spot to fully saturate the network. For the first of these, we note that the hot spot packets make up a significant fraction of the packets buffered in the system. So in order to get a bound, we examine the time required to accumulate the surplus number of hot spot packets in the network. For the second bound, we examine the flow rates of packets in and out of buffers in the saturating tree as it congests. By inspecting the rates at the fringe of the tree of buffers, we can estimate the rate at which it grows. With that rate, it is possible to project a congestion time for the entire network. This is the second approximation. We

then attempt to improve this bound by accounting for average buffer fills. This adjustment yields a third approximation.

3.2.1 Overall Congestion Bound

Consider the total number of hot spot packets in a fully saturated system, $HSP(h)$[3]. The system cannot become fully saturated until at least $HSP(h)$ hot spot packets have been injected into the system. Hence the time to inject $HSP(h)$ hot spot packets into the system is a conservative approximation (less than the actual time) to the time required for full saturation.

The rate at which hot spot packets enter the system is:

$$G(h,a) = nra[h + \frac{(1-h)}{n}] \tag{3.12}$$

Dividing the number of hot spot packets required to congest the network by the generation of these packets gives the following bound on the time to congest the network:

$$CB1(h,a) = \frac{HSP(h)}{G(h,a)} = \frac{\sum_{k=1}^{log_2 n} \frac{nh+(1-h)}{nh+2^k(1-h)} 2^k}{nra[h + \frac{(1-h)}{n}]} \tag{3.13}$$

This bound is useful because it tells us that we must have *at least* $CB1(h,a)$ packet times for each packet of router buffering before the network is fully saturated. However, it is not a tight bound. The expression for $CB1(h,a)$ does not take into account the flow of hot spot packets out of the system as it is becoming congested. For very severe (large values of $G(h,a)$) hot spots, the flow is not very significant, and the bound should be reasonable. For mild hot spots ($G(h,a)$ slightly greater than 1), this flow may be significant. In this

[3]The b has been dropped for the purpose of presenting results independent of buffer size.

case, we would not expect this bound to be very tight. These expectations are verified by simulation results in Table 3.1. The larger network sizes are the more severe hot spot cases.

3.2.2 Stage by Stage Congestion Bound

The bound derived in the previous section is interesting, but for moderate hot spots does not give us a good indication of how fast the network will congest. In order to get a better approximation, we must account for the hot packets that leave the network during the Congestion Time. To do so, at each stage, we consider the distribution of hot spot packets and the accumulation rate of the hot packets in those buffers.

After the hot spot traffic has begun to arrive at the hot spot, it gradually fills the buffers all the way back to the sources. The rate at which a given buffer saturates is determined by the rate at which hot spot packets enter and leave that buffer. By examining this rate difference, and the steady state number of hot spot packets in the saturated buffers, we can approximate the time to fill the buffers in each stage. Summing over the stages gives us an approximation for the time to congest the entire network.

At stage k in the tree of buffers, the rate at which hot spot packets enter a buffer ($RI(h,a,k)$, the Rate In) is the rate at the sources, $ra[h+\frac{1-h}{n}]$, multiplied by an exponential factor. The exponential factor arises from the "concentration" of packets destined for a given output as the routing network sends packets towards their destinations.

$$RI(h,a,k) = ra[h + \frac{(1-h)}{n}]2^{log_2 n - k} \tag{3.14}$$

We will assume that once the hot spot packets begin to arrive at the hot output, that link becomes 100% utilized. Thus it will sink packets at rate r. Since the relative rates at which packets move through different buffers in the saturating tree is fixed by the traffic distribution, knowing this one rate tells us the rates of flow in all buffers of the saturating

tree. Thus at stage k, the rate at which hot spot packets leave a buffer ($RO(k)$, the Rate Out) in the saturating tree is:

$$RO(k) = \frac{r}{2^k} = r2^{-k} \tag{3.15}$$

Putting the previous two equations together with the expression for the hot packet distribution $HS^{FR}(h,k)$ derived earlier, we get the following expression for the fill time for each packet of buffering at stage k.

$$FT(h,k,a) = \frac{HS^{FR}(h,k)}{RI(h,a,k) - RO(k)} \tag{3.16}$$

$$FT(h,k,a) = \frac{\frac{nh+(1-h)}{nh+2^k(1-h)}}{ra[h + \frac{(1-h)}{n}]2^{log_2 n-k} - \frac{r}{2^k}} \tag{3.17}$$

Which slightly simplified is:

$$FT(h,k,a) = \frac{\frac{nh+(1-h)}{nh+2^k(1-h)}2^k}{ra[nh + (1-h)] - r} \tag{3.18}$$

Using this expression, we can estimate the fill time for the entire network, provided we make some assumption about how much buffering is available in the network. It is important to note that we need to know more than just the physical size of the buffers. The quantity of interest is the number of empty buffers remaining in stage k when congestion reaches that stage. In order to get a conservative bound, we assume that the buffers were completely empty before congestion began. Summing over the stages of the network yields the following expression for the overall network congestion time (assuming b buffers per stage):

$$CB2(h,a) = \frac{b}{ra[nh+(1-h)]-r} \sum_{k=1}^{log_2 n} \frac{nh+(1-h)}{nh+2^k(1-h)} 2^k \tag{3.19}$$

In order to get a closer approximation to the fill time, we can make the following assumption. We assume that the buffer fill distribution in the network before the onset of congestion is approximated by the steady state distribution of packets in the network under uniform load. Accounting for the average buffer fill before a hot spot should give us a closer bound for moderate hot spots. This quantity was derived by Kruskal and Snir[10].

$$BUFF(a) = \frac{a}{4(1-a)} \tag{3.20}$$

Using this equation to calculate the buffering available to absorb hot spot congestion in the network, we see that the buffering is reduced by a constant amount at each stage. If we use this estimate and sum over the stages of the network, we get:

$$CB3(h,a) = \frac{b-\frac{a}{4(1-a)}}{ra[nh+(1-h)]-r} \sum_{k=1}^{log_2 n} \frac{nh+(1-h)}{nh+2^k(1-h)} 2^k \tag{3.21}$$

3.2.3 Comparison to Simulation Results

A simulation study was performed in order to determine the accuracy of the congestion bounds we have derived. Again, the congestion time is the interval from the time the hot spot is introduced to the time the network is congested. In the simulation study, the criterion for declaring a network to be congested was that the buffers in the last stage (nearest the sources) be 80% occupied. It was necessary to use this threshold because during a simulation, due to statistical fluctuations and the fact that packets are discrete quantities, it might take very much longer to reach the state where $\approx$100% of the last stage

# of Nodes	Bound 1 Pred.	Bound 2 Pred.	Bound 3 Pred.	Simulated Time Measured
16	25.1	142.1	137.4	113.4
32	28.8	71.1	68.8	89.4
64	30.9	56.0	54.2	65.7
128	32.0	50.4	48.8	57.8
256	32.6	48.0	46.4	55.0
512	32.9	46.8	45.3	54.2

Table 3.1: Congestion Times with 16% Hot Spot

buffers were be occupied. As in previously described simulations, we studied a 16% hot spot. This hot spot was simulated for networks from size 16 to 512. In this simulation, as in all the simulation results presented in this thesis, the data presented reflects the results of averaging over a large number of runs to improve the accuracy of the simulated average congestion times.

The $CB1$, $CB2$, and $CB3$ bounds are plotted along with the simulation results in Figure 3.1. For reference, the data is tabulated in Table 3.1. Columns 2, 3, and 4 reflect the values for the average congestion time predicted by the derived approximations. The last column contains the simulation results for the average congestion time. From the simulation data, we see that congestion bounds $CB1$ and $CB2$ are indeed lower bounds on the time to congest the network. As we expected, $CB1$ is a poor approximation for moderate hot spots, but much better for severe hot spots. We also point out that congestion times are relatively short ($\approx 50 - 100$ packet times). This means that any effective network control mechanism must detect a hot spot early and respond to it rapidly.

We see from Figure 3.1 that $CB2$ and $CB3$ capture the essential behavior of the network during the onset of congestion. In fact, the two bounds are directly proportional to each other, so the fact that they have the same shape is not a surprise. $CB2$ yields a very good approximation of the time to congest the network. We hoped that the constant factor used to scale $CB3$ would make it more accurate. However, this did not turn out to be

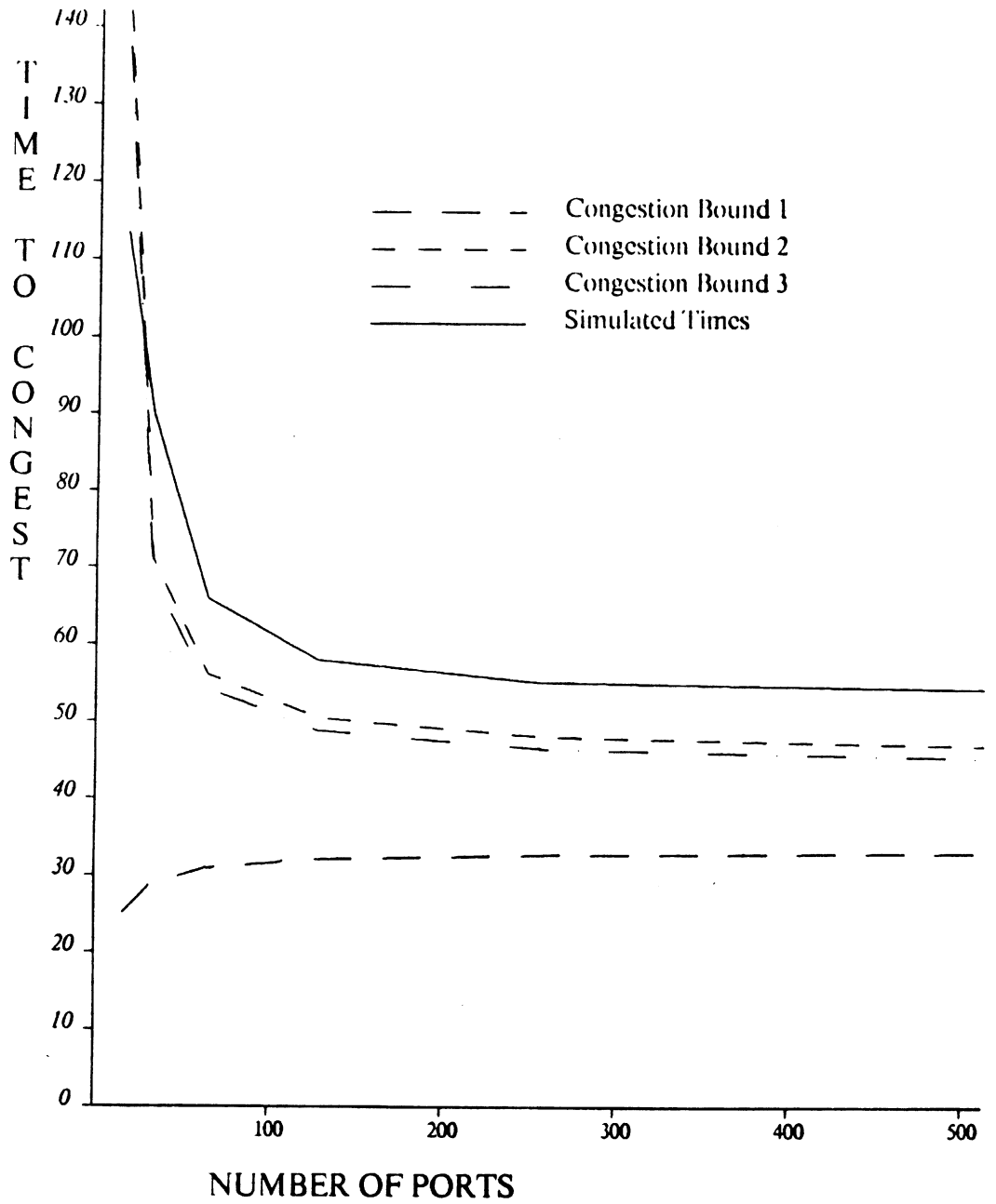

Figure 3.1: Three Congestion Bounds along with Simulation Results.

the case. Thus, $CB2(h, a)$ is our most realistic bound for the time to congest the network. The bounds in this chapter are similar to the bounds derived in [12]. However, these were derived independently.

3.3 Dissipation Bounds

After a hot spot has ended, i.e. the source traffic has become balanced again, there is a period of transience in the network from the saturated state to the normal uncongested state. We define the Dissipation Time as the time from when the hot spot ends to the time that the network has returned to its normal uncongested state. The length of this period is crucial in determining the significance of the "hot spot" problem. If this period is very long, hot spots that last relatively short periods will be a serious problem. If this period is very short, then only the hot spots of long duration will be a problem.

During congestion the tree of saturating buffers fills from the sinks backwards through the network. Conversely, during dissipation the network clears from the sources to the sinks (i.e., the stages near the sources get decongested before those near the sinks). As a result, the major effects of congestion may be relieved when only the first few stages have been drained. Despite this observation, we proceed to consider approximations for the entire Dissipation Time.

In deriving bounds, we will draw on the expression we derived for $HSP(h, k)$, the number of hot spot packets in each buffer of a saturated network stage. Using this expression, we will derive two bounds: a strict lower bound and a closer approximation. First, let us consider the strict lower bound.

3.3.1 A Simple Approximation for the Dissipation Time

The time to dissipate the traffic in the saturated tree of buffers and hence the congestion in the network can be characterized by examining the population of hot spot packets in the

network. At the time the hot spot ends, we assume that the network is fully saturated, and the distribution of hot spot packets in the buffers follows the distribution of $HSP(h,k)$. When the congestion has been relieved, the packets for the formerly hot sink have the same distribution as the packets destined for any other sink. Therefore, we can approximate the time to return to the normal distribution by the time for the hot sink to absorb all the hot packets in the saturated buffers.

At what rate do the hot spot packets leave the network in this transient period? As long as some part of the tree of buffers is saturated, the hot sink will be fully utilized. With the traffic rate normalization used in this paper, that means the hot spot packets leave the network at rate r. A simple lower bound on the time to dissipate the congestion in the network is the time for the hot spot to absorb the hot packets in the tree of saturating buffers. Thus the time to decongest one packet of buffering at each stage ($DT(h,k)$, the Drain Time[4]) can be found by dividing the number of packets at each stage by the sink rate.

$$DT(h,k) = \frac{\frac{nh+(1-h)}{nh+2^k(1-h)}2^k}{r} = \frac{nh+(1-h)}{nh+2^k(1-h)}\frac{2^k}{r} \tag{3.22}$$

By summing over the stages, we get a lower bound on the time to decongest the entire network:

$$DB1(h) = \sum_{k=1}^{log_2 n} \frac{nh+(1-h)}{nh+2^k(1-h)}\frac{2^k}{r} \tag{3.23}$$

The $DB1(h)$ bound provides a strict lower bound on the time to decongest the network. However, it neglects the entry of hot packets during the decongestion period. As a result, it will only be a tight bound for the case when the decongestion period is very short. This will be the case when we consider marginal hot spots and when the average network load,

[4]Normalized by the number of buffers per switch stage.

a, is low. In both of these cases, the influx of new hot packets during the dissipation phase will be less significant. Thus we would expect the $DB1$ bound to be tightest for these cases.

3.3.2 Stage by Stage Dissipation Bound

The bound derived in the previous section is interesting, but fails to capture the finer level behavior of the network. For this reason, it is not intuitively satisfactory. In this section we characterize the dissipation of congestion by examining the rate mismatch for the hot spot packets on a stage by stage basis. The results of this analysis give us a significantly better bound.

To consider the rate mismatch, we must first characterize the rate at which hot spot packets move into and out of each stage during the hot spot dissipation period. In fact, we will only consider the rates at the boundary of the saturating tree. That is, the tree stage where the next stage (toward the destinations) is fully saturated, and the previous stage is unsaturated. For the buffers at the boundary of the saturating tree (stage k), rate at which hot spot packets move out can be written as:

$$OUT(k) = \frac{r}{2^k} = r2^{-k} \tag{3.24}$$

This follows from the reasoning of the previous section – if any of the buffers are saturated, the hot sink must be fully utilized. The rate at which hot spot packets enter the stage k buffers is difficult to characterize precisely. We recall that the input traffic during the decongestion period is assumed to be balanced, so we will approximate the hot spot traffic as follows:

$$IN(h,k) = \frac{ra}{n}2^{log\ n-k} = ra2^{-k} \tag{3.25}$$

We assume that the traffic into the buffer is the same as if there were no congestion

anywhere in the network. Combining the difference in rates and the fraction of the packets in the buffers that are destined for the hot spot, we get a bound for the time to decongest a buffer in each stage of the network.

$$DT2(h,k) = \frac{HS^{FR}(h,k)}{[OUT(k) - IN(h,k,n)]} \tag{3.26}$$

Plugging in for the various terms, and summing over all of the stages of the network we get a bound for the time to congest the network:

$$DB2(h) = \frac{1}{r(1-a)} \sum_{k=1}^{log_2 n} \frac{nh + (1-h)}{nh + 2^k(1-h)} 2^k \tag{3.27}$$

This bound assumes that the network is clear of congestion as soon as the excess hot packets are gone. This need not be the case, as we explain in Section 5.3. As with the $DB1(h)$ bound, the $DB2(h)$ bound is most likely to be close to experimental values for marginal hot spots.

3.4 Comparison to Simulation Results

Simulations were performed in order to evaluate the accuracy of the derived dissipation bounds. Recall that the dissipation time is the interval from the time to the hot spot is removed (the traffic becomes balanced) to the time the network is uncongested. We defined a network stage as being uncongested when less than 80% of its saturating tree buffers are full. As a result, the entire network is considered to be uncongested when less than 80% of the switch buffers in the switch nearest to the hot spot are full. In this study, we again simulated a 16% hot spot in a variety of network sizes (ranging from 16 to 512).

The $DB1$ and $DB2$ bounds are plotted along with the simulation results in Figure 3.2.

# of Nodes	Bound 1 Pred.	Bound 2 Pred.	Simulated Time Measured
16	24.8	85.3	46.7
32	29.5	164.1	110.5
64	34.0	320.5	216.6
128	38.7	632.8	438.4
256	43.4	1257.0	871.2
512	48.2	2505.2	1720.0

Table 3.2: Dissipation Bounds with 16% Hot Spot

For reference, the data is also tabulated in Table 3.2. The values for average dissipation time predicted by the bounds are presented in the second and third columns. The simulation values for the average dissipation time are listed in the last column. From the simulation data, we see that $DB1$ is not a very tight lower bound on the time to decongest the network. Our other bound, $DB2$ seems to capture the essential behavior of the congestion dissipation process. However, the times predicted by $DB2$ are significantly greater than the actual measured times. We also note that the dissipation times can be quite long. In fact, considerably longer than the congestion times. This raises the possibility that a very brief but intense hot spot may degrade system performance for a long period.

3.5 Summary

We have attempted to characterize the temporal behavior of networks when hot spots are introduced and removed. To facilitate our study, we derived expressions for the number of hot spot packets in a fully saturated system, as well as the distribution of hot spot packets throughout the tree of saturated buffers. Three bounds for the congestion time were derived. One represented a lower bound and is useful for determining the minimum time we will have to respond to a traffic imbalance. The second was a closer approximation which gives us an indication of the average time until total saturation. The third bound

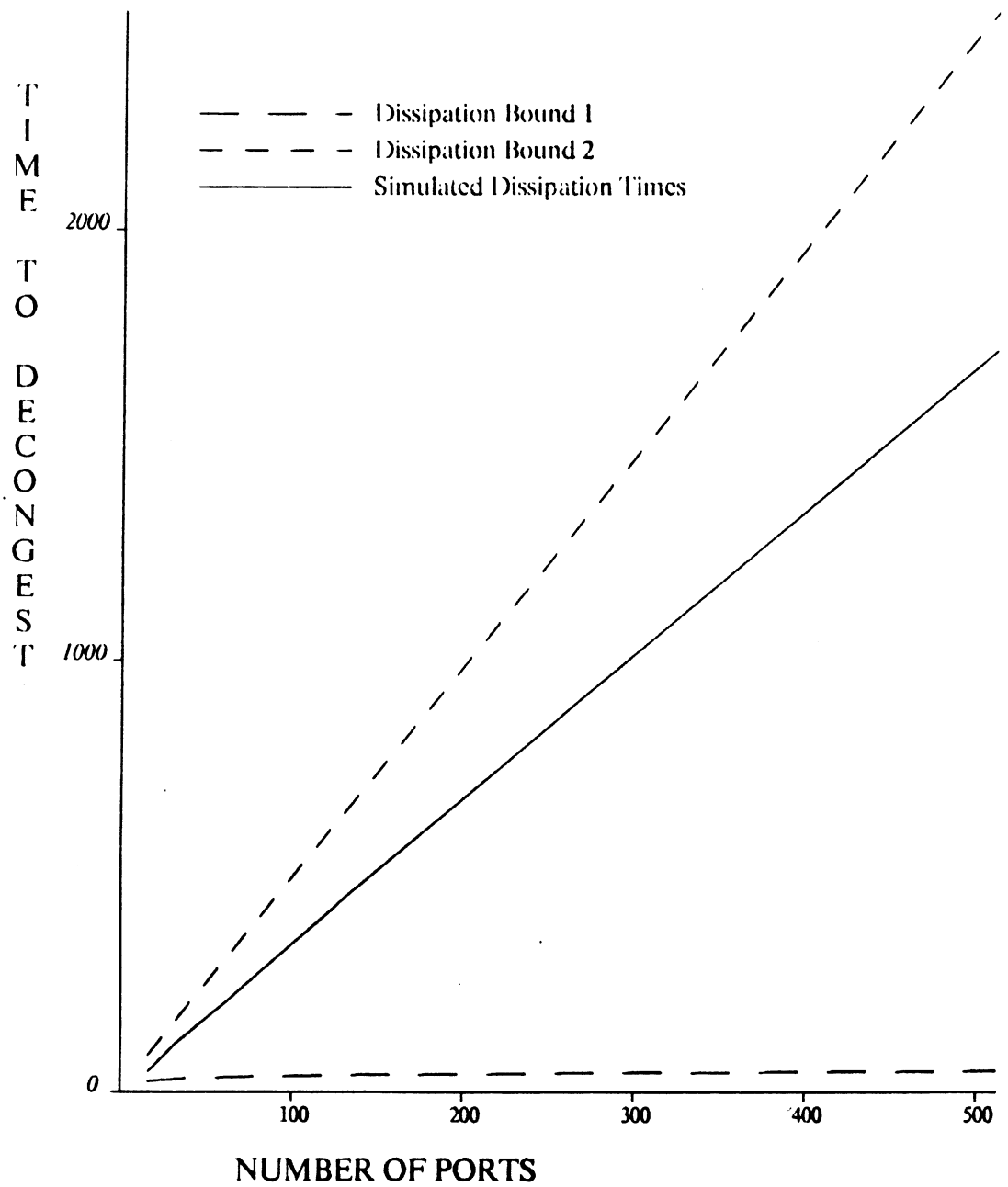

Figure 3.2: Two Dissipation Bounds along with Simulation Results.

was based on more sophisticated assumptions about the background traffic, yet failed to give us a better approximation of the actual congestion time.

Two bounds for the time to dissipate congestion were derived. One gave us a strict lower bound on the dissipation time. The second bound gave us a reasonable approximation of the actual time to dissipate "hot spot" congestion. These bounds give us a guide for how long the effects of a hot spot will linger in the network.

The bounds for Congestion and Dissipation Bounds were compared to simulation results. These results showed the stage by stage bounds ($CB2$, $CB3$, and $DB2$) to be reasonably accurate. Together, these bounds provide a reasonably accurate analytic characterization of the onset of congestion and the dissipation of congestion. The results from our study the congestion and dissipation of congestion tell us several things. First, congestion occurs very rapidly. This means that any network that can deal with the problem must be able to detect it quickly and take effective action. Second, rapid congestion means that hot spots of short durations will also congest the network. Hence short hot spots may be a problem. Finally, relatively long decongest times tell us that the network requires a long time to recover from a hot spot. This means that the effects of a brief imbalance can last far beyond the brief duration of the imbalance. This raises further concern because brief hot spot are very likely. We must be able to avert this sort of degradation, or network performance is likely to be abysmal.

Chapter 4

Controlling Hot Spot Degradation

4.1 The Basic Problem

The basic hot spot problem is a rate mismatch between those sources contributing to the problem, and the hot spot sink. This mismatch has dramatic effects on system throughput because of the link and buffer sharing in typical multistage routing networks. Any solution which does not address the basic rate mismatch can only hope to defer the onset of congestion. Thus, techniques such as additional buffering, queue reordering, multipath routing will only delay the buildup of the tree of saturating buffers.

In order to solve the rate mismatch problem, it is necessary to make certain assumptions about the network traffic. These assumptions are described and motivated. The circumstances under which they are reasonable are described. Within the framework of these assumptions, a network management system for controlling hot spot degradation is proposed.

4.2 Flow Control in Data Networks

Rate mismatch problems in Local Area Networks or Distributed Data Networks (DDN) are typically addressed by flow control techniques. Such techniques include isarithmic, discarding of packets, flow balancing, and various window schemes. However, these techniques were developed in systems where computation at the nodes was relatively cheap (compared to communication bandwidth) and relatively large delays were acceptable. As a result, the direct application of such techniques to routing networks would result in unacceptably large overhead – significant hardware requirements or increased delay.

Some interesting work done in DDN flow control is of value here. The "Drop and Throttle" flow control technique developed for the French Data Network (CIGALE) [14] was the first system to implement explicit throttling of the sources suspected of causing congestion. In this system, the utilization of each link was monitored by one adjacent switching node. When the utilization exceeds a certain threshold, all virtual circuits using that link have their traffic rate reduced. This is the "Throttle" aspect of the system. At the same time, the node controlling the congested link begins to discard packets that would otherwise use the congested link. This is the "Drop" aspect of the system. These two techniques coupled together were very effective in both preventing and dissipating network congestion.

In the context of previous flow control techniques, this scheme is a significant departure. The CIGALE system represented a step away from the implicit feedback techniques – full windows or a scarcity of buckets causing the source rate to decrease. CIGALE introduced explicit manipulation of traffic source rates as a method for combating congestion. However, their system was still based on the concept of "virtual circuits" and traffic was throttled on that basis. The explicit manipulation of source traffic rates used in CIGALE sparked some of the ideas for the congestion control mechanism presented in this chapter.

4.3 Traffic in a Routing Network

We consider routing networks in the general context of multiprocessors. However, actual traffic load characteristics will depend factors such as the particular computation model (shared memory, message passing, somewhere in between), the particular program, and even the data on which the program is run. In effect, it is impossible to come up with a single precise model of how the traffic in such machines will behave. We do not try to do so in this thesis. Instead, we will focus on the traffic characteristics that enable us to control hot spot degradation. We describe a range of behavior in the context of two important traffic characteristics – the amount of excess traffic and the elasticity of the traffic. We give examples that illustrate the significance of these characteristics and motivate their realism.

4.4 Excess Traffic Assumption

There are many factors that may influence the design of a network control policy. However, the fundamental assumption underlying any practical flow control system is the existence of excess traffic. If there is no excess traffic, a "hot spot" does not block the network. If there is little other traffic in the network, the hot spot is simply a sequential section in the program execution. A sequential section that may have been introduced by the algorithm or worse yet, the computer system. If we presume the existence of excess traffic, in the presence of a hot spot, the network throughput is degraded. The degradation may be severe, despite the fact that the hot spot traffic is only a small percentage of the overall traffic. Here we describe several possible scenarios to illustrate the excess traffic assumption.

4.4.1 A Scenario for No Excess Traffic

Consider a network that is part of a multiprocessor based on von Neumann style microprocessors. This system is programmed with some form of augmented FORTRAN. This version of FORTRAN has been extended with some parallel constructs – process spawning

and synchronization primitives. One common synchronization primitive is called a Barrier. This construct allows no processor to execute code beyond the Barrier until all processors (or processes) have reached the Barrier. If such a Barrier were implemented with a global lock, then as more and more processors reached the Barrier, that global lock would become a hot spot. There would be less and less other traffic in the network, as more processors reached the barrier and began polling the global lock. This is a good example of a hot spot scenario where there is little or no other traffic to be transmitted by the network (no excess traffic).

4.4.2 Two Scenarios for Excess Traffic

If we consider the same system as above, and look at a lock for a particular popular data structure, the scenario is quite different. Though a single data structure may be quite popular, if a program has a significant amount of parallelism, there will also be a great deal of other traffic traversing the network. This is a scenario in which we are likely to have excess traffic during a hot spot.

Another example, consider a multiprocessor which implements resource management as "function calls" to a centralized or hierarchical resource manager. At certain points in the program, there will be very large amounts of traffic to a regional manager or the single global manager. This traffic may appear to the network as a hot spot. At these times, there may also be significant interprocessor communication and data access. This would result in significant amounts of excess traffic during a hot spot.

4.5 Traffic Elasticity

We define traffic elasticity as the amount of decrease in the source rate of traffic as a function of increasing delay experienced by network traffic. If a small increase in delay caused a sharp decrease in traffic, then we would describe that traffic as extremely elastic.

If a small increase in delay resulted in little or no decrease, we would term that traffic inelastic.

4.5.1 Elastic Traffic

Assuming that the hot spot packets are critical to continuing the execution of the program is one extreme in characterizing the elasticity of the traffic. If the hot spot traffic is essential to the progress of the computation, then no network control policy is going to solve the problem. Unless some highly specialized scheme – such as combining – can cause the hot spot traffic to be served significantly faster, the system will have to wait until the hot spot requests can be processed sequentially. If no such specialized scheme is feasible, then independent of network blockage, network and overall system throughput will decrease. In this case a complete crossbar interconnection would not result in any better performance. The routing network is not the bottleneck.

If the hot spot traffic is critical, the long delay experienced by hot spot packets will eventually cause the nodes (perhaps processors) to idle and hence decrease the source traffic rate. This is due to "feedback" through the program structure. No clever schemes for the network can raise the throughput of useful traffic if the traffic source rate is too low. This situation is not a network problem, rather it is a memory contention problem that is a result of the mapping of the program to the machine. Or more generally, the mapping of a workload to the system. Such degradation is not caused by the network topology or other non-idealities in communication. Rather, similar degradation would result if instantaneous communication were provided.

"Hot spot criticality" says that the traffic is very elastic. Any increase in delay whether it be due to congestion in the network or a drop in system throughput, will result in a sharp decrease in the rate at which traffic is generated. A multiprocessor system naively built from von Neumann processors might be a good example of a system that might exhibit such extremely elastic behavior.

4.5.2 Inelastic Traffic

At another extreme is the assumption of total inelasticity. In this case, the rate at which traffic is generated is not a function of the network performance. The aggregate source rate is constant. Under this assumption, if the network throttles[1] traffic to a set of destinations, the sources still generate traffic for the network at the same overall rate, but the destination distribution changes. This may be a realistic assumption in a Tagged Token Dataflow Machine [1,2]. In such a machine, the "feedback" effects of increased network delay are not very strong. The fundamental limitation in the generation of network traffic is the limited processor pipeline bandwidth. If the bandwidth used by computation generating hot spot traffic decreases, due to the feedback effects of throttling, or other rate limiting techniques, then the bandwidth available to other computation may increase proportionately. If such is the case, then the overall traffic intensity should remain roughly constant.

Network traffic is even more likely to appear inelastic in systems with large numbers of ports – hundreds, or even thousands of ports. In such systems, a very small percentage of the traffic from each source can cause a hot spot. Throttling such a small percentage of the traffic would have little immediate effect on the overall traffic intensity. Only at some later time may the feedback effect come into play – resulting in a decrease in the traffic load.

If the traffic is inelastic, the effects of a hot spot are disastrous. In this case, we have a great deal of traffic that could – if the hot spot were not present – pass through the network quickly. However, almost all of this traffic is delayed significantly. In such a situation, network control techniques may indeed be able to do something to improve the overall throughput of the system.

[1] Reduces by some means at the source.

4.6 A Traffic Model

In considering the spectrum of possible traffic models, we see that the amount of "feedback" is a complicated function of the system architecture, program structure, and run time management. For these reasons, it difficult to model this feedback precisely. Unfortunately, neither simple assumption (completely elastic or inelastic traffic) is a realistic model.

In multiprocessor system, all programs will exhibit some "feedback." Even in a dynamic dataflow machine such as the MIT TTDA[1], where we are able to exchange parallelism for latency, we can only do so to some finite bound. Limited amounts of resources and finite program parallelism will eventually cause the source rate for traffic to drop in response to poor network performance. However, the fact that we can trade some parallelism for tolerance of latency in the TTDA means that for some range of network performance, the traffic will be inelastic. Furthermore, the dataflow instruction scheduling mechanism allows us maximum flexibility in generating network traffic. Thus, the traffic should also be inelastic over some range of throttling.

From a network point of view, one important traffic characteristic is the elasticity. Whether the level of inelasticity simulated is realistic is a question beyond the scope of this thesis. We simply point out that a spectrum of systems exhibiting different levels of elasticity in their traffic can be constructed. This elasticity has a major impact on the effectiveness of the network control techniques developed in this thesis.

4.6.1 Inelastic Traffic Model

For the purposes of study, we assume that traffic for a particular destination, if throttled will be replaced by traffic destined for other destinations. Thus, the traffic intensity applied to the network remains constant in the face of hot spots and throttling. This assumption is based on several points: First, the references to hot spot locations are likely to make up only a very small fraction of the overall source traffic (in systems of reasonably large

size). Second, it may be that only a small part of the overall program contributes traffic to the hot spot. If this part of the program is not tightly coupled to the entire program, other parts of the program will continue to progress. These other parts of the program will introduce traffic in a relatively even manner (at least there should not be a hot spot in the same place!), hence maintaining the overall intensity. Finally, if systems cannot be designed to generate an inelastic traffic load, in order to utilize a significant amount of hardware in a multiprocessor, it will be necessary to have nearly optimal mappings of work to all parts of the system. Any fluctuations or imbalances will likely cause hot spot degradation in the network. Such excellent mappings do not seem feasible at this time. Thus, we must have systems that can generate somewhat inelastic traffic loads.

4.7 A Mechanism for Controlling Hot Spot Degradation

There are two separable problems in dealing with hot spot congestion. It is clear that the only long term solution to a hot spot is to match the rates between the sources and the sinks. Any other scheme is only a stop gap measure to limit the immediate effects of congestion. In the short term, because a significant amount of time is required to dissipate congestion, we would like to implement some scheme for unblocking traffic in the tree of saturated buffers. Otherwise, system performance will still suffer for significant periods of time.

4.7.1 Long Term Rate Matching

Many systems have interleaved memory units. By interleaving, we hope to produce an even pattern of traffic for the network. This means that each of the network inputs may need to send to any of the network outputs at any given time. Thus, any flow control scheme based on virtual circuits or connections is likely to have excessive overhead. To avoid such overhead, we will match source and sink rates by using a scheme in which each switch explicitly throttles those source-destination pairs which could be contributing to the local

congestion. Since the routing function is static, the appropriate pairs to throttle can be determined by the switch's position in the network. This long term rate matching technique addresses the serious degradation caused by long duration hot spots.

In our proposed system, we will throttle each source that can send packets to the congested switch. This simple, crude grain control of the throttling should be adequate to solve the long term hot spot problem. The throttling will reduce the permitted source traffic to a specific set of destinations by a constant factor via probabilistic (or timer driven) methods. Eventually, the throttle will expire, and the source will be free to send packets at the full rate again. Repeated throttling will have a cumulative effect on the sources. As the throttles expire, the congestion causing sources will be able to send traffic to the hot spot sink at the unrestricted rate. These throttles are not expected to occur often, and hence are assumed to be transmitted via some low speed broadcast medium, a bus for example.

4.7.2 Short Term Unblocking

Analysis of the rate of hot spot congestion tells us that the network backs up very rapidly. If each router has buffer space for five packets (for each input link), then the time to congest the network in the presence of a serious hot spot would be $\approx 50 - 100$ packet times. This is a very short period. None of the detection schemes we have investigated will be able to detect a hot spot reliably before it congests many stages of the network. Therefore, it is likely that sufficient traffic to congest the network will already be committed before we can communicate the information about the hot spot and act on it.

If we are unable to prevent hot spot congestion, once it occurs we would like to dissipate it as rapidly as possible. To that end, we propose the following scheme: As soon as a switch detects a hot spot, it begins to misroute some of the packets. Only packets destined for the "hot side" of the switch are misrouted. Further, packets are only misrouted when the "hot" output for this switch is in use. This misrouting continues for some fixed period. We adjust this period experimentally so that congestion is dissipated with only a small amount

of misrouting. Short term unblocking addresses the degradation caused by brief, intense hot spots. With misrouting, the effects of such hot spots should be minimized.

Misrouting should have a very beneficial effect the dissipation of congestion. It should increase the "effective bandwidth" of the hot spot in that hot packets will be removed from the network at a higher rate. This should allow the congestion to disperse rapidly. We will verify this by simulation of the model. Furthermore, a misrouting scheme makes no assumptions about the network being faster than the traffic sinks. This assumption is implicit in any approaches that place additional buffering at all output ports of the network. Rather, we are using the bandwidth of another link to supplement the bandwidth of the hot spot link. This misrouting scheme is attractive because it seems to be scalable in some sense because it makes use of the buffering in the network. The amount of this buffering increases with network size.

The primary drawback of misrouting is that delivery is indeterminate. Given that packets can be misrouted, then a serious of hot spots in the same place could result in packets being misrouted many times. This could result in unacceptably long delivery times. This is a minor difficulty and can be solved by only allowing packets to be misrouted once.

4.8 Summary

Two assumptions are necessary before a network control system will be able to address the problem of hot spots. We must first assume that when a hot spot occurs, there is indeed excess traffic that, if the hot spot were not present, could be transmitted through the network. The second assumption is that the traffic is somewhat inelastic in response to delay to the hot spot traffic. Such delay is unavoidable, because the hot spot server must serve all of the hot spot packets sequentially. This assumption is roughly equivalent to saying that there is excess traffic that does not depend directly on the rapid serving of hot spot requests. A traffic model is defined for simulation study.

The hot spot congestion problem was divided into two parts: long term rate matching and short term network unblocking. The proposed solution consists of two techniques. We throttle sources to solve the long term rate mismatch problem. In addition, we misroute packets to solve the short term congestion problem. These two techniques are proposed together, as niether is very effective alone. The results of simulating the proposed solution are presented in Chapter 5.

Chapter 5

Simulation Results

Having proposed a solution to addressing the hot spot problem, we would like to analyze its effectiveness. Such analysis is difficult because of the complex behavior of both the network and the flow control system. Of course, their interaction further complicates the problem. At present, attempts to analyze the flow control system's effects have not yet been completed. Consequently, we resort to simulation to demonstrate the effectiveness of the the proposed system.

Before presenting simulation results, we briefly discuss a triggering mechanism for our flow control system. The sensitivity and simplicity of this mechanism is fundamental to the practicality of the overall system. After that, we present several types of simulation results. We first consider the steady state performance of the flow control system. Subsequently, throughput and delay statistics for the controlled network in the presence of hot spots are reported. These statistics confirm our prognosis that the proposed flow control system is effective with the Inelastic Traffic Model.

In order to give the reader some insight into the temporal behavior of the network, we then proceed with a case study of a 64-port network. In the case study, we consider two scenarios: the system performance, under a hot spot load with no flow control, and under a hot spot load with our flow control system in place. Histograms of various network characteristics are presented. These histograms show that the proposed flow control system

is be very effective in this example. Flow control significantly improved system performance during the hot spot. The system performance approaches levels measured under balanced traffic loads.

After the case study, the results from a more thorough study of our flow control system are presented. This study involved several different network sizes, hot spot intensities, and hot spot durations. These studies showed several important features of the proposed system. First, the flow control system responds rapidly – maintaining throughput even for brief hot spots. Second, the system was effective for a wide range of hot spot intensities and durations. Finally, the system not only maintained high overall system throughput, but also kept the hot spot sink nearly saturated in all cases – an important factor in keeping hot spot durations to a minimum.

5.1 A Detection Mechanism

An important issue in developing an effective network management system is the detection of hot spots. The problem is difficult because very small source imbalances can lead to quite severe hot spots. Unfortunately, at a single source, these small imbalances are indistinguishable from normal statistical fluctuations in the traffic. Detection of hot spots is a crucial problem, but was not the primary focus of this thesis. An effective, though probably not optimal scheme was developed. This scheme is described below. For more complete documentation of this triggering system, and the other systems that were considered, the reader is referred to [4].

Because routing networks are typically used in high speed applications, collecting global information about source traffic rates is not a feasible alternative. We choose instead to collect information at each of the routers. This means that the information can be collected very cheaply (no communication is necessary for this component of the system) and the collection mechanism can be captured in hardware and replicated along with the routers.

In order to detect hot spots, each router keeps statistics that roughly correspond to the

number of packets that it has sent in either direction (remember these are 2-by-2 routers) in the last x time steps. The interested reader is referred to [4] for a detailed description of this mechanism. For the purposes of the simulation, we used a synchronous model, but it should be easily extensible to the asynchronous case. If the proportion of packets sent in one direction exceeds some threshold, then the switch enters a warning state and is deemed to have detected a hot spot. At low traffic levels, the statistics kept are slightly different. At such levels, a compensation mechanism inserts null entries – "dummy" records of packets being sent – for idle switch cycles to prevent alarms. This reflects the fact that, at low traffic levels, imbalances may not indicate serious congestion.

The detection of a hot spot by a switch gives us two kinds of information. From the position of the switch, we can deduce which sources could be contributing to the hot spot, and we can also deduce which destinations could be the hot spot. This information allows us some coarse grain control in how we throttle the source traffic. In fact, knowing the switch position in the network, and the link which is overloaded allows us to isolate the hot spot to $\frac{1}{n}$ of the source destination pairs. Thus, if the traffic is almost balanced, then our throttling will affect only $\frac{1}{n}$ of the overall system traffic. In larger systems, we gain finer and finer grain control.

5.2 Steady State Performance of the System

In order for a flow control system to effectively address the problem of hot spots, it must eventually match the source and sink rates for the hot spot packets. If it can do so, then it will be able to maintain system throughput in the presence of a hot spot. A simulation study was performed to investigate the steady state performance of the controlled system. These simulations we performed using the Inelastic Traffic Model. More complete results from this study are presented in Table A.1.

These simulations showed that it was indeed possible to maintain system throughput

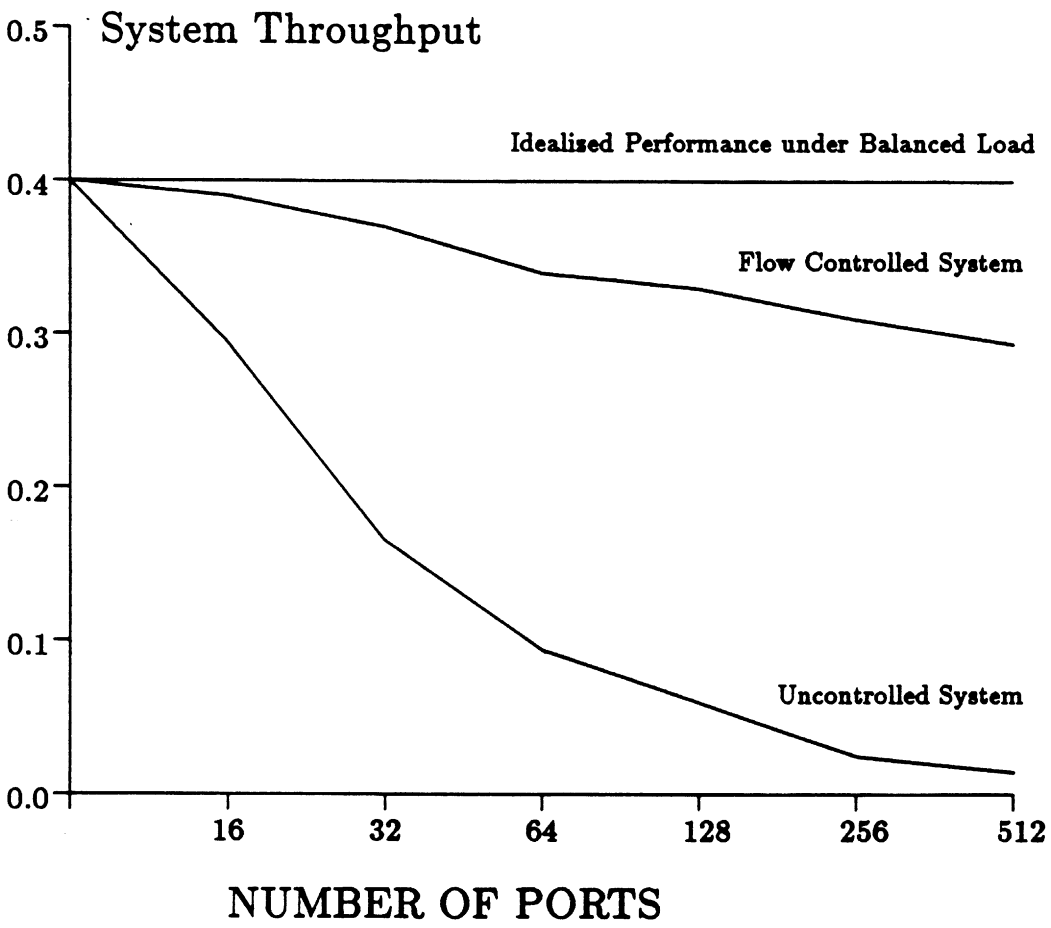

Figure 5.1: Steady State Throughput with a 16% Hot Spot

in the presence of hot spots. In most cases, the proposed system was able to maintain system throughput many times that sustainable in an uncontrolled routing network. The throughput data is presented in Figure 5.1. The system was also able to effectively dissipate congestion within the network. This is clear from the statistics for the average delay. The delay data is presented in Figure 5.2. The average delay in the steady state is very close to the value for the network under a homogeneous traffic load. These simulation results show that the proposed system will be very effective in solving the problem of long duration hot spots.

5.3 A Case Study in Controlling Hot Spot Degradation

Analysis of routing networks in the presence of hot spots gives us insight into their behavior. However, our criteria for declaring networks congested and uncongested are somewhat arbitrary. As a result, our predictions may be misleading. What we really would like to see is the dynamic behavior of the uncontrolled and flow controlled network with a hot spot traffic load. Due to statistical fluctuations (mostly due the stochastic method of traffic generation), any single network simulation run would probably not be very informative. In an attempt to reduce the amount of this variation, and extract the common underlying features of the network behavior, we have averaged the results of many runs (in this section, we averaged 100 iterations). The results in this section confirm what our intuition told us was going on in the network. Further, these results show dramatically the effectiveness of the proposed flow control system.

5.3.1 System Performance without Flow Control

As a reference we first present the graphs for a network without flow control. However, the following description applies to all graphs in this section (5.3). The simulated network is a 64-port network built from 2-by-2 routers. The average load on the inputs, a, is 0.4.

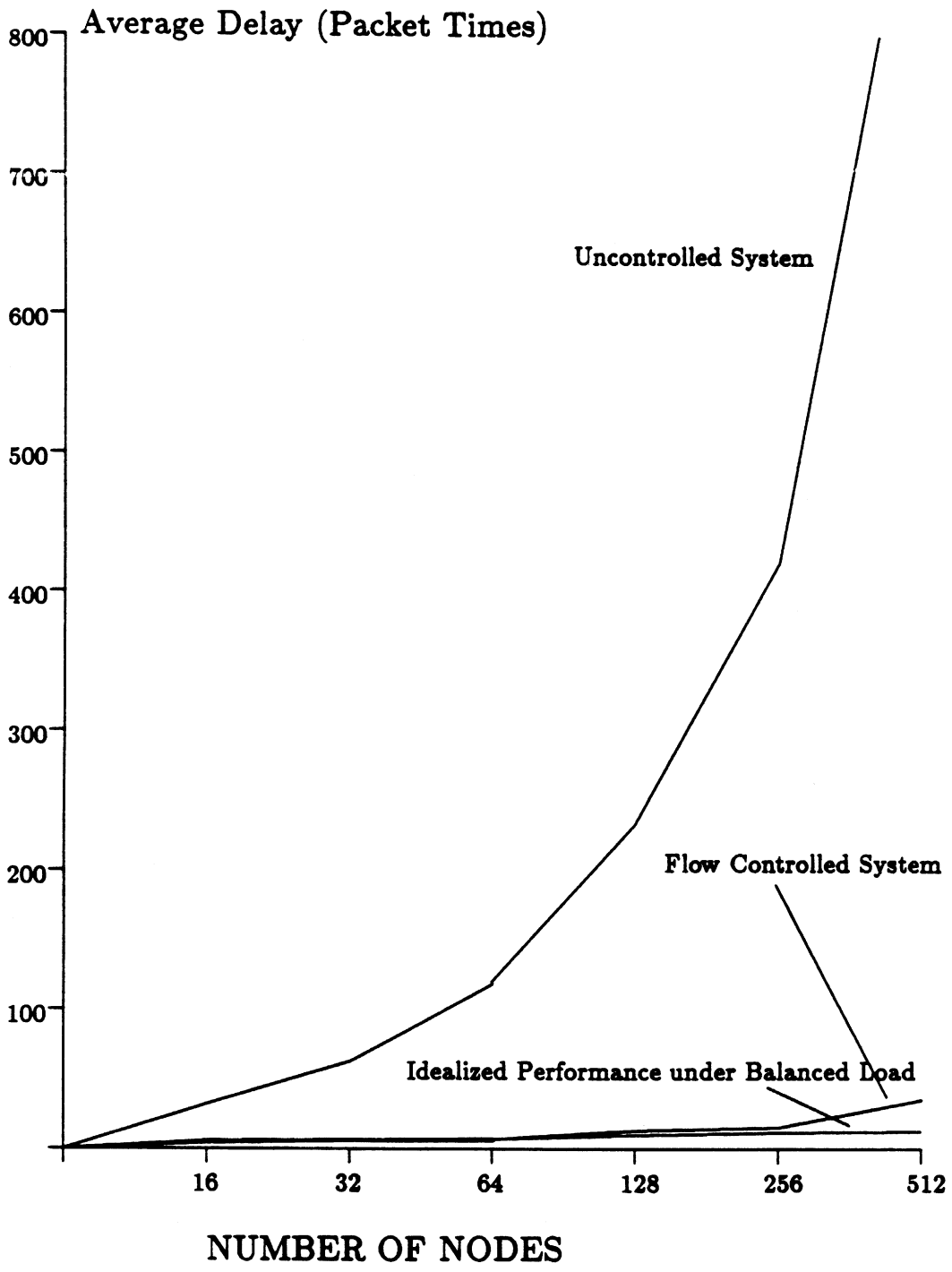

Figure 5.2: Steady State Delay with a 16% Hot Spot

The x-axes of all of the graphs in this section are marked in packet times, where a packet time is the interval required to send a packet from one switch to the next. The hot spot is introduced at time 100 and removed at time 800. The hot spot intensity, h, is 0.16 or 16%. This imbalance, coupled with the background traffic intensity, implies that the demand for the hot spot sink will be $\approx$ 4.432 times its maximum capacity. This corresponds to an elevation of $\approx$ 11 times the average level. In this section, we will consider the system throughput and average delay as figures of merit for the network performance.

Uncontrolled System Throughput The unnormalized system throughput as a function of time is shown in Figure 5.3. The performance of the uncontrolled network is quite poor after the hot spot is introduced. As we can see, shortly after the hot spot introduction at time 100, the system throughput drops precipitously. Well before time 200, system throughput has fallen to $\approx \frac{1}{5}$ of its sustainable value under balanced traffic. This low level is the steady state value of the system throughput in the presence of this 16% hot spot.

Though the network becomes congested in much less than 100 packet times, the throughput graph shows clearly that dissipating the congestion takes quite a bit longer. The hot spot is removed at time 800, but the system takes until about time 1000 before throughput recovers to balanced traffic levels. This confirms our analytic results that showed hot spot dissipation to be much more time consuming than congestion for moderate to severe hot spots. Around time 1000, we see that the system throughput recovers to balanced traffic levels[1].

[1]Not only does it attain the balanced traffic throughput levels, it in fact overshoots slightly. This phenomenon is not yet fully understood, but may arise from the simple mechanism described below. We note that the average system load is less than the maximum sustainable. In this configuration, the network can sustain somewhere around 0.6 normalized throughput per input. The overshoot phenomenon is not observed when the background load is this high.

As the hot spot packets leave the stages of saturating buffers, they are replaced by a more balanced mix of traffic. However, because the throughput of the system is still low, this traffic also backs up in these buffers. When the critical few stages of the network are unblocked, the system throughput recovers. At this point, the buffered packets in the network are like "increased load" and allow the network to have higher throughput while the backlog of packets is worked off. The overshoot is an artifact of our definition of throughput, which is the number of packets *delivered* to the sinks per unit time. It is not observed when we redefine throughput as the number of packets successfully submitted to the network per unit time.

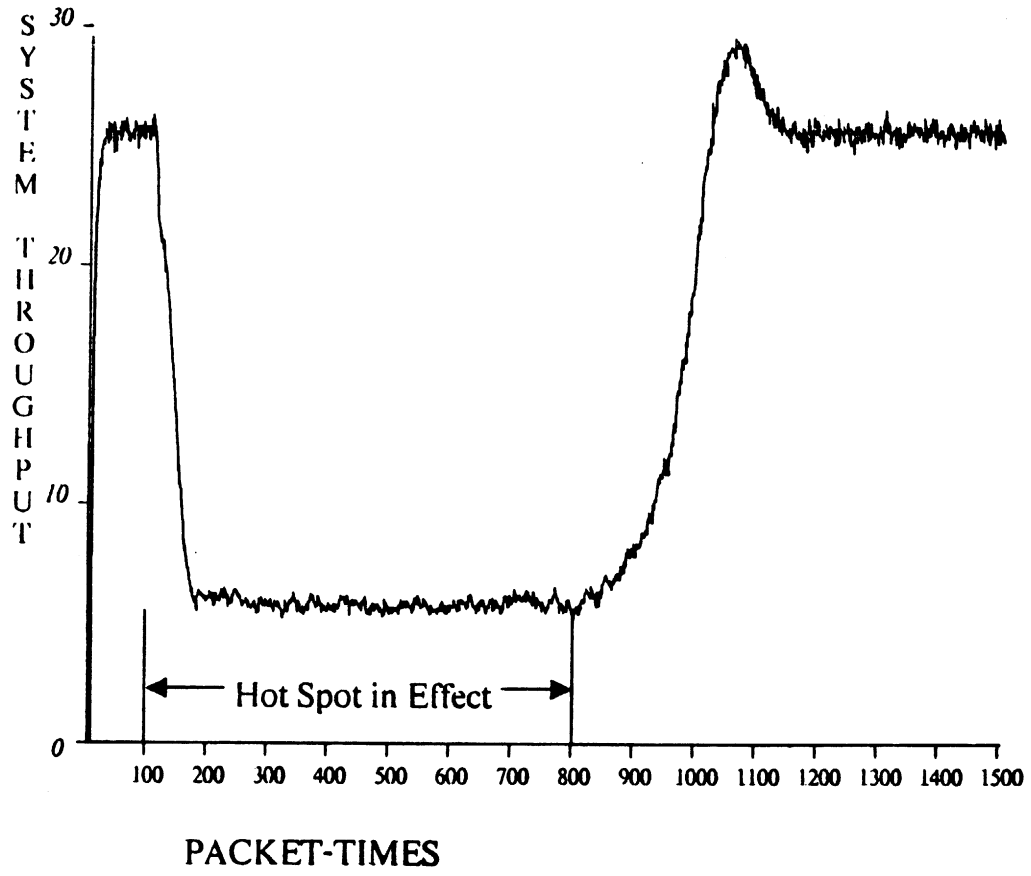

Figure 5.3: Throughput of an Uncontrolled System with a 16% Hot Spot

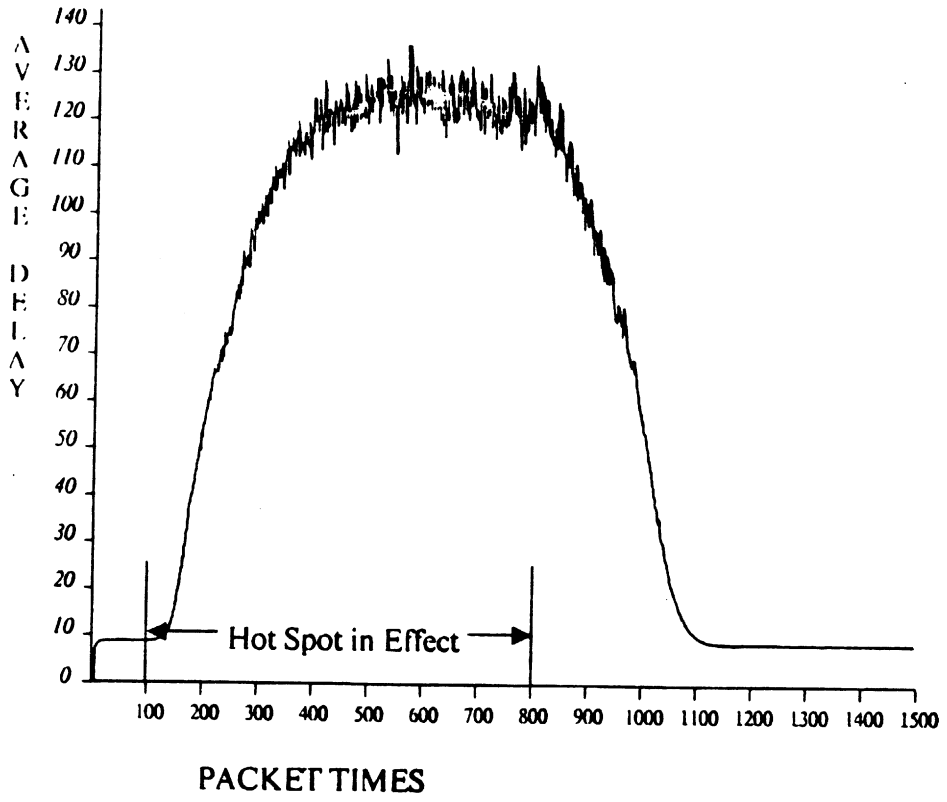

Figure 5.4: Delay in an Uncontrolled System with a 16% Hot Spot

Uncontrolled System Average Delay In Figure 5.4, we see the average delay of the delivered packets as a function of time. We see that the average delay rises rapidly[2] from a value around nine packet times under the balanced load to about 120 packet times. This is more than an order of magnitude increase in delay. It is also interesting to note that this does not even account for the increased delay due to the decreased throughput of the network (traffic waiting to enter the network).

After the hot spot is removed, the average delay figure gradually decreases as the congestion is removed from the system and all of the older packets are delivered. The average delay does not recover to balanced traffic levels until about time 1100 – around 300 packet

[2]It cannot rise any faster than linearly with slope 1 for any substantial period.

times after the hot spot has been removed.

5.3.2 System Performance with Flow Control

For comparison, we simulated the same network configuration and traffic load[3] as in the uncontrolled case, except that we implemented the proposed flow control scheme. The traffic was assumed to be completely inelastic. In this section, we present the comparable throughput and delay graphs for the controlled system. We also present a few additional graphs which reflect characteristics of the flow control system. These graphs give us some insight concerning the amount overhead a real implementation of the proposed might incur.

Controlled System Throughput When some packets are misrouted, we must be careful to calculate system throughput properly. We count only those packets that are received at the correct destination in the throughput. Even with this accounting change, the flow control system did an excellent job of maintaining system throughput in the presence of a hot spot. The throughput graph is presented in Figure 5.5. The throughput dipped briefly after the hot spot was introduced but was quickly restored to near balanced traffic levels. In fact, within 100 packet times, the system was operating at $\approx$ 85% of balanced traffic levels.

This high level of performance was maintained for the duration of the hot spot. After the hot spot was removed, the system throughput returned to balanced traffic levels almost immediately. No overshoot was observed. This was attributed to the fact that network congestion had been thoroughly dissipated. With relatively few packets buffered in the network, there is no "increased load" to temporarily elevate the system throughput and cause the overshoot.

[3]Here we mean the same statistical traffic load, not the exact same load.

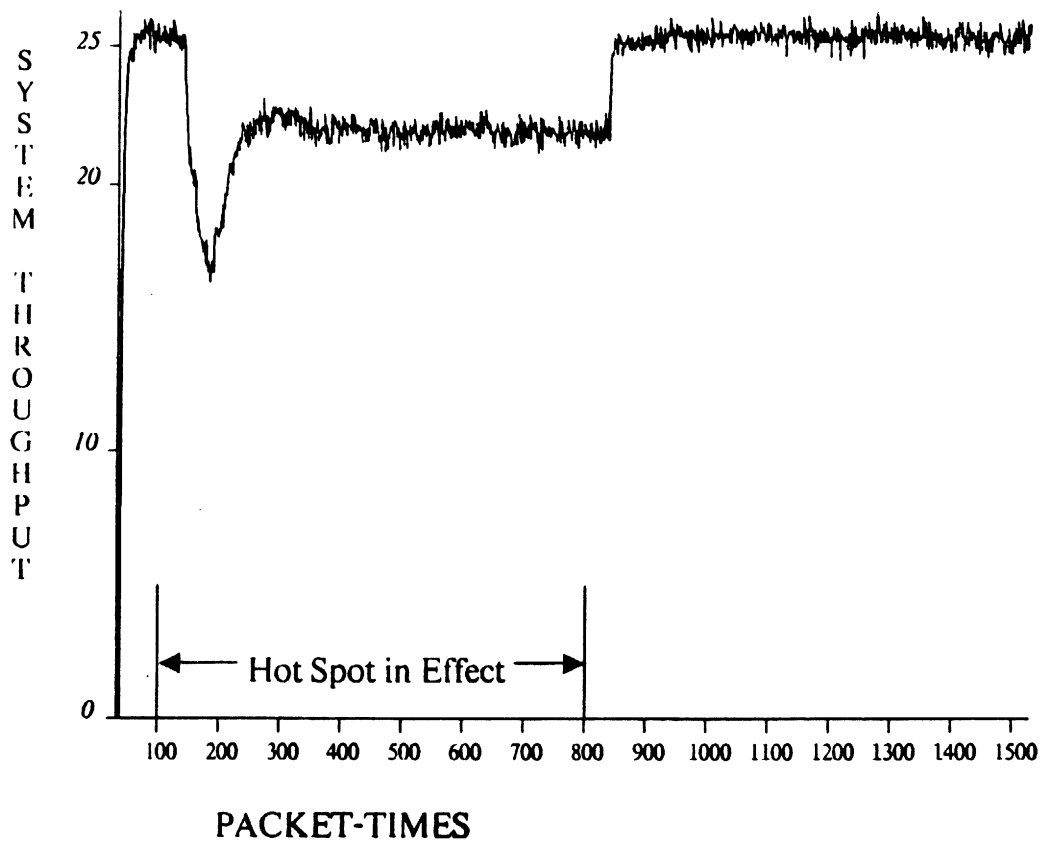

Figure 5.5: Throughput of a Flow Controlled System with a 16% Hot Spot

Controlled System Average Delay With misrouting, calculating the average delay is tricky. We computed the average delay by averaging the ages of all of the correctly routed packets received in a packet time. This is somewhat deceptive because misrouted packets were not resubmitted to the network. As a result, the average delay statistics presented should be considered to be somewhat optimistic for what is likely to occur in an actual network[4]. Because the number of misrouted packets is only a small portion of the total traffic, the average delay statistics should not change greatly if the misrouted packets are resubmitted.

The average delay statistics were indeed very encouraging (see Figure 5.6). After the hot spot was introduced, the delay increased sharply, but never came anywhere close to the levels observed in the uncontrolled case. The average delay levels in this case peaked at below 20 packet times. This is only about twice the ordinary delay levels. Shortly after the peak, average delay levels dropped rapidly and approached balanced traffic levels. This drop tells us that congestion in the network is being dissipated. The steepness of the drop tells us that the network is dissipating congestion rapidly – our short term unblocking scheme is effective. When the delay levels out, we can see that the system has reached a stable operating point. Further, this operating point provides performance far superior to the uncontrolled system. This performance approaches that possible with a balanced traffic load.

Characteristics of the Flow Control System In order to evaluate the flow control system, we collected several other important statistics for the network. These statistics are the number of misrouted packets, the number of warnings in effect, and the utilization of the various sinks (network outputs). These figures tell us a little about the overhead incurred by the flow control scheme, and as well as how it is modifying the delivered traffic distribution.

The misrouted packets graph (Figure 5.7) tells us how often packets are going to have

[4]It should also be noted that resubmission of misrouted packets is also likely to improve system throughput because it will elevate the applied load slightly.

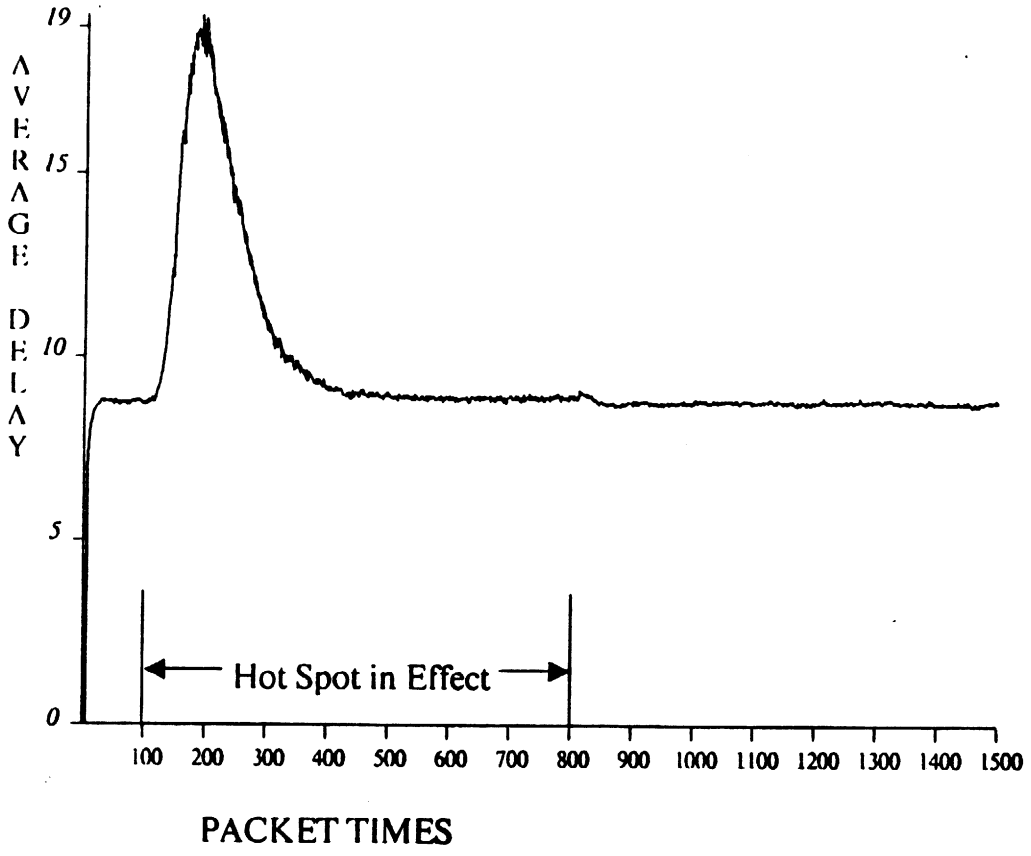

Figure 5.6: Delay in a Flow Controlled System with a 16% Hot Spot

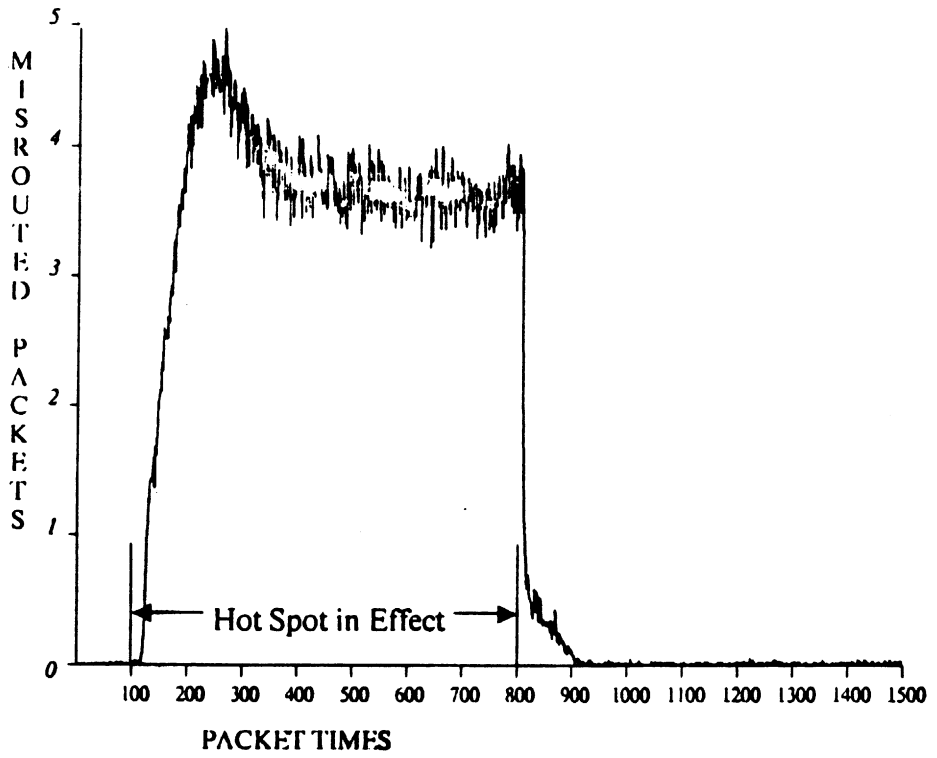

Figure 5.7: Misrouted Packets in a Flow Controlled System with a 16% Hot Spot

to be resubmitted to the network. In this example, even with a fairly severe hot spot, only about 15% of the traffic is misrouted. This leads us to believe that the average delay statistics collected may be very close to what is actually achievable in the framework of this flow control system.

The number of warnings graph (Figure 5.8) tells us about the system overhead. Each warning lasts for 100 packet times in this simulation, so a steady value of 20 warnings tells us that about 20 times in 100 time units, some switch gives out a warning. This is not a very large communication requirement. For example, it could easily be satisfied if all of the switches and sources were connected to some sort of low speed bus.

The sink throughputs graph (Figure 5.9) tells us how the flow control system is affecting

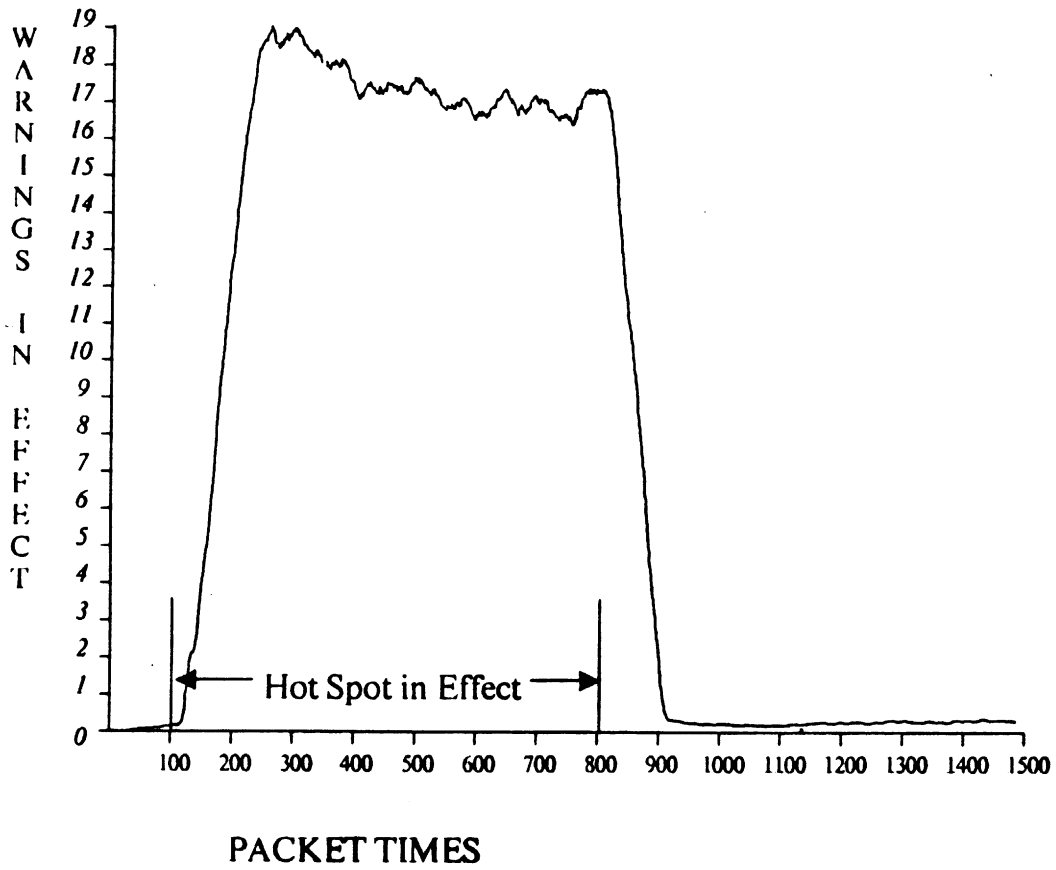

Figure 5.8: Warnings in a 64-port System with a 16% Hot Spot

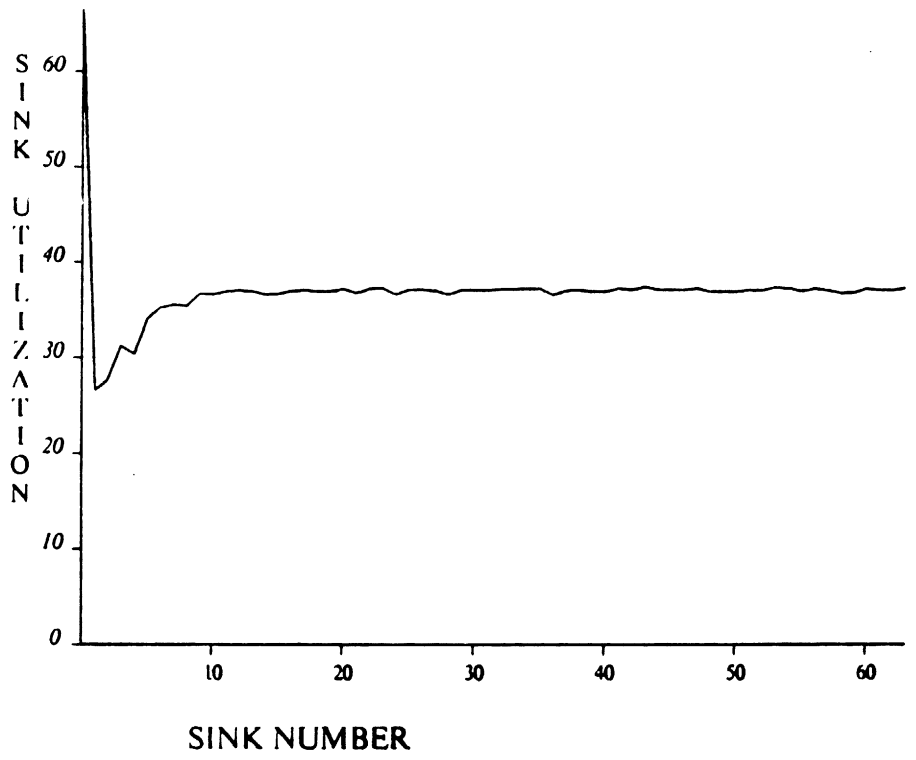

Figure 5.9: Throughputs of Different Packet Sinks

the distribution of the delivered traffic. The offered load, if there were no throttling, would be perfectly even with a spike at sink 0 (the hot spot sink). However, we can see that the sinks close to the hot sink in the network (sinks 1, 2, 4, etc. in an indirect-n-cube) are suffering some loss of throughput due to the coarse grain misrouting and throttling of the flow control system. Despite that fact, all of the sinks, including the ones close to the hot sink, experience greater throughput than in the uncontrolled case.

In summary, our case study showed the proposed flow control scheme to be effective in controlling hot spot congestion. In the example simulated, the network performance was restored to near balanced traffic levels. Furthermore, simulation statistics lead us to believe that the overhead for this system is need not be very large.

5.4 Overall System Performance Experiments

5.4.1 Motivation and Structure of Experiments

In many cases, hot spots will be due to transient imbalances in the overall traffic load. Under such conditions, one might expect hot spots of widely varying durations. The steady state performance, detailed earlier, tells us that our flow control system will perform well in the presence of long duration hot spots. However, we expect many more hot spots of short and medium duration. This means that the network's performance in the presence of short duration hot spots will be crucial to network performance. Consequently, to explore the effectiveness of our system for shorter hot spots,, we performed a series of simulations to study the effects of various duration hot spots ($\approx$ 200 to $\approx$ 600 packet times).

We simulated a variety of network sizes, hot spot intensities, and hot spot durations. Unfortunately, due to excessive computational requirements, the largest network we used in this study had 64 inputs. We have simulated larger networks, but they required so much computer time as to make systematic study of such network sizes impractical with our Symbolics 3600 based simulator. It seems, however, that networks of various sizes exhibit similar behavior under equal "hot spot" load – the effective load that the hot spot output experiences. Using this rule it is possible to extrapolate and predict performance for larger networks on a reasonable range of hot spot intensities.

We varied the hot spot intensities from very low $\approx$ 1.36 times the maximum capacity of the output link to $\approx$ 4.432 times the maximum capacity of the output link. The background traffic level was 40% of maximum capacity, so these levels represent $\approx$ 3 times the average level at the low and and $\approx$ 11 times the average level at the high end.

The hot spot durations were varied for several reasons. We wanted to see how long it took the network control system to respond to the hot spot. We also wanted to see what percentage of time a particular intensity hot spot would have to be present before it would significantly affect the network performance. Finally, if we could determine how long

hot spots need to be before they affect the system significantly, then we could evaluate the response time of our system in that context. Another important observation is that hot spot durations also affect the proportion of time the system experiences hot spots. This means that the longer duration hot spot statistics also correspond to a view of traffic where hot spots are present in the system a larger proportion of the time. This behavior is reflected in our experiments.

5.4.2 Results

The simulation results[5] (shown in Table A.2) showed that the proposed system responded to hot spots rapidly and maintained throughput at close to maximum levels (the source input rate). This can be seen by considering a particular network size and hot spot intensity and varying the hot spot duration. In all cases, the network throughput falls off much more slowly with the flow control system than without. The network throughput under low intensity hot spots was very close to the 0.40 maximum possible. This limit is imposed by the source traffic rate. An important point to note is the low overhead that the flow control system imposed on the network. Throughout our simulations, we found that very few false alarms occurred. This makes the system quite attractive as the additional hardware required by the proposed scheme is minimal.

Simulation results (presented in Table A.3) show that our flow control system is very effective in controlling degradation due to brief hot spots. The controlled network maintained throughput effectively in the presence of brief hot spots. The small decreases in throughput for introduction of a 200 packet time hot spot tell us that our network management system does in fact respond rapidly to hot spot imbalances. In all cases, the uncontrolled system experienced much more serious throughput degradation.

The graphs in Figure 5.10 show the decrease in overall system throughput as a function

[5] All simulation results in this section are the result of runs of 1000 packet times long, averaging data from 20 iterations.

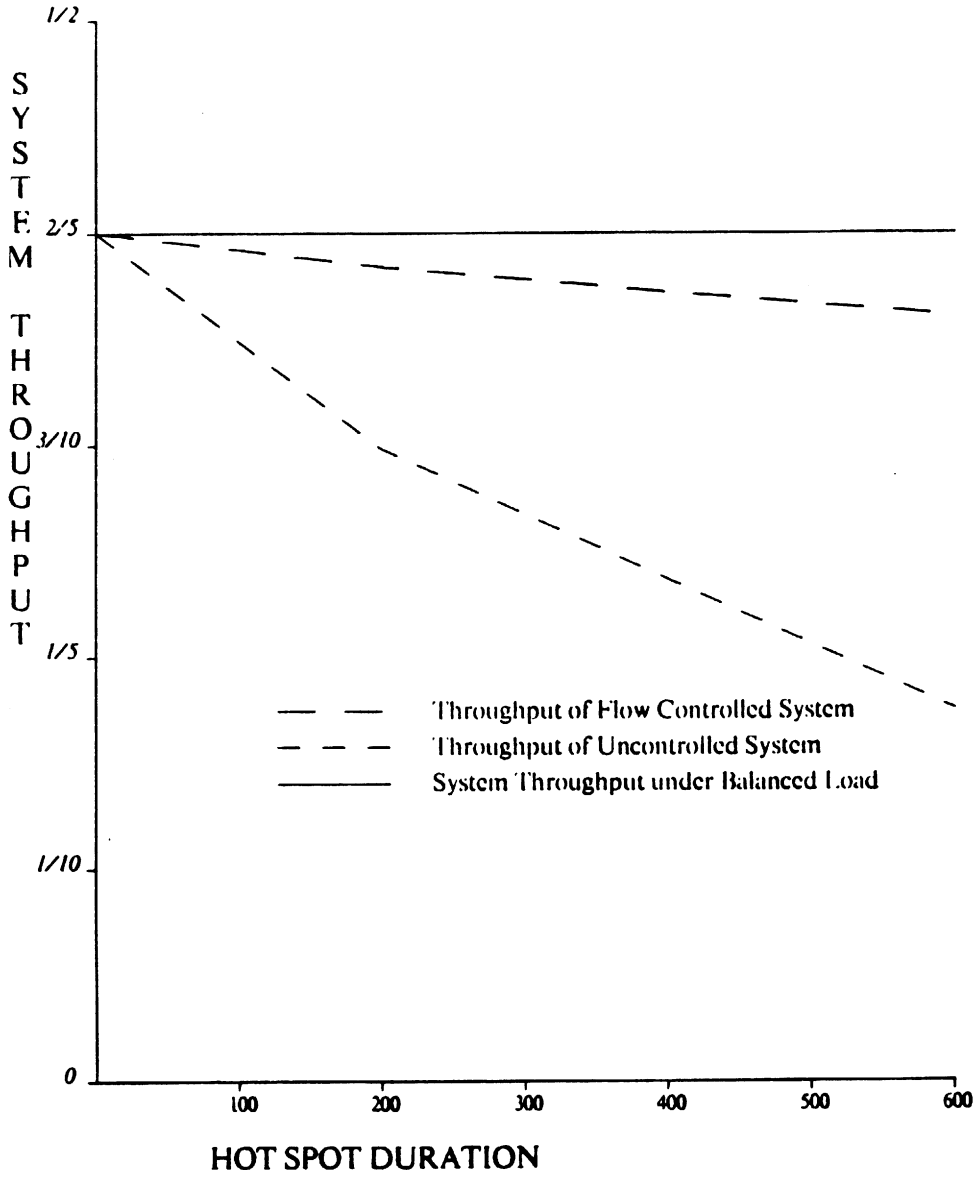

Figure 5.10: System Throughput vs. Hot Spot Duration, a 64-port Network

of hot spot duration. In Figure 5.10, we only consider a 64-port network with a 16% hot spot. The throughput of the flow controlled system and the uncontrolled system under this traffic load are plotted. As a reference the idealized system throughput for balanced traffic is also plotted. These results show graphically that the decrease in network throughput with an uncontrolled network can be quite dramatic. They also show that the flow control system significantly lessened the rate of decrease in throughput.

The controlled network was also able to maintain average delay at a level very close to those achievable in balanced traffic. This is very encouraging because this delay is much lower than that observed in the uncontrolled system. The low average delay recorded when the network management system was used also tells us that congestion was dissipated effectively. Such low levels would not be possible if significant congestion persisted in the network. This further affirms what we saw in the Case Study presented in Section 5.3. The network management system is able to respond to the hot spots rapidly and keep congestion to a minimum.

The graphs in Figure 5.11 show the average delay for delivered packets as a function of the hot spot duration. In Figure 5.11, we consider only a 64-port network, in the presence of an 16% hot spot. The flow controlled and uncontrolled systems are plotted. The average network delay under a balanced traffic load is also plotted for reference. The delay increases sharply for the uncontrolled system. There are several important things to note with regard to the delay measurements: misrouted packets are not averaged in to the delay measurements and misrouted packets are not resubmitted to the network. Thus, average delay in the flow controlled system is probably somewhat understated.

5.5 Hot Spot Sink Throughput

We have seen that the proposed flow control system is able to maintain good network performance in the presence of a hot spot. Of course, as we discussed earlier, the effectiveness of the proposed system depends crucially on the Inelastic Traffic Assumption. In our sim-

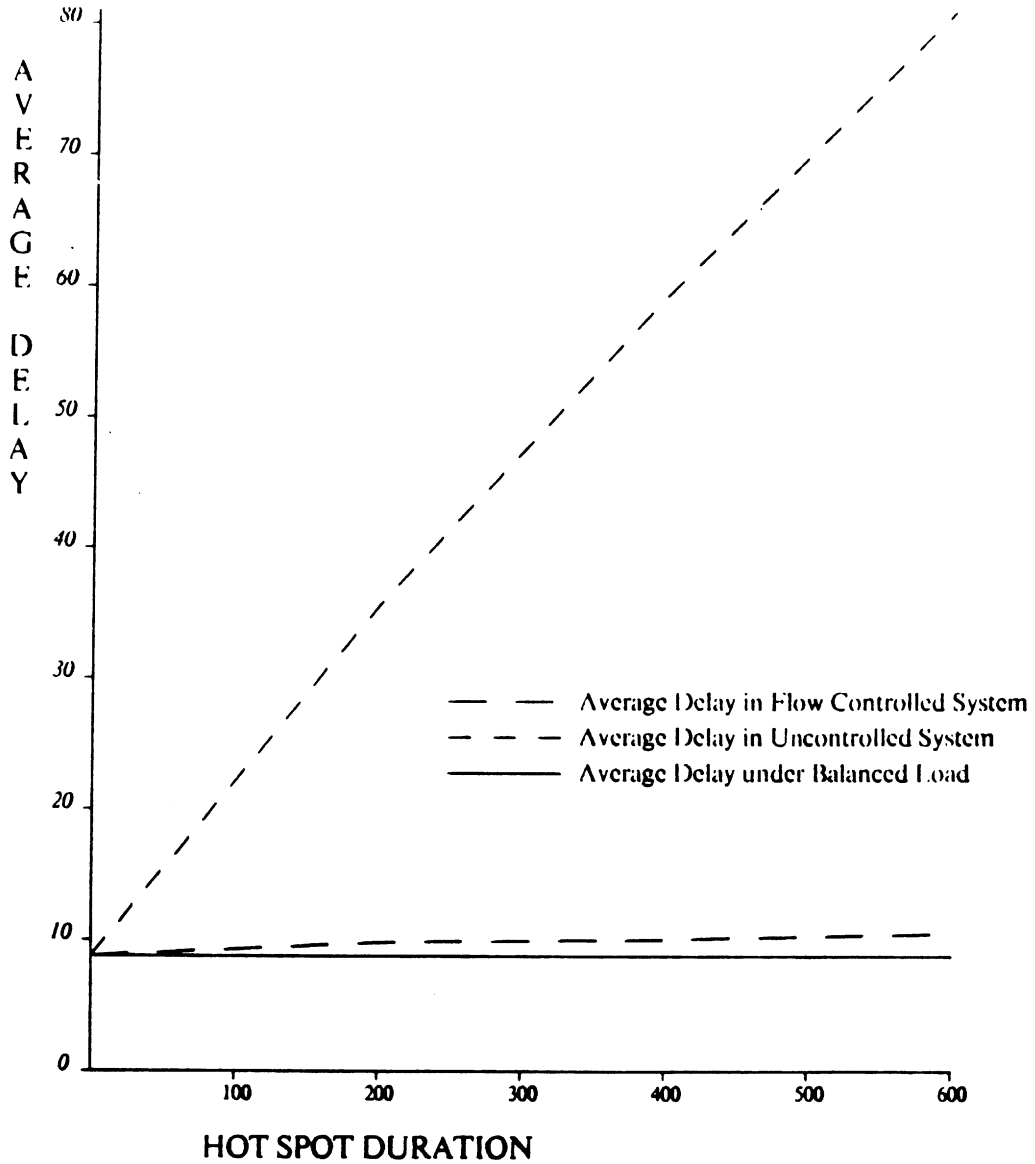

Figure 5.11: Average Delay vs. Hot Spot Duration, a 64-port Network

ulations, we assumed that the traffic presented to the network is completely inelastic. In an actual system (even a Tagged Token Dataflow Machine), the traffic is only inelastic over some finite time period. This means that if we severely suppress (throttle) the traffic for a particular destination, this throttling will eventually cause the overall traffic rate to decrease. As a direct consequence, the system performance will be degraded.

We conjecture that system traffic will only be inelastic over some finite period. Furthermore, the length of this period will vary over time. Therefore, we see that if a temporary imbalance in the load occurs, it is important for the network to "work off" this imbalance as soon as possible[6]. Otherwise, the persistent imbalance will eventually cause the traffic to behave elastically. And, the resulting reduction in traffic rate will degrade system performance.

In more practical terms, wanting to "work off" imbalances as fast as possible means that we would like to utilize the oversubscribed network outputs at as close to 100% as is possible. If we can achieve close to 100% utilization, we see that the excess traffic is being "worked off" as fast as the system can service it. In the context of hot spots, this discussion implies that we want the hot spot sink utilization to be as close to 100% as possible during the hot spot. In this section, we compute the "effective" hot spot sink utilization during the hot spot from the overall hot spot sink utilization. We then compare this "effective" utilization with the goal of 100% utilization.

In order to calculate the "effective" hot spot sink utilization during the hot spot, we must make some assumption about the hot spot sink utilization when the traffic is balanced. Because we know that in the flow controlled system hot spot congestion is dissipated rapidly, we can approximate the balanced traffic utilization by, a, the average network load. Compensating for the length of the hot spot, denoted by d, we get the following expression for the hot spot sink utilization during the hot spot:

[6]In this context, "working off" an imbalance means sustaining different utilizations for different network outputs. After an interval, the imbalance will have been "worked off" and the traffic presented to the network (with no throttling) will have an approximately balanced destination distribution.

$$HSUTIL(a,d) = \frac{u-(l-d)a}{d} \tag{5.1}$$

The symbol l denotes the length of the entire simulation run. And, the symbol u denotes the average hot spot sink utilization for the entire simulation run.

Simulated results for hot spot sink utilization for various network sizes, hot spot intensities, and hot spot durations are shown in Table 5.1. Hot spot sink utilizations are converted to "effective" utilizations using the formula in Equation 5.1. As we discussed earlier, an effective utilization of 100% is our goal. As one can see, our system approaches this level of performance for a wide variety of network sizes and hot spot intensities. We see that hot spot imbalances are worked off very promptly. And consequently, very little usable hot sink bandwidth is being wasted. This is good news indeed because it means that our system is not excessively throttling traffic to the hot spots.

5.6 Summary of Results

The simulation results presented in this chapter have given us a great deal of information about the effectiveness of the proposed flow control system. We have seen that the performance of the proposed flow control system under the Inelastic Traffic Model is quite good. Network throughput is maintained at close to balanced traffic levels over a variety of network sizes and hot spot intensities. The average delay experienced by packets is also kept to near balanced traffic levels. This tells us that congestion in the network is being dissipated effectively.

Furthermore, the increased overhead due to the flow control system appears to be minimal. Even with a severe hot spot, less than 15% of the traffic was misrouted. Under the same conditions, only an average of $\approx$ 20 warnings were in effect. This corresponds to a request rate of only 0.2 requests per packet time. This is the aggregate rate for all of the

# of Nodes	Hot Spot Intensity	Hot Spot Duration	Hot Spot Sink Throughput	Effective Utilization
16	0.16	200	47.75	78.75
16	0.16	400	55.58	78.95
16	0.16	600	63.48	79.13
16	0.32	200	48.86	84.30
16	0.32	400	59.20	88.00
16	0.32	600	68.77	87.95
32	0.08	200	49.64	88.20
32	0.08	400	58.65	86.62
32	0.08	600	68.00	86.67
32	0.16	200	51.49	97.45
32	0.16	400	61.33	93.32
32	0.16	600	72.00	93.33
32	0.32	200	51.07	95.35
32	0.32	400	63.15	97.88
32	0.32	600	74.13	96.88
64	0.04	200	50.75	93.75
64	0.04	400	60.17	90.43
64	0.04	600	70.18	90.30
64	0.08	200	51.58	97.90
64	0.08	400	62.16	95.40
64	0.08	600	73.30	95.50
64	0.16	200	51.81	99.05
64	0.16	400	63.18	97.95
64	0.16	600	74.93	98.22

Table 5.1: Hot Spot Sink Utilizations with Short Hot Spots

switches in the network and could easily be supported by a single bus.

In addition, we see that despite the throttling of source traffic, the hot spot sink is still achieving high levels of utilization during the hot spot. We are not trading only a small amount of the usable hot output bandwidth for a large increase in overall network performance. As a result, our flow control system is not likely to prolong hot spots significantly by lowering the utilization of the hot spot sink.

Chapter 6

Conclusions

6.1 Summary

Significant progress has been made in understanding and addressing the hot spot problem. In this thesis, we defined the hot spot problem clearly and outlined several mechanisms by which hot spots are likely to arise. These mechanisms are both plausible and varied. Further, early multiprocessor simulation results have demonstrated evidence of hot spots. All of this leads us to believe that hot spots may be a problem in large scale multiprocessors.

Following that, we defined and analyzed the mechanisms behind hot spot degradation, hot spot congestion, and dissipation of hot spot congestion. This analysis has yielded an accurate model of hot spot effects. These effects can be very significant at very low levels of source traffic imbalance. These models for hot spot network behavior were verified via extensive simulation. Simulation results showed, as was previously reported, that hot spot degradation is very severe. Such degradation can occur very rapidly (the onset of hot spot congestion is very short) even for low intensity hot spots. This degradation lingers long after the imbalance has passed. Thus, a hot spot can congest networks rapidly and degrade their performance severely. In addition, network performance only recovers from this degradation slowly.

Traffic models for multiprocessors were discussed and we chose one for extensive study. Within the framework of this traffic model, a flow control system was proposed and motivated. This flow control scheme was simulated in a variety of network configurations and under a variety of different hot spot and non-hot spot loads. The flow control system performed quite well in the vast majority of these configurations. In the presence of hot spots, system throughput was maintained and network delay did not increase significantly. The overhead of the flow control system in terms of switch hardware, network interface hardware, and communication is quite low. For this reason, further study of this system may yield an effective, low cost flow control system.

As the communication requirements of multiprocessors continue to increase, wiring, fault-tolerance, and cost constraints will require that we utilize multistage networks at increasingly greater levels. As we push operating points closer to the maximum sustainable load, the problems associated with traffic imbalances (especially hot spots) become increasingly severe. This situation makes imperative the development of networks resistant to hot spot style degradation. The system presented in this thesis has been shown to be effective in controlling hot spot degradation. It also may prove to be effective in preventing degradation by a more general class of traffic imbalances. Such a system has the potential to make multistage routing networks resistant to traffic imbalance induced performance degradation.

6.2 Applications of Results

The results of this thesis are strongly dependent on several assumptions about the network traffic load. These assumptions have not been verified, and in fact are very difficult to verify. The difficulty is that the run time behavior of programs and multiprocessor systems is sufficiently complex that we cannot precisely characterize the traffic unless we actually build the system. Further, these traffic characteristics are likely to be strongly related to the architecture of these machines.[1] We discuss our assumptions about the network traffic

[1] Parallel processor architecture is currently a very active area of research.

briefly in this section.

We have conjectured that hot spots will occur in multiprocessors. In fact, we have explained a number of plausible mechanisms by which they may arise. However, we do not know how often hot spots will arise, and how severe they will be. This means that we cannot ascertain the importance of controlling hot spot degradation in achieving high performance in multiprocessors.

We have focused our study on one type of traffic imbalance – hot spots. This focus was necessary for several reasons: First, hot spots seem to be a likely type of traffic imbalance. Second, hot spots result in severe performance degradation. Finally and more pragmatically, the behavior of the network in the presence of hot spots is simple enough that we hoped to be able to understand it analytically. We also assumed that all network inputs contribute equally to each hot spot imbalance. If this is not the case, those inputs participating in the hot spot are likely to experience even greater performance degradation than we described. Those not participating in the hot spot are likely to experience less degradation.

Finally, we have assumed that the network traffic load is wholly inelastic. This traffic characteristic is affected greatly by the processor architecture. If Von Neumann style processing elements are used, then the traffic may not be as inelastic as we assumed. But in fact, dataflow architectures do provide the promise in this direction. They may in fact be able to produce a traffic load that is essentially inelastic for all interesting programs. The work presented here can be seen as motivation for machines that will generate an inelastic traffic load. On the basis of our results, we know that with such machines we will be able to build a network which will tolerate greater imbalances in traffic load.

One of the really difficult problems in multiprocessing today is how to map a problem to a machine. When the network traffic is not spread evenly over a multiprocessor, one consequence of a poor mapping, system performance may suffer. Flow controlled networks offer tolerance of significant traffic imbalances. This tolerance will effectively loosen the constraints of efficiently mapping work on to the machine, perhaps rendering the mapping

problem practically solvable.

6.3 Future Research

There are several significant remaining topics for further research in this area. First, we have shown that given the traffic is inelastic, the proposed flow control system is quite effective. One might wonder, how true is the inelastic assumption is for current and proposed multiprocessors.

Second, our dissipation bounds were not as tight as the congestion bounds. This fact, in conjunction with the overshoot phenomenon reported in Chapter 5 leads us to believe that some more complex behavior is occurring in the dissipation of congestion. Investigation of this behavior would allow us to better understand the decongestion process.

Third, for complexity reasons, we only considered networks with static routing. However, dynamic routing has been proposed for many networks. The implications of such a choice on hot spot behavior would be an interesting topic to study.

Fourth, we only considered multi-stage networks constructed from 2-by-2 routers. This is a small subset of the possible routing networks. Investigation of networks built from larger switches, networks with replicated links, or duplicated networks would also be interesting.

Fifth, we discussed only one type of traffic imbalance – hot spots. However, we have conjectured that this system may allow multistage routing networks to higher utilizations, by providing greater resistance to degradation from various types of traffic imbalances. Further inquiry along these lines would be very interesting as well as very important to large scale multiprocessor design.

Finally, we still do not know how severe a problem hot spot degradation will be. Although we have studied the performance degradation due to hot spots and a means of mitigating that degradation, we still do not know how often hot spots will occur, how

intense they will, be and how long they will last. To answer these questions, we must characterize the distribution of memory references. Simulation studies [6,8] have shown that the spatial distribution of memory references (in the address space) in parallel machines to be very uneven. This skew is due to the algorithms and hence, is not likely to change significantly. The fact that the spatial distribution is very skewed and leads us to believe that hot spots will occur.

The temporal distribution of memory references will tell us how long and how intense hot spots will be. However, this distribution is much more difficult to study. These statistics depend strongly on topics of current research – how to allocate structures and how to allocate work. As these issues are resolved, the changing policies will affect the temporal distribution of memory references significantly. Consequently, characterizing the temporal distribution of memory references in parallel processors is still an open research topic. When we can accurately assess run time traffic characteristics we will be able to determine how essential controlling hot spot congestion will be to overall performance.

Appendix A

Simulation Data

This chapter contains more comprehensive data from the simulation experiments conducted. This information is presented here as a reference for those interested in the range of network scenarios that were actually simulated. For more complete simulation data or for clarification, the interested reader is referred to the relevant research memos listed in the bibliography.

# of Nodes	Throughput w/o System	Delay w/o System	Throughput w/ System	Delay w/ System	Delay w/ Bal. Traffic
16	0.293	30.4	0.39	6.53	6.1
32	0.168	63.5	0.37	7.89	7.4
64	0.092	119.3	0.34	9.55	8.75
128	0.049	232.1	0.33	11.98	10.0
256	0.024	421.5	0.31	15.58	11.4
512	0.012	908.5	0.295	35.80	12.7

Table A.1: Steady State Throughput and Delay in Controlled System – 16% Hot Spot.

# of Nodes	Hot Spot Intensity	Hot Spot Duration	Throughput w/o System	Throughput w/ System
16	0.16	200	0.386	0.396
16	0.16	400	0.363	0.394
16	0.16	600	0.343	0.391
16	0.32	200	0.351	0.390
16	0.32	400	0.308	0.384
16	0.32	600	0.261	0.378
32	0.08	200	0.384	0.397
32	0.08	400	0.362	0.393
32	0.08	600	0.340	0.393
32	0.16	200	0.350	0.391
32	0.16	400	0.301	0.385
32	0.16	600	0.255	0.377
32	0.32	200	0.311	0.378
32	0.32	400	0.249	0.362
32	0.32	600	0.186	0.344
64	0.04	200	0.384	0.396
64	0.04	400	0.360	0.395
64	0.04	600	0.339	0.394
64	0.08	200	0.346	0.392
64	0.08	400	0.298	0.388
64	0.08	600	0.251	0.384
64	0.16	200	0.298	0.384
64	0.16	400	0.236	0.372
64	0.16	600	0.175	0.361

Table A.2: Normalized System Throughput with Short Duration Hot Spots

# of Nodes	Hot Spot Intensity	Hot Spot Duration	Delay w/o System	Delay w/ System
16	0.16	200	9.77	6.20
16	0.16	400	14.59	6.28
16	0.16	600	19.45	6.36
16	0.32	200	16.78	6.39
16	0.32	400	27.40	6.57
16	0.32	600	38.40	6.58
32	0.08	200	10.73	7.50
32	0.08	400	15.76	7.71
32	0.08	600	20.53	7.65
32	0.16	200	17.97	7.78
32	0.16	400	30.10	7.89
32	0.16	600	40.93	7.98
32	0.32	200	31.96	8.29
32	0.32	400	54.13	8.49
32	0.32	600	77.45	8.46
64	0.04	200	12.00	8.80
64	0.04	400	16.96	8.91
64	0.04	600	21.51	8.94
64	0.08	200	20.13	9.19
64	0.08	400	31.53	9.33
64	0.08	600	42.99	9.48
64	0.16	200	35.59	9.80
64	0.16	400	59.37	10.04
64	0.16	600	80.98	10.49

Table A.3: System Average Delay with Short Duration Hot Spots

Bibliography

[1] Arvind, et. al. The tagged token dataflow architecture. to be published as an MIT Technical Report, August 1983.

[2] Arvind, Gostelow and Plouffe. *An Asynchronous Programming Language and Computing Machine.* Technical Report 114a, University of California, Irvine, Department of Computer Science, December 1978.

[3] G. A. Boughton. *Routing Networks for Packet Communication Systems.* PhD thesis, Massachusetts Institute Technology, August 1984.

[4] A. A. Chien. A comparison of hot spot detection mechanisms. Massachusetts Institute Technology CSG Memo 267, Chapter 2, 1986.

[5] A. A. Chien. The impact of buffer policies on network behavior in the presence of hot spots. Massachusetts Institute Technology CSG Memo 267, Chapter 3, 1986.

[6] A. A. Chien. Structure referencing in the tagged token dataflow architecture – an initial investigation. Massachusetts Institute Technology CSG Memo 268, 1986. A GITA Study.

[7] D. M. Dias and J. R. Jump. Analysis and simulation of buffered delta networks. *IEEE Transactions on Computers*, C-30(4):273–82, April 1981.

[8] G. F. Pfister F. Darema-Rogers and K. So. Memory access patterns of parallel scientific programs. 1986. IBM T. J. Watson Research Center.

[9] Gottlieb, et. al. The nyu ultracomputer – designing an mimd shared memory parallel computer. *IEEE Transactions on Computers*, C-32(2):175–189, February 1983.

[10] C. P. Kruskal and Marc Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, C-32(12):1091–98, December 1983.

[11] M. Kumar and J. R. Jump. Performance enhancement in buffered delta networks using crossbar switches and multiple links. *Journal of Parallel and Distributed Computing*, 1(1):81–103, 1984.

[12] M. Kumar and G. F. Pfister. The onset of hot spot contention. In *Proceedings of ICPP*, IEEE, August 1986. To Appear.

[13] Rosanna Lee. On hot spot contention. *Computer Architecture News*, 13(5):15–20, December 1985.

[14] Majithia, Irland, et. al. *Flow Control In Computer Networks*, pages 211–34. North Holland, 1979. Experiments in Congestion Control Techniques.

[15] J. A. Patel. Procesor-memory interconnections for multiprocessors. In *Proceedings of the 6th Annual Symposium on Computer Architecture*, pages 168–177, IEEE Computer Society, IEEE Press, April 1979.

[16] G. F. Pfister. The ibm research parallel processor prototype (rp3): introduction and architecture. In *Proceedings of the ICPP*, pages 764–771, IEEE Computer Society, IEEE Computer Society Press, August 1985.

[17] G. F. Pfister and V. A. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, C-34(10):943–948, October 1985.

[18] L. T. Wu. Mixing traffic in a buffered banyan network. In *Proceedings of the Ninth Conference on Data Communications*, pages 134–9, SIGCOMM, ACM, British Columbia, Canada, September 1985.

CS-TR Scanning Project
Document Control Form

Date : 4/11/95

Report # LCS-TR-382

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 102(111-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP (1-4) UNNUMBERED TITLE, ABSTRACT, ACKNOWLEDGEMENT, & DEDICAT. PAGES	
(5-8) PAGES #'ED 1-4 (TABLES OF CONTENTS	
(9-102) PAGES #'ED 1-94	
(103-108) SCANCONTROL, COVER, SPINE, PRINTERS NOTES, DOD(2)	
(109-111) TRGTS(0)	

Scanning Agent Signoff:

Date Received: 4/11/95 Date Scanned: 4/12/95 Date Returned: 4/20/95

Scanning Agent Signature: Michael W. Cook

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MIT/LCS/382	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Congestion Control in Routing Networks		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Andrew Andai Chien		8. CONTRACT OR GRANT NUMBER*(s)* Office of Naval Research Contract N00014-84-K-0099
9. PERFORMING ORGANIZATION NAME AND ADDRESS MIT Laboratory for Computer Science 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/DOD 1400 Wilson Blvd. Arlington, VA 22217		12. REPORT DATE November, 1986 13. NUMBER OF PAGES 94
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Multistage Routing Networks, Hot Spots, Crossbar, Dataflow, Multiprocessors, Tagged Dataflow Architecture, Von Neumann Architecture, Throughput, Delay

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Multistage routing networks present an attractive cost-effective method of interconnection for medium to large scale multiprocessors. Recent results concerning performance degradation in the presence of "hot spots" have raised serious questions about the robustness of previous performance estimates for these routing networks. Research to date has focused on a limited class of hot spots - those in which all the hot spot traffic is destined for the same memory address.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

We consider a more general kind of traffic imbalance - the hot spot traffic may be of arbitrary composition. By taking this more general view, we hope to understand a wider class of traffic imbalances. In this thesis, we define an analytic framework in which to study the problem of performance degradation due to "hot spots." We characterize the performance degradation due to those hot sports. This degradation is very severe. We then employ approximate methods to estimate the time to congest the network, and the time to dissipate that congestion. These approximations are validated by extensive simulation of the model.

We subsequently propose a solution to prevent performance degradation due to "hot spot" traffic imbalances. The effectiveness of this solution depends crucially on the traffic model one considers. In our studies, we assume that the traffic load is completely inelastic. This assumption is not verified. The proposed solution involves two primary mechanisms - misrouting and throttling the congesting traffic. Simulations results show that the proposed scheme is very effective in maintaining good network performance in the presence of "hot spots." Network throughput and delay are maintained at near balanced load levels. The simplicity of the shceme and a few key simulation statistics indicate that implementation of the proposed scheme should require only minimal hardward and limited run time communication.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 4/12/1995

M.I.T. Libraries
Document Services