Performance Evaluation of Multicomputer Networks for Real-Time Computing

by

John T. McHenry

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED

[illegible]

Dr. S. F. Midkiff, Chairman

[illegible]

Dr. N. J. Davis IV

[illegible]

Dr. I. Jacobs

April 1990

Blacksburg, Virginia

LD
5655
V855
1990
M433
c. 2

Performance Evaluation of Multicomputer

Networks for Real-Time Computing

by

John T. McHenry

Dr. S. F. Midkiff, Chairman

Electrical Engineering

(ABSTRACT)

Real-time constraints place additional limitations on distributed memory computing systems. Message passing delay variance and maximum message delay are important aspects of such systems that are often neglected by performance studies. This thesis examines the performance of the spanning bus hypercube, dual bus hypercube, and torus topologies to understand their desirable characteristics for real-time systems. FIFO, TDM, and token passing link access protocols and several queueing priorities are studied to measure their effect on the system's performance. Finally, the contribution of the message parameters to the overall system delay is discussed. Existing analytic models are extended to study delay variance and maximum delay in addition to mean delay. These models separate the effects of node and link congestion, and thus provide a more accurate method for studying multicomputer networks. The SLAM simulation language substantiates results obtained analytically for the mean and variance of message delay for the FIFO link access protocol, as well as providing a method for measuring the message delay for the other link access protocols and queueing priorities. Both analytic and simulation results for the various topologies, protocols, priorities, and message parameters are presented.

Acknowledgements

With the completion of this effort, I find that it is time to remember the many people who have helped me toward this goal. First, I would like to thank Dr. Scott Midkiff for his guidance and direction in the course of this research and for the many hours he spent editing preliminary versions of this paper. I would like to acknowledge Dr. Nathaniel Davis for his helpful comments and contributions in refining the scope and information content of this research. Also, I would like to thank Dr. Ira Jacobs for serving on my graduate committee.

This effort was made possible due to the generosity of Mrs. Marion Bradley Via's provision of an endowment to the Bradley Department of Electrical Engineering at Virginia Tech. To her, I would like to express my sincere gratitude.

Above all, I would like to thank my parents for the support and encouragement they have provided throughout my many years as a student.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Current clock speeds push the limits of silicon technology, and the constant desire for faster processing is becoming more difficult to fulfill. Distributed memory processors, or multicomputers, are currently being studied as a way of lessening the dependence of high-speed computing on innovations in silicon technology [1]. Real-time processing is one area that will benefit from advances in distributed memory computing [2].

Advances in many areas of computing are helping to enhance the efficiency and performance of real-time processing. The backbone of distributed memory parallel processing is the network through which processors share information. Innovations such as the Fiber Distributed Data Interface (FDDI) are increasing the speeds at which data can be transferred between stations in local area networks [3]. On a scale of smaller distances, innovations in optical communication technology can greatly increase the bandwidth of internodal links, thus increasing the speed at which data can be passed between processors in a distributed memory system. New technology is of the most benefit when it is efficiently and effectively implemented. For this reason, it is necessary to study how advances in electro-optical communication and processor design can be used most effectively in a distributed memory computing system.

It is the purpose of this research to study several hardware organizations and network contention protocols to find effective methods for interprocessor communication in real-time processing systems. The factors examined include the effect of network topologies, link protocols, queueing priorities, and message characteristics on the overall system performance. Real-time processing constraints such as high data throughput, low information delay, and low variance of the delay time, including limits on maximum delay, are examined to see how they are affected by changes in the above mentioned network specifications.

This work extends beyond research done previously in this area in two ways. First, it examines the variance of message passing delay in addition to the mean. Secondly, the effects of contention protocols are explicitly modeled and the effects of node and link congestion are considered separately. It also identifies the reasons why significant message variance occurs and studies how different contention protocols affect the mean and variance of message delay. This research weighs these various effects and recommends methods for limiting the mean and variance of message delay in future systems.

The three topologies considered in this research are the spanning bus hypercube (SBH), the dual bus hypercube (DBH), and the torus. Initially, a first-come-first-serve queueing model is used for networks of size W^D for all three topologies. Results for this model are obtained both analytically and through simulation. Simulation is then used to determine system performance for both a token passing and a TDM contention protocol for link access. The performance effects of additional message parameters of the model such as the message lengths and the number of hops a message will typically take are examined in detail for the SBH topology. In addition to these network performance issues, the use of the SLAM simulation language and its ability to model complicated networks are described.

Chapter 2 gives background information needed to understand the problem under consideration. The restrictions introduced by real-time constraints are outlined. The three topologies used in this study are explained in detail, and basic information about why each was chosen for this study is given. The contention protocols used for link access are also described, and finally, several commercially available distributed memory parallel processing machines are briefly discussed. Chapter 3 presents previous research in this area. It outlines areas where little work has been performed and indicates how the research presented herein will provide needed insight. Also discussed is the origin of the network model being used in this study. Chapter 4 describes the methodology used to analyze the problem under consideration. Complete descriptions of the node and link models used in this study are given, along with the assumptions used by the network model. Analytic calculations are then formulated that predict message delay for a FIFO protocol for all three topologies. A discussion of the network model created using the SLAM simulation language is included that describes how simulations of the network for both the FIFO contention protocol and several more complicated access schemes are performed. The results obtained from the analytic equations and SLAM simulations are detailed in Chapter 5. Simulation results obtained using SLAM are compared with results obtained analytically to verify the validity of the simulations. The data obtained through these methods are analyzed to see how real-time processing constraints are affected by the various parameters and to suggest system characteristics that are best suited to real-time processing applications. The methods used to number nodes and links in the network and the numerical results obtained from simulations are presented in the appendices.

Chapter 2. Background

Before discussing the methods used to examine hypercube multicomputer systems, it is necessary to become familiar with the terminology and other important aspects of the subject under consideration. This chapter begins by discussing the restrictions that real-time operation places on a network. Descriptions of three hypercube networks are given with an emphasis on the features of these networks that are important in this research. Next, basic information on the protocols used in this research is given, along with a discussion of how each of the protocols may affect the delay of messages in the system. Finally, the NCUBE and Intel iPSC systems are discussed as examples of commercial distributed memory machines.

2.1. Real-Time System Constraints

Real-time systems are different than non real-time systems because proper operation of a real-time system is determined not only by the actions of the system, but also by the time at which these actions occur [2]. This property causes real-time systems to be constrained by time related factors in addition to aspects that affect all computer systems. Timing relations in real-time systems become more important than in typical computer systems. Although many timing aspects affect the performance of a distributed memory system, one major component of system performance is the network through which processors communicate [4]. Thus, knowledge of the timing constraints is a major factor when attempting to choose a communication structure for a distributed memory system to be used in a real-time application [5]. The important characteristics needed for a real-time system are discussed in the following paragraphs.

2.1.1. Average Delay Time

Joseph and Goswami [6] describe a real-time system as being made up of an environment and a programmed system that measures and attempts to control the state of this environment. Viewed in this way, the importance of speed in real-time systems is apparent. The speed at which the state of the environment changes determines what the speed of the real-time system must be. As applications of real-time systems are developed for rapidly changing environments, it is necessary for these systems to be able to keep up with these changes. Thus, increasing the speed of a real-time system will increase the number and types of environments in which the system can be implemented.

Coupled with the need for increased reaction time desired in real-time systems is the ability of the system to handle larger environments. As the size of the environment increases, it is necessary to increase the number of sensors and other devices measuring the current state of the environment. For the processing elements to gain a complete picture of the environment's state, it is necessary to pass these measurements to many processors in the system.

From the above paragraphs, it can be seen that it is important for a real-time system to be able to react to a large number of rapidly changing events in the environment. For a real-time multicomputer system, the communication network that sends messages to and from nodes in the network is one of many factors that limits the rate at which the system can react to outside events. For this reason, it is important for such systems to have low delay for internode communication. The required network speed is determined by the nature of the environment being considered, but increasing the rate at which separate processors can share information helps to increase the number of applications for which the real-time system is suited.

2.1.2. Variance of Delay Time

Although reducing the average communication delay time can help to increase the system performance, it is not the only characteristic of the communication delay in real-time systems that must be considered. Instead, the success of a real-time systems is more dependent upon predictability than on speed [2]. Since communication delay can greatly affect the predictability of a real-time system, it is important that the variance of communication delay in the system be small. This fact can be seen from the large amount of research done to strictly confine the timing characteristics of the communication structure of real-time systems [4,5,7,8,9]. The variance of the message delay is used in this research to determine the predictability of the communication network's timing.

2.1.3. Maximum Delay

There are many ways of scheduling and prioritizing tasks in a real-time processing system. For some applications it may be sufficient to know only the average and variance of message delay to assure proper operation, but there is also a major emphasis in research concerning real-time computing to develop methods for assuring that tasks are completed within some maximum time [9,10]. A variety of techniques for scheduling events in real-time systems have been studied to find optimal ways of assuring hard limits on delay [8]. For the research presented in this paper, the way that different topologies affect the maximum message passing delay is considered as one criterion for determining the relative worth of a network topology.

2.2. Topology Descriptions and Routing Methods

The topology of the communication network through which processors communicate has a large effect on the communication performance of the network. This section describes the three network topologies examined in this research. In this discussion, both the physical layout of each topology and their expected contribution to the research are presented. Also, the methods used to route messages to their destination are explained.

In this research, the performance of different topologies is compared only for systems with an equal number of nodes. The 4^3 arrangement of nodes used for simulations of each topology in this research is shown in Figure 2.1. A typical W^D configuration consists of a D dimensional lattice that is W nodes wide in each dimension. The different hypercube topologies do not affect

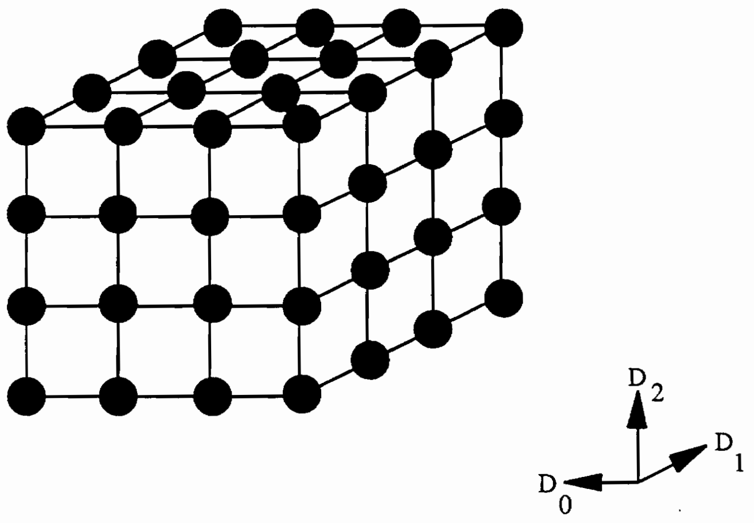


Figure 2.1. Arrangement of Nodes in a 4^3 Lattice

the placement of the nodes in the lattice, but instead affect the placement of communication links between the nodes.

Nodes in a hypercube can be specified by referring to where they are located in this lattice. A node in the corner of the lattice is chosen as the origin. Next, the orthogonal dimensions of the network are defined starting at d_0 and ending with d_{D-1}. Coordinates are then assigned to each node by counting the number of lattice spacings that this given node is away from the origin in the ith dimension, and then assigning this value to the d_i coordinate. This processes is repeated for each of the node's D coordinates. Thus the coordinate of the origin is (0,0,...,0), and the highest numbered coordinate is (D-1,D-1,...,D-1).

2.2.1. Spanning Bus Hypercube

For this topology, each node has one link attached to it in each of the *D* dimensions of the lattice. Each link in the network is a communication bus that connects *W* different nodes. All of the links in the network lie along each of the dimensional lines in the lattice. An example of a 4^3 SBH is shown in Figure 2.2. Additional information on the structure of the spanning bus hypercube is contained in [11].

Each link in the SBH is shared by the *W* different nodes connected to it. At any given time, only one of the *W* nodes can transmit on the link and this transmission must be to one of the *W-1* other nodes connected to the link. Thus, some access scheme is needed to determine which node has permission to transmit on the link. Because each node in the network is not connected to every other node in the network, transmission between pairs of nodes often requires the use of several link-node hops.

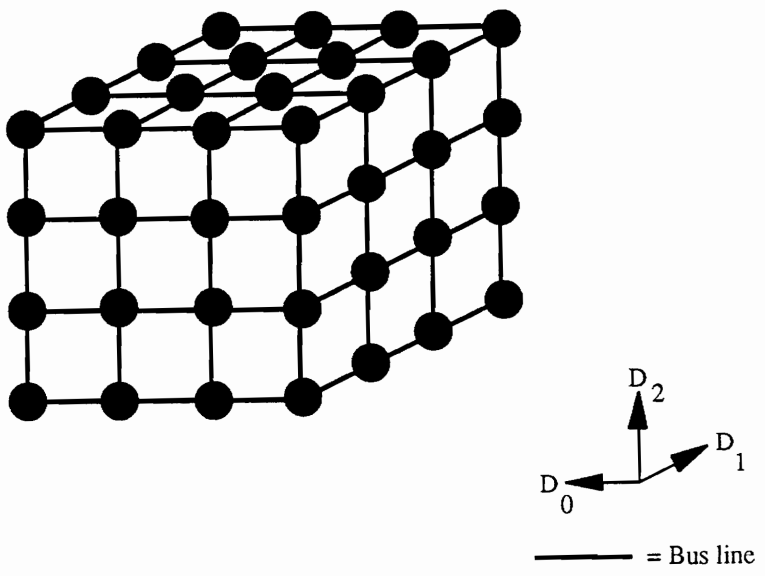


Figure 2.2. 4^3 Spanning Bus Hypercube

A major benefit of the SBH topology is that a message is required to take a maximum of *D* hops between any two nodes in the network. For different topologies with a W^D configuration, this maximum distance, known as the network diameter, is at least as small for the SBH as for the other two topologies considered in this research.

The SBH modeled in this research uses the simplest routing scheme of the three topologies considered. After examining the structure of the network, it can be seen that transmitting the messages on a link in the i^{th} dimension will change the value of the message's current d_i coordinate to the value of the d_i coordinate of its eventual destination. Also, since links exist in all dimensions it is possible to change any of the message's D coordinates from any node in the network. Thus, message routing consists of examining each coordinate of the current node the message is at and each coordinate of the destination node of the message. This process is done consecutively for d_0 through d_{D-1}. As soon as a coordinate is detected that is not the same for the current node as for the destination of the message, the message is sent to be transmitted on a link that traverses the dimension of the differing coordinate. The message is transmitted to the node on the link whose coordinate in the dimension of the link is equal to this coordinate at the message's final destination. In this way, it is easy to see that it takes a maximum of D different transmissions to get a message from its source to destination. Also, if the coordinates of the starting node and destination node are uniformly distributed, the probability distribution for traffic on a link is identical for all links in the network.

A special case of the SBH is the binary hypercube. This topology is a SBH with a width, *W*, of two. The binary hypercube is the topology chosen as the basis for most of the distributed memory machines currently on the market [12,13,14].

2.2.2. Dual Bus Hypercube

Although the SBH offers the advantage of a large number of multiple paths between nodes, it has a drawback in that it requires a large number of links and connections. In an effort to limit the cost incurred by this high level of connectivity, and to create a more easily expandable network topology, the DBH was proposed [11].

An example of a DBH is shown in Figure 2.3. A thorough description of this topology is contained in [12]. The DBH is constructed by starting with a SBH and then removing selected links. One dimension is designated as the 0-dimension, and all links in this dimension remain intact. All other dimensions are called secondary dimensions. Nodes in the network with d_0 equal to zero are considered. Links connected to these nodes in all directions except for dimension 0 and 1 are removed. Then, nodes with d_0 equal to one are considered. Links connected to these nodes except for dimension 0 and dimension 2 are removed. This process continues where links connected in all but the 0 and the X dimensions are removed for each node in the network, where

$$X = [(d_0\text{-}1) \text{ MOD } (D\text{-}1)] + 1 .$$

The modulo (MOD) function accounts for cases where W is larger than D-1. In such cases, links in a given secondary dimension are used for more than one value of d_0. From this equation, it can be seen that only in cases where *W* is a multiple of *D-1* will the number of links in a given secondary dimension be equal for all of the secondary dimensions in the network.

Completion of this process leaves two links attached to each node in the network. Limiting the number of links in the network lowers the cost of the network, but it may also hinder the network's performance. This research will help to expose how much performance degradation occurs when the topology is changed from a SBH to a DBH while measuring performance as it

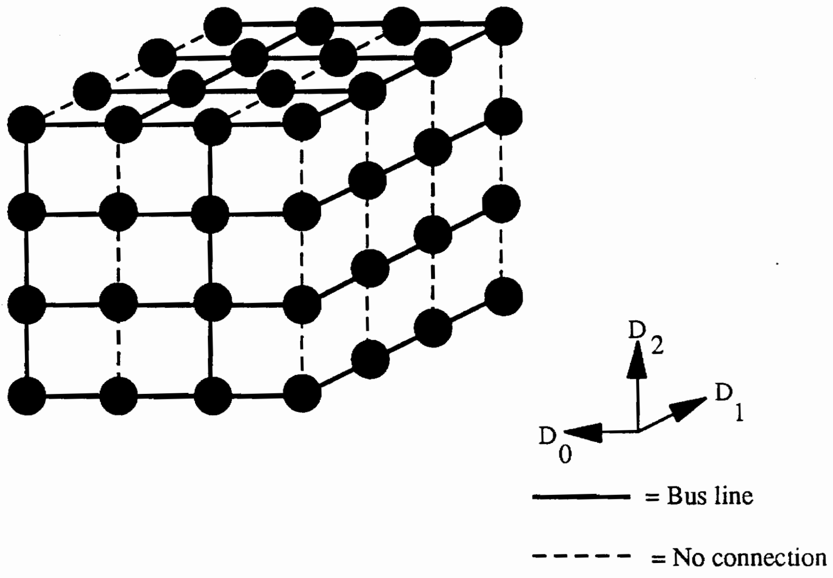


Figure 2.3. 4^3 Dual Bus Hypercube

pertains to real-time processing. Thus, it will give an indication as to the value of the additional complexity of the SBH and serve as a guide when choosing between these two topologies.

Routing for the DBH is similar to SBH routing, but it has additional considerations due to the links that were removed for the network. The basic process of trying to correct one coordinate at a time still applies. In this topology, however, links in only the primary and one secondary dimension are accessible from any given node.

The dimension of the secondary link attached to the current node is examined first to see if it differs from the corresponding destination node coordinate. If so, the message is transmitted on this secondary link to correct this coordinate. If this coordinate is already equal to the destination's coordinate in this dimension, the coordinates of the other secondary directions are checked one at a time. If one of these dimensions is found to differ, the message is transmitted first along the primarily link to a node that is connected to the link in the secondary direction that is to be changed. To evenly distribute traffic on the secondary links, the message is transmitted to the node which has the proper link connections as described above and has the closest d_0 coordinate value that is larger than the d_0 value of the current node. If there is no proper node containing a higher d_0 coordinate value, then the proper node with a d_0 value between 0 and D-2 is selected. Because of the construction of the network, only one such node will exist. After this first transmission is made, the message is then transmitted on the secondary link and its coordinate in this secondary dimension is corrected.
This process is continued until all of the secondary dimensions have been corrected. At this point, if the value of the message's primary dimension coordinate does not need adjustment, then the message is at its final destination. If the value of the message's primary dimension coordinate needs adjustment, then the message is transmitted one more time on a primary link

to its final destination. One final note is that this final adjustment of the primary dimension coordinate can be avoided in cases where the coordinate of the secondary dimension that has a link connected to it at the destination node needs to be adjusted as the message is transported from its original node to its destination node. If this secondary dimension needs adjustment, then this coordinate is the last coordinate corrected. This causes the message to have the proper primary dimension coordinate after this final secondary link is corrected.

Using this algorithm, messages are routed from source to destination in a uniform manner. Thus, the traffic distribution on primary links will be identical for each primary link in the network, and the traffic distribution for secondary links will be identical for each secondary link in the network.

2.2.3. Torus

The torus topology offers the same connectivity as the SBH, but provides the advantage of using only point-to-point links. This characteristic makes it easier to use fiber optic links with the torus topology. This topology is similar to the SBH, except that each link that connects a string of nodes in the SBH is replaced by a ring of links which connects these nodes one pair at a time. For example, links in the i^{th} direction will connect the node with $d_i = 0$ to the node that has the same coordinates in every dimension except that it has $d_i = 1$. In a similar fashion, the node with $d_i = 1$ will be connected to the node with $d_i = 2$ and so on until the ring is completed with the node with $d_i = (D-1)$ connected to the node with $d_i = 0$. A further discussion of the torus topology can be found in [12].

One effect of using only point-to-point links in the network is that a large number of links and connections are needed to build the network. The average distance between two nodes in the network also increases in the torus topology as compared to the SBH. This large number of links, though, will also prove to be advantageous in that the number of messages the network can carry until the links become saturated will increase. Also, the point-to-point links in the torus can be more easily implemented with fiber optic technology than the bus links of the SBH and DBH. The simulations performed in this research demonstrate how these various factors play against each other in determining the network's performance.

Routing in the torus topology is similar to routing in the SBH. Each coordinate of the message's current node and destination node are examined and each of the coordinates are corrected one at a time. The difference between the torus and the SBH is that messages in the torus will sometimes require more than one hop to correct a single coordinate. For instance, suppose the current node of a message has $d_i = 1$ and the destination of this message has $d_i = 4$ in a torus with $W = 5$. The routing scheme must determine which direction to transmit the message around the ring. It could transmit it along path P_1 or path P_2 as defined below.

$P_1 : d_i = 1$ ---> $d_i = 2$ ---> $d_i = 3$ ---> $d_i = 4$

$P_2 : d_i = 1$ ---> $d_i = 0$ ---> $d_i = 4$

All other coordinates are assumed to be constant for these transmissions. In this case, it can be seen that transmitting in the negative direction along the link provides a shorter path than transmitting in the positive link direction. The positive direction is defined as the direction of $d_i = 0$ to $d_i = 1$ transmissions, and the negative direction is defined as the direction of $d_i = 1$ to

$d_i = 0$ transmissions. For a general case, the following algorithm determines which transmission direction should be used. Let X be defined as

$$X = [\ (d_i \text{ of current node}) - (d_i \text{ of destination node})\ ] \text{ MOD } W.$$

Then,

$$\text{If } X \geq \frac{W+1}{2} \text{ then transmit in the positive direction },$$
$$\text{and} \quad \text{if } X < \frac{W+1}{2} \text{ then transmit in the negative direction.}$$

This process of correcting the message's current coordinates is repeated until each of the message's coordinates are equal to the destination's coordinates, and at this time the message will be at its destination. With this algorithm, messages are transmitted from source to destination in a way that causes the distribution of link traffic to be identical for a links in the network.

2.3. Link Access Protocols

All of the topologies examined in this research use multiple links that have more than one node attached to them. For the SBH and DBH, there are *W* nodes attached to each link, and in the torus there are two nodes per link. Since the link is the only mechanism through which different nodes communicate, there is no way for different nodes to independently decide who has the right to transmit on the link in cases where more than one node has information to transmit. For this reason, protocols are developed that allow nodes to determine if they have access to the link. Three different link access protocols are studied in this research. These protocols are adapted from protocols that are used in real-world systems, and they have been chosen to study how they will affect the network's performance with respect to real-time processing constraints.

A thorough discussion of how each of these protocols is modeled for simulation is described in Section 4.3.2.2.

2.3.1. FIFO Protocols

The First-In-First-Out (FIFO) protocol is the easiest protocol to understand, but is often the most difficult protocol to implement. For this protocol, messages are transmitted on the link in the order in which they are ready for transmission. Thus, the first message to ask permission to use the link will be the first message transmitted. The problem with implementing this protocol concerns the fact that it assumes that each node on a link knows the status of messages requesting transmission from all of the nodes connected to the link. Since nodes only communicate through the link, there is no easy way for this information to be shared on a single link system. Also, for the case where two nodes get ready to transmit a message on an empty link at the same time, each of the nodes will not be aware that the other node is also beginning to transmit. This would cause both transmissions to be destroyed as they collide on the link.

For the FIFO models used in this research, it is assumed that there is some central arbiter that keeps track of the time at which each message first requests permission to use the link. This arbiter then grants permission for message transmission in the exact order of these requests. The arbitration process is assumed to be 100 percent efficient, and thus no link transmission time is wasted as the arbiter is deciding which message to give permission to next. Implementation of the model thus requires only a single queue that stores all messages in chronological order and releases them to a server in a FIFO manner.

Although this model is more simplified than any real world examples, it does provide a simplified version of arbitration protocols that do exist. The unrealistic efficiency of this logic will cause the results of this model to serve as an upper bound for the performance that could be achieved with a FIFO protocol.

2.3.2. TDM Protocols

Time Division Multiplexing protocols break down the link time into separate time slots. Only one node on the link is given permission to transmit during a given time slot. In the simplest versions of TDM, each node has access to the timing of some global clock or each node generates its own clock and synchronizes it with the clocks of all other nodes attempting to transmit on a link by examining the link transmission of these other nodes. The clock period is broken down into separate clock sub-periods. The clock time for which a node has permission to transmit is predetermined, and each node transmits only at these predetermined times.

Although this protocol does have some problems, TDM protocols impart some desirable characteristics for real-time processing that have caused it to be used in many current customized real-time systems [5]. Regularly scheduled transmission opportunities for messages help to schedule the transmission of all messages in the network. Thus, predictable transmission times may help to add to the overall predictability of communication in the network. This would be advantageous in a real-time system.

This protocol eliminates the requirements concerning global queue state information seen in the FIFO protocol, but it also has some drawbacks. Some time is wasted in this protocol, because other nodes are not allowed access to the link even if the node controlling the link has no information to transmit. Thus, the efficiency of the link is reduced. The method used to implement TDM for the models used in this research contains some modifications which attempt to enhance link efficiency. Both the implementation and modifications of this protocol are discussed in Chapters 4 and 5.

2.3.3. Token Passing Protocols

The final link access protocol used in this research uses token passing to assign transmission permission to nodes connected to a link. A token is a unique set of bits transmitted on the link. A node that receives a token has permission to transmit on the link. If the node has no messages to transmit, it transmits the token to the next node on the link. If the node has one or more messages to transmit, it will begin transmitting messages until it has no more messages to transmit or until it transmits some predefined maximum number of messages per token capture. Upon completion of transmission, the node transmits the token to the next node on the link. The last node on the link will pass the token to the first node of the link, and thus the token is passed in a circular fashion around the link.

The token passing scheme provides a sequential scheduling of link access, but access times are not as strictly defined as in the TDM protocol. A benefit of the token passing scheme is that the link will never sit idle if any of the nodes connected to the link have a message to transmit. Some link time, however, will be wasted when the token is transmitted. Thus, the token scheme contains some link inefficiency as did TDM. The effect of this wasted time is examined in the simulations of this research by varying the time needed to transmit a token.

Strosnider, et. al. [5] argue that as the complexity of real-time systems increase, the effectiveness of token protocols will cause it to replace currently used TDM methods. An example of a token passing protocol is the IEEE Standard 802.4 [15]. This protocol operates on token passing buses, but token protocols have also been developed for point-to-point links in ring connected networks.

2.4. Currently Available Distributed Memory Machines

To understand the applications of distributed memory computers for future real-time systems, it is important to recognize the beneficial characteristics of current systems. Both the NCUBE and Intel iPSC machines are representative of the types of computers being studied in this research. Information available on these systems covers a wide range of issues from internode communication to in-depth descriptions of each node's internal architecture. This section briefly discusses the design and operation of each machine and presents some information on their typical internode communication times. This information enables the analytic models and simulation runs of this research to use representative numbers for the ratio of node to link processing rates.

2.4.1. NCUBE

The NCUBE line of supercomputers contains machines with between 4 and 8192 processors [16]. Each node in the NCUBE computers is made up of several custom-designed VLSI chips tailored to control all necessary processing and communication functions [17]. NCUBE computers use a binary hypercube topology to transmit between nodes. The number of nodes in a given system is scalable, and additional nodes can be added as need permits up to the maximum of 8192 processors.

As with most communication systems, the efficiency of the NCUBE communication links depends on the length of the message being transmitted. Gustafson, et. al. [18] measured the communication latency and the actual transmit time for messages of varying size in a 1024 node NCUBE/ten computer. One case in this study shows that for a 768 byte message being passed to an adjacent node, it takes 0.9 ms for the DMA transfer of the message between the nodes, and it takes 0.4 ms of node processing at each end of the transmission. For another case in this study, communication between two adjacent nodes was seen to take an initial delay of 880 μs plus an additional 1.53 μs per byte contained in the message. From these two examples, it is seen that the delay time incurred by messages with lengths on the order of one kilobyte is approximately equally contributed by node and link delays. Using the initial case of a 768 byte message, it was decided that this research would assume that the rate at which nodes process messages will be approximately twice as fast as links can transmit messages.

2.4.2. The Intel iPSC

The iPSC line of supercomputers contains both the iPSC and iPSC/2 family of distributed memory computers. A binary hypercube is used to connect nodes in both of the above mentioned machines. The model described in this research is a closer approximation of the iPSC architecture than the iPSC/2, but the iPSC/2 is included as an example of the evolution of distributed memory machines.

2.4.2.1 iPSC

Communication in the iPSC uses a packet-swithed, store and forward message technique. Thus, all processors in the communication path between two nodes are slowed down during transmission because of the time they spend handling the message [19]. Typical message transfer rates can be calculated from information provided in [20]. Results for communication delays are obtained by recording the amount of time needed to transmit various length messages between two adjacent nodes in the hypercube. These messages were transmitted in links in an idle system, and thus the link contention effects are not included in these results. From the plots produced from these tests, the time needed transmit a message is seen to be approximately 1.7 ms plus 2 μs for each byte contained in the message [20]. For message lengths of one kilobyte, this data is in accordance with our assumption that the node service rate is approximately twice the link service rate for a message. It must be noted that for this architecture, delay times do not increase linearly as the number of hops between the source and destination nodes is increased, and thus the time it takes a message to travel two hops is not exactly twice as large as the time it takes a message to travel one hop.

2.4.2.2. iPSC/2

The iPSC/2 has significantly better performance characteristics than its predecessor [20]. To speed message transmission between distant nodes for configurations that employ a large number of processors, the iPSC/2 contains a circuit-swithed, Direct Connect Module that assigns communication paths between any two nodes in the network as they are needed [19]. Although some time is needed to create this communication channel, once the channel is created it can transmit bytes to any node in the network without hindering the operation of any nodes through which the data path was constructed. Additional advancements in the iPSC/2 include the use a faster processor and the addition of more memory at each node. This system shows how complex routing algorithms can be implemented within hypercube topologies, but most of its interesting effects go beyond the scope of this research.

Chapter 3. Previous Work

Numerous studies have been performed that relate message delay times in a distributed memory computing system to a wide variety of parameters. These studies often examine the average delay for various system conditions but rarely examine the variance of this delay. This chapter discusses results found in previous research and comments on effects reported in such research that are relevant to the research described in this thesis. In particular, results concerning the model used in this research are discussed, as well as findings that deal with how different network models and protocols affect the overall message delay.

3.1. The Network Model

The network model used in this research is based on the model proposed by Protopapas and Denenberg [21]. This model was created to demonstrate that it is not adequate to model communication in large scale multicomputer systems by considering only the transmission time of messages on links. Instead, it is necessary to include the effects of message processing time at nodes in the overall message delay.

Protopapas and Denenberg [21] contend that not only can the message processing time at nodes be a significant portion of delay, but that it can in fact be the dominant component of the overall message delay for some systems. The significance of the node processing time was found to be a function of the ratio of the node to link processing rates, the lengths of the messages, and the topology of the network being considered. Thus, the basic emphasis of the Protopapas paper is that in current systems the effect of the message processing delay at nodes cannot be ignored. In addition to the results obtained for current processing rates, this model may also be used to analyze future systems that use faster technology that may cause a different balance between node and link processing times. Protopapas and Denenberg consider only the average message delay. Thus, the discussion concerning the variance of the delay contained in this thesis is an extension to Protopapas's research.

3.2. Performance Modeling

Many other network models have been used to demonstrate the effects of various system parameters in a multicomputer network. The articles discussed below contain information that is related to the system parameters being discussed in this research.

Reed and Schwetman discuss the cost-performance bounds of multicomputer networks [22]. Although the research in this thesis does not focus on the cost advantages of one communication network over another, it is concerned with the methods used by Reed to measure the performance of such networks. Performance of the interconnection networks examined by Reed is based on bottleneck analysis. Thus, this measure attempts to examine the portion of the network with the poorest performance, and then bound the overall network performance by this result. The study of variance performed in this thesis research is related to such an analysis.

The effect of a long message delay in any part of the network will be more pronounced in the second moment of the delay time than the first, and thus the effects of a system bottleneck are more apparent in the variance of the message delay than the mean.

Reed and Grunwald [12] continue the discussion of multicomputer interconnection networks. Performance in this paper is measured by such parameters as the bound on the message completion rate obtained through bottleneck analysis, the internode distance, and average message delay. In discussing these aspects of communication networks, several important results and conclusions can be seen that help to justify the research presented in this thesis. In comparing the average message delay of the Intel iPSC and the Ametek S/14, it is noted that the number of bytes transmitted in a message has a large effect on the choice of which system is more efficient. In their research, it was found that the Intel iPSC had a larger delay than the S/14 for low message lengths, but as the message lengths increased the iPSC's average message delay was seen to become smaller than the S/14's delay. It should also be noted that the calculation of the average delay did not include any additional delays that would be caused by buffering in a busy system. What this simplified model does expose is how the efficiency of a communication network is highly dependent upon the typical message lengths used in the system. Another important point made by this research is that the amount of parallelism in a system that will optimize performance is highly dependent upon the communication topology of the system. This justifies the research performed in this thesis that attempts to determine the network conditions that must be considered when choosing a communication topology for a real-time system. One final note is that this research did not attempt to gain information on the delay variance in any of the network topologies it discussed.

Kruskal et. al. [23,24] present in depth calculations for the delay variance in multiprocessor systems. The major focus of these papers is the analysis of queueing buffers in clocked multistage interconnection networks for shared memory computers. The queueing effects of such networks do, however, have some similarities with the queueing models of single stage distributed memory computers. Assumptions used for these networks, such as exponential distributions of arrival and service times, are used as simple approximations in many queueing models. Kruskal also discusses how some of the discrete queueing models used to approximate a clocked network can be modified to represent continuous time models. Thus, the actual analytic equations derived by Kruskal for multistage networks are not useful in this thesis research, but many of the techniques and results obtained are relevant.

First, the work of Kruskal, et. al. demonstrates the ability of queueing models to accurately predict the mean and variance of system delays in parallel processing systems. Also, their work confirms that accurate results can be obtained even after simplifying assumptions are made to limit the complexity of the final analytic equations. Finally, their work explains that the performance of a parallel machine is not based solely on average system delays. Instead, it states that a large variance in system delays can greatly hinder a parallel machine's performance.

Some information concerning multicomputer networks is found in a discussion of multiprocessor networks by Bhuyan, et. al. [25]. Although this discussion concentrates on shared memory multiprocessor machines, it does comment on multicomputer interconnection networks. It states that the cost of fully connected crossbar interconnection networks can cause them to be less effective than topologies with lower connectivity such as shared bus networks. It states that the effectiveness of networks should first be thoroughly tested through simulation and analytic techniques before choosing an interconnection network for a system. The article also states that the large overhead of interprocessor communication in multicomputer networks is a major limiting factor of the machine's performance. Thus, the scope of this thesis research is seen to cover a basic area of concern for multicomputer networks.

Research by Abdalla and Midkiff [26,27] examines the performance of parallel multicomputer networks based upon the average end-to-end delay and the maximum system throughput. This research uses a modified version of the model proposed in [21] to examine interconnection networks. The changes made to the model allow it to be used for packet and virtual cut-through routing. Also, the modified model can be used for both point-to-point and bus communication links. Analytic equations used to calculate the end-to-end delay of various interconnection networks are obtained from the network queueing model, and these equations are used to predict the expected delay in torus and spanning bus hypercube interconnection networks with 1024 and 4096 nodes. The bandwidth for the SBH links is W times the bandwidth of the torus links to account for the fact that each bus link in the SBH is made up of W point-to-point links in the torus. The end-to-end delays obtained in each of the two topological models is plotted versus the traffic load of the network. In the plots for W^D equal to 4^5 and 4^6, the SBH has a shorter message delay than the torus. This result however, is only intended to be a demonstration of how the analytic equations can be used to compare two interconnection networks for a specific set of system parameters and in no way states that the SBH will outperform the torus in all cases. Also, these analytic equations do not consider the variance of the end-to-end delay as do the analytic equations created in this thesis research.

3.3. Protocol Results

One method of controlling the distribution of delay times in a communication network is by changing the protocol used to resolve contention among multiple nodes attempting to use a single communication link at the same time. Many articles have been written concerning the basic characteristics of TDM and token passing networks. This section explains some of these characteristics and research findings as they pertain to the research presented in this thesis.

3.3.1. FIFO Protocols

The FIFO protocol is easily modeled using M/M/1 queues. Implementation of a FIFO protocol in an actual system, however, is much more complicated. A discussion of the structure and performance of a FIFO protocol by Guibaly [28] mentions the necessary elements of a FIFO implementation for multiprocessor systems. Many of these aspects also affect multicomputer systems. An important aspect of a FIFO system is that each node attached to a given link must know information about all other nodes attached to that link. Such needed information includes whether the shared link is currently being used, the total number of messages awaiting transmission on the link, and the times at which these awaiting messages first requested use of the link.

In a distributed memory computing system, sharing this information between the computers in the system would require additional messages to be passed on the link or a separate communication network used only to handle link access requests. Logic implemented in hardware and software using this separate communication structure would grant permission for nodes to transmit messages on the link according to the times at which transmission requests were received.

The simulations performed by Guibaly [28] use average access time and throughput as a measure of performance. From results of simulations performed by Guibaly, the FIFO protocol is seen to perform better than nonpersistant CSMA and static priority arbitration schemes in most cases, but not as well as dynamic priority schemes. The FIFO protocol does perform better than dynamic priority schemes when there are fewer than 10 nodes attached to the shared link and the traffic level on the link is high. Since TDM and token passing schemes do, in effect, change the priority of the nodes attached to links, a similar performance comparison may be seen in the simulations in this thesis research. It is important to note, however, that the FIFO protocol model used in this research does not account for the additional message passing or communication structures needed to implement a this protocol.

3.3.2. TDM Protocols

As was discussed earlier, TDM protocols are currently used to provide for consistent scheduling of tasks in communication networks in real-time systems [5]. Although efficient real-time scheduling uses additional techniques such as priority schemes to complement TDM protocols, the research in this thesis examines whether the use of TDM protocols will limit the variance of delay in these hypercube topologies while understanding that additional refinements may be necessary to optimize system performance.

One way of refining a TDM system is demonstrated by the reservation ALOHA protocol [15]. ALOHA was developed as a protocol that allows each radio or satellite station in a network to transmit whenever it has messages in its queue. The protocol allows the system to resolve contentions as they occur. As variations of the protocol developed, a slotted ALOHA protocol was created followed by a reservation ALOHA protocol. The slotted ALOHA protocol breaks the use of the transmission media up into separate time frames in much the same way as TDM.

Stations in the network can begin transmitting only at the beginning of frames that are signaled by a single node in the system. Although this limits the times at which each node is allowed to start transmission, contentions still occur when two or more stations attempt to transmit at the same time. To increase the efficiency of this protocol for high traffic conditions, a reservation ALOHA protocol was developed that allows transmitting stations in the system to ask for permission to use a given time slot before using it, thus assuring that no other nodes in the system will contend with the transmitting node's reserved time slot. Except for the time interval at which a station in the network asks for a reserved time slot, this system is equivalent to a TDM system in which one and only one node has permission to transmit on the link at any given time.

Tanenbaum [15] discusses ALOHA and the reasons for ALOHA's evolution. The basic problem is that under light loading conditions, very few collisions will occur on the transmission media. Thus, the additional complexity and time used while stations attempt to reserve time slots is of little benefit. When the traffic load becomes heavy, however, the collision avoidance of the reservation system greatly enhances the overall network performance. This characteristic is quite similar to the problems encountered with TDM systems. At low transmission rates, the time used by time slots of idle nodes will reduce the network's performance. As the data rate increases, fewer nodes in the system will be idle, and the efficiency of the network will increase.

3.3.3. Token Passing Protocols

As Cheng, et. al [29] discuss parameters for the network they are simulating, they mention that the method used to determine the number of messages a node can transmit per token capture is an important issue. Both exhaustive and nonexhaustive techniques are discussed. For the exhaustive technique, a node can transmit all of the messages it currently holds once it captures the token. For the nonexhaustive technique, a node can only transmit one message per token capture. Cheng also states that the exhaustive technique will utilize link bandwidth more efficiently, but the nonexhaustive technique prevents a single node in the network from hogging the link resource. The model used for token passing in this thesis research is slightly different from token ring networks used by Cheng in that the token is passed to nodes sharing a common link bus, not to nodes in a ring arrangement. Several effects of exhaustive and nonexhaustive techniques, though, carry over from one implementation to the next. Specifically, each token transmitted on a link uses bandwidth that could otherwise be used for data transmission. This obviously limits the efficiency of the link.

The token passing model used in the simulations of this thesis uses a hybrid version of exhaustive and nonexhaustive techniques. This hybrid model allows each node to transmit up to three messages per token capture. This should increase the bandwidth utilization by allowing a node to transmit all of its messages most of the time, but it will prevent a single node from dominating the link in cases where a large number of messages have accumulated in a node's queue.

A similar method is used in the IEEE Standard 802.4 protocol to prevent a single node from hogging the link [15]. As was discussed earlier, this protocol is used to implement a token passing access scheme on bus connected architectures. In this protocol, instead of limiting the number of messages a node can transmit on a link during a token capture, timers are used to limit the maximum time a single node is allowed to transmit during a given token capture [15]. Although this method uses a different criteria to limit a node's transmission time, it produces basically the same effect by preventing a single node from excessively using the shared link resource.

3.4. Effectiveness of SLAM

Several simulation tools were considered for simulating the topologies and protocols used in this research. SLAM is a FORTRAN based simulation language used to simulate the queueing networks studied in this research. A discussion of how SLAM is used in this research is contained in Chapter 4. The decision to use the SLAM programming language was based primarily on the research done by Raines, et al. [30,31]. Raines demonstrated how SLAM can be used to model a variety of network topologies. In addition to modeling single stage cube networks, Raines used SLAM to model mesh networks and multistage cubes. Although crossbar switching elements are used in the Raines models, the basic properties of SLAM demonstrated by these simulations showed that many other structures could be studied using SLAM. Also, these simulations showed that many useful results could be obtained from SLAM simulations. The ability and effectiveness of SLAM demonstrated by Raines's research justified the decision to use SLAM for simulations in this research.

Chapter 4. Methodology

This chapter presents the queueing structures used to model the network and the assumptions made concerning the operating conditions of the system. Also, the techniques used to formulate analytic approximations of network delay are described. Finally, this chapter explains how SLAM is used to model networks using simple FIFO protocols and more complex contention schemes.

4.1. The Network Model

To properly examine different interconnection networks, it is necessary to form basic units common to all of the networks under consideration upon which these varied networks can be based. It is possible to construct all of the networks under consideration out of the basic units of a link model and a node model. These models are adapted from the model presented in [21] and are discussed below. The key feature of these models is that they allow the effects of link and node congestion to be considered separately, and thus are suited for research that attempts to isolate the causes of delay in a message passing network.

4.1.1. Assumptions

To model the network topologies presented here, it is necessary to make assumptions about the system being represented. The networks chosen for study are distributed memory networks that use message passing to communicate between processors. The networks considered contain W^D nodes, and numerical results are reported for 64-node networks with $W^D = 4^3$. Other assumptions made about the networks are outlined below. It should be noted that in some of the research, these network specifications are changed to view interesting effects. However, unless otherwise stated, the following rules are used for each network being considered.

1. Exponentially distributed message generation times
2. Exponentially distributed message lengths
3. Fixed node service time (routing time)
4. Infinite buffers at all nodes and links
5. Uniform distribution of message origins and destinations
6. First-in-first-out (FIFO) node service
7. Link service is determined by FIFO, TDM, or token passing protocol
8. The system is in steady state

Exponentially distributed message generation times are appropriate for a network that has a large number of independent sources generating messages [21,30]. Fixed node service time, exponential message lengths, and FCFS node service are inherited from the network model proposed by Denenberg [21]. Infinite buffering is an assumption commonly made in queueing models [32,33]. Studies that examine the amount of buffering needed in an actual system to approximate infinite buffering are conducted by Reed, et. al. [33] and Raines, et. al. [30]. Uniform distribution of message origins and destinations is commonly employed in hypercube network models [12,33]. The link access protocols used in this research are discussed in detail in

Chapter 2. Simulations discussed in Chapter 5 examine the time required for the network to reach steady state.

4.1.2. The Node Model

The basic node model used to construct the system is shown in Figure 4.1. The queueing structure in this model shows a server that routes messages arriving at the node either from one of the K links attached to the node or a message generated from the node's internal processor. The rate at which processors generate messages is denoted by λ_N. Arriving messages are stored in a queue while they await service. The variable λ_{NM} represents the total rate at which messages enter the node queue from all sources. When the node is finished serving a message, it will choose a message from the queue. The rule used to determine which of the messages in the queue it chooses is usually FIFO, but in some of the simulations other queue selection rules are incorporated.

The node service rate, μ_N, represents the rate at which the message is processed by the node including preparing the message for transmission and the calculations performed to route the message to its destination. For the work done with the model in this research, a fixed node service time, independent of message length, is used for each trial. When a message is finished in the server, it is passed on to one of the K links exiting the node or, if the message is at its final destination, to the local processor. As the topology being considered changes, the only part of this model that changes is the number of links, K, connected to each node. This property enables this node model to be used as a basic construction block for a variety of network topologies.

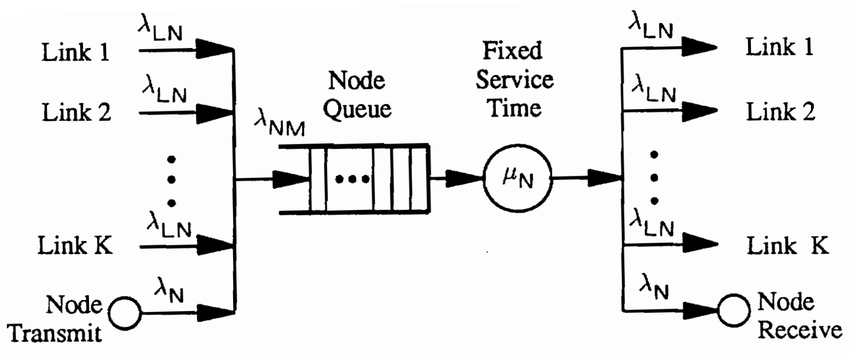


Figure 4.1. The Node Model

4.1.3. The Link Model

The basic link model used to construct the network is shown in Figure 4.2. In this queueing structure, messages from J nodes awaiting transmission arrive at the link queue. The aggregate rate at which messages arrive at the link queue is λ_L. When the link server finishes transmitting a message, it begins transmission of another message from the link queue. The selection rule used to determine which message in the queue is chosen is based on the link contention protocol. In this study, FIFO, TDM, and token ring link protocols are examined. The average service rate of a link is denoted by μ_L. For most of the simulation runs, the time needed to transmit a message is exponentially distributed to represent the fact that variable length messages are transmitted. After service, each message is passed on to the queue of the node it is transmitted to. Modeling an entire network requires that the number of nodes, J, attached to each link and the total number of links in the network be adjusted to represent the chosen topology. A slight modification of this model, shown in Figure 4.3, is used to implement different link protocols such as TDM and token passing. In this model, the global queue in front of the link server is replaced by separate queues at each node input. In this way, selection rules can be implemented that choose messages from specific nodes connected to each link.

4.1.4. The Network-Level Model

To construct a full network model, it is necessary to use one basic link model for each link in the network and one basic node model for each node in the network. The number of links attached to each node in the model is set equal to the number of links attached to each node in the topology being represented. Similarly, the number of nodes in the topology being represented determines the number of nodes used in the model. Nodes and links are connected to each other

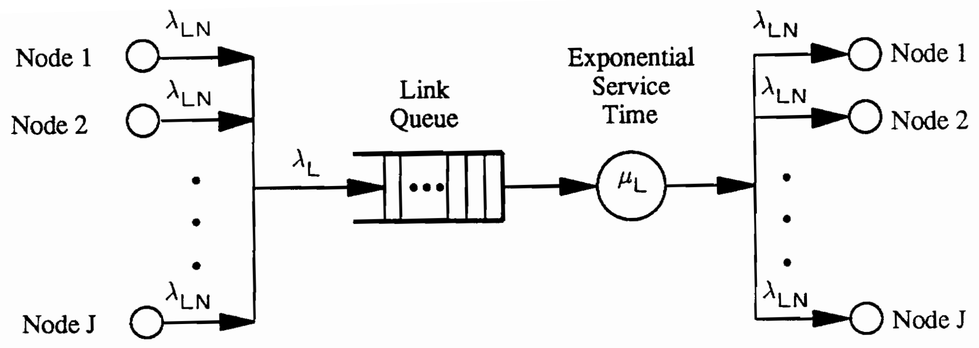


Figure 4.2. Link Model for the FIFO Protocol

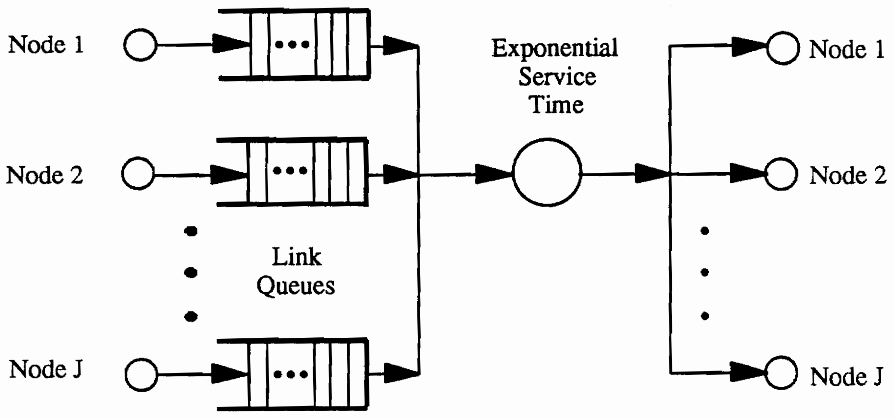


Figure 4.3. Link Model for Other Protocols

in the model in the same fashion as they are connected in the topology. In this way, SBH and torus networks of any size can easily be modeled.

To make the DBH model tractable, only cases where *W* is a multiple of *D-1* are considered. The number of links in the dual bus hypercube in a given secondary dimension is not necessarily the same for all of the secondary dimensions due to the construction technique of having links in only one secondary dimension in each plane perpendicular to the D_0 dimension. Restricting the value of *W* results in an equal number of links in all of the secondary dimensions and thus causes all secondary links to have an equivalently distributed traffic load.

4.2. Analytic Calculations

Equations for the mean and variance of the message passing delay time can be derived using the model. These equations are derived by assuming the system is in steady state and examining the queues and servers in the model in a piecemeal fashion. Initially, the equations for delay in the basic components are presented, and then a description of how these equations are put together to provide analytic results for the mean and variance of message delay is given.

4.2.1. Basic Queues

Two general types of queueing models are used in the network model. The link queue is an infinite queue that feeds a single server with an exponentially distributed service time. This type of queueing model is an M/M/1 queue [32]. The node queues are infinite queues that feed a single server having a fixed service time. This type of queueing model is an M/D/1 queue

[32]. The mean and variance of the delay are calculated for each of these single queueing systems in the sections below.

4.2.1.1. The Link Queue

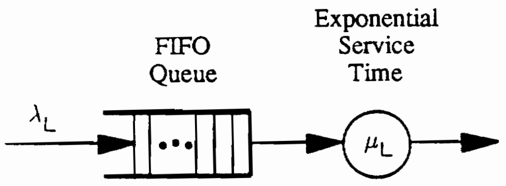
The M/M/1 queue that models each of the links in the system has been studied in great detail in many queueing theory books. Specifically, results concerning many aspects of the delay time for messages in such queueing systems are presented in [32]. A diagram of an M/M/1 queue is shown in Figure 4.4a. The rate at which entities arrive at this queue is λ_L. Messages in the queue are serviced in a first-come-first-served (FCFS) manner. Service times are exponentially distributed with an average rate of μ_L. The utilization, ρ_L, of the system is

$$\rho_L = \frac{\lambda_L}{\mu_L} . \tag{4.1}$$

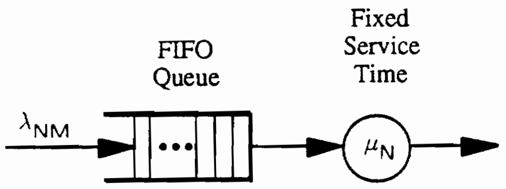
For an M/M/1 queue, as for any queueing model, the total system time for a message is the time the entity waits for service, T_{LW}, plus the time spent serving the entity, T_{LS}. Also, the wait time and the service time distributions are independent of each other. The mean and variance of delay can thus be calculated from the first and second moments of the service and wait times in an M/M/1 queue. Equations for the distribution of the service and wait times for such systems are given in [32] and are used to obtain

$$E[\, T_{LW} \,] = \frac{\rho_L}{\mu_L \,(\, 1 - \rho_L \,)} , \tag{4.2}$$

$$E\,[\, T_{LW}^{\,2} \,] = \frac{2\rho_L}{\mu_L^{\,2}\,(\, 1 - \rho_L \,)^2} , \tag{4.3}$$



(a) M/M/1 Queue



(b) M/D/1 Queue

Figure 4.4. Basic Queues

$$E[\,T_{LS}\,] = \frac{1}{\mu_L}\,, \tag{4.4}$$

and

$$E[\,T_{LS}^2\,] = \frac{2}{\mu_L^2}\,. \tag{4.5}$$

4.2.1.2. The Node Queue

Equations for the M/D/1 queue that models each of the nodes in the system are more difficult to obtain than the M/M/1 queue equations. A diagram of an M/D/1 queue is shown in Figure 4.4b. The rate at which entities arrive at the queue is λ_{NM}. Messages in the queue are serviced in a first-come-first-served manner. The service time, N_{ST}, at the queue is a constant and is related to the service rate, μ_N, by

$$N_{ST} = \frac{1}{\mu_N}\,. \tag{4.6}$$

Also, the node server utilization is

$$\rho_N = \frac{\lambda_N}{\mu_N}\,. \tag{4.7}$$

The total service time for a message at the node, T_R, is equal to the time the message spends waiting in the queue, T_{NW}, plus the service time of the server. Since the service time at the server is constant, the variance of the total service time is equal to the variance of the time the message spends in the queue. The equations for the mean and variance of the message delay in a node are

$$E\,[\,T_R\,] = E\,[\,T_{NW}\,] + N_{ST} \tag{4.8}$$

and

$$E\,[\,T_R^2\,] = E\,[\,T_{NW}^2\,] - E\,[\,T_{NW}\,]^{\,2}\,. \tag{4.9}$$

Equations for the first and second moment of the waiting time in an M/D/1 queue as calculated in [32] are

$$E[T_{NW}] = \frac{\rho_N \times N_{ST}}{2\,(\,1 - \rho_N\,)} \tag{4.10}$$

and

$$E[T_{NW}{}^2] = \frac{\rho_N \times N_{ST}{}^2}{3\,(\,1 - \rho_N)} + \frac{(\,\rho_N \times N_{\,ST}\,)^2}{2\,(\,1 - \rho_N)^2}\,. \tag{4.11}$$

With these calculations, all of the pieces needed to calculate the first and second moments of the time a message spends in an M/D/1 queue are available.

One approximation that is made for the network is that the output of an M/D/1 queue is exponentially distributed. The validity of this approximation is verified in [21]. Thus, the times at which messages flow out of the node server are approximately exponentially distributed. This approximation becomes better as the traffic in the network decreases. Thus, the analytic results can be expected to become closer to actual results as traffic in the network is decreased.

4.2.2. Queueing Network Calculations

Some basic properties exist for all three network topologies being studied in this research that are beneficial when creating analytic equations to describe mean and variance of delay. Messages travel through the networks in a similar fashion. Each node creates messages at exponentially distributed time intervals with an average rate of λ_N. The newly created message is passed to the node queue where it awaits service. The deterministic node service time represents the time it takes a communication processor to determine which link the message should be passed to or whether the message has reached its final destination. If the node detects that the message has reached its destination it will be removed from the network. Otherwise, the node will select the link to which it should pass the message. Upon exiting the node, the message will enter a queue and await service at the link which the node has chosen. The message eventually reaches the link server and is delayed an exponentially distributed amount of time that represents time needed to transmit a message with an exponentially distributed length. When transmission is complete, the message arrives at the node queue and this process is repeated. From this cycle of events, it can be seen that the message initially incurs a routing delay at the node in which it is created, and then incurs both a transmission and routing delay at each link-node pair it passes through as it hops through the network.

It should be noted that in the following equations the time to route each entry at the node queue is represented as a single quantity, but the time to transmit a message on a link is separated into the waiting time and the service time. This is because of the difference between node and link service times in the system. The node service time is fixed, and thus all messages in the network will encounter the same distribution of delay for both the node waiting time and the node service time. This is not the case with the link queueing structure. All messages do encounter the same distribution for the amount of time they must wait in a queue before being service by a link, but the service time for messages on a link is not the same for all messages in the system. The service time of a given message is the same for all links over which the message is transmitted. Thus, handling the distribution of the link waiting time separately from the link service time accounts for the facts that link service times are exponentially distributed and that the link service time of a given message is the same for all hops the message takes while travelling from its source to its destination.

Typical message delay can be calculated by examining how many hops the message must take and then examining the delay that the message will undergo for each hop as it passes through the network. Since differences exist in the network structure as the topology is changed, the mean and variance of delay need to be calculated differently for each of the topologies being studied.

4.2.2.1. Analytic Equations for the Spanning Bus Hypercube

To study the traffic in any of the topologies, it is first necessary to examine the number of hops, δ, a message needs to take from its source to its destination. To make it easier to calculate both the first and second moments of this number of hops, the probability that a message will require n hops in the network is calculated. From our initial network assumptions, message origins and destinations are assumed to be uniformly distributed. Each hop in the network is equivalent to changing one of the D different coordinates in the network. Thus, if a message requires n hops, then n of the D coordinates must change value. The number of different ways this can happen is equal to the combination of D items taken n at a time. Each of the n coordinates that change can change to any of W possible values other than its current value. Thus, each of the n changed coordinates can take on one of W-1 possible values. Multiplying the number of different coordinates that can change by the number of different values that they can change to will yield the total number of nodes that can be reached in *exactly* n hops. The probability of the message requiring n hops is equal to the number of nodes that can be reached in n hops divided by total number of nodes that the message can be transmitted to, i.e. the number of nodes in the network minus the node that originates the message since nodes do not transmit messages to themselves. Putting this logic into a mathematical form yields the following equation for the probability that a message will require n hops is

$$\text{Prob}\,(\delta{=}\text{n}) = \frac{\text{D!}\ (\text{W} - 1)^{\text{n}}}{\text{n!}\ (\text{W}^{\text{D}}\text{-}1)\ (\text{D} - \text{n})!}\,. \tag{4.12}$$

From this equation and the fact that the maximum distance between any two nodes in a SBH network is D, the first and second moments of δ are

$$E[\delta] = \sum_{n=1}^{D} [\ n \times Prob(\delta{=}n)\ ] \tag{4.13}$$

and

$$E[\delta^2] = \sum_{n=1}^{D} [\ n^2 \times Prob(\delta{=}n)\ ]. \tag{4.14}$$

Using this knowledge of the number of hops a typical message will take, it is possible to calculate the amount of traffic in the network at steady state. Each of the W^D nodes in the network generates messages at the rate of λ_N. Also, each of these messages take an average of $E[\delta]$ hops. Since node message generation is exponentially distributed and the superposition of exponential distributions is also exponential, the average rate of traffic in all links of the network, denoted λ_{NT}, is

$$\lambda_{NT} = E[\delta] \times \lambda_N \times W^D. \tag{4.15}$$

Because the system is assumed to be in steady state and the message origins and destinations are uniformly distributed, the message rate at each link in the network is equal to the total link traffic rate divided by the number of links in the network. For the SBH, the number of links in the network, L, is

$$L = D \times W^{D-1}. \tag{4.16}$$

Thus, the link traffic rate, denoted λ_L , is

$$\lambda_L = \frac{\lambda_{NT}}{L} = \frac{E[\delta] \times \lambda_N \times W^D}{D \times W^{D-1}} = \frac{E[\delta] \times \lambda_N \times W}{D}\ . \tag{4.17}$$

In the SBH, each link is connected to *W* different nodes, thus the rate of traffic on a given link going to a specific node, denoted λ_{LN}, is

$$\lambda_{LN} = \frac{\lambda_L}{W} = \frac{E[\delta] \times \lambda_N}{D} . \tag{4.18}$$

Each node in the SBH topology is connected to *D* different links, and each of these links supplies messages at the rate of λ_{LN}. Also, each node generates messages at the rate of λ_N. Thus, the total rate of traffic entering each of the nodes, λ_{NM}, is

$$\lambda_{NM} = \lambda_N + (D \times \lambda_{LN}) = \lambda_N + (E[\delta] \times \lambda_N) . \tag{4.19}$$

This is the final system parameter needed to calculate the mean and standard deviation of the message delay in the system.

Using the values for λ_L and λ_{NM} calculated in Equations 4.17 and 4.19, respectively, in the basic queueing calculations of Equations 4.1 through 4.9, the average end-to-end message delay is

$$E\,[T_{EE}] = E\,[T_R] + (E\,[\delta] \times E\,[T_R]) + (E\,[\delta] \times E\,[T_{LW}]) + (E[\delta] \times E\,[T_{LS}]). \tag{4.20}$$

To find the variance of the end-to-end delay, some simplifying assumptions are made. All of the terms in the equation used to calculate the mean of the message delay are assumed to be independent. Because E[δ] exists in each of these terms, this independence is not exact. Simulations examined later in this thesis, however, verify that accurate results can be obtained with this assumption. If all terms are independent, then the variance of the end-to-end delay is equal to the sum of the variance of all of the terms in this equation. Again, using Equations 4.1 through 4.9, the variance of delay is

$$\sigma^2_{EE} = \sigma^2_{T_R} + \sigma^2_{\delta T_R} + \sigma^2_{\delta T_{LW}} + \sigma^2_{\delta T_{LS}} \,. \tag{4.21}$$

Each of the terms of this equation is calculated below.

$$\sigma_{T_R}{}^2 = E\,[\,T_R{}^2\,] - (\,E\,[\,T_R\,]\,)^2 \tag{4.22}$$

$$\sigma_{\delta T_R}{}^2 = \left(\sum_{n=1}^{D} \mathrm{Prob}(\delta{=}n) \times \{\, (n \times E[T_R])^2 + n \times \sigma_{T_R}{}^2 \,\} \right) - (\,E[\delta] \times E[T_R]\,)^2 \tag{4.23}$$

$$\sigma_{\delta T_{LW}}{}^2 = \left(\sum_{n=1}^{D} \mathrm{Prob}(\delta{=}n) \times \{\, (n \times E[T_{LW}])^2 + n \times \sigma_{T_{LW}}{}^2 \,\} \right) - (\,E[\delta] \times E[T_{LW}]\,)^2 \tag{4.24}$$

$$\sigma_{\delta T_{LS}}{}^2 = (\,E\,[\delta^2] \times E\,[T_{LS}{}^2]\,) - (\,E\,[\delta] \times E[T_{LS}]\,)^2 \tag{4.25}$$

where

$$\sigma_{T_{LW}}{}^2 = E[T_{LW}{}^2] - (\,E[T_{LW}]\,)^2 \tag{4.26}$$

4.2.2.2. Analytic Equations for the Torus

The equations for the mean and variance of message delay for the torus are the same as for the SBH except for the underlying terms representing the traffic rates in the network and the number of hops required for a message transmission.

For the torus topology, analytic equations for the probability of a message needing to take a given number of hops were not discovered. Instead, message hop probability is determined for specific values of W and D by calculating the number of hops needed to reach each node in the network from node (0,0,...,0), and dividing the number of nodes that are a given distance away by the number of nodes in the network minus one. From these results, it is possible to calculate the first and second moments of δ in the same manner used for the SBH.

The total traffic rate of the network is

$$\lambda_{NT} = E[\delta] \times \lambda_N \times W^D \,. \tag{4.27}$$

The number of links in a W^D torus is

$$L = D \times W^D. \tag{4.28}$$

As with the SBH, traffic is uniform in the network so the traffic rate on each link is

$$\lambda_L = \frac{E[\delta] \times \lambda_N \times W^D}{D \times W^D} = \frac{E[\delta] \times \lambda_N}{D} \,. \tag{4.29}$$

Each link is attached to two nodes, so the rate of traffic on a given link going to a specific node is

$$\lambda_{LN} = \frac{E[\delta] \times \lambda_N}{2 \times D} \,. \tag{4.30}$$

Each node has two links attached to it in each of D dimensions, and each node generates messages itself. Thus, the total rate of traffic arriving at the node queue is

$$\lambda_{NM} = \{ 2 \times D \times \lambda_{LN} \} + \lambda_N = (E[\delta] + 1) \times \lambda_N \, . \qquad (4.31)$$

This is the final parameter needed to calculate the mean and variance of message delay. All calculations for the end-to-end delay and the variance of this delay are the same as for the SBH except that Equations 4.29 and 4.31 are used to calculate the link and node traffic rates instead of Equations 4.17 and 4.19.

4.2.2.3. Analytic Equations for the Dual Bus Hypercube

The DBH calculations are more complicated than those above because the traffic in the DBH is not uniform. Links in the primary dimension must be considered differently than links in the secondary dimensions. These analytic results require that W be a multiple of D-1 so that traffic on all of the secondary links is equal. To calculate the traffic on each link, it is first necessary to calculate the average number of hops a message must take. Initially it is necessary to calculate the number of hops, δ, using the same method used for the torus topology. Next, this calculation is repeated, but the number of hops taken on primary links is calculated separately from the number of hops taken on secondary links. For specific values of *W* and *D*, choose one node as a reference node, and then examine each of the other nodes one node at a time and count how many hops on primary links must be taken from this node to the reference node. After performing this calculation on all nodes in the network, it is possible to check how many nodes require one hop on primary links to the reference node, how many nodes require two hops on primary links, and so forth. When all of the nodes are considered, divide the number of nodes that require *n* hops on primary links by the total number of nodes minus the one reference node. This result gives the probability that a generated message will require *n* hops on the primary links. It is necessary to calculate δ_1, the number of hops on primary links, and δ_2, the number of hops on secondary links, separately. δ equals δ_1 plus δ_2. The necessary first and second moments of these three variables are

$$E[\delta_1] = \sum_{n=1}^{\delta_{1_{MAX}}} [\, n \times \text{Prob}(\delta_1 = n) \,] \,, \tag{4.32}$$

$$E[\delta_2] = \sum_{n=1}^{\delta_{2_{MAX}}} [\, n \times \text{Prob}(\delta_2 = n) \,] \,, \tag{4.33}$$

$$E[\delta] = \sum_{n=1}^{\delta_{MAX}} [\, n \times \text{Prob}(\delta = n) \,] \,, \tag{4.34}$$

and

$$E[\delta^2] = \sum_{n=1}^{\delta_{MAX}} [\, n^2 \times \text{Prob}(\delta = n) \,] \,. \tag{4.35}$$

These values are used to calculate the total traffic rate on primary and secondary links as shown below.

$$\lambda_{NT_1} = E[\delta_1] \times \lambda_N \times W^D \tag{4.36}$$

and

$$\lambda_{NT_2} = E[\delta_2] \times \lambda_N \times W^D. \tag{4.37}$$

The number of primary and secondary links in a DBH network are given by

$$L_1 = W^{D-1} \tag{4.38}$$

and

$$L_2 = W^{D-1}. \tag{4.39}$$

Thus, the link traffic rates are

$$\lambda_{L_1} = E[\delta_1] \times \lambda_N \times W \tag{4.40}$$

and

$$\lambda_{L_2} = E[\delta_2] \times \lambda_N \times W. \tag{4.41}$$

Each node in the network receives messages from one primary link, one secondary link, and from its processor. Thus, the traffic rate into each node in the network is given by

$$\lambda_{NM} = \lambda_{L_1} + \lambda_{L_2} + \lambda_N = \{ [(E[\delta_1] + E[\delta_2]) \times W] + 1 \} \times \lambda_N . \tag{4.42}$$

The end-to-end message delay is calculated in the same way as the other two topologies except that there is an extra term to account for the two different types of links that exist in the network. It should be noted that calculations for the link queue given in Equations 4.1 through 4.5 are performed in the same fashion as was explained earlier, but they are done once for each type of link. Thus, the mean end-to-end delay equation becomes

$$\begin{aligned} E\,[T_{EE}] = E\,[T_R] + \{ E\,[\delta] \times E\,[T_R] \} + \{ E\,[\delta_1] \times E\,[T_{LW1}] \} \\ + \{ E\,[\delta_2] \times E\,[T_{LW2}] \} + \{ E\,[\delta] \times E\,[T_{LS}] \} . \end{aligned} \tag{4.43}$$

The link and node traffic rates obtained from Equations 4.40 through 4.42 are used to calculate the basic queueing delays as were given in Equations 4.1 through 4.9. Before calculating the standard deviation of delay, some simplifying assumptions are made. If all terms of an equation are independent, then the variance of a the sum is equal to the sum of the variances. In the above equation, it is obvious that there is some dependence between terms because some of the terms contain the same variables. For these analytic results, however, all of the terms are assumed to be independent. With this assumption, the variance of the end-to-end delay is

$$\sigma^2_{EE} = \sigma^2_{T_R} + \sigma^2_{\delta T_R} + \sigma^2_{\delta_1 T_{LW1}} + \sigma^2_{\delta_2 T_{LW2}} + \sigma^2_{\delta T_{LS}} \quad (4.44)$$

where,

$$\sigma^2_{T_R} = E\,[\,T_R^{\,2}\,] - \{\,E\,[\,T_R\,]\,\}^2, \quad (4.45)$$

$$\sigma^2_{\delta T_R} = \left(\sum_{n=1}^{\delta_{MAX}} \mathrm{Prob}(\delta{=}n) \times \{\, (n \times E[T_R])^2 + n \times \sigma_{T_R}^{\,2} \,\} \right) - (\, E[\delta] \times E[T_R]\,)^2, \quad (4.46)$$

$$\sigma^2_{\delta_1 T_{LW1}} = \left(\sum_{n=1}^{\delta_{1MAX}} \mathrm{Prob}(\delta_1{=}n) \times \{\, (n \times E[T_{LW1}])^2 + n \times \sigma_{T_{LW1}}^{\,2} \,\} \right) - (\, E[\delta_1] \times E[T_{LW1}]\,)^2, \quad (4.47)$$

$$\sigma^2_{\delta T_{LW2}} = \left(\sum_{n=1}^{\delta_{2MAX}} \mathrm{Prob}(\delta_2{=}n) \times \{\, (n \times E[T_{LW2}])^2 + n \times \sigma_{T_{LW2}}^{\,2} \,\} \right) - (\, E[\delta_2] \times E[T_{LW2}]\,)^2, \quad (4.48)$$

and

$$\sigma^2_{\delta T_{LS}} = E\,[\delta^2] \times E\,[T_{LS}^{\,2}] - \{\,E\,[\delta] \times E[T_{LS}]\,\}^2. \quad (4.49)$$

Also,

$$\sigma^2_{T_{LW1}} = E[T_{LW1}^{\,2}] - \{\,E[T_{LW1}]\,\}^2 \quad (4.50)$$

and

$$\sigma^2_{T_{LW2}} = E[T_{LW2}^{\,2}] - \{\,E[T_{LW2}]\,\}^2\,. \quad (4.51)$$

4.3. Simulations

The results obtained from the above equations can predict the mean and standard deviation of delay for a large range of network sizes. The drawback of these calculations is that they use simple idealistic queueing models, and cannot easily be modified to provide for more advanced protocol structures. Also, the effects of the assumptions made in this model concerning the independence of the variables and the distribution of the message times are believed to be small, but are not known precisely. To verify the analytic equations and adapt the model to other network parameters, the SLAM simulation language is used to model these network topologies. With SLAM, it is possible to use more complex link and node queueing disciplines to model other communication protocols. Also, simulation runs performed on the FIFO queueing models help to validate the analytic results obtained earlier.

4.3.1. Description of SLAM

The *S*imulation *L*anguage for *A*lternative *M*odeling is marketed by the Pritsker Corporation. This FORTRAN-based simulation language enables the user to build complicated queueing networks to simulate real world events. A complete description of SLAM is contained in [34]. The capabilities of SLAM include the creation and termination of system entities, the assignment of attributes to entities, and the construction of queueing structures. A large number of built-in statistical distributions allow the user to easily model many types of random events. This language also keeps statistics on system events such as how long a message has been in the system and the utilization of each queue and server in the network. Finally, it is important to mention that SLAM also supports the use of FORTRAN subprograms. This feature enables the user to create routing schemes for each of the topologies used in this research and to provide a way of controlling several different link protocols.

4.3.2. SLAM Models

Many different networks were created using SLAM. Even though some of the essential network parameters, such as the topology or the protocol used to resolve link contention, are varied considerably, some basic units of the SLAM code remain similar for all simulation runs. The following paragraphs discuss the basic elements used to construct the simulation network. This discussion is followed by an explanation of how these basic elements are modified to simulate the various network protocols under consideration.

4.3.2.1. Basic SLAM Units

Just as the analytic model is built from the basic units of M/M/1 and M/D/1 queues, the simulation models use small building blocks. The three basic types of code used to construct the network models are the node code, the link queueing code, and the link service code. These basic coding elements are shown in Figure 4.5 and discussed below. The method used to number the nodes and links in the simulations is described in Appendix A.

4.3.2.1.1. The Node Code

The basic block of code used to simulate the function of each of the nodes in all three network topologies is shown in Figure 4.5a. These lines of code are repeated for each node in the network. The CREATE line of the code schedules the creation of messages at time intervals specified by an external FORTRAN subprogram. This subprogram, designated USERF(4), specifies that message generation times should be exponentially distributed with a mean of λ_N. After a message is generated, it receives attributes that will be attached to the message as it travels through the network. These attributes include the node that the message is currently at, the destination node of the message, and the amount of time it will take to transmit this message across a link network. The node that the message is currently at, ATRIB(2) is set equal to the node that created the message. The destination node of the message, ATRIB(3), is calculated in a FORTRAN subprogram. This subprogram chooses a destination node from a random uniform distribution, and makes sure that this destination is not equal to the node where the message originated. Thus, no node will attempt to transmit messages to itself. A final FORTRAN subprogram is used to calculate the transmission time of the message, ATRIB(5), from an exponential distribution.

Upon receiving its initial attributes, the message is passed on to the queue that contains all messages awaiting service at this node. The ACTIVITY statement pulls messages from the queue one by one and delays them for an amount of time determined by the USERF(6) subprogram. This subprogram sets all node message delay times equal to the fixed node service time. The ACTIVITY statement will not pull a message from the queue until the message it is currently serving has waited for its allotted delay time. Upon leaving the ACTIVITY, each message will be assigned a value by the USERF(2) subprogram. This function looks at the message's current node and destination node and picks which link it should be transmitted on

```
        CREATE,USERF(4),0,1;
        ASSIGN,ATRIB(2)= X;
        ASSIGN,ATRIB(3)=USERF(7);
        ASSIGN,ATRIB(5)=USERF(5);
NX      QUEUE(X);
        ACT(1)/1,USERF(6);
        ASSIGN,ATRIB(4)=USERF(2);
        GOON,1;
          ACT,0,ATRIB(2).EQ.ATRIB(3),RCVD;
          ACT,0,ATRIB(4).EQ.1,LX;
          ACT,0,ATRIB(4).EQ.2,LX;
          ACT,0,ATRIB(4).EQ.3,LX;
```

(a) NODE CODE

```
LX      QUEUE(X),,,,SX;
```

(b) LINK QUEUE CODE

```
SX      SELECT,NQS(X),,,LX,LX,LX,LX,TOKX;
        ACT(1)/X,ATRIB(5);
        ASSIGN, ATRIB(2)=USERF(1),
                ATRIB(6)=0,
                ATRIB(6)=USERF(3);
        GOON,1;
          ACT,0,ATRIB(2).EQ.1,NX;
          ACT,0,ATRIB(2).EQ.2,NX;
          ACT,0,ATRIB(2).EQ.3,NX;
          ACT,0,ATRIB(2).EQ.4,NX;
          ACT,0,ATRIB(2).EQ.0,TOKX;
```

(c) LINK SERVICE CODE

X = Integer whose value is depends depending on the topology used and the node, link, or server in the network it is representing.

FIGURE 4.5. Basic SLAM Code Elements

next. Attribute 4 is marked with the value of the next link. This attribute is then examined by the GOON and ACTIVITY statements that send the message to the queue used to hold messages while they await transmission on this specified link. Also, it is this phase of operation where the destination of the message is checked to see if the message has reached its destination. If it has, the message is sent to a terminating node where statistics on its travel time are recorded and the message is purged from the network.

4.3.2.1.2. Link Queueing Code

The link queue code consists only of the one line shown in Figure 4.5b. This line is repeated for every link queue in the network. This line specifies the SLAM queue number that will hold all of the messages awaiting transmission on a given link. Also, the link server, or SELECT statement, that will service messages in this queue is specified.

4.3.2.1.3. Link Service Code

Code for the link service routine is shown in Figure 4.5c. Messages enter link service through the SELECT statement. Each SELECT statement in the network consists of an argument for the NQS subroutine and a list of links. The list of links in the SELECT statement contains all of the link queues that contain messages that are waiting to be transmitted on the link. The NQS subroutine is called whenever a link server is ready to accept a message from a queue. The NQS subroutine decides which of the queues listed in the SELECT statement a message should be chosen from. The argument passed to the NQS subroutine identifies the number of the SELECT statement that is requesting service. Once a message is chosen, it is sent to the link

ACTIVITY statement. This ACTIVITY delays the message for the amount of time specified by the message's fifth attribute. No other message will be chosen by the SELECT statement until the link ACTIVITY has delayed the message for this specified time period. This simulates the time a link spends transmitting a message. Once the link service ACTIVITY has been completed, the message has several of its attributes reassigned. First, the current node location of the message is changed to reflect the fact that the message has been transmitted to a different node in the network. Additional link service assignments may be included that perform optional protocol functions such as updating the token count as tokens are passed around the ring. These optional assignments are discussed in further detail in the next section of this chapter. After the message has had its attributes modified, GOON and ACTIVITY statements are used to pass the message to the queue of the node that receives the message from the link transmission.

4.3.2.2. Protocol Modeling in SLAM

Although the system parameters provide most of the information needed to construct a network model, additional specifications are needed to simulate specific link protocols. The method used to model each of these protocols in SLAM is discussed in the paragraphs below.

4.3.2.2.1. FIFO Model

The first simulations use the same network model that is used for the analytic calculations. The default priority setting for queues in a SLAM network is FIFO. Thus, no programming adjustments are needed to change the priority of the system queues. This simple model also has only one queue per link, so all messages awaiting transmission on each link in the network are contained in a single queue. Every node connected to a given link places all messages that it wants to transmit on that link in the same queue. Thus, for this model, the link server has only one queue from which to choose messages. This makes the arbitration logic of the SELECT statement of the link service code unnecessary, and thus the link service ACTIVITY statement can directly follow each link queue. This model is the simplest of the protocol models, and it will simulate the same network for which analytic equations were derived.

4.3.2.2.2. Token Passing Model

The token passing model is designed to give each node an equal chance of obtaining transmission privileges on the link. Each node is allowed to transmit up to three messages each time it receives the token. If a node has no messages to transmit, or if it has already transmitted its three message maximum, it passes the token to the next node along the link. To distinguish which nodes are attempting to transmit messages on the link, each link contains one queue for each node to which it is connected. Thus, for $W^D=4^3$ network, there are four queues per link in the SBH and DBH topology models, and two queues per link for the torus topology model. In the node code for this model, each node has a separate queue for each link it is attached to. When a node finishes its routing calculations on a packet, it passes this message to the queue associated with the link that the message is to be transmitted upon. The link queueing code for this protocol contains multiple queues per link as was explained above. It should be noted that while the token protocol chooses which node should transmit the next message, it uses a FIFO priority for each individual queue. Thus, all of the messages from a given node will be transmitted on a first come first served basis by the link, but the total link traffic will also contain messages from other nodes connected to the link. Messages from separate nodes in the network are not necessarily transmitted in a FIFO manner.

The SELECT node of the link service code makes the choice of which queue it will select a message from for each transmission. In addition, the link queue must decide whether it should transmit a token instead of transmitting a message on the link. The NQS function carries out the calculations for link arbitration. The argument of this function specifies which link is being considered. The program checks status variables to see which node currently has control of the link, and sees whether this node has additional messages available for transmission and whether the node has already transmitted its maximum of three messages. From these considerations, the NQS function determines whether a message should be transmitted from one of the nodes connected to the link or whether a token should be sent on the link to pass on transmission permission to the next node in the network. After a message is serviced by the link service ACTIVITY statement, it goes to an assign node that examines the attributes of the message or token and updates the link status to reflect which node now has control of the link and how many messages this node has already transmitted during its current token capture. If a message is transmitted, the GOON node of the link service code places the message in the queue of the node that it is transmitted to. If a token is transmitted, the GOON node returns the token to a bin queue that holds all of the tokens when they are not being transmitted. In this way, tokens and messages share the transmission network and simulate the effects of a typical token passing protocol. The amount of time that it takes to transmit a token on a link is varied in order to see how this timing overhead affects the network's performance.

4.3.2.2.3. TDM Protocol Model

Another protocol that is commonly encountered in communication systems is time division multiplexing. To simulate this protocol, a global clock is created in the network. The period of the global clock is varied to examine how it affects the network's performance. For each link, only a single node is given permission to transmit during a given clock period. The node code used to simulate this effect is the same as is used for the token ring case. Each link has one queue associated with it for each node it is connected to. Each of these queues uses a FIFO protocol to order the messages it stores. Each of the separate link queues, however, competes for transmission time on the link, and the messages from all of the nodes connected to the link are transmitted in an interspersed manner. For each clock period, a unique node has the right to transmit messages. The permission is passed from one node to the next in a cyclic fashion at the end of each clock cycle. When the link finishes transmitting a message, the NQS function of the SELECT statement of the link service code chooses which node's associated link queue it should service a message from. This decision is based solely on looking at the current period of the global system clock. By looking at this clock, the link knows which node has the right to transmit a message. If this node does not have any messages to transmit, the link will remain idle until this chosen node presents a message that needs to be transmitted or until the next clock cycle starts. Each time a message appears at the output queue of a node for a link, the queue looks ahead to see if it has permission to transmit and if the link is idle. If so, the message will be transmitted. If not, the message waits in the queue until it is requested by the link server.

In SLAM, a SELECT node will only check to see if queues have messages to be transmitted when the server finishes service of a prior message or when a message arrives at a queue designated to be served by that server. Thus, once a queue server is idle, it will not transmit again until a message arrives at one of its designated queues. Thus, if a message is waiting in the output queue of a node, but this node does not have permission to transmit in the current time slot, the link server would not normally check this node at the time when the clock period changes to a state where this queue now has permission to transmit. Thus, this message would have to wait until another message arrived to signal the link server that its services were needed.

To correct this problem, zero length messages are generated and sent to a specially designated queue for each link when the clock period changes. These special queues contain only zero length messages, and are always the first queue checked by each link server. Since these messages take no time to transmit, they do not affect the delay time results of the simulation. What these messages do is force the server to look for messages that may need to be transmitted each time the transmission permission of the link changes. Thus, if a link is in the middle of transmitting a message when the permission is changed, it will complete the transmission of the current message. It will then transmit the zero length message in zero time, and upon completion of the zero length transmission, it will check to see if the node that currently has permission has any messages to transmit. If a link is idle at the time the permission is changed, it will be signaled by the arrival of the zero length message and proceed to transmit it. It will then check the node that has just received permission to transmit to see if it has any messages to transmit. In this way, the link server never remains idle at times when it should be transmitting messages. When a server is finished transmitting a zero length message, it will terminate the special message from the network. This technique allows the model to simulate the effect of a TDM permission scheme on the system performance.

4.3.3. Protocol Variations

In addition to these protocol models, other slight variations are made to the network protocols. These changes try to isolate the effects of some of the system parameters and help develop more efficient network protocols. These modifications can be grouped into two different categories. The first category consists of changes that are made to the output priorities of the network queues, and the second consists of changes made to the characteristics of the messages created by the network. It should be noted that the modifications discussed above are observed only for the

SBH topology. Preliminary results for the DBH and torus not discussed in this thesis demonstrated that the effects observed for the SBH are similar for the other topologies.

4.3.3.1. Queueing Priority Modifications

For the previously mentioned simulation models, all of the queues in the network use FIFO priority when deciding which message to output. The different protocols for link access are created not by changing the priority scheme of individual link queues, but by changing the number of queues associated with each link and the way in which queues are interconnected. Additional control can be added to the protocol by changing the FIFO priority for the queues.

Three separate alternative priorities are used. The first such modification has all queues give priority to the oldest message they contain. Message age is defined as the current time of the simulation minus the time at which the message was created. Thus, the longer a message remains in the network, the higher its priority becomes. This priority scheme should help to limit the maximum time that any given message can expect to spend in the network.

The two other priority schemes used are similar to each other. One sets the queueing priority to give preference to longer length messages, and the other gives preference to messages of shorter length. Message length, as was explained earlier, is what determines how long it will take to transmit the message on a link. It is important to note, however, that these queueing priorities are used for both node and link queues in the network. Thus, even though node service time is equal for all messages, the nodes still give preference according to the amount of time needed to transmit these messages on links.

4.3.3.2 Message Parameters

For the standard network model used in this research, message length is assumed to be exponentially distributed and message destinations are assumed to be uniformly distributed. Modifications are made to the simulation models to see how the above mentioned assumptions affect the message transmission delay. Simulations are run for the case where all message lengths are equal instead of being exponentially distributed. Then, exponential message lengths are again used, but the number of hops each message needs to take from its source to destination is constant. For the final simulation runs, both a constant message length and a constant number of hops for each message in the network is used. With these simulations, it is possible to measure the way each of the initial network assumptions affects the message delay results.

4.4. Summary

This chapter discussed the network model that is used to obtain the results presented in Chapter 5. Analytic equations obtained from the model were presented that predict the mean and standard deviation of the message delay for systems with the three topologies studied in this research. Modifications of the model and methods for simulating them using SLAM were then discussed. These modifications included changes in the communication protocol and changes in the initial network assumptions. From the results obtained using the methods presented in this chapter, it is possible to study a variety of characteristics associated with message passing delay in distributed memory computers.

Chapter 5. Results

The models proposed in Chapter 4 are used to obtain results for each of the topologies and protocols discussed in this research. Initial simulations performed using the SLAM simulation language verify the analytic equations derived to calculate the mean and variance of delay for the three different topologies using a FIFO link access protocol. These results are analyzed to determine the performance characteristics of the three topologies under consideration. The simulation program is then modified to model TDM and token passing protocols. The results of these simulations are compared to the results obtained with the FIFO protocol, and performance characteristics of each protocol are identified. Finally, the simulation program is modified to examine the effects of queueing priority schemes, message lengths, and the distance between each message's source and destination nodes in order to identify the contribution of each of these parameters to delay variance.

5.1. The Simulation Parameters

To make the results obtained in these simulations both informative and tractable, it is necessary to hold several system parameters constant. The size of the networks being simulated is set to $W^D = 4^3$. Because timing in any system is relative, a standard must be created to which all results can be compared. For these simulations, the unit of comparison is chosen to be the average time between the creation of consecutive messages at a given node. This time is set to one time unit. Thus the message generation rate at each node is

$$\lambda_N = 1.0 \ .$$

For this node generation rate, it is necessary to determine base case link and node processing rates representative of high and low traffic conditions. The traffic level in a system is measured by the utilization of nodes and links. Based on the Intel iPSC and NCUBE machines, as discussed in Chapter 2, the ratio of the node processing rate to the link processing rate is chosen to be 2-to-1. For this processing ratio and value of λ_N, it is found from the analytic equations for the FIFO protocol that utilization ratios of the link and node servers in the model, ρ_L and ρ_N, will vary from approximately 0.1 to 1.0 as the link service rate, μ_{LST}, is varied from 2.5 to 17.5. This corresponds to the link service time being varied from 0.05 to 0.4. Utilization ratios approximately equal to 0.1 are representative of lightly loaded conditions, and utilization ratios approaching 1.0 are representative of heavily loaded conditions. Thus, with these system parameters, initial simulations are run and their results are studied. Modifications of these base parameters are made to test how performance is affected.

A final simulation parameter that is chosen is how long the simulation has to run before the system is considered to be in steady state. A simulation was run for the SBH topology with the token passing protocol. The time needed to transmit a token is set to be equal to one-third the average message passing time. Link and node service rates are set to

$$\mu_L = 5.0 \qquad \mu_N = 10.0$$

The simulation is run for 400 time units. The mean, variance, and maximum delay are recorded at 50 time unit intervals. Also, the statistics on delay are cleared each time they are recorded meaning that the messages passed between time T = 0 and T = 50 are not considered in the statistics recorded for messages at time 100, etc. The results of this simulation are shown in Table 5.1. Initially, all of the queues in the network are empty. Thus, the first messages that travel through the network rarely need to be queued and will only encounter service time delays. As more messages enter the network, some of the messages will be held in queues until the server this message is waiting for finishes serving previous messages. Eventually, the queue lengths will reach a steady state value. At this time, all messages entering the system will encounter both service time delays and waiting time delays. By examining the average message times for a simulation, the steady state time can be recoginized as the time after which the end-to-end delay is no longer monotonically increasing. From these results shown in Table 5.1, it can be seen that the system appears to be in steady state for all time intervals recorded after time T = 50. Similar results are found for the other topologies and protocols used in this research. From these results, it is determined that all simulations should be run from time 0 to time 150. Also, the statistics are cleared at time T = 75, and thus the results gathered for each simulation will include all messages that are passed between time T = 75 and T = 150. In this time interval, the simulation model appears to be stable, and the results obtained are steady state values.

Table 5.1. **Determination of Steady State Run Time**

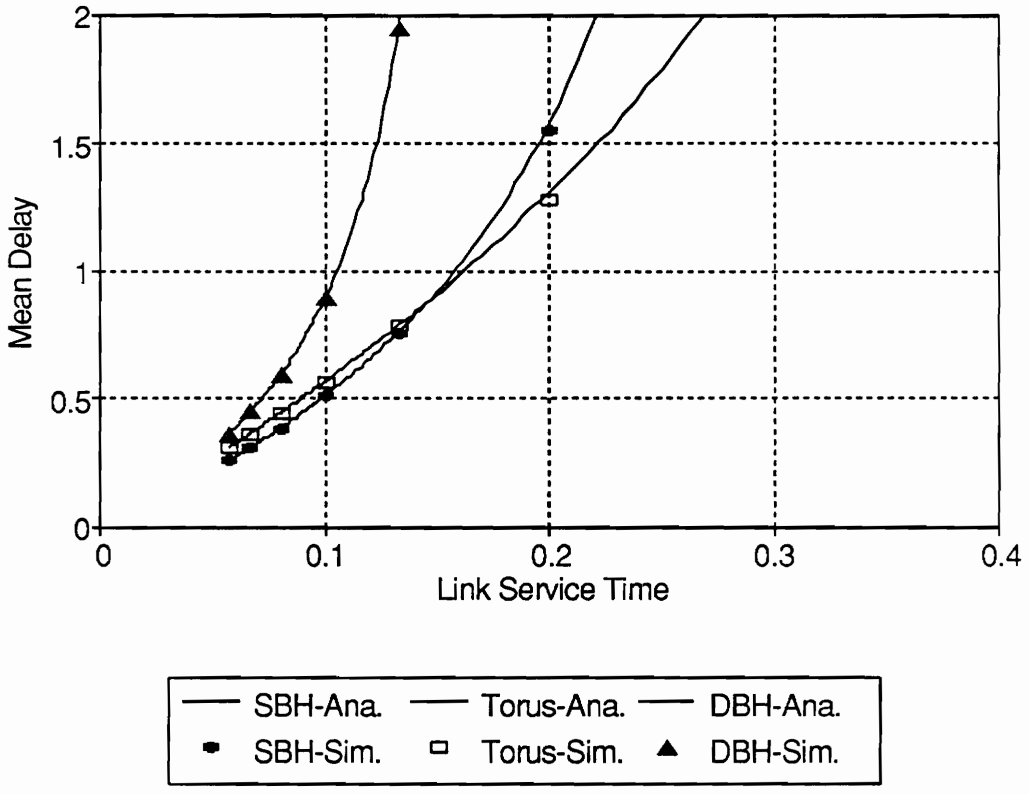
SPANNING BUS HYPERCUBE			
Time	μ_{EE}	σ_{EE}	Max
50.0	2.347	1.383	9.497
100.0	2.367	1.466	10.75
150.0	2.503	1.568	11.24
200.0	2.341	1.435	12.28
250.0	2.456	1.491	13.66
300.0	2.229	1.331	8.820
350.0	2.395	1.447	10.01
400.0	2.400	1.435	11.41

The first results discussed concern the FIFO protocol with the node service rate set to twice the link service rate. Figures 5.1a, 5.1b, and 5.1c are plots of the mean, standard deviation, and maximum delay, respectively, versus the average link service time. Results for the mean and standard deviation of delay are obtained both analytically and through simulation, and results for the maximum delay are obtained only through simulation. These plots show that the analytic results are in close agreement with the simulation results.

The torus outperforms the SBH for long link service times based on all three criteria. As the link service time decreases from .20 and .13, the SBH performance improves and overtakes the torus's performance in all three categories. As the link service time decreases beyond 0.08, the DBH begins to show a smaller maximum delay than does the torus, but the torus continues to hold an edge over the DBH in both the mean and standard deviation of delay.

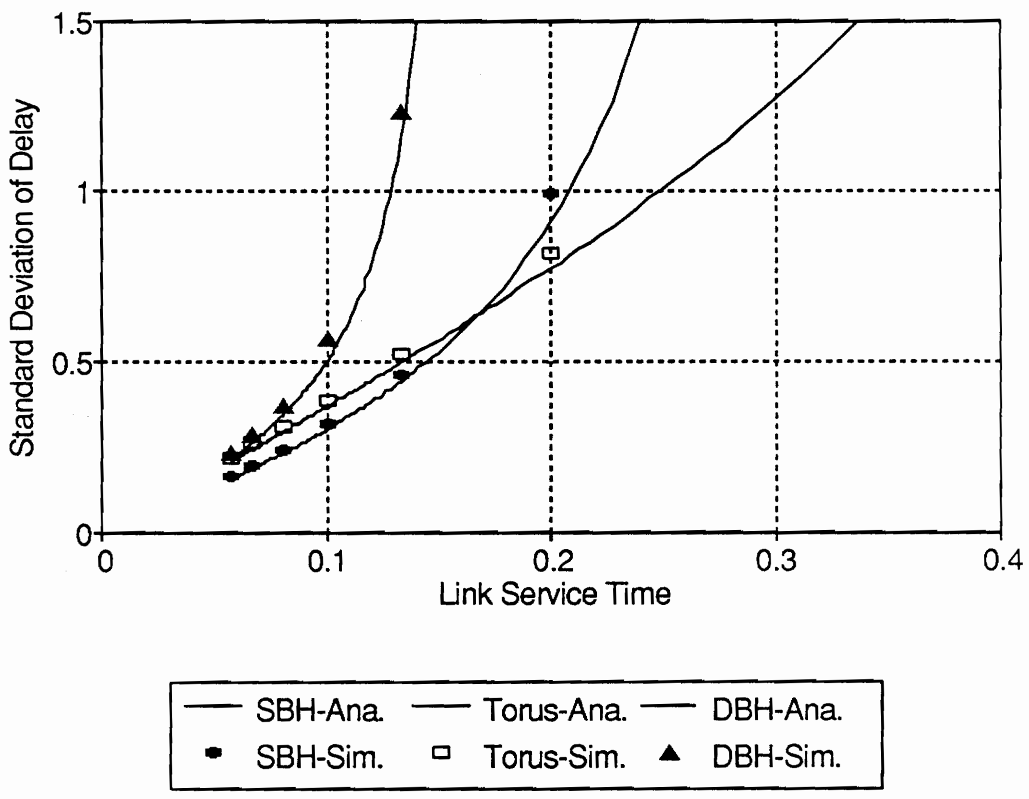
These results can be explained by examining the three topologies under consideration. The links in the torus are used by only two nodes, whereas links in the SBH and DBH are used by four nodes. Thus, when the link service time is large, the links of the SBH and DBH saturate faster than those in the torus. This factor decreases the performance of both bus based topologies.

As the link service time decreases, the congestion in the links decreases, and this advantage of the torus is not as significant. The number of hops the messages must take from source to destination becomes apparent. For a $W^D = 4^3$ network with uniform distribution of traffic, the average number of hops is 2.29 for the SBH, 2.86 for the DBH, and 3.05 for the torus. Thus, as the link service time decreases, the performance of the SBH overtakes the performance of the



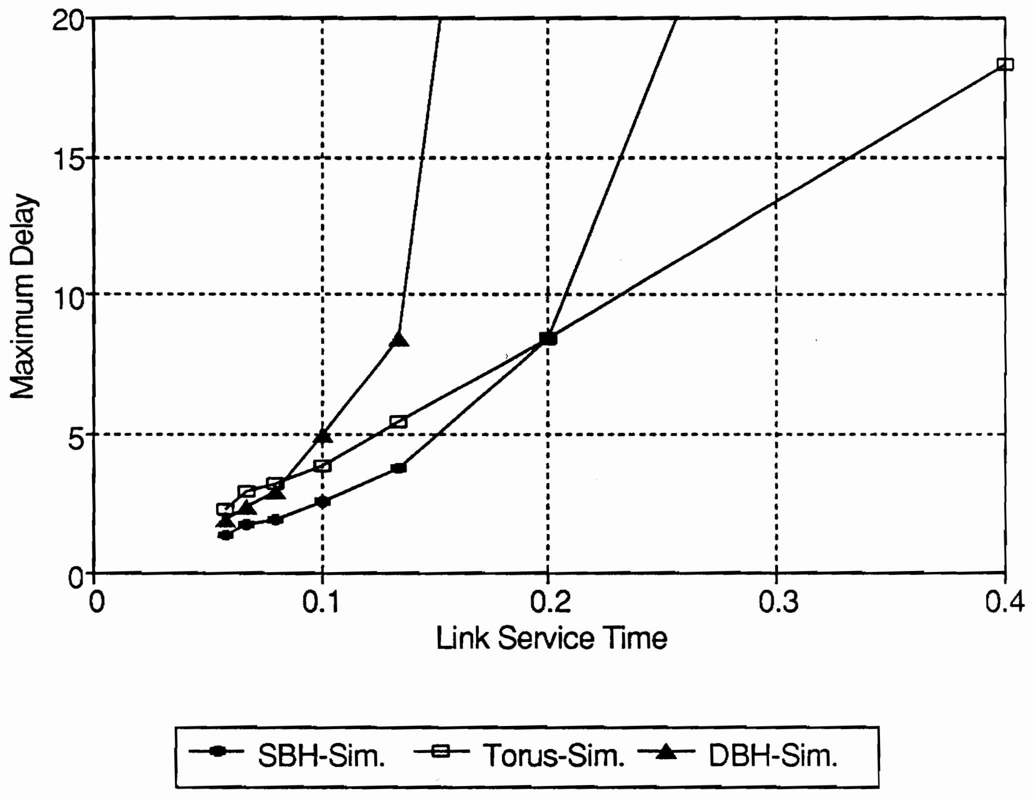
(a) Mean Delay vs. Link Service Rate

Figure 5.1. FIFO Delay: $\lambda_{NST} = 2 \times \lambda_{LST}$



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.1. FIFO Delay: $\lambda_{\mathrm{NST}} = 2 \times \lambda_{\mathrm{LST}}$ (continued)



(c) Maximum Delay vs. Link Service Rate

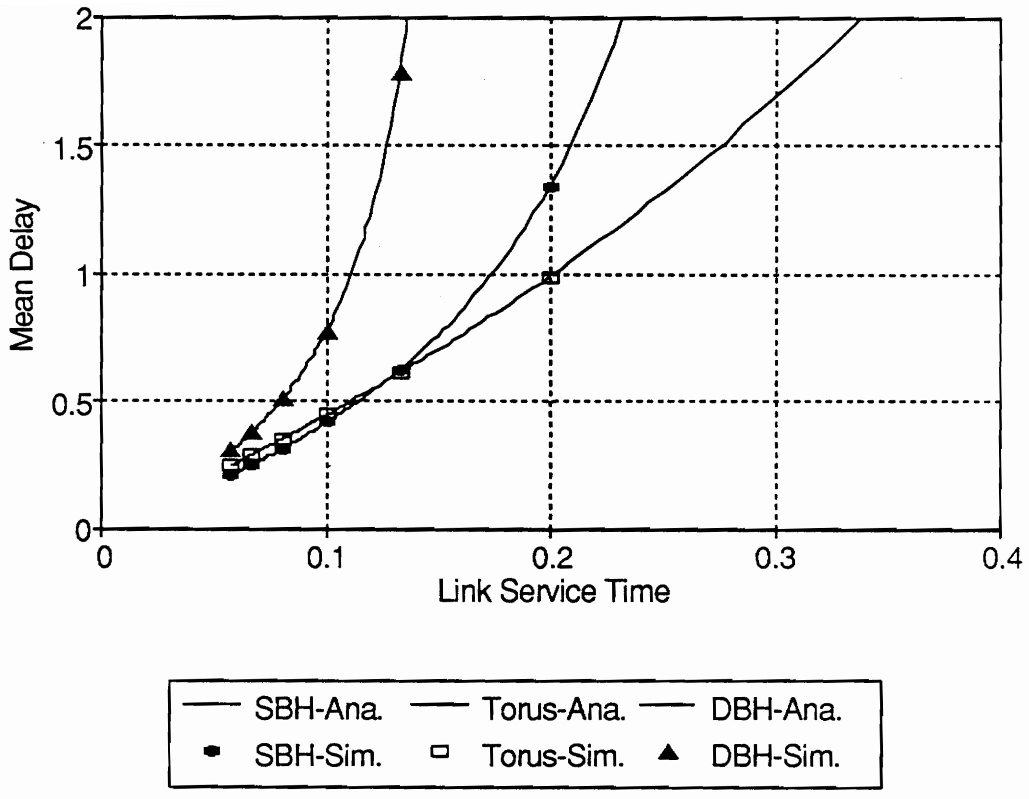
Figure 5.1. FIFO Delay: $\lambda_{NST} = 2 \times \lambda_{LST}$ (continued)

torus. As the service time decreases further, the effects of link congestion in the links of the DBH are alleviated, and its performance nearly approximates that of the torus.

It should be remembered that as the link service time changes, the node service time changes accordingly to maintain a 2-to-1 node-to-link rate ratio. Thus, in addition to link congestion, node congestion is also occurring in the system. Because each of the topologies contains a single node server for each of the 64 system nodes, the major factor affecting node service times is the number hops that each message must take. The more hops a message must take, the more node processing time must be spent on the message.

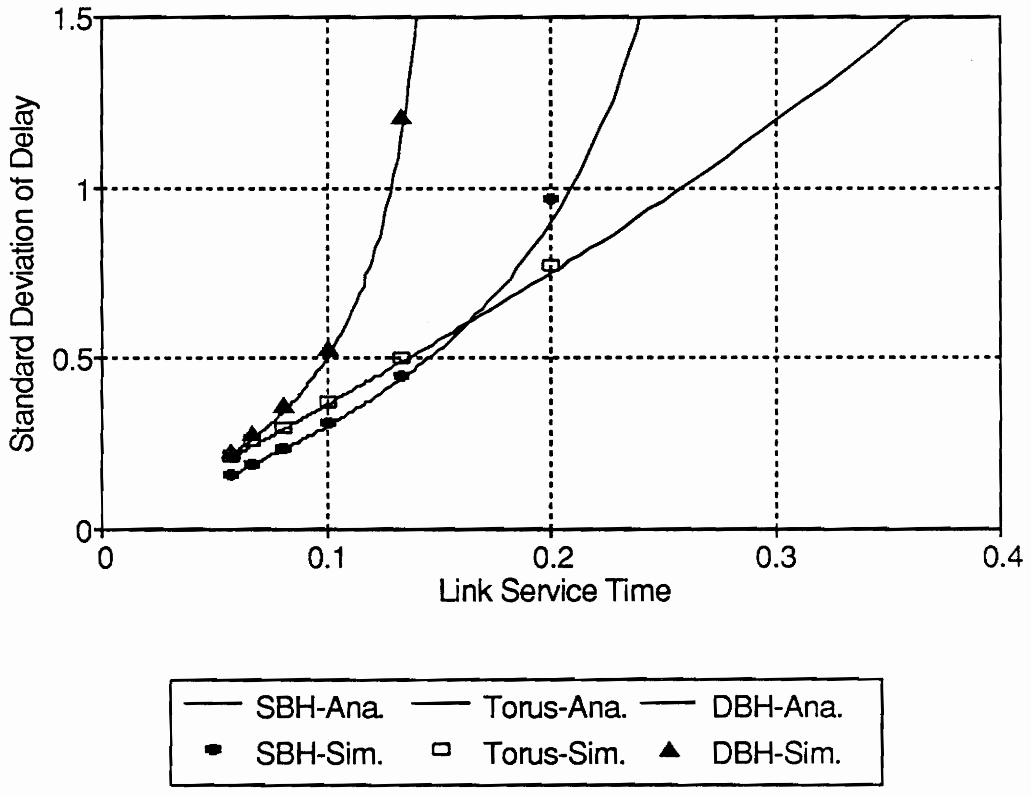
To see how large the effects of the node processing are in the above systems, the message delay in the above topologies is examined for a system with a node-to-link service rate ratio of 4-to-1. Plots for the mean, standard deviation, and maximum delay are contained in Figures 5.2a, 5.2b, and 5.2c, respectively. The plot of the maximum delay in Figure 5.2c shows very little change from the plot of Figure 5.1c. Similarly, the standard deviation of the delay for a 4-to-1 node-to-link service rate ratio is nearly equal to what it is for a 2-to-1 ratio. Thus, it can be concluded that for a 2-to-1 ratio, the standard deviation and the maximum delay are not affected by node congestion. It should be noted, however, that changing the size of the network could significantly affect where bottlenecks in the system occur.

The mean delay time for a 4-to-1 ratio is, however, seen to be different than for the 2-to-1 case. Whereas in the 2-to-1 case the mean delay of the torus and SBH topologies became equal at a link service time of approximately 0.15, in the 4-to-1 case this equality is achieved at a value of approximately 0.12. Thus, raising the node service rate widens the range of link service times over which the torus has a smaller mean delay than the SBH. Of particular note is that the link service time value at which the mean message delay of the two topologies becomes equal



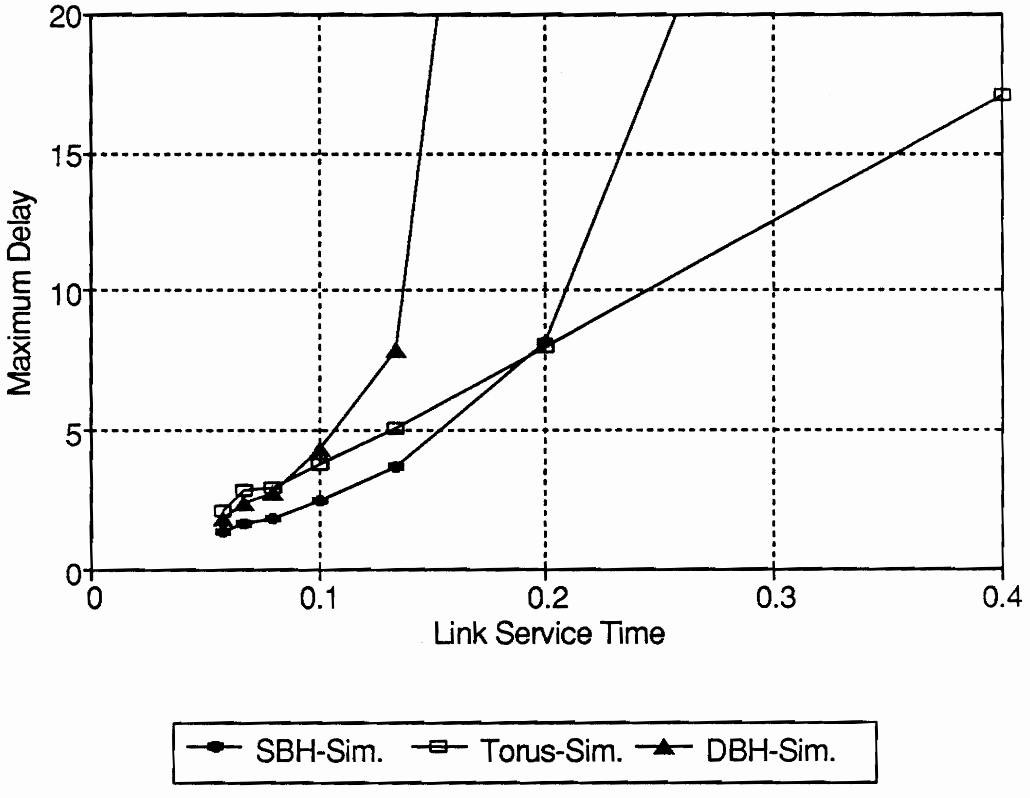
(a) Mean Delay vs. Link Service Rate

Figure 5.2. FIFO Delay: $\lambda_{NST} = 4 \times \lambda_{LST}$



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.2. FIFO Delay: $\lambda_{NST} = 4 \times \lambda_{LST}$ (continued)



(c) Maximum Delay vs. Link Service Rate

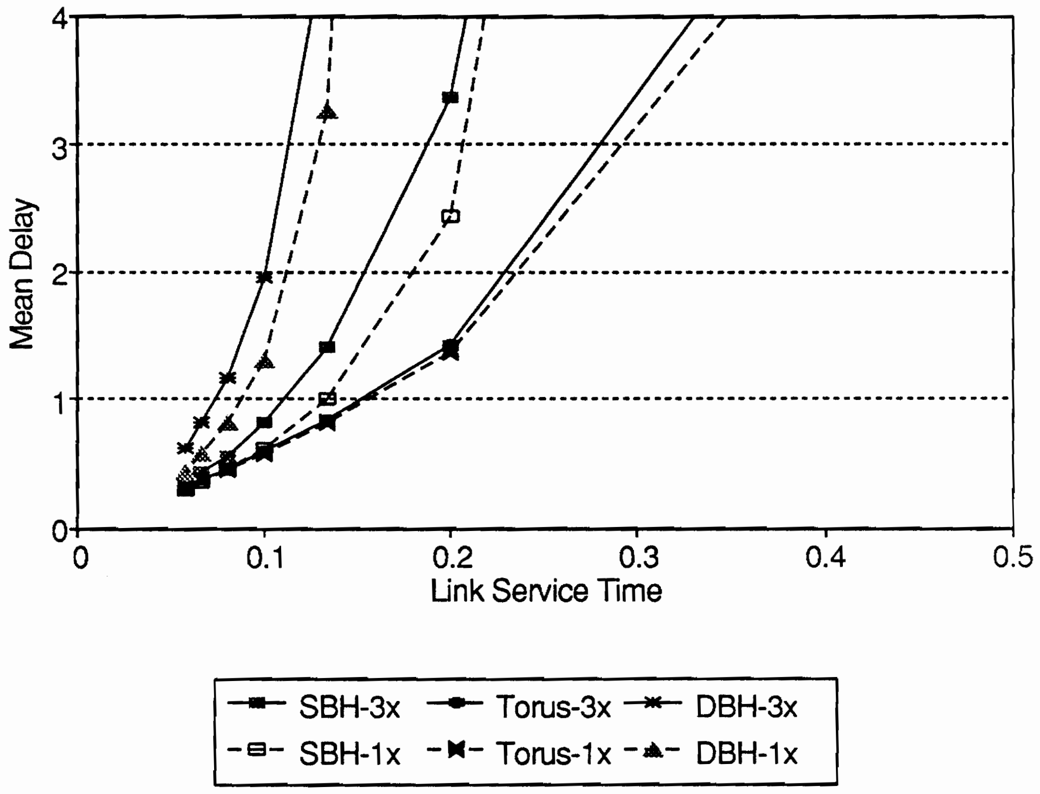
Figure 5.2. FIFO Delay : $\lambda_{NST} = 4 \times \lambda_{LST}$ (continued)

changed despite the fact that the link service time value at which the standard deviations of the message delay became equal remained unchanged. This shows that one factor that limits the performance of the torus in the 2-to-1 ratio case was the additional amount of node processing that the torus had to perform because messages in the torus require a larger number of hops. It also shows that measuring the performance of a topology solely by the mean message delay does not necessarily account for second order effects that may be occurring in the system.

5.3. TDM Results

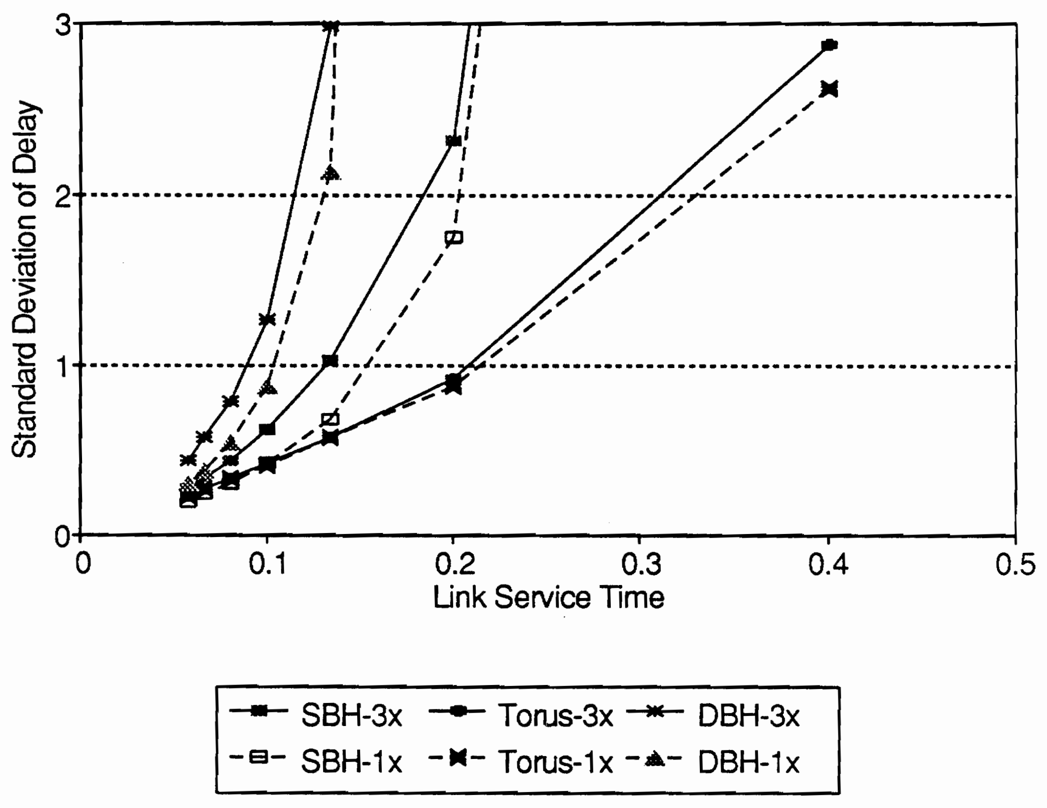
System performance is known to be affected not only by the system topology but also by the protocol used in the communication structure. This section presents and discusses results for the TDM protocol. Because analytic models were not obtained for this protocol, all of the results in this section are obtained through simulation. The same 64 node topologies are used for the TDM protocol as were used for the FIFO protocol. The ratio of the node service rate to the link service rate was held constant at 2-to-1 for all of the TDM simulations. As was discussed earlier, an additional parameter used in the TDM protocol that is not present in the FIFO simulations is the length of the TDM transmission frame. For one set of simulations, this time frame was set to three times the average transmission time of a message on a link, and for a second set of simulations it was set equal to one times the average message transmission time. The results of these simulations for both the three times and one times clock periods are shown in Figures 5.3a, 5.3b, and 5.3c for the mean, standard deviation, and maximum delay, respectively.

These plots show that for all simulations except for heavily loaded torus networks, the one times clock period outperforms the three times clock period in all three performance categories. This



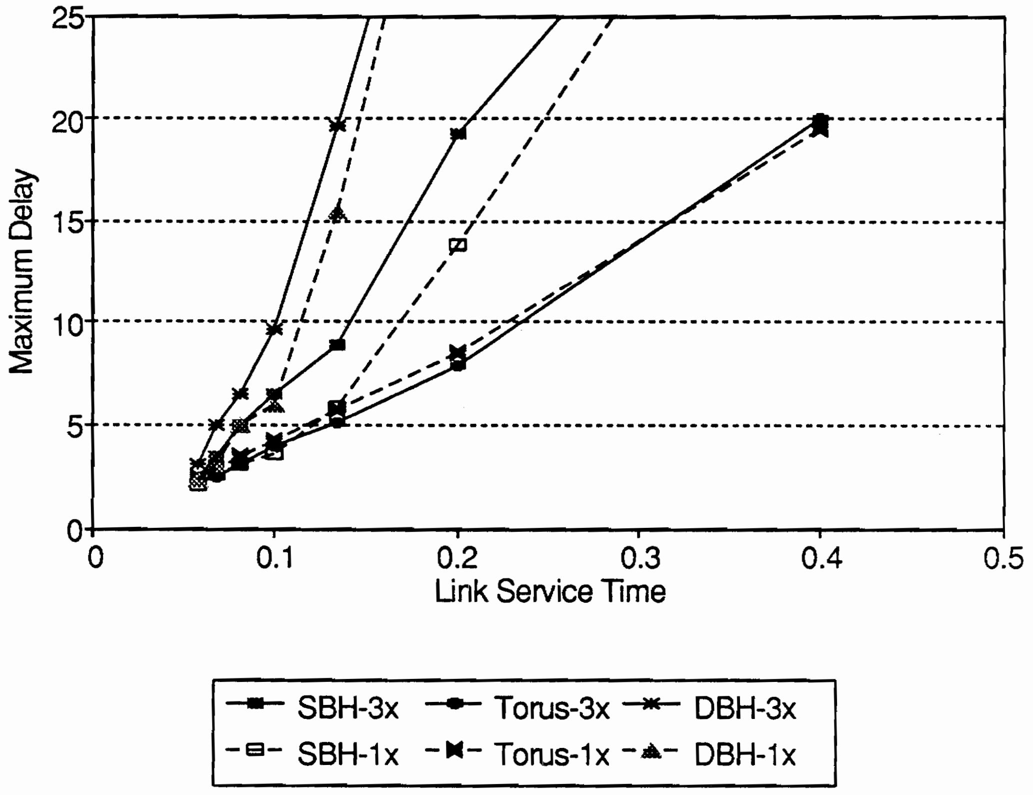
(a) Mean Delay vs. Link Service Rate

Figure 5.3. TDM Delay



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.3. TDM Delay (continued)



(c) Maximum Delay vs. Link Service Rate

Figure 5.3. TDM Delay (continued)

shows that the time wasted in cases where a node sits idle with no messages to transmit during a clock cycle where it has permission to transmit degrades the system's performance. It also is noted that this effect of wasted time is more pronounced in the bus topologies than in the torus. This is because in the torus there are only two nodes attached to each link, whereas there are four nodes in the two bus based topologies. Thus, if one of the nodes is waiting to transmit a message in the torus topology, it has to wait through only one time slot. In the bus based topologies, however, a node has to wait for up to three time slots before it gets a chance to transmit. Thus, if there is, on average, time wasted during each transmission cycle, the bus topologies will be affected by this wasted time for a larger number of clock periods than the torus topology. This will cause the performance of the torus topology to be less affected by this wasted time than the other two topologies.

By comparing the plots of the three topologies of the one times clock case, several interesting results occur. In comparing both the mean and standard deviation of delay, the torus and SBH topologies both outperform the DBH topology by a wide margin. Based on the maximum message delay criteria, the torus and SBH outperform the DBH for long service times, and become approximately equivalent for short link service times. Thus, overall, for the TDM protocol, the torus and SBH display a significant performance advantage over the DBH.

In comparing the SBH to the torus using a TDM protocol, the torus is seen to outperform the SBH for long link service times, while the performance of both topologies is approximately equal at moderate to short service times. This is different from the FIFO topology where the SBH held a performance advantage over the torus for short link service times. Thus, the inefficiency of the TDM protocol for bus based topologies as compared to point-to-point link topologies has erased the performance advantage that the SBH held over the torus for short link service times using the FIFO protocol.

A final set of results obtained for the TDM topology are obtained to determine the clock period that should be used to optimize the performance of the TDM protocol. The length of the TDM clock period was varied from between 0.1 times the average transmission time to 3 times the average transmission time. A SBH topology was used with

$$\lambda_N = 1.0, \quad \mu_L = 5.0, \text{ and } \quad \mu_N = 10.0 .$$

The results obtained from these simulations are shown in Table 5.2. The performance of the network based on all three criteria is seen to steadily improve as the clock period is decreased. As the clock period is decreased, less time will be wasted by idle nodes that are given permission to transmit on the link. These results show that the effect of this wasted time outweighs any regularity added to the network by assigning time slots. These results are somewhat misleading however, because they do not account for the fact that a certain guard time would be needed between adjacent time slots so that if the synchronization of two nodes is slightly misaligned, two messages will not be simultaneously transmitted if one node attempts to transmit at the very end of its clock period while the next node on the link begins transmitting at the very beginning of its clock period.

In an attempt to improve the performance of the network, some type of reservation system would be needed to prevent idle nodes from wasting the link time. Basic examples of such methods, such as the reservation ALOHA system, were discussed in Chapter 3. The simulations presented here, however, do indicate how a TDM scheme can affect the mean and standard deviation of delay for the topologies studied in this research.

Table 5.2. **Effects of TDM Clock Period**

SPANNING BUS HYPERCUBE			
T_{CLK}	μ_{EE}	σ_{EE}	Max
3.00	3.374	2.307	19.19
1.00	2.447	1.753	13.81
0.80	2.300	1.626	13.28
0.50	2.075	1.417	11.95
0.30	1.877	1.258	9.511
0.10	1.697	1.116	8.595

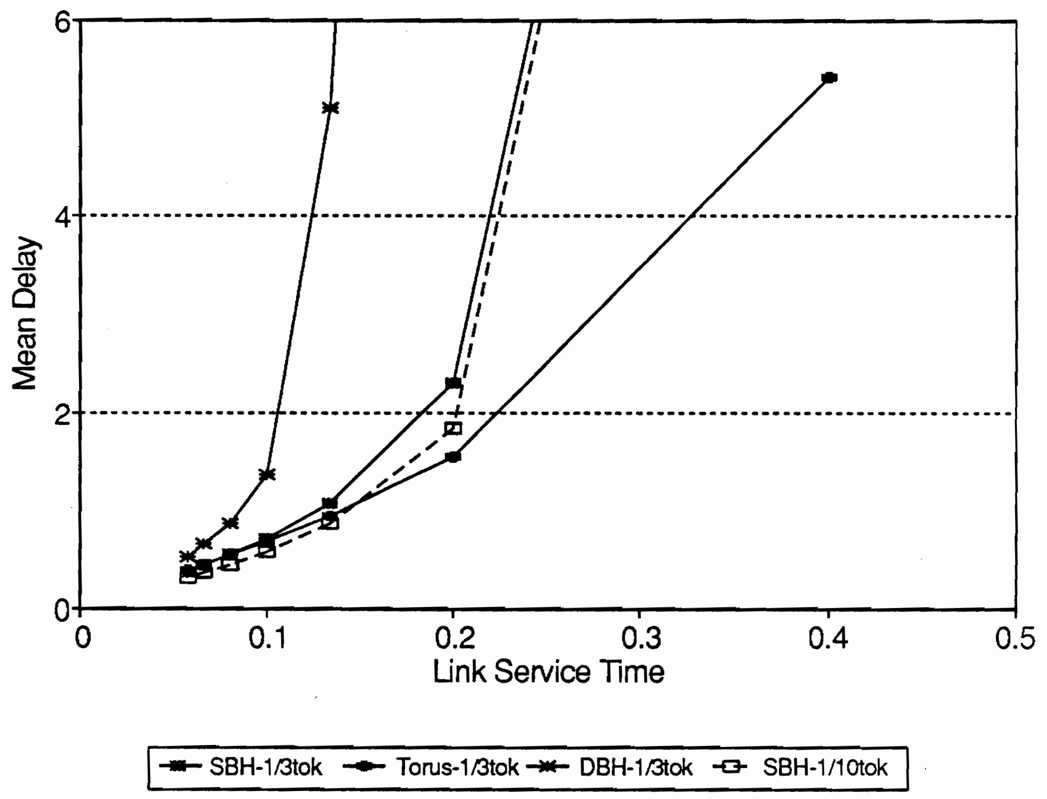
5.4. Token Passing Protocol Results

The final protocol studied in this research is token passing. The time required to pass the token between nodes can affect the performance of a token passing system. In the simulations performed in this research, the token passing time is set equal to one-third of the average time required to pass a message on the link for all three topologies. As an example of how this token time affects the performance of the system, an additional set of simulations is performed for the SBH topology with a token time of one-tenth the average message passing time. All of the other network parameters such as the size of the network and the ratio of node-to-link service rates are held at the same values that were used for the simulations of the TDM protocols. The results of these simulations for the mean, standard deviation, and maximum delays are shown in Figures 5.4a, 5.4b, and 5.4c.

One characteristic that can be quickly seen is that changing the token passing time in the SBH topology has only a small effect on the system performance for all three categories. Thus, the simulations using the token time equal to one-third the message passing time are assumed to be a good approximation of the performance of each topology for all reasonable token passing times. It should be noted that the smaller the token passing time becomes, the more efficient the total communication will be.

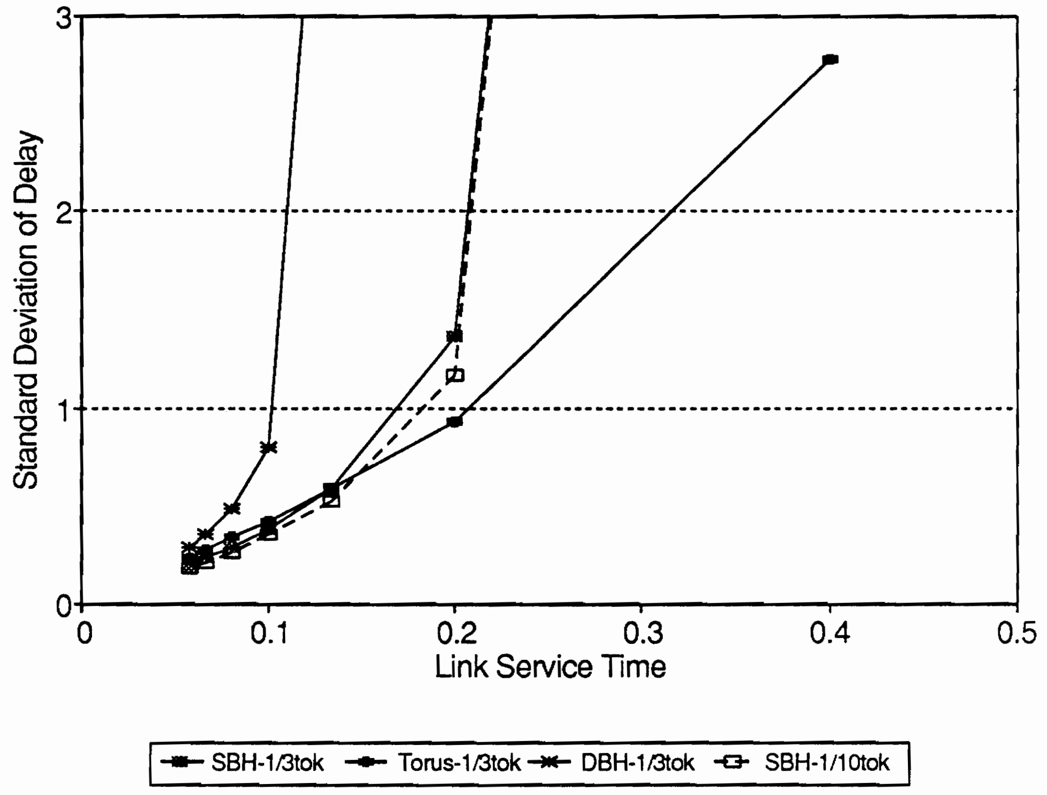
The SBH and torus topology outperform the DBH by a wide margin for long link service times, and also hold a small advantage over the DBH for short link service times. This performance advantage exists for all three of the performance measures.

The performance comparison between the SBH and the torus is somewhat more interesting. The torus outperforms the SBH for long link service times, and the SBH holds a slight advantage



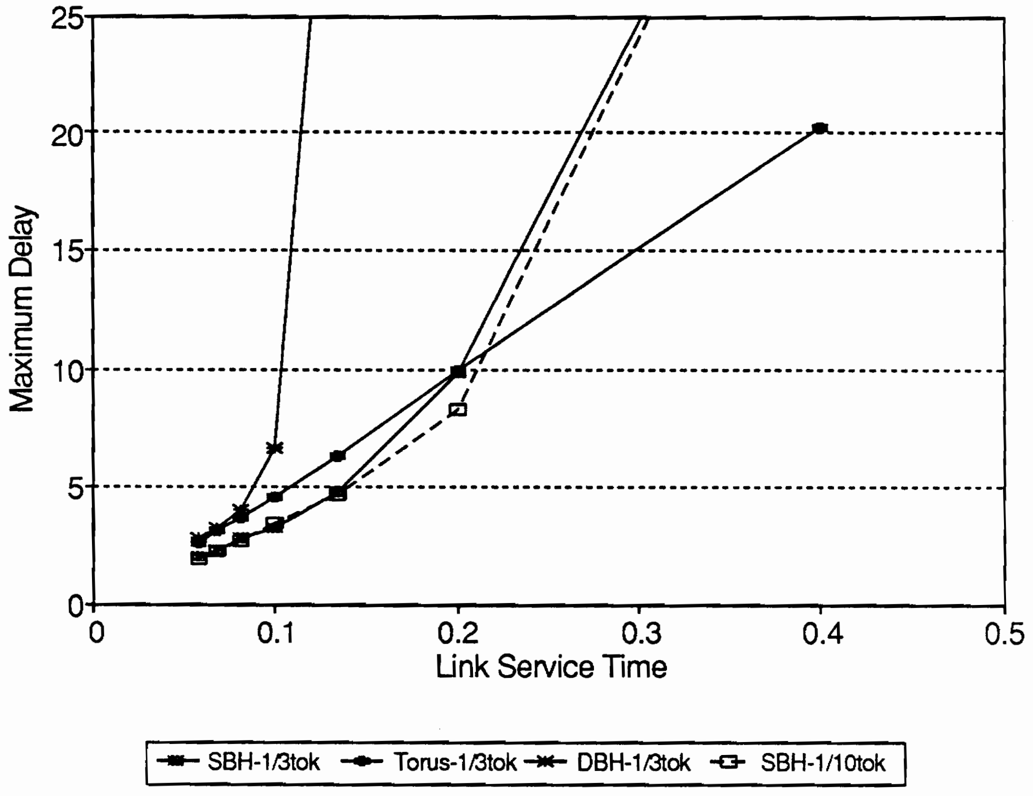
(a) Mean Delay vs. Link Service Rate

Figure 5.4. Token Passing Delay



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.4. Token Passing Delay (continued)



(c) Maximum of Delay vs. Link Service Rate

Figure 5.4. Token Passing Delay (continued)

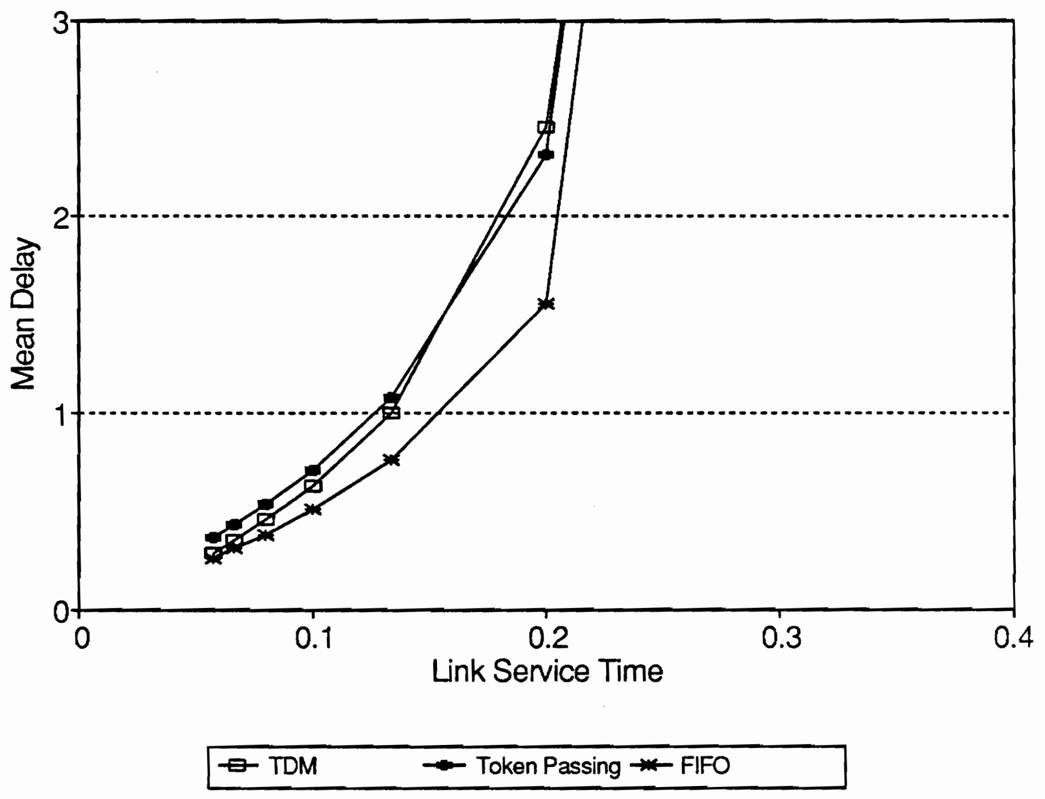
over the torus for short link service times. The SBH shows the largest advantage over the torus in the maximum delay time comparisons. For the mean delay comparisons, the advantage of the SBH over the tours is minimal.

5.5. Comparison of Protocols

From the studies of each individual protocol, it is seen that the DBH is only effective when short link service times are used. Because a real-time distributed system should be able to function properly under a wide variety of system conditions, the torus and the SBH appear to be appropriate topologies for real-time distributed memory computing systems. The previous discussion of protocols compared the performance of a given protocol when used with different topologies. The plots examined in this section study the effects of different protocols on a fixed topology. Figures 5.5a, 5.5b, and 5.5c plot the mean, standard deviation, and maximum delay versus the average link service time for the three protocols studied in this research for a SBH topology. Figures 5.6a, 5.6b, and 5.6c show these same quantities for a torus topology. Other system parameters such as the message generation rate, the size of the network, and the node-to-link service rate ratio are set to the same values used in the previous simulation runs,

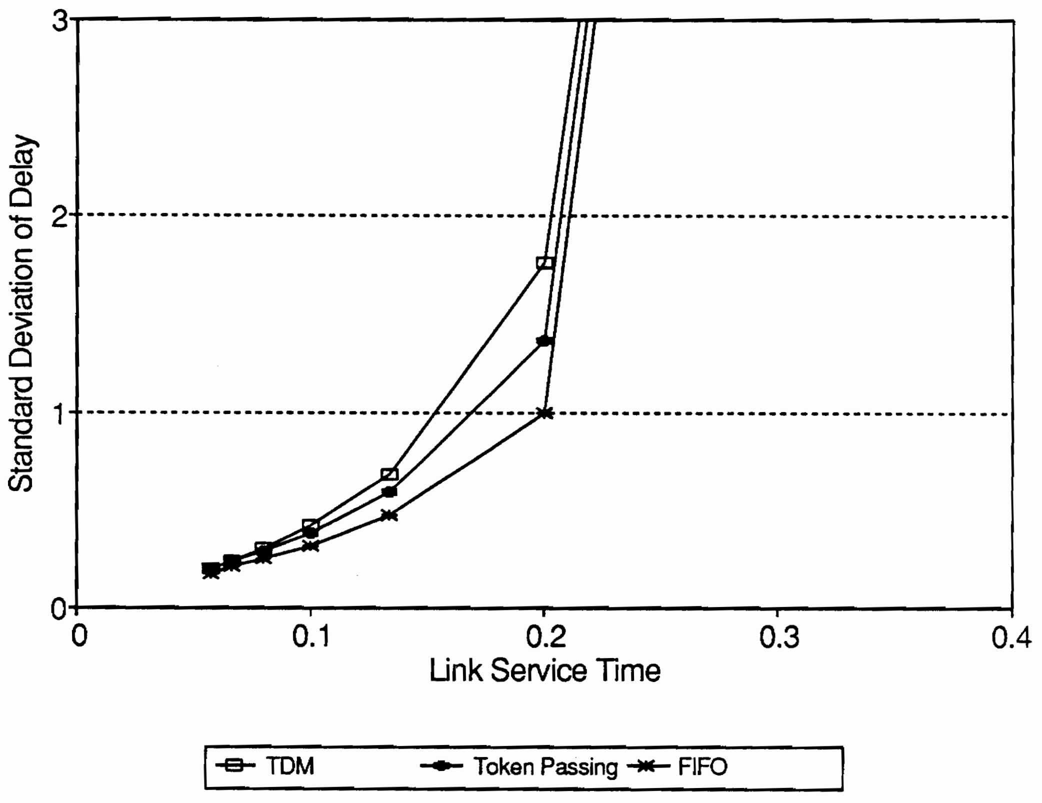
$$\lambda_N = 1.0, \quad W^D = 4^3, \text{ and } \frac{\mu_N}{\mu_L} = 2.0 \ .$$

For the TDM protocol, the simulations are performed using the clock period equal to one times the average transmission time. For the token passing system, a token time equal to one-third the average message passing time is used.



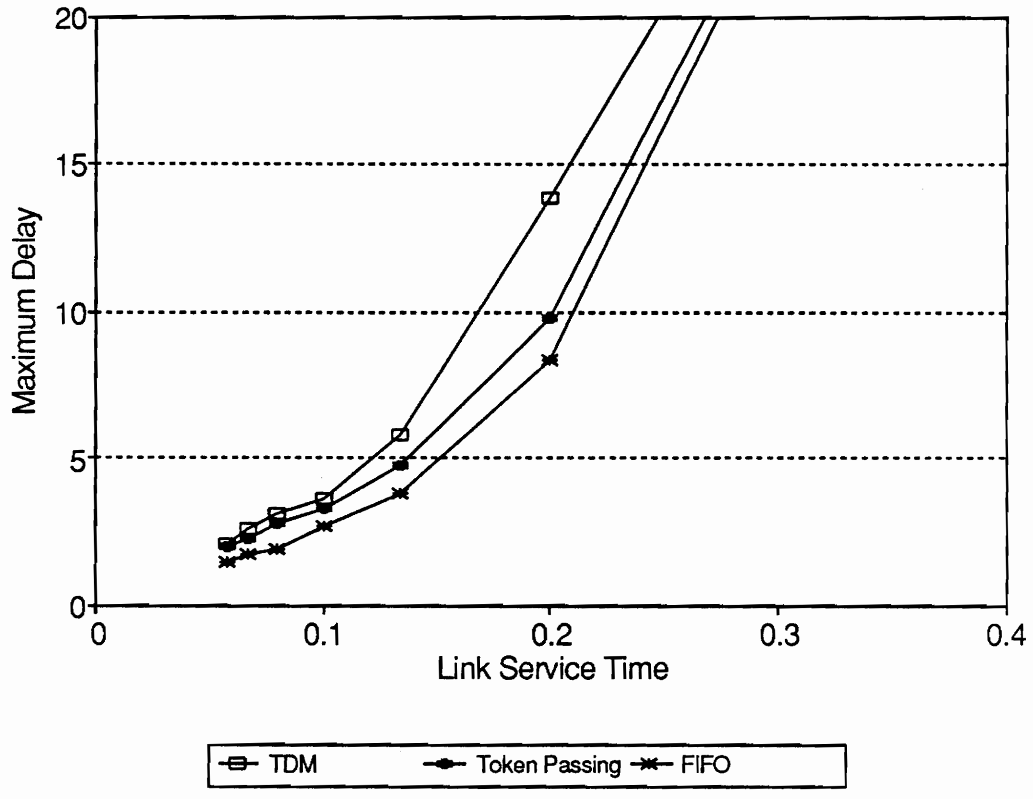
(a) Mean Delay vs. Link Service Rate

Figure 5.5. Comparison of Protocols for the Spanning Bus Hypercube



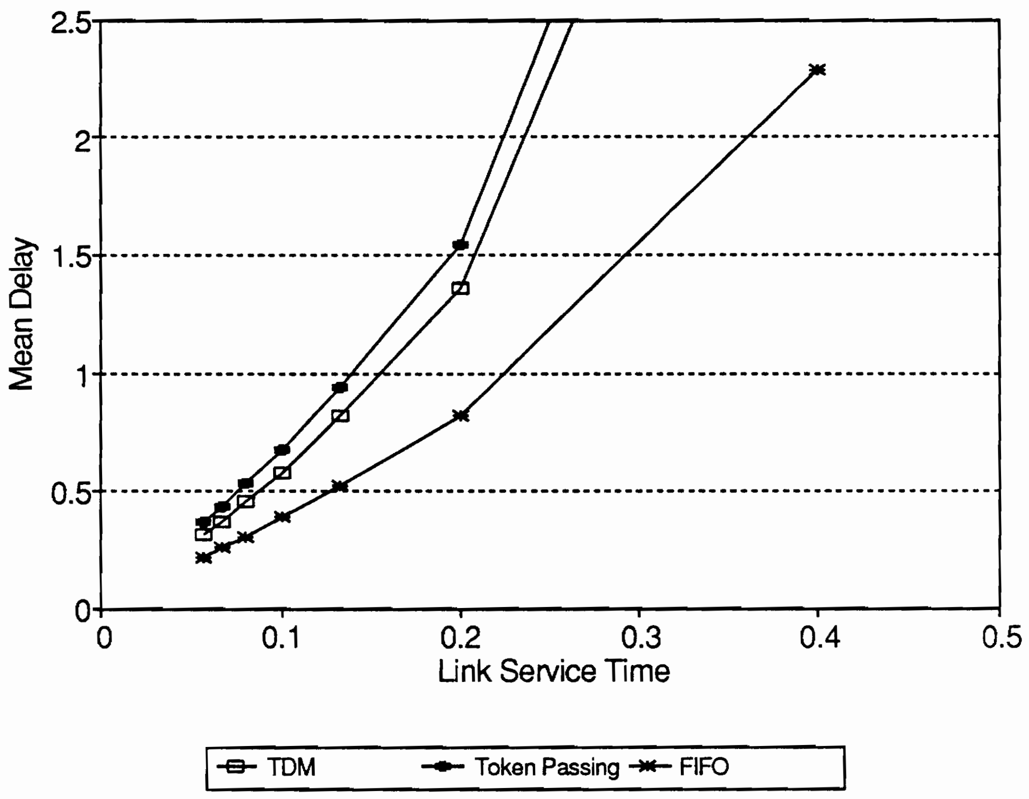
(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.5. Comparison of Protocols for the Spanning Bus Hypercube (continued)



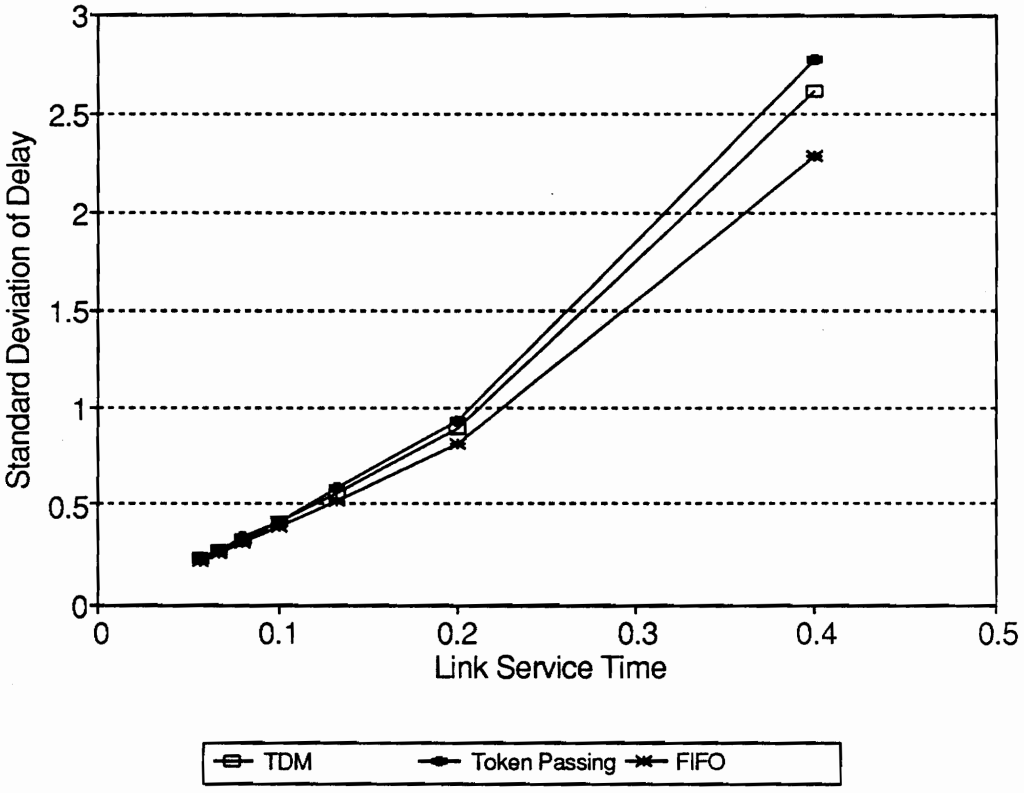
(c) Maximum Delay vs. Link Service Rate

Figure 5.5. Comparison of Protocols for the Spanning Bus Hypercube (continued)



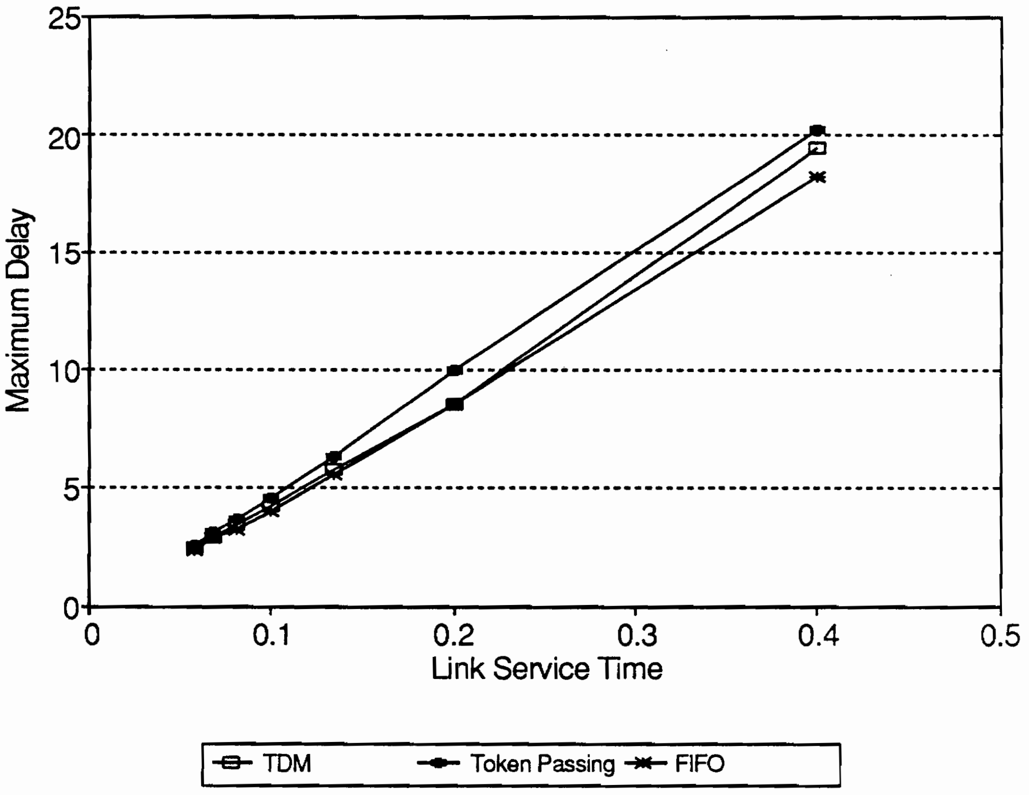
(a) Mean Delay vs. Link Service Rate

Figure 5.6. Comparison of Protocols for the Torus



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.6. Comparison of Protocols for the Torus (continued)



(c) Maximum Delay vs. Link Service Rate

Figure 5.6. Comparison of Protocols for the Torus (continued)

For the SBH topology, the FIFO protocol has the best performance characteristics for all three performance criteria and at all of the link service times studied. The mean delay of the FIFO protocol for long link service times has the most noticeable advantage over the two other link access protocols. The token passing protocol has a small performance advantage over the TDM protocol for long link service times as measured by all three parameters. As the link service time decreases, however, the TDM protocol performance improves in relation to the performance of the token passing protocol. For short link service times, the performance of the token passing and TDM protocol are almost equal for standard deviation and maximum delay, and TDM holds an advantage over token passing as measured by the mean message delay.

Results for the torus topology, which can be seen in Figures 5.6a, 5.6b, and 5.6c, indicate that the FIFO protocol performs better than either the TDM or token passing protocol. In addition, the TDM protocol outperforms token passing over the entire range of link service times studied. What is interesting in these comparisons, however, is that the standard deviation of delay and the maximum delay for the torus topology become essentially equal for all three protocols as the link service time is lowered below 0.1. This shows that the torus topology is less affected than the bus based topologies when it is subjected to the inefficiencies of the TDM and token protocols used in this research. This is true because only two nodes in the system are contending for use of any given link in the torus topology, so each node has to wait a shorter period of time before it is given access to the link.

From the results obtained for all three topologies it is seen that the FIFO protocol seems to outperform the TDM and token passing protocols. What must be remembered, however, is that the FIFO protocol used in this research is an idealized version that does not account for any of the communication overhead needed to transmit the status of the system between nodes attached to the link. The token passing and TDM schemes, however, do include some of the inefficiencies these protocols place on the system. Also, the performance of the FIFO based protocol is only marginally better than the other two protocols for all but very long link service times. Thus, for the cases with short to moderate link service times, the small advantage of the FIFO protocol may be caused solely by the fact that such an idealized version of this protocol is used. For this reason, it is concluded that a proper protocol for a real time system may need to employ more advanced priority schemes to limit the mean and variance of message delay.

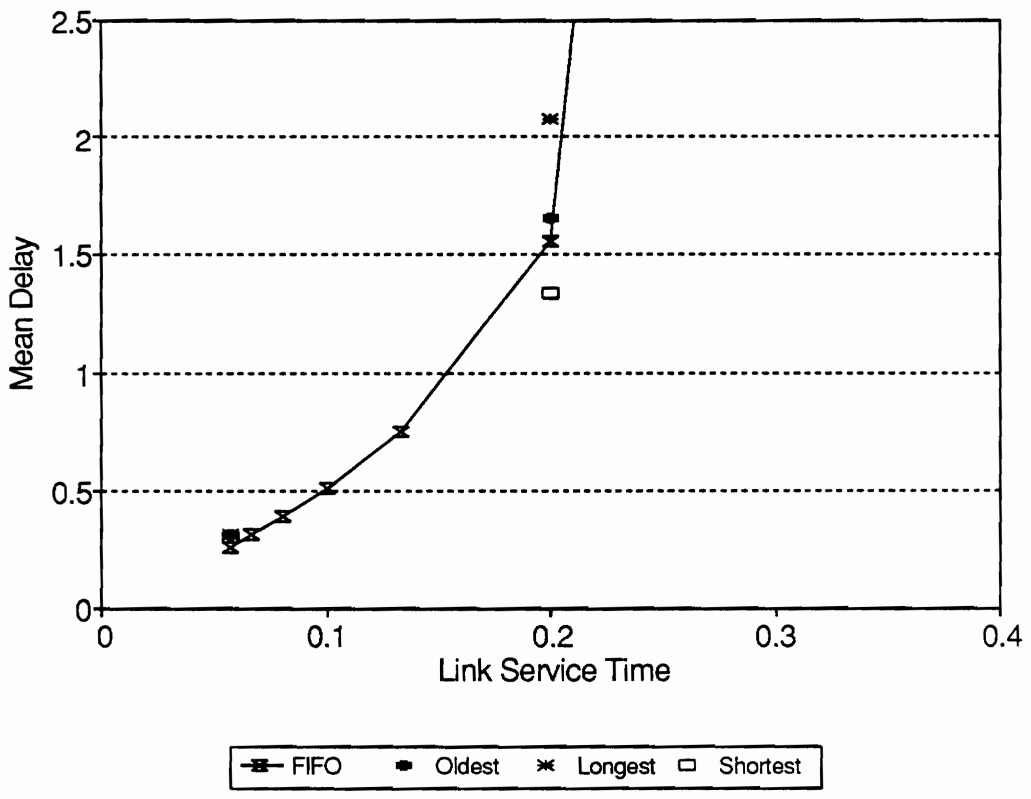
5.6. Additional Effects

Several different schemes can be used to control the mean and standard deviation of delay in a communication network. Two areas of concern in this research include how changes in the priority structure for queues in the system affect the message delay and which message parameters cause the most message delay. For both of these studies, a SBH topology using a FIFO protocol is used.

5.6.1. Queueing Priority Effects

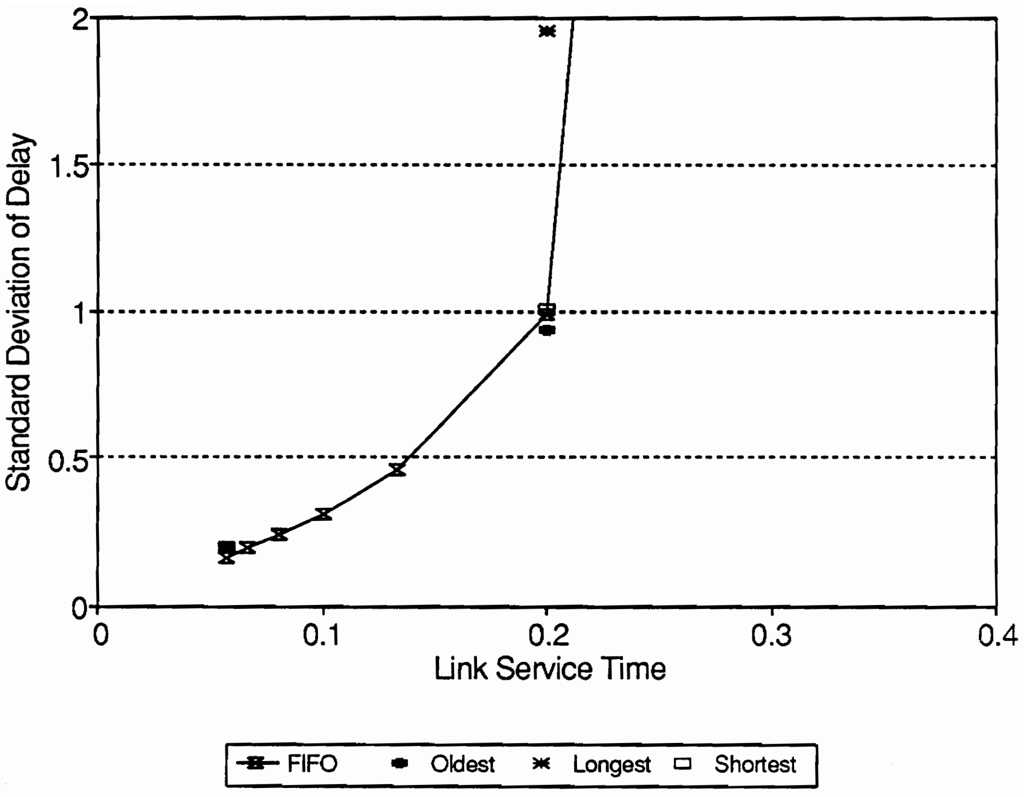
Three variations on the FIFO queueing priority are studied in this research. These queueing priorities are used for the queues in both the links and the nodes of the FIFO protocol model. The queueing priority used for the results reported thus far is the FIFO priority. The three additional queueing priorities employed include transmitting the oldest message in the queue first, transmitting the longest message in the queue first, and transmitting the shortest message in the queue first. Determination of the oldest message in the network is based upon the time at which the message was initially generated by its source node. The length of the message is determined by the amount of time required to transmit the message on a link. Simulations are performed for the three newly introduced priority-based protocols for link service times of 0.057 and 0.2. The results shown for these simulations include a plot of the results for the standard FIFO protocol. In this way, the effect of each of the new priority schemes is examined for both long and short service times.

The results of these simulations for the mean message delay are shown in Figure 5.7a. For the mean message delay at long service times, transmitting the longest message in the queue first has the worst performance. This is because short messages in a queue containing a long message are delayed by the transmission of this long message on the link. Transmitting the oldest message in the queue first leads to a mean message delay slightly higher than the delay using the FIFO priority. This is because the length of a message affects how long the message will remain in the system. Long messages will tend to take longer to go from source to destination, and thus transmitting the oldest message in the queue first tends to cause longer messages to be transmitted ahead of shorter messages and the mean message delay of the system to increase slightly from its value in the FIFO priority. Finally, transmitting the shortest messages first creates a mean message delay smaller than is seen for the FIFO priority. This is because the short messages are quickly transmitted from their source to destination, and because it does not take long to transmit them, they do not add a significant amount of delay to the longer length messages transmitted after them. As the link service time of the system is decreased, however, all three of the priority schemes cause the message delay to be longer than in the FIFO case. The mean delay, though, is almost the same regardless of the priority structure. This is because for a short link service time, very few messages need to be queued in the network, and thus the priority arbitration in the queue is rarely needed.



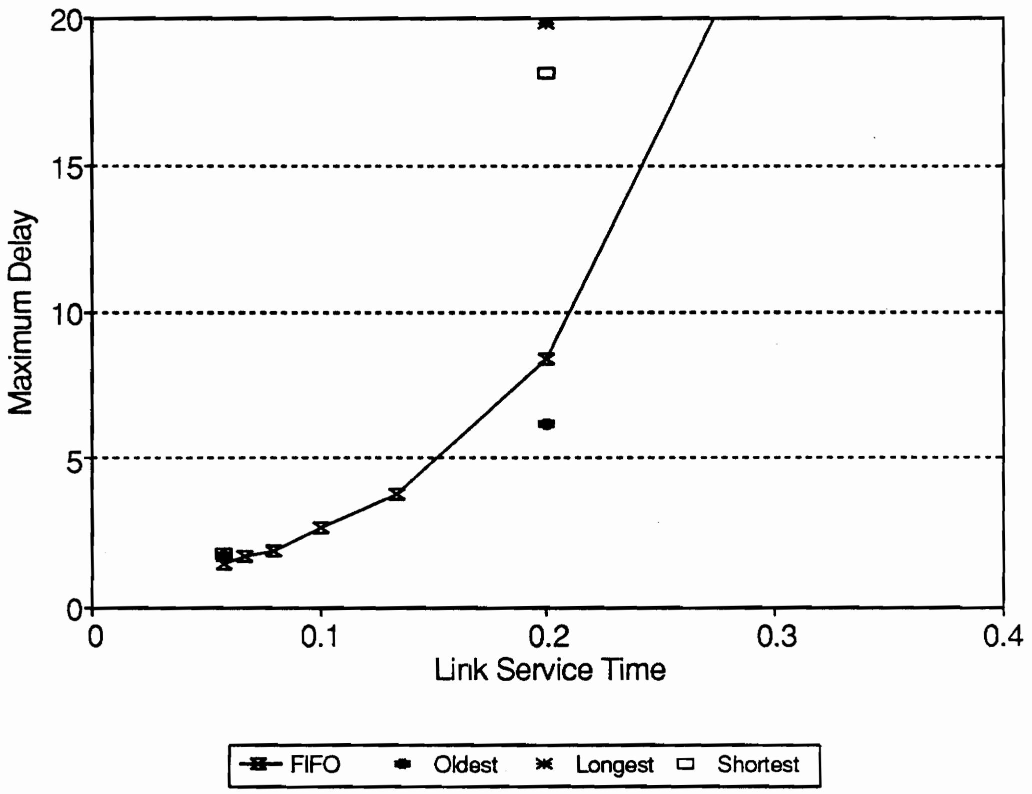
(a) Mean Delay vs. Link Service Rate

Figure 5.7. Queueing Priority Effects



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.7. Queueing Priority Effects (continued)



(c) Maximum Delay vs. Link Service Rate

Figure 5.7. Queueing Priority Effects (continued)

Figures 5.7b and 5.7c show the results of the queueing priorities on the standard deviation of the message delay and the maximum message delay. For long link service times, the oldest message first priority leads to both the lowest standard deviation of delay and the lowest maximum delay. This is because this priority scheme grants high priority to all messages that have been in the system for a long period of time, and thus the messages tend to remain in the network longer, but tend to arrive at their destinations at a more regular interval. The shortest message first priority has the second lowest standard deviation and maximum delay at long link service times. This was caused by the fact that the mean message delay for this priority scheme was very small. Thus, even though the message arrival times were scattered over a wide range, the average of these arrival times was small enough that the calculation of the standard deviation also was small. The worst priority scheme with respect to both standard deviation of delay and maximum delay is the longest message first priority. This priority acts the same way as the shortest message first priority in that it scatters the typical times at which messages will arrive at their destinations, but the mean time at which these messages arrive is larger than the FIFO priority. Thus the standard deviation increases because the mean time is high and the typical arrival times are widely scattered. For short link service times, the value of both the standard deviation of delay and the maximum delay is not affected very much by any of the priority schemes. None of these priority schemes, though, are able to perform as well as the FIFO priority. The reason for there being similar results for all of the priorities is that there is very little priority arbitration needed for networks with short service times since the queues in the network are usually empty.

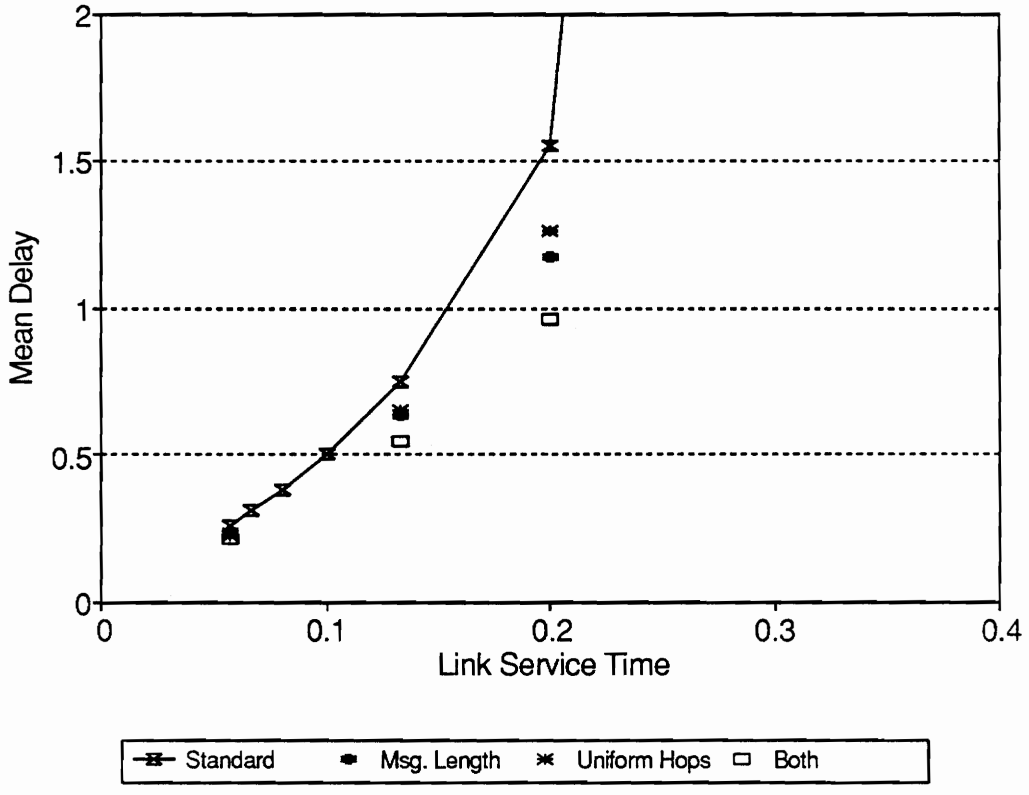
Thus, from the results of the simulations it can be seen that the FIFO priority scheme appears to have the most desirable overall characteristics. The other priority schemes had advantages over the FIFO scheme for certain conditions, but each of these priority schemes also performed poorly on at least one of the three measures of network performance.

5.6.2. Message Parameter Effects

The final area of study in this research concerns examining how message parameters of the original system model contributed to the system delay. Two parameters that are believed to be major contributing factors to the mean and standard deviation of delay are the exponentially distributed message lengths and the varying number of hops that messages must take in the network. To quantitatively study the effects of each of these parameters, the network model is modified. First, simulations are performed with every message needing to take two hops from its source to its destination. Next, simulations are performed where source-destination pairs are again randomly chosen, but the message lengths are set to a constant instead of being exponentially distributed. Finally, simulations are performed with both constant message lengths and the number of hops between source and destination fixed at two.

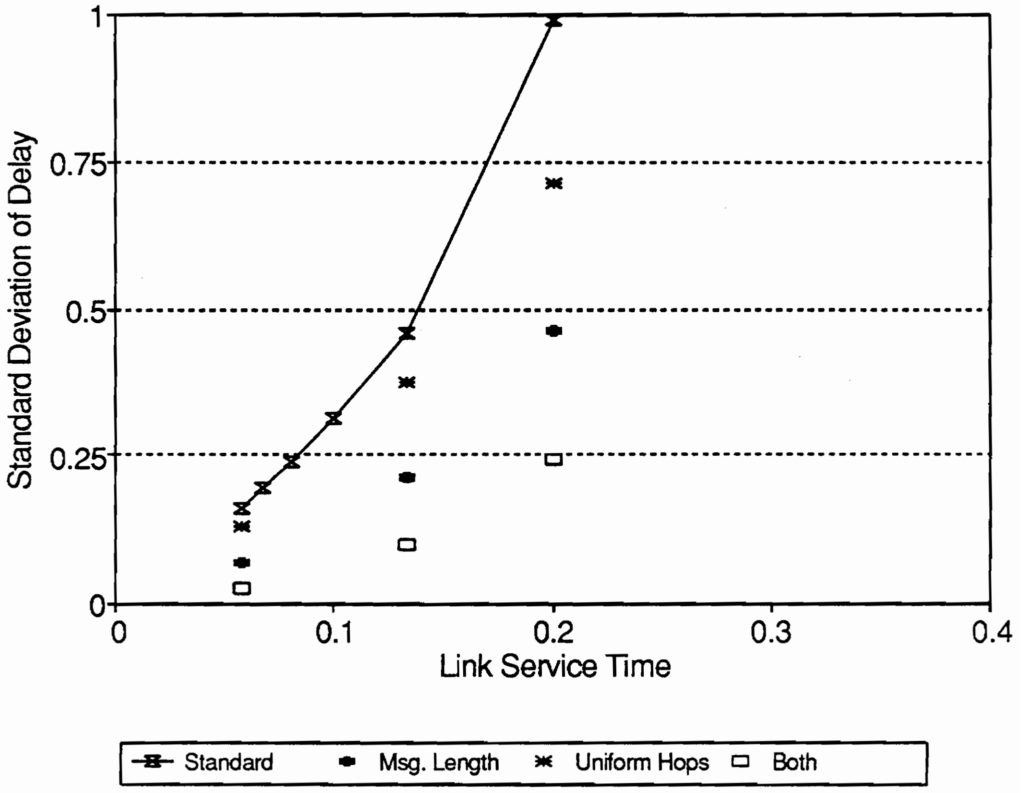
One parameter of these simulations that must be discussed is the distribution of source-destination pairs. Under our original network conditions, the source was chosen to be any one of the sixty-four nodes in the network, and the destination was chosen to be any one of the remaining sixty-three nodes in the network. With this method of choice, each message required an average of 2.29 hops between its source and destination. Thus, when the distance between source and destination in these simulations is set to 2 hops, the mean message delay is expected to decrease by about 10 percent.

Figure 5.8a shows a plot of the mean message delay versus the average link service time for the four cases discussed above. As is expected, the delay of the uniform hops case is approximately 10 percent below the delay of the standard system. Making the message length uniform also significantly decreases the mean message delay. This occurs because having a constant message



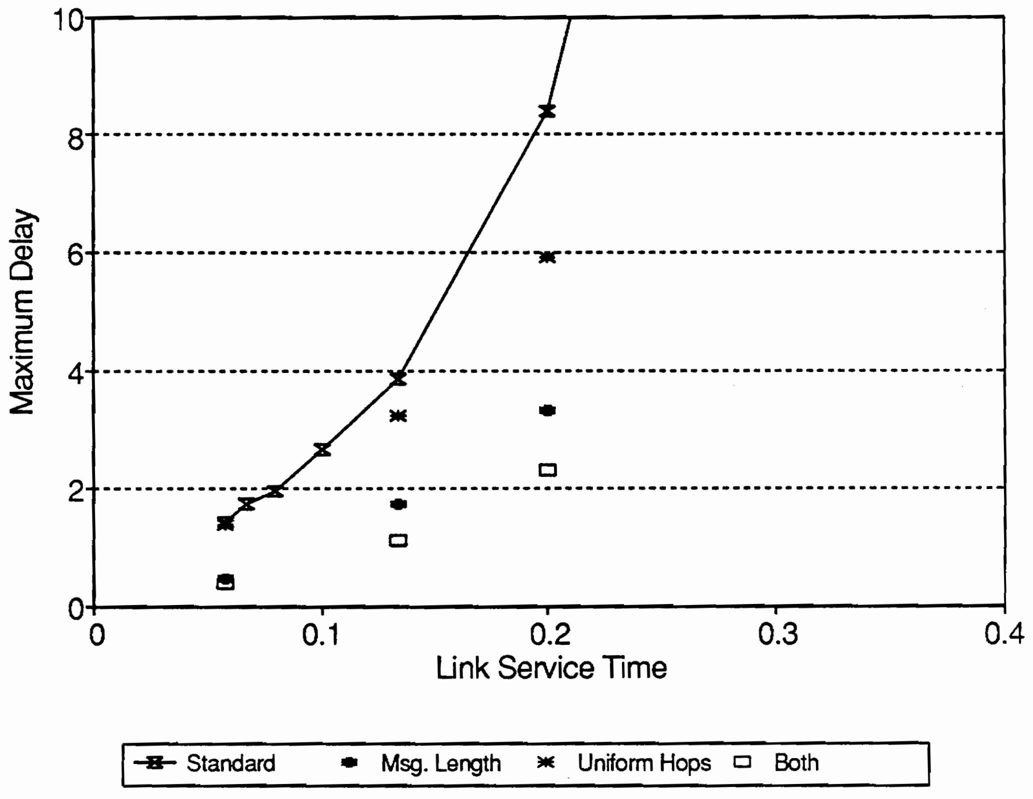
(a) Mean Delay vs. Link Service Rate

Figure 5.8. Message Parameter Effects



(b) Standard Deviation of Delay vs. Link Service Rate

Figure 5.8. Message Parameter Effects (continued)



(c) Maximum Delay vs. Link Service Rate

Figure 5.8. Message Parameter Effects (continued)

length causes there to be no long messages in the system. Thus, although the average length of the messages are the same whether the lengths are exponentially distributed or fixed, the mean delay is reduced because long messages cause all of the messages behind them in the system to wait for a long time. This effect is not totally offset by the fact that the exponential distribution also causes there to be short messages in addition to the long ones because a message arriving at a queue is more likely to arrive while a longer length message is using the service resource than when a short message is using this resource. As could be expected, the mean message delay is smallest when both uniform hops and a fixed message length are used.

The standard deviation and maximum message delays are affected much more by changing these message parameters than is the average delay. From the results of Figures 5.8b and 5.8c, which plot the standard deviation and maximum message delay versus the average link service time for the four message parameters described in this section, it is seen that the case using a fixed message length exhibits a significantly smaller standard deviation and maximum delay than the exponentially distributed case. To a lesser extent, making the number of hops uniform also decreases the delay. In similar fashion to the mean delay, the standard deviation and maximum delays are lowest for the case where both the message length and number of hops are made constant. Of particular interest is the fact that this significant reduction of delay occurs both for long and short link service times. This shows that the distribution of message lengths is a crucial parameter that must be carefully considered in the design of a real-time system. It also shows that the algorithm used for task allocation for a real-time application on a distributed memory computer should attempt to account for the distance between nodes that have to communicate in the system.

5.7. Summary of Results

The analytic and simulation results of this research reveal many interesting aspects of the design of real-time systems. This chapter showed that the results obtained analytically from the equations proposed in Chapter 4 are in close agreement with the results obtained from simulations of the FIFO protocol network model. Results obtained from the analytic equations also demonstrate the degree to which changes in the service times of the system affect the message delay.

Examination of the three network topologies showed that the torus and SBH topologies are well suited to certain applications. The torus performs effectively over a wide range of link service times, but the SBH performs better than the torus for short service time values. The DBH was seen to have a much poorer performance than both the torus and SBH.

Three network protocols were simulated and compared using the SLAM simulation language. The FIFO protocol was seen to outperform both the TDM and token passing protocols proposed in this research. It was noted, however, that the FIFO protocol assumes an efficiency that would be difficult to obtain in an actual system. Thus, the performance advantage of the FIFO protocol might be difficult to replicate in practice. From this reasoning, it was determined that none of the protocols studied in this research demonstrated a decisive advantage over the other protocols.

The final set of results obtained in this chapter tested methods for reducing delay by changing the queueing protocol used in the system and changing the message parameters of the system. The exponentially distributed message length parameter was determined to be the most significant contributor to message delay.

Chapter 6. Conclusion

6.1. Summary of Work

This research studies the performance of several network topologies, protocols, queueing priorities, and message parameters for real-time distributed memory computer systems. The three network topologies examined are the spanning bus hypercube, the dual bus hypercube, and the torus. The performance of the system is based on the mean, standard deviation, and maximum message delay.

The network model used in this research is based on a model proposed by Protopapas and Denenberg [21]. The separation of node and link queues in this model enables the relative contribution of node and link congestion to the total delay time to be studied separately. The ratio of the node-to-link service rate has a notable effect on the system performance. Initially, a network using a FIFO link contention protocol is studied to compare the performance of the three topologies as the service times of the links and nodes in the system are varied. Analytic equations are developed that predict the mean and standard deviation of delay for the three topologies under consideration. Also, the SLAM simulation language is used to verify results obtained from these equations and to approximate the maximum message delay time.

Time division multiplexing and token passing link access protocols are then simulated using SLAM, and the performance of these protocols is compared to the performance of the FIFO protocol. In addition, the priority schemes for the queues in the system are altered from the FIFO priority initially used, and the performance effects are examined. Finally, message parameters such as the distribution of message lengths and the distribution of message destinations are modified to quantify their contribution to the message delay.

6.2. Summary of Results

Two types of results were obtained in this research. The first results of this research are the equations discussed in Chapter 4 for the mean and standard deviation of message delay for the three network topologies. These equations can be used as a quick way to predict the performance characteristics of the communication network in a message passing distributed memory computing system. From this initial estimate, different proposed systems can be compared.

The second results obtained in this research are derived from the comparison of topologies, protocols, priority structures, and message parameters for a $W^D = 4^3$ network. Performance results obtained for these systems are discussed below.

The SBH demonstrates the best performance when the average link service time is small, whereas the torus topology demonstrates the best performance for large link service times. The performance of the DBH compares favorably with the SBH's and torus's performance only when the average link service time is very low. The DBH may be the preferred topology, however, in systems where the lower cost of the DBH outweighs its performance liabilities.

In the $W^D = 4^3$ network, the three protocols considered demonstrate similar performance effects. The FIFO protocol displays a slight advantage over the other two, but this advantage could very well disappear in an actual system that does not have the 100 percent efficiency of the FIFO protocol assumed in this research.

Changes in the priority schemes used in the queues of the network cause several interesting performance effects. When the average link service time is small, the FIFO priority scheme outperforms the other three priorities based on all three performance measures. When the average link service time is large, the oldest message first priority scheme performs best based on the maximum and the standard deviation of delay. Transmitting the shortest message first, however, displays the smallest mean message delay for this case. Giving priority to the longest messages in the queue has the worst performance based on all three performance measures for all of the link service times examined in this research.

The most significant reduction in delay variance occurs when the lengths of the messages in the system are constant instead of being exponentially distributed. To a lesser degree, making the distance a message travels in the system constant also produces better performance than systems where the message destinations are uniformly distributed. These two examples demonstrate that subtle changes in the message parameters of the system can drastically affect performance.

6.3. Future Research

The research described in this thesis demonstrates many aspects that affect the performance of a distributed memory system. The scope of this research, however, prohibited an exhaustive study of all of the network specifications discussed above. Specifically, only single stage cube topologies and three protocols are examined in this research. Additional research is needed to compare the performance of other topologies, such as multistage cubes, to the performance of single stage cubes. Similarly, the performance of a carrier-sense multiple access (CSMA) link access protocol could be compared to the protocols examined in this research.

The comparison of the three topologies in this research acknowledged that a cost analysis would be needed to determine which topology is preferable for a given application. Thus, the comparisons of this research could be expanded to examine the costs of the networks in addition to the system performance based on the delay parameters.

Although the study of additional topologies and communication protocols may produce interesting results, the most promising areas of additional research concern priority assignment and message parameters. The three queueing priorities studied in addition to the FIFO queueing priority demonstrate that the choice of a queueing priority significantly affects the system performance. All of the priorities discussed in this research used only local information to adjust the message flow in the network. New priority schemes that account for the current state of other queues in the network or record past system trends may very well be able to produce significant performance advantages over the localized priorities discussed in this research. Similarly, additional examination of message parameters, such as message length distributions and message destination distributions, may also produce valuable results.

Bibliography

1. W. Chu and L. Lan, "Task Allocation and Precedence Relations for Distributed Real-Time Systems," *IEEE Trans. Computers*, vol. C-36, no. 6, pp. 667-679, June 1987.

2. J. Stankovic, "Misconceptions About Real-Time Computing: A Serious Problem For Next Generation Systems," *Computer*, vol. 20, no. 10, pp. 10-19, Oct. 1989.

3. L. Green, "Planning for FDDI," *Proc. 12th Conf. on Local Computer Networks*, 1987, pp. 4-6.

4. E. Chow, H. Madan, and J. Peterson, "A Real-Time Adaptive Message Routing Network for the Hypercube Computer," *Proc. California Real-Time Systems Symposium,* 1987, pp. 88-96.

5. J. Strosnider, T. Marchok, and J. Lehoczky, "Advanced Real-Time Scheduling Using the IEEE 802.5 Token Ring," *Proc. Alabama Real-Time Systems Symposium,* 1988, pp. 42-52.

6. M. Joseph and A. Goswami, "What's 'Real' About Real-Time Systems ?," *Proc. Alabama Real-Time Systems Symposium,* 1988, pp. 78-85.

7. W.W. Chu and C. Sit, "Estimating Task Response Time With Contentions For Real-Time Distributed Systems," *Proc. Alabama Real-Time Systems Symposium,* 1988, pp. 272-281.

8. M.C. McElvany, "Guaranteeing Deadlines in MAFT," *Proc. Alabama Real-Time Systems Symposium,* 1988, pp. 130-139.

9. R. Rajkumar, L. Sha, and J. Lehoczky, "Real-Time Synchronization Protocols for Multiprocessors," *Proc. Alabama Real-Time Systems Symposium,* 1988, pp. 130-139.

10. I. Lee and S. Davidson, "Adding Time to Synchronous Process Communications," *IEEE Trans. Computers*, vol. C-36, no. 8, pp. 941-948, Aug. 1987.

11. L. Wittie, "Communication Structures For Large Networks of Microcomputers," *IEEE Trans. Computers*, vol. C-30, no. 4, pp. 264-273, April 1981.

12. D. Reed and D. Grunwald, "The Performance of Multicomputer Interconnection Networks," *Computer*, vol. 20, no. 6, pp. 63-73, June 1987.

13. J. Hayes, S. Colley, and J. Palmer, "Architecture of a Hypercube Supercomputer," *Proc. International Conf. on Parallel Processing*, 1986, pp. 653-660.

14. M. Rosenbaum, S. O'Malley, J. Gilmer Jr., "Survey of High Performance Parallel Architectures," BDM/ROS-86-1246-TR, BDM Corporation, McLean, Virginia, Dec. 1986.

15. A. Tanenbaum, "Computer Networks," Second edition, Prentice Hall, Englewood Cliffs, NJ, 1988.

16. "NCUBE At-A-Glance," NCUBE Corporation, Beaverton, Oregon, 1989.

17. "The New NCUBE Scalar Supercomputer: 27 GFLOPS for Science and 60,000 MIPS for Business," NCUBE Corporation, Beaverton, Oregon, 1989.

18. J. Gustafson, G. Montry, R. Benner, "Development of Parallel Methods for a 1024-Processor Hypercube," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 4, July 1988, pp. 609-638.

19. R. Arlauskas, "iPSC/2 System: A Second Generation Hypercube," *The Second Generation: A Technical Summary of the iPSC/2 Concurrent Supercomputer*, Intel Corporation, Beaverton, Oregon, 1988, pp. 9-13.

20. S. Nugent, "The iPSC/2 Direct-Connect Communications Technology," *The Second Generation: A Technical Summary of the iPSC/2 Concurrent Supercomputer*, Intel Corporation, Beaverton, Oregon, 1988, pp. 59-68.

21. D.A. Protopapas and J.N. Denenberg, "A New Model for Performance Analysis of Large Scale Multicomputers," *Proc. Phoenix Conf. Computers and Comm.*, 1987, pp. 451-456.

22. D. Reed and H. Schwetman, "Cost-Performance Bounds for Multimicrocomputer Networks," *IEEE Trans. Computers*, vol. C-32, no. 1, pp. 83-95, Jan. 1983.

23. C. Kruskal, Marc Snir, and Alan Weiss, "The Distribution of Waiting Times in Clocked Multistage Interconnection Networks," *IEEE Trans. Computers*, vol. 37, no. 11, pp. 1337-1352, Nov. 1988.

24. C. Kruskal, M. Snir, and A. Weiss, "On the Distribution of Delays in Buffered Multistage Networks for Uniform and Nonuniform Traffic (Extended Abstract)," *Proc. 1984 Int. Conf. Parallel Processing*, pp. 215-219.

25. L. Bhuyan, Q. Yang, and D. Agrawal, "Performance of Multiprocessor Interconnection Networks," *Computer*, vol. 22, no. 2, pp. 25-37, Feb. 1989.

26. H. Abdalla and S. Midkiff, "Performance Analysis of Interconnection Networks for Massively Parallel Multicomputers," *Proc. Symp. on the Frontiers of Massively Parallel Computation*, 1988, pp. 639-642.

27. H. Abdalla, "Performance Analysis of Multicomputer Network Interconnection Designs," *M.S. Thesis*, Virginia Polytechnic Institute and State University, Nov. 1987.

28. F. Guibaly, "Design and Analysis of Arbitration Protocols," *IEEE Trans. Computers*, vol. 38, no. 2, pp. 161-171, February 1989.

29. W. Cheng and J. Liu, "Performance of ARQ Schemes in Token Ring Networks," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 826-834, July 1988.

30. R. Raines, N. Davis, and W. Shaw, "Modeling, Simulation, and Comparison of Interconnection Networks for Parallel Processing," *Proc. Summer Computer Simulation Conference*, 1988, pp. 87-92.

31. R. Raines, "The Modeling, Simulation, and Comparison of Interconnection Networks for Parallel Processing," *M.S. Thesis*, Air Force Institute of Technology, Dec. 1987.

32. L. Kleinrock, "Queueing Systems: Vol. 1," John Wiley & Sons, Inc., New York, 1975.

33. D. Reed and R. Fujimoto, "Multicomputer Networks: Message Based Parallel Processing," MIT Press, Cambridge, Mass., 1987.

34. A.A.B. Pritsker, *Introduction to Simulation and SLAM II*, Systems Publishing Corporation, West Lafayette, IN, 1986.

APPENDIX A. MODEL NUMBERING SCHEME

The task of simulating the network model using SLAM requires that all of the nodes and links in the network be identified by a number. The number assignment is necessary to distinguish between different nodes in the network and to distinguish between different links in the network.

A.1. Node Numbering

All of the topologies used in the network use the same 4^3 lattice locations for the 64 network nodes. Individual nodes in the network are distinguished by the coordinate of the lattice location that they occupy. For a 4^3 network, the node coordinates are (d_2,d_1,d_0) where each of the three coordinates has a value between 0 and 3. Assigning a specific numerical value to each node is accomplished by assuming the three coordinates are actually a base four number. The numbers 0 through 63 can be assigned to nodes in the system by converting the base four number represented by the node coordinates to a base ten value. Thus, the node number is calculated as shown below.

$$\text{Node Number} = (d_2 \times 4^2) + (d_1 \times 4) + d_0.$$

Similarly, the node coordinates can be calculated from the node number by converting the base ten node number to a base four value. The FORTRAN program shown below will perform this operation.

```
      N = NODE NUMBER
     DO 100, I = 0,2
         TEMP = N
         N = INT ( N / 4 )
         D(I) = TEMP - ( N × 4 )
 100  CONTINUE
```

For this program shown above, D(I) is coordinate d_i. These equations can be used for all three of the topologies examined in this research.

A.2. Link Numbering

Both the number and the locations of the links in the network are different for the three topologies examined in this research. Thus, the scheme used to number links is discussed separately for each topology.

A.2.1. Spanning Bus Hypercube Link Numbering

Link numbers are assigned to links in the D_0 direction first, the D_1 direction second, and the D_2 direction last. In the 4^3 SBH, there are 16 links in each direction. Thus, links 0 through 15 are in the D_0 direction, links 16 through 31 are in the D_1 direction, and links 32 through 47 are in the D_2 direction. Begin by specifying a link direction, i.e. D_0, D_1, or D_2. Link numbers are assigned for this direction using the following procedure. The first number assigned to a link in this direction is equal to number to the direction, either 0, 1, or 2, multiplied by 16. Numbers are then assigned sequentially for links in the specified direction by stepping sequentially through the nodes of the network. Starting at node 0 in the network, the first link number in that direction is assigned. Then, examine node 1 in the network. If the link running through node 1 in the direction of interest has already been assigned a value, then proceed on to the next node of the network. Otherwise, assign the link at node 1 the next unassigned link value, i.e. if the link attached to node 0 was assigned the value 16, then the link attached to node 1 would be assigned a value of 17. Continue this process until all links in the given direction have been assigned values, then repeat this above process for all link directions until all links in the network have been assigned a value.

Using this technique to number links in the network, algorithms have been developed that can determine the number of the link attached to a given node in a given direction. Also, given the number of a link, it is possible to determine what direction the link is in and what nodes are attached to it.

For the node at location (d_2,d_1,d_0), the link attached to this node in the D_n direction is given by

$$\text{Link Number} = \begin{cases} d_1 + (4 \times d_2) & \text{if } n=0 \\ 16 + d_0 + (4 \times d_2) & \text{if } n=1 \\ 32 + d_0 + (4 \times d_1) & \text{if } n=2 \end{cases}$$

Given a link numbered L, the nodes attached to this link are calculated by the below algorithm.

First, the direction of the link, n, can be calculated by the equation

$$n = \text{INT}\,(L/16)\ .$$

Since the SBH links attach all of the nodes in a given direction, the four nodes connected to the link have the same coordinates except for the d_n coordinate, which takes on values 0 through 3 for the four different nodes. To calculate the other coordinates, two values, p_0 and p_1, are calculated by the steps shown below.

1) $N = L - (n \times 16)$
2) $p_0 = N - (4 \times (\mathrm{INT}\ (N/4)))$
3) $N = \mathrm{INT}\ (N / 4)$
4) $p_1 = N - (4 \times (\mathrm{INT}\ (N/4)))$

The coordinates of the nodes attached to the link can then be calculated by assignment of the values of p_0 and p_1 according to direction of the link. This assignment is done as shown below.

If $n = 0$, then	$d_0 = \{ 0,1,2,3 \}$,	$d_1 = p_0$,	$d_2 = p_1$.
If $n = 1$, then	$d_0 = p_0$,	$d_1 = \{ 0,1,2,3 \}$,	$d_2 = p_1$.
If $n = 1$, then	$d_0 = p_0$,	$d_1 = p_1$,	$d_2 = \{ 0,1,2,3 \}$.

Using this technique, a distinct value is assigned to all of the links in a SBH topology. Also, from the link number it is possible to calculate both the dimension that the link traverses and the coordinates of the nodes attached to the link.

A.2.2. Torus Link Numbering

The links in the torus are similar to those in the SBH, but there are four point to point links in the torus for every one bus link in the SBH. Thus, the method used to number links in the torus is similar to the method employed for the SBH topology.

The first quantity assigned for a link value in the torus is the value of the bus link in the SBH that connects the nodes that the torus point-to-point link is connecting. From this calculation, a value between 0 and 47 is obtained. Next, this value is multiplied by four. A final addition of between 0 and 3 to this value, as explained below, completes the numbering scheme. Letting n be the dimension of the torus link, if the link is between nodes $d_n=0$ and $d_n=1$, then 0 is added to this quantity. If the link is between nodes $d_n=1$ and $d_n=2$, then 1 is added. If the link is between nodes $d_n=2$ and $d_n=3$, then 2 is added. Finally, if the link is between nodes $d_n=3$ and $d_n=0$, then 3 is added. Using this technique, all of the links in the torus have a value assigned to them.

Given a specific link number, L, it is possible to calculate which two nodes are connected by this link in the following manner. First calculate the link number in the spanning bus hypercube that connects the same two nodes connected by this torus link. This is calculated by

$$L_{SBH} = \text{INT}\ (\ L_{TORUS}\ /\ 4\ )$$

By using the methods used for the SBH, it is then possible to determine the direction of the link and the coordinate values of the nodes attached to the link in all dimensions except for the dimension of the link. The values of the coordinates in the link dimension of the two nodes attached to the link are calculated by

$$d_n = L \text{ MOD } 4 \quad \text{and} \quad d_n = (L+1) \text{ MOD } 4.$$

This technique thus assigns a specific number between 0 and 191 to the 192 links in the torus topology, and it provides a method for calculating which nodes are attached to a given link.

A.2.3. Dual Bus Hypercube Link Numbering

An important difference between the DBH topology and the torus and SBH topologies is that there are not the same number of links in each dimension of the network. Because the DBH is a modification of the SBH where certain links in only the nonzero-dimensions are deleted, the zero-dimension links in the DBH are numbered in exactly as they were in the SBH. Thus, link numbers 0 through 15 are assigned to zero dimension links. Links in the higher dimensions, however, have to be handled differently. Nodes whose d_0 coordinate is 0 or 2 are connected to links in the D_1 dimension, whereas nodes whose d_0 coordinate is 1 or 3 are connected to links in the D_2 dimension. Numbers are assigned to links in secondary dimensions based on the calculations below.

For a node with coordinates (d_2,d_1,d_0), the direction, n, of the secondary link connected to this node is calculated by

$$n = (d_0 \text{ MOD } 2) + 1$$

The number of this link is given by

$$\text{Link Number} = \begin{cases} 16 + (4 \times d_0) + d_2 & \text{if } n = 1 \\ 16 + (4 \times d_0) + d_1 & \text{if } n = 2 \end{cases}$$

Next, calculations are shown that calculate which nodes a given link passes through. Assume that the link number under consideration is L. If L is less than 16, then the link is in the zero-dimension, and the nodes it is attached to can be calculated using the same method used for the SBH. If L is greater than or equal to 16, the dimension, n, of the link can be calculated by

$$n = [((L - 16) / 4) \text{ MOD } 2] + 1$$

The link connects the four nodes in the n dimension, and thus

$$d_n = \{ 0,1,2,3 \}.$$

The coordinate in the zero-dimension is given by

$$d_0 = \mathrm{INT}[(L - 16) / 4] .$$

The final coordinate, d_f ,i.e. d_1 if n=2 and d_2 if n=1, is calculated by

$$d_f = L - 16 - (4 \times d_0) .$$

This technique assigns the numbers 0 to 31 to the 32 links of the DBH and also enables the nodes attached to any link in the network to be calculated.

A.3. Assignment of SLAM Statement Numbers

The numerical assignments described above are used as a basis for numbering the nodes, link queues, and link servers in the SLAM simulation models. Some modifications of these numbers are made to make the numerical assignments compatible with the operation of SLAM. These changes are discussed below.

The node model used in SLAM assigns the number of the node to the queue that holds messages for this node while they await service. The QUEUE statement in SLAM, however, requires that positive integers be used to number all queues. Thus, the node values of 0 to 63 are incremented by one to cause the nodes in the SLAM model to be labeled 1 to 64. This number is used both to label the QUEUE statement representing the node queue and the ACTIVITY statement representing the node server.

Another adjustment with the network numbering scheme concerns the link servers and the link queues. ACTIVITY statements are used to represent the servers at both links and nodes in the network. Each ACTIVITY must be labeled with a unique number. The node code uses ACTIVITY numbers 1 through 64. Thus, while the SELECT statements of the SLAM code are labeled with S0 through SMAX, where SMAX is the largest link number in the network, the ACTIVITY statement numbers used by the link service code must begin with 65 instead of zero so the activity numbers used in the node code are not repeated.

The final numbering adjustment that must be made for the SLAM model is the number and labeling of the link queues. For the FIFO model, there is one link queue per link server, and the label of the link QUEUE statement is the same number as the number of the link ACTIVITY statement serving this queue. When the token passing and TDM protocols were introduced, however, one link queue is assigned to a link server for every node to which the link is connected. Thus, there are four QUEUE statements for each link server in the SBH and DBH, and there are two QUEUE statements for each link server in the torus. Also, the numbering of link QUEUE statements must begin at 65 so that none of the QUEUE values used in the node code are repeated. Thus, for link L, the number of the ACTIVITY serving this link is L+65. The queues corresponding to link L are denoted by 4L+1, 4L+2, 4L+3, and 4L+4 for the bus topologies, and 4L+1 and 4L+2 for the torus. In addition, the number assigned to the QUEUE statement corresponding to link queue Q is (Q + 64).

Appendix B. Numerical Simulation Results

This appendix reports the numerical results that are used in the plots of Figures 5.1 through 5.8. A short description of the quantities being plotted and the system parameters used for each plot are given for the data being displayed. For all of the results reported in this appendix, a 64 node network with $W^D = 4^3$ and $\lambda_N = 1.0$ is used. Results that increase without bound are reported as "xxxx."

Table B.1 FIFO Protocol Simulation Results

a) $\lambda_{NST} = 2 \times \lambda_{LST}$

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	1.553	.9890	8.417
7.5	15.0	.7542	.4594	3.817
10.0	20.0	.5060	.3126	2.650
12.5	25.0	.3816	.2370	1.938
15.0	30.0	.3078	.1952	1.743
17.5	35.0	.2553	.1605	1.435

DUAL BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	xxxx	xxxx	xxxx
7.5	15.0	1.949	1.231	8.469
10.0	20.0	.8942	.5637	5.012
12.5	25.0	.5953	.3702	2.970
15.0	30.0	.4494	.2854	2.376
17.5	35.0	.3634	.2324	1.956

TORUS

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	4.286	2.281	18.23
5.0	10.0	1.283	.8141	8.471
7.5	15.0	.7802	.5213	5.488
10.0	20.0	.5616	.3837	3.957
12.5	25.0	.4386	.3029	3.271
15.0	30.0	.3636	.2608	2.951
17.5	35.0	.3052	.2134	2.300

Table B.1 FIFO Protocol Simulation Results (continued)

b) $\lambda_{NST} = 4 \times \lambda_{LST}$

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	10.0	xxxx	xxxx	xxxx
5.0	20.0	1.337	.9658	8.163
7.5	30.0	.6250	.4473	3.721
10.0	40.0	.4128	.3038	2.525
12.5	50.0	.3090	.2303	1.858
15.0	60.0	.2480	.1904	1.704
17.5	70.0	.2052	.1562	1.392

DUAL BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	10.0	xxxx	xxxx	xxxx
5.0	20.0	xxxx	xxxx	xxxx
7.5	30.0	1.784	1.206	7.929
10.0	40.0	.7708	.5274	4.378
12.5	50.0	.5071	.3587	2.815
15.0	60.0	.3791	.2766	2.389
17.5	70.0	.3048	.2260	1.913

TORUS

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	10.0	2.552	1.839	17.09
5.0	20.0	.9809	.7671	7.985
7.5	30.0	.6103	.4958	5.103
10.0	40.0	.4436	.3665	3.822
12.5	50.0	.3471	.2895	3.041
15.0	60.0	.2883	.2505	2.876
17.5	70.0	.2422	.2045	2.167

Table B.2 TDM Protocol Simulation Results

a) Clock Period is Three Times the Average Message Transmission Time

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	3.647	2.533	18.39
7.5	15.0	1.439	1.036	8.977
10.0	20.0	.8384	.6294	4.584
12.5	25.0	.5714	.4348	3.351
15.0	30.0	.4316	.3333	2.773
17.5	35.0	.3480	.2678	2.563

DUAL BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	xxxx	xxxx	xxxx
7.5	15.0	4.718	2.975	19.62
10.0	20.0	1.951	1.275	9.637
12.5	25.0	1.153	.7785	6.515
15.0	30.0	.8122	.5644	4.959
17.5	35.0	.6141	.4384	3.130

TORUS

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	5.373	2.873	19.97
5.0	10.0	1.415	.9207	7.914
7.5	15.0	.8302	.5617	5.111
10.0	20.0	.5911	.4079	3.917
12.5	25.0	.4588	.3192	3.067
15.0	30.0	.3734	.2618	2.521
17.5	35.0	.3157	.2235	2.282

Table B.2 TDM Protocol Simulation Results (continued)

b) Clock Period is One Times the Average Message Transmission Time

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	2.447	1.753	13.81
7.5	15.0	.9998	.6696	5.858
10.0	20.0	.6217	.4158	3.611
12.5	25.0	.4478	.2995	3.112
15.0	30.0	.3496	.2319	2.597
17.5	35.0	.2857	.1921	2.035

DUAL BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	xxxx	xxxx	xxxx
7.5	15.0	3.258	2.130	15.50
10.0	20.0	1.303	.8677	5.986
12.5	25.0	.8079	.5402	5.006
15.0	30.0	.5777	.3811	3 .128
17.5	35.0	.4470	.2967	2.340

TORUS

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	4.948	2.612	19.49
5.0	10.0	1.354	.8807	8.483
7.5	15.0	.8104	.5596	5.792
10.0	20.0	.5806	.4072	4.253
12.5	25.0	.4517	.3191	3.459
15.0	30.0	.3691	.2636	2.862
17.5	35.0	.3126	.2235	2.458

Table B.3 Token Passing Protocol Simulation Results

a) Token Passing Time Is One-Third the Average Message Transmission Time

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	2.303	1.355	9.812
7.5	15.0	1.064	.5873	4.805
10.0	20.0	.7060	.3821	3.259
12.5	25.0	.5332	.2871	2.791
15.0	30.0	.4276	.2284	2.209
17.5	35.0	.3574	.1900	1.968

DUAL BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	xxxx	xxxx	xxxx
7.5	15.0	5.100	4.726	37.93
10.0	20.0	1.343	.7921	6.666
12.5	25.0	.8596	.4808	4.007
15.0	30.0	.6365	.3519	3.250
17.5	35.0	.5089	.2769	2.759

TORUS

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	5.398	2.772	20.20
5.0	10.0	1.543	.9240	9.962
7.5	15.0	.9387	.5787	6.331
10.0	20.0	.6756	.4222	4.617
12.5	25.0	.5299	.3332	3.665
15.0	30.0	.4341	.2738	3.074
17.5	35.0	.3681	.2332	2.614

Table B.3 Token Passing Protocol Simulation Results (continued)

b) Token Passing Time Is One-Tenth the Average Message Transmission Time

SPANNING BUS HYPERCUBE				
μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
2.5	5.0	xxxx	xxxx	xxxx
5.0	10.0	1.832	1.162	8.320
7.5	15.0	.8641	.5154	4.656
10.0	20.0	.5713	.3439	3.402
12.5	25.0	.4273	.2559	2.688
15.0	30.0	.3433	.2070	2.258
17.5	35.0	.2870	.1731	1.951

Table B.4 Queueing Priority Simulation Results

a) Oldest Message Sent First

SPANNING BUS HYPERCUBE				
μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	1.646	.9313	6.194
15.0	30.0	.3086	.1947	1.743

b) Longest Message Sent First

SPANNING BUS HYPERCUBE				
μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	2.076	1.949	19.75
15.0	30.0	.3124	.2003	1.743

c) Shortest Message Sent First

SPANNING BUS HYPERCUBE				
μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	1.333	1.008	18.11
15.0	30.0	.3032	.1950	1.784

Table B.5 Message Parameter Simulation Results

(FIFO protocol used for all simulations)

a) Constant Message Lengths

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	1.176	.4646	3.329
7.5	15.0	.6391	.2122	1.724
17.5	35.0	.2400	.0695	.4535

b) Constant Number of Hops

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	1.259	.7123	5.889
7.5	15.0	.6570	.3743	3.213
17.5	35.0	.2264	.1302	1.382

c) Constant Message Lengths and Number of Hops

SPANNING BUS HYPERCUBE

μ_{LST}	μ_{NST}	μ_{EE}	σ_{EE}	Max
5.0	10.0	.9583	.2448	2.315
7.5	15.0	.5517	.0979	1.113
17.5	35.0	.2122	.0214	.3768

Vita

John Mchenry received his B.S. and M.S. degrees in electrical engineering from the Virginia Polytechnic Institute and State University in 1988 and 1990, respectively. He is currently a Bradley Fellow and PhD student at VPI&SU where he is studying communication networks in parallel processing systems.

Since 1985, Mr. McHenry has been involved with the cooperative education program and has worked every summer for the Department of Defense in Fort Meade, Maryland. As a result of this work, he was awarded the Outstanding Electrical Engineering Cooperative Education Senior Award from VPI&SU in 1988.

[illegible]