1985

Token bus interconnection network for tightly-coupled multiprocessor systems

Douglas Wayde Jacobson
Iowa State University

Jacobson, Douglas Wayde

TOKEN BUS INTERCONNECTION NETWORK FOR TIGHTLY-COUPLED MULTIPROCESSOR SYSTEMS

Iowa State University

PH.D. 1985

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other _____

_____

_____

University
Microfilms
International

Token bus interconnection network for tightly-coupled multiprocessor systems

by

Douglas Wayde Jacobson

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering
Major: Computer Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1985

TABLE OF CONTENTS

I	INTRODUCTION	1
II	PROBLEM STATEMENT	4
III	RELATED WORK	8
IV	SYSTEM OVERVIEW	21
V	BUS INTERFACE UNIT DESCRIPTION	28
VI	MEDIA DESCRIPTION	66
VII	PERFORMANCE COMPARSON	74
VIII	CONCLUSIONS	93
IX	BIBLIOGRAPHY	96
X	ACKNOWLEDGMENTS	97

I INTRODUCTION

The field of computer architecture has changed dramatically in the past few years. One of the largest changes is the steadily declining cost of processors, memories, and related components. This decline in cost enables computer designers to include more processors in their designs. As the number of processors in a computer increases, the need for higher speed data communication paths increases. The interconnection network is the current limiting factor in most multiprocessor schemes. The various methods of interconnecting processors will be discussed in chapter III.

One argument for increasing the number of processors in a computer is to support a new generation of computers and computer languages. A certain class of problems contain a high degree of parallelism which maps well onto the new generation of computers. Ideally, multiprocessor configurations should greatly speedup the computation of problem solutions; however, in practice, commonly used interconnection schemes limit or even degrade the system throughput due to a variety of factors.

It should be mentioned that solving the interconnection problem is only part of the whole problem that is involved in using a multiprocessor system to its greatest potential. This dissertation will, however, only deal with the interconnection problem and not the problem of mapping software

onto a multiprocessor system.

The research described here will identify the limiting factors and provide a novel interconnection scheme based on local area network topologies that mitigates these factors.

The interconnection scheme described in the remainder of the dissertation is based on a token passing bus network. The number of buses has been increased to provide a higher system throughput. Each bus has a separate implementation of the network protocol. The token bus was chosen for its high bandwidth during heavy loads.

The intent of this research was not to devise a novel protocol for the interconnection network, but was to use the local area network concept to interconnect processors.

The node interface to the interconnection network is through a standard channel. The node communicates to the other nodes on the network as if the connection was point-to-point between the two.

The bus interface unit (BIU) handles all message routing and bus management. The node presents a destination address and a message to the BIU. This message is encoded and transmitted to the correct node on the next available bus.

There are two types of media supported by the network. One type consists of base band transmission over coaxial cable with a transmission rate of 10 Mbits. The other type is a byte wide bus with a transmission rate of 10 Mbytes.

Since the distance between processors is relatively small the media is not as critical as in a standard network where the distances can be thousands of feet.

Sufficient specification of the network is provided to enable the construction of a prototype system.

This dissertation is divided into several chapters. Chapter II is a statement of the problems that will be addressed in the thesis. Chapter III will briefly discuss prior work in the area of multiprocessor interconnection topologies. Chapter IV is an overview of the interconnection scheme along with a block diagram. The network protocol and media specifications are discussed in Chapters V and VI. Chapter VII compares the performance of the interconnection network with the other schemes discussed in chapter III. Chapter VIII contains the conclusions of the research.

II PROBLEM STATEMENT

There are several problems that occur with contemporary interconnection networks for multiprocessor topologies. These problems are focused on the number of connections to each processor and the number of lines in the network. Another problem is the need for complex switching mechanisms. Other schemes use complex routing techniques to interconnect the processors. The bandwidth of some interconnection methods is too small to support a large number of processors. There is a need to broadcast messages to all the processors. Not all interconnection schemes provide this capability. Each of these problems will be briefly discussed below.

A. Number of Connections

The number of connections for each processor can be a limiting factor in the implementation of some interconnection schemes. Since each processor connection will usually consist of several logic elements, the number of connections should be minimized. Another problem is the physical connection to the processor. Each physical connection requires additional space and generally is a costly component, along with decreasing the MTBF (mean time between failure) of the system (1).

The cost of connections is not linear when the number

of connections increases. This is due to the physical limitation of the space on the board and to the routing problems on the board.

B. Number of Lines

The number of interconnections in a network can be greater than the number of nodes, thereby limiting the number of processors on the network. This limit is due to the physical separation needed to run multiple lines with out cross talk. There are also physical problems with routing the lines to each processor. A backplane would be cost prohibitive if the number of processors is much greater than 200. For example, if there were 1000 lines between processors it would become difficult to separate the lines from the bundle of wires.

C. Complex Switching

Some interconnection schemes use large arrays of switches to connect the processors. The switching algorithms can become complex if the number of processors is much greater than 16. The switching must be done by each processor by adding routing information to the data packet or by a central controller. These methods do not always guarantee that a data path can be found, nor do they guarantee that the path will be conflict free. The switches themselves can be complex devices which increase the system cost

and decrease the MTBF of the system.

D. Complex Routing

Interconnection schemes which are not fully interconnected tend to use complex routing algorithms to route the information between processors. The processor needs to be able to store and route packets to and from other nodes. This adds complexity to the nodes. If a node becomes inoperative, an entire section of the network could be isolated from the rest of the network.

E. Bandwidth Limitations

The bandwidth of the industry standard bus structures is too low for a large number of processors (2). These buses are designed to interconnect a small number of processors when the message traffic is light. The low bandwidth becomes a problem when the amount of traffic between processors is large. This forces the software designer to restructure the programs to limit the communication between processors

F. Broadcasting

The multiprocessor environment has the need for a broadcast facility. Distributed system that handles replicated data on each node is a prime example of a system that needs to broadcast messages. Some interconnection schemes

cannot support broadcast communications and others can broadcast but with great difficulty. Some methods, hypercube for example, provide a separate bus for broadcasting (3).

G. Conclusions

Some of the six problems detailed above exist in all interconnection schemes. It would be next to impossible to eliminate all these problems from an interconnection scheme. The goal is minimize the effects the problems while maximizing the system throughput.

Chapter III will discuss several interconnection schemes proposed in the literature. Each of these schemes will be compared and the effects of the six problems on each method will be discussed.

III RELATED WORK

Eight interconnections methods have been proposed in the literature. Each connection method will be compared on the basis of performance, maximum number of processors, message delay, and the number of nodes a message must pass through. The way in which each method deals with the problems and constraints of a tightly-coupled multiprocessor system will be included.

A. Fully Interconnected Network

A fully interconnected network has a direct connection between each processor. Therefore, the number of interconnections is proportional to the square of the number of processors (4). This leads to both a large number of connections on each processor, and to a large number of lines in the network.

The problems of routing and switching do not exist in a fully interconnected network. The processor must have additional circuitry to route the message out the correct line to get to the desired processor. This routing is not complex since there is a one to one correspondence between the lines and the destination processor. The receiver must be complex to simultaneously receive the signals from all the buses and to buffer the messages until the processor can receive them.

This network is also capable of broadcasting. The bandwidth available in the network is generally greater than the bandwidth needed.

B Crossbar Network

The crossbar network, shown in Figure 1, has been used extensively in telephone switching exchanges. The number of switches needed can be as high as the square of the number of processors (4). There is only one connection to each processor. The switching algorithm is straight forward in that there is only one switching element needed to connect two processors. This network cannot be used to broadcast messages. Crossbar switches are not used much due to the high cost and complexity compared to the decreasing costs of the processors; however, the bandwidth of the structure exceeds the requirements of most applications (3).

C. Star Network

The star network, shown in Figure 2, has a central switching point which switches data among the processors. There is only one connection and one line per processor. The wiring in and out of the central unit can become overly complex if the number of processors is large. The routing and switching algorithms are simple, but the central controller will become more complex as the number of processors increases.

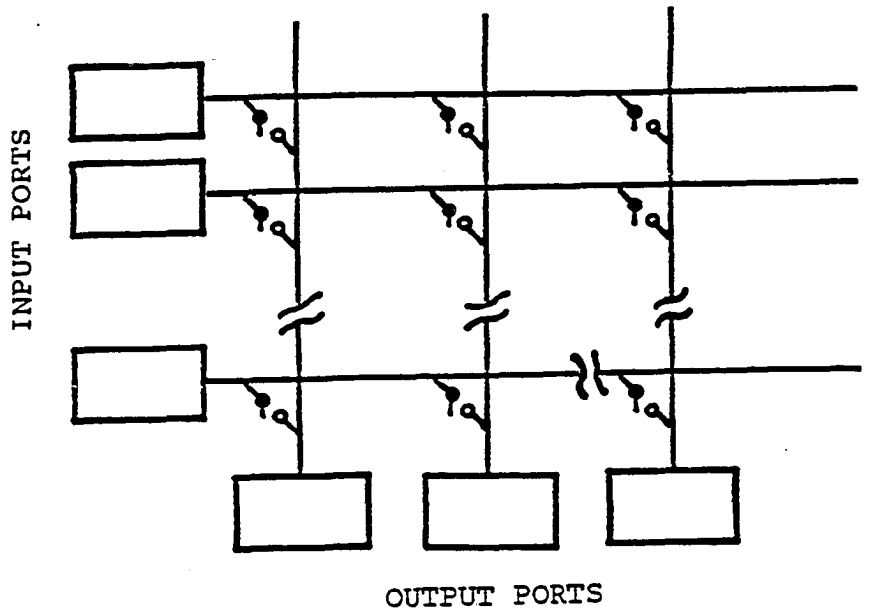


Figure 1 Crossbar network

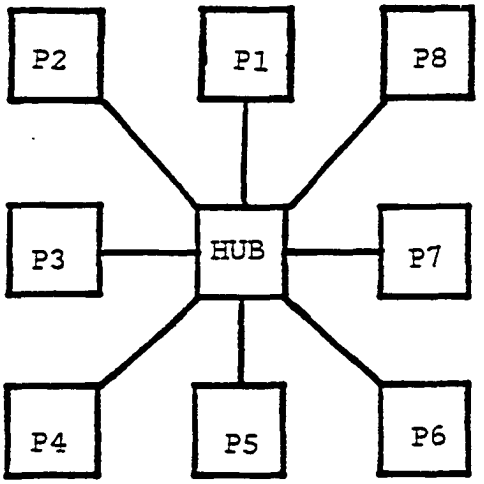


Figure 2 Star network

The central controller is a single failure point of the network. If the controller fails the entire network is inoperative. One advantage to the star network is that network can support broadcasting of messages.

D. Cube Network

The cube network, shown in Figure 3, uses levels of switching elements to interconnect the processors. The number of connections to each processor is one. The number of lines in the network is a function of the number of processors and the number of levels in the network. The switching algorithm can be complex if the switching is done dynamically. This dynamic switching can lead to blocking. Blocking occurs during the process of routing a message when a needed switching element has been used to connect another set of processors, thus disabling the processor from making the connection. Blocking can be eliminated by using an external controller, but this adds to the complexity of the network. This controller is also a central point of failure. Broadcasting is supported on this network.

E. Ring Network

The ring network, shown in Figure 4, consists of a series of shift registers all connected in a ring. All registers are clocked with the same clock.

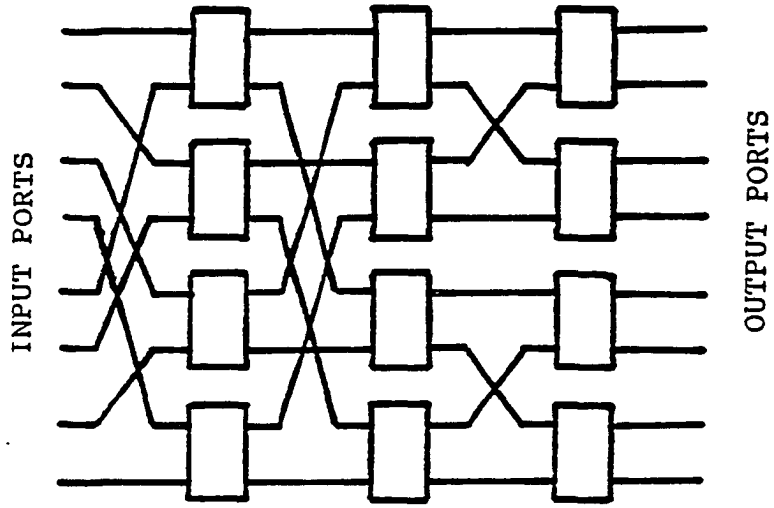


Figure 3 Cube network

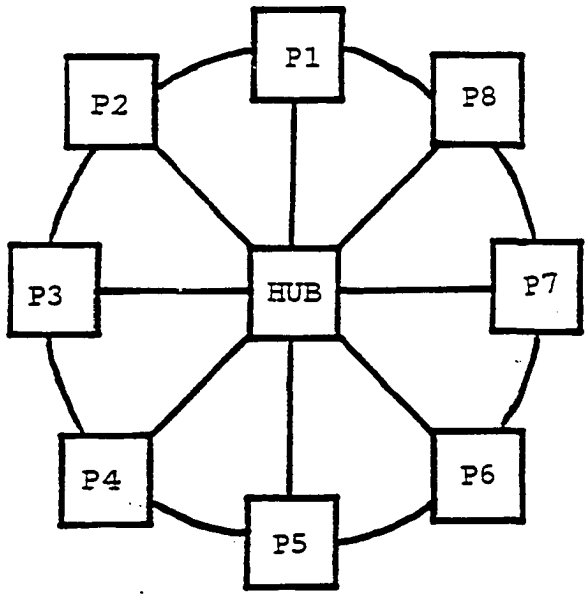


Figure 4 Ring network

The ring network has two connections for each processor. There is no routing problem since the message travels in a ring and eventually visits all processors. There is a problem with a possible failure of a register, which would disable the ring. Another problem exists when a message circulates around the ring without being received by any of the processors. One advantage of this topology is it supports message broadcasting.

F. Single Bus

The bus network is generally used in uniprocessor systems for data transfers between the processor, memory, and I/O units. The bus method can also be used for multiple processors on the same bus, however, only one processor can use the bus at a time. There is broadcast capability on the bus. The routing scheme is not very complex. A local area network, such as Ethernet (5), can be used as a bus network. This method will provide an increase in bandwidth over general purpose buses but is more costly to implement. These buses are limited to a small number of processors due to the bandwidth limitations of the bus. The bus generally transfers data at the speed of the processor and is synchronized to all the processors.

G. Hypercube

The Hypercube, shown in Figure 5, is a multidimensional cube with each processor having multiple connections. These connections are via point to point high speed channels (3). The number of channels depends on the number of processors in the cube. The routing and network management is accomplished with a global system master. The processors are connected to the manager via a separate high speed bus. The number of processors is limited by the bandwidth of the channel and by the number of connections to the processors. The system manager is the single failure point for the system. Since the system is not fully interconnected, there must be message routing among the processors. This system can broadcast messages via the global bus.

H. Multibus Network

A multibus network, shown in Figure 6, consists of multiple buses for communications among the processors. Since processors contend for the use of the buses, the number of buses limits the number of processors. The data are transferred at the speed of the processor and in synchronization with all other processors.

There is a bus arbiter to assign the buses to the processors (6).

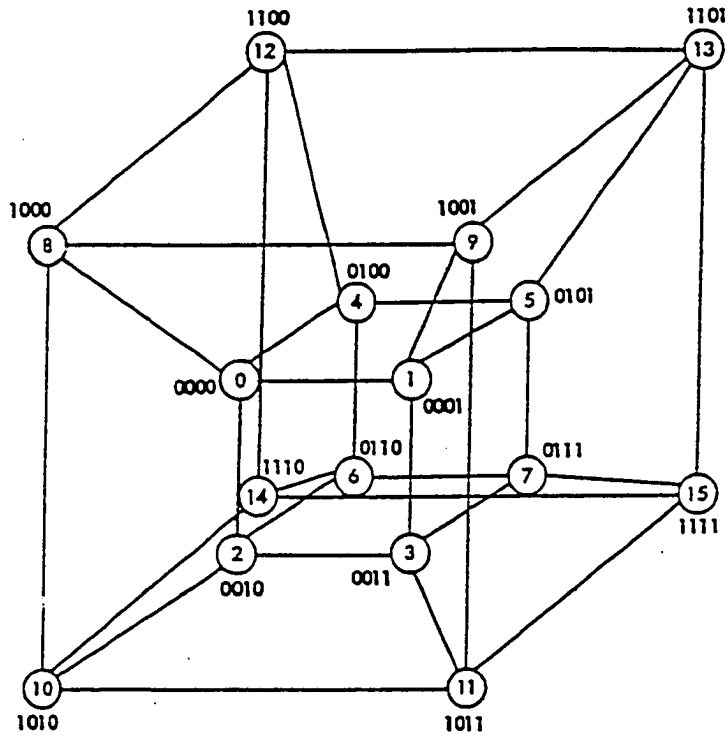


Figure 5 Hypercube network

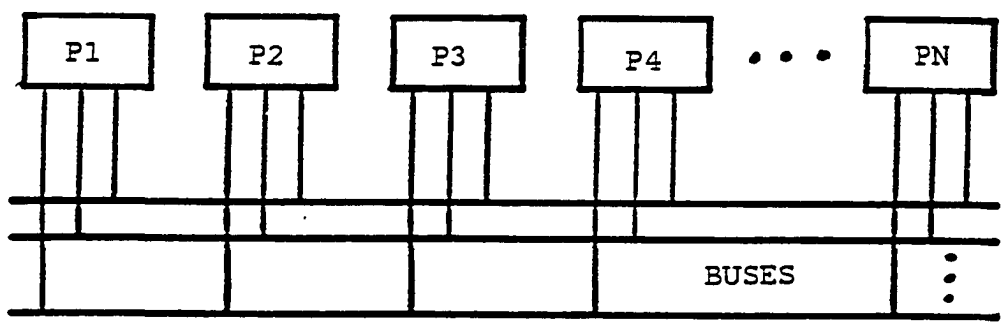


Figure 6 Multibus network

Once a processor is granted the bus, no other processor can use the bus until the request has been completed. This network is fault-tolerant since several buses must fail before the system becomes inoperative. The arbiter is a single point of failure for the system. The arbiter can be very complex if there is a large number of buses and processors. The number of connections can be large if the number of buses is large. A 16 processor system needs 11 buses to get the same bandwidth as a crossbar network (6).

I. System Comparisons

Three measures; performance, cost, and quality will be used to compare the networks. These measurements are described in a paper by E. Swartzlander and B. Gilbert (4). This paper deals with the first six networks discussed in this section. Parameters for the other two networks are obtained elsewhere from the literature (7).

1. Network performance

The parameters for network performance are media bandwidth and path length. Media bandwidth is the product of the number of active data links and the bandwidth of the individual links. Path length is the delay incurred in the transfer of the message from one node to another and back to the originating node. Table 1 shows the ideal network performance for all the networks.

Since the differences of a factor of 2 to 4 are probably negligible, Table 1 shows that the fully interconnected network offers the best performance while the single bus shows the worst performance. The other six networks are comparable in performance, if the number of buses approaches the number of processors in the multibus.

The fully interconnected network only offers good performance if the cost of the connection is linear. This assumption is only valid for a small number of processors. The fully interconnected network also provides more bandwidth than is needed by the system.

2. Network cost

The cost or complexity of a network is the sum of the link cost and the switch cost. Both costs have been assigned equal weight. Table 2 shows the network costs for each of the eight networks.

3. Network quality

The network quality is a ratio of network performance to network cost. The network quality for each of the eight networks is shown in Table 3.

4. Conclusions

In all cases, the network quality decreases with an increasing number of processors, indicating that these

networks are better suited for a small number of processors. Figure 7 is a plot of the network quality of the networks versus the number of processors. This shows the fully interconnected network is better for all sizes of networks.

These systems all present a problem when the number of processors increases above 64. The goal of this research will be to design a system that will allow more than 64 processors to be interconnected.

Table 1 Network performance

NETWORK TYPE	NUMBER OF MESSAGES (M)	LINK BAND-WIDTH (B)	ROUND TRIP DELAY (D)	PERFORMANCE NP = MB/D
CROSSBAR	N	1/N	4	1/4
STAR	1	1	4	1/4
CUBE	N	1	2LOG N	N/2 LOG N
RING	N	1	N	1
BUS	1	1/N	2	1/2N
FULLY CONNECTED	N	1	2	N/2
HYPERCUBE	LOG N 2^{N-1}	1	2LOG N	2^{N-1}
MULTIBUS	B	1/N	2	B /2N

Table 2 Network cost

NETWORK TYPE	LINKS (L)	COUNT (I)	TYPE (P T)	COMPLEXITY S = IPT	COST NC = L + S
CROSSBAR	2N	N^2	1:1	N^2	$N^2 + 2N$
STAR	2N	1 N	N:1 1:1	2N	4N
CUBE	N+N LOG N	(N LOG N)/2	2:2	2N LOG N	N + 3N LOG N
RING	N	N	2:2	4N	5N
BUS	1	N	1:1	N	N+1
FULLY CONNECTED	$(N^2 - N)/2$	N	1:N	N	$(3N^2 - N)/2$
HYPERCUBE	LOG N 2^{N-1}	N	1:8	8N	8N LOG N 2^{N-1}
MULTIBUS	B	BN	1:1	NB^2	B(N+1)

Table 3 Network quality

NETWORK TYPE	PERFORMANCE (NP)	COST (NC)	QUALITY NQ = NP/NC
CROSSBAR	1/4	$N^2 + 2N$	$1/(4N^2 + 8N)$
STAR	1/4	4N	1/16N
CUBE	N/2 LOG N	N + 3N LOG N	1/(2 N LOG N)(1+3 LOG N)
RING	1	5N	1/5N
BUS	1/2N	N+1	$1/(2N^2 + 2N)$
FULLY CONNECTED	N/2	$(3N^2 - N)/2$	1/(3N -1)
HYPERCUBE	2^{N-1}	8N LOG N 2^{N-1}	1/(8N LOG N)
MULTIBUS	B /2N	B(N+1)	$1/(2N^2 + 2N)$

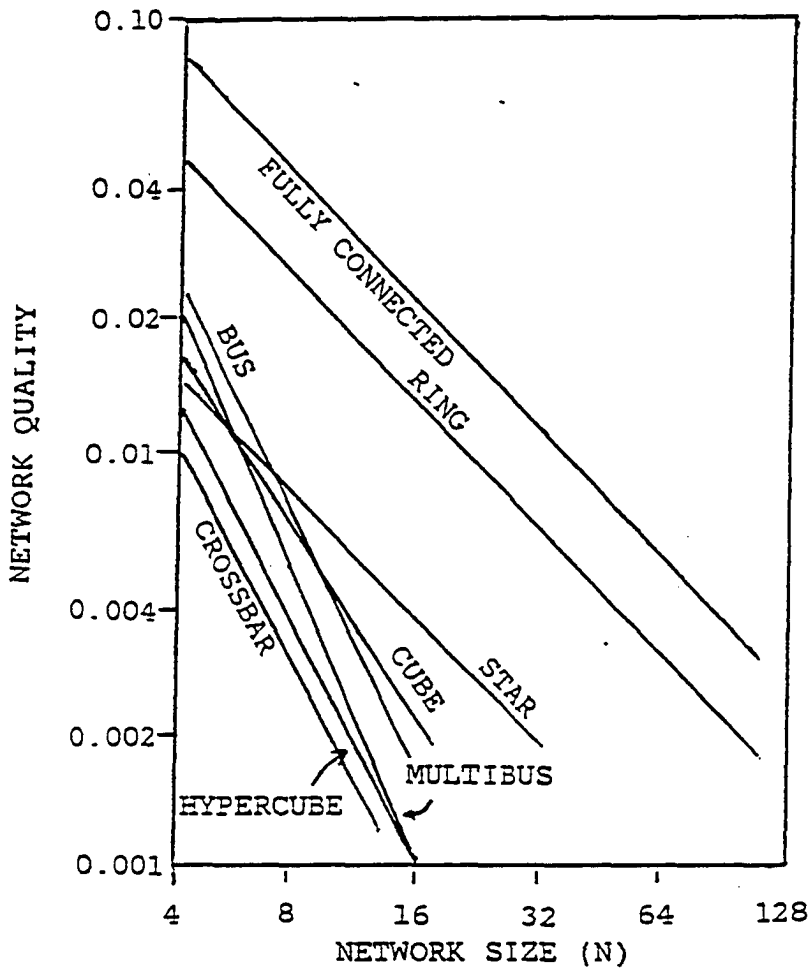


Figure 7 Network quality

IV SYSTEM OVERVIEW

The proposed system topology is presented in this chapter, along with a discussion of the system.

A. Block Diagram and Description

A block diagram of the system is shown in Figure 8. The system consists of a multibus network with multiple processors connected to the buses via a bus interface unit. The bus interface unit (BIU) is the heart of the system design, and this research effort. The transmission media for this system will be discussed in detail in Chapter VI. The number of buses and other system parameters will be discussed in Chapter VII. The buses use a token protocol to make the logical connections among the processors.

The logical interconnections among the processors can be reconfigured dynamically without the use of a central network controller. This simplifies the network and reduces the routing complexity that generally accompanies a dynamic network topology.

The multiple bus configuration, along with the distributed control, provides a fault tolerant network. Although it is not a topic within the scope of this research, this network topology could be used as the backbone for a fault tolerant system. The network is immune to failures on the media and in the transmitters and receivers.

B. Bus Interface Unit

A block diagram of the proposed BIU is shown in Figure 9. The BIU provides the connection and routing for the processors on the network. The interface between the BIU and the processor is a standard interface. The BIU contians memory to buffer the both the transmitted and received information. The BIU has a transmitter and receiver which connect to the media and can transmit and receive over multiple buses.

The exact configuration and function of the BIU will be shown in Chapter V. The protocol used between the different BIUs on the system is a token bus protocol and will be specified in Chapter V. The BIU is responsible for routing the data on to the correct bus and for making the logical connection between the processors. In addition the BIU is responsible for broadcasting messages system reconfiguration, lost tokens etc., in the event of a network failure. The BIU monitors all buses for information destined to the processor and for broadcasted messages.

C. Media

The media must be able to support multiple taps and have a high bandwidth. The bandwidth of the media must be sufficient to handle the traffic from multiple processors with a guaranteed maximum latency. The maximum latency,

along with the number of buses, will be shown in Chapter VII and is based on the number of processors in the network. The interface between the BIU and the network will be specified in Chapter VI along with the media description.

D. Token Bus Overview

The basic concept behind the token bus protocol is that a token controls the right of access to the medium (8-9). The token frame contains a destination address. The station which has the token has control over the media. The station may transmit one or more frames and may poll stations and receive responses. When the station is done or the time has expired, it passes the token on to the next station in the logical ring. The token is passed from station to station on the bus, thus forming a logical ring on the bus. The token passing frames are separate from the data passing frames.

Each station is required to provide basic network maintenance. This maintenance consists of ring initialization, lost token recovery, new station addition to the logical ring, and general house keeping of the ring.

Figure 10, shows a logical ring on the physical bus. This bus is a broadcast bus with each station receiving all the transmissions. Note that the physical connectivity has little impact on the order of the logical ring and that

the station can respond to a query from the token holder even without being a part of the logical ring.

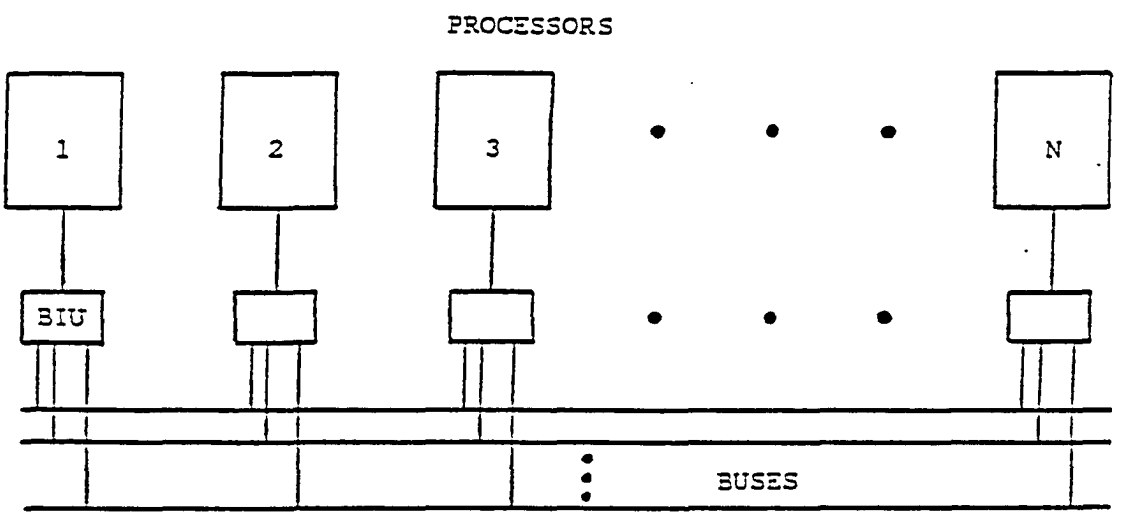


Figure 8 System block diagram

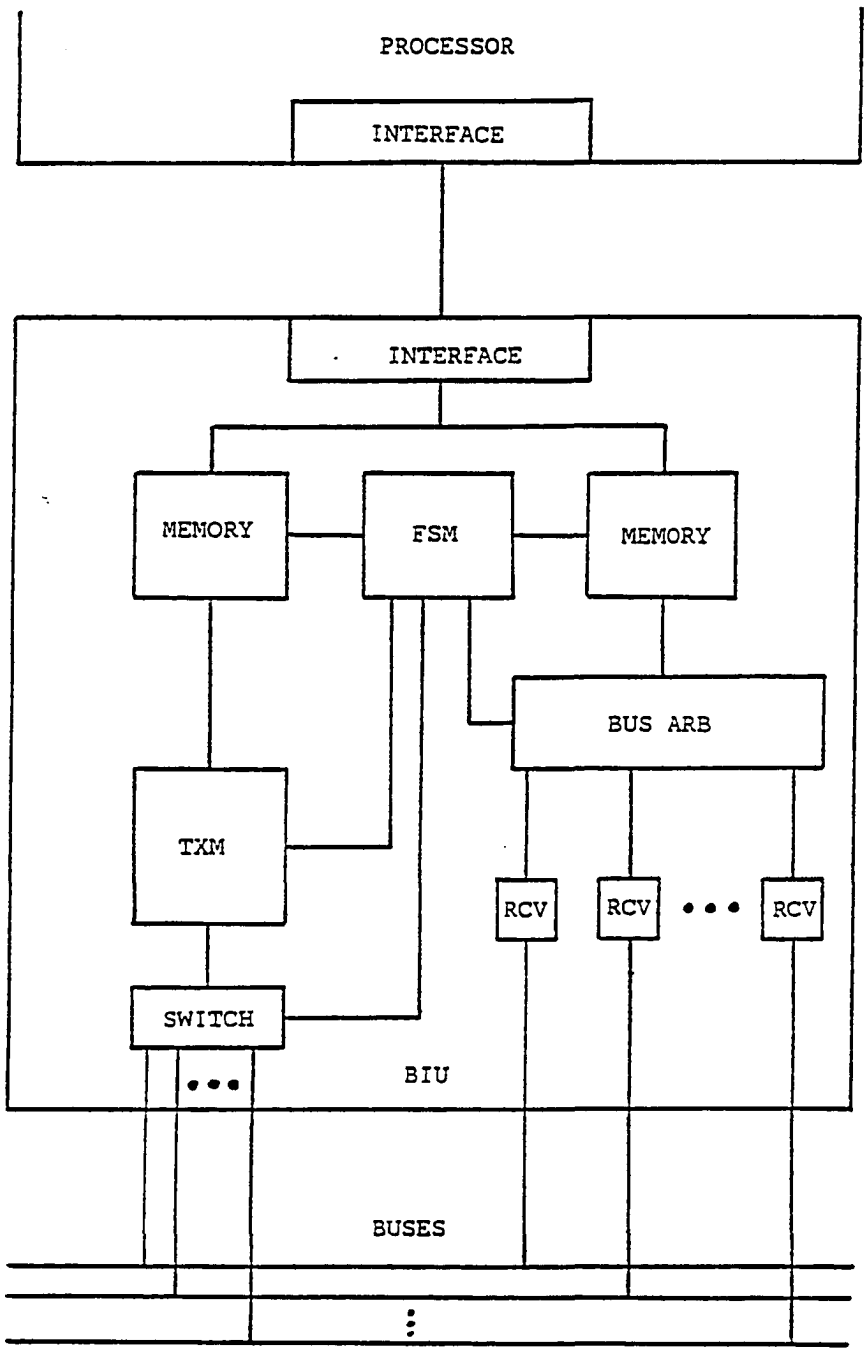


Figure 9 BIU block diagram

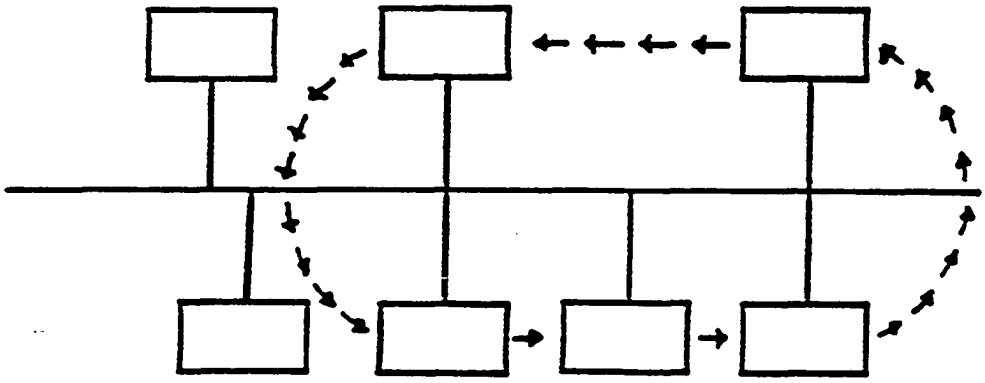


Figure 10 Logical ring on a physical bus

V BUS INTERFACE UNIT DESCRIPTION

This chapter will describe in detail the functions of the BIU and the protocol that is used by the network. Section A will describe functions provided by the BIU to the processor. Frame specifications will be described in section B. In section C a description of a proposed implementation will discussed along with the protocol definition

A majority of the information presented in this chapter was obtained from the IEEE 802.4 Token Passing Bus protocol specification (8). The protocol has been modified to better facilitate the implementation of this network. However, the basic structure of the 802 standard was maintained in order to minimize the complexity involved with creating a new protocol. By maintaining a close resemblance between the proposed protocol and the 802 protocol, a designer could use a standard chip set with minor modifications to implement the BIU.

A. BIU Functions

This section describes the services provided to the processor by the BIU. The functions discussed in the first section are used to provide a transparent transmission path between processors. These services include data transmission, data reception, and data confirmation.

The second section describes the local administrative services between the processor and the BIU. These services provide a method of resetting the BIU, selecting the node address, specifying the values of variables, and determining which buses the processor will use.

The service routines will be described as subroutine calls, but the exact form of the service call will not be specified. This specification will be done by the system designer and has no impact on this research.

1. Data service functions

The first data function provided by the BIU is transmit data request. This request is used when a packet of data is transmitted to another processor in the system. The service call has two parameters which consist of destination address and the data. The destination address specifies either an individual or a group address. The group address is used for broadcasting messages. Data packets consist of 0 to 8170 bytes.

The second data service function is "receive data indication." This service is used to indicate when a data packet has arrived from another processor on the system. This service call has three parameters which consist of destination address, source address, and data. The destination and source addresses are those of the receiving BIU, and the sending BIU respectively. Data packets consist of 0 to 8170

bytes.

The third service class is data transmit confirmation. This service provides confirmation on the success or failure of the last transmit data request. This service call has one status parameter to indicate the success or failure of the transmission.

A summary of the data service functions is listed below:

Transmit data request function
DATA.request(dest_addr, data)

Receive data request function
DATA.receive(dest_addr, src_addr, data)

Data transmit confirmation
DATA.confirmation(status)

2. Network service functions

The first two administrative service routines are used to initialize and reset the BIU. The first routine is initialize request which is used to initialize the BIU. The second routine is initialize confirmation which is used to confirm initialization. The request routine has no parameters and the confirmation returns the status of the initialization.

The next two service routines are used to set the values of the BIU internal variables. These variables include: station address, slot time, and timer values. These variables will be described in section C of this chapter.

The first of these two routines is a request to set a value. This call has two parameters which are variable name and value. The other routine is a confirmation of the set variable call. This call returns a success or failure status.

The next two routines are used to read the values of the BIU variables. The variables which can be read are the address of the successor and the address of the predecessor. The first of these two routines is a read variable request. This call is used to inform the BIU which variable is to be read. The second routine will read the value and the status of the read request.

The next two service routines are used to enter and exit the logical ring on a given bus. This first routine is used to set the ring membership and has two parameters. The first parameter is the ring number for which the action will take place. A ring number of zero will indicate the action will apply to all buses in the network which the station can connect to. The second parameter is the action to be taken. The two actions are in_ring and out_ring, which are used to enter and exit the ring. The second routine is the confirmation of the ring membership call. This routine has only the status as a parameter.

The next two service routines are used to determine which stations are on which buses. The first call is a read request which has one parameter. The parameter is the ring number for which the list will be returned. The second call

is the confirmation for the read request and has two parameters. The first parameter is the value of the membership list, and the second parameter is the status of the call.

The next service routine is used for reporting faults that the BIU has detected. The faults which are reported are "duplicate address" and "faulty transmitter". This call has just one parameter which is the fault that was detected.

The next two calls are used to set the group addresses that the BIU will respond to. The parameter to the first call is a list of group addresses only the addresses which were passed during the last call will be recognized. Loading zero group addresses will deactivate the group address recognition service. The broadcast group address (all bits one) can not be disabled and will always be recognized. The second call returns the status of the set group address call.

A summary of the network service functions is shown below:

```
Initialize BIU
     ADM_INIT.request()

Initialize BIU confirmation
     ADM_INIT.confirmation(status)

Set value of variable
     ADM_SET.request(name, value)

Confirmation of set value request
     ADM_SET.confirmation(status)

Read the value of a variable
     ADM_READ.request(name)
```

Confirmation of read value request
ADM_READ.confirmation(value, status)

Set ring membership
ADM_MEMBER.request(ring, action)

Confirmation of set ring request
ADM_MEMBER.confirmation(status)

Read membership list
ADM_LIST.request(ring)

Confirmation of read membership list request
ADM_LIST.confirmation(value, status)

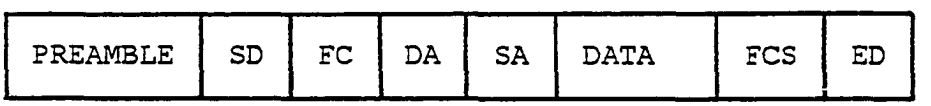
Report a network fault
ADM_FAULT.report(fault)

Set the group addresses
ADM_GROUP.request(set of addresses)

Confirmation of set group address request
ADM_GROUP.confirmation(status)

B. Frame Specification

This section describes the frame formats used by the BIU. The frames are sent by a BIU to another BIU. The valid frame types will be enumerated in this section. All frames have the following general format:



where

PREAMBLE = pattern sent to sync the receiver clock
SD = start delimiter (1 byte)
FC = frame control (1 byte)
DA = destination address (2 bytes)
SA = source address (2 bytes)
DATA = information (0 or more bytes)
FCS = frame check sequence (4 bytes)
ED = end delimiter (1 byte)

The following acronyms need to be defined:

TS - this station's address
NS - next station's address
PS - previous station's address

1. Frame components

a. Preamble The preamble precedes every frame and consists of an appropriate number of symbols which is denoted by the variable pad_idle. The preamble is used by the receiver to synchronize with the signal. The preamble symbol is chosen for each modulation scheme and data rate. The symbols chosen for the media will be discussed in Chapter VI.

A secondary purpose for the preamble is to guarantee a minimum ED to SD time period to allow stations to process the frame. This minimum amount of preamble is a function of the data rate and the modulation scheme.

b. Start delimiter The frame structure requires a start delimiter which consists of a pattern that is always distinguishable from data. The form of the start delimiter is shown below:

where
N = non_data symbol
0 = zero symbol

c. Frame control field The frame control byte determines what type of frame is being sent. These frames are classified into four general categories:

1) BIU control
2) Processor data
3) Station management
4) Special purpose data - reserved for future use

The frame formats for each of these categories is shown below:

where,

F F = Frame type:

1 2 ← bit positions

0 0 = BIU control frame
0 1 = Processor data frame
1 0 = Station management frame
1 1 = Reserved

A A A A A A = Action type:

F F = 0 0 (BIU control frame)

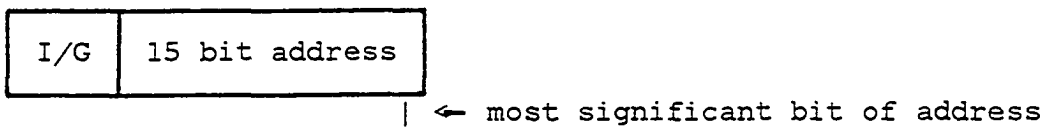
3 4 5 6 7 8 <— bit positions

```
0 0 0 0 0 0  =  Claim token
0 0 0 0 0 1  =  Solicit successor_1 (1 window)
0 0 0 0 1 0  =  Solicit successor_2 (2 windows)
0 0 0 0 1 1  =  Who follows (3 windows)
0 0 0 1 0 0  =  Resolve contention (4 windows)
0 0 1 0 0 0  =  Token
0 0 1 1 0 0  =  Set successor
```

```
F F = 0 1   (Processor data frame)
    = 1 0   (Station management)
```

```
3 4 5 6 7 8  <-- bit positions
-----------
0 0 0 0 0 0  =  request with no response
0 0 1 0 0 0  =  request with response
0 1 0 0 0 0  =  response
```

d. Address fields Each frame will contain two address fields. The first field is the destination field and is of the form shown below:



The I/G bit is used to distinguish between group and individual addresses.

where,
0 = individual address
1 = group address

Each station has a distinct individual address which is used to identify that station on the network. A group address is used to broadcast a message to many different stations on the bus. A group address of all ones indicates that the broadcast message is for all stations on the network.

The station address is used in determining delays in the contention process and transmissions lengths used in the claim token process.

e. Data field The data field consists of zero or more bytes of information that is sent between processors or BIUs depending on the control frame.

f. Frame check sequence field The frame check sequence (BIU) is a 32 bit Cyclic Redundancy Check (CRC) generator polynomial of degree 32. The FCS check sequence checks the frame control field, the address (DA, SA) fields, and the data field. The generator polynomial is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

g. End delimiter The frame requires an end delimiter to mark the end of the frame and indicate the position of the FCS field. The end delimiter consists of a pattern which is distinguishable from the data. The form of the end delimiter is as follows:

N	N	1	N	N	1	I	0
1	2	3	4	5	6	7	8

<— bit positions

where

N = non_data symbol
1 = one symbol
I = intermediate bit (1 = more messages from station
0 = end of messages)

h. Abort sequence The abort sequence is used to terminate a transmission which has already started. The abort sequence is shown below:

N N 0 N N 0 0 0	N N 1 N N 1 I 0

i. Enumeration of types The data below show how the components of a frame are arranged in the various frame types. Section C discusses the frames and describes the terms used in the data.

CLAIM TOKEN:
The frame has a data field whose value is arbitrary and whose length in bytes is 0, 2, 4, 6 times the system's slot time measured in bytes.

PRE	SD	00000000	DA	SA	(0, 2, 4, 6) * SLOT TIME BYTES	FCS	ED

SOLICIT_SUCCESSOR_1:
The DA of this frame is equal to the contents of the stations NS register and has a null data field. One response window follows the frame.

PRE	SD	00000001	DA	SA	FCS	ED	

one response window

SOLICIT_SUCCESSOR_2:
The DA of this frame contains the value of the stations NS or TS register and has a null data field. Two response windows follow this frame.

PRE	SD	00000010	DA	SA	FCS	ED		

two response windows

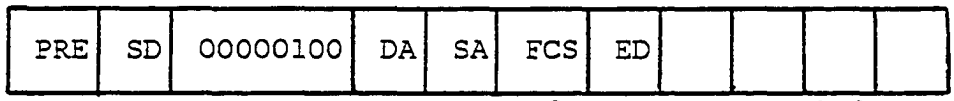
WHO_FOLLOWS:

The data unit in this frame has the value of the stations NS register. The format and length of the data unit is the same as the source address field. Three response windows follow this frame.

three response windows

RESOLVE_CONTENTION:

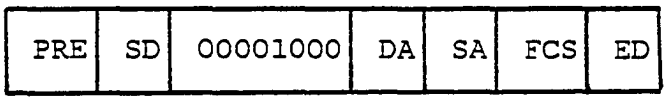
The data field of frame is null, and the there are four response windows following this frame.



four response windows

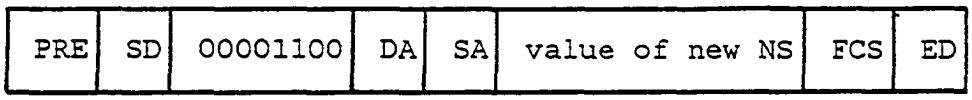
TOKEN:

The DA field of this frame contains the contents of the stations NS register and has a null data field.



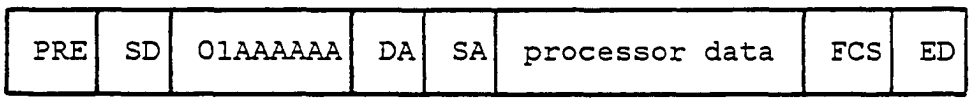
SET_SUCCESSOR:

The DA of this frame contains the SA of the last frame received, and the data unit contains the value of the NS or TS register. The format and length of the data field is the same as that of a source address field.



PROCESSOR DATA:

The DA of this frame will be the receiving stations address. The data field will be passed to the processor.



BIU MANAGEMENT DATA:
The DA of this frame is the receiving stations address. The data field will be passed to the BIU management unit.

PRE	SD	10AAAAAA	DA	SA	BIU data	FCS	ED

SPECIAL PURPOSE DATA:
The DA is this frame is the receiving stations address. The data field will be passed to an appropriate unit depending the service requested.

PRE	SD	11AAAAAA	DA	SA	data	FCS	ED

C. BIU Description

1. BIU internal structure

Figure 11 shows the internal structure of the BIU. The BIU is partitioned into four types of asynchronous machines. Each machine handles a different function of the BIU. Three of the four machines are replicated for each bus in the system and are identical for each bus. These partitions consist of the interface machine, access control machine, receive machine, and transmit machine. The access control, receive, and transmit machines are replicated for each bus.

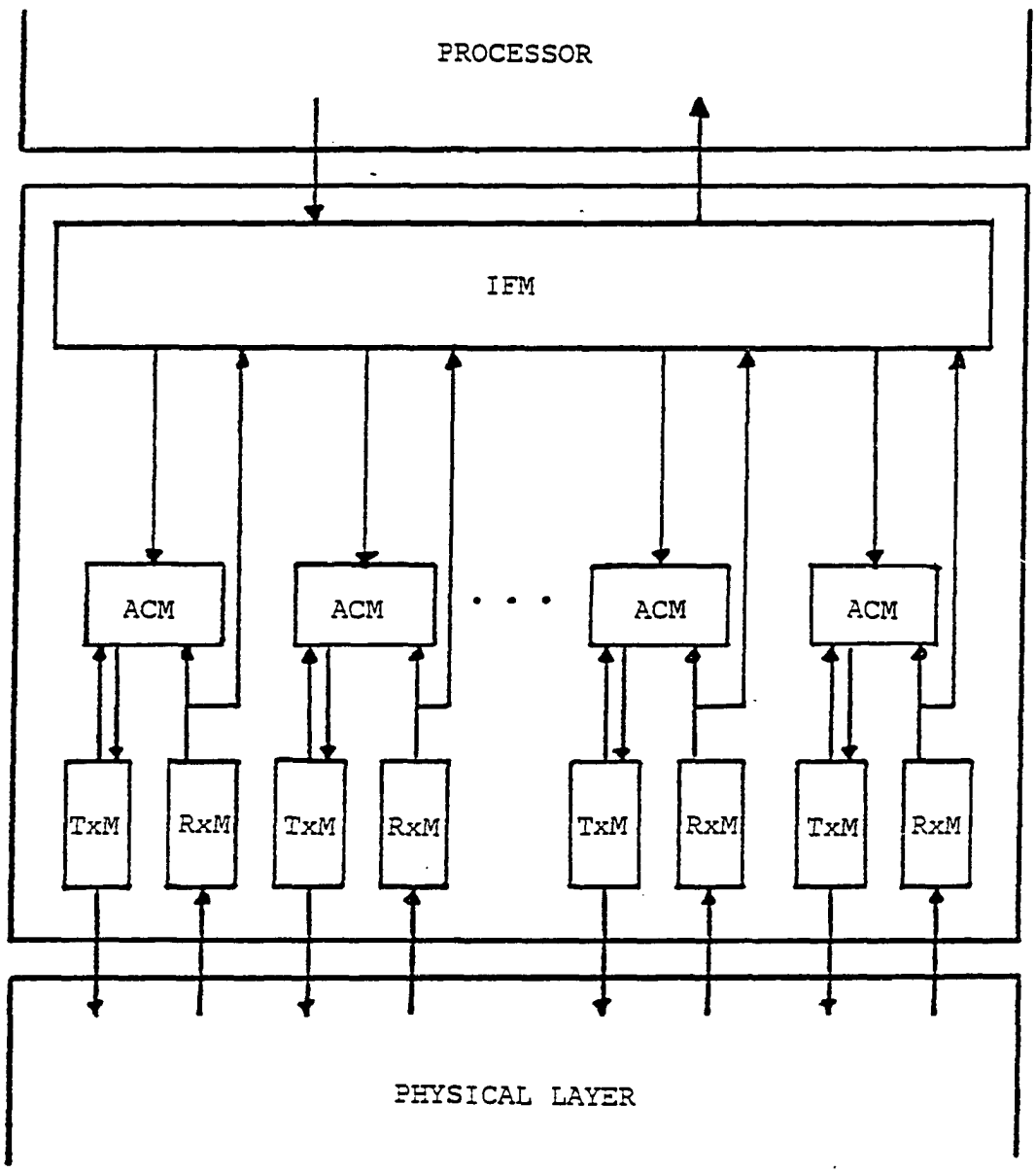


Figure 11 Internal structure of the BIU

The interface machine is the buffer and interface between the processor and the other BIU functions. This machine interprets all incoming function requests from the processor and generates outgoing service primitives. This machine also handles the address recognition function on received processor frames, accepting only those address to the processor. The interface machine is also responsible for the selection of the bus that will be used for the next message transmission. In addition, the interface machine will maintain the ring membership list for each bus.

The access control machine handles the bus protocol and the token management. The initialization and logical ring maintenance is controlled by this machine. The access control machine also has the responsibility for handling the detection and possible recovery from bus errors and network faults. There is one access control machine for each bus in the network.

The receive machine accepts input symbols from the physical layer and assembles them into frames. These frames are validated and passed to both the access control and interface machines. The receive machine detects the frame delimiters (SD and ED) and then checks the frame using the FCS. There is one receive machine for each bus in the network.

The transmit machine accepts data from the access control machine and transmits data to the physical layers as

symbols in the proper format. The transmit machine builds the frame by adding the preamble, start delimiter, end delimiter, and FCS. There is one transmit machine for each bus in the network.

Each of these machines will be discussed in detail in the next sections. The access control machine is the most critical and most complex of the machines.

The responsibilities of the BIU involve managing ordered access to the media, providing a means for admission and deletion of stations and fault recovery.

The faults which are handled are caused by communication errors or station failures and include:

1) Multiple tokens

2) Lost tokens

3) Token-pass failure

4) "Deaf" station

5) Duplicate station address

Some basic observations are useful in understanding the operation of the protocol.

1) Stations are connected in parallel to media. Therefore, a station's transmission is heard by all other stations. Other stations can interfere with, but can not predictably alter the contents of a station's transmission.

2) A transmitting station can assume that all other stations hear something, but not necessarily what was transmitted.

3) When a station receives a valid frame, it can infer that some station transmitted the frame and that all stations heard something.

4) When a station receives something other than a valid frame it can not assume anything about what another station might have heard.

5) Not all stations connected to media need to be involved in the token passing.

6) Any station can detect multiple or lost tokens. There is no special central controller.

7) Due to the spatial separation, stations can not be guaranteed to have common perception of the state of the system at any instant.

8) If one bus is inoperative, the station can not assume the condition of the other buses.

9) A station does not need to be a member of each bus in the network.

2. Basic operation

The following subsections describe major elements and features of the bus protocol used in the network. A majority of these elements are used to grant new stations access to the ring and to maintain the logical ring in case of failure.

a. Slot time Slot time is the maximum time a station needs to wait for a response from another station.

The slot time is known to the station before transmission can begin. The slot time is the same for each bus in the network. Slot time is defined as

slot_time = INT (((2 * (TPD + SD)+ SM) / ST + 7) /8

where,
TPD = Transmission path delay
SD = Station delay
SM = Safety Margin
ST = Symbol time

SM >= Symbol time

Station delay is the time between the station receiving the last symbol from the bus (end delimiter) and the time the station takes to start transmitting.

A symbol is the smallest unit of information exchanged between the BIUs. The six symbols used in the protocol are:

NAME	ABBREVIATION
zero	0
one	1
non_data	N
pad_idle	P
silence	S
bad_signal	B

b. Right to transmit The right to transmit is passed from station to station in descending numerical order of the station address. When the station receives a token addressed to itself, it has the right to transmit. When the station has completed transmission of the data, it passes the token to the next station in the logical ring. A station can be granted the right to transmit on several buses

at once. If a station has the token on one bus and receives the token on another bus, the station will pass the last token received to the successor, unless the token allows the station to send a message to station which is not on the other bus.

c. Token passing After a station has completed transmitting any messages, the station passes the token to its successor. After sending the token, the station listens for evidence that the successor has heard the token frame and is active. If the sender hears a valid frame following the token, it assumes the successor has the token. If a valid frame is not heard, it shall attempt to assess the state of the network.

The token is passed from a higher station address to a lower station address except when the station with the lowest address passes the token to the station with the highest address.

If the station hears a noise burst or the FCS is incorrect, the station waits four more time slots. If nothing is heard the station can assume the noise that was received was its garbled token, and this station can repeat the transmission of the token. If anything is heard on the bus, the station can assume the token was passed to the successor.

If the station does not hear anything after sending the token the first time, it will repeat sending the token and

perform the same monitoring.

If the successor does not transmit after the second token transmission, the sending station can assume the successor has failed. The sender then transmits a who_follows frame with the successor address in the data field. The station whose predecessor was the successor of the transmitting stations responds to the who_follows by sending a set_successor frame. This gives the sender a new successor.

If this fails twice the sending station sends a solicit_successor_2 frame with its own address as the DA and SA, asking any station in the ring to respond to it. If any station hears the request and desires to be in the ring, it responds and the ring is reestablished using response windows.

If the station still has no response it can assume that either all other stations have left the ring or that there is a major failure. The station will send all messages that are waiting to be transmitted and then repeats the token passing until there are no more messages. The station then listens to the bus waiting for a station to start transmitting.

d. Response windows New stations are added to the network via a controlled contention process using the response windows. A response window is one slot_time in length, and starts after the transmission of a frame. During the response window, the sending stations listens for a

response. If the station hears a transmission start during the response window, the station will continue to listen until the transmission is complete.

Two frame types, solicit_successor_1 and solicit_successor_2, indicate the opening of response windows for stations wishing to enter the ring. These frame specify a range of station addresses to which any station whose address fall in the range may respond. The solicit_successor_1 frame is used when the successor has a lower address than the station. The solicit_successor_2 frame is used for successors with any address, with the successors with the lower addresses responding to the first window and the successors with the higher addresses responding to the second window.

The sender transmits the frame and then waits for a response. If the sender receives a valid request, it will make the requesting station its successor and pass the token to it.

If more than one station responds to the solicit successor request, the sending station will hear only noise during the response window. The sending station then sends a resolve_contention frame. The station which tried to join the ring choose a two bit number from there address and listen for 0, 1, 2, or 3 slot times as determined by the two bit number. If the contending stations hear anything, they eliminate themselves from the arbitration. If they hear

silence, they continue to respond to resolve contention requests.

e. Bound on rotation time The rotation time is bounded by deferring the solicit successor procedure whenever the token rotation time is longer than a defined value. Each station has a timer used to measure the token rotation time, and if this time is too long the station will not perform the solicit successor procedure.

f. Initialization Initialization occurs when the inactivity timer expires. When this happens, the station sends a claim_token frame. When multiple stations try to initialize the network, a sorting algorithm is used to resolve the contention. The initialization of one of the buses has no effect on the state of the other buses.

Each initializing station sends a claim token frame whose length is a multiple of the system slot_time (0, 2, 4, 6). This multiple is selected by using bits from the station address. After sending the frame, the station waits one slot_time for his message and messages of other stations with same frame length to pass. If the station hears non-silence, the station reenters the idle state. If the station heard silence on the bus and unused bits remain in the address, the station will send the claim_token frame again using the next two bits of the address. After all bits are used and silence is still heard on the bus, the station holds the token.

g. Leaving the ring A station can leave the ring by simply not responding to the token passed to it. The network will then use the recovery methods described to patch out the station. A more efficient method is for the station to send a set_successor frame to its predecessor and pass the token.

h. Ring membership list Each station maintains a list of the processors which the station heard on the bus during the last complete token pass. There is a separate list for each bus in the network. The list is updated when the station receives the token frame for another station. If the station receives the token for another station it can assume the station is in the network. During the time it takes for the station to send the token to its successor and the time it takes for the token to return to the station, the station will update the list. If a station that was on the list during the prior token rotation did not have a token transmitted to it, that station can be removed from the list. This list is used to determine which bus will be used to transmit the message to a given station.

The ring membership list allows a station to be a member of only certain buses. This could be used to allow a station to have only one transmitter and receiver if the station was not a highly used resource. The membership list also increases the fault tolerance of the network since a station can still operate without all the transmitters and

receivers functioning.

i. Token hold timer The token hold timer is the amount of time a station can hold the token before it must be passed to the next station. The value of this timer is determined at system configuration time and has an impact on the token rotation time.

j. Broadcasting Routing of the broadcast messages is handled by the BIU by using the ring membership lists. The BIU will determine which buses the message must be sent down in order to broadcast the message to the correct stations. The BIU will also maintain a group membership list to determine which groups the nodes belong to.

3. Interface machine

The interface machine (IFM) acts as an interface between the processor and the other machines of the BIU. The IFM translates the service requests from the processor into internal requests. The IFM has seven primary functions, which are listed below:

1) Accept or generate the service requests specified in section A of this chapter.

2) Queue the pending service requests.

3) Recognize the addresses of the frames destine for this station.

4) Maintain a FIFO output queue for the station which

can be routed to any access control machine.

5) Maintain a FIFO input queue for the station which queues data from all the buses.

6) Maintain the ring membership list.

7) Message broadcasting

The functions and variables provided by the interface machine to the access control machine are described below. This interface between the two machines is described at a function level only.

a. Get pending frame This routine is used to transfer a frame from the output queues managed by the interface machine to the access control machine. This function has the bus number as the only parameter.

b. Any pending frame This boolean variable is true when there is a pending frame for the access control machine. There is one any_pending_frame variable for each access machine in the station.

c. In ring desired This boolean variable is true when the station desires to be in the ring for a given bus. There is one in_ring_desired variable for each access machine in the station.

d. Station has token This global boolean variable is true when any access control machine in the station has the token.

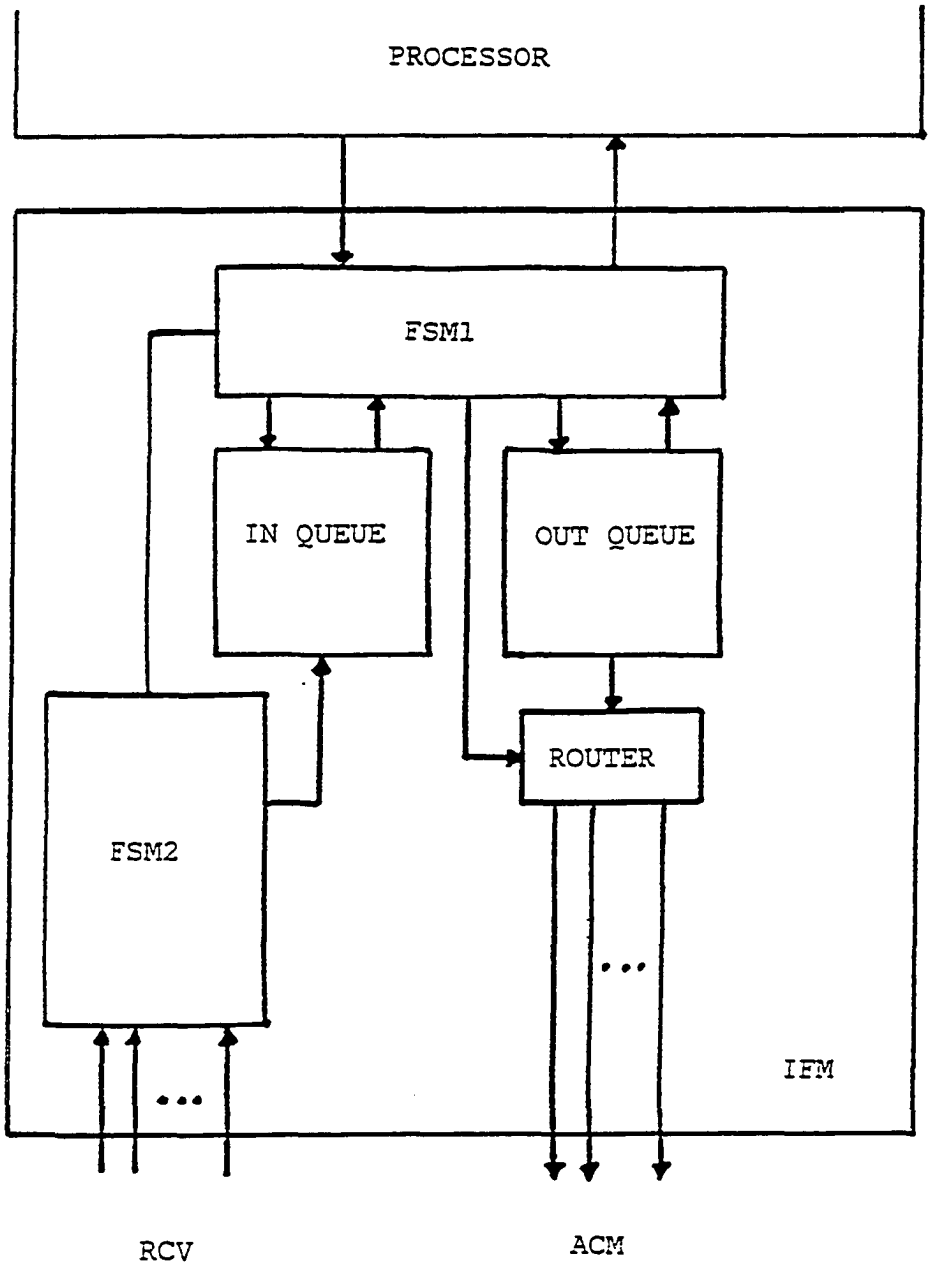


Figure 12 Block diagram of the interface machine

e. Power ok This boolean variable is true when the station has completed any power up testing and is ready.

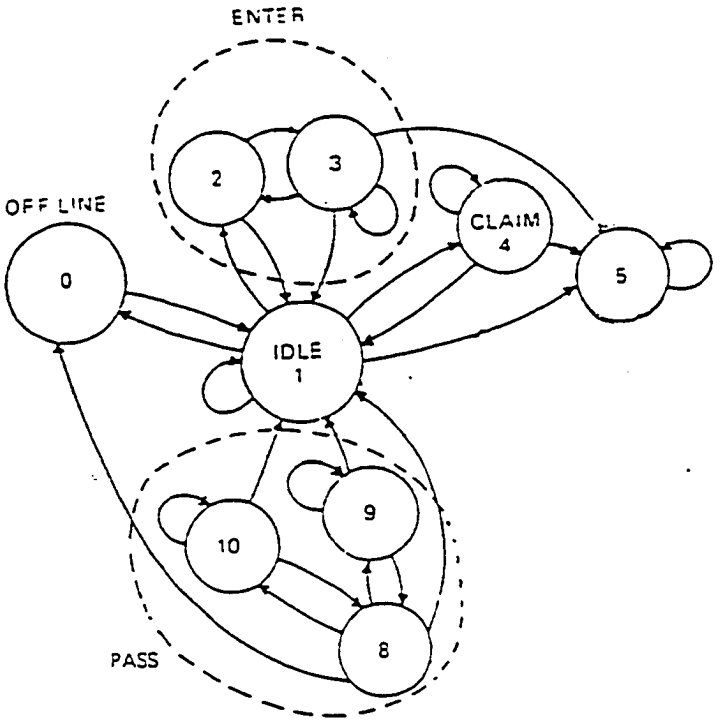
4. Access control machine

The access control machine (ACM) implements the protocol in the network. This machine is a state machine with eight distinct states. Each of these states will be described in this section. The state diagram for the access control machine is shown in Figure 13. A block diagram of the access control machine is shown in Figure 14.

a. Offline This state is the starting state which is entered after the power-up sequence or on certain fault conditions. The ACM remains in this state until instructed to go online.

b. Idle The station is listening to the medium but is not transmitting while in this state. If a control frame is received the ACM will take the appropriate action and enter a new state depending on the frame type.

c. Demand in This state is entered from the idle state when a solicit successor frame that spans the stations address is received. The station sends a set_successor frame from this state and goes to the demand delay state.



0 - OFFLINE
1 - IDLE
2 - DEMAND_IN
3 - DEMAND_DELAY
4 - CLAIM_TOKEN
5 - USE_TOKEN
6 - PASS_TOKEN
7 - CHECK_TOKEN_PASS
8 - AWAIT_RESPONSE

Figure 13 Access control machine state diagram

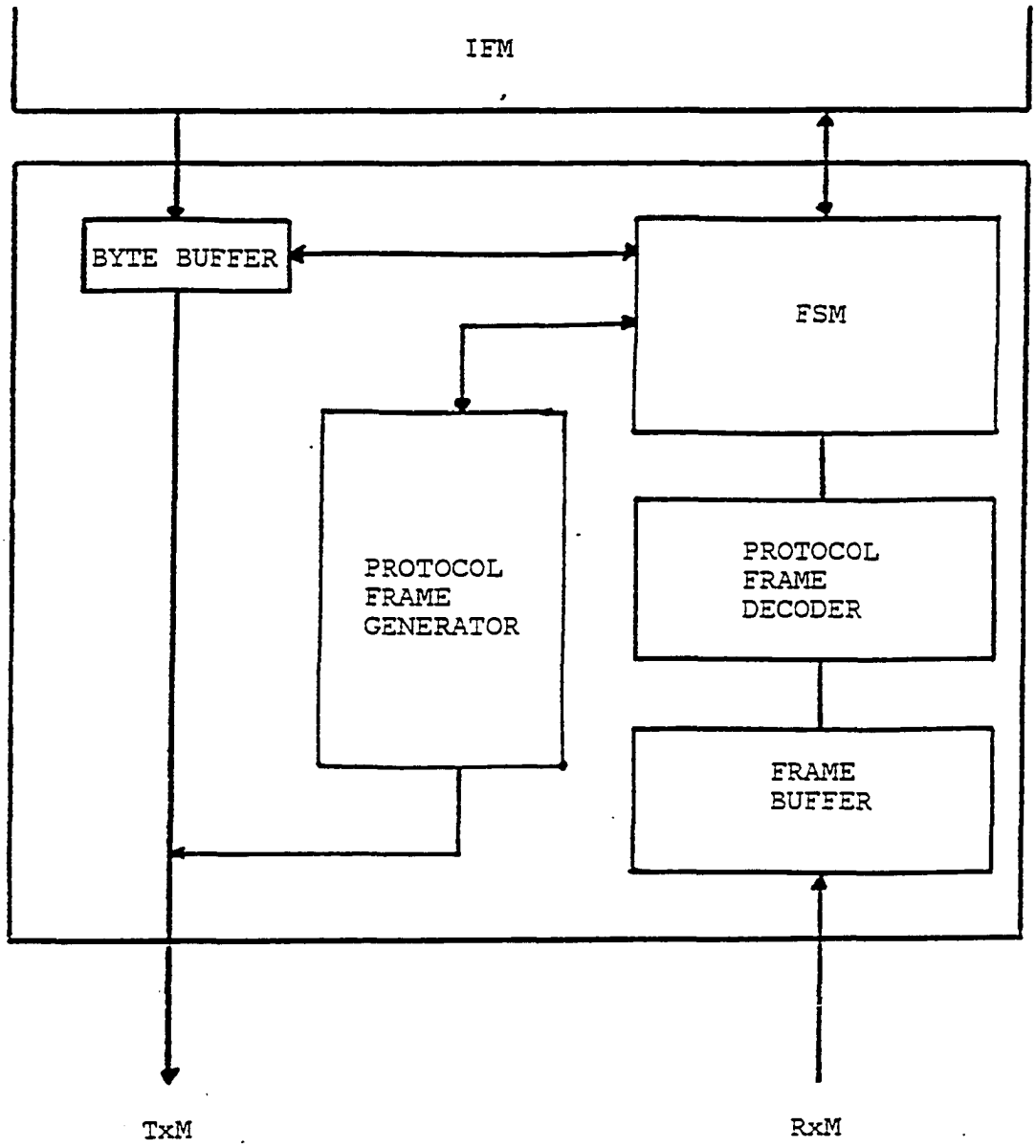


Figure 14 Access control machine block diagram

d. Demand delay The ACM waits for a response on the bus. There are four possible responses the ACM can receive. The first response is a token from the token holder. The token indicates the set_successor was valid and the ACM enters the use token state. The second response is a resolve_contention frame from the token holder indicating the ACMs involved should perform another set_successor. The third response is the ACM hears a set_successor from another station, which the ACM ignores. The final response is if any other frame or nothing is heard. In this case, the ACM returns to the idle state.

e. Claim token This state is entered when the inactivity timer has expired and the ACM desires to be in the ring. The ACM then attempts to initialize the ring.

f. Use token This state is entered after receiving or claiming the token. The ACM can start transmitting data frames until the token hold timer expires.

g. Pass token This state is entered when the ACM is trying to pass the token. After the token is passed, the ACM enters the check token pass state.

h. Check token pass This state is where the ACM waits for the response to the token pass. If the token pass was successful, then the ACM enters the idle state. If the token pass failed, the ACM returns to the pass token state.

i. Await response The ACM tries to find a successor. This state can be entered if the ACM does not know the

current successor or if the ACM is trying to add a new station to the ring.

Listed below are the state transitions for the ACM. These are the major transitions involved in the operation of the ACM.

Current State	Transition Name	Next State
0. OFFLINE		
	initialize	1. IDLE
1. IDLE		
	new_successor	1. IDLE
	no_successor	1. IDLE
	own_frame	1. IDLE
	non_bus_idle	1. IDLE
	other_heard	1. IDLE
	duplicate_address	0. OFFLINE
	entry_demand_in	2. DEMAND_IN
	repair_demand_in	2. DEMAND_IN
	ring_patch	2. DEMAND_IN
	no_token	4. CLAIM_TOKEN
	receive_token	5. USE_TOKEN
2. DEMAND_IN		
	lost_contention	1. IDLE
	continue_contention	3. DEMAND_DELAY
3. DEMAND_DELAY		
	end_all_contention	0. OFFLINE
	lost_contention	1. IDLE
	end_contention	1. IDLE
	unexpected_frame	1. IDLE
	long_bus_idle	1. IDLE
	contention_delay	2. DEMAND_IN
	ignore_contenders	3. DEMAND_DELAY
	ignore_noise	3. DEMAND_DELAY
	won_contention	5. USE_TOKEN
4. CLAIM_TOKEN		
	lose_address_sort	1. IDLE
	continue_address_sort	4. CLAIM_TOKEN
	won_address_sort	5. USE_TOKEN
5. USE_TOKEN		
	send_frame	5. USE_TOKEN
	no_send	6. PASS_TOKEN

6. PASS_TOKEN

sole_station	1. IDLE
pass_token	7. CHECK_TOKEN_PASS
open_response_windows	8. AWAIT_RESPONSE
who_follows_query	8. AWAIT_RESPONSE
solicit_any	8. AWAIT_RESPONSE

7. CHECK_TOKEN_PASS

pass_ok	1. IDLE
token_pass_failed	6. PASS_TOKEN
own_frame	7. CHECK_TOKEN_PASS

8. AWAIT_RESPONSE

unexpected_frame	1. IDLE
no_response	6. PASS_TOKEN
resolution_succeeded	6. PASS_TOKEN
resolution_failed	6. PASS_TOKEN
own_address	8. AWAIT_RESPONSE
hear_successor	8. AWAIT_RESPONSE
collision	8. AWAIT_RESPONSE
send_resolve	8. AWAIT_RESPONSE

Listed below is a short description of each transition signal.

Transition name	Description
initialize	station power-up initialization
new_successor	received set_successor frame
no_successor	received set_successor frame with this stations address as new_successor
own_frame	received own frame
non_bus_idle	heard noise burst
other_heard	heard something other than what was expected
duplicate_address	detected duplicate address
entry_demand_in	heard solicit successor frame
repair_demand_in	heard solicit successor frame
ring_patch	heard who_follows with address in range

no_token	bus_idle_timer expired
receive_token	heard token frame for this station
lost_contention	heard frame or noise burst
continue_contention	bus is quiet
end_all_contention	error condition
lost_contention	another station won contention
end_contention	contention ended without a winner
unexpected_frame	heard non_token frame
long_bus_idle	bus idle timer expired
contention_delay	still resolving contention
ignore_contenders	heard set_successor frame
ignore_noise	heard noise
won_contention	received token
lose_address_sort	heard other station
continue_address_sort	heard no other station and sort not finished
won_address_sort	heard no other station and sort is finished
send_frame	data to send and token hold timer not expired
no_send	no data to send or token_hold timer expired
sole_station	no other stations detected
pass_token	send the token
open_response_windows	allow other stations in the ring
who_follows_query	determine who follows after failed token pass
solicit_any	try to find any station after failed

	token pass
pass_ok	heard another station
token_pass_failed	heard no other station
own_frame	heard own frame
unexpected_frame	heard an unexpected frame
no_response	heard silence
resolution_succeeded	found new successor
resolution_failed	found no new successor
own_address	heard own frame
hear_successor	heard a successor frame
collision	heard noise burst
send_resolve	response window timer expired

5. Receive machine

The receive machine (RxM) accepts symbols from the physical layer and generates higher level data structures and signals for the access control and interface machines. The interface between the receive machine and the physical layer will be described in Chapter VI. A block diagram of a receive machine is shown in Figure 15.

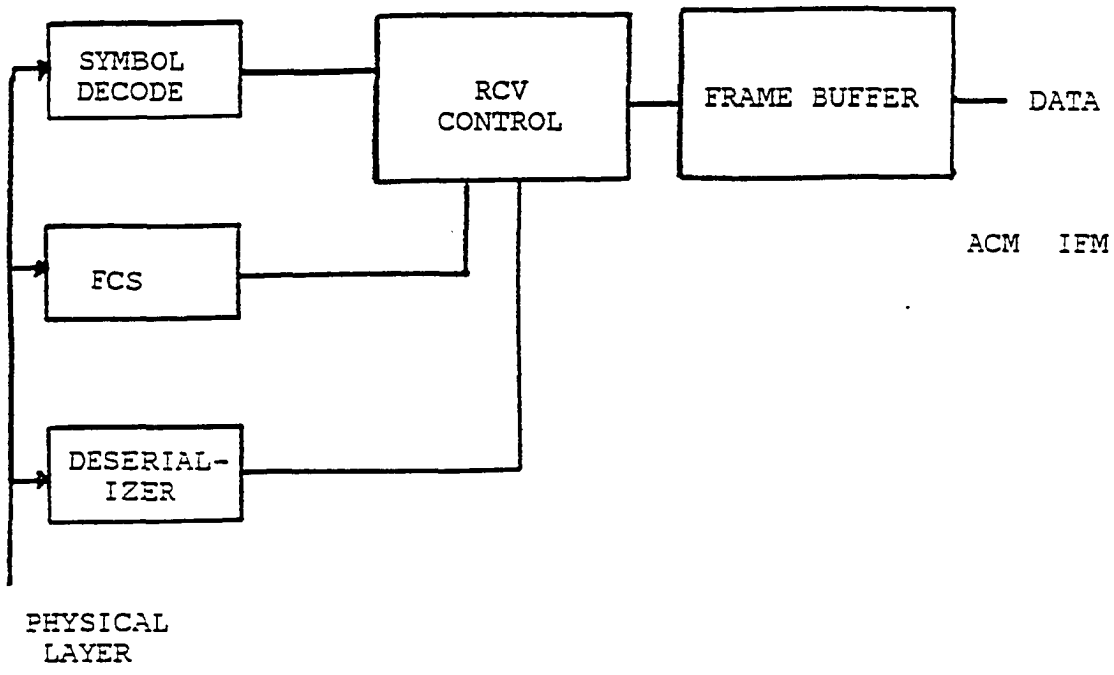


Figure 15 Receive machine

The signals used by the receive machine are described below:

a. bus quiet The signal is asserted when the bus is inactive. The receive machine sets and clears this signal, which is used by the access control machine.

b. Rx protocol frame This boolean variable is true when a valid control frame has been received. The variable is set by the receive machine, and is read and cleared by the access machine.

c. Rx data frame This boolean variable is true when a valid data frame has been received. The variable is set by the receive machine, read by the access and interface

machines, and cleared by the access machine.

d. noise burst This signal indicates that the bus has activity but there is no valid frame. The signal is set by the receive machine, and is set and cleared by the access machine.

e. frame buffer A valid frame is stored in the frame buffer before it is sent to the access or interface machines. The frame buffer contains the frame format, destination address, source address, data unit, and FCS fields of the received frame. When Rx_protocol_frame is set, the frame is used by the access control machine. When Rx_data_frame is set, the frame is used by the interface control machine.

6. Transmit machine

The transmit machine (TxM) has a simple operation. The transmit machine forwards the frames to the physical layer for transmission from the access control machine. The transmit machine first sends the preamble, then the SD and lastly the frame from the access machine. After the frame from the access machine is sent, the transmit machine sends the ED and then computes and sends the FCS. All transmissions from the transmit machine to the physical layer are done one symbol at a time. A block diagram of the transmitter is shown in Figure 16.

The functions and variables provided by the transmitter

to the access control machine are listed below. This interface between the interface and access control machines is described at the function level only.

a. Frame buffer empty The boolean variable is true when the transmitter frame is empty. The variable is set and cleared by the transmitter.

b. Transmit frame This function is used to transfer the frame from the access control to the transmit frame buffer.

c. Abort transmission This function is used to stop the transmission and send the abort sequence.

d. Transmitting This boolean variable is true when the transmitter is transmitting. The variable is set and cleared by the transmitter.

D. Conclusions

This chapter detailed the functions of the BIU and the protocol used by the BIU. The description of the BIU provided in this chapter was at the functional level and did not include implementation detail. Implementation detail depends on the system designers choice of hardware and software. However, the description provides all necessary guidelines implementation of the network.

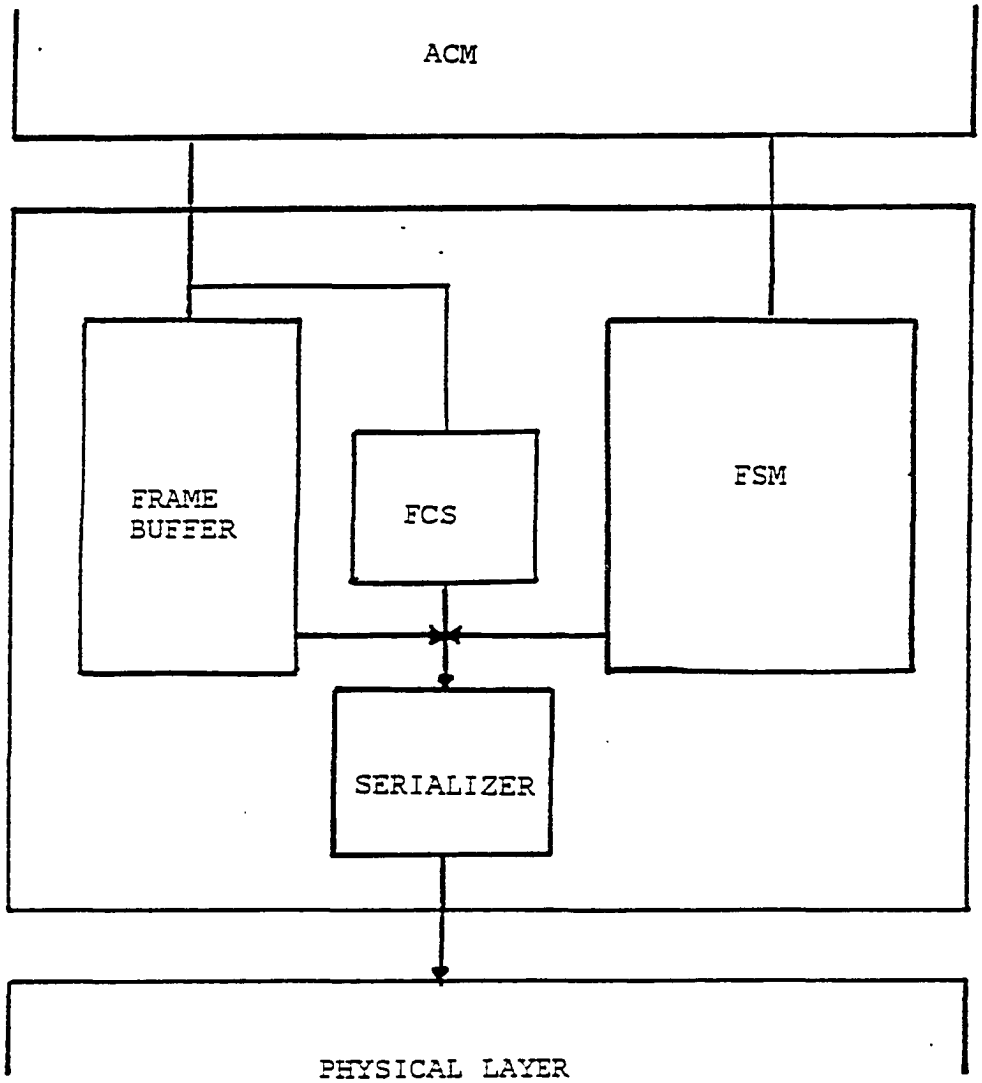


Figure 16 Transmitter block diagram

VI MEDIA DESCRIPTION

This chapter describes in detail the functions provided by the physical layer to the BIU. Section A describes the functions provided by the physical layer. The description of two types of media is included in section B.

Some of the information presented in this chapter was obtained from the IEEE 802.4 Token Passing Bus protocol specification (8). The byte wide bus was added as a new type of media. The close resemblance to the 802 standard was done to minimize the complexity. Designing a new type of physical layer is beyond the scope of this research.

A. Physical Layer Function

This section describes the services provided by the physical layer to the BIU. The two types of services provided by the physical are data service and network service. The data service functions include data transmission, data reception, and end of reception. The only network service request is to reset the physical layer.

1. Data service functions

The first data function provided by the physical layer is transmit symbol request. This request is used to transmit a symbol to the media. The only parameter to this

request is the symbol to be transmitted.

The second data function provided by the physical layer is "receive symbol indication." This function is used to receive a symbol from the media. The only parameter to this function call is the received symbol.

The last data function provided by the physical layer is used to indicate to the receiver when the end of a message was received. The receiver can then enter a state to detect the start of new data.

A summary of the data service functions is listed below.

```
Transmit symbol request
        PHY.request(symbol)

Receive symbol indication
        PHY.indication(symbol)

End of data notification
        PHY.end_data
```

2. Network service functions

The two network service functions provided by the physical layer are reset request and reset confirmation. The reset request function is used to reset the physical layer to a known state. The confirmation function is used to indicate the status of the reset request. The reset request has no parameters, and the confirmation returns the status of the reset request.

A summary of the network service functions is shown

below.

Physical layer reset
PHY.reset

Physical layer reset confirmation
PHY.confirmation(status)

B. Media Description

This sections provides a description of two possible types of media. The first type is a byte wide bus and is the preferred medium because of the ease of data encoding and the higher bandwidth. The second type is a phase coherent FSK coding on a 75 ohm coax cable. This method is taken directly from the IEEE 802 standard (8) and can be used if the distance between the processors is too large for the byte wide bus.

1. Byte wide bus

The byte wide bus consists of eight data lines, one strobe line, and a special symbol line. The transmitter should be either an open collector driver or a tri-state driver. The receiver should be designed to provide a minimum load to the bus.

The receive machine described in Chapter V should be modified for the byte wide bus. These modifications include the elimination of the deserializer and a simpler symbol interpreter. The FCS checker should be modified to handle

the data in bytes. Figure 17 shows a block diagram of the modified receive machine.

The transmit machine described in Chapter V should be modified by eliminating the serializer. The other machines described in Chapter V remain the same. A block diagram of the modified transmit machine is shown in Figure 18.

The different symbols which are generated by the physical layer are shown in Table 4.

Table 4 Symbol Definition

SYMBOL	DATA	STROBE	SPECIAL SYMBOL	
Silence	off	off	off	
Pad_idle	00H	High	High	
Data	XX	High	Low	
Start Delim	D8H	High	High	
End Delim	FEH	High	High	(More data)
End Delim	FCh	High	High	(End data)

2. Phase Coherent FSK

FSK (frequency shift keying) modulation is used on 75 ohm coax cable with passive taps. The medium will support a data rate of 10Mbits. FSK modulation consists of two frequencies to encode two logic levels (10). The logic level zero is one full cycle of a 10MHz signal and the logic level one is two full cycles of a 20MHz signal. The frequencies change at the zero crossings. Figure 19 shows an encoded waveform.

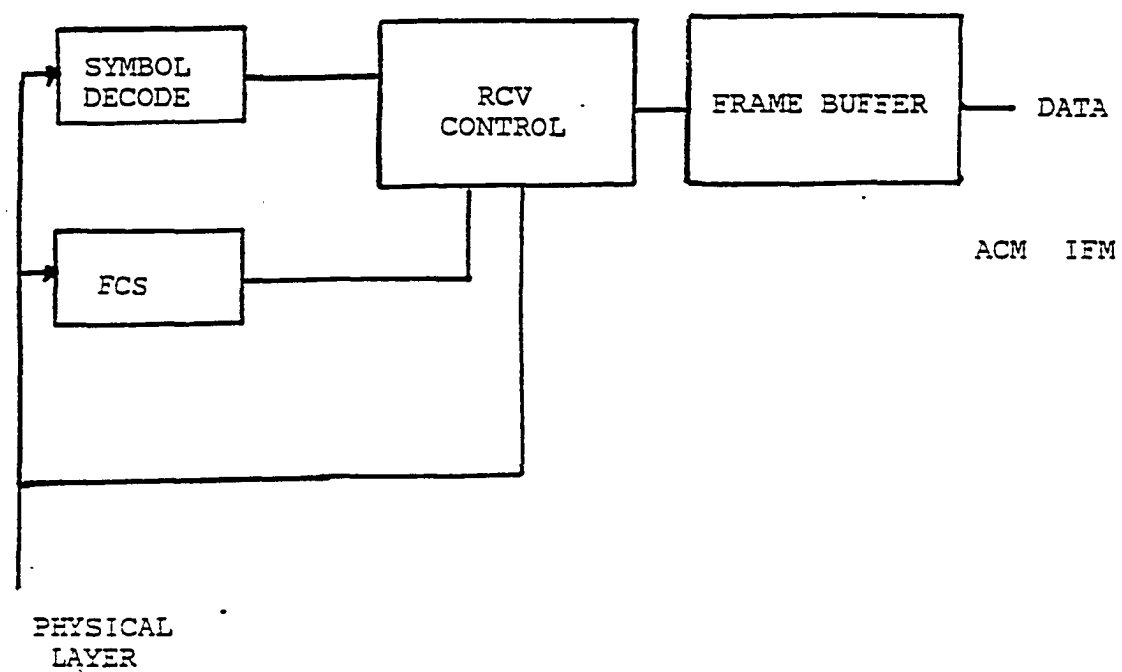


Figure 17 Modified receive machine

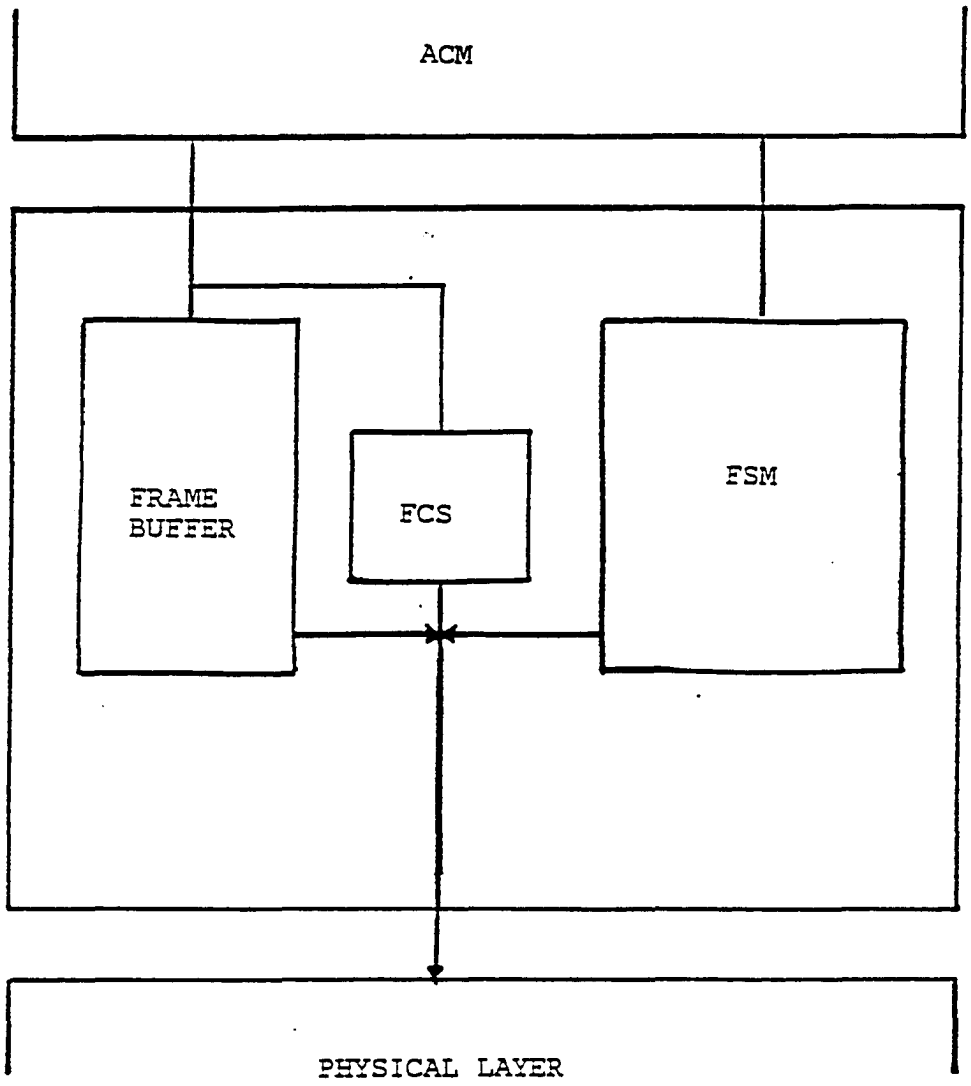


Figure 18 Modified transmit machine

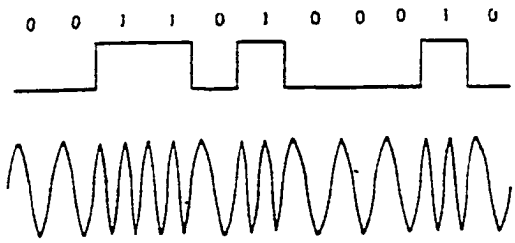


FIgure 19 Frequency Shift Keying Modulation

Each of the BIU symbols is encoded into a pair of physical symbols from the three different symbols (H), (L), and (off). The different BIU symbols are encoded as follows:

(1) Silence - (off,off)

(2) Pad_idle - (L,L)

(3) Zero - (H,H)

(4) One - (L,L)

(5) Non_data - (H,L) (L,H)

Non_data symbols are transmitted in pairs.

The start frame delimiter is encoded as follows

(H,L) (L,H) (H,H) (H,L) (L,H) (H,H) (H,H) (H,H)

The end frame delimiter is encoded as follows

(H,L) (L,H) (L,L) (H,L) (L,H) (L,L) (L,L) (H,H)

C. Conclusions

The byte wide bus is the preferred medium since the bandwidth is higher. The byte wide bus also requires less complicated transmitters and receivers. The bus could be implemented by using a standard backplane which is more cost effective than coaxial cable.

Coaxial cable has the advantage of passive taps, which eliminate the loading problems associated with the byte wide bus. Coaxial cable can be used over longer distances than the byte wide bus.

Fiber optic cable could be used as a medium. The taps present a problem with fiber. A nonactive tap is difficult to achieve in fiber with out signal attenuation. Active taps introduce failure points and require power from the station which then must be active at all times.

VII PERFORMANCE COMPARISON

This chapter will discuss the performance of the system as compared with the performance of the other systems shown in Chapter III. A discussion of the performance of the token bus protocol will also be contained in this chapter.

A. System Comparison

The proposed system is compared with the other methods shown in Chapter III. Both the byte wide bus and the phase coherent FSK media types are compared against the other methods. This section also includes a discussion of how the proposed system solves the problems of interconnection networks shown in Chapter II.

1. System measurement

Measures of performance, cost, and quality will be used to compare the networks. These measurements are described in a paper by E. Swartzlander and B. Gilbert (4). This paper deals with the first six networks discussed in Chapter III. The parameters for the multibus and hypercube networks shown in Chapter III are obtained from the literature (7). The parameters for the token bus network are obtained from Chapter V.

a. Network performance Parameters for network performance are media bandwidth and path length. Media bandwidth is the product of the number of active data links and the bandwidth of the individual links. Path length is the delay incurred in the transfer of the message from one node to another and back to the originating node. Table 5 shows the ideal network performance for all the networks.

The media in the proposed system (token bus) are capable of transferring data at a rate greater than the rate at which the processor can transfer data. Due to the increased speed of transfers by the BIU, the bandwidth of each bus is greater than one. The link bandwidth for the byte wide media is limited by the bus length. The FSK media do not have this problem and exhibits a higher link bandwidth at greater bus lengths.

Since differences of a factor of 2 to 4 are negligible, Table 5 shows that the token bus network offers the best performance while the single bus offers the worst. The other seven are comparable in performance. The multibus network is only comparable if the number of buses approaches the number of processors. The fully interconnected network has the same performance as the token bus system if the number of buses (B) times the link bandwidth (W) is equal to the number of processors.

The fully interconnected network only offers good performance if the cost of the connection is linear. This

assumption is only valid for a small number of processors. The fully interconnected network also provides more bandwidth than is needed by the system.

b. Network cost The cost or complexity of a network is the sum of the link cost and the switch cost. Both costs have been assigned equal weight. Table 6 shows the network costs for the nine networks. The bus, star, and ring offer the best cost. The token bus has a comparable network cost.

c. Network quality The network quality is a ratio of network performance to network cost. The network quality for the nine networks is shown in table 7.

d. Conclusions In all cases, the network quality decreases with an increasing number of processors, indicating that the networks shown here are better suited for a small number of processors. Figure 20 is a plot of the network quality of the networks versus the number of processors. This shows the token bus network is better for all sizes of networks.

Table 5 Network performance

NETWORK TYPE	NUMBER OF MESSAGES (M)	LINK BAND-WIDTH (B)	ROUND TRIP DELAY (D)	PERFORMANCE NP = MB/D
CROSSBAR	N	1/N	4	1/4
STAR	1	1	4	1/4
CUBE	N	1	2LOG N	N/2 LOG N
RING	N	1	N	1
BUS	1	1/N	2	1/2N
FULLY CONNECTED	N	1	2	N/2
HYPERCUBE	LOG N 2^{N-1}	1	2LOG N	2^{N-1}
MULTIBUS	B	1/N	2	B /2N
TOKEN BUS BYTE WIDE	B	W	2	WB / 2
TOKEN BUS FSK	B	W	2	WB / 2

Table 6 Network cost

NETWORK TYPE	LINKS (L)	COUNT (I)	TYPE (P T)	COMPLEXITY S = IPT	COST NC = L + S
CROSSBAR	2N	N^2	1:1	N^2	$N^2 + 2N$
STAR	2N	1 N	N:1 1:1	2N	4N
CUBE	N+N LOG N	(N LOG N)/2	2:2	2N LOG N	N + 3N LOG N
RING	N	N	2:2	4N	5N
BUS	1	N	1:1	N	N+1
FULLY CONNECTED	$(N^2 - N)/2$	N	1:N	N	$(3N^2 - N)/2$
HYPERCUBE	LOG N 2^{N-1}	N	1:8	8N	8N LOG N 2^{N-1}
MULTIBUS	B	BN	1:1	NB	B(N+1)
TOKEN BUS BYTE WIDE	B	BN	1:1	NB	B(N+1)
TOKEN BUS FSK	B	BN	1:1	NB	B(N+1)

Table 7 Network quality

NETWORK TYPE	PERFORMANCE (NP)	COST (NC)	QUALITY NQ = NP/NC
CROSSBAR	1/4	$N^2 + 2N$	$1/(4N^2 + 8N)$
STAR	1/4	4N	1/16N
CUBE	N/2 LOG N	N + 3N LOG N	1/(2 N LOG N)(1+3 LOG N)
RING	1	5N	1/5N
BUS	1/2N	N+1	$1/(2N^2 + 2N)$
FULLY CONNECTED	N/2	$(3N^2 - N)/2$	1/(3N -1)
HYPERCUBE	2^{N-1}	$8N \text{ LOG } N\ 2^{N-1}$	1/(8N LOG N)
MULTIBUS	B /2N	B(N+1)	$1/(2N^2 + 2N)$
TOKEN BUS BYTE WIDE	WB /2	B(N+1)	W/2(N + 1)
TOKEN BUS FSK	WB /2	B(N+1)	W/2(N + 1)

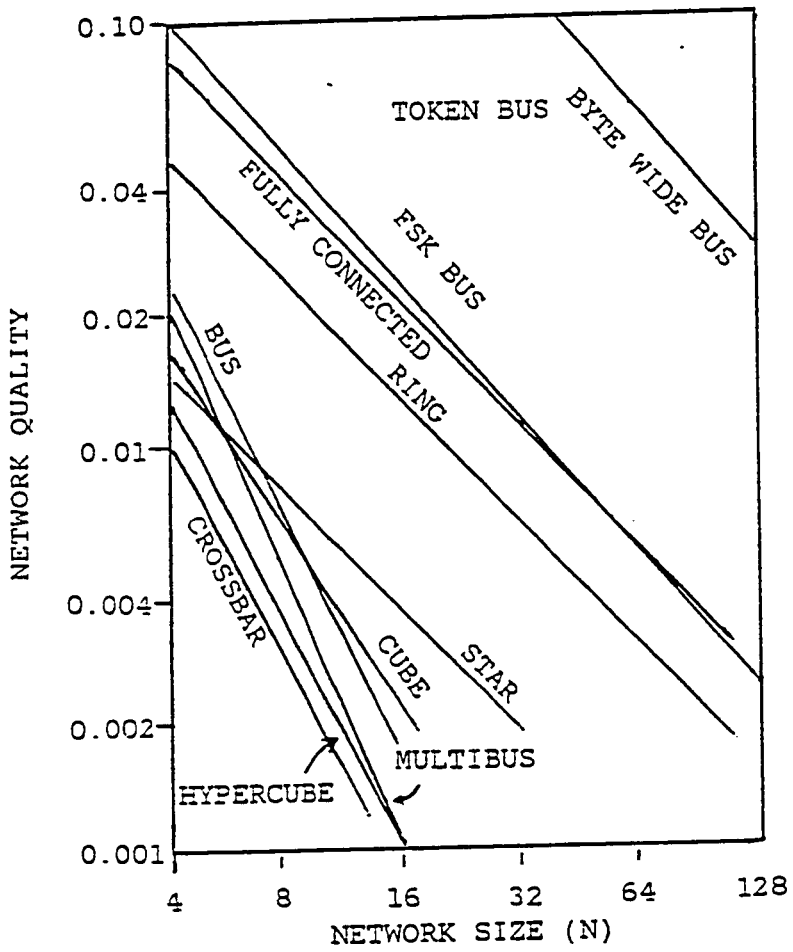


Figure 20 Network quality

2. Problem solutions

a. Number of connections The number of connections on each processor is equal to the number of buses. Some of the seldom used processors can have fewer connections. The token bus has fewer connections than the fully connected network. However, the token bus also has more connections than the other networks. The larger number of connections is not much of a problem since the number of connections is less than the number of processors. The connections are also simple in the case of the byte wide bus.

b. Number of lines The number of lines is a function of the number of buses. In the FSK method, the number of lines is equal to the number of buses. In the byte wide method, the number of lines is ten times the number of buses. Due the bus configuration of the byte wide method, the number of lines is not a large problem. The byte wide buses can be placed in a simple backplane, which help eliminate some of the problems with a large number of lines.

c. Complex switching This system has no switching since the BIU is connected the buses at all times and can use any bus.

d. Complex routing The routing used in the system is handled by the BIU and the token bus protocol. The token bus protocol handles most of the routing problems.

The processor will include the destination address along with any message and the BIU will route the message on the proper bus and to the correct processor.

e. Bandwidth limitations The bandwidth on the byte wide bus is limited by the length of the bus. This limit at a 10 Mb date rate is about 10 to 20 feet. The FSK bus can be used over a distance of about 300 to 500 feet.

f. Broadcasting The token bus system is capable of broadcasting messages to all processors or any group of processors.

g. Conclusions The token bus system has solved the problems of routing, switching, broadcasting, and the bandwidth limitation. The problems of number of connections and number of lines has been minimized. The problem with the possible large number of lines in the byte wide method has been minimized due to the bus structure of the lines.

The token bus structure addresses all the problems discussed in Chapter II and provides a solution to each problem. Since the token bus also has good performance characteristics, it is a better interconnection network for a large number of processors.

B. Protocol Performance

This section will address the performance of the token bus protocol. The performance measurements are obtained

from the literature and are to be used as a rough measurement of the throughput on a single bus. The system throughput can be estimated from the throughput on a single bus. A paper by William Stallings (11) presents a performance model for a token bus protocol. Stallings also defines the bus utilization for the token bus protocol. A paper by W Bux (12) shows the effect of various load patterns on the token bus throughput.

1. Bus utilization

In order to determine the bus utilization, the effect of the transmission rate and the propagation delay on the network must be studied. The product of the transmission rate (R) and the propagation delay (D) is the single most important parameter for determining the network performance. Other things being equal, network performance will remain the same. The product of delay time and data rate is equal to the length of the transmission medium in bits.

Stallings defines the length of the medium in bits compared to the packet length as:

$$a = \frac{\text{Length of Data Path in Bits}}{\text{Length of the Packet}}$$

$$a = \frac{\text{Propagation Time}}{\text{Transmission Time}}$$

$$a = \frac{R\ D}{V\ L}$$

where
R = Data Rate
D = Distance
V = Velocity 2×10^8 m/s
L = Packet Length

Typical values for a range from about 0.01 to 0.1. Table 8 gives some example of the factor a for the bus topology. The parameter a gives an upper bound on the utilization of the network.

Table 9 Values of a

Data Rate	Packet Size	Cable Length	a
1 Mb/s	100 bits	1 km	0.05
1 Mb/s	1000 bits	10 km	0.05
1 Mb/s	100 bits	10 km	0.5
10 Mb/s	100 bits	1 km	0.5
10 Mb/s	1000 bits	1 km	0.05
10 Mb/s	1000 bits	10 km	0.5
10 Mb/s	10000 bits	10 km	0.05
50 Mb/s	10000 bits	1 km	0.025
50 Mb/s	100 bits	1 km	2.5

The utilization of the network can be expressed as a function of a, as shown below:

Utilization (U) = Throughput / Data Rate (R)

$$U = \frac{L\ /\ (\text{propagation} + \text{transmission time})}{R}$$

$$U = \frac{L / (D + L/R)}{R}$$

$$U = \frac{1}{1 + a}$$

Figure 21 shows the throughput of the network versus the offered load to the network as a function ofthe factor a. The ideal case is when a=0, which allows 100% utilization of the network. Stallings shows the upper bound on the utilization of the network is 1 / (1+a), regardless of the medium access protocol used. It is desirable to minimize the factor a for a given network.

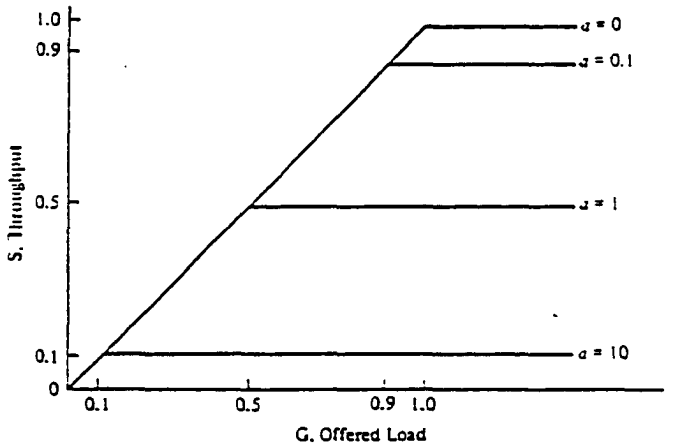


Figure 21 Throughput versus offered load

2. Performance model

The performance model presented by Stallings defines several variables. These variables that are listed include

the number of active stations (N) and the system throughput (S). The throughput for the token bus network can be defined as:

$$S = \frac{T1}{T1 + T2}$$

where,

T1 = average time to transmit a data packet
T2 = average time to pass the token

Stallings derives the throughput as a function of the number of stations (N) and of (a). This derivation leads to the following equation:

$$S = \begin{cases} \dfrac{1}{1 + a/N} & a < 1 \\ \dfrac{1}{a(1 + 1/N)} & a > 1 \end{cases}$$

Figure 22 shows the throughput as a function of (a) for the token passing network and for the CSMA/CD network. Figure 23 shows the throughput as a function of N for the token passing network and for the CSMA/CD network. As shown in Figure 22 the throughput drops as (a) increases. For a small value of (a) the throughput is one. As the number of stations increases, the throughput increases. This is caused by a higher percentage of data packets being sent.

The CSMA/CD network throughput decreases as the number of stations increases. This decrease in throughput is due to the increased number of collisions.

For the proposed system shown in Chapter V the values for (a) and U can be estimated as follows. The length of the transmission path is assumed to be 10 meters and the maximum packet length is 8196 bytes.

$$a = \frac{80 \times 10^{6} \ (10)}{2 \times 10^{8} \ (8196)} \quad \text{byte wide bus}$$

$$a = \frac{10 \times 10^{6} \ (10)}{2 \times 10^{8} \ (8196)} \quad \text{FSK bus}$$

$a = 0.00048$ for byte wide bus

$a = 0.00006$ for FSK bus

In both types of buses, the utilization is approximately one. The system throughput for both types of buses is also one.

The number of buses needed in the system shown in Chapter V can easily be determined since the throughput for each bus is approximately one. The number of buses is the desired system throughput divided by the data rate of each bus.

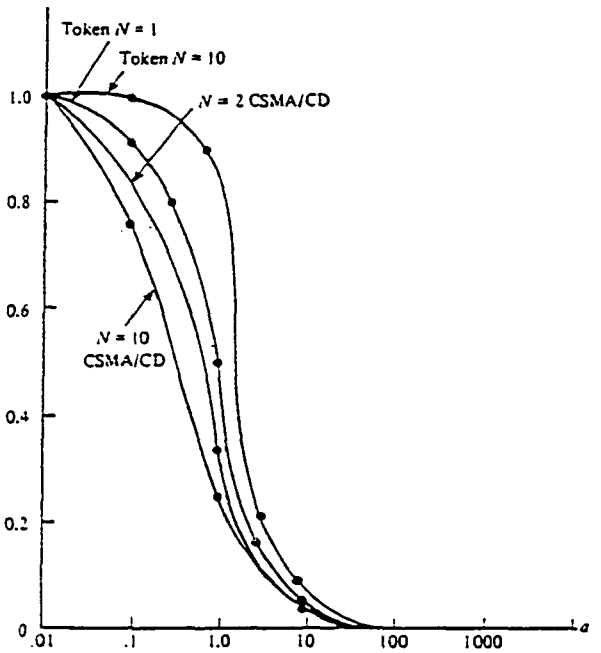


Figure 22 Throughput as a function of a

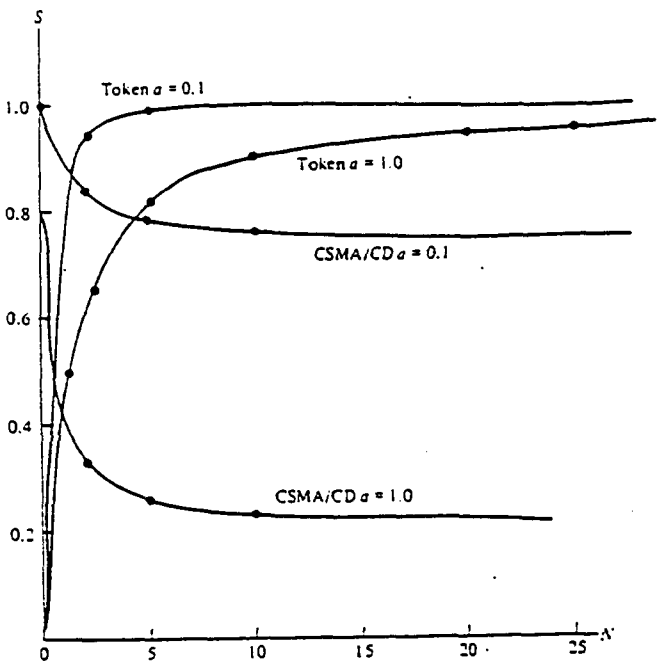


Figure 23 Throughput as a function of N

3. Load patterns

Five figures 24-28 showing the effects of various load patterns and station activity on the performance of the token bus are included in this section. The first four figures are obtained from Stallings (12). The next figure is obtained from Bux (13).

Figures 24-27 show the maximum potential data rates for the token bus, token ring, and the CSMA/CD networks over various station loads. These figures show the token bus and token ring provide the same data for the same data load. The CSMA/CD network shows a much lower data than the token networks at high data loads.

Figure 28 shows the throughput as a function of station traffic patterns. Traffic pattern one is a totally symmetrical case. Traffic pattern two is where two stations each generate 40 percent of the load and the rest of the traffic is from the other stations. Traffic pattern three is where one station generates 80 percent of the load and the rest of the traffic is from the other stations. This figure shows that with increasing the asymmetry of the traffic, the average delay decreases slightly.

C. Conclusions

This chapter provided several methods of measuring the performance of the interconnection network shown in Chapter

V. The first set of measurements were used in Chapter III to compare the various networks shown in the literature. The token bus interconnection network showed a better network performance than most of the other networks. The network quality of the token bus was higher than the other networks for sizes of networks.

The problems with interconnection networks shown in Chapter II were addressed in this chapter. The token bus network solved the problems presented in Chapter II. The only problem which was not completely solved was the number of connections per processor. The processor only has one connection to the BIU, but the BIU has one connection for each bus in the network. This is only a problem when the number of buses increases.

The performance of the token bus protocol was also discussed in this Chapter. The performance of the token bus network was shown to be better than the performance of a CSMA/CD network. The throughput for each bus in the token bus interconnection network is equal to one. This allows for maximum utilization of the buses in the network.

All the measures show the token bus system described in chapter V is a better interconnection system.

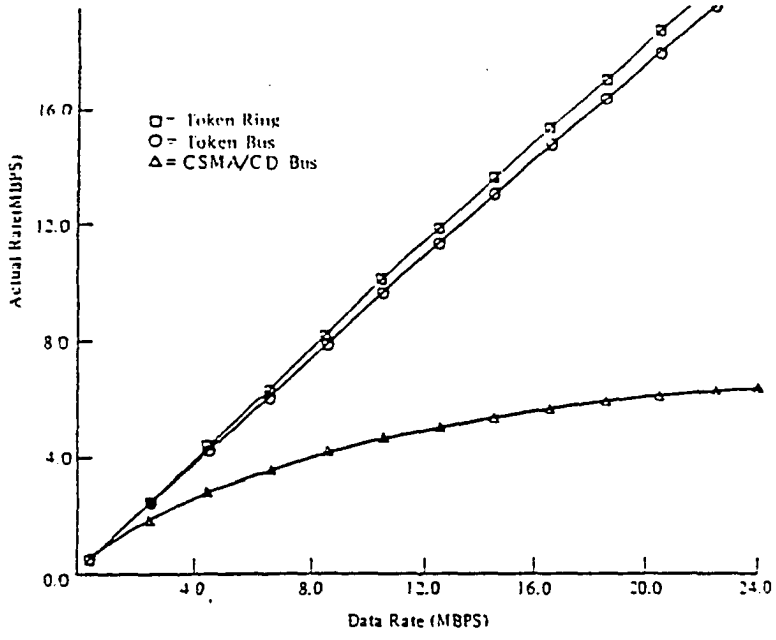


Figure 24 Data Rate for 2000 bits per packet

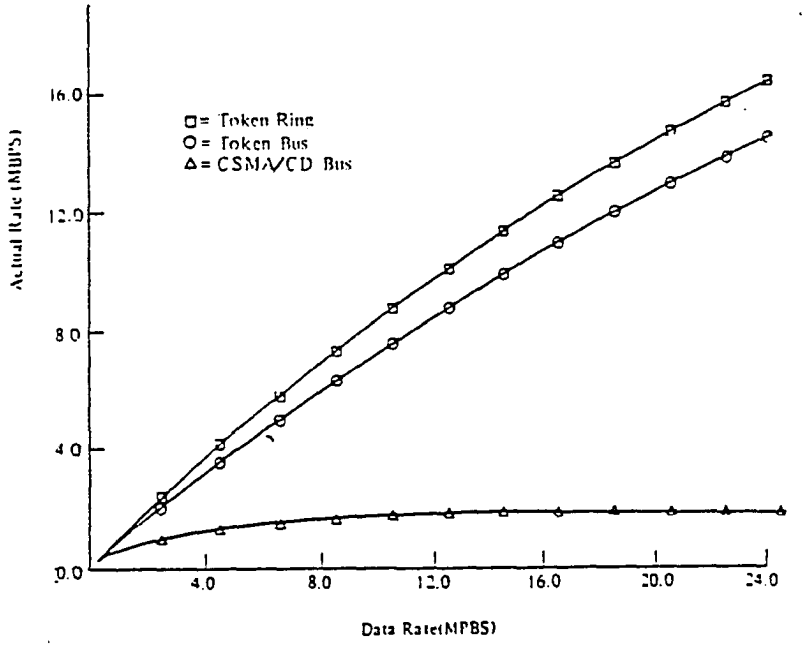


Figure 25 Data Rate for 500 bits per packet

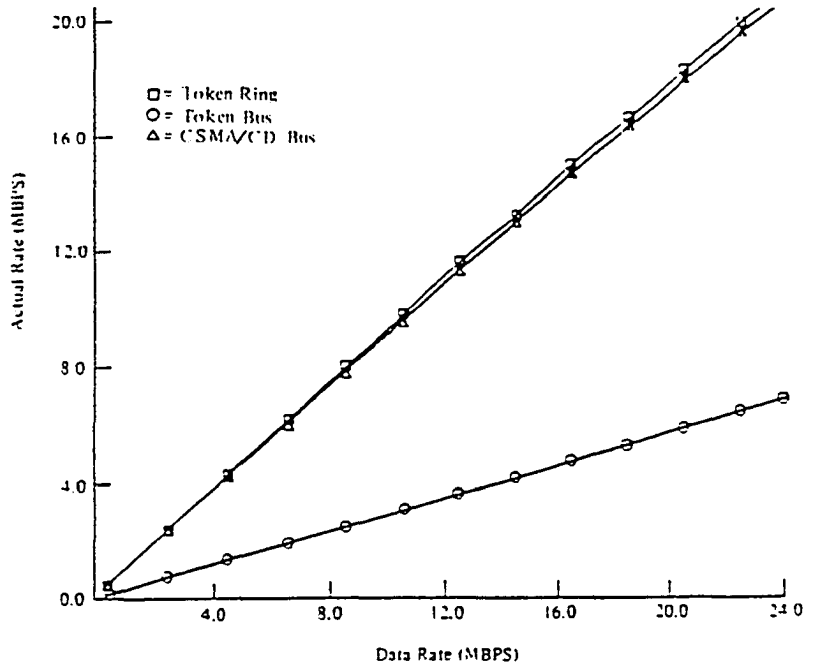


Figure 26 Data Rate for 2000 bits per packet

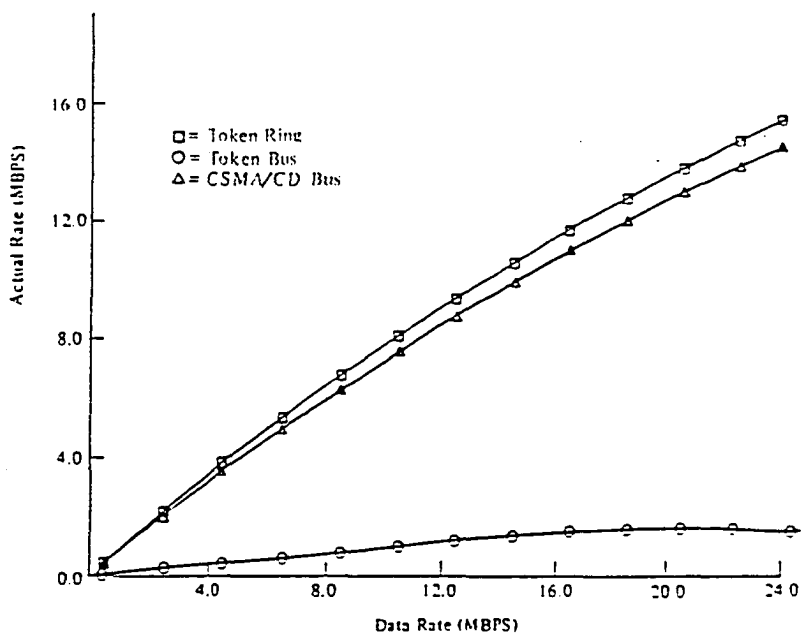


Figure 27 Data Rate for 500 bits per packet

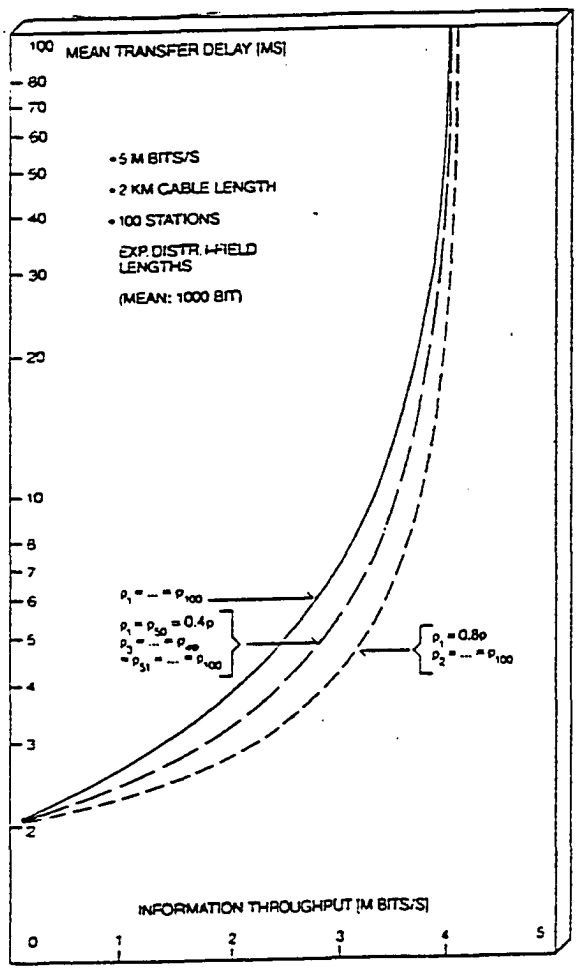


Figure 28 Throughput versus traffic patterns

VIII CONCLUSIONS

As discussed in Chapter I, there is a need for the development of an interconnection scheme to connect a large number of processors. The schemes found in the literature do not work well when the number of processors is large; therefore, a new method needs to be developed to connect these processors.

Chapter II shows that there are several problems that must be overcome in the interconnection of multiple processors. Each of the eight schemes discussed in Chapter III solved one or more, but not all the problems.

A system which uses multiple buses connected to each processor with a bus interface unit was discussed in Chapter IV. The bus interface unit will handle all the routing and bus arbitration between the processors and the buses. A token bus bus protocol was suggested for the system, and was a variation of a standard protocol.

The protocol for the token bus interconnection network was described in Chapter V. The block diagram of the various components which are in the BIU were also shown in Chapter V.

The communication media for the proposed system needed to have a high bandwidth to minimize the maximum latency. The exact media were specified in Chapter VI. Two possible media were chosen for the token bus system. The first type

was a byte wide bus with a strobe line. The other type of medium was a coaxial cable using a frequency shift keying encoding technique. The byte wide bus is used for high bandwidth short distance connections. The coaxial media is used for longer distance connections.

Both the byte wide bus and the FSK bus use a 10 Mhz clock rate. The byte wide bus transmits the data a byte at a time thus achieving an 80 Mbit data rate. The transmitter and receivers are simpler for the byte wide bus since the bus uses digital signal levels. The transmitter and receivers do not need to convert the data from 8 bit data to serial data and back again. The transmitters and receivers for the FSK bus need to convert the digital signal to an analog signal and back again.

Chapter VII shows the performance of the token bus interconnection network is superior to the performance of the other networks using the measurement techniques described in Chapter III. The new network also solved the problems with the other networks that were described in Chapter II. The performance of the token bus protocol was detailed in Chapter VII. The token bus interconnection network exhibits a 100 percent utilization of the buses. The 100 percent utilization allows the network throughput to be increased as the technology increases.

The next medium likely will be a fiber optic one. With fiber optics the number of buses needed could be reduced

while the system throughput remained constant.

This research provided several contributions to the field. The first contribution was the application of local area network techniques to an interconnection network for tightly-coupled multiprocessor systems. A second contribution was the adaptation of a token bus protocol to a multiple bus system with each bus operating independently of the other buses. Communication activity was made transparent to the processor through the introduction of a bus interface unit. A new simpler medium was defined for the token bus to take advantage of the topology of a tightly-coupled multiprocessor system. The research demonstrated how multiprocessor systems can be substantially improved over those currently described in the literature.

The next step to be taken in this research is to implement the network on a small scale. The protocol will also be refined and simulated. This simulation will be done in conjunction with the implementation of the network.

The fault tolerant aspects of the network will be pursued as a continuing research topic. This research could lead to a high performance fault tolerant network for general communications.

BIBLIOGRAPHY

1. Blakeslee, Thomas R. Digital Design with Standard MSI and LSI. New York: John Wiley and Sons, 1979.

2. Marson, Marco A., Balbo, Gianfranco, and Conte, Gianni "Comparative Performance Analysis of Single Bus Multiprocessor Architectures." IEEE Transactions of Computers 31 (December 1982): 1179-1191.

3. Intel Corporation. "A New Direction in Scientific Computing." Intel Scientific Computers, Portland, OR, 1985.

4. Swartzlander, Earl E.; and Gilbert, Barry K. "Supersystems: Technology and Architecture." IEEE Transactions on Computers 31 (May 1982): 399-409.

5. Tanenbaum, Andrew S. Computer Networks. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

6. Lang, Tomas; Valero, Mateo; and Alegro, Ignacio "Bandwidth of Crossbar and Multiple-Bus Connections of Multiprocessors." IEEE Transactions on Computers 31 (December 1982): 1227-1234.

7. White, Larry D. "Communication Structures for Large Networks of Microcomputers" IEEE Transactions of Computers 30 (April 1981): 264-273.

8. IEEE Standards Board. IEEE Standards for Local Area Networks: Token-Passing Bus Access Method and Physical Layer Specifications. New York: IEEE, 1985.

9. Stallings, William. Local Networks an Introduction. New York: Macmillan, 1984.

10. Stallings, William. Data and Computer Communications. New York: Macmillan, 1985.

11. Stallings, William. "Local Network Performance." IEEE Communications Magazine 22 (February 1984): 27-36.

12. Bux, Werner. "Performance Issues in Local Networks." IBM Systems Journal 23 (April 1984): 351-374.

X ACKNOWLEDGMENTS

The author would like to thank the following people for making this dissertation. First, Dr. Art Pohm, the author's major professor, for his guidance and technical support. Second, Dr. Clair Maple for his technical and financial support. Third, Dr. Jim Davis for his technical and moral support. Finally, to my wife Gwenna for her understanding and support which is needed to complete a dissertation.