1987

Performance analysis of token bus protocol with maintenance functions

Joon-Nyun Kim
Iowa State University

Performance analysis of token bus protocol with maintenance functions

by

Joon-Nyun Kim

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

1987

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Local Area Network Overview

During the past decade, local area networks have become one of the most publicized and controversial topics in the data communications field, that have produced numerous publications [1-8] dealing with various aspects of local area networks. The publicity surrounding local area networks stems from what they are purported to do for an organization; the controversy comes from how they are to do it. Due to the versatility, however, local area networks will play a prominent role in computer communications and distributed processing in the 1980s.

Basically, a local area network is an interconnected set of computers which is geographically limited from a distance of several thousand feet to a few miles, and is structured around a high-speed, low-noise connecting link or channel that typically supports data rate of 500 Kbps to 50 Mbps.

As illustrated in Figure 1, there are three basic topologies in local area networks: the bus, the ring, and the star. The bus topology is appropriate for transmission medium such as coaxial cable which allows high-impedance taps. In principle, these taps do not affect the medium and a large number of stations can be connected. This topology is particularly suitable for the random accessing techniques.

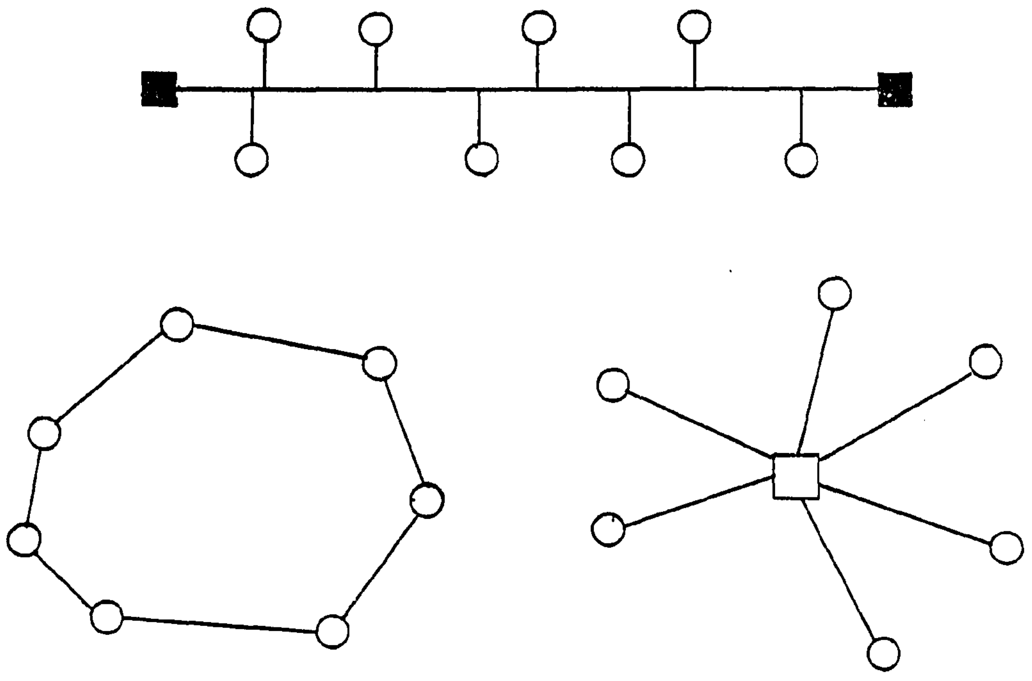


FIGURE 1. Bus, ring, and star configurations

The ring topology is a sequence of point-to-point links with flow of data in one direction around the ring. In this case, there is a delay due to data processing at each station and for reliability reason, there are provisions to bypass stations if they become inoperative. In both cases of bus and ring, the control of traffic is distributed among stations.

In the star topology, the control is concentrated in a central hub where all the data are collected and routed to appropriate stations through high-speed channels. In contrast to this, data are packetized and all packets traverse the same channel in most bus and ring systems, and the path of a given packet carries it past all stations on the network. Thus, although addressing information is still required, routing is unnecessary.

The main motivation for local area networking is resource sharing. Resources include not only devices such as computer and printer, but also data files which are costly to duplicate and expensive to maintain. It is often expedient to maintain central files or data banks that are shared by devices at work stations throughout a local area network.

In addition to the primary justification for local area networks, there are a number of secondary effects. For example, the distributed resources of a local network provide redundancy for many devices and thus backup in the event of failures. Also, many local area networks provide speed and code conversion that enables equipment from different manufacturers to be connected easily without expensive special purpose interfaces.

Performance Issue in Local Area Networks

As the interest grows in local area networks, the performance analysis of these networks has become one of the main topics in local network study [9-29]. Network controls, architectures, and protocols are other topics in local area network.

Physically a local area network consists of a communication channel to which a number of intelligent stations are coupled. The intelligent stations have several functions; they must transmit and receive signals on the channel; they must control access to the channel; and they must interface the device coupled to the network at the station.

Standards are being developed for a layered architectural structure [30-32] with three layers for covering the communication aspects of the network. These layers are identified as the physical layer, the medium access layer, and the logical link layers as depicted in Figure 2.

The physical layer deals with generating physical signals and transmitting them over the channel. The media access layer manages access to the channel, determining which station can transmit at any given time. The logical link layer operates with bits grouped into packets and

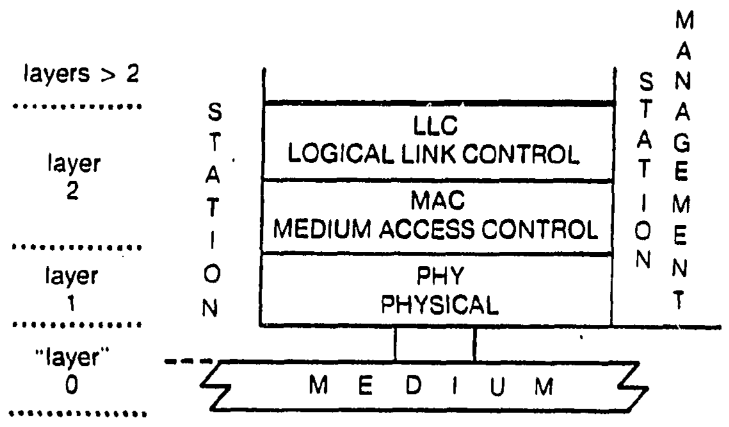


FIGURE 2. Layer structure of LAN model [31]

manages those functions associated with activities such as error control.

Physical layer functions are largely carried out by hardware with some support from software, which usually runs on microprocessor equipment at the stations. The medium access functions tend to use software with a little support from hardware. The logical link control functions are mostly carried out by software.

Performance analysis of local area network has come to mean the development and study of mathematical models that predict the performance of networks in some well-defined sense. Performance analysis is performed after the fact.

That is to say that various common local network designs are taken as given and then analyzed to determine their performance.

Performance analysis is often carried out in conjunction with the design of network architecture. Usually several designs and their performance are considered and compared. The particular media access technique chosen tends to dictate several different implementations that may have different performance models and thus have different performance.

The main reasons for performance analysis are to improve productivity so that more work is accomplished in the same amount of time and to add functionality to the system. One thing that should be addressed here is that the computer system evaluation cannot be based solely on the performance analysis. It should be accompanied by other factors such as economic issues and situational considerations, even though performance analysis still offers firsthand knowledge of the system behavior.

Problem Definitions and Research Objectives

The target system of this research is the token bus protocol. In dealing with the performance analysis of the token bus protocol, there have been many different assumptions and models. Most of these studies, however, deal with token bus protocol in normal operating mode. In other words, most of the analyses are done with the assumption that the network is working properly and no faulty conditions are occurring. This assumption is valid if the system under investigation is relatively small and simple. But there are some situations where the collapse of the logical ring of the token bus protocol causes catastrophic results. This brings some attention to the ring maintenance functions which provide various house keeping work for logical ring and to the study of the performance degradation due to these maintenance functions.

There are all kinds of maintenance functions conceivable in token bus protocol, but some functions are essential to the system and they are listed below:

- Ring initialization: When the network is started up or after the logical ring has broken down, it must be reinitialized. Some cooperative, decentralized algorithm is needed to sort out who goes first, who goes second, and so on.
- Station addition: Periodically, nonparticipating stations must be granted the opportunity to insert themselves into the ring.

- Station deletion: A station can voluntarily remove itself from the ring by splicing its predecessor and successor together.
- Token recovery: The token can be lost during normal operation and the protocol goes through steps to recover the token. And in case of multiple tokens in the network, two or more stations think it's their turn to transmit data and cause bus error. The protocol needs to be prepared for the situations like this.
- Receiver/Transmitter fault recovery: Each station has an algorithm to resolve the problems caused by faulty receiver or transmitter.

Even though there have been several research efforts in the area of token bus performance analysis, the effects of maintenance functions on the network performance have been mostly overlooked. When the actual system is put into operation, it is not meaningful to estimate the performance of the system until maintenance functions are considered, because these functions may cause considerable degradation in network performance. In spite of some research work in token bus performance analysis, following problems still remain unsolved:

- There are no clear cut definitions for maintenance functions in the token bus protocol.
- The quantifications of the overhead associated with maintenance functions are done very poorly and thus it is very difficult to reflect the effects of maintenances on network performance
- No analytic model has been developed to encompass the maintenance functions in token bus protocol.

The main objective of this research is to develop an analytic model for the token bus protocol with consideration of maintenance functions. To achieve the objective, following goals have been set:

- Classify maintenance functions in the token bus protocol and select important functions for analysis.
- Develop an analytic model which has the flexibility to cover the maintenance functions.
- Quantify maintenance functions mathematically for the convenience of analysis.
- Develop a scenario which clearly shows the good and bad behavior of the network.

Outline of Dissertation

Research motivation, problem definitions, and the objective of the research are introduced in Chapter 1 after a brief review of the local area networks. Chapter 2 includes description of the token bus protocol basic operation and Chapter 3 describes maintenance functions of the token bus protocol. A basic analytic model is developed in Chapter 4 and modified model for maintenance functions are also discussed. In Chapter 5, sample studies of the token bus system are presented with the worst case analysis based on the model developed in Chapter 4. Some plots are also supplied to show the effects of maintenance functions on network performance. Finally, conclusions and some suggested future work are discussed in the last chapter, Chapter 6.

CHAPTER 2. TOKEN BUS PROTOCOL DESCRIPTION

Since the standardization of the token bus protocol by IEEE [31], it has been a good subject for performance analysis. Before describing the analysis, it would be appropriate to introduce the token bus protocol in brief. The basic operations and maintenance functions of the token bus protocol are described in the following sections.

Basic Operation of the Token Bus Protocol

Under steady-state conditions, operation of a token bus consists of alternating data transfer and token transfer phase. When a station finishes transmitting its data or when its access time expires, it sends a control token frame to the next station in the logical ring. In Figure 3, note that the token medium access method is always sequential in a logical sense. That means the right to access the medium passes from station to station.

Furthermore, note that the physical connectivity has little impact on the order of the logical ring and that stations can respond to a query from the token holder even without being part of the logical ring. During its access time, the token holder may transmit one or more frames and may poll other stations and receive responses. Control in a token bus is, however, completely distributed anu when a

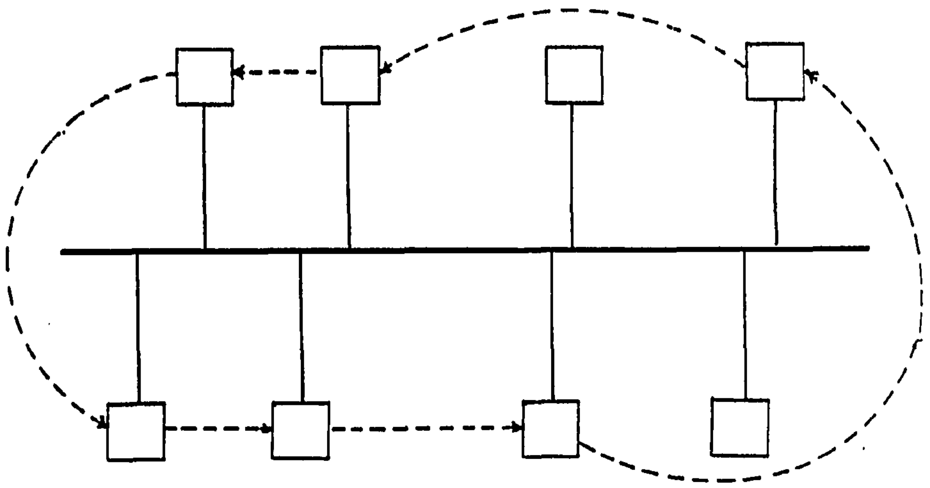


FIGURE 3. Logical ring on physical bus

station has used its allocated time, it must relinquish control of the bus.

Whereas operation of a token bus generally consists of the transmission of the data frame and token frame, at periodic intervals the network goes through a contention process in which stations that were not participating in the logical ring up to that time have the opportunity to join the ring. This contention process is controlled with response_window.

In addition to the ability to join the ring, any station can also remove itself from the ring by splicing its predecessor and successor stations together. This is done at the time token is received: the predecessor station is informed of the successor station's address and it is asked that this address be its revised successor station address.

Each station is assigned a maximum token holding time, which ensures that no station monopolizes the ring and that the access time is always finite. Access to the bus can also be prioritized by providing up to four different classes of service. This is accomplished by placing token rotation timers at each station for each class of service with priority less than the highest priority. The objective of the priority system is to allocate network resources by assigning amounts of network time to each class of service. The highest priority class of data is transmitted first and after its time is up, lower priority classes are served in a systematic manner provided time is available.

Frame Formats of Token Bus Protocol

The basic frame format specified at the medium access control level is shown in Figure 4. The number of octets between Start delimiter and End delimiter, exclusive, should be 8191 or fewer.

PREAMBLE	SD	FC	DA	SA	DATA_UNIT	FCS	ED

SD:	Start delimiter
FC:	Frame Control
DA:	Destination address
SA:	Source address
FCS:	Frame check sequence
ED:	End delimiter

FIGURE 4. Frame format for token bus protocol

Preamble

The preamble pattern precedes every transmitted frame. Preamble is primarily used by the receiving modem to acquire signal level and phase lock by using a known pattern. The preamble pattern is chosen for each modulation scheme and data rate for this purpose. A secondary purpose for the preamble is to guarantee a minimum ED to SD time period to allow stations to process the frame previously received. It is required that the duration of the preamble must be at least 2 μsecond regardless of data rate.

Start delimiter

The frame structure requires a start delimiter which indicates the beginning of the frame. The start delimiter consists of signaling patterns that are always distinguishable from data.

Frame control

The frame control field determines what class of frame is being sent among the following general categories:

- MAC control
- LLC data
- Station management data
- Special purpose data

Address field

Address field consists of destination address and source address. Addresses are either 16 bits or 48 bits in length and all addresses on a given network must be of the same length. Addresses are bit strings which serve as unique station identifiers or group identifiers. Additionally, the address bits are used in determining delays in the contention process and transmission lengths in the token claiming process.

Data unit

Depending on the bit pattern specified in the frame control field, data unit field can contain one of the three following information:

- An LLC protocol data unit which is used to exchange LLC information between LLC entities.

- A MAC management data frame which is used to exchange MAC management information between MAC management entities
- A value specific to one of the MAC control frames

<u>Frame check sequence</u>

The frame check sequence is a 32 bit frame checking sequence based on the standard generator polynomial of degree 32.

<u>End delimiter</u>

The frame structure requires an end delimiter which mark the end of the frame and determines the position of the frame check sequence. All bits between the SD and the ED are covered by the FCS. The end delimiter consists of signaling patterns that are always distinguishable from data.

Some of the important access control frame formats are given in Appendix A for reference.

MAC Layer Operation

This section describes the token bus medium access control (MAC) layer's operational and exception recovery functions. The MAC layer lies between logical link control and physical layer, and plays vital role in token bus protocol.

MAC layer functions

Specific responsibilities of the medium access control layer for a broadcast medium involve managing ordered access to the medium, providing a means for admission and deletion of stations, and handling fault recovery.

The faults are those caused by communications errors or station failures. These faults include:

- Multiple tokens
- Lost token
- Token pass failure
- Receiver/Transmitter failure
- Duplicate station address

This medium access protocol is intended to be robust in the sense that it should tolerate and survive multiple concurrent errors.

Basic operations

When the network is in steady state, a logical ring has been established and no error conditions are present. It simply requires the sending of the token to a specific successor station as each station finishes transmitting.

Other essential and more difficult tasks are establishment of the logical ring at initialization or re-establishing it in the case of catastrophic error and maintenance of logical ring allowing stations to enter and leave the ring without disrupting the other stations in the network.

Right to transmit The token, which represents the right to transmit, is passed station to station in descending numerical order of station address. When a station hears a token frame addressed to itself, it has the token and may transmit data frames. When a station has completed transmitting data frames, it passes the token to the next station in the logical ring.

Token passing After each station has completed transmitting any data frames it may have, and has completed other maintenance functions, the station passes the token to its successor by sending a token MAC control frame.

After sending the token frame, the station listens for evidence that its successor has heard the token frame and is active. If the sender hears a valid frame following the token, it assumes that its successor has the token and is transmitting. If the token sender does not hear a valid frame following its token pass, it attempts to access the state of the network.

If the token sending station hears a noise burst or frame with an incorrect FCS, it cannot be sure from the source address which station sent the frame. If a noise burst is heard, the token sending station continues to listen in the check_token_pass state for up to four more slot_times. If nothing more is heard, the station assumes that it heard its own token that has been garbled and so repeats the token transmission. If anything is heard during the following four slot_time delay, the station assumes its successor has the token.

In short, if the token holder does not hear a valid frame after sending the token the first time, it repeats the token pass operation once more performing the same monitoring as during the first attempt. If the success does not transmit after a second token frame, the sender assumes that its successor has failed and goes further into recovery procedures that grow drastic as the station repeatedly fails to find a successor station.

MAC Layer Internal Structure

The MAC layer performs several functions which are loosely coupled. Figure 5 shows functional partitioning of the MAC layer which has five asynchronous logical machines that handle MAC functions.

Interface machine

This machine acts as an interface and buffer between the LLC and MAC layer, and between station management and MAC layer. It interprets all incoming data and other

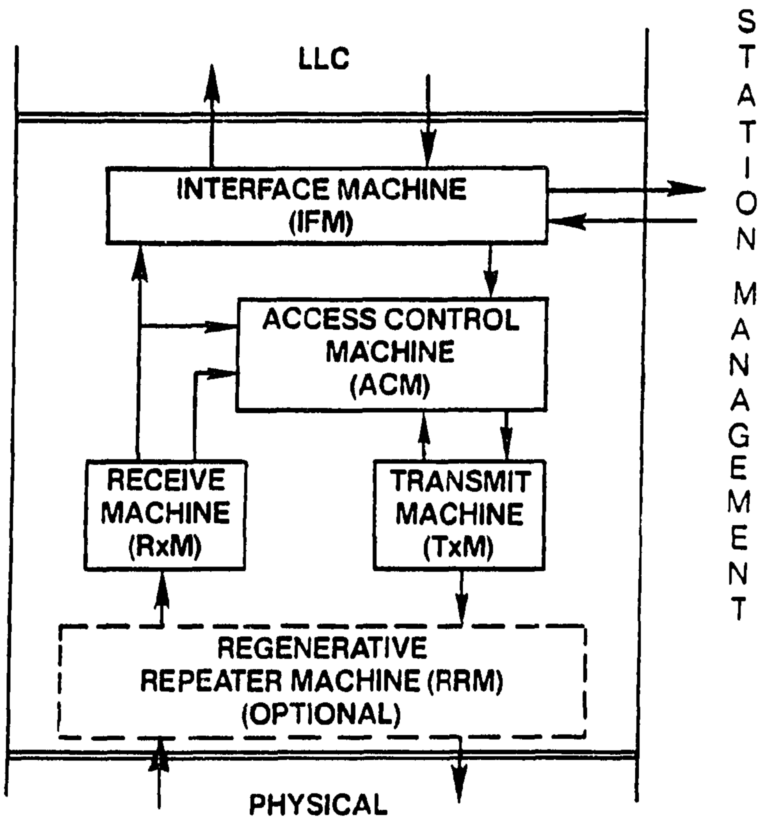


FIGURE 5. MAC layer functional partitioning

service primitives and generates appropriate outgoing service primitives.

This machine handles the mapping of quality of service parameters from the LLC view to the MAC view where this is necessary. It handles queueing of service requests and performs the address recognition function on received LLC frames, accepting only those addressed to this station.

Access control machine

This machine cooperates with the access control machines of all other stations on the bus in handling the token to control transmission access to the shared bus. The ACM is also responsible for initialization and maintenance of the logical ring, including the admission of new stations. Finally, it has responsibility for the detection of and recovery from faults and failures in token bus network.

Receive machine

Receive machine accepts atomic symbol inputs from the physical layer, assembles them into frames which it validates and passes to the interface machine and ACM. The receive machine accomplishes this by recognizing the frame start and the frame end delimiters, checking the frame check sequence and validating the frame's structure. The receive machine also identifies and indicates the reception of noise bursts and the bus quiet condition.

Transmit machine

This machine generally accepts a data frame from the access control machine and transmit it, as a sequence of atomic symbols in the proper format, to the physical layer. The transmit machine builds a MAC protocol data unit by prefacing each frame with the required preamble and SD, and appending the FCS and ED.

ACM Finite State Machine Description

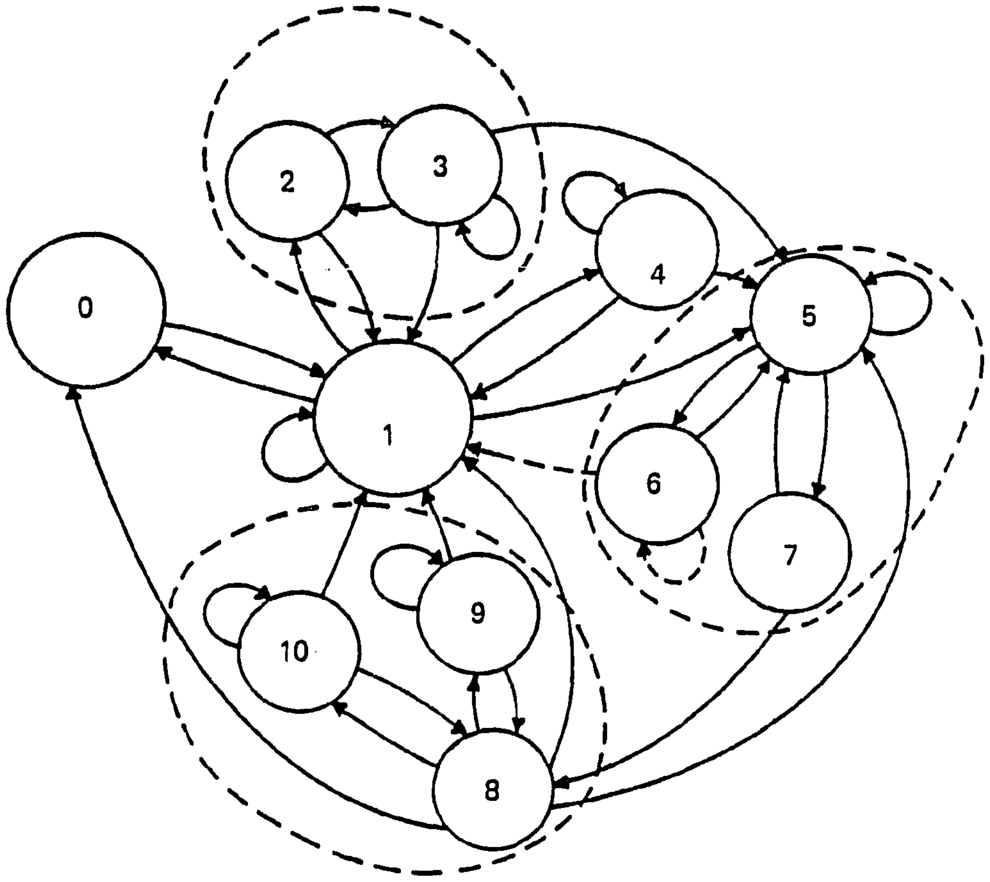
The medium access logic in a station can be described as a finite state machine which sequences through a number of distinct states. These states and transitions among them are illustrated in Figure 6. The dashed lines group states into functional areas.

Offline

Offline is the state the access machine is in immediately following power up or the detection of certain fault conditions. After powering up, a station tests itself and its connection to the medium without transmitting on the medium. Upon completion of a power up procedures, the station remains in the offline state until it has all necessary internal parameters initialized and is instructed to go online.

Idle

Idle is the state where the station is listening to the medium and not transmitting. If a MAC control frame is received for which the station needs to take action, the appropriate state is entered.



0	OFFLINE	6	AWAIT_IFM_RESPONSE
1	IDLE	7	CHECK_ACCESS_CLASS
2	DEMAND_IN	8	PASS_TOKEN
3	DEMAND_DELAY	9	CHECK_TOKEN_PASS
4	CLAIM_TOKEN	10	AWAIT_RESPONSE
5	USE_TOKEN		

FIGURE 6. ACM finite state machine transition diagram

<u>Demand in</u>

The demand_in state is entered from the idle state if a solicit_successor frame that spans the station's address is received by a station desiring logical ring entry. In the demand_in state the contending station sends the token holder a set_successor frame and goes to the demand_delay state to await a response. While staying in demand_in state, if the station hears any transmissions, it assumes that another station with a higher address is requesting the token and returns to the idle state.

<u>Demand delay</u>

Demand_delay is the state the station enters after having sent a set_successor frame in the demand_in state. In the demand_delay state a station can expect to hear:

- A token from the token holder indicating its set_successor frame was heard, which allows soliciting station to go to the use_token state and start transmitting.
- Set_successor frames from other stations, which the station ignores.
- A resolve_contention frame from the token holder indicating that all stations which are still demanding into the logical ring should perform another step of the contention resolution process,

which puts the station into the more complicated situation.

<u>Claim token</u>

Claim_token is entered from the idle state after the network inactivity timer expires and the station desires to be included in the logical ring. In this state, the station attempts to initialize or reinitialize the logical ring by sending claim_token frame. The station successfully claims the token if it hears nothing after sending maximum number of claim_token frames.

<u>Use token</u>

Use_token state is entered after receiving or claiming a token. This is the state in which a station can send data frames. As the station enters the state, it starts the token hold timer which limits how long the station may remain sending before passing the token.

<u>Await IFM response</u>

Await_IFM_response is entered when a data frame has been sent. If the frame sent was a request_with_response frame, the station waits in the await_IFM_response state for either a response frame addressed to the requestor, any other valid frame, or a timeout.

If a response is heard, the station returns to use_token state to check for another frame or token hold timer timeout. If any other valid frame is heard, an error has occurred. The station returns to the idle state and process received frame. If a timeout occurs, the station repeats the same process by sending request_with_response data frame.

Check access class

Check_access_class controls the transmission of frames for different access classes. If the priority option is not implemented, all frames are considered to be high priority and the check_access_class state only serves to control entry to token passing.

Pass token

Pass_token is the state in which a station attempts to pass the token to its successor. When its inter_solicit_count value is zero and time remains on the ring maintenance timer, the station allows new stations to enter the logical ring before passing the token. The token holding station does this by sending a solicit_successor_1 or solicit_successor_2 frame as appropriate.

<u>Check token pass</u>

Check_token_pass is the state in which the station waits for a reaction from the station to which it just passed the token. The station sending the token waits one slot_time for the station receiving the token to transmit. If a valid frame is heard which started during the response window, the station assumes the token pass is successful. The frame is processed as if it were received in the idle state. If noise or invalid frame is heard, the station continues to listen for additional transmissions.

<u>Await response</u>

Await_response is the state in which the station attempts to sequence candidate successors through a distributed contention resolution algorithm until one of those successor's set_successor frame is correctly received or until no successor appears. The state is entered from the pass_token state whenever the station determines it is time to open a response window or if the station does not know its successor as in initialization or when a token pass fails.

CHAPTER 3. TOKEN BUS PROTOCOL MAINTENANCE FUNCTIONS

The token bus protocol is a technique in which the stations on the bus form a logical ring as described in Chapter 1. The stations are assigned positions in an ordered sequence, with the last member of the sequence followed by the first.

It is very important for the network to keep and maintain this logical ring, and maintenance functions are implemented to support this. In a sense, these maintenance functions are overhead to the system because they don't contribute to actual data transmission directly. Maintenance functions, however, should be well defined and cleverly implemented to support reliable bus operation.

Ring Initialization

Ring initialization is primarily a special case of adding new stations; it is triggered by the exhaustion of an inactivity timer in one or more stations. This can be due to a number of causes such as the network has just been powered up or a token holding station fails. If the inactivity timer expires, the station sends a claim_token frame. The initialization algorithm assumes that more than one station can try to initialize the network at a given instant. This is resolved by address sorting the initializers.

Each potential initializer sends a claim_token frame having an information field length that is a multiple of the system slot_time. Each initializing station then waits one slot_time for its own transmission, and those of other stations that chose the same frame length, to pass. Then the station samples the state of the medium. If a station senses non-silence, it knows that some other station sent a longer length transmission. The station defers to those stations with the longer transmission and reenters the idle state.

If silence was detected and unused bits remain in the address string, the station attempting initialization repeats the process using the next two bits of its address to derive the length of the next transmitted frame. If all bits have been used and silence is still sensed, the station has won the initialization contest and now holds the token. Once there is a unique token in the network, the logical ring builds by way of the station addition process.

Station Addition

To accomplish station addition to the logical ring, each station in the ring has the responsibility of periodically granting an opportunity for new stations to enter the ring. While holding the token, the station issues a solicit_successor frame, inviting stations with an address between itself and the next station in logical sequence to demand entrance. The transmitting station then waits for one response_window or slot_time which is about equal to twice the end-to-end propagation delay of the medium.

If there is no response, the station passes the token to its successor as usual. If there is one response, the token holder sets its successor station to be the requesting station and transmits the token to it; the requestor sets its linkage accordingly and proceed. If more than one station demands to enter the ring, the token holder will detect a garbled response.

The conflict is resolved by an address_based contention scheme. The token holder transmits a resolve_contention packet and waits four response_windows. Each demander can respond in one of these windows based on the first two bits of its address. If a demander hears anything before its window comes up, it refrains from demanding. If the token holder hears valid frame, it has found its successor. Otherwise, it tries again and only those stations that responded first time are allowed to respond this time, based on the second pair of bits in their address.

This process continues until a valid frame is heard, no response is received, or retry limit timer expires. In the latter two cases, the token holder gives up and passes the token to its original successor.

Station Deletion

A station can remove itself from the logical ring at any time by simply choosing not to respond to a token passed to it, allowing the fault recovery mechanisms in the medium access protocol to patch it up. A more efficient method is: when the station has the token and desires to exit the logical ring, the station sends a set_successor frame to its predecessor, the station that transmitted the token to it, containing the address of its successor. The station then simply sends the token as usual to its successor, and it is out of the logical ring.

Token Recovery

A token in the network can be lost when just powered up or bit error occurs in the token frame. In this case, there is no station in the network which has the right to transmit. This makes the bus go quiet and inactivity timer is started. From this point on, the network follows the steps in ring initialization process to get the token back in the system.

Another case is that while holding the token, a station may hear a valid frame, which indicates multiple tokens in the network. In this situation the station which heard a valid frame on the bus drops the token by going into idle state to listen. In this way the number of token holders drop immediately to 1 or 0. If the number comes down to zero, then the ring initialization process starts after inactivity timer expires to get a unique token in the system.

Receiver Fault Recovery

If the receiver of a station is inoperative, the predecessor of the faulty station cannot pass the token to its successor. The predecessor tries twice to pass the token in vain and assumes that its successor has failed. The predecessor then sends a who_follows frame asking for the identity of the station that follows the failed station in the logical ring. If the predecessor of the failed station gets the set_successor frame from the second station down the line, then the predecessor adjusts its linkage and passes the token.

If the predecessor gets no response after two who_follows process, it sends set_successor_2 frame with the full address range. From this point, the predecessor follows ring initialization process described earlier.

Transmitter Fault Recovery

If the transmitter of a station is faulty, the token will be eventually lost because once the token arrives at the faulty station, it will not be transmitted again onto the bus. This makes network inactivity timer start and ring initialization process will be invoked upon the timer expiration.

Overhead Analysis of Maintenance Functions

For the development of analytic models for maintenance functions, overhead generated by maintenance functions should be quantified. Each maintenance function described requires certain amount of time to be performed in the network. The overhead will be reflected into performance analysis in the next chapter.

Ring initialization

The overhead associated with the ring initialization can be quantified as follows:

```
Hi_min = A*[(claim_token with no data field)+slot_time]
         +L*[minimum station addition time]

Hi_max = A*[(claim_token with 6*slot_time length data)
         +slot_time]+L*[maximum station addition time]
```

where, A = (address length)/2

L = number of stations in a logical ring

Station addition

The overhead associated with a station addition can be quantified as follows:

Ha_min = (solicit_successor_1)+slot_time+(set_successor)

Ha_max = [(solicit_successor_1)+slot_time]
+A*[(resolve_contention)+4*slot_time]

Station deletion

The overhead associated with a station deletion can be quantified as follows:

Hd = (set_successor)

Token recovery

The overhead associated with token recovery can be quantified as follows:

Ht_min = A*[(claim_token with no data field)+slot_time]

Ht_max = A*[(claim_token with 6*slot_time length data)
+slot_time]

<u>Receiver fault recovery</u>

The overhead associated with receiver fault recovery can be quantified as follows:

```
Hr_min = [(who_follows)+3*slot_time+(set_successor)]

Hr_max = 2*[(who_follows)+3*slot_time]
         +2*[(solicit_successor_2)+2*slot_time]
```

<u>Transmitter fault recovery</u>

The overhead associated with transmitter fault recovery can be quantified as in the case of token recovery:

```
Hx_min = A*[(claim_token with no data field)+slot_time]

Hx_max = A*[(claim_token with 6*slot_time length data)
         +slot_time]
```

CHAPTER 4. PERFORMANCE ANALYSIS OF TOKEN BUS PROTOCOL

The logical ring of a token bus protocol can be modeled as a network of queues as depicted in Figure 7. From the analytic modeling point of view, this system has the same characteristics as polling network. A straight forward analysis of a polling network can be made to determine the average cycle time [28], but sophisticated techniques are required for a rigorous determination of the average delay and the average number of packets stored in the station buffer [9]. A derivation of an expression for average delay can also be made in a heuristic manner [29], which is the approach used to develop basic analytic model in this dissertation.

Among many performance measures message delay and throughput or channel utilization have become two distinguished measures. For a network user, message delay which indicates how fast his message is serviced is a good measure to evaluate the network, and for a server, it is very important to know how well it is being used which is indicated by its utilization. Virtually all performance analyses deal with these two parameters and they are investigated here, also.

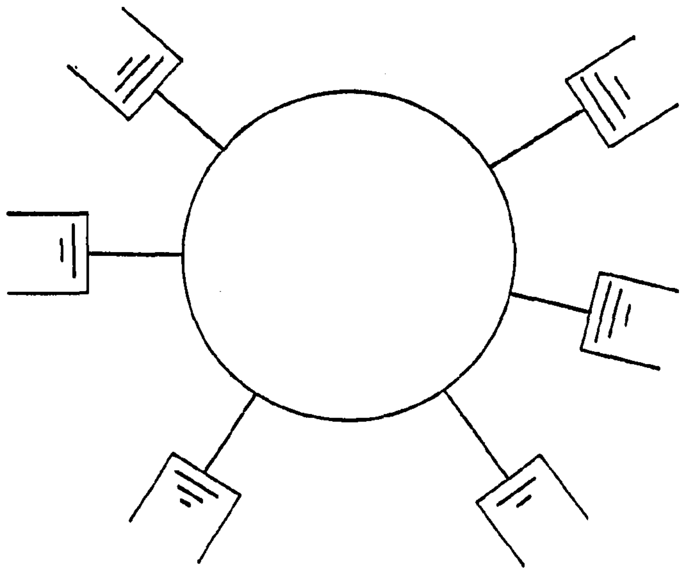


FIGURE 7. Queueing model for token bus logical ring

Assumptions for the analysis are:

- Message arrival processes are Poisson processes with average arrival rate $\overline{\lambda i}$ and average message length $\overline{mi}$ for each station i.
- All queues are infinite in size.
- When a station has the token, it transmits all messages in the queue.
- Message length is exponentially distributed.

and variables are:

Tc	:	cycle time to poll all stations
Tp	:	propagation time through the length of the bus
Ti	:	transmission time for all messages in station i

Toi	: overhead time at station i
Tpi	: propagation time for messages in station i
N	: number of stations on the logical ring
C	: channel bit rate
ρi	: traffic intensity of station i
$\overline{\lambda i}$	: average message arrival rate of station i
$\overline{mi}$	: average message length of station i
D	: message delay
U	: channel utilization

Token Bus Protocol without Maintenance Functions

A basic analytic model for token bus protocol is developed in this section. This is a conventional model with the assumption that no faulty conditions are occurring in the network. In the next section, this model will be modified to implement maintenance functions.

Average cycle time

The average cycle time which represents the time required by the server to offer access to all stations is very important in calculating message delay, and is discussed here.

The cycle time can be written in terms of message transmit time, overhead time, and propagation time.

$$Tc = \sum_{i=1}^{N} Ti + \sum_{i=1}^{N} Toi + \sum_{i=1}^{N} Tpi \qquad (4.1)$$

Since the message arrival process is Poisson process, the number of messages in a queue depends on the cycle time, which again depends on the time spent at each queue Ti, i=1..N. This interdependency between cycle time and message transmit times gives rise to the dependencies among random variables Ti, i=1..N.

In spite of these dependencies, an expression for the average cycle time can be derived quite easily. From the linearity of the expectation operator, the average cycle time can be expressed as

$$\overline{Tc} = \sum_{i=1}^{N} \overline{Ti} + \sum_{i=1}^{N} \overline{Toi} + \sum_{i=1}^{N} \overline{Tpi} \qquad (4.2)$$

The average time required to transmit all of the messages that arrived in a cycle can be calculated as follows with the assumptions of Poisson arrival and infinite buffer.

$$\overline{Ti} = \overline{\lambda i}\,\overline{mi}\,\overline{Tc}/C = \overline{\rho}i\overline{Tc}/C \qquad (4.3)$$

The average propagation time on the bus is statistically the half of the maximum propagation time. That is

$$\overline{Tpi} = Tp/2 \qquad (4.4)$$

Substituting equations (4.3) and (4.4) into (4.2), and rearranging for $\overline{Tc}$ gives

$$\overline{Tc} = \frac{NTp/2 + \sum_{i=1}^{N} \overline{Toi}}{1 - \sum_{i=1}^{N} \overline{\rho i}/C} \qquad (4.5)$$

and channel utility U can be found as

$$U = \sum_{i=1}^{N} \overline{\rho i}/C \qquad (4.6)$$

For (4.5) and (4.6) to be valid, the total offered traffic must not exceed channel data rate C.

<u>Message delay</u>

In a token bus network, an arriving message at a typical station must wait until reaching the head of the queue in the station buffer to be transmitted. This waiting delay can be divided into two components:

1. The waiting delay, Dw, in the station buffer while other stations are being served. In other words, it's a delay while the station is inactive and awaiting its turn to be polled.
2. The queue delay, Dq, in the station buffer while that particular station is being served.

Figure 8 shows the physical relationship between the different waiting period.

These delays are related mathematically by

$$D = Dw + Dq \qquad (4.7)$$

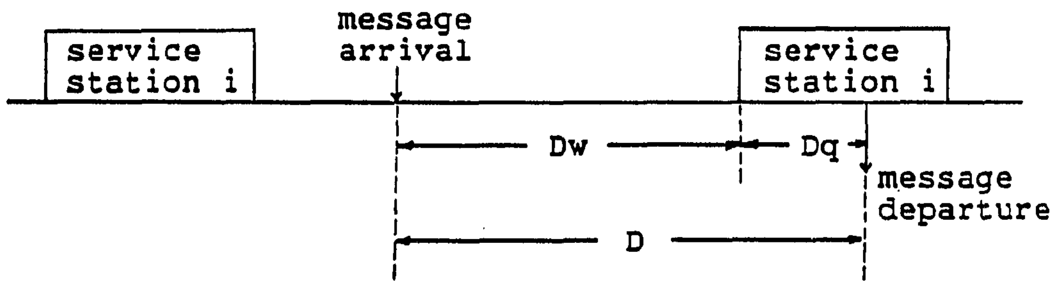


FIGURE 8. Division of delays for a typical message

All three variables in the equation (4.7) are random variables, and in general, Dw and Dq are not independent. As in the case of cycle time, however, the average message delay can be expressed as

$$\overline{D} = \overline{Dw} + \overline{Dq} \qquad (4.8)$$

It is assumed that $\overline{Dw}$ and $\overline{Dq}$ can be evaluated independently, and that $\overline{Dw}$ can be obtained through consideration of a cycle with average parameter values rather than with a more rigorous approach with all parameters of the cycle random. An additional heuristic argument is used to evaluate $\overline{Dq}$.

The pattern of activities for a polling network in a cycle with average parameters is shown in Figure 9. The average number of messages that must be transmitted over the channel by a particular station i while it is being served is $\overline{\lambda i}\overline{Tc}$. The corresponding average service time for the station is then $(\overline{\lambda i}\overline{Tc})\overline{mi}/C$ or $\overline{\rho i}\overline{Tc}/C$. So, as shown in Figure 9, the average service time for the station i can be expressed as $\overline{\rho i}\overline{Tc}/C$ and the remaining part of the average cycle during which the station is idle is then, of course, $(1 - \overline{\rho i}/C)\overline{Tc}$. If they are generalized for all stations in the network, they can be expressed as

$$\text{Average service time} = \overline{\rho}\,\overline{Tc}/C \tag{4.9}$$

$$\text{Average idle time} = (1 - \overline{\rho}/C)\overline{Tc} \tag{4.10}$$

where, $\overline{\rho} = \sum_{i=1}^{N} \overline{\rho i}/N$.

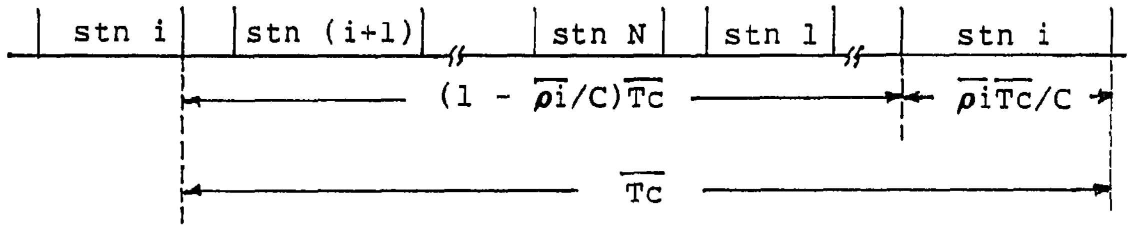


FIGURE 9. A cycle for a polling network.

Now consider message arriving at random during the idle time given by the equation (4.10). The messages arrive from a Poisson process, and for such random arrivals it is intuitive that the average waiting delay, $\overline{Dw}$, is one half of the idle interval. That is

$$\overline{Dw} = \frac{(1 - \overline{\rho}/C)\overline{Tc}}{2} \tag{4.11}$$

From equation (4.5) and (4.11), Dw can be expressed as

$$\overline{Dw} = \frac{(1 - \overline{\rho}/C)(NTp/2 + \sum_{i=1}^{N}\overline{Toi})}{2(1 - N\overline{\rho}/C)} \tag{4.12}$$

The second component, $\overline{Dq}$, of the total average delay experienced by arriving messages is the average time messages must wait to reach the head of the queue in the station buffer after the station begins receiving service. To obtain an expression for this delay, consider an equivalent network for which there is no overhead so that some station is always being served if there are messages in the network [33-44]. It is reasonable to regard this equivalent network as a M/G/1 queue with N stations and total arrival rate $\sum_{i=1}^{N}\overline{\lambda i}$. In other words, the N individual queues can be regarded as a single lumped queue with the arrival rate aggregated.

An expression for the average delay, $\overline{d}$, for an M/G/1 queue is given by equation (4.13) [28].

$$\overline{d} = \overline{Tm} + \lambda\overline{Tm^2}/[2(1 - \rho)] \tag{4.13}$$

where, $\overline{Tm}$: average message transmission time

$\overline{Tm^2}$: mean square message transmission time

The pure delay portion of the equation (4.13) is the second term of the equation which corresponds to $\overline{Dq}$. After changing variables into present notation, $\overline{Dq}$ can be expressed as

$$\overline{Dq} = \frac{\overline{\rho}\,\overline{m^2}}{2\overline{m}C^2(1 - \overline{\rho})} \tag{4.14}$$

where, $\overline{m} = \sum_{i=1}^{N} \overline{mi}/N.$

Now, the total average message delay $\overline{D}$ can be calculated. From the equation (4.8), (4.12), and (4.14), and from the exponential message length assumption where $\overline{m^2} = 2(\overline{m})^2$, the final expression for D can be deduced as

$$D = \frac{(1 - \overline{\rho}/C)(NTp/2 + \sum_{i=1}^{N} \overline{Toi})}{2(1 - N\overline{\rho}/C)} + \frac{\overline{\rho}\,\overline{m}}{C^2(1 - \overline{\rho})} \tag{4.15}$$

Token Bus Protocol with Maintenance Functions

As mentioned in Chapter 3, maintenance functions are projected into performance analysis in this section. Maintenance functions will effect average message delay expressed in equation (4.15) by increasing the value of the term $\sum_{i=1}^{N} Toi$.

The overhead at certain station i can be expressed in terms of normal station delay which includes token passing and maintenance delay. This can be expressed as

$$Toi = Dsi + Dmi \tag{4.16}$$

where, Dsi : station delay

Dmi : maintenance delay

The values of Dsi and Dmi are up to the system variables such as bus speed, quality of receiver and transmitter, and bus material. Generally speaking Dsi is a certain constant value common to all stations. The value of Dmi, however, depends on the kind of maintenance functions performed at each station.

In order to calculate the value of the maintenance delays, it is necessary to evaluate the transmission times of MAC control frames and they are listed in Table 1, where Ts represents the slot_time and C is the channel data rate in Mbps.

TABLE 1. MAC control frame transmission time

MAC control frame	transmission time (μsec)
Claim_token (minimum)	2 + 88/C
Claim_token (maximum)	2 + 6*Ts + 88/C
Solicit_successor_1	2 + 88/C
Solicit_successor_2	2 + 88/C
Who_follows	2 + 104/C
Resolve_contention	2 + 88/C
Set_successor	2 + 104/C

Now, the values of maintenance delays can be calculated from overheads explained in previous chapter and they are summarized in Table 2. The maintenance delays cannot be averaged because maintenance of the network occurs irregularly and is totally unpredictable. So, the appropriate approach might be scenario studies where certain types and amounts of failures are assumed. This technique is discussed in the next chapter.

TABLE 2. Maintenance delays

Maintenance function		Maintenance delays (μsec)
Ring initialization	min	8*[(2+88/C)+Ts] +L*[(2+88/C)+Ts+(2+104/C)]
	max	8*[(2+88/C+6*Ts)+Ts] +L*[(2+88/C)+Ts +8*(2+88/C+4*Ts)]
Station addition	min	(2+88/C)+Ts+(2+104/C)
	max	(2+88/C)+Ts+8*(2+88/C+4*Ts)
Station deletion		2+104/C
Token recovery	min	8*[(2+88/C)+Ts]
	max	8*[(2+88/C+6*Ts)+Ts]
Receiver fault recovery	min	(2+104/C)+3*Ts+(2+104/C)
	max	2*[(2+104/C)+3*Ts] +2*[(2+88/C)+2*Ts]
transmitter fault recovery	min	8*[(2+88/C)+Ts]
	max	8*[(2+88/C+6*Ts)+Ts]

CHAPTER 5. REAL SYSTEM APPLICATION

From the analytic model developed in Chapter 4, now it is possible to perform some sample studies of the token bus protocol delay-throughput analysis. First, the system variables must be assigned with proper values for the analysis.

- C = 50 Mbps
- N = 16 stations
- Ts = 1 μsec
- Ds = 2 μsec/message
- Tp = .5 μsec

These values are acquired from the token bus network being actually built at Rockwell Collins International Co. along with benchmark input data which is supplied in Appendix B.

Effect of Maintenance Functions

With values of system variables given above and setting L to 2 for minimum case and to 16 for maximum case, maintenance delays can be recalculated as shown in Table 3.

Each of the six maintenance functions mentioned contributes some additional delays to the delay-throughput characteristics of the system. Figure 10 through 15 show the effects of maintenance functions on message delay and

TABLE 3. Recalculated maintenance delay

Maintenance function		Delay (μsec)
Ring initialization	min	55.76
	max	1155.52
Station addition	min	8.84
	max	66.84
Station deletion		4.08
Token recovery	min	38.08
	max	86.08
Receiver fault recovery	min	11.16
	max	25.68
Transmitter fault recovery	min	38.08
	max	86.08

Table 4 through 9 contain actual values of delays. Normal delays which include no maintenance are also supplied for comparison.

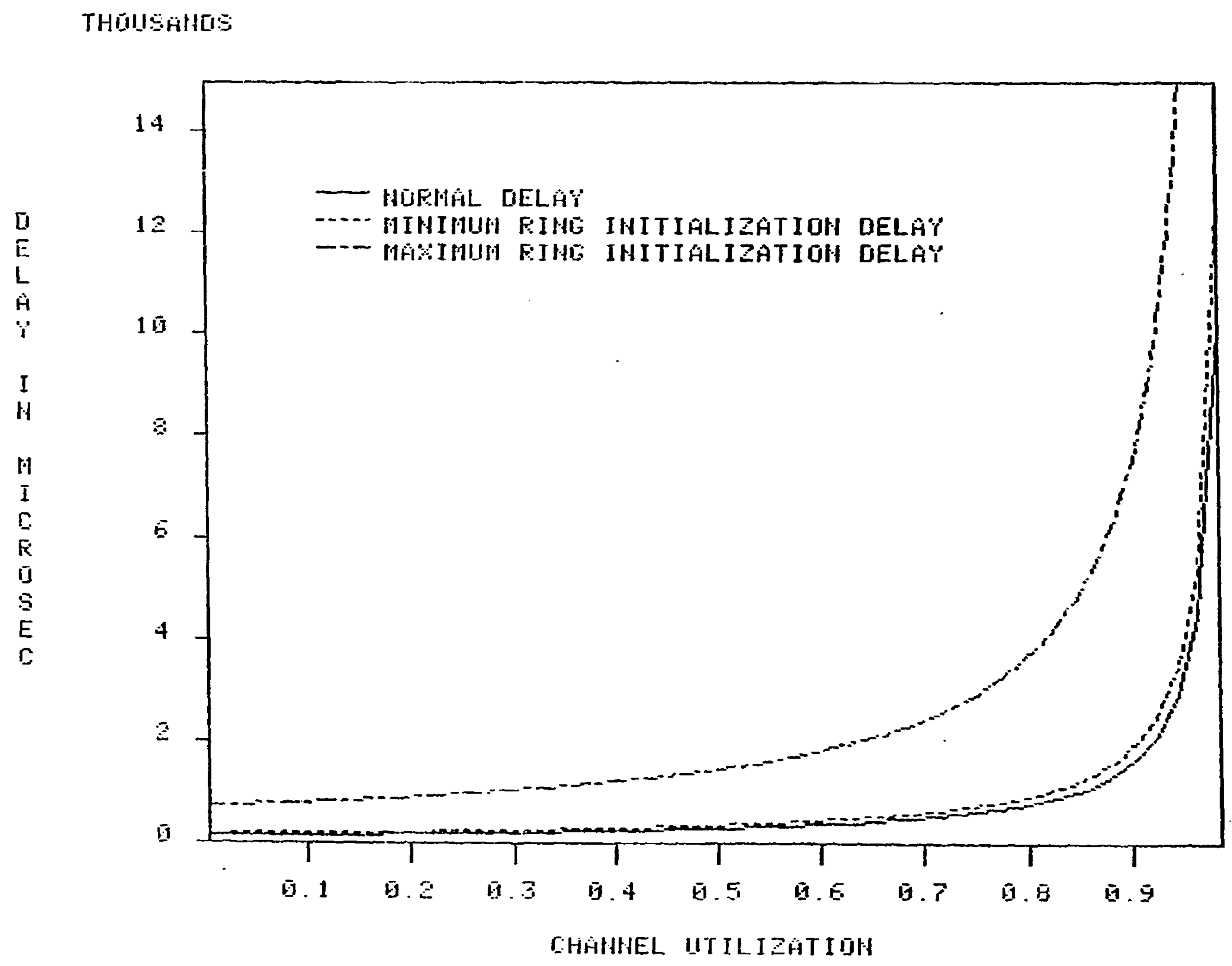


FIGURE 10. Ring initialization maintenance delay

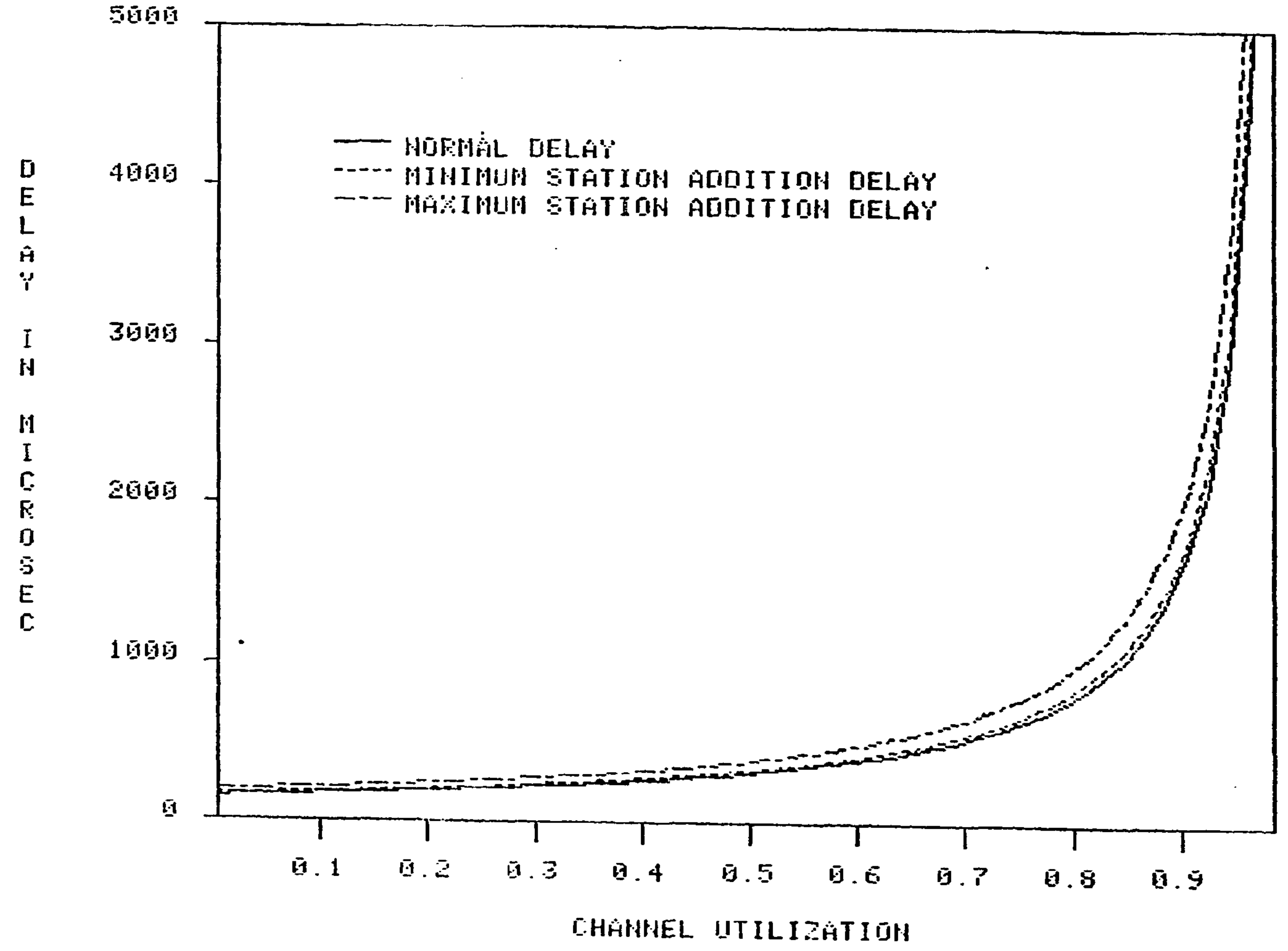


FIGURE 11. Station addition maintenance delay

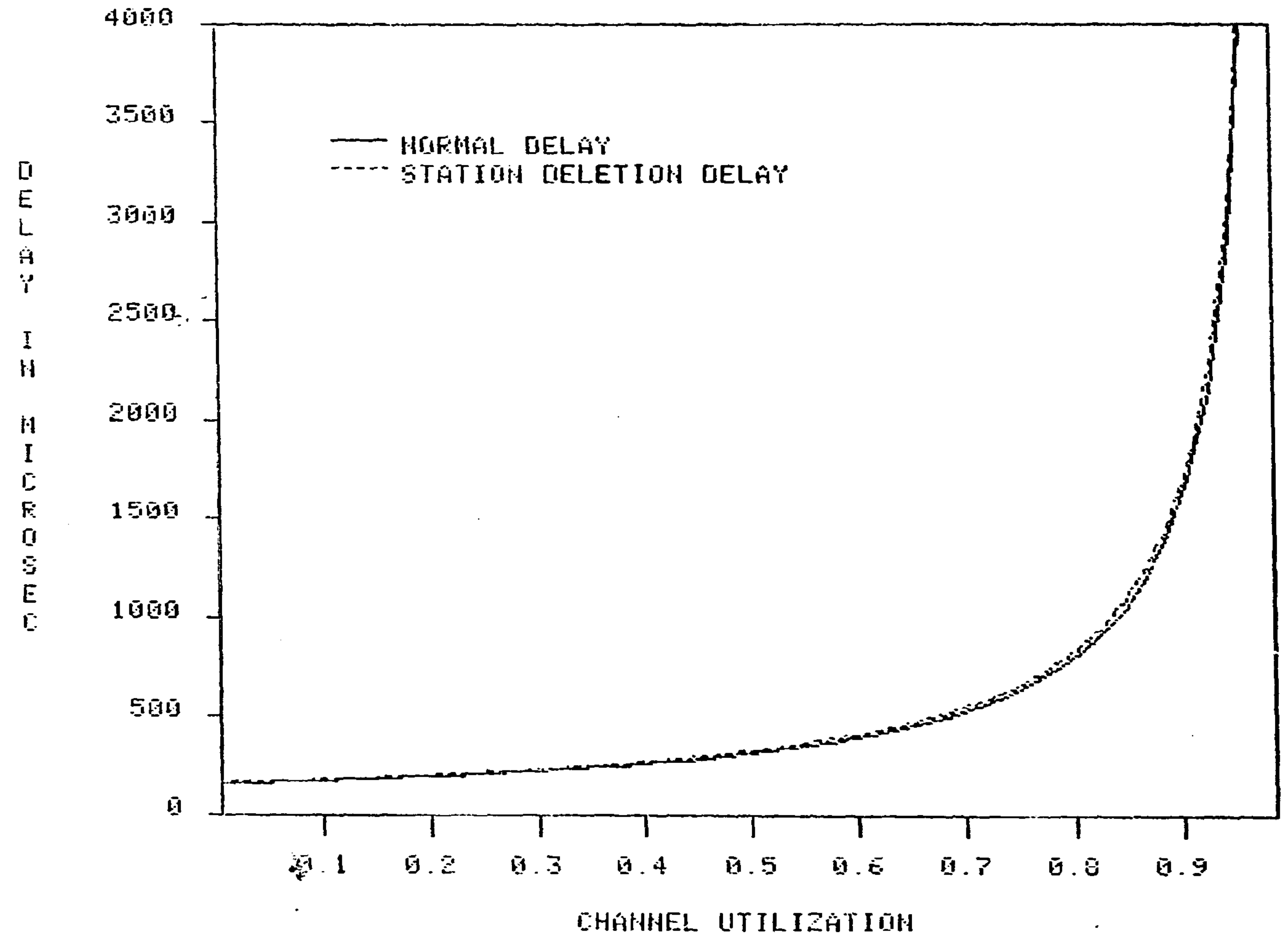


FIGURE 12. Station deletion maintenance delay

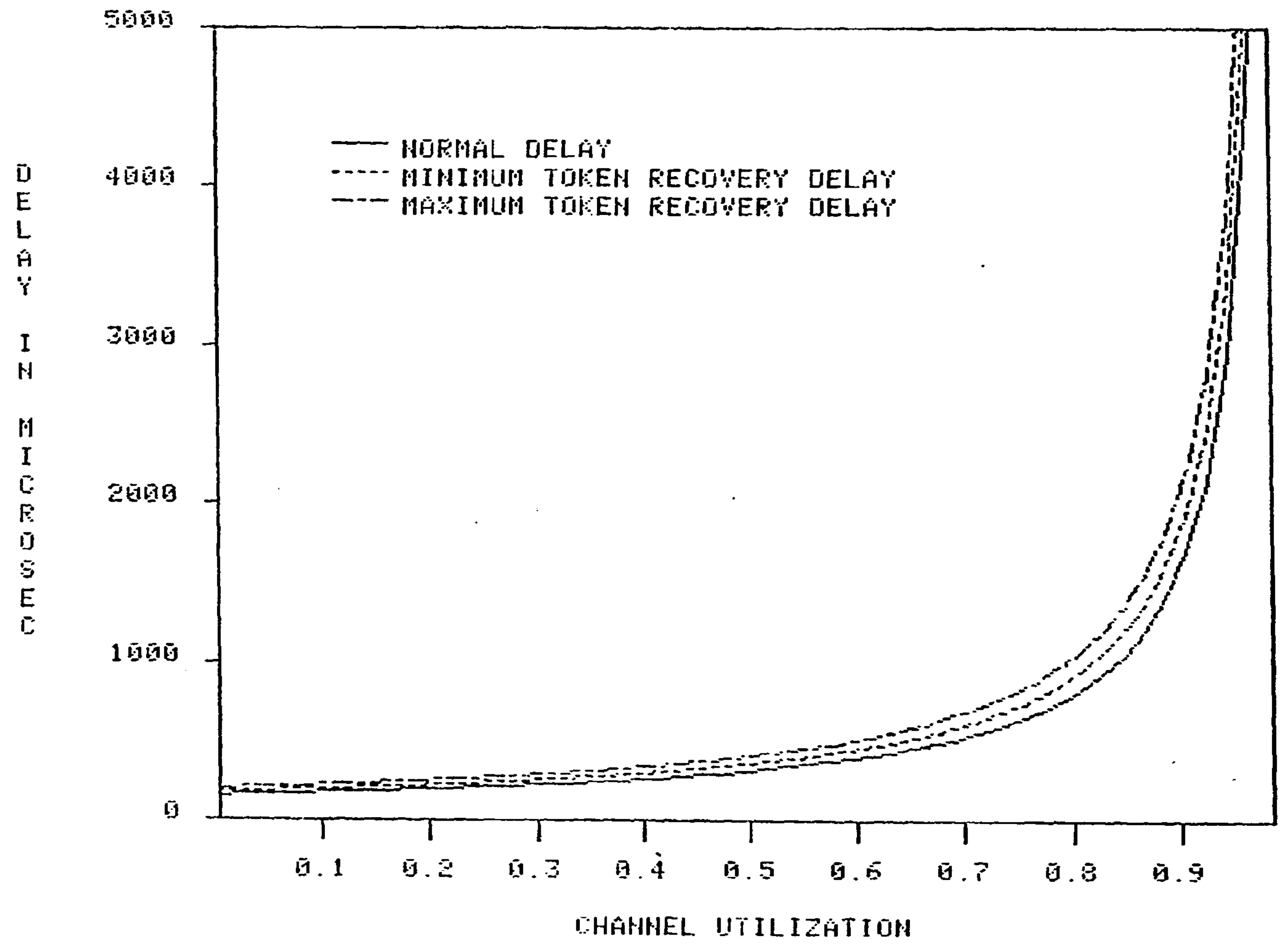


FIGURE 13. Token recovery maintenance delay

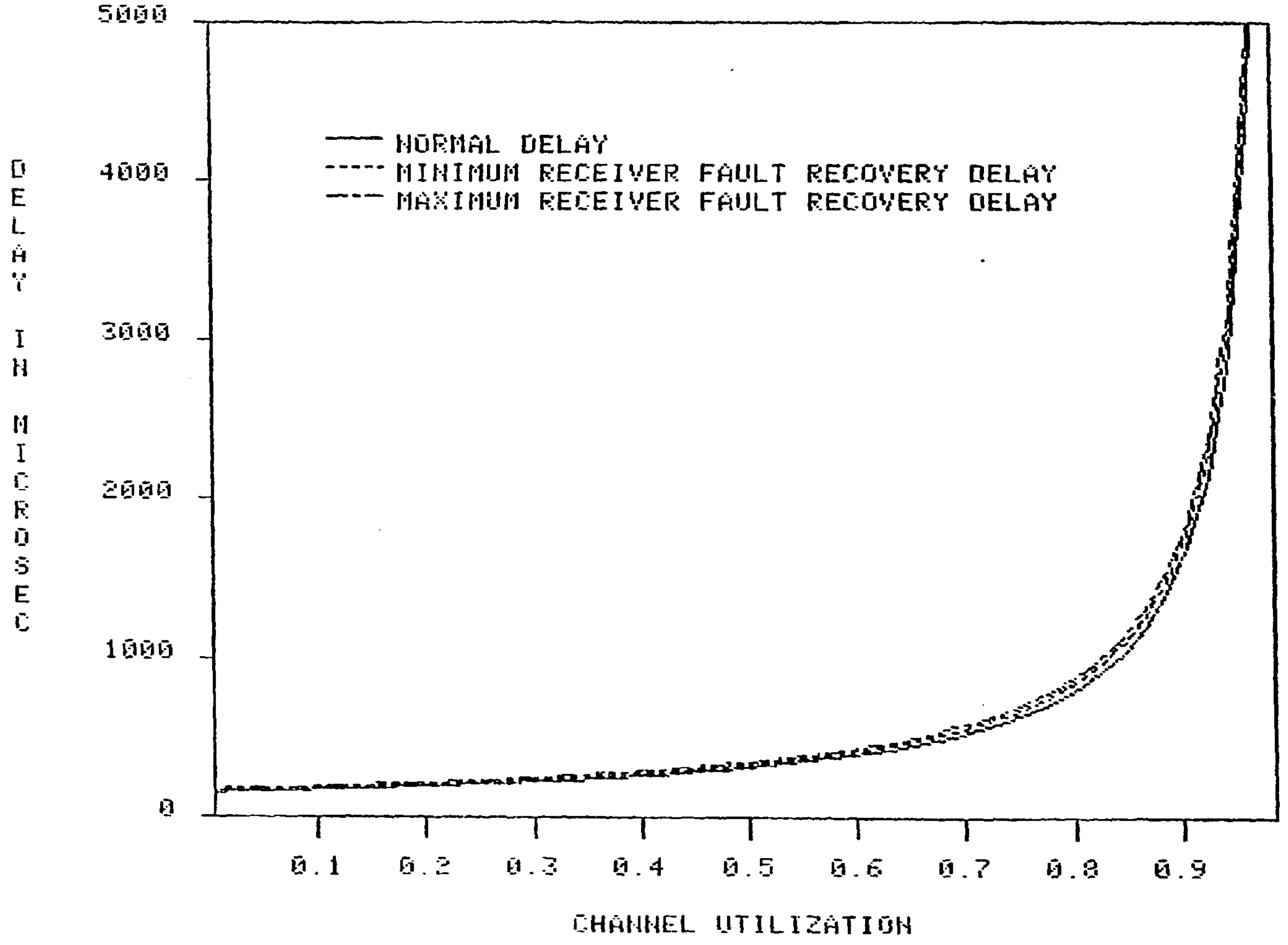


FIGURE 14. Receiver fault recovery maintenance delay

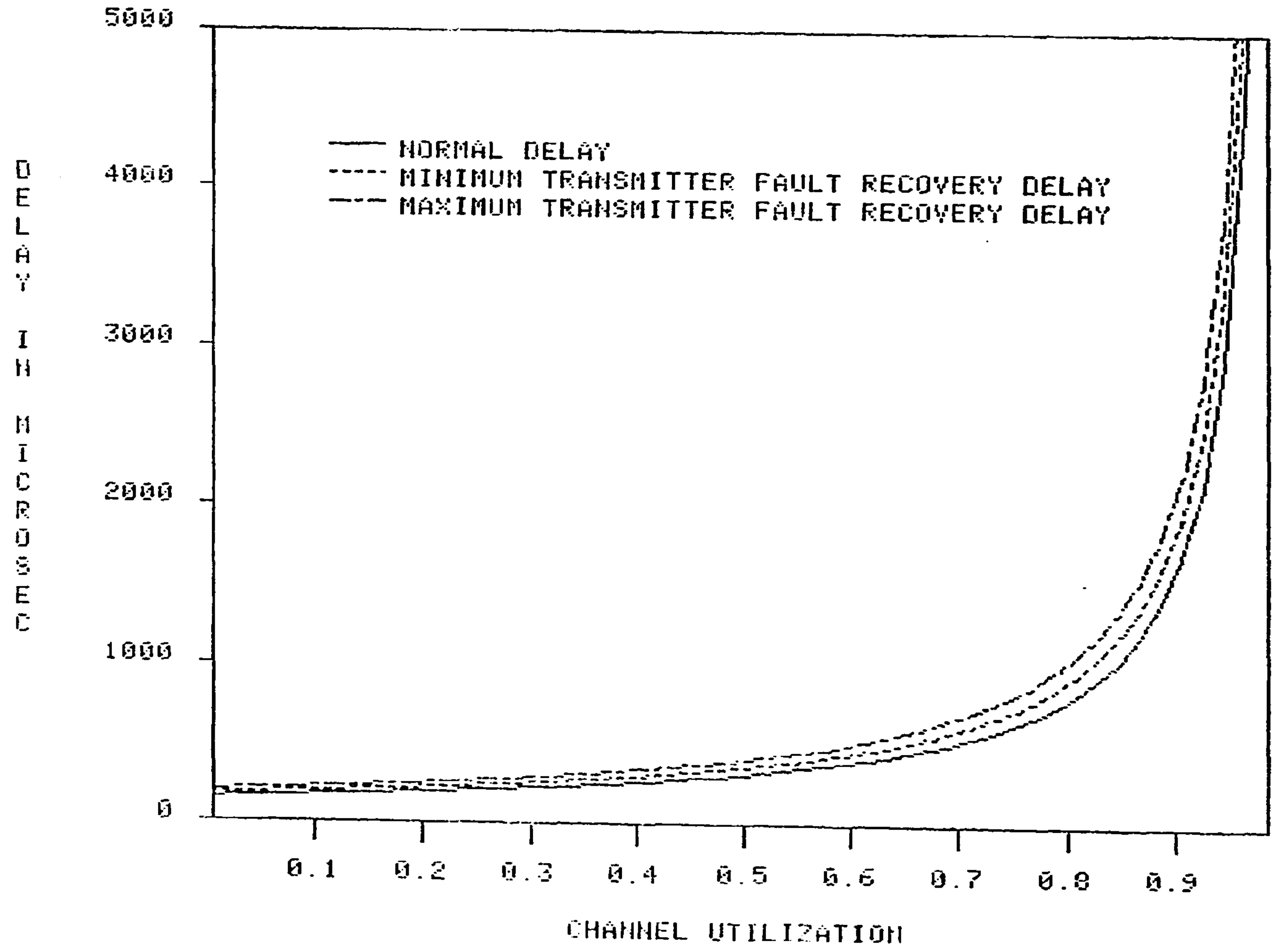


FIGURE 15. Transmitter fault recovery maintenance delay

TABLE 4. Ring initialization maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	MINIMUM DELAY	MAXIMUM DELAY
0.1608	191.84	229.83	885.05
0.1804	196.43	235.33	906.22
0.2000	201.25	241.09	928.43
0.2196	206.30	247.15	951.75
0.2392	211.62	253.52	976.28
0.2588	217.22	260.23	1002.11
0.2784	223.12	267.30	1029.33
0.2980	229.35	274.76	1058.08
0.3176	235.94	282.66	1088.48
0.3372	242.92	291.02	1120.68
0.3568	250.32	299.89	1154.84
0.3764	258.19	309.32	1191.15
0.3960	266.58	319.36	1229.82
0.4156	275.52	330.08	1271.09
0.4353	285.08	341.53	1315.21
0.4549	295.34	353.82	1362.52
0.4745	306.36	367.02	1413.35
0.4941	318.23	381.24	1468.12
0.5137	331.06	396.61	1527.31
0.5333	344.96	413.27	1591.47
0.5529	360.09	431.39	1661.25
0.5725	376.61	451.18	1737.44
0.5921	394.71	472.87	1820.95
0.6117	414.64	496.74	1912.90
0.6313	436.69	523.16	2014.62
0.6509	461.21	552.54	2127.78
0.6705	488.66	585.42	2254.40
0.6901	519.58	622.46	2397.04
0.7097	554.68	664.51	2558.95
0.7293	594.86	712.65	2744.32
0.7490	641.31	768.30	2958.64
0.7686	695.64	833.39	3209.28
0.7882	760.02	910.52	3506.31
0.8078	837.54	1003.38	3863.92
0.8274	932.67	1117.35	4302.78
0.8470	1052.17	1260.51	4854.09
0.8666	1206.80	1445.76	5567.45
0.8862	1414.70	1694.83	6526.60
0.9058	1709.15	2047.58	7885.01
0.9254	2158.38	2585.77	9957.52
0.9450	2927.97	3507.74	13507.93
0.9646	4550.52	5451.58	20993.46
0.9842	10206.24	12227.20	47085.62

TABLE 5. Station addition maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	MINIMUM DELAY	MAXIMUM DELAY
0.1608	191.84	201.88	236.43
0.1804	196.43	206.70	242.09
0.2000	201.25	211.77	248.02
0.2196	206.30	217.09	254.25
0.2392	211.62	222.68	260.80
0.2588	217.22	228.58	267.70
0.2784	223.12	234.79	274.97
0.2980	229.35	241.34	282.65
0.3176	235.94	248.28	290.78
0.3372	242.92	255.62	299.38
0.3568	250.32	263.41	308.50
0.3764	258.19	271.70	318.20
0.3960	266.58	280.52	328.53
0.4156	275.52	289.93	339.56
0.4353	285.08	299.99	351.34
0.4549	295.34	310.78	363.98
0.4745	306.36	322.38	377.56
0.4941	318.23	334.87	392.19
0.5137	331.06	348.37	408.00
0.5333	344.96	363.01	425.14
0.5529	360.09	378.92	443.78
0.5725	376.61	396.30	464.14
0.5921	394.71	415.35	486.45
0.6117	414.64	436.32	511.01
0.6313	436.69	459.53	538.18
0.6509	461.21	485.34	568.41
0.6705	488.66	514.22	602.24
0.6901	519.58	546.75	640.34
0.7097	554.68	583.68	683.59
0.7293	594.86	625.97	733.11
0.7490	641.31	674.85	790.37
0.7686	695.64	732.02	857.32
0.7882	760.02	799.77	936.67
0.8078	837.54	881.34	1032.20
0.8274	932.67	981.44	1149.44
0.8470	1052.17	1107.20	1296.72
0.8666	1206.80	1269.91	1487.28
0.8862	1414.70	1488.69	1743.51
0.9058	1709.15	1798.53	2106.39
0.9254	2158.38	2271.26	2660.04
0.9450	2927.97	3081.09	3608.49
0.9646	4550.52	4788.51	5608.17
0.9842	10206.24	10740.00	12578.40

TABLE 6. Station deletion maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	DELAY
0.1608	191.84	199.04
0.1804	196.43	203.80
0.2000	201.25	208.80
0.2196	206.30	214.04
0.2392	211.62	219.56
0.2588	217.22	225.36
0.2784	223.12	231.49
0.2980	229.35	237.95
0.3176	235.94	244.79
0.3372	242.92	252.03
0.3568	250.32	259.71
0.3764	258.19	267.88
0.3960	266.58	276.58
0.4156	275.52	285.86
0.4353	285.08	295.78
0.4549	295.34	306.42
0.4745	306.36	317.85
0.4941	318.23	330.17
0.5137	331.06	343.48
0.5333	344.96	357.91
0.5529	360.09	373.60
0.5725	376.61	390.73
0.5921	394.71	409.52
0.6117	414.64	430.19
0.6313	436.69	453.07
0.6509	461.21	478.52
0.6705	488.66	506.99
0.6901	519.58	539.07
0.7097	554.68	575.48
0.7293	594.86	617.17
0.7490	641.31	665.37
0.7686	695.64	721.74
0.7882	760.02	788.54
0.8078	837.54	868.96
0.8274	932.67	967.66
0.8470	1052.17	1091.64
0.8666	1206.80	1252.07
0.8862	1414.70	1467.77
0.9058	1709.15	1773.27
0.9254	2158.38	2239.36
0.9450	2927.97	3037.81
0.9646	4550.52	4721.24
0.9842	10206.24	10589.13

TABLE 7. Token recovery maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	MINIMUM DELAY	MAXIMUM DELAY
0.1608	191.84	219.30	247.89
0.1804	196.43	224.54	253.82
0.2000	201.25	230.04	260.04
0.2196	206.30	235.82	266.58
0.2392	211.62	241.90	273.45
0.2588	217.22	248.30	280.68
0.2784	223.12	255.05	288.31
0.2980	229.35	262.17	296.36
0.3176	235.94	269.70	304.87
0.3372	242.92	277.68	313.89
0.3568	250.32	286.15	323.46
0.3764	258.19	295.14	333.63
0.3960	266.58	304.72	344.46
0.4156	275.52	314.95	356.02
0.4353	285.08	325.88	368.38
0.4549	295.34	337.60	381.63
0.4745	306.36	350.20	395.86
0.4941	318.23	363.77	411.21
0.5137	331.06	378.43	427.78
0.5333	344.96	394.33	445.75
0.5529	360.09	411.62	465.30
0.5725	376.61	430.50	486.64
0.5921	394.71	451.19	510.03
0.6117	414.64	473.98	535.78
0.6313	436.69	499.18	564.28
0.6509	461.21	527.22	595.97
0.6705	488.66	558.59	631.43
0.6901	519.58	593.93	671.39
0.7097	554.68	634.05	716.74
0.7293	594.86	679.98	768.66
0.7490	641.31	733.09	828.69
0.7686	695.64	795.19	898.89
0.7882	760.02	868.79	982.08
0.8078	837.54	957.40	1082.25
0.8274	932.67	1066.14	1205.17
0.8470	1052.17	1202.74	1359.59
0.8666	1206.80	1379.50	1559.39
0.8862	1414.70	1617.15	1828.04
0.9058	1709.15	1953.74	2208.52
0.9254	2158.38	2467.26	2789.01
0.9450	2927.97	3346.98	3783.44
0.9646	4550.52	5201.73	5880.07
0.9842	10206.24	11666.81	13188.23

TABLE 8. Receiver fault recovery maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	MINIMUM DELAY	MAXIMUM DELAY
0.1608	191.84	203.26	211.91
0.1804	196.43	208.12	216.98
0.2000	201.25	213.22	222.30
0.2196	206.30	218.58	227.88
0.2392	211.62	224.21	233.75
0.2588	217.22	230.14	239.94
0.2784	223.12	236.39	246.45
0.2980	229.35	243.00	253.34
0.3176	235.94	249.98	260.62
0.3372	242.92	257.37	268.33
0.3568	250.32	265.22	276.51
0.3764	258.19	273.56	285.20
0.3960	266.58	282.44	294.46
0.4156	275.52	291.91	304.34
0.4353	285.08	302.05	314.90
0.4549	295.34	312.91	326.23
0.4745	306.36	324.58	338.40
0.4941	318.23	337.16	351.51
0.5137	331.06	350.76	365.68
0.5333	344.96	365.49	381.05
0.5529	360.09	381.52	397.76
0.5725	376.61	399.02	416.00
0.5921	394.71	418.19	435.99
0.6117	414.64	439.31	458.01
0.6313	436.69	462.67	482.36
0.6509	461.21	488.66	509.46
0.6705	488.66	517.74	539.77
0.6901	519.58	550.50	573.93
0.7097	554.68	587.68	612.69
0.7293	594.86	630.25	657.08
0.7490	641.31	679.47	708.39
0.7686	695.64	737.03	768.40
0.7882	760.02	805.25	839.52
0.8078	837.54	887.38	925.14
0.8274	932.67	988.16	1030.22
0.8470	1052.17	1114.78	1162.22
0.8666	1206.80	1278.60	1333.02
0.8862	1414.70	1498.88	1562.67
0.9058	1709.15	1810.85	1887.92
0.9254	2158.38	2286.81	2384.14
0.9450	2927.97	3102.19	3234.22
0.9646	4550.52	4821.29	5026.49
0.9842	10206.24	10813.54	11273.77

TABLE 9. Transmitter fault recovery maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	MINIMUM DELAY	MAXIMUM DELAY
0.1608	191.84	219.30	247.89
0.1804	196.43	224.54	253.82
0.2000	201.25	230.04	260.04
0.2196	206.30	235.82	266.58
0.2392	211.62	241.90	273.45
0.2588	217.22	248.30	280.68
0.2784	223.12	255.05	288.31
0.2980	229.35	262.17	296.36
0.3176	235.94	269.70	304.87
0.3372	242.92	277.68	313.89
0.3568	250.32	286.15	323.46
0.3764	258.19	295.14	333.63
0.3960	266.58	304.72	344.46
0.4156	275.52	314.95	356.02
0.4353	285.08	325.88	368.38
0.4549	295.34	337.60	381.63
0.4745	306.36	350.20	395.86
0.4941	318.23	363.77	411.21
0.5137	331.06	378.43	427.78
0.5333	344.96	394.33	445.75
0.5529	360.09	411.62	465.30
0.5725	376.61	430.50	486.64
0.5921	394.71	451.19	510.03
0.6117	414.64	473.98	535.78
0.6313	436.69	499.18	564.28
0.6509	461.21	527.22	595.97
0.6705	488.66	558.59	631.43
0.6901	519.58	593.93	671.39
0.7097	554.68	634.05	716.74
0.7293	594.86	679.98	768.66
0.7490	641.31	733.09	828.69
0.7686	695.64	795.19	898.89
0.7882	760.02	868.79	982.08
0.8078	837.54	957.40	1082.25
0.8274	932.67	1066.14	1205.17
0.8470	1052.17	1202.74	1359.59
0.8666	1206.80	1379.50	1559.39
0.8862	1414.70	1617.15	1828.04
0.9058	1709.15	1953.74	2208.52
0.9254	2158.38	2467.26	2789.01
0.9450	2927.97	3346.98	3783.44
0.9646	4550.52	5201.73	5880.07
0.9842	10206.24	11666.81	13188.23

Scenario Study

Since the analytic model has been set up, it is possible to perform many different scenario studies on the network. For general scenario studies, it may be assumed that certain number of stations in the network suffers certain maintenance delays. This effect can be represented as

$$Dmi = Dri*Nri + Dsa*Nsa + Dsd*Nsd + Dtr*Ntr + Drf*Nrf + Dtf*Ntf \qquad (5.1)$$

where,

Dri:	ring initialization delay
Dsa:	station addition delay
Dsd:	station deletion delay
Dtr:	token recovery delay
Drf:	receiver fault recovery delay
Dtf:	transmitter fault recovery delay
Nri:	number of ring initializing stations
Nsa:	number of stations adding a station
Nsd:	number of stations deleting themselves
Ntr:	number of stations recovering token
Nrf:	number of stations with faulty receivers
Ntf:	number of stations with faulty transmitters

Now, any variables in the equation (5.1) can be modified to implement certain maintenance delays caused by the scenarios of the network maintenance.

It is a matter of vital importance for the working system, however, to evaluate the worst case network performance. This is because the maximum allowable delay suffered by the network is usually one of the given parameters of the system specification. And, in this token bus network the worst case would be such that each station in the network loses the token after transmission of its data and goes through ring initialization. Delay-channel utilization curve in this case is shown in Figure 16 and Table 10 contains the actual values of worst case delays.

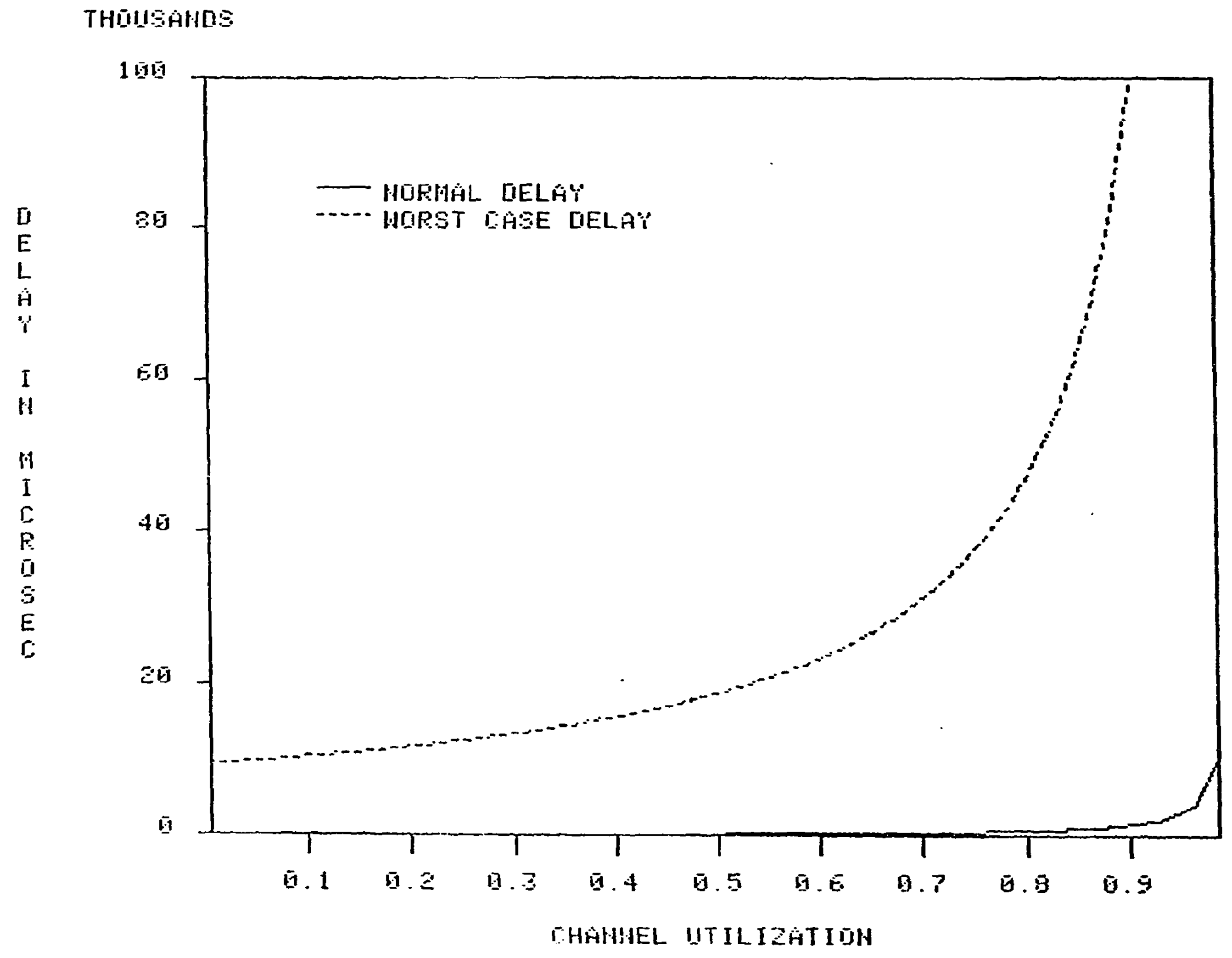


FIGURE 16. The worst case maintenance delay

TABLE 10. The worst case maintenance delay

CHANNEL UTILIZATION	NORMAL DELAY	DELAY
0.1608	191.84	11211.66
0.1804	196.43	11479.85
0.2000	201.25	11761.19
0.2196	206.30	12056.66
0.2392	211.62	12367.36
0.2588	217.22	12694.50
0.2784	223.12	13039.42
0.2980	229.35	13403.60
0.3176	235.94	13788.71
0.3372	242.92	14196.60
0.3568	250.32	14629.37
0.3764	258.19	15089.34
0.3960	266.58	15579.18
0.4156	275.52	16101.90
0.4353	285.08	16660.90
0.4549	295.34	17260.12
0.4745	306.36	17904.04
0.4941	318.23	18597.87
0.5137	331.06	19347.65
0.5333	344.96	20160.41
0.5529	360.09	21044.47
0.5725	376.61	22009.61
0.5921	394.71	23067.53
0.6117	414.64	24232.29
0.6313	436.69	25520.92
0.6509	461.21	26954.32
0.6705	488.66	28558.30
0.6901	519.58	30365.27
0.7097	554.68	32416.34
0.7293	594.86	34764.58
0.7490	641.31	37479.57
0.7686	695.64	40654.58
0.7882	760.02	44417.30
0.8078	837.54	48947.55
0.8274	932.67	54506.90
0.8470	1052.17	61490.83
0.8666	1206.80	70527.56
0.8862	1414.70	82677.88
0.9058	1709.15	99885.98
0.9254	2158.38	126140.18
0.9450	2927.97	171116.23
0.9646	4550.52	265941.66
0.9842	10206.24	596472.81

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

Conclusions from this research and some related future work are presented in this chapter. The conclusions section includes the review of the research and discussions on the results produced. Simulation of the system, reliability of maintenance functions, and statistical performance analysis are suggested for the future work.

Conclusions

An analytic model for the token bus protocol with the consideration of maintenance functions has been proposed. Some sample studies are also presented to demonstrate the effects of maintenance functions on the network performance, especially on the average message delay. Analysis of the performance is done with the average message delay and channel utilization which are two most commonly accessed network parameters in this research area.

The problem of analyzing real token bus systems was not familiar to researchers in network performance field. This was simply because there were no urgent necessities for this. As the technique of manufacturing improves, however, the high speed token bus system comes into existence and the necessity for the analysis of the system in action grows rapidly.

The system in operation is different from the conventional model used in analysis in that the real system frequently breaks down and requires maintenance to be operational again. This is really the case if the system is a prototype.

Now, this brings up an issue of maintenance functions in the token bus network. Even though maintenance functions are mentioned in the standard, they are not very well separated from normal operations of the system. The performance analysis of the token bus protocol has been done extensively lately, but few researches consider the effect of maintenance functions into their model.

The problem of maintenance functions attacked indirectly in this dissertation. First, an analytic model for normal operational mode is developed. This model is a unique model in the sense that it contains a variable, Toi in (4.15), whose value would be decided later to include the effect of the maintenance in the system.

Overhead times are used to quantify the effects of the maintenance functions and reflected back into the normal mode analytic model previously defined. This overhead is converted into maintenance delays expressed in (5.1) which is generalized to encompass all types of maintenance. The basic idea in the analytic model, which can be applied to many different network situations, is flexibility.

Analytic model approach was chosen over simulation model with two reasons. First, there is no good simulation language available for computer network simulation. Even if the SLAM, a simulation language for alternative modeling [45], has been used in this area, it has inherent problems such as limited number of files. The second reason is that the token bus system manufacturer, Rockwell Collins Co., has already developed a simulation model.

The result of the performance analysis based on the analytic model shows close similarity with the simulation result supplied by Rockwell Company. As can be seen in Figure 10 through 15, maintenance functions affect the performance of the system greatly. The average message delay is increased considerably in some cases. The increment varies from 3.8% in the station deletion case to 361% in the ring initialization case. The worst case analysis in Chapter 5 shows 5744% increment in message delay which establishes the upper bound of delay.

In conclusion, the analytic model developed is justified and is flexible enough to encompass the effects of maintenance functions in the token bus protocol.

Future Work

<u>Simulation modeling</u>

A simulation is a very useful tool in performance analysis and is usually used to verify the predicted performance from the analytic model. The development of the simulation model for the token bus protocol with maintenance functions would be worthwhile.

More challenging work might be developing a new simulation language for computer networking, since few of the simulation languages in the current market is developed for specific usage in this area.

<u>Reliability of maintenance functions</u>

Maintenance functions considered in this research are not a complete set of functions in the sense that there might be some catastrophic failures which cannot be recovered from with these maintenance functions. For example, the token bus protocol relies on various timers to initiate maintenance functions and if something goes wrong with these timers, there is no guarantee that the network would be completely recovered.

This brings some attention to redundancy in the system and reliability analysis of the system. This area needs intensive research, because it could be an important issue in real time implementation of the system.

Statistical performance analysis

If a token bus system were actually built, it would be possible to gather real time information on message delays and network throughputs. Also, the delays due to maintenance can be monitored from the system. With some statistical process, the average maintenance delay and the distribution of maintenances could be calculated from the observations.

BIBLIOGRAPHY

1. Abramson, N. "The ALOHA System - Another Alternative for Computer Communications." AFIPS Conference Proceedings 37 (1970): 281-285.

2. Metcalf, R. M. and Boggs, D. R. "Ethernet: Distributed Packet Switching for Local Computer Networks." Communications of ACM 9, No. 7 (July 1976): 395-404.

3. Sundstrom, R. J. "Formal Definition of IBM's System Network Architecture." Proceedings of the National Telecommunications Conference (Dec. 1977): 03A1-1-03A1-7.

4. Bochmann, G. V. "Finite State Description of Computer Protocol." Computer Networks 2 (Oct. 1978): 362-372.

5. Lam, S. S. "A Carrier Sense Multiple Access Protocol for Local Networks." Computer Networks 4 (1980): 21-32.

6. Black, U. D. Data Communications, Networks, and Distributed Processing. Reston, Virginia: Reston Publishing Company Inc., 1983.

7. Chorafas, D. N. Designing and Implementing Local Area Networks. New York, New York: McGraw-Hill Book Company, 1984.

8. Stallings, W. Local Networks: An Introduction. New York, New York: Macmillan Publishing Company, 1984.

9. Konheim, A. G. and Meister, B. "Waiting Lines and Time in a System with Polling." Journal of ACM 21, No. 3 (July 1974): 470-490.

10. Pawlikowski, K. "Message Waiting Time in a Packet Switching System." Journal of ACM 27, No. 1 (1980): 30-42.

11. Tobagi, F. A., Hunt, V. B. "Performance Analysis of Carrier Sense Multiple Access with Collision Detection." Proceedings of Local Area Communication Network Symposium (May 1979): 217.

12. Kleinrock, L. and Tobagi, F. A. "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple Access Modes and their Throughput-Delay Characteristics." IEEE Trans. on Communications 24, No. 12 (Dec. 1975): 1400-1416.

13. Kleinrock, L. and Tobagi, F. A. "Packet Switching in Radio Channels: Part III - Polling and Split-Channel Reservation Multiple Access." IEEE Trans. on Communications 25, No. 8 (Aug. 1976): 832-845.

14. Kuehn, P. J. "Multiqueue Systems with Nonexhaustive Cyclic Service." Bell System Technical Journal 58 (Mar. 1979): 671-698.

15. Shoch, J. F. and Hupp, J. A. "Measured Performance of an Ethernet Local Network." Communications of ACM 23 (Dec. 1980): 711-721.

16. Cherukuri, R., Li, L., and Louis, L. "Evaluation of Token Passing Scheme in Local Area Networks." Proceedings of Computer Networking Symposium (Dec. 1982): 57-68.

17. Bux, W. "Local Area Subnetworks: A Performance Comparison." IEEE Trans. on Communications 29 (Oct. 1981): 1465-1473.

18. Bux, W. "Performance Issues in Local Area Network." IBM System Journal 23, No. 4 (1984): 351-374.

19. Stallings, W. "Local Network Performance." IEEE Communication Magazine 22, No. 2 (Feb. 1984): 27-36.

20. Sachs, S. R. and Kan, K. "Performance of a Token Bus Protocol and Comparison with other LAN Protocols." IEEE 10th Conference on Local Computer Networks (1985): 46-51.

21. Ulug, M. E. "Calculation of Waiting Times for a Real Time Token Passing Bus." Proceedings of Computer Networking Symposium (Dec. 1983):

22. Ulug, M. E. "Comparison of Token Holding Time Strategies for a Static Token Passing Bus." Proceedings of Computer Networking Symposium (Dec. 1984): 37-44.

23. Sastry, A. R. "Maximum Mean Data Rate in a Local Network with a Specified Maximum Source Message Load." Proceedings of IEEE INFOCOM 85 (Mar. 1985): 216-221.

24. Seidler, J. Principles of Computer Communication Network Design. New York, New York: Halsted Press, 1983.

25. Stallings, W. Data and Computer Communications. New York, New York: Macmillan Publishing Company, 1985.

26. Stuck, B. W. and Arthurs, E. A Computer & Communications Network Performance Analysis Primer. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1985.

27. Pickholtz, R. L. Local Area & Multiple Access Networks. Rockville, Maryland: Computer Science Press, 1986.

28. Hayes, J. F. Modelling and Analysis of Computer Communications Networks. New York, New York: Plenum Press, 1984.

29. Hammond, J. L. and O'Reilly, P. J. P. Performance Analysis of Local Computer Networks. Reading, Massachusetts: Addison-Wesley Publishing Company, 1986.

30. IEEE. IEEE Standard 802.3: CSMA/CD Access Method. Silver Spring, Maryland: IEEE Computer Society Press, 1984.

31. IEEE. IEEE Standard 802.4: Token Passing Bus Access Method. Silver Spring, Maryland: IEEE Computer Society Press, 1984.

32. IEEE. IEEE Standard 802.5: Token Ring Access Method. Silver Spring, Maryland: IEEE Computer Society Press, 1984.

33. Kleinrock, L. Queueing Systems. Vol. I: Theory. New York, New York: John Wiley, 1975.

34. Kleinrock, L. Queueing Systems. Vol. II: Computer Applications. New York, New York: John Wiley, 1975.

35. Farmer, W. D. and Newhall, E. E. "An Experimental Distributed Switching System to Handle Bursty Computer Traffics." Proceedings of the ACM Symposium, Problems in Optimization Data Communication System. (1969): 1-34.

36. Cooper, R. B. and Murray, G. "Queues served in Cyclic Order." Bell System Technical Journal 48, No. 3 (Mar. 1969): 675-689.

37. Cooper, R. B. "Queues Served in Cyclic Order: Waiting Times." Bell System Technical Journal 49, No. 3 (Mar. 1970): 399-413.

38. Hayes, J. F. and Sherman, D. N. "A study of Data Multiplexing Techniques and Delay Performance." Bell System Technical Journal 51 (Nov. 1972): 1985-2011.

39. Halfin, S. "An Approximate Method for Calculating Delays for a Family of Cyclic Type Queues." Bell System Technical Journal 58, No. 10 (Dec. 1975): 1733-1754.

40. Hayes, J. F. "Performance Models of an Experimental Computer Communications Network." Bell System Technical Journal 53, No. 2 (Feb. 1974): 225-259.

41. Spragins, J. D. "Simple Derivation of Queueing Formulas for Loop Systems." IEEE Trans. on Communications 23 (Apr. 1977): 446-448.

42. Fayolle, G., Gelembe, E., and Pujolle, G. "An Analytic Evaluation of the Performance of the 'Send and Wait' Protocol." IEEE Trans. on Communications 26, No. 3 (Mar. 1978): 313-320.

43. Swarz, G. B. "Polling in a Loop System." Journal of ACM 27, No. 1 (Jan. 1980): 42-59.

44. Bux, W., Closs, T., Janson, P., Kummerle, K., and Muller, H. S. "A Reliable Token System for Local-Area Communication." National Telecommunication Conference (Dec. 1981): A2.2.1-A2.2.6.

45. Pritsker, A. A. B. Introduction to Simulation and SLAM II. New York, New York: Halsted Press, 1984.

ACKNOWLEDGEMENT

I wish to express my deepest appreciations to my major professor, Dr. Arthur V. Pohm and co-major professor, Dr. Douglas W. Jacobson for providing me an opportunity to do this research and for their excellent guidance and encouragement throughout the research. Special thanks are extended to Dr. Thomas A. Barta, Dr. Grover R. Brown, Dr. James A. Davis, and Dr. Terry A. Smay for their valuable suggestions as my graduate advisory committee.

I am also very grateful to Dr. J.O. Kopplin, chairman of the Electrical and Computer Engineering Department and Mr. Jack Rector of the Rockwell Collins International Co. for the financial support during the research.

I also sincerely thank my wife, Jong-Ok for her remarkable patience and sacrifice, and my little son, Jung-Hoon without whose help the typing of this dissertation would have been done months earlier. Finally, I wish to dedicate this Ph.D. dissertation to my parents for their endless love and inspiration in all my life. And I owe them this.

APPENDIX A. MAC CONTROL FRAME FORMATS

The following frames are used to control the access of the bus and to keep track of maintenance in token bus protocol system.

- Claim_token: The frame has a data unit whose value is arbitrary and whose length is 0, 2, 4, or 6 times the system slot_time.

PREAMBLE	SD	00000000	DA	SA	arbitrary value	FCS	ED

- Solicit_successor_1: The frame has a DA = the contents of the station's next station register and a null data unit. One response_window always follows this frame.

PREAMBLE	SD	00000001	DA	SA	FCS	ED

- Solicit_successor_2: The frame has DA = the contents of the staion's NS or TS register and a null data unit. Two response_windows always follow this frame.

PREAMBLE	SD	00000010	DA	SA	FCS	ED

- Who_follows: The frame has a data unit = the value of the station's NS register. The frame and length of the data unit is the same as a source address. Three response_windows always follow this frame.

PREAMBLE	SD	00000011	DA	SA	value NS	FCS	ED

- Resolve_contention: The frame has null data unit. Four response_windows always follow this frame.

PREAMBLE	SD	00000100	DA	SA	FCS	ED

- Set_successor: The frame has DA = the SA of the last frame received, and data unit = the value of the station's NS or TS register. The format and length of the data unit is the same as that of a source address.

PREAMBLE	SD	00001100	DA	SA	new value of NS	FCS	ED

- Token: The frame has DA = the contents of the station's NS register and has a null data unit.

PREAMBLE	SD	00001000	DA	SA	FCS	ED

APPENDIX B. BENCHMARK INPUT DATA

This is the benchmark input data used for performance analysis in Chapter 5.

Number of Messages	Message Length (Words)	Arrival Interval (μsec)
15	10.00000	200000.0
	10.00000	19607.84
	20.00000	79365.08
	10.00000	200000.0
	20.00000	79365.08
	10.00000	200000.0
	20.00000	19607.84
	20.00000	41666.67
	10.00000	200000.0
	20.00000	19607.84
	4.000000	79365.08
	20.00000	19607.84
	20.00000	19607.84
	4.000000	79365.08
	20.00000	806451.6
24	10.00000	79365.08
	14.00000	20408.16
	250.0000	251889.2
	20.00000	20408.16
	30.00000	41666.67
	20.00000	20408.16
	20.00000	20408.16
	250.0000	66225.16
	10.00000	79365.08
	150.0000	39682.54
	24.00000	20408.16
	20.00000	200000.0
	20.00000	20408.16
	20.00000	20408.16
	10.00000	806451.6
	250.0000	251889.2
	40.00000	806451.6
	250.0000	251889.2
	20.00000	20408.16
	10.00000	200000.0
	20.00000	79365.08
	250.0000	251889.2

(continued)

Number of Messages	Message Length (Words)	Arrival Interval (μsec)
	10.00000	20408.16
	250.0000	52910.06
12	40.00000	200000.0
	30.00000	20408.16
	14.00000	20408.16
	10.00000	806451.6
	56.00000	79365.08
	250.0000	100806.5
	54.00000	79365.08
	200.0000	80645.16
	30.00000	41666.67
	24.00000	19607.84
	4.000000	200000.0
	250.0000	52910.06
3	20.00000	200000.0
	10.00000	806451.6
	250.0000	125944.6
14	10.00000	200000.0
	30.00000	19607.84
	30.00000	19607.84
	20.00000	200000.0
	24.00000	19607.84
	30.00000	19607.84
	30.00000	19607.84
	30.00000	19607.84
	10.00000	806451.6
	30.00000	19607.84
	250.0000	537634.4
	20.00000	19607.84
	30.00000	19607.84
	30.00000	19607.84
4	10.00000	200000.0
	10.00000	200000.0
	10.00000	200000.0
	10.00000	806451.6
6	10.00000	20408.16
	4.000000	200000.0

(continued)

Number of Messages	Message Length (words)	Arrival Interval (μsec)
	4.000000	200000.0
	10.00000	806451.6
	10.00000	20408.16
	10.00000	20408.16
5	20.00000	19607.84
	20.00000	19607.84
	20.00000	19607.84
	20.00000	19607.84
	10.00000	806451.6
11	4.000000	20408.16
	4.000000	20408.16
	20.00000	250000.0
	4.000000	20408.16
	4.000000	20408.16
	4.000000	20408.16
	4.000000	20408.16
	20.00000	200000.0
	4.000000	20408.16
	4.000000	20408.16
	4.000000	20408.16
10	30.00000	200000.0
	100.0000	79365.08
	100.0000	19607.84
	250.0000	133868.8
	250.0000	133868.8
	100.0000	19607.84
	20.00000	806451.6
	30.00000	200000.0
	30.00000	200000.0
	30.00000	200000.0
3	40.00000	806451.6
	10.00000	200000.0
	20.00000	806451.6

(continued)

Numer of Messages	Message Length (words)	Arrival Interval (μsec)
22	100.0000	200000.0
	10.00000	20408.16
	10.00000	200000.0
	10.00000	79365.08
	10.00000	200000.0
	40.00000	200000.0
	10.00000	200000.0
	30.00000	200000.0
	30.00000	200000.0
	4.000000	200000.0
	4.000000	806451.6
	10.00000	200000.0
	10.00000	806451.6
	40.00000	806451.6
	60.00000	200000.0
	10.00000	20408.16
	20.00000	200000.0
	10.00000	200000.0
	20.00000	20408.16
	10.00000	200000.0
	100.0000	806451.6
	20.00000	200000.0
14	20.00000	19607.84
	30.00000	41666.67
	20.00000	19607.84
	20.00000	19607.84
	20.00000	19607.84
	20.00000	19607.84
	50.00000	19607.84
	10.00000	200000.0
	20.00000	19607.84
	250.0000	96153.85
	20.00000	19607.84
	250.0000	96153.85
	70.00000	19607.84
	150.0000	384615.4
2	10.00000	806451.6
	10.00000	806451.6

(continued)

Number of Messages	Message Length (Words)	Arrival Interval (μsec)
5	4.000000	19607.84
	10.00000	19607.84
	10.00000	806451.6
	40.00000	19607.84
	4.000000	19607.84
7	10.00000	200000.0
	30.00000	79365.08
	10.00000	200000.0
	10.00000	79365.08
	30.00000	79365.08
	20.00000	806451.6
	30.00000	79365.08