CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r. 82/08)

Canada

HYMAP: A NEW PROTOCOL FOR LOCAL AREA NETWORKS

by

Miguel Felix Rios Ojeda

A thesis
presented to The University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
The Department of Electrical Engineering, Faculty of
Science and Engineering.

OTTAWA, Ontario, 1983

I hereby declare that I am the sole author of this thesis.

I authorize The University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Miguel Felix Rios Ojeda

I further authorize The University of Ottawa to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Miguel Felix Rios Ojeda

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

This thesis is addressed to the designing aspects of a number of protocols, aimed to be used in Local Area Networks.

A Graphical method is introduced to analyze and develop the protocols. This graphical approach eases the analysis and design processes.

Several protocols are developed and analyzed, that improve the performance of some well known protocols, such as CSMA/CD and GBRAM.

A new hybrid multiple access protocol, HYMAP, is introduced. This protocol combines features of both CSMA/CD and a collision-free protocol. The voice performance of HYMAP is studied using simulation.

Simulation analysis is also used to compare the performance of all the protocols discussed in the thesis.

The results indicate that HYMAP performs better than most of the protocols studied, including CSMA/CD and GBRAM.

ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his thesis supervisor Dr. Nicolas Georganas and to his family, for their friendship and total support. He also wants to thank Dr. Sorin Cohn-Sfetcu of B.N.R. for his comments.

He also highly appreciates the help and friendship of all his colleagues, especially the one of Mr. Kevin Wong and Mr. Esper Bitar.

The invaluable help of the university staff is also acknowledged.

The economical support of the University of Ottawa and especially the one of the Universidad Catolica de Chile is deeply recognized.

Finally, the author wishes to highlight the very especial support he received from his beloved wife and daughter, without which this work would not have been possible.

CONTENTS

Chapter I.

INTRODUCTION

Local Area Networks (LANs) generally have the following characteristics [1,7,13,33]:

1- A geographical area of not more than a few square kilometers;

2- A total data rate exceeding 1 Mb/sec and

3- Ownership by a single organization.

LANs are evolving from a need to interconnect local mainframes with a number of minicomputers, microcomputers, terminals and other peripheral devices.

The LAN's role is to provide a "friendly" interface to each of these devices, so that they may be unified into a more powerful set of tools, without unduly affecting the performance of each individual device.

The functionality of the LAN is determined both by the network interface design and the network interface protocol, affecting ease of use, transition capability, vendor independence, cost and flexibility of the network.

Another reason for the interest in LAN is the possibility of exploiting the advantages of functionally distributed

computing. In this case, some of the devices perform specific functions, such as file storage, data base management, terminal handling, etc. By having different devices perform different tasks, the goal is to make the implementation simpler or more efficient.

LANs differ from long-haul networks in several ways. Long-haul networks use the public telephone network due to economic or legal reasons. LANs generally use a high-bandwidth cable, so bandwidth is no longer a precious resource as it is in long-haul networks. Another major difference is the ability of each station in LANs to sense the state of the broadcast channel before attempting to use it.

In this thesis, we will be mainly interested in single hop multiple access broadcast networks as typified by ALOHA, SATNET and ETHERNET. Here a single transmission medium is shared by all devices in the system. The medium is allocated to each device for the time required to transmit a single packet. Each station is connected to the common channel through a smart interface, which listens to all transmissions and absorbs packets addressed to it.

So the problem is to share a high-bandwidth channel by a number of independent users, trying, at the same time, to optimize the use of the medium. Short propagation delays and high data rates are the main characteristics that are exploited in the design of multiple access schemes for LANs.

Multiple access schemes are evaluated according to various criteria: high bandwidth utilization, low message delays, the ability to support different types of traffic or different priorities, robustness with respect to errors, etc.

Multiple access protocols are usually grouped into the following classes [4]:

1- Fixed assignment techniques, which allocate the channel bandwidth to the users in a static fashion, independently of their activity. Examples are FDMA, STDMA and CDMA.

2- Random access techniques, which provide the entire bandwidth to the users as a single entity to be accessed randomly. Examples are ALOHA and CSMA/CD.

3- Demand assignment techniques, which require that explicit control information regarding the users' need for the channel be exchanged. The decision-making process can be centralized or distributed, and each user individually executes an algorithm based on control information exchanged among the users. Examples of centralized protocols are polling and probing; distributed protocols are MSAP [34], BRAM [18], GBRAM [15], etc.

4- Adaptive strategies and mixed modes, which include all techniques that can not be classified in the previous classes.

The thesis will discuss designing aspects of a number of protocols of the second and third classes.

The main contributions of the thesis are:

1- Utilization of a basic graphical approach that eases the analysis and design of protocols.

2- Development and simulation analysis of a number of protocols that improve the performance of some well known protocols.

3- Detailed description and simulation analysis of HYMAP, a new hybrid multiple access protocol that combines features of both CSMA/CD and a collision free protocol.

The organization of the thesis is as follows: In chapter II, the graphical approach model is introduced and used to analyze two well known protocols. The protocols, CSMA/CD [5] and GBRAM [15], are briefly presented together with simulation results.

In chapter III, a brief description of a number of protocols that we developed is made, together with simulation results that show their relative advantages and disadvantages.

In chapter IV, a detailed description of HYMAP is performed. The discussion includes practical implementation aspects as well as simulation results. It also presents a voice performance study of the protocol.

Finally in chapter V, the conclusions of the work done and provisions for future work are presented.

Chapter II

ANALYSIS AND SIMULATION OF LAN PROTOCOLS

2.1 INTRODUCTION

In this chapter we introduce the model that we will use to analyze the different protocols that we will discuss. We also present a brief description of some well known protocols as Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and Group Broadcasting Recognizing Access Method (GBRAM) [5,15].

Simulation analysis is used to describe the characteristics of both protocols.

There are some general assumptions we make in all the protocols considered in this chapter and the rest of the thesis. The assumptions are:

1- The packet size is considered constant.

2- No acknowledgement traffic is assumed.

3- Source-destination pairs are not defined.

4- The traffic is assumed uniformly distributed among stations.

5- The stations are assumed uniformly distributed along the cable.

6- All queued packets in a station are delayed together and the queue is not limited.

7- The channel is assumed noiseless.

8- There is no capture effect on the channel , so interference between packets is allways destructive.

2.2 - MODEL OF ANALYSIS

We introduce here the graphical model that we will use to analyze the different protocols we will discuss. The model assumes a cable bus system to which N stations are connected at different points of its length L (see fig 2.1).

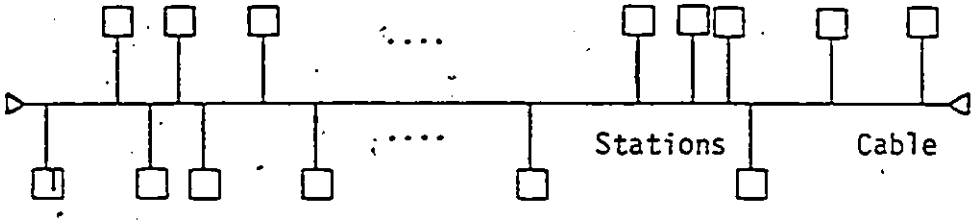


Fig 2.1 Structure of the Cable Bus System.

For simplicity and without loss of generality, we will assume the stations are uniformly distributed along the cable. Furthermore, we will also assume that only five

stations are connected to the system in the examples we will use to explain how each protocol works.

The time-distance diagram is then a graphical representation of what is going on the channel (see fig 2.2).

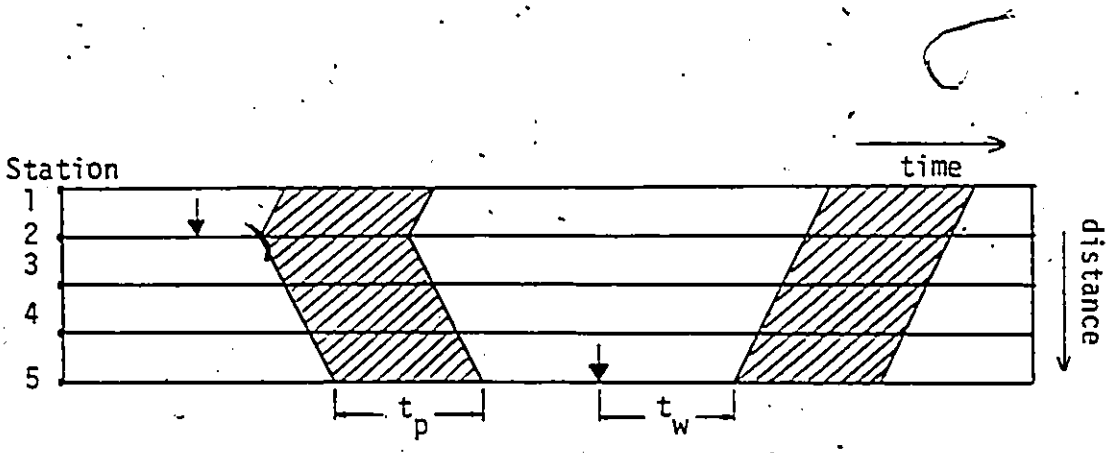


Fig 2.2 Time-distance Diagram.

As we can see in fig 2.2, the y-axis represents the distance along the cable. Each station position is indicated as a value of the distance axis. The x-axis represents the time scale at each position of the cable. Arrows will be used to represent information arrivals (packets) to the channel. As soon as a packet is allowed to go onto the channel (after a waiting delay t_w), it propagates towards the ends of the cable, where it is absorbed by cable

terminating loads. The packet length is assumed constant and equal to t_p sec. So we see from fig 2.2 the time where, at each station position, the interaction is taking place.

This model has the advantage of showing clearly at any position on the cable the time of occurrence of the different events. This can be very useful in the analysis of the effects of interference over the information and in the design of protocols to meet certain objectives, as we will show.

2.3 THE M/D/1 MODEL

We will define the approach for designing a LAN protocol by first considering the ideal situation and then discussing the different protocols and comparing them to that ideal model [17,31].

The ideal situation in a LAN is that where each station, if it has something to transmit, can use the channel without delay, when the system is idle, or has to delay its transmission for the amount of time just necessary for the system to become idle. The situation is pictured in fig 2.3, from where we can observe the channel utilization is the best possible.

The situation is ideal because each station knows instantaneously the fact another station has started a

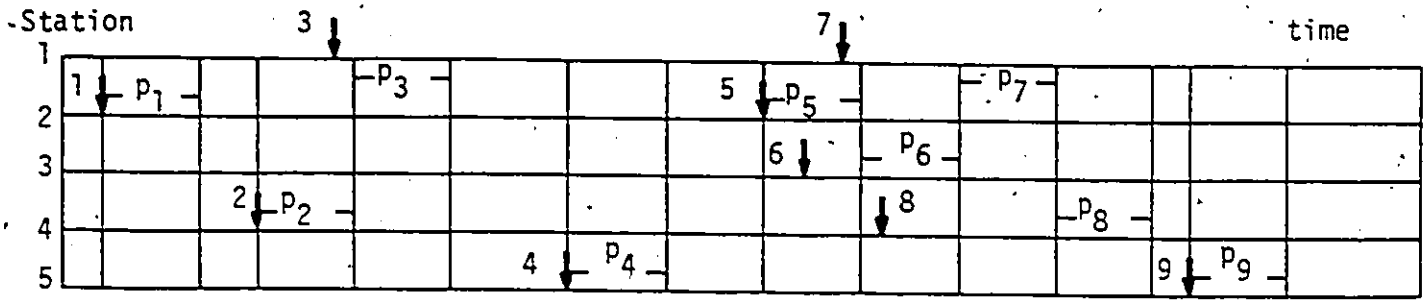


Fig 2.3 M/D/1 Traffic Characteristic.

transmission, that is, an infinite propagation speed is assumed.

For the situation of fig 2.3, where the arrivals are considered to be exponentially distributed and the service time is constant (fixed-length packets), the waiting delay in terms of the throughput (S) is given by the well known expression for the M/D/1 queue:

$$\text{Waiting delay} = St_p / 2(1-S) \qquad (2.1)$$

where $S = \lambda t_p$ is the channel utilization or throughput, λ is the packet arrival rate in packets/sec and t_p is the packet length in sec/packet.

The M/D/1 delay-throughput characteristic is shown in fig 2.4. We can see there that even in this ideal situation as

the throughput approaches 1.0, the waiting delay approaches infinity.

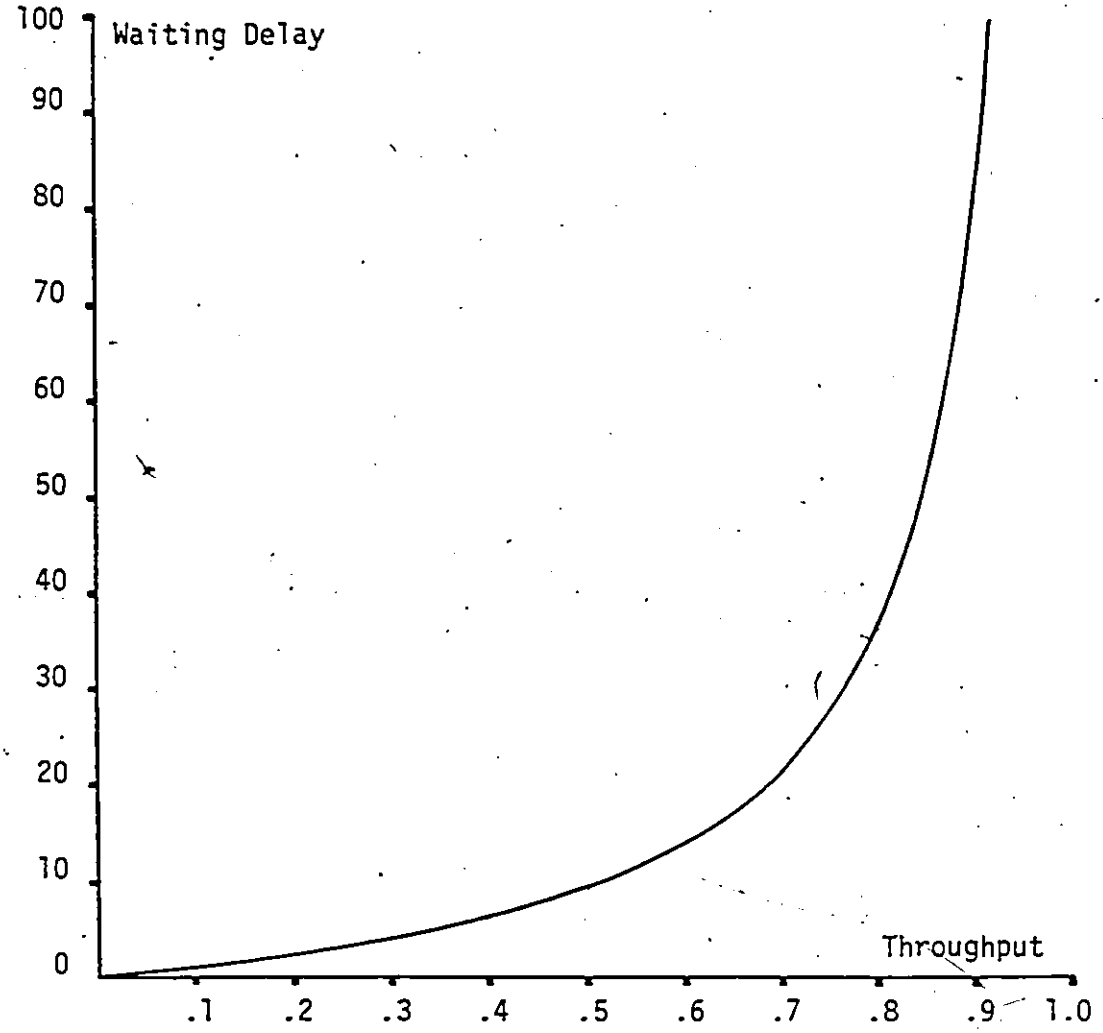


Fig 2.4 M/D/1 Delay-Throughput Characteristic.

2.4 CARRIER SENSE MULTIPLE ACCESS WITH COLLISION DETECTION (CSMA/CD)

In this scheme [5,8,9,10,11,20], a station wishing to transmit is said to contend for use of the common shared communications channel until it acquires the channel; once the channel is acquired the station uses it to transmit a packet.

The acquisition scheme is as follows: Stations check whether the network is busy (that is, use carrier sense) and defer transmission of their packet until the channel is sensed idle. When quiet is detected, the deferring station immediately begins to transmit. During transmission, the station monitors its own transmission, checking for interference from other transmissions (collision detection). In a correctly functioning system, collisions occur only within the interval of time needed by each station to detect the beginning of the transmission of another station. This time interval is a function of the end-to-end propagation delay.

If no collisions occur during this time, the station has acquired the channel and can continue with the transmission of the rest of the packet.

To ensure that all parties to the collision have properly detected it, any station that detects a collision invokes a collision consensus enforcement procedure that briefly jams

the channel. Each transmitter involved in the collision then schedules its packet for retransmission at some later time.

To minimize repeated collisions, each station involved in a collision tries to retransmit at a different time by using a random delay. After some maximum number of collisions, the transmitter gives up and reports a suitable error back to the station (after 15 collisions in the Ethernet case).

The algorithm used in Ethernet is called Binary Exponential Back-off and its objective is to obtain delay periods, after a collision, that will reschedule each station at times quantized in steps at least as large as a collision interval [5]. This time quantization is called the retransmission slot time. To guarantee quick use of the channel, this slot time should be short; yet to avoid collisions it should be larger than a collision interval. Therefore, the slot time is usually set to the round-trip time of the channel and the real time delay is the product of some retransmission delay (a positive integer) and the slot time.

To minimize the probability of repeated collisions, each delay is selected as a random number from an interval between zero and some upper limit. The length of the interval is doubled with each successive collision. To avoid undue delays and slow response, the doubling is stopped

after 10 collisions. This is referred to as Truncated Binary Exponential Back-off.

The truncated binary exponential back-off algorithm approximates the ideal situation where the probability of transmission of a packet is 1/M, where M is the number of ready stations. The interval is truncated after M equals N, the maximum number of stations.

In Ethernet [25], the retransmission delay is a positive integer chosen from the interval indicated in table 2-1 and dependent on the number of collisions experienced.

No. of Collisions	Interval
1	[0,1]
2	[0,3]
3	[0,7]
4	[0,15]
.	.
.	.
9	[0,512]
10-15	[0,1023]

Table 2.1 Retransmission intervals.

The back-off algorithm restarts with a zero retransmission interval after a successful transmision.

One of the problems of the binary exponential back-off is that the probability of two stations, that have collided between them for the first time, colliding again between them is very high. After the first collision, each station

will choose an integer number, that is zero or one, at random. So the probability of both stations choosing the same number, and so colliding again, is 50 %. That is the reason other back-off algorithms have been proposed that use a linear incremental interval instead of an exponential one [23].

The expressions used for the back-off algorithms mentioned before are as follows (in FORTRAN IV notation):

a) Binary Exponential Back-off.

$$\begin{aligned} \text{LBAKOF} &= \text{INT}[\text{RAN}(\bar{N},\bar{M})*1023] && K >= 10 \\ &\quad \text{INT}[\text{RAN}(\bar{N},\bar{M})*(2**K-1)] && K < 10 \end{aligned} \qquad (2.2)$$

b) Linear Incremental Back-off.

$$\text{LBACKOF} = \text{INT}[\text{RAN}(\bar{N},\bar{M})*(4+6*K)] \qquad (2.3)$$

where K is the number of collisions experienced, $\bar{N}$ and $\bar{M}$ are integers that determine a random number $\text{RAN}(\bar{N},\bar{M})$.

Figure 2.5 shows the interval length in terms of the number of collisions experienced by the packet. It can be seen there that for a large number of collisions, the exponential back-off introduces larger delays as compared with the linear back-off. However, if the load is very high, only the exponential back-off will be stable because at most N stations can be ready at one time.

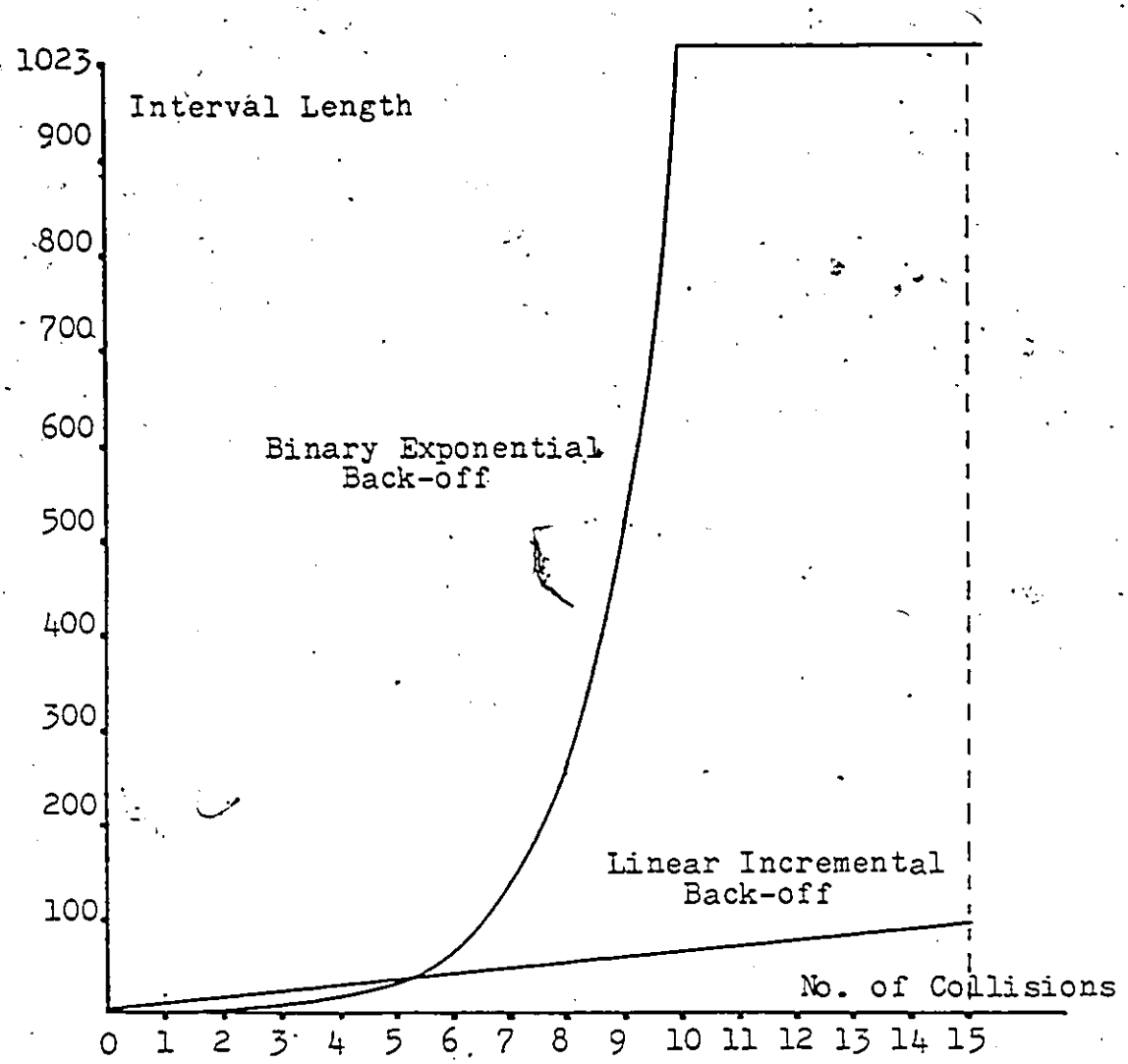


Fig 2.5 Interval Length of Binary Exponential and Linear Incremental Back-offs.

A program was designed to simulate CSMA/CD. The program is written in FORTRAN IV and represents very closely what CSMA/CD does. The program was written in that language because a general purpose language provides greater flexibility in the process of designing a simulation

program. The flow diagram of the program is presented in fig 2.6 and the program listing is included in the Appendix.

After inputting the simulation parameters, the program generates an event list containing both the arrival time of packets and the station number identification.

The arrival times are taken from an exponential distribution function of the form:

$$t(k+1) = t(k)-\log[RAN(\bar{N},\bar{M})]/XLAMDA \qquad (2.4)$$

where $RAN(\bar{N},\bar{M})$ is a random number uniformly distributed in the interval [0,1] and XLAMDA is the packet arrival rate in packets/sec. k is the packet index in the event list.

With the event list ordered according to the arrival time of each packet, the simulation starts.

The first thing the program does is to check if the simulation has ended, by testing if the simulation time allocated has passed. If so, the program prints the statistics of the simulation and stops.

In any other case, the program checks if a collision will occur between the first packet and any other packet on the event list. That collision can only take place if the first bit of the transmission of the first packet on the event list, arrives to any other station after those stations have started transmitting. Hence the unsafe period depends on the

relative position of the stations. Note furthermore that, if the second packet in the event list does not collide with the first one, that is not a guarantee that the third, fourth or so on packet, will not collide with the first one.

Now after detecting a collision, the program determines the exact time when the collision is detected by each station participating on it. Also the time where the channel becomes idle again (after the colliding stations jam the channel) is determined.

Following that, the back-off algorithm is applied to determine the new scheduled time for the packet retransmission. However, if the packet has experienced 16 collisions, it will be discarded by the program (the delay so far introduced to that packet is taken into account in the final statistics) and a new packet will be introduced to replace the deleted packet in the event list. After all packets participating on the collision are processed, a backlog routine is used in order to queue the packets of the same station to preserve the original order of arrival.

Also the arrivals that take place during the collision and jamming period are deferred until the channel becomes idle again.

After a period equal to one end-to-end propagation time, no collisions are possible since every station on the system

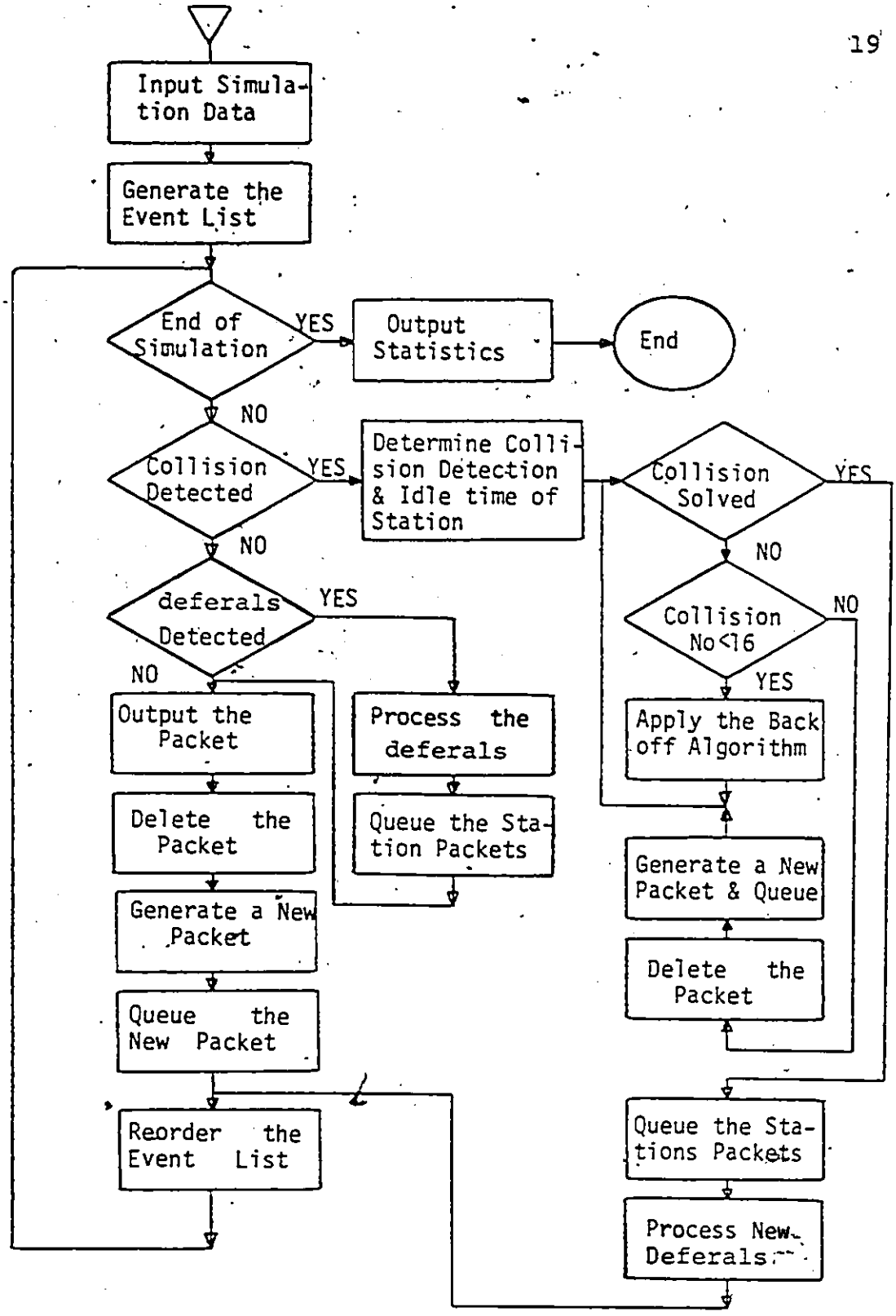


Fig 2.6 .Flow Diagram of the CSMA/CD Simulation Program.

has already sensed the transmission of the first packet. However, arrivals may occur during that transmission time, so those packets must defer the time of their transmissions until the channel is sensed idle. So the program checks that event and executes a deference routine.

The deferal routine adjusts the scheduled time of packets in the event list by calculating the exact time the channel will be sensed idle at each deferring station. Again, a backlog process is necessary to queue packets of the same station. After all deferring packets have been processed, the program outputs the first packet on the event list as explained below.

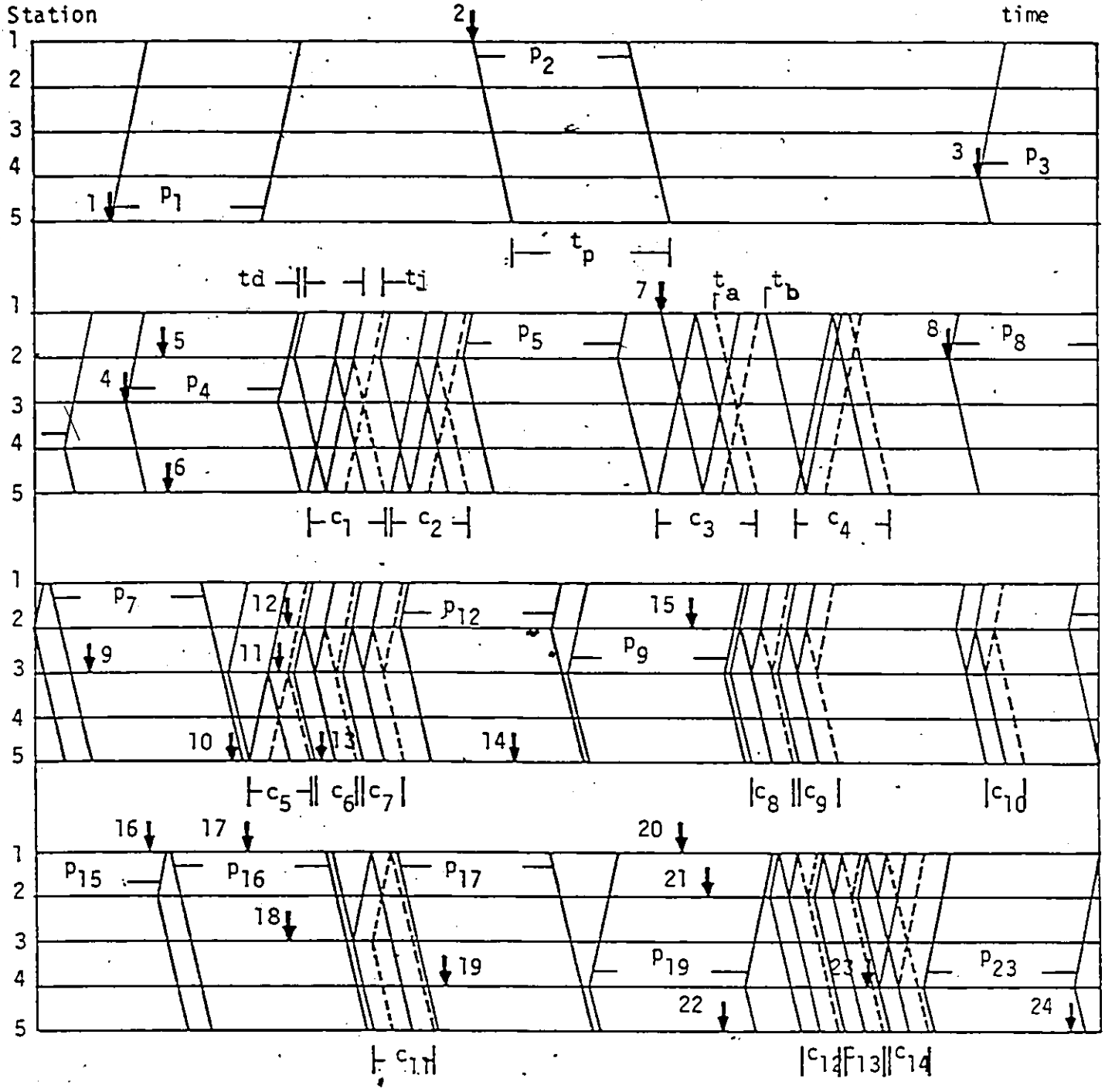
If neither collisions nor deferals are detected, the packet is free to use the channel. So the program outputs the packet, updating the statistical counters, discards that packet from the event list and generates a new arrival to substitute the deleted packet. The new arrival is placed at the bottom of the event list or is queued (in the same list) if another packet of the same station has a scheduled time (due to delays) greater than the arrival time of the new packet.

Finally, the event list is reordered (using a Shell-sort algorithm) because both the deference and collision processes may alter the order in it.

In figure 2.7 an example for five stations is presented. The period of time covered corresponds to 35 msec and in that period 24 packets arrive. We see there that packets 1-4 use the channel without delay. The packet size is represented by tp seconds. While station 3 is transmitting packet 4, two packets become ready (packets 5 and 6). These packets wait for the channel to become idle and start transmitting, generating a collision (segment c1 in fig 2.7). t_d is the minimum packet spacing.

After the collision is detected by the participating stations, they jam the channel for t_j seconds (the broken line represent the end of the jamming period). After the jamming is completed, each station chooses a random retransmission delay (integral multiples of twice the end-to-end propagation delay, taken from the intervals given in Table 2.1) according to the back-off algorithm. Unfortunately, in the example, both stations choose the same number and collide again (segment c2 in fig 2.7). This is due, as we explained before, to the discrete nature of the retransmission delay. After finishing the jamming, the stations choose different numbers and so packet 5 can use the channel without problems.

Now packet 6 defers to the transmission of packet 5 and then starts transmitting, colliding this time with the transmission of packet 7 from station 1 (segment c3 in fig



$t_p = 100\ \mu sec$ Packets Generated = 24 Throughput = 0.5123

$a/t_p = 0.25$ Successful Transmissions = 14

$t_j/t_p = 0.125$ Average Delay = 96.2 μsec Collisions = 14

$t_d/t_p = 0.05$

Fig 2.7 CSMA/CD Simulation Example for Five Stations.

2.7). Here we have an interesting problem: both stations choose different retransmission delays and yet they collide again (segment c4 in fig 2.7). This problem is caused by the jamming process: because station 1 chooses t_a for the retransmission of packet 7 but at that time the channel is busy (due to the jamming of station 5), the transmission of packet 7 can only start at t_b, thus causing the new collision.

After that, both stations choose longer retransmission delays (recall that packet 6 has experienced four collisions so far) and station 2 finds the channel idle and transmits successfully packet 8. Packet 7 also is transmitted after deferring to packet 8.

After that a new collision occurs between packet 6 and packet 9 (segment c5 in fig 2.7). Note here that at the moment of the collision station 5 has already two packets ready in its buffer. Since packet 6 has experienced five collisions, it chooses a very long retransmission delay and in fact back-logs packets 13,14,22 and 24, so at the end of the example it has six ready packets in its buffer.

The rest of the example is easy to follow. Note the triple collision occurring in segment c14, among stations 1,2 and 4.

The final results for the example show that only 14 packets were successfully transmitted during the period at a throughput of 0.5123 and an average delay of 96.2 microseconds (for the successful packets).

This same example will be used to compare all the protocols discussed in this thesis. Figures 2.8,2.9 and 2.10 presents results of the simulation concerning the delay-throughput characteristic, the delay distribution and the distribution of the number of collisions for CSMA/CD, for different values of the end-to-end propagation time to the packet length ratio (a/t_p).

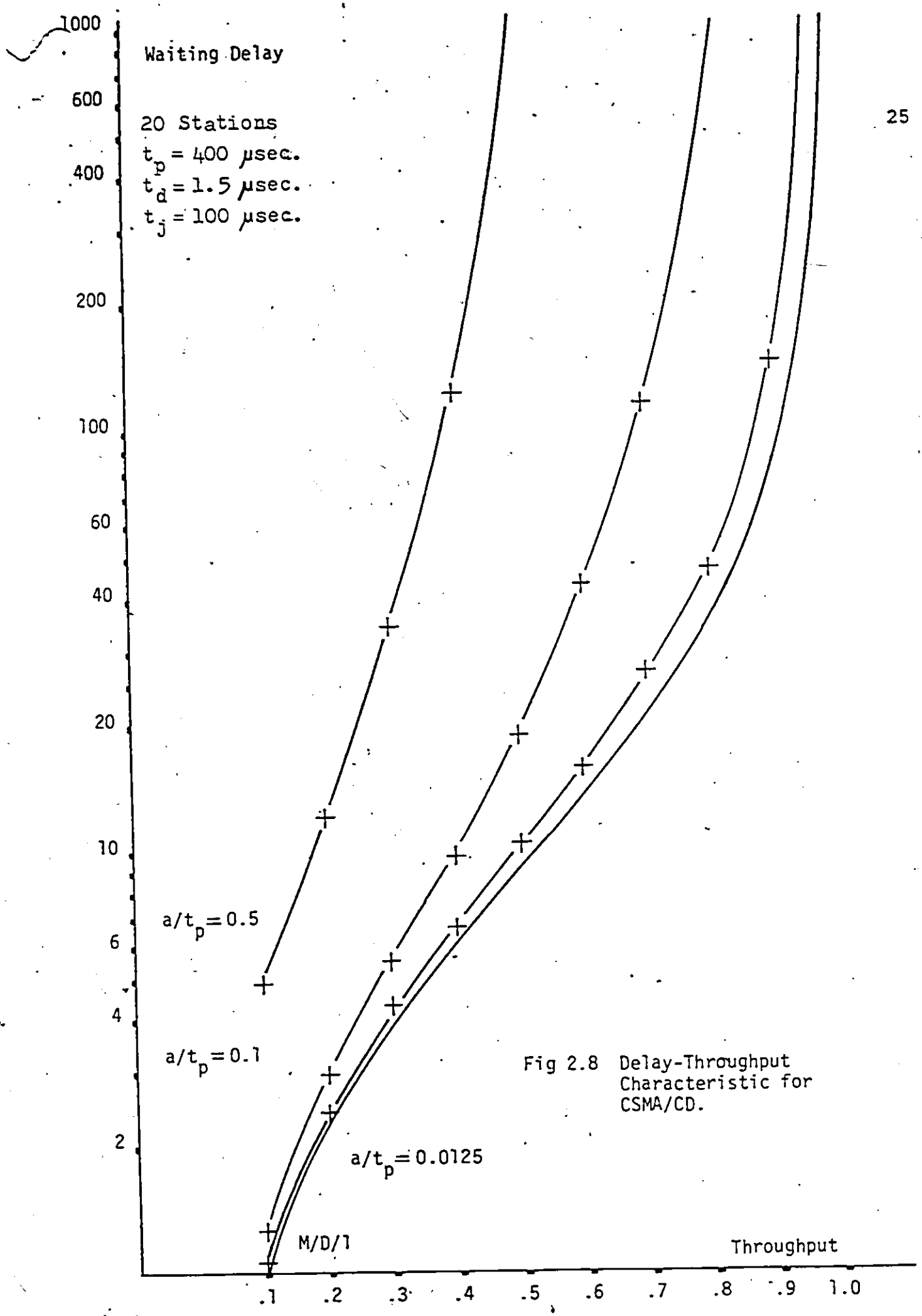


Fig 2.8 Delay-Throughput Characteristic for CSMA/CD.

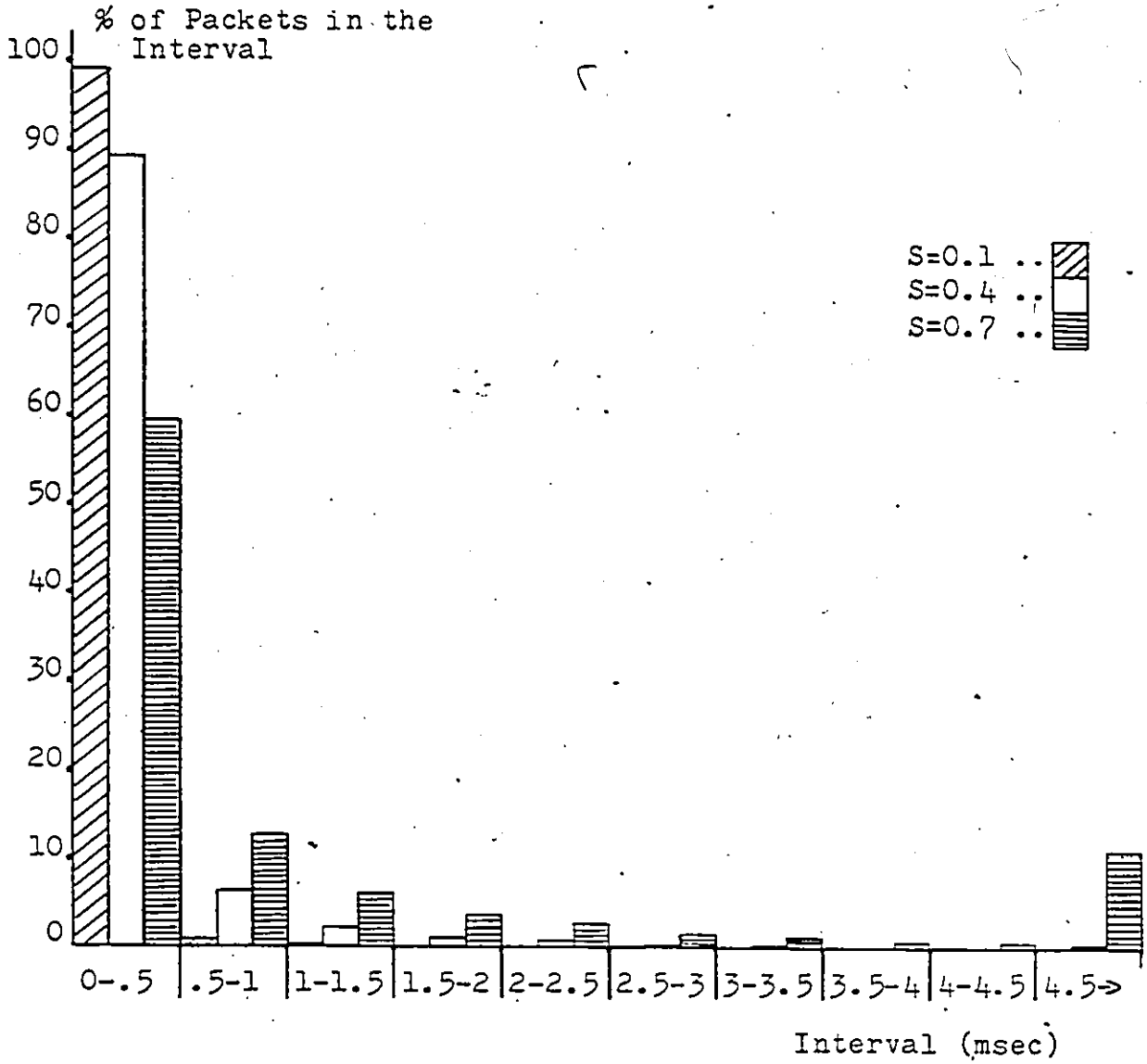


Fig 2.9 Delay Distribution for CSMA/CD.

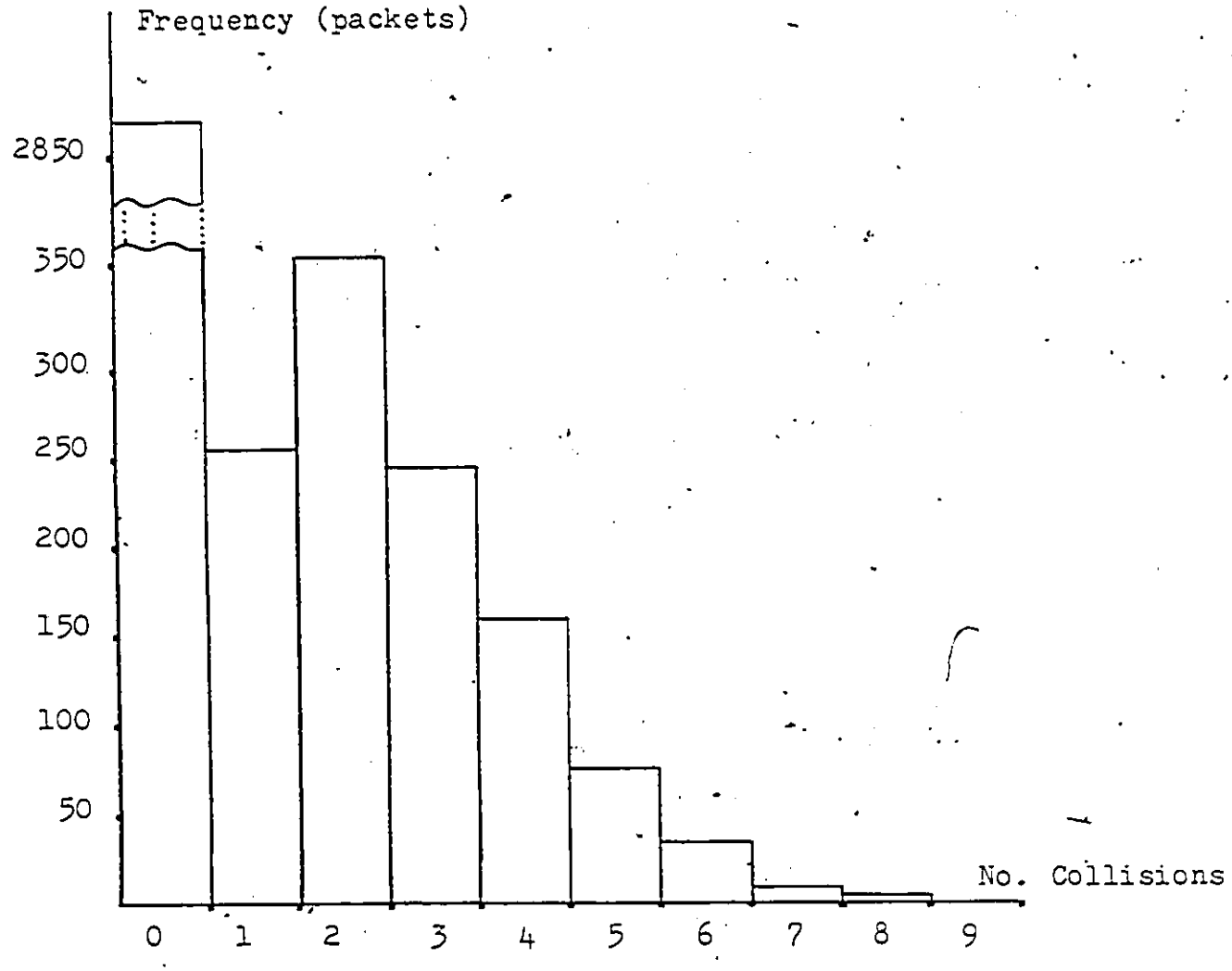


Fig 2.10 Distribution of the Number of Collisions.

2.5 GROUP BROADCASTING RECOGNIZING ACCESS METHOD (GBRAM)

This protocol utilizes information concerning the distribution of the users along the bus system [15,18,27]. It is assumed that in most systems the terminals are grouped forming clusters, so all member of a given group are connected to the bus within a small segment of it. (see fig 2.11).

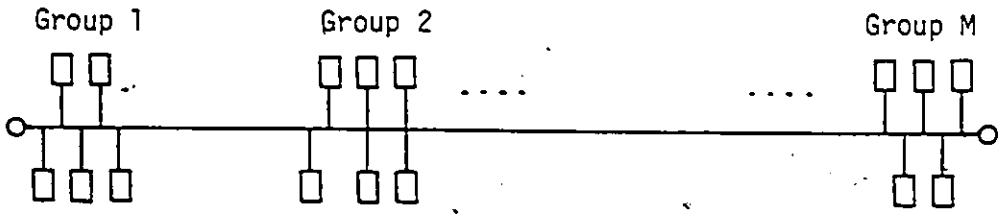


Fig 2.11 Structure of the Cable Bus System in GBRAM.

It follows then that the propagation delay a_j within a group j, is smaller than the end-to-end propagation delay a. A virtual token passing scheme is then used, that gives access rights to the members of a group using the group j propagation delay. After all the members of a group j have

had the right (which some of them may have not used) to utilize the channel, the token is passed to the following group (see fig 2.12).

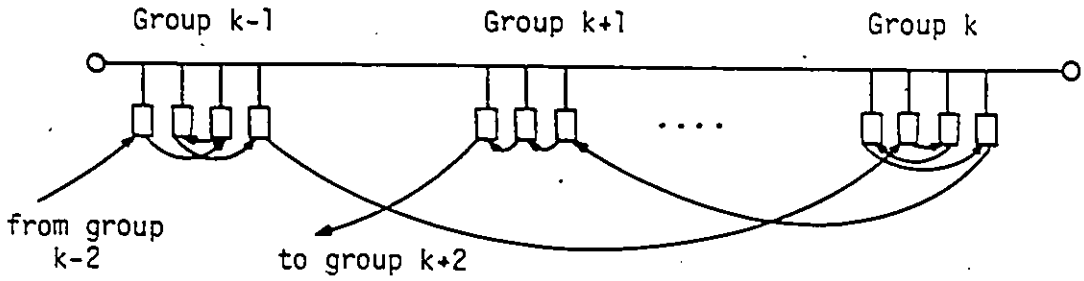


Fig 2.12 Token Passing Scheme in GBRAM.

So in GBRAM, the N users are divided into M groups according to their physical locations. Each group g_j, is given a group slot time of length equal to the end-to-end propagation delay a for its intra-group token passing. The transmission attempts from different users of a group are separated at least one intra-group propagation delay a_j, so each group can accommodate a/a_j+1 nodes. Following the group slot, GBRAM allows another a seconds for the token to travel to the next group. Figure 2.13 shows the channel scheduling for user (g_j,n_j), where n_j

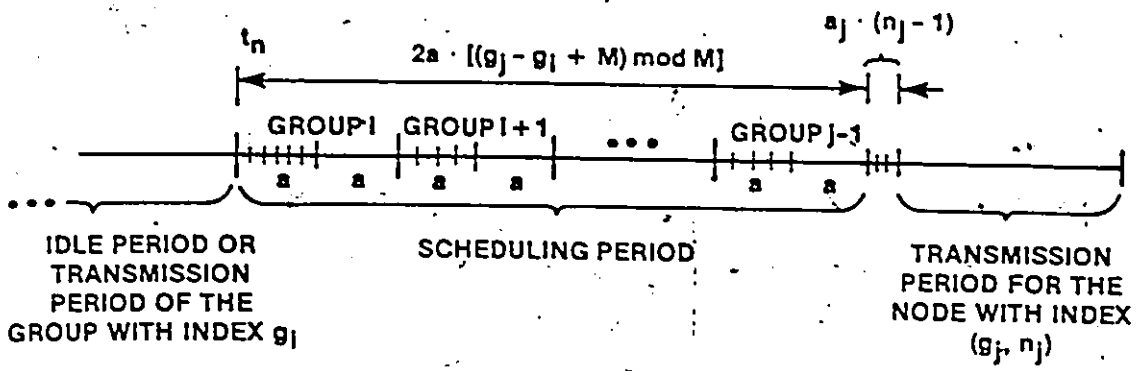


Fig 2.13 Channel Scheduling for GBRAM.

is the node index within the group g_j, following a transmission of node (g_i,n_i). The scheduling function is then given as:

$$F(g_j,n_j)=\begin{cases}2a((g_j-g_i+M)\bmod M)+a_j(n_j-1) & g_j\neq g_i \\ a_j(n_j-n_i) & g_j=g_i, n_j>n_i \\ 2aM+a_j(n_j-1) & g_j=g_i, n_j\leq n_i\end{cases} \quad (2.5)$$

The algorithm each user executes is as follows:

a) If node (g_j,n_j) senses the channel to become idle at time t_n, then the node schedules its transmission at time $t=t_n+F(g_j,n_j)$ where t_n is the end-of-text time of node (g_i,n_i). If the channel is sensed busy on or before this

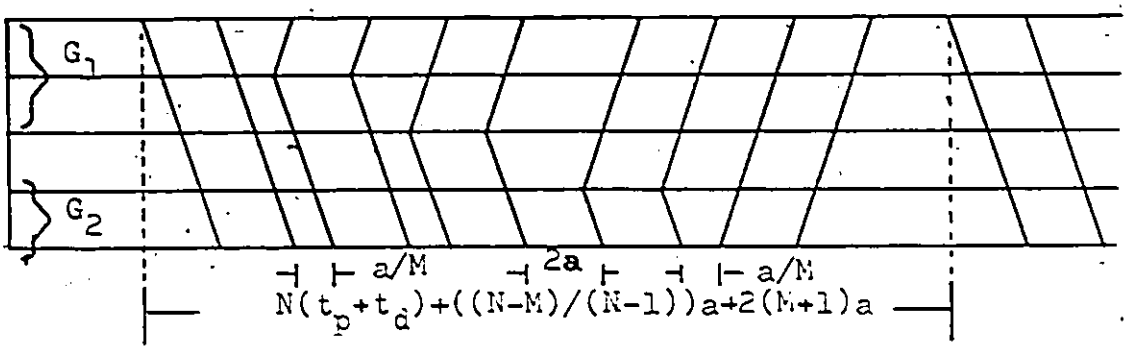


Fig 2.14 GBRAM Protocol at a Very High Load.

time t, then part (b) proceeds; otherwise, the user (g_j,n_j) can start transmitting at time t.

b) If the channel is sensed busy, then node (g_j,n_j) waits for the end of current transmission, so that a new t_n (originated by (g_m,n_m)) can be determined and then proceeds with part (a).

Using fig 2.14 (representing the channel situation at a very high load), we can derive the maximum throughput for GBRAM to be given by:

$$Smax = Nt_p / (N(t_p+t_d)+((N-M)/(N-1))a+2(M+1)a) \quad (2.6)$$

Using typical values (t_p=400 microsec, t_d=1.5 microsec), we get (with a in microsec):

	N=20	N=1000
a= 5	Smax= .9896	Smax= .9952
a=200	Smax= .7844	Smax= .9507

A program was designed to simulate GBRAM. the program considers all stations as uniformly distributed along the cable. M groups were formed using M=INT(2(SQRT(N)), as given in the reference [15]. The flow diagram of the program is shown in fig 2.15 and the simulation results in figures 2.16 and 2.17.

Figure 2.16 shows the simulation example for five stations. We can appreciate there that each packet must wait until its scheduled transmission time arrives. In the figure the segment of transmission before the arrival of packet 1 and packet p0 represent transmissions of previously generated packets. Groups are formed using the following relations, to determine to which group station k belongs and what position it has in the group.

$$Group(k) = int[(k-1)*M/N]+1 \quad (2.7)$$

$$Position(k) = int[k-N/(M*(Group(k)-1))] \quad (2.8)$$

This corresponds to the worst case, where all the stations are uniformly distributed, so in fact there are no groups in the system. In the example, stations 1,2 and 3 belong to group 1 and station 4 and 5 to group 2. The intragroup propagation delay is then a/2 (not a great advantage in this case). Note the round robin scheme followed by packets 13,16,15,18 and 19.

The protocol transmitted 17 of the 24 packets arrived, reaching a throughput of 0.6576 and an average delay of 293.6 microseconds. This result compares favorably with the one of CSMA/CD.

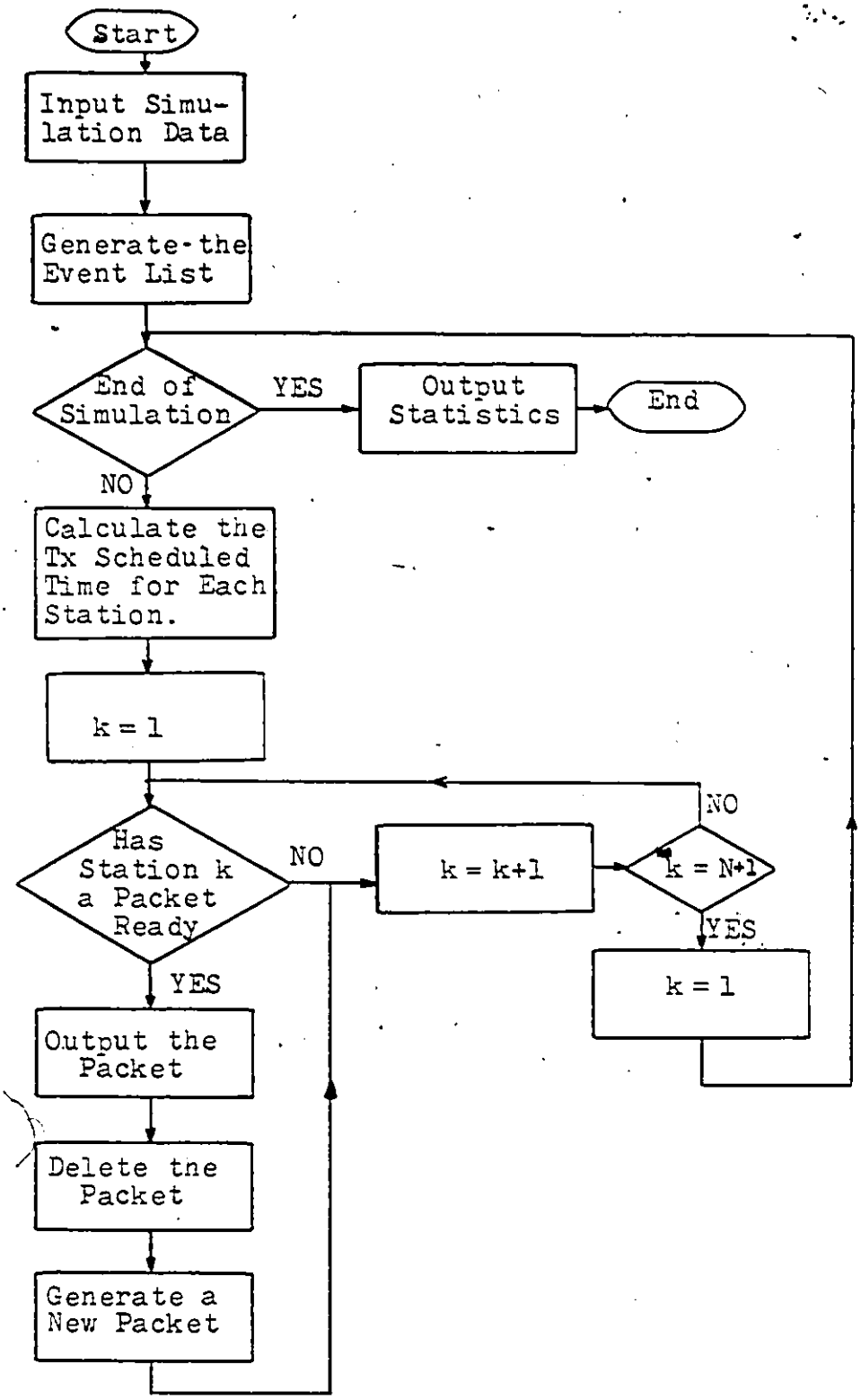
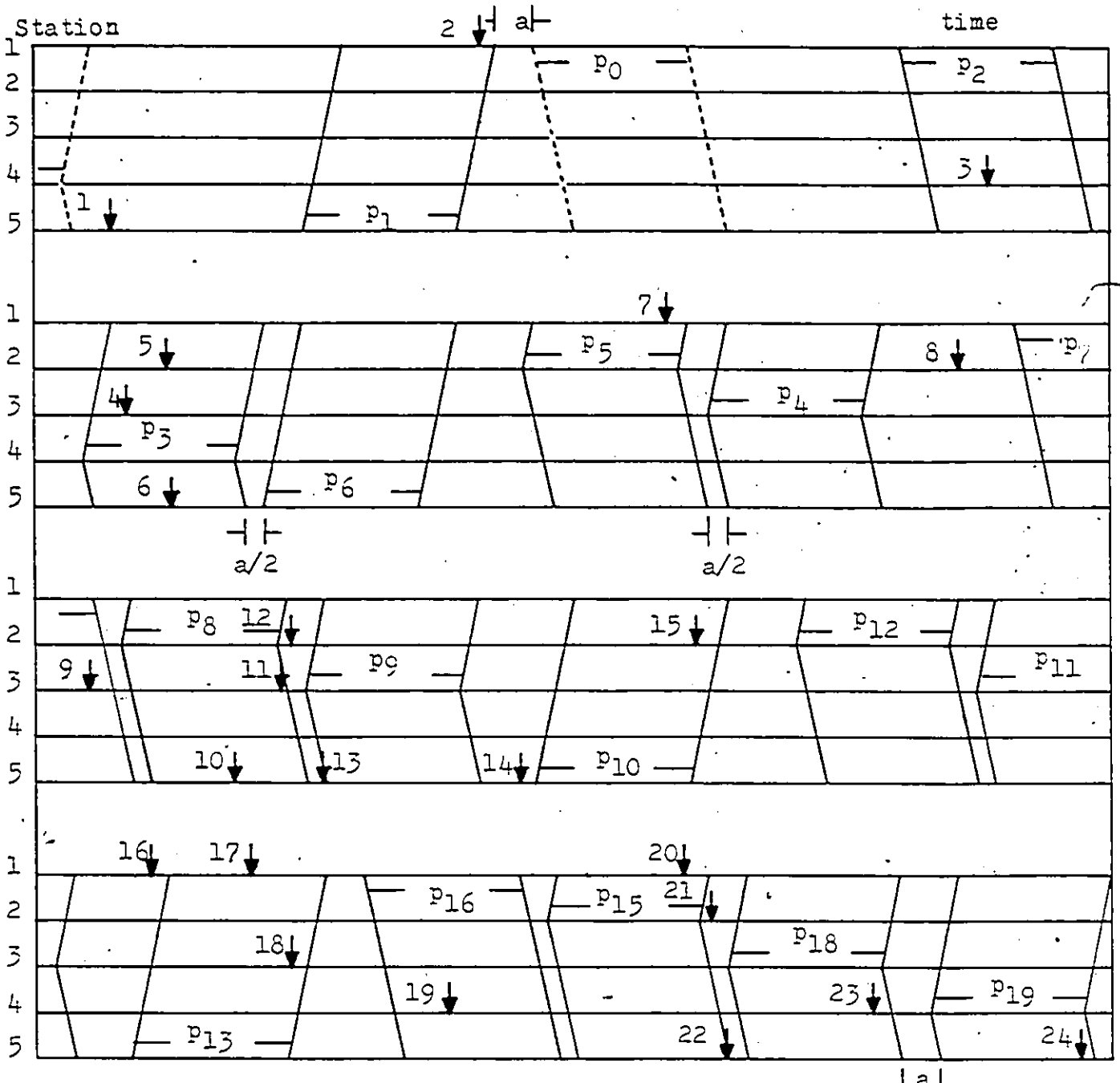


Fig 2.15 Flow Diagram of the GBRAM Simulation Program.



$t_p = 100\ \mu sec$ Packets Generated = 24 Average Delay = 293.6 μsec

$a/t_p = 0.25$ Succesful Transmissions = 17

$t_d/t_p = 0$ Throughput = 0.6576

Fig 2.16 GBRAM Simulation Example for Five Stations.

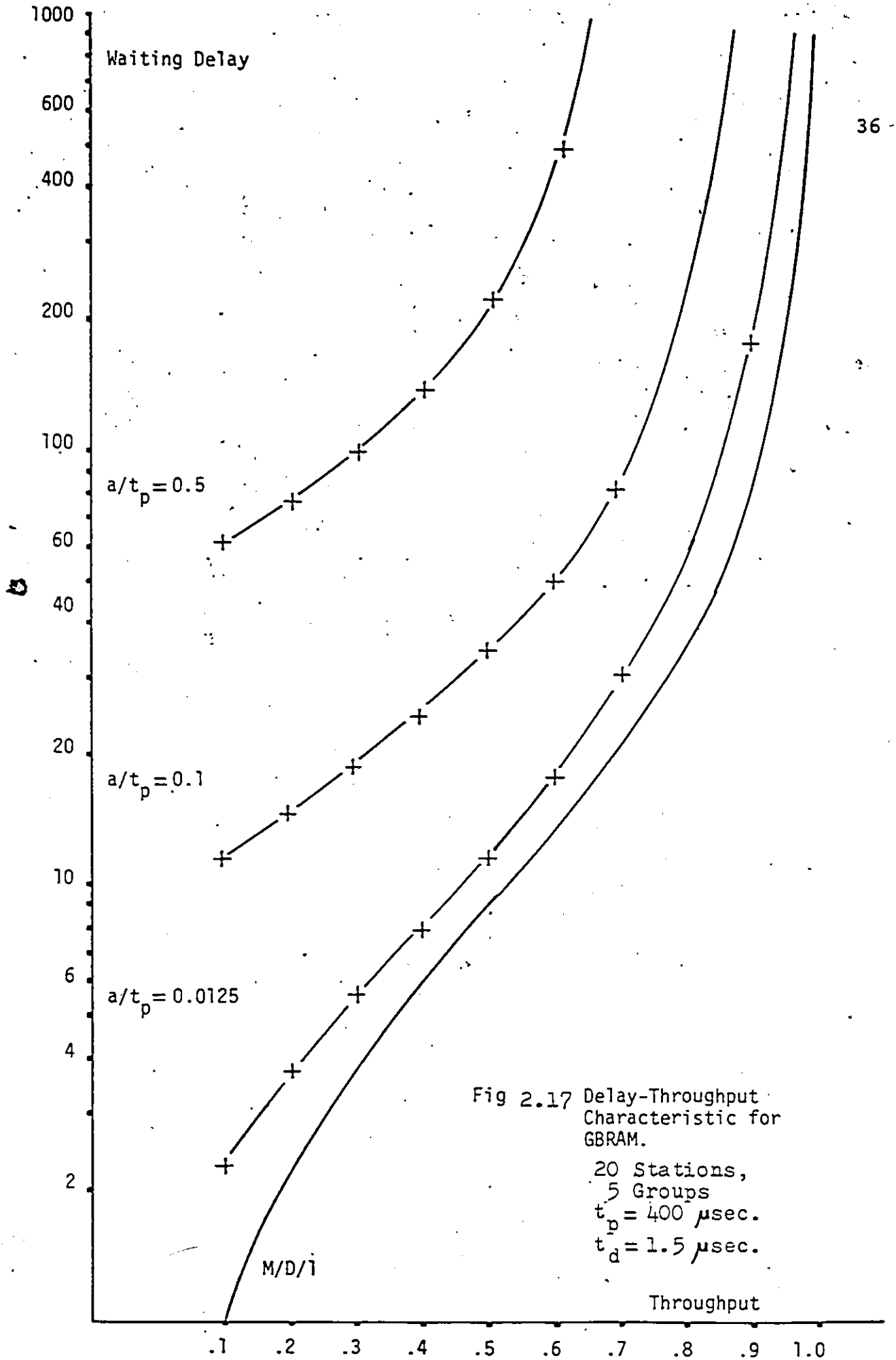


Fig 2.17 Delay-Throughput Characteristic for GBRAM.

20 Stations,
5 Groups
$t_p = 400\ \mu sec.$
$t_d = 1.5\ \mu sec.$

2.6 COMPARISON BETWEEN CSMA/CD AND GBRAM

From the previous simulation results it is clear what the main differences between CSMA/CD and GBRAM are.

The main advantage of CSMA/CD is at low and medium loads where the average delay remains low because many packets can use the channel without any delay (see fig 2.8 and 2.17). This advantage becomes less and less obvious when the ratio between the end-to-end propagation delay and the packet size increases. In those cases the maximum throughput CSMA/CD can reach decreases to the slotted-ALOHA limit (1/e) and furthermore the protocol becomes unstable at high loads.

On the contrary, GBRAM has a great advantage from medium to high loads due to its collision-free characteristic. Its throughput remains very high and stable as the load increases. However the characteristic at low load is not so good due to the protocol overhead (each packet experiences a delay, whichever the load is).

It is also clear from the simulation results, that at low values of the ratio a/t_p the differences between both protocols characteristics are minimal and they perform close to M/D/1.

Chapter III

PROTOCOL IMPROVEMENTS

3.1 INTRODUCTION

In this chapter, a number of protocols, that we developed, are discussed. The protocols presented aim to provide improvements to the performance of CSMA/CD or GBRAM.

3.2 DISTRIBUTED BROADCAST RECOGNIZING ACCESS METHOD (DBRAM)

We developed this protocol looking for an improvement over GBRAM. The protocol description is as follows:

1- Each station knows the geographical positions of the other stations, for example in a ROM table, so it can determine the propagation time it takes the signal to go to (or to come from) each other station.

   Each station knows who is transmitting, by reading the address field of each packet passing by.

2- The right of transmission is passed in a round-robin fashion, that is, after station k has (or has not) transmitted, station k+1 has then the right to use the channel. After all stations have had their right to transmit, the cycle restarts again with station 1.

3- After station j senses the end of the transmission of station k, it will wait one propagation time between station j and k to see if any station in between has started transmitting. In other words, if T_{kj} represents the time when the end of transmission of station k was sensed at station j, d_k represents the position of station k and v_p is the propagation speed of the media, we have then that the scheduled time for the transmission of station j will be given by:

$$t_j = \begin{cases} T_{kj} + (d_j - d_k)/v_p & j > k \\ T_{kj} + 4(d_N - d_1)/v_p - 3(d_k - d_j)/v_p & j \leq k \end{cases} \quad (3.1)$$

where 1 and N are the index numbers of the first and last station on the network (see fig 3.1).

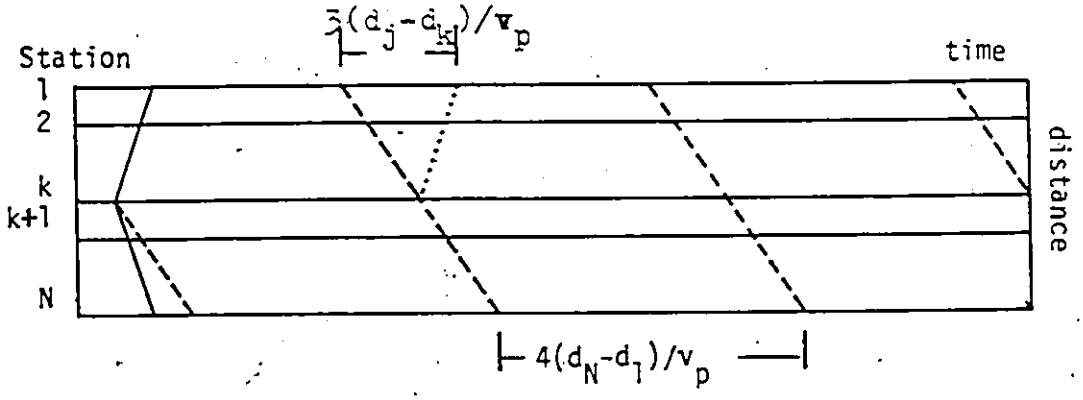


Fig 3.1 Channel Access Scheduling Scheme for DBRAM.

We observe from fig 3.1,that the maximum time to pass the token is equal to $4(d_N-d_1)/v_p$, that is, two round trip delay times.

One problem, of practical nature, is that the propagation time between two neighbouring stations can be smaller than the bit-time duration.In that case, there is no way of detecting the start or the end of a transmission. The problem is solved by adding to equation (3.1) a time t_d, which accounts for the time needed for detecting the start or the end of a transmission and processing the information internally (the minimum packet spacing).

The advantages of this protocol are:

1- Is a collision free protocol. Although collisions are possible when the system is turned on, it is a matter of software to provide synchronization with the transmission of station 1 (or a successor) which should be on at all times, transmitting some indicator packet from time to time.

2- The protocol is stable at heavy loads. Because there is no collision and each station has at least one opportunity to transmit every N packets, the throughput remains very high as the load increases.

3- The delay obtained is smaller than the one of GBRAM or BRAM, because the propagation delay time between stations is used instead of the end-to-end propagation delay time.

The disadvantages of DBRAM are:

1- Adding or moving a station implies changing ROM information in all stations on the network. This makes the system too inflexible. One solution to this problem would be sending the position information of each station instead of its address. With four bytes dedicated to this function, a highly accurate position information would be available at each station.

2- The station complexity in terms of software is greater.

A simulation program was designed to simulate DBRAM. Results of that simulation program are shown in fig 3.2 for the delay-throughput characteristic. We observe that DBRAM performs better than GBRAM (see fig 2.17).

The five stations example diagram for DBRAM is shown in fig 3.3. If we compare it with the one of GBRAM (see fig 2.16), we can see that the throughput increases to 0.6668 from 0.6576 in GBRAM and the average delay is now 231.9 microsec., compared with 293.6 microsec. in GBRAM.

The maximum throughput for DBRAM is obtained from fig 3.4 as:

$$Smax = Nt_p/(N(t_p+t_d)+4a) \qquad (3.2)$$

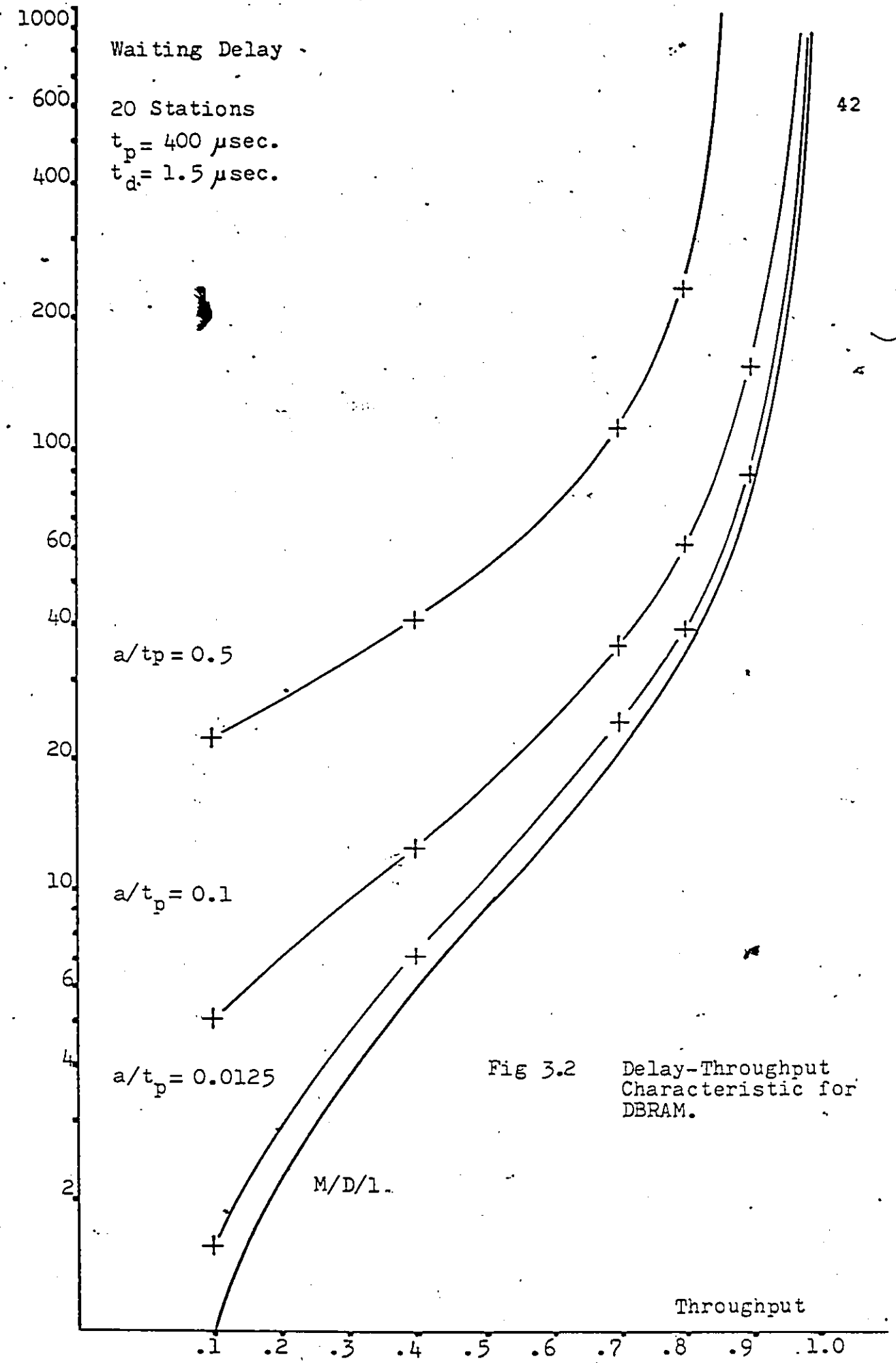
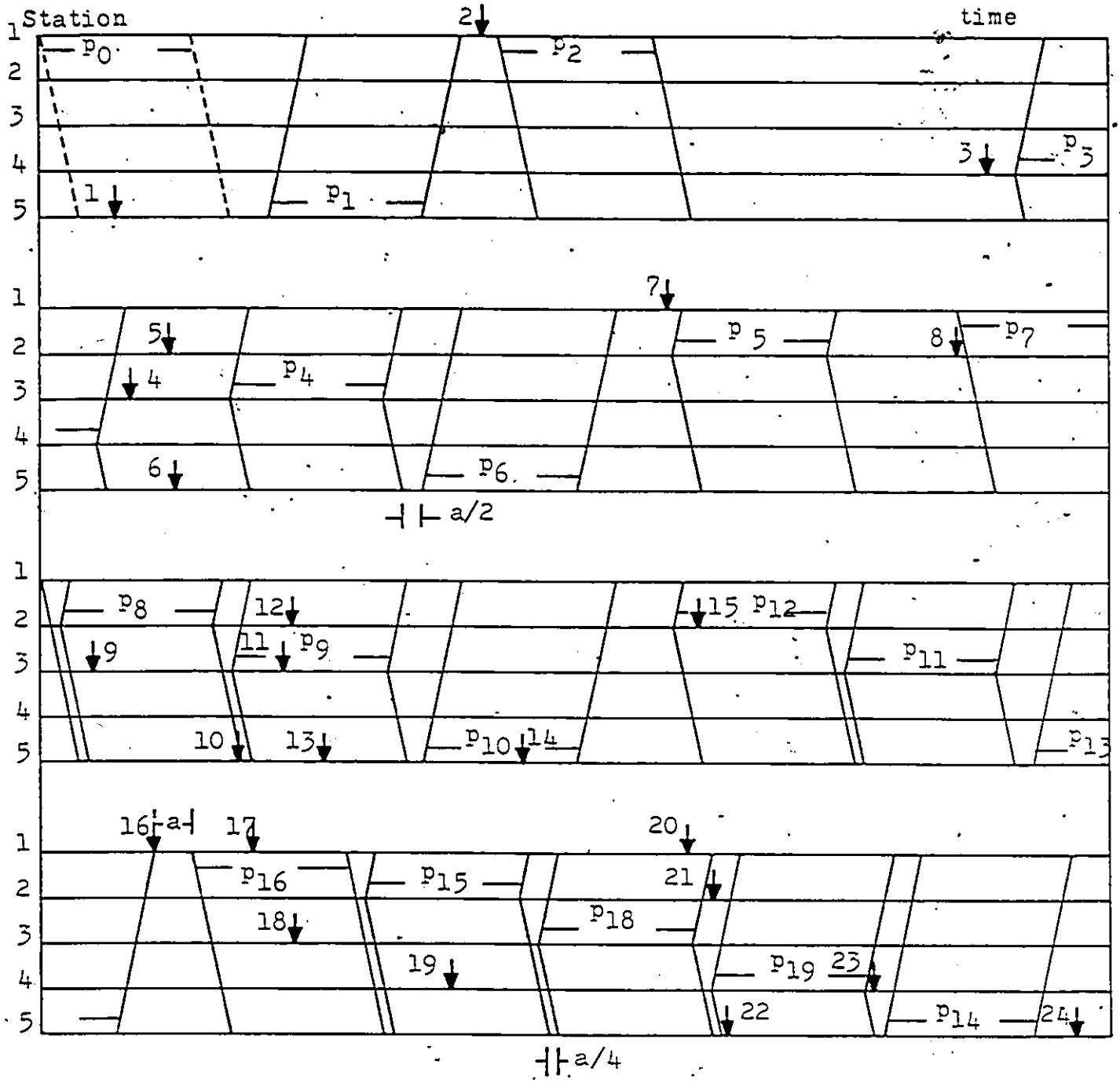


Fig 3.2 Delay-Throughput Characteristic for DBRAM.



t_p = 100 μsec Packets Generated = 24 Throughput = 0.6668
a/t_p = 0.25 Successful Transmissions = 18
t_d/t_p = 0 Average Delay = 231.9 μsec

Fig 3.3 DBRAM Simulation Example for Five Stations.

Using typical values (t_p=400 microsec, t_d=1.5 microsec), we get:

	N=20	N=1000
a=.5	Smax= .9938	Smax= .9962
a=200	Smax=.9060	Smax=.9943

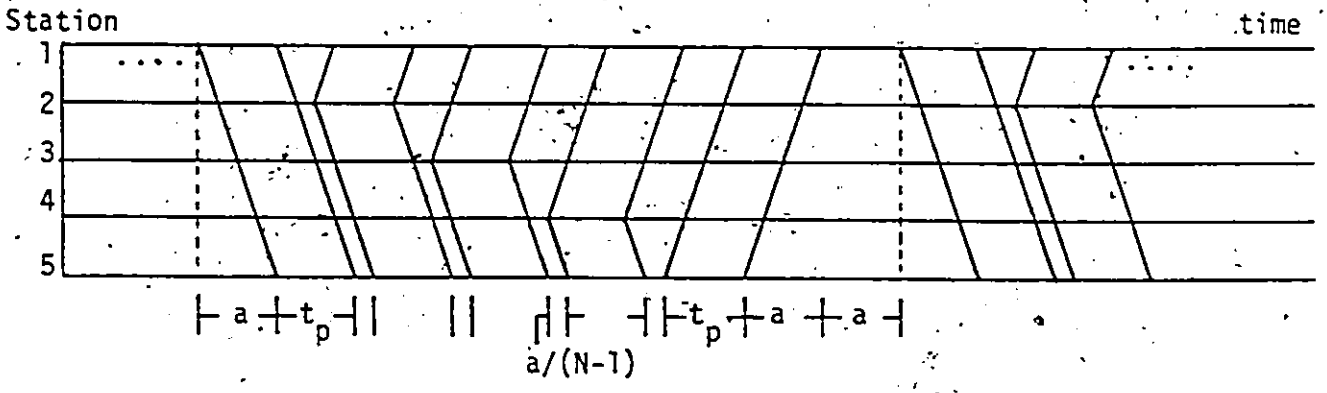


Fig 3.4 DBRAM Protocol at a Very High Load.

We can appreciate the high performance possible with this protocol.

3.3 PROTOCOL C

This protocol is easily derived from the DBRAM protocol as follows. Recall that in DBRAM, after station j senses the end of the transmission of station k, it waits one propagation time between station j and k to see if any station in between has started transmitting.

Let us consider now fig 3.5. We see there that if the transmission of station k ends at time t_k, that information will arrive at station j at time $t_j=t_k+(d_j-d_k)/v_p$, where d_k is the position of the station k on the cable and v_p is the propagation speed.

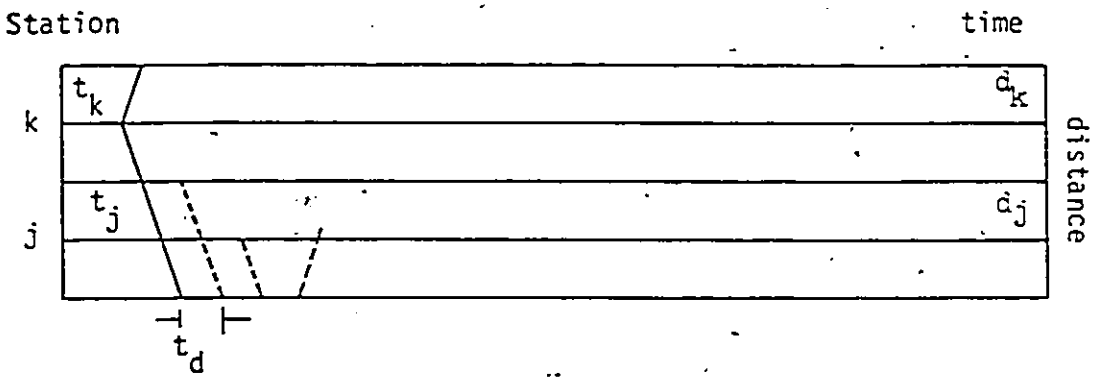


Fig 3.5 Time Relations on the Channel.

If decisions could be taken instantaneously, then station j could start transmitting immediately at t_j (we assume that

j>k for the time being). That would create problems because many stations could jump onto the channel colliding among them. Fortunately, decisions can not be taken instantaneously, so each station will take t_d seconds, from the time it senses the end of the transmission of station k, until it can put the first bit of its packet onto the channel (the minimum packet spacing).

Hence we can define t_d as the minimum decision time. So t_d seconds after t_j, station j will have completed the process of determining whether or not to use the channel and started the transmission if so.

It follows then that t_d can be used as the separation time between transmissions. In other words, in protocol C every station waits the time just necessary to know whether the previous stations have declined their right to use the channel [26,28].

The scheduling algorithm for protocol C is shown in fig 3.6 and is defined as follows:

1- The system is assumed in the steady state and station 1 has been the last one to use the channel, ending its transmission at time t_1. This event will be sensed by the other stations at times given by the following expression:

$$t_j=t_1+(d_j-d_1)/v_p \qquad (3.3)$$

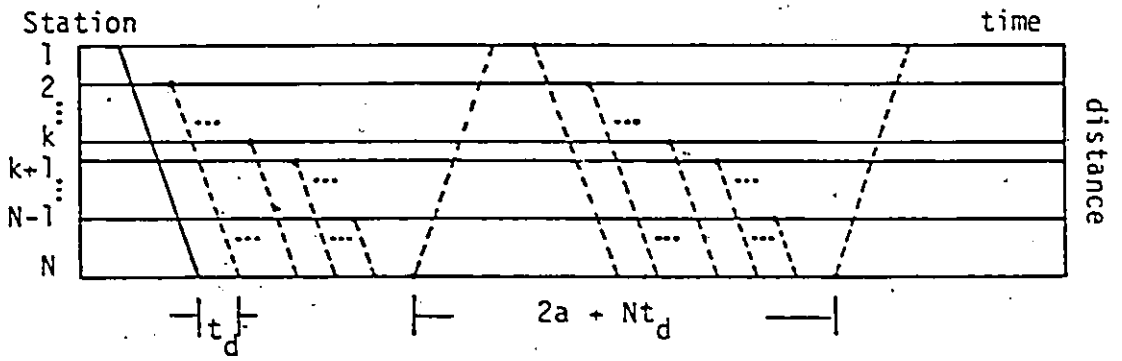


Fig 3.6 Channel Access Scheduling Scheme for Protocol C.

2- After sensing the occurrence of t_1 (at time t_j), station j schedules its transmission (by loading appropiate counters) $(j-1)t_d$ seconds after t_j. If at that time station j has something to transmit and no other station has started transmitting, then station j can begin transmitting immediately. If station j has nothing to transmit, it will wait $2a + Nt_d$ seconds to try again.

3- If station k, with $k<j$, starts transmitting before the scheduled time for station j, then each other station in the system will stop its delay counter and wait until sensing the end of the transmission of station k to resume the counting.

Note that station 1 should synchronize the system by transmitting, from time to time, its own information or some dummy packets (also called locomotive packets in the literature).

We can appreciate how simple the protocol C is, as compared with CSMA/CD, GBRAM or DBRAM. Each station has only to manage counters and sense the channel. Furthermore, if the distribution of the stations is not uniform, the scheduling function remains unchanged since the decisions are taken considering only the value of td, which only depends on the hardware used.

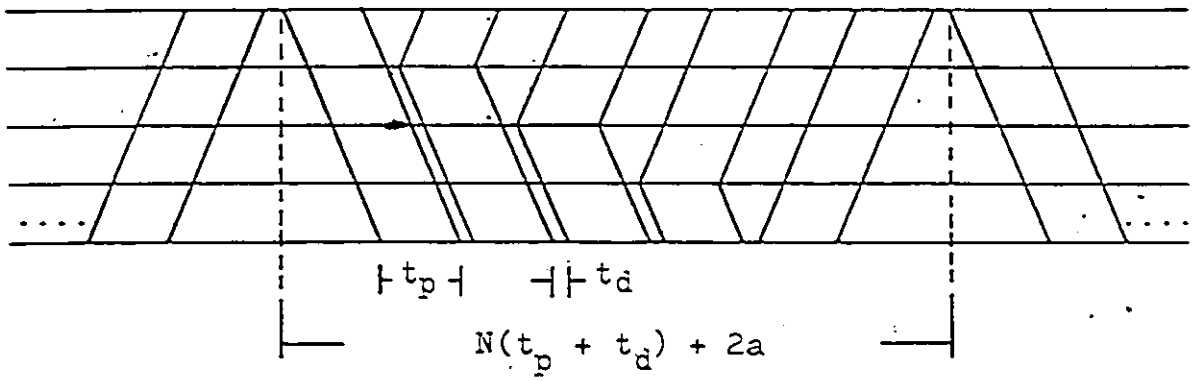


Fig 3.7 Protocol C at a Very High Load.

At a very high load, the channel will look as in fig 3.7. From that figure, we can obtain the maximum throughput of the protocol C to be:

$$Smax = Nt_p/(N(t_p+t_d)+2a) \qquad (3.4)$$

Using typical values (t_p=400 microsec, t_d=1.5 microsec), we get the following results for Smax in terms of N and a :

	N=20	N=1000
a= 5	Smax= .9950	Smax= .9962
a=200	Smax= .9490	Smax= .9953

A program was designed to simulate Protocol C. The flow diagram of the program is shown in fig 3.8 and the program listing is included in the appendix. The example for five stations is shown in fig 3.9. Figure 3.10 presents the simulation results for the delay-throughput characteristic. By comparing fig 3.10 with fig 3.2, we observe that the performance of protocol C is better than the one of DBRAM.

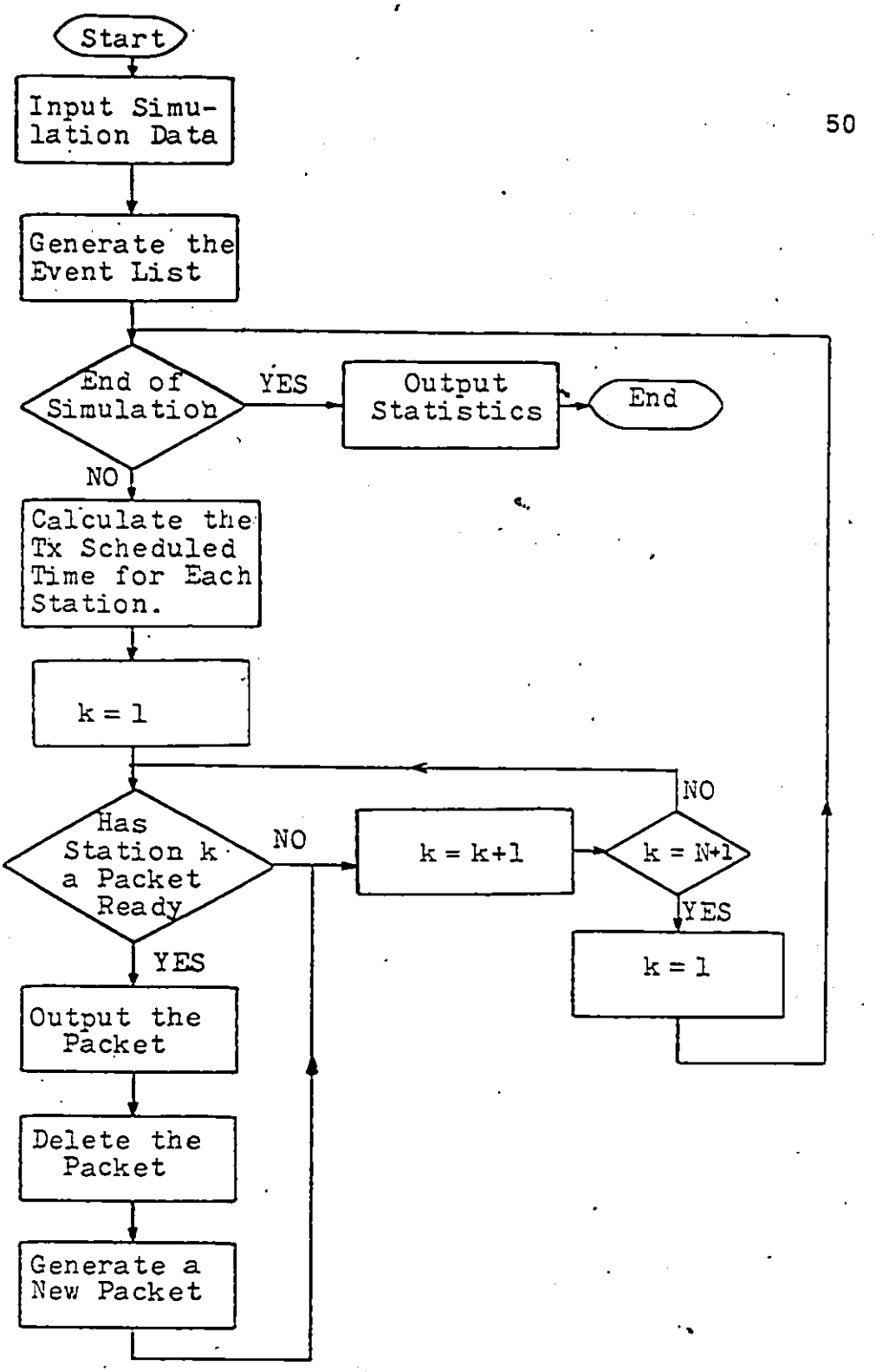
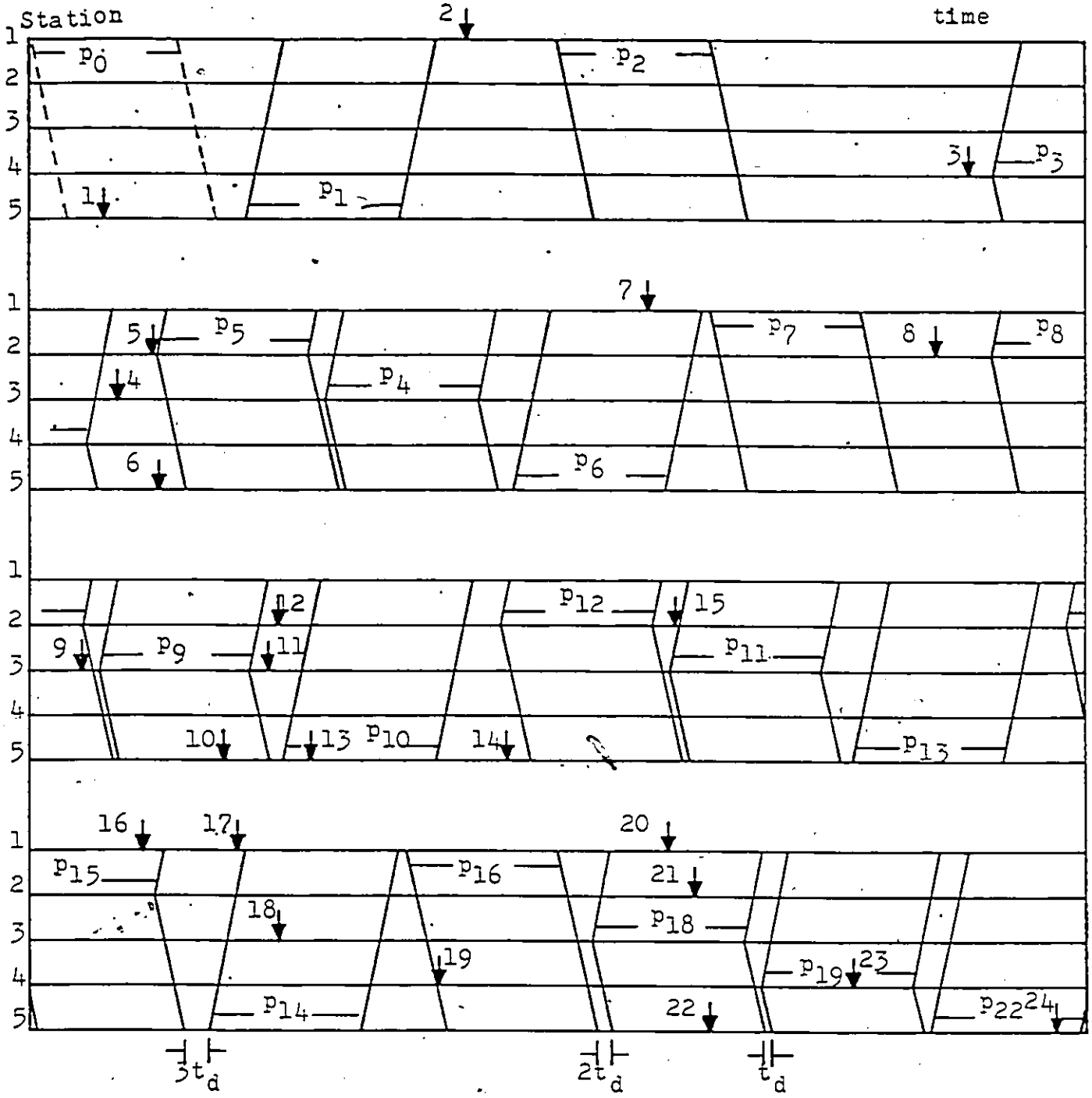


Fig 3.8 Flow Diagram of the Protocol C Simulation Program.



$t_p = 100\ \mu$sec
$a/t_p = 0.25$
$t_d/t_p = 0.05$

Packets Generated = 24
Successful Transmissions = 19
Average Delay = 155.0 μsec

Throughput = 0.6923

Fig 3.9 Protocol C Simulation Example for Five Stations.

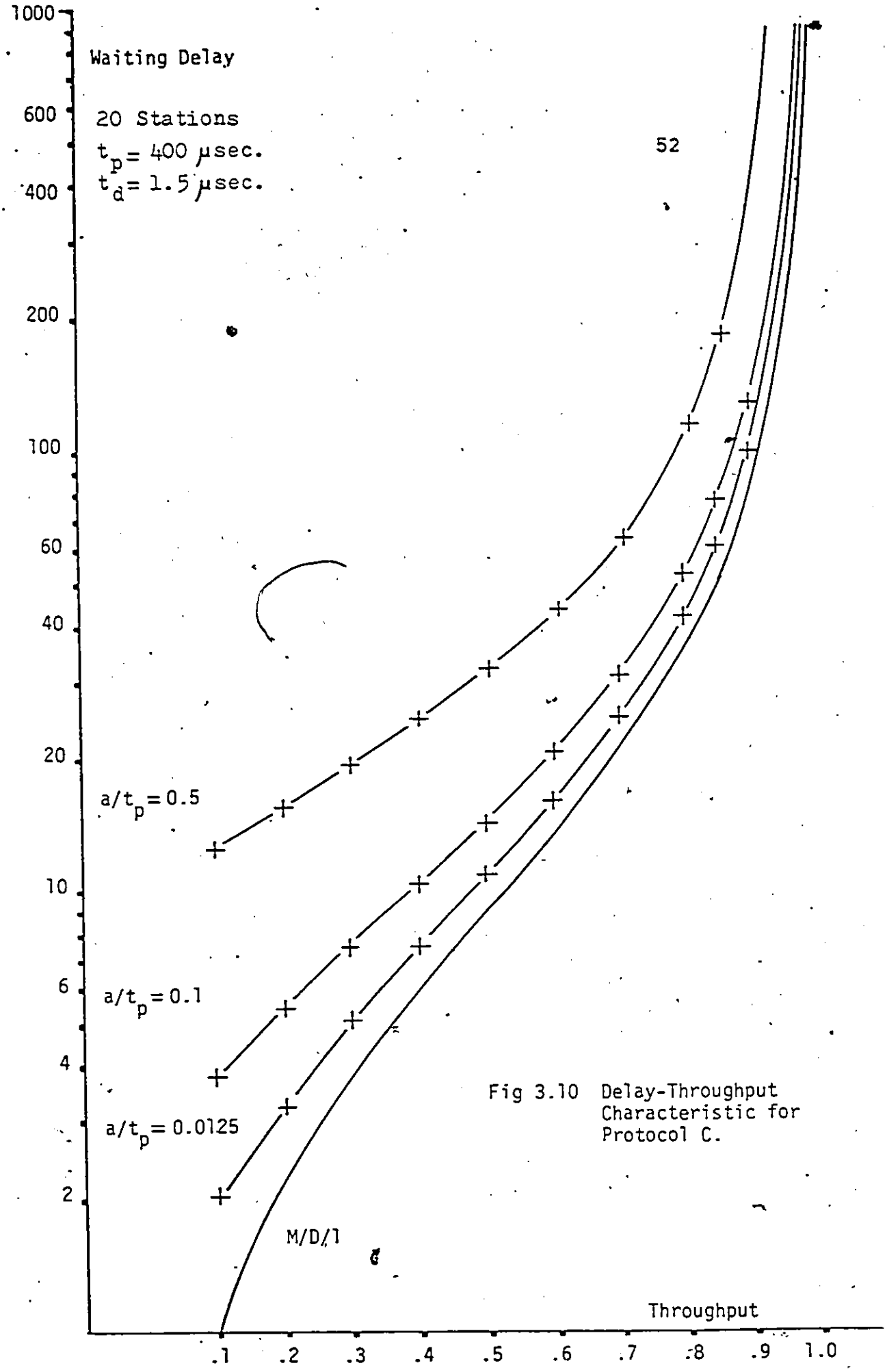


Fig 3.10 Delay-Throughput Characteristic for Protocol C.

3.4 PROTOCOL D

The next protocol we developed was called protocol D. This protocol uses a reservation period before the transmission period (in a similar way to the bit-map protocol [33]). The protocol works as follows (see fig 3.11).

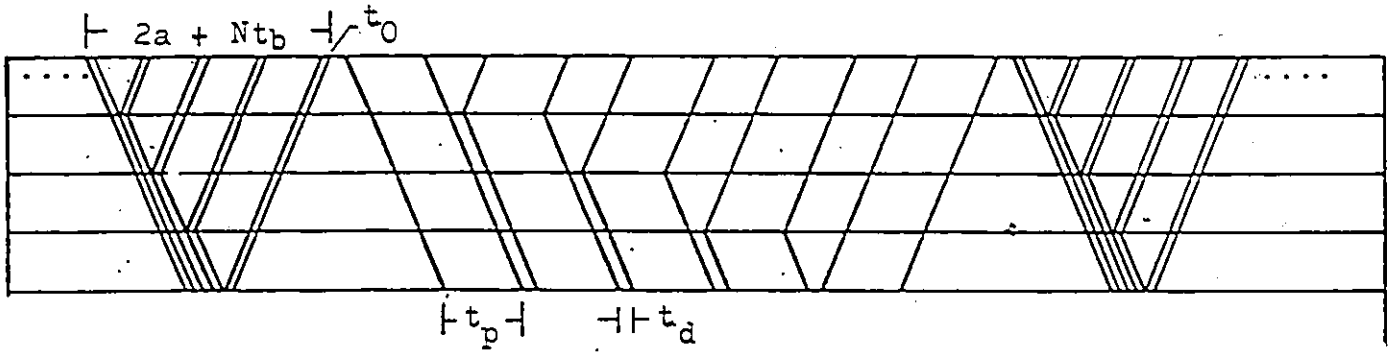


Fig 3.11 Protocol D Scheduling Scheme.

We have there a reservation period that consists of exactly N slots (N being equal to the number of stations on the system). If station 1 has a packet to send, it transmits a "one" bit during the first slot.

No other station is allowed to transmit during this slot. Regardless of what station 1 does, station 2 gets the

opportunity to transmit a "one" during its slot, again only if it has a packet ready to go. After all N bit slots have passed by, each station has a complete knowledge of which stations wish to transmit. At that point (t_0 in fig 3.11) they begin transmitting, in numerical order. Since every one agrees on who goes next, there will never be any collision. After the last ready station has transmitted its packet, another reservation period is begun. Note that if a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until the reservation period starts again.

We can appreciate in fig 3.11, the lack of symmetry of the reservation period due to the finite propagation speed. It is also clear that the reservation period will last for $2a+Nt_b$ seconds, where t_b is the time duration of a bit. At a very high load, the reservation period will consist of a sequence of ones, followed by the transmission of N packets (separated among them by t_d seconds, the interpacket time). So it follows that the maximum throughput for this protocol, considering fixed size packets, is given by:

$$Smax = Nt_p/(N(t_p+t_d+t_b)+t_d+4a) \qquad (3.5)$$

Using typical values (t_p=400 microsec, t_d=1.5 microsec, t_b=1 microsec), the following values result for S:

	N=20	N=1000
a= 5	Smax= .9911	Smax= .9937
a=200	Smax= .9038	Smax= .9918

The simulation results for this protocol are shown in fig 3.12 for the delay-throughput characteristic. Figure 3.13 shows the simulation example for five stations. We can see there the reservation periods (RE 1 to RE 9) followed by transmission periods. For instance, during the reservation period RE 4, packets 4,5 and 6 become ready and their originating stations send indication bits to inform that. After the reservation period ends, the packets are transmitted using the station ordering, that is, packet 5 followed by packets 4 and 6. Packet 7, that become ready during the transmission of packet 4, must wait until the reservation period RE 5.

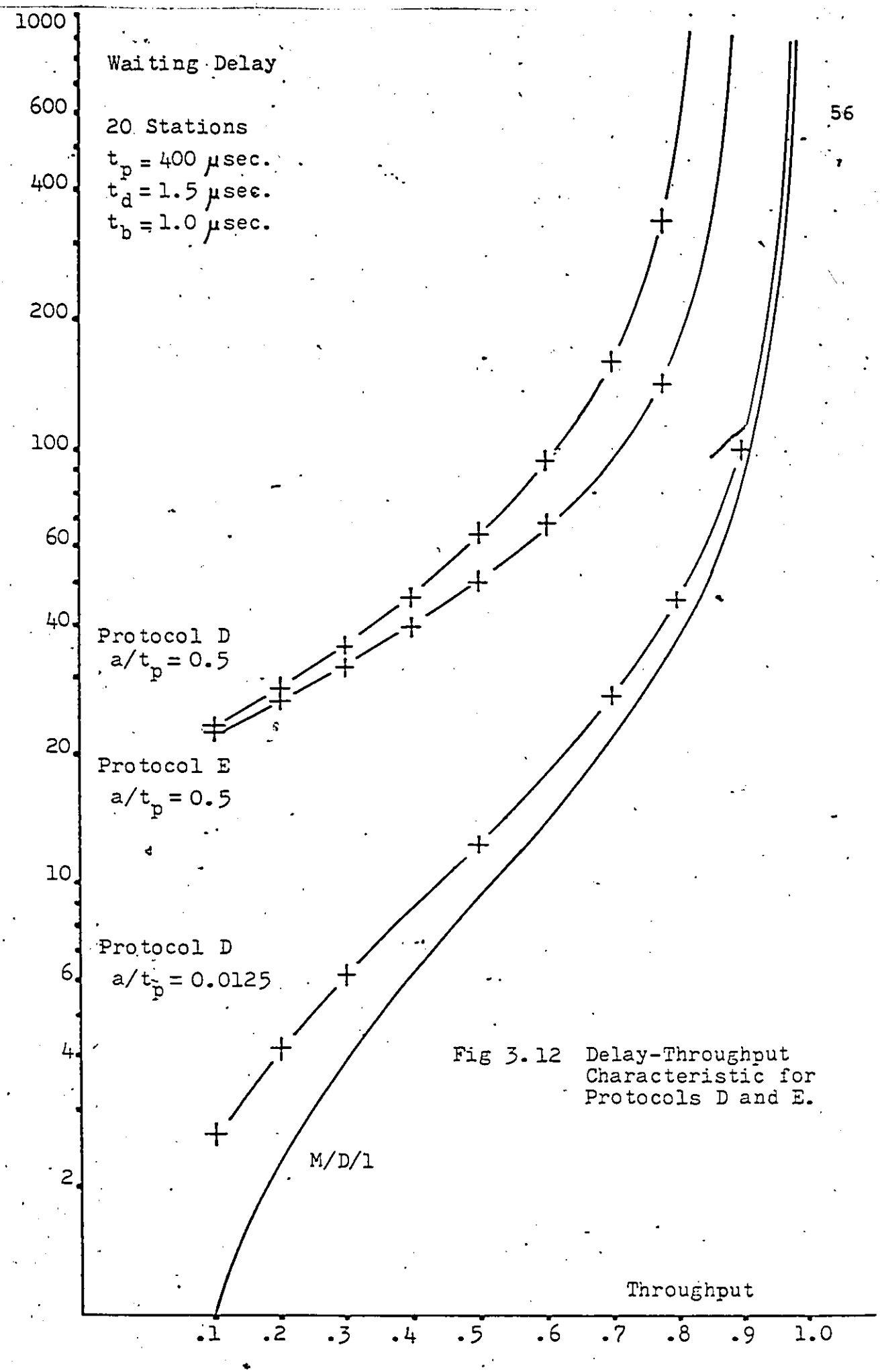
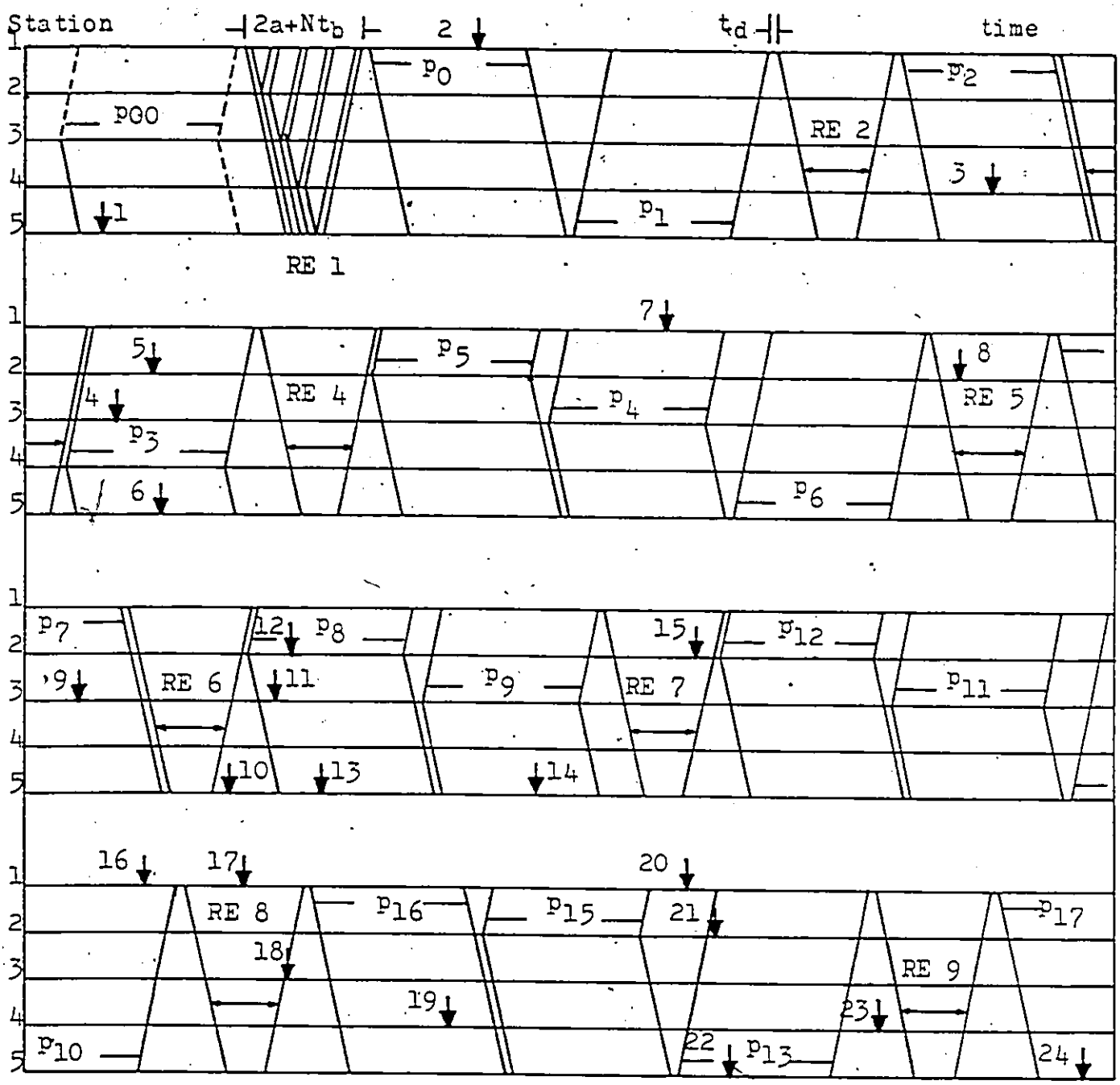


Fig 3.12 Delay-Throughput Characteristic for Protocols D and E.



t_p = 100 μ sec Packets Generated = 24 Throughput = 0.6360

a/t_p = 0.25 Successful Transmissions = 16 Average Delay = 351.2 μsec

t_d/t_p = 0.05 t_b/t_p = 0.05

Fig 3.13 Protocol D Simulation Example for Five Stations.

3.5 THE HOLE FILLING CONCEPT: PROTOCOLS E AND F

As we have shown, protocols C and D perform very well. The next step would be to think in improvements to these protocols. By using the graphical model, it was easy to discover how to improve both protocols.

Let us consider fig 3.14. We show there the pattern as seen on the channel in a system with five stations at a very high load. We can appreciate how both protocol C and protocol D do not make full use of the channel by leaving the "holes" indicated in fig 3.14 (a) and (c). those "holes" are due to the round-robin characteristic of the protocols, that is, after station N (the last in the system) has finished its transmission, station 1 will transmit next or will initiate the reservation period depending on the protocol.

Considering protocol D, the "holes" can be filled by reversing the order in which the station transmits, that is, after station N sends its reservation bit it will start transmitting immediately, as shown in fig 3.14 b). This variant of protocol D, was named protocol E and was also simulated, the results being presented in fig 3.12 for the delay-throughput characteristic and fig 3.15 for the simulation example for five stations. Again the reservation

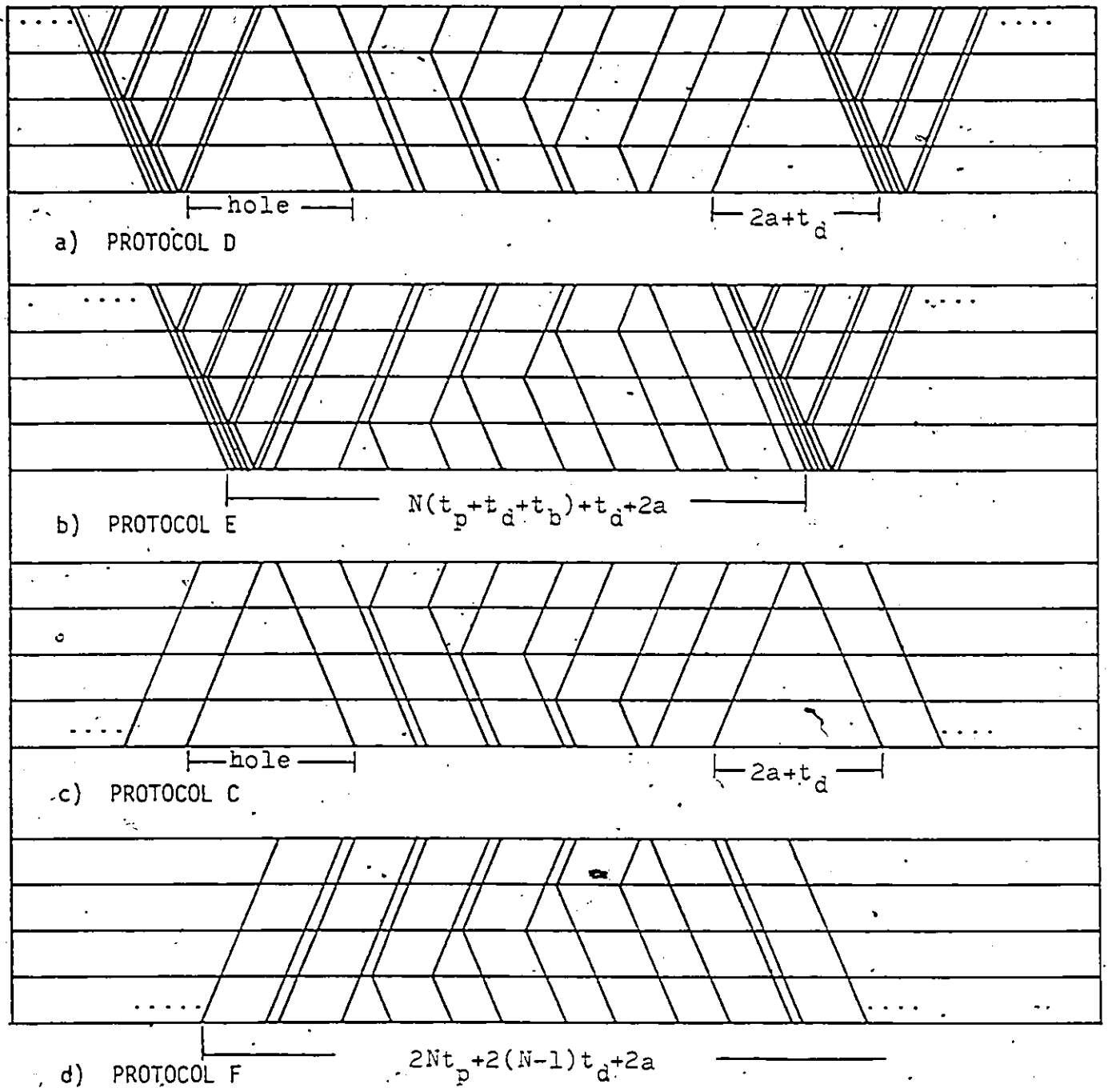
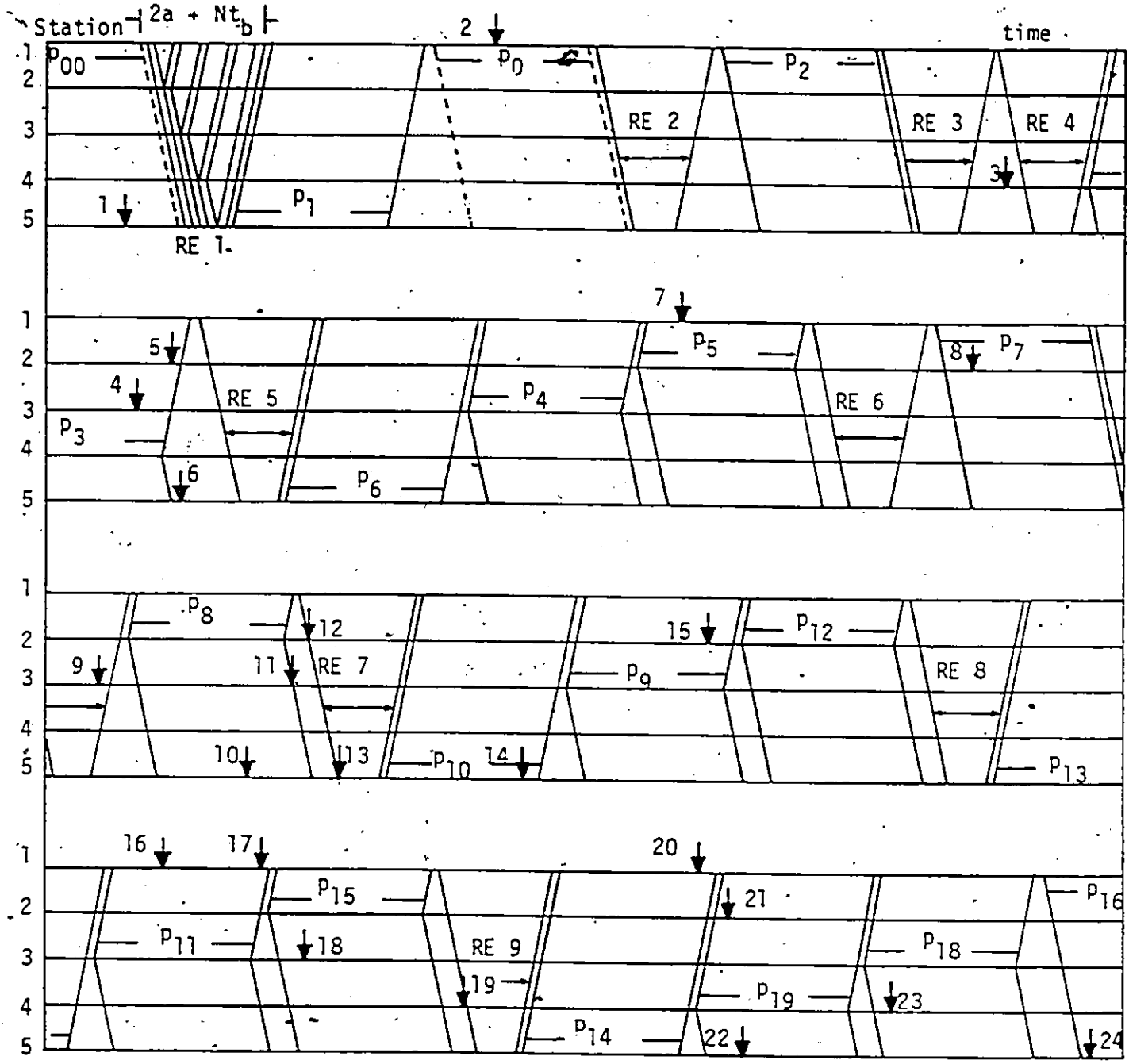


Fig 3.14 The "Hole-Filling" Concept: Protocols E and F.



t_p = 100 μsec
a/t_p = 0.25
t_d/t_p = 0.05
t_b/t_p = 0.05

Packets Generated = 24
Successful Transmissions = 18
Late Packets = 2

Throughput = .6810
Average Delay = 299.5 μsec

Fig 3.15 Protocol E Simulation Example for Five Stations.

periods are denoted by RE 1 to RE 9. The throughput increases to 0.6810 from 0.6360 in protocol D, whereas the average delay decreases to 299.5 microsec from 351.2 microsec in protocol D.

In protocol C, the "holes" are filled by reversing the order in which stations transmit in an alternating pattern, that is, station k transmit after station k-1, then the next time it follows station k+1 and so on (see fig 3.14 (d)). In this case the variant was named protocol F and the simulation results are presented in fig 3.16 for the delay-throughput characteristic. Fig 3.17 shows the simulation example for five stations. Note that the terminal stations (station 1 and 5) are allowed to transmit two successive packets (if they have two or more ready packets). That allows a fair distribution of the average delay among stations. The throughput is 0.6960 and the average delay is 166.9 microsec.

From fig 3.14, we can derive the maximum troughput for protocols E and F to be given as follows:

Protocol E

$$Smax = Nt_p/(N(t_p+t_d+t_b)+t_d+2a) \qquad (3.6)$$

Protocol F

$$Smax = 2Nt_p/(2Nt_p+2(N-1)t_d+2a) \qquad (3.7)$$

Using typical values (t_p=400 microsec, t_d=1.5 microsec and t_b= 1 microsec), we get the following results:

For protocol E,

	N=20	N=1000
a= 5	Smax= .9924	Smax= .9938
a=200	Smax= .9466	Smax= .9928

For protocol F,

	N=20	N=1000
a= 5	Smax= .9958	Smax= .9999
a=200	Smax= .9722	Smax= .9994

As we can see from the above results, protocol F has the highest throughput of the studied protocols. However, the performance of all these protocols at very light loads is not very good as compared with the one of CSMA/CD. That is because each packet in those protocols is delayed by a certain time, as given by the scheduling algorithm [14,29].

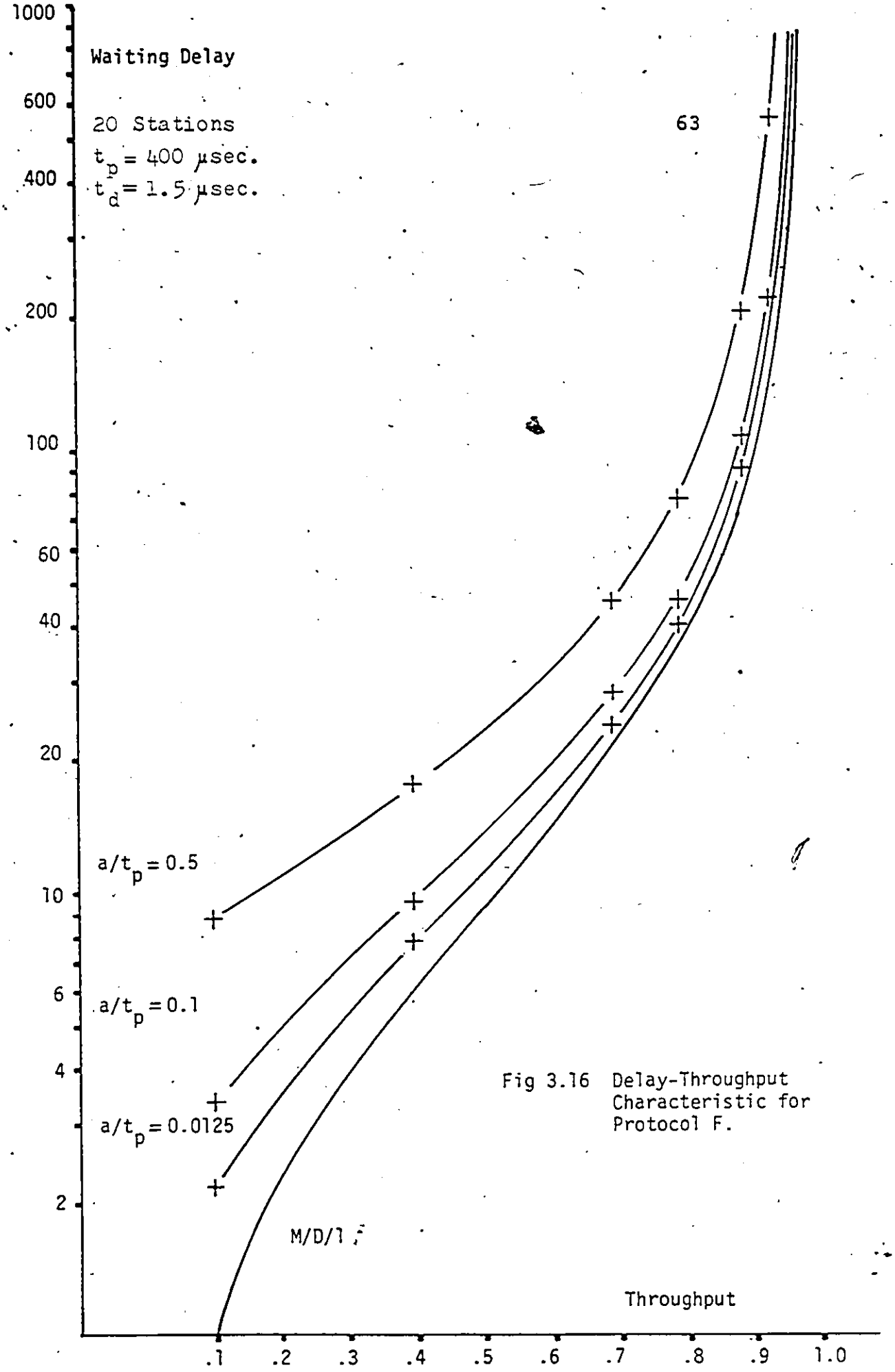
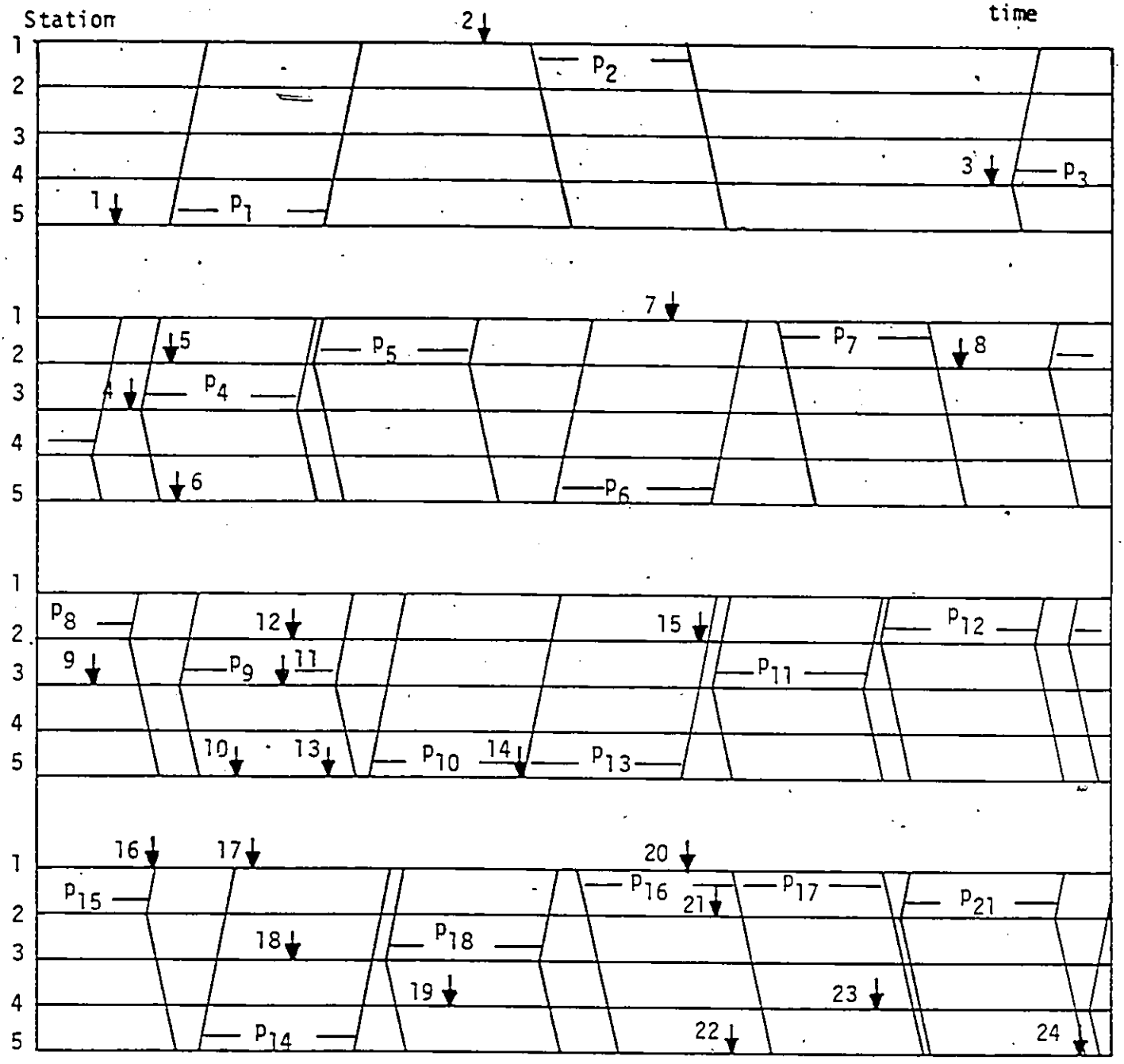


Fig 3.16 Delay-Throughput Characteristic for Protocol F.



$t_p = 100\ \mu sec$ Packets Generated = 24 Average Delay $= 166.9\ \mu sec$

$a/t_p = 0.25$ Succesful Transmissions = 19

$t_d/t_p = 0.05$ Throughput = 0.6960

Fig 3.17 Protocol F Simulation Example for Five Stations.

These facts were used to define a hybrid approach that functions as CSMA/CD at light loads and changes to a collision free protocol as the load increases. That approach will be discussed in the next chapter.

Chapter IV

THE HYMAP PROTOCOL

4.1 INTRODUCTION

In this chapter we describe HYMAP, a new Hybrid Multiple Access Protocol for Local Area Networks. The protocol combines features of both CSMA/CD and a collision-free protocol. Control is being transferred from one protocol to the other according to state information sensed on the channel. The performance of HYMAP is evaluated, for both data and voice packets, using simulation and compared favorably with other protocols already discussed before in this thesis.

4.2 PROTOCOL DESCRIPTION [35]

The protocol we will describe utilizes the best features of both CSMA/CD and a collision-free protocol. Each station is assumed to take full advantage of all the information that is available on the channel.

Each station is assumed to have the capability of doing the following tasks:

1- Sense its own collisions. Each station compares bit-by-bit the information it sends with the one that it senses on the channel.

2- Sense collisions of other packets on the channel. The station detects irregularities in the information passing through it. The irregularities are things such as packet incompleteness, groups of bits without any order, etc.

3- Determine the source and destination addresses. The station reads this information from the packets passing by.

4- Determine the starting and ending time of a packet transmission. In this sense, stations should also be capable of detecting very short transmission lengths (down to one bit).

5- Defer to outgoing transmissions before using the channel.

We also assume that the channel is noiseless and there is no capture effect on the packets.

In order to have a better understanding of the protocol, we will explain it through an example. In the example (see fig 4.1), we have a cable system with five stations connected to it. The protocol is as follows:

1- Each station, if it has something to transmit, can start the transmission of a packet immediately after sensing

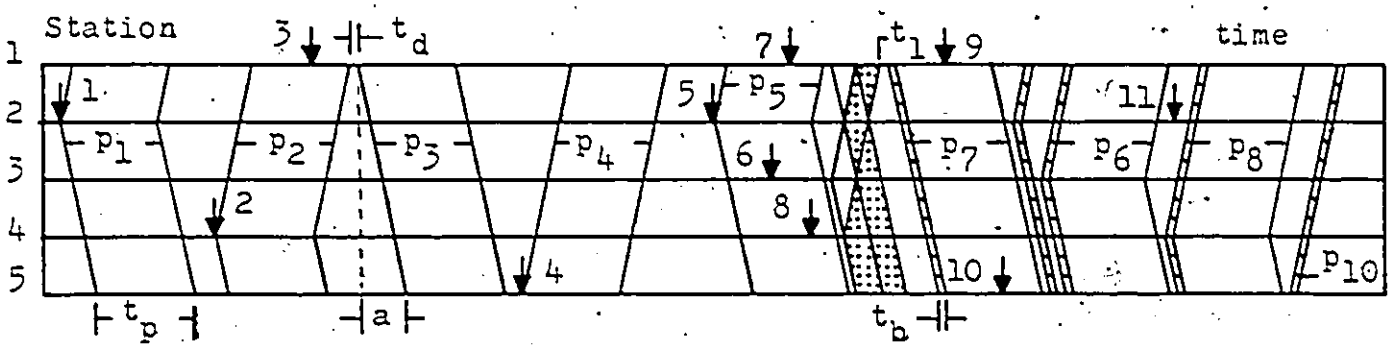


Fig 4.1 HYMAP Protocol Example.

that the channel is idle. In the example shown, packets 1 and 2 follow that pattern. The packet duration is t_p sec.

2- If the channel is sensed busy and if no collision is detected on the channel, then the stations ready to transmit defer the time of their transmissions until the channel is sensed idle.

In the example, packet 3 defers to packet 2. The minimum separation time between packets is denoted by td sec. After that, packets 4 and 5 go onto the channel without problems. Up to this point the system follows a CSMA/CD protocol. Note that at the time packet 5 is going through the channel, three packets become ready (packets 6,7 and 8). The stations with ready packets will wait until they sense the end of transmission of packet 5 and then they will transmit their packets, generating a collision, as shown in fig 4.1.

The collision will be detected by the stations participating in it by comparing what they transmit with what they sense on the channel. The rest of the stations will use the irregularities detection capability described before to detect the collision.

3- If a collision is detected, then each station in the system (with the exception of station 1) enters to a wait-while-listening state. In this state, stations wait to hear an indication transmitted by station 1 and then follow the procedure indicated below.

4- Station 1 after sensing a collision on the channel waits until it senses the channel idle and starts a synchronization procedure. The procedure consists of sending a bit (or group of bits) to indicate whether or not station 1 will transmit a packet following the synchronization bit. tb is the duration of this synchronization bit (or bits).

Note that it is possible that station 1 detects the channel idle, while there is still collision produced information on the channel travelling towards station 1. In that case, a new collision will occur and it will be detected by the system, forcing station 1 to go back to point 4 of the protocol.

In the example, station 1 senses the channel idle after the collision at t_1, then it sends a "one" as synchronizing bit and transmit packet 7.

5- Station k, after sensing the indication bit or bits from station 1, will schedule its next transmission using the algorithm described below.

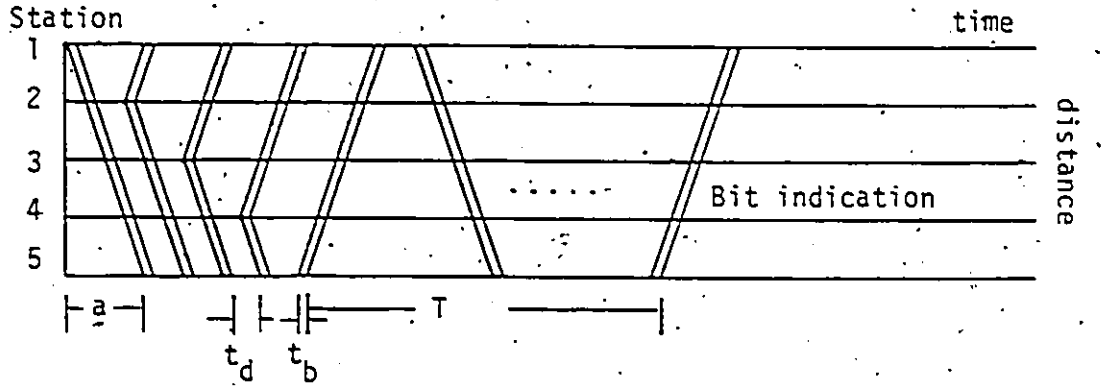


Fig 4.2 Channel Access Scheduling Scheme for HYMAP.

Let us consider fig 4.2. We see there that each station waits to see the indication bit of station 1 pass by its location. After seeing that indication, station k schedules its transmission (by starting decrementing counters) using the following expression (assuming N stations in the system):

$$t_s(k)=(k-1)(t_d+t_b) \quad \text{for } k>1 \qquad (4.1)$$

This synchronization sequence is repeated after T sec, where:

$$T = N(t_d+t_b) + 2a \qquad (4.2)$$

Note that just before station k starts transmitting, it sends a bit indication to inform the other stations of its intention. If a station j with $j<k$ has already set its bit indication to "one" and started transmitting, then all stations, after sensing that, stop their counters until they sense the end of the transmission of station j.

The scheduling scheme could also work without requiring from stations (other than station 1) to send a bit indication. We will comment more about this later. In that case the scheduling function is the same but with $t_b=0$.

In the example (fig 4.1), after station 1 sends packet 7, station 2 sends a "zero" bit indication to inform the system it will not use its right to transmit. Then station 3 sends a "one" bit indication and packet 6 and after that station 4 does the same. Note that up to this time three packets (packets 9,10 and 11) have become ready, so they also go to the channel using the same procedure (see fig 4.3).

6- If no station has packets ready to send, then each station will see negative bit indications passing by. So after counting N of those negative indications each station assumes the load has decreased and the procedure returns to the CSMA/CD mode (point 1 in fig 4.3).

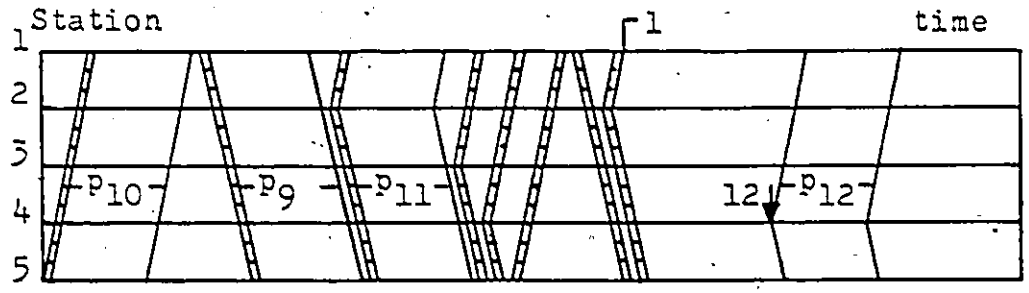


Fig 4.3 Example Continuation.

In the example after packet 11, five "zero" bit indications go to the channel, so the protocol changes to CSMA/CD and packet 12 goes without delay.

Note that if we do not use bit indications, a time-out counter can be used at each station to detect that the load has decreased. After that time-out, if no packet transmission is sensed, the system returns to CSMA/CD mode of operation.

It is not difficult to see from the protocol description that it should perform better than CSMA/CD at light loads. The reason is that a packet is allowed to collide only once. After that collision, the packet will be delivered using a collision-free protocol, whereas in CSMA/CD that packet, after experiencing a back-off delay, has a certain

probability of colliding again. On the other hand, even at high loads there is a probability of having some periods of time without requests on the channel. In those periods the protocol jumps to CSMA/CD and the next request is served without delay. So the protocol jumps back and forth from one mode of operation to the other at all levels of loading.

The schematic diagram of operation for station 1 is shown in fig 4.4. In fig 4.5 the corresponding scheme for each other station is shown.

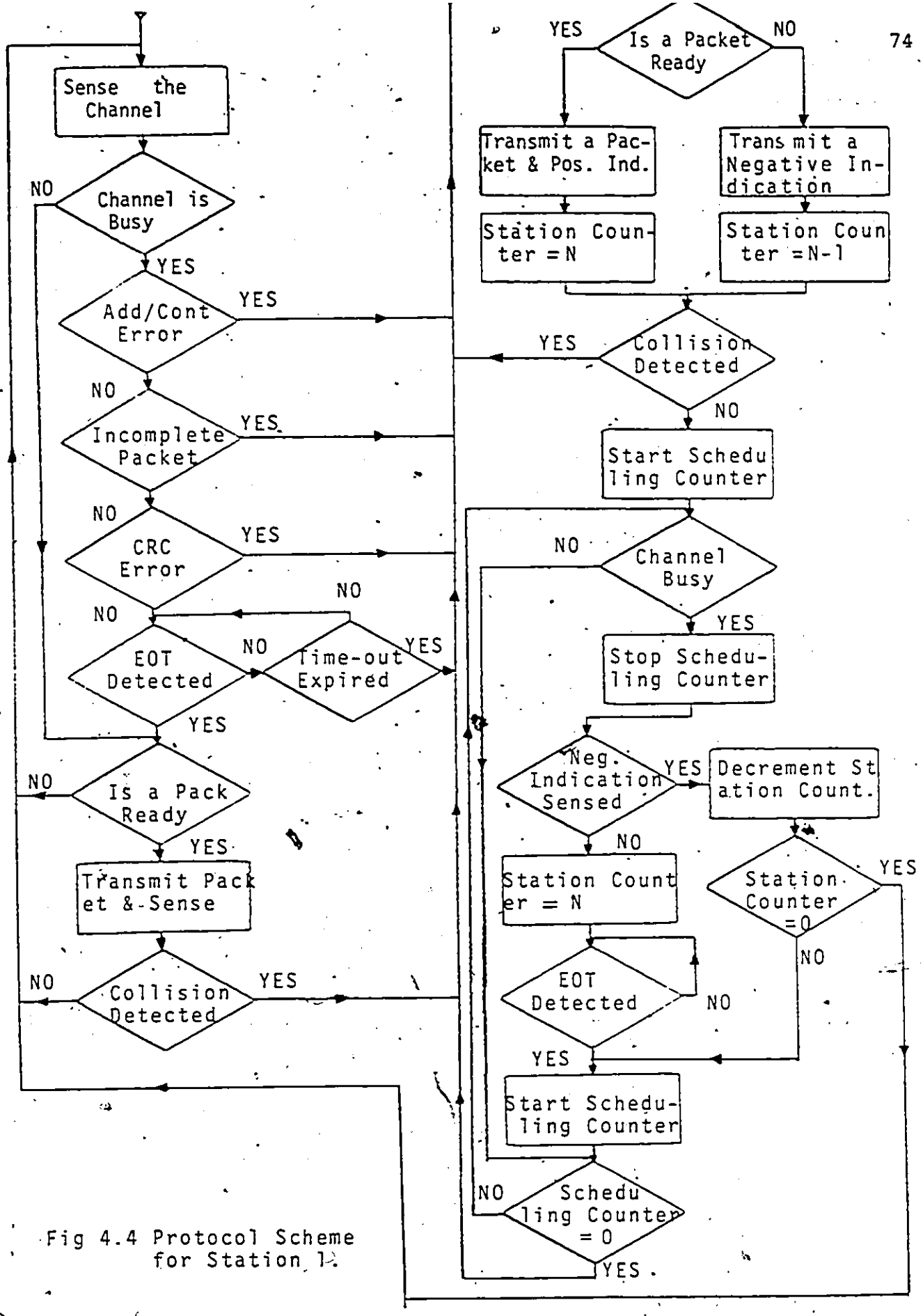


Fig 4.4 Protocol Scheme for Station 1.

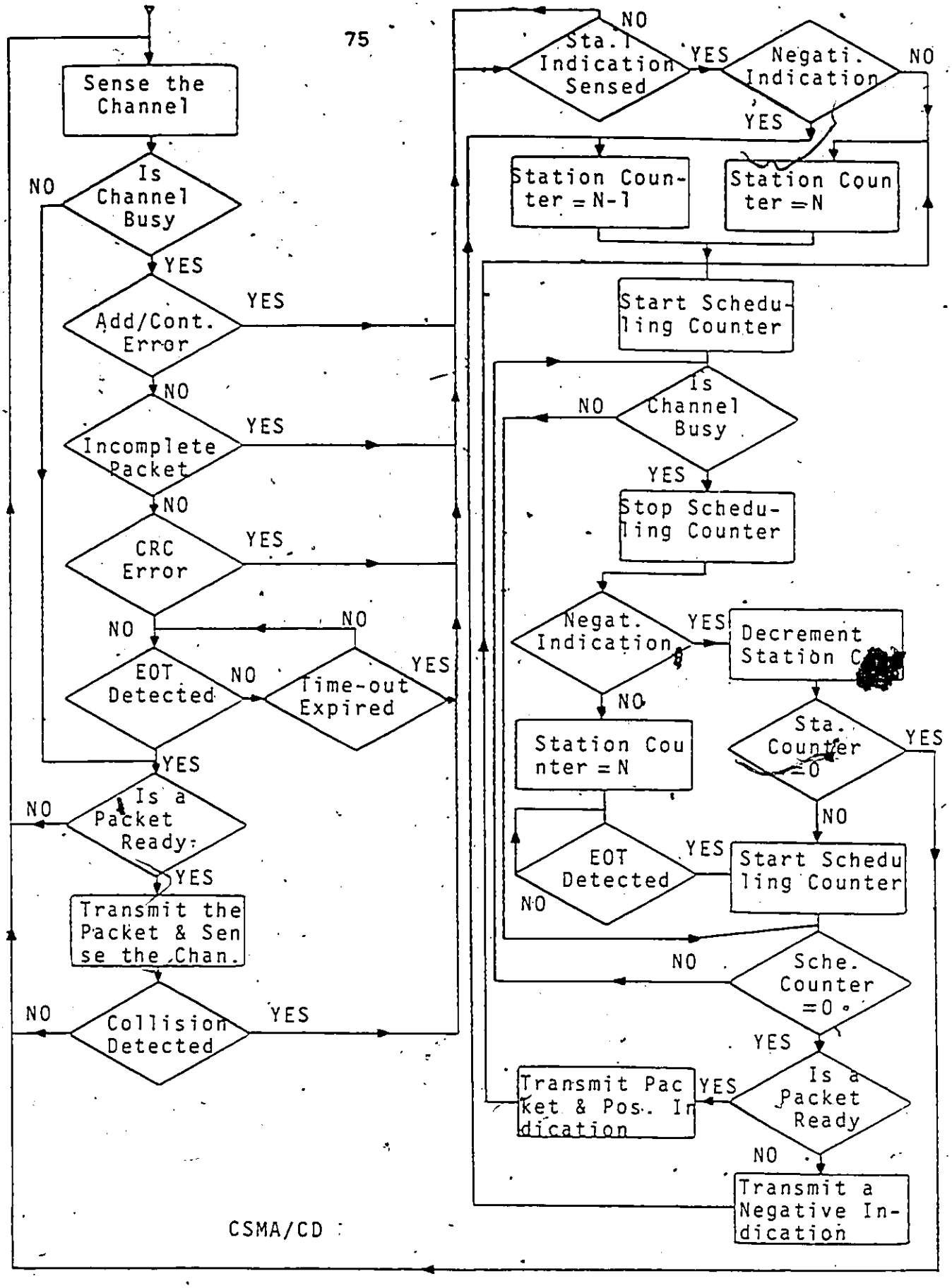


Fig 4.5 Protocol Scheme for Station k.

4.3 PROTOCOL FEATURES

We will comment on some of the protocol features with respect to a practical application.

a. Simplicity

As we can see from the schematics shown in figs 4.4 and 4.5, the actions each station is required to perform are very simple.

There is no problem in reading packets since this is a task each station must perform anyway. As we explained before, the collision detection procedure for packets generated by other stations requires just checking certain properties of a packet. Among these properties we have the presence of a starting flag character, control characters, address field, information field, CRC field and a termination flag. All these processes just require very simple bit storing and manipulation.

Other processes involved only require the loading and management of counters, which do not need any complex mathematical calculation.

b. Robustness

At light loads the protocol is identical to CSMA/CD, so if a node fails, this does not make any difference in the system operation.

At medium and heavy loads problems may arise due to timing inaccuracies in the stations or due to line errors produced by noise and generating incorrect bit patterns.

These problems can be avoided by making the stations from time to time, synchronize their clocks with the transmission of station 1.

In the case that station 1 fails, then after a time-out the following station (i.e. station 2) can assume its job.

In the case that any other station fails or is off-line, the operation is not complicated. For instance, in a N station system, each station after sensing the indication of station 1 or its successor, schedules its transmission using equation (4.1). That scheduling is totally independent of the state of the other stations. The only information from the other stations that is used, is the negative indication each station transmits, which is counted before transferring the procedure to CSMA/CD. This operation can be realized by using time-out counters in the case that a station fails.

c. Optimality

For light loads the access to the channel is granted immediately. If a collision occurs, the packet delay will be bounded since no more collisions are allowed after the first one.

For heavy loads the protocol uses a fast access method, that is waiting the time just necessary to be certain the previous station has declined its right to transmit.

For these reasons the protocol is expected to perform better than other protocols, including the adaptive tree walk and the urn protocols [6,21,33].

4.4 PERFORMANCE OF HYMAP

Due to the complexity of describing analytically the jump, back and forth, process from CSMA/CD to a collision-free protocol, we will only present results obtained from a simulation study.

The maximum throughput for HYMAP is easily obtained from fig 4.6, which represents the channel situation at a very high load.

It follows that:

$$Smax=Nt_p/(N(t_p+t_d+t_b)+2a) \qquad (4.3)$$

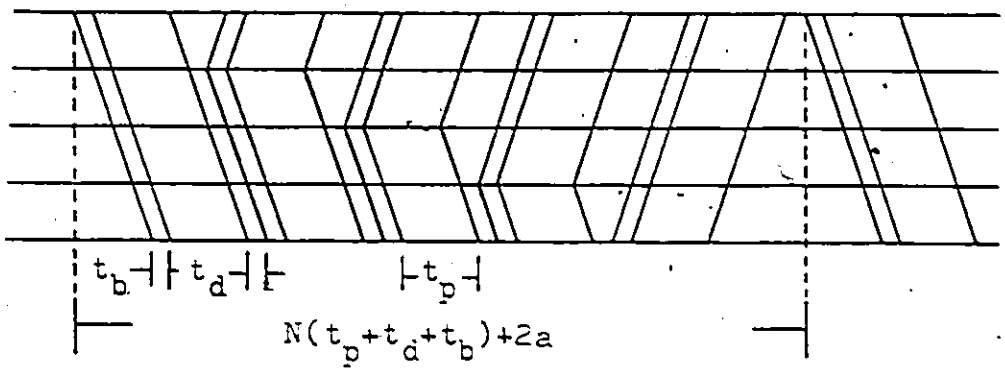


Fig 4.6 HYMAP Protocol at a Very High Load.

Using typical values (t_p=400 microsec, t_d=1.5 microsec and t_b=1 microsec), we obtain the following results:

	N=20	N=1000
a= 5	Smax= .9926	Smax= .9938
a=200	Smax= .9468	Smax= .9928

Comparing these results with the results obtained for the other protocols, it follows that HYMAP has a very good high load performance.

A program was designed to simulate HYMAP. The flow diagram of the program is shown in fig 4.7 and the program listing is included in the appendix.

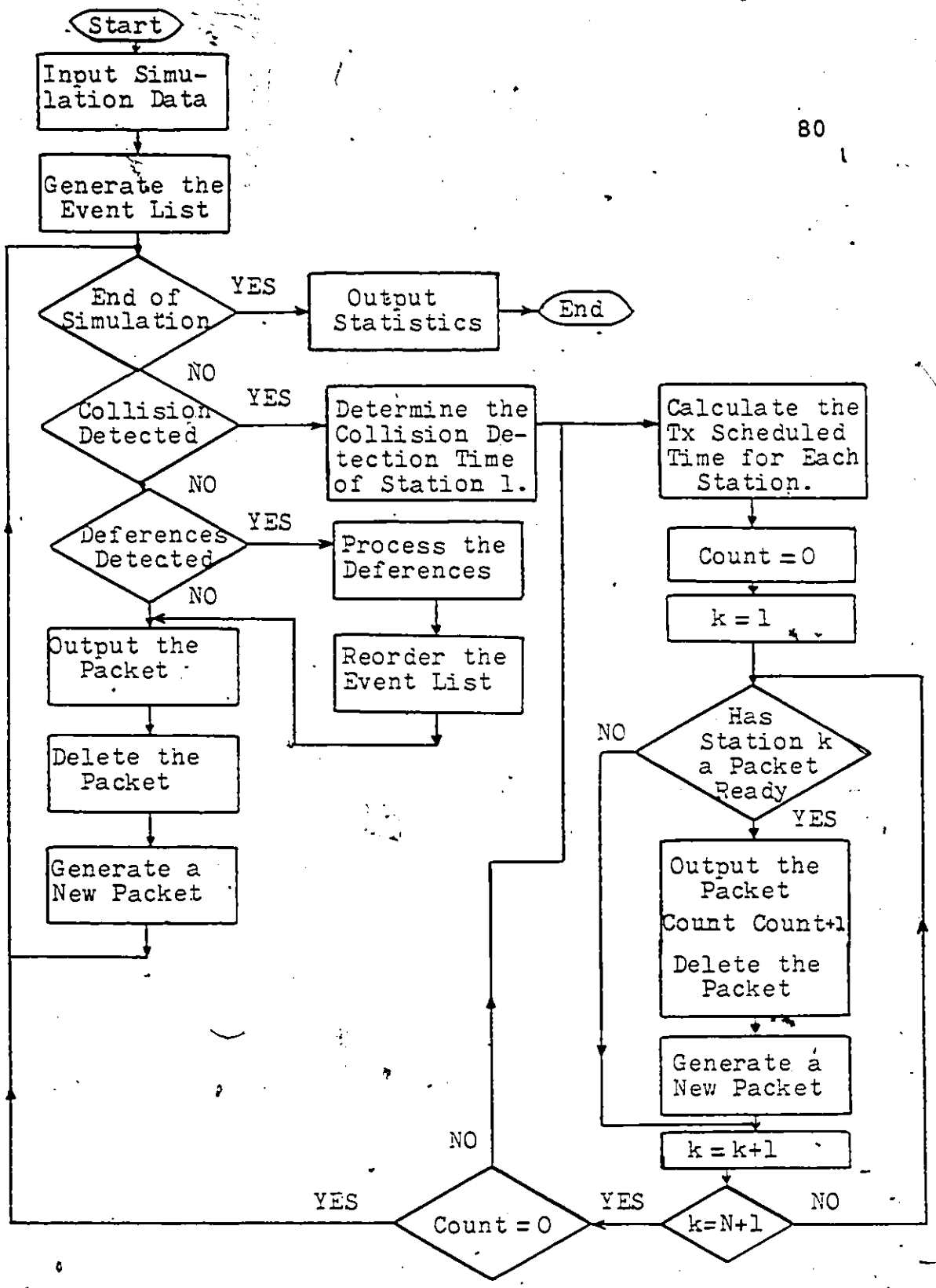
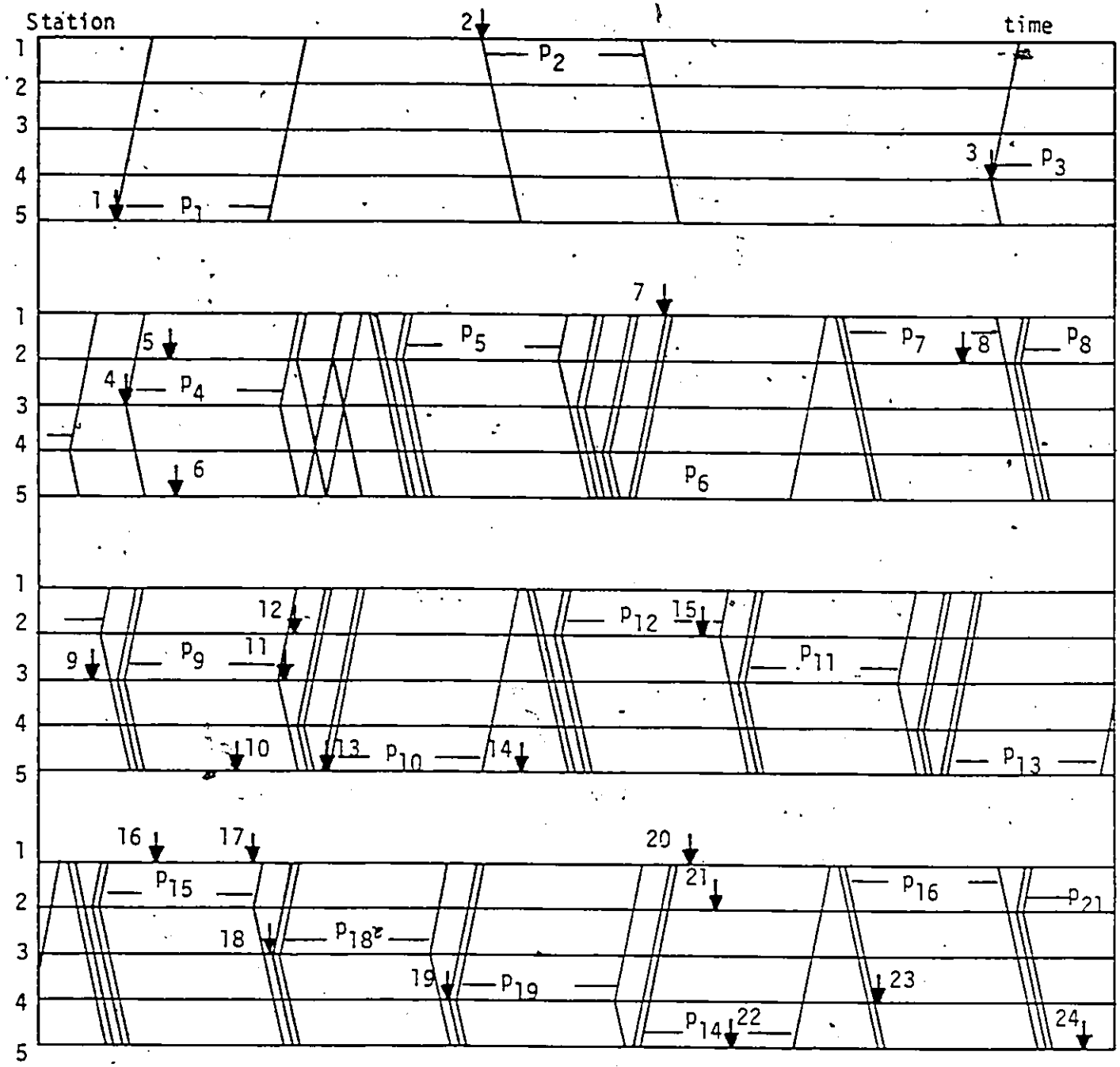


Fig 4.7 Flow Diagram of the HYMAP Simulation Program.

The simulation example for five stations is shown in fig 4.8. We observe that the collision of packets 5 and 6, after the transmission of packet 4, makes the system to jump to the collision-free procedure. After the channel is sensed idle by station 1, it starts sending the bit indication and the procedure continues as described before. The throughput reached is 0.6767 at an average delay of 180.5 microsec. These results compare favorably with all the previous examples shown.

Figures 4.9,4.10 and 4.11 show the delay-throughput characteristic of HYMAP, together with the characteristics of other protocols such as GBRAM, CSMA/CD and SDMA. It is clear from the figures the advantage of HYMAP over those protocols, both at light and heavy loads.

Table 4.1 shows the 95% confidence intervals of the simulation, expressed as a percentage of the mean value of the waiting delay. It follows then that the simulation results are reliable enough, especially considering that these confidence intervals were obtained using only 8 samples for each point [32].



t_p = 100 μsec
a/t_p = 0.25
t_d/t_p = 0.05
t_b/t_p = 0.05

Packets Generated = 24
Successful transmissions = 19
Collisions = 1

Throughput = .6767
Average Delay = 180.5 μs

Fig 4.8 HYMAP Simulation Example for Five Stations.

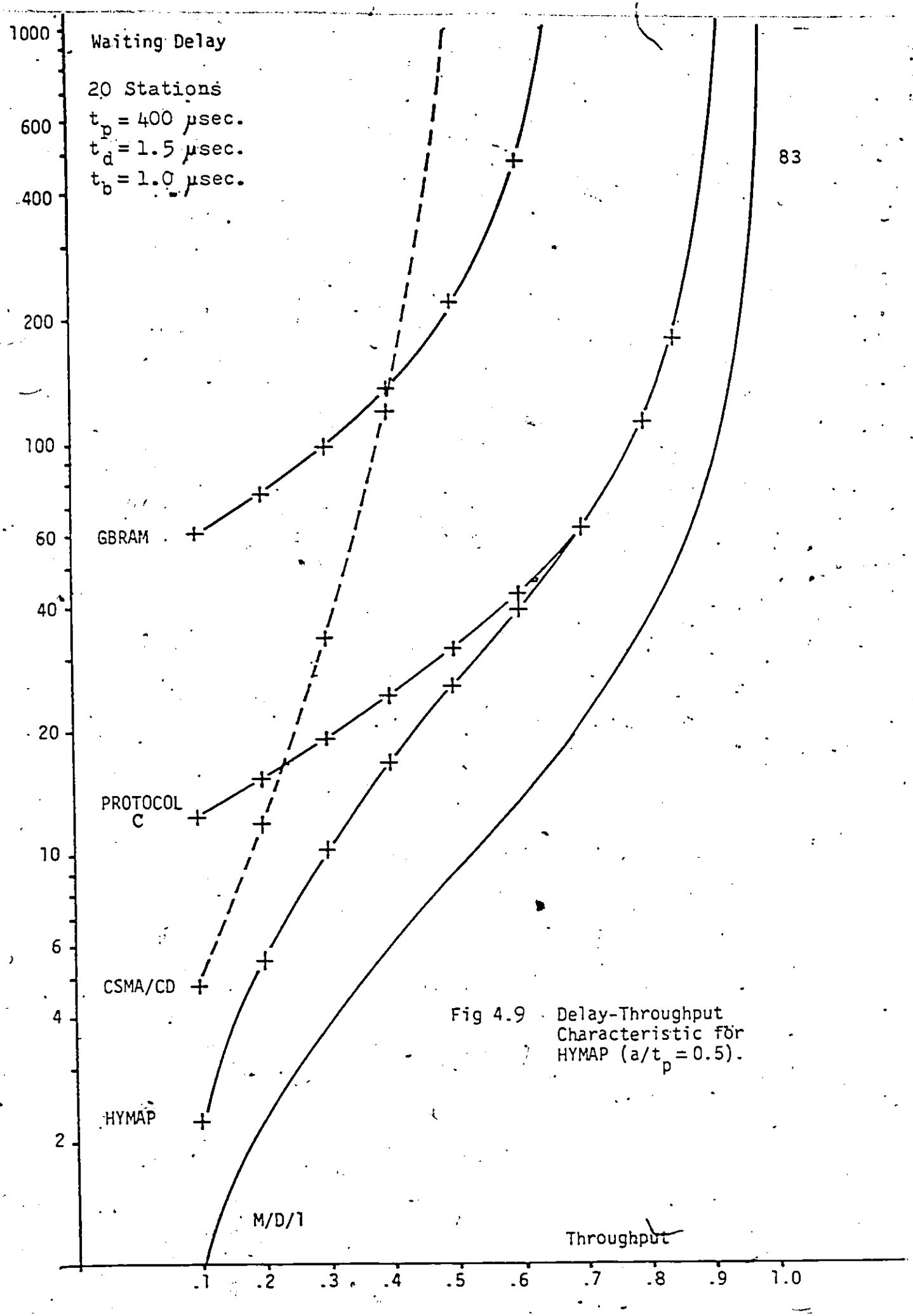


Fig 4.9 Delay-Throughput Characteristic for HYMAP ($a/t_p = 0.5$).

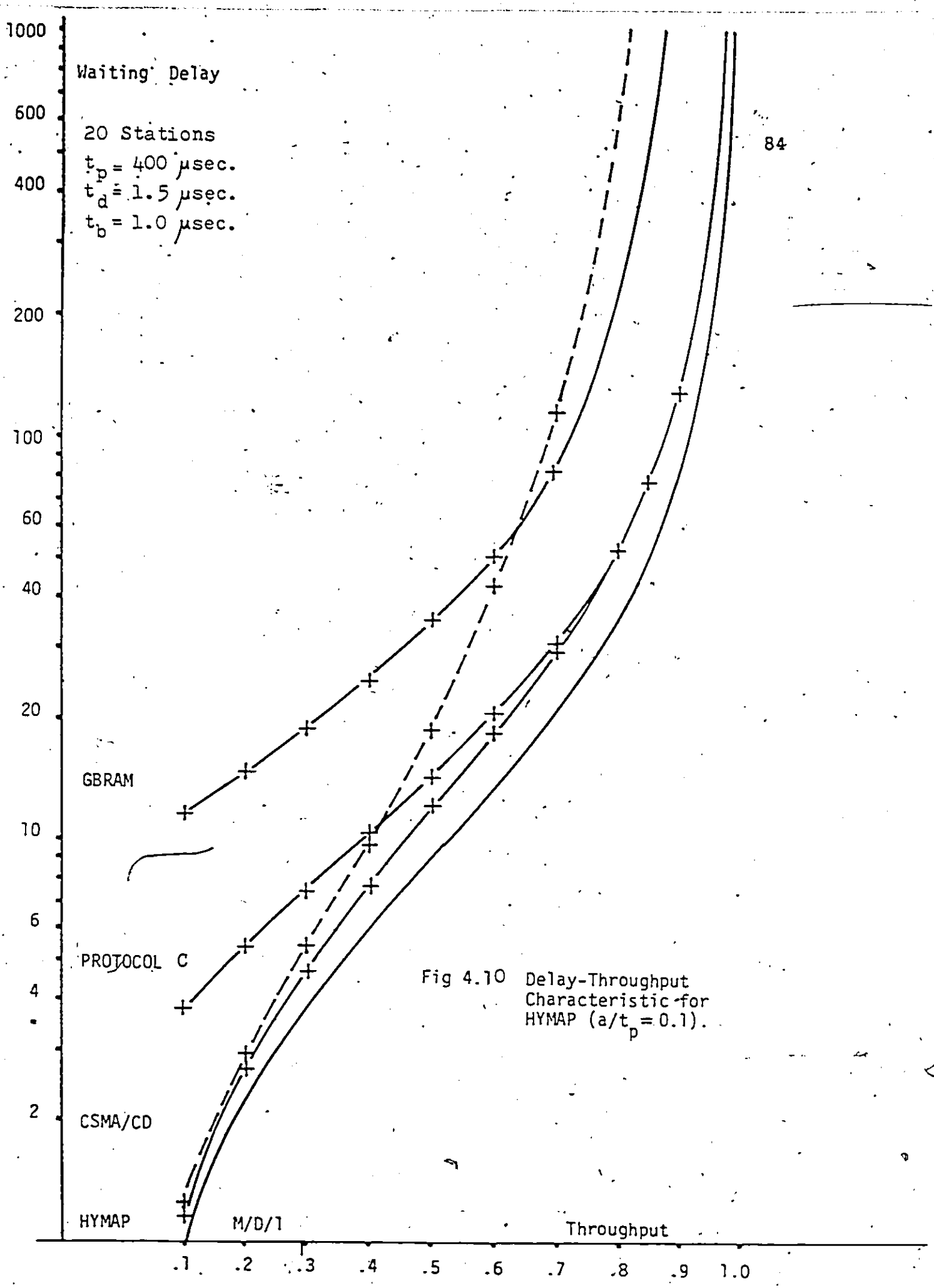


Fig 4.10 Delay-Throughput Characteristic for HYMAP ($a/t_p = 0.1$).

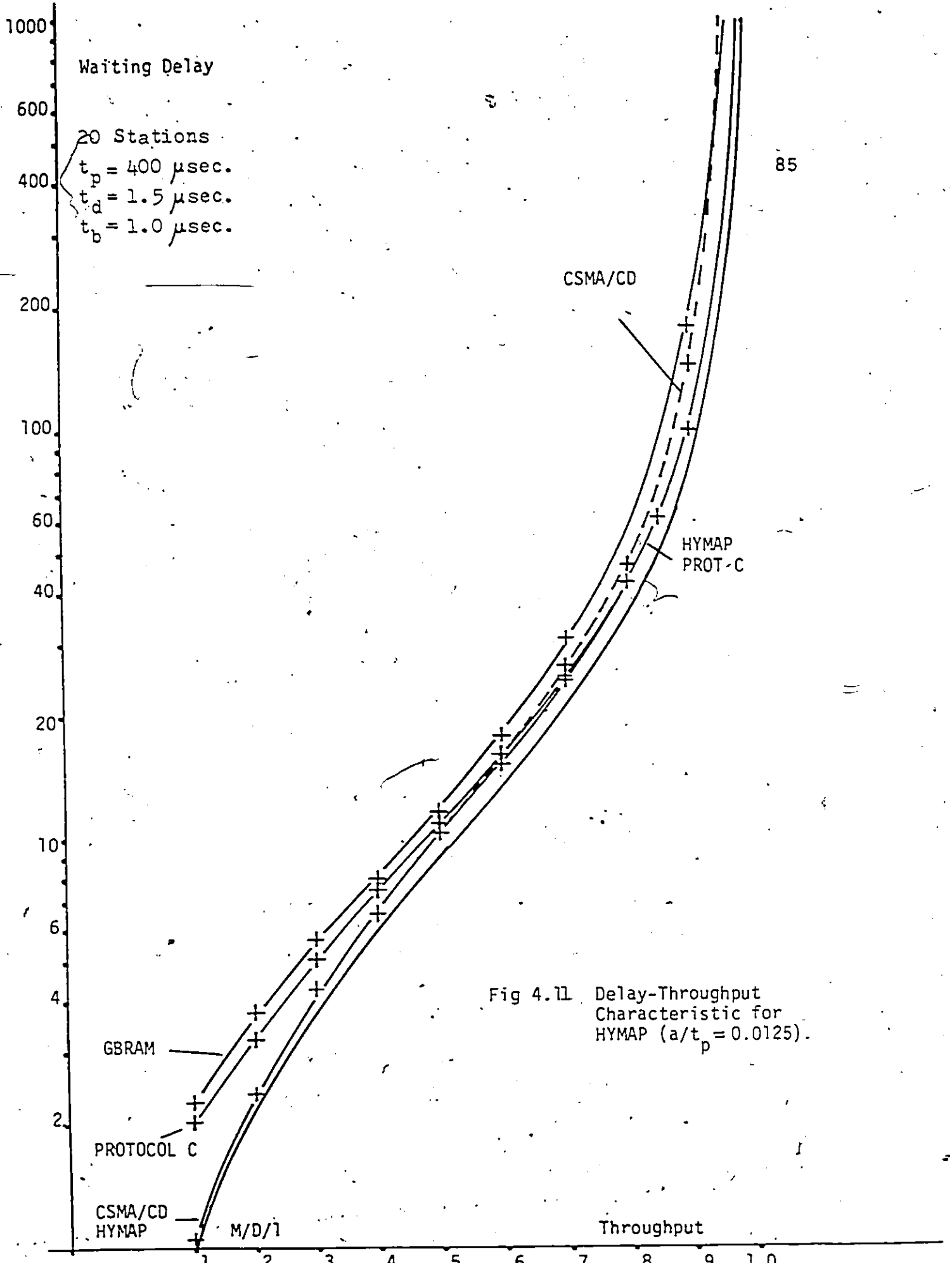


Fig 4.11 Delay-Throughput Characteristic for HYMAP ($a/t_p = 0.0125$).

THROUGHPUT	95 % CONFIDENCE INTERVALS		
	a/tp=0.5	a/tp=0.1	a/tp=0.0125
0.1	± 4.56	± 4.28	± 4.04
0.2	± 4.37	± 3.60	± 3.45
0.3	± 4.09	± 3.91	± 3.89
0.4	± 4.31	± 3.83	± 3.66
0.5	± 4.29	± 4.95	± 4.92
0.6	± 3.67	± 3.86	± 4.06
0.7	± 4.59	± 5.40	± 5.56
0.8	± 7.69	± 8.08	± 8.21

Table 4.1 95 % Confidence Intervals for HYMAP
(± % of the Mean Value).

The delay distribution per station is shown in fig 4.12, for both a low and a high throughput. It is observed that at low values of S, low numbered stations have lower mean delays, but the spreading is not important, especially considering the large value of a/t_p.

The distribution of the duration of the delays for CSMA/CD and HYMAP is shown in fig 4.13, for two values of S. It is seen that HYMAP has a lower spreading in the values of the delay duration, especially at S=.7. Over 10% of the

packets in CSMA/CD experience delays longer than 4.5 milisec, whereas only 1% of HYMAP packets experience such delays. Furthermore, 86.6% of HYMAP packets experience delays shorter than 1.5 milisec, as compared with 77.9% in the CSMA/CD case.

From the figures, it is clear that HYMAP performs better than the other protocols, especially when the propagation delay is comparable with the packet length.

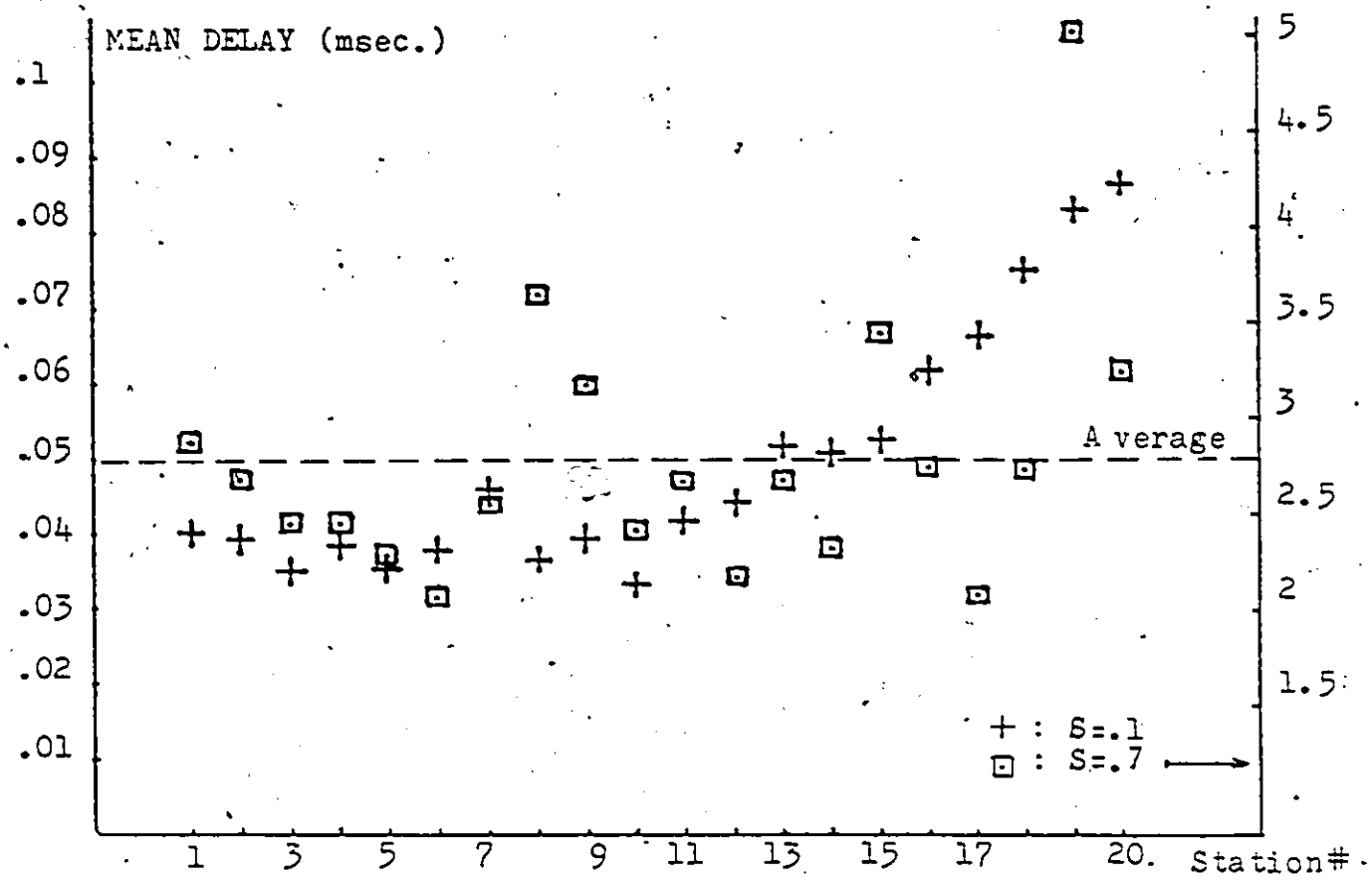


Fig4.12 Delay Distribution per Station. (a/t_p=.5)

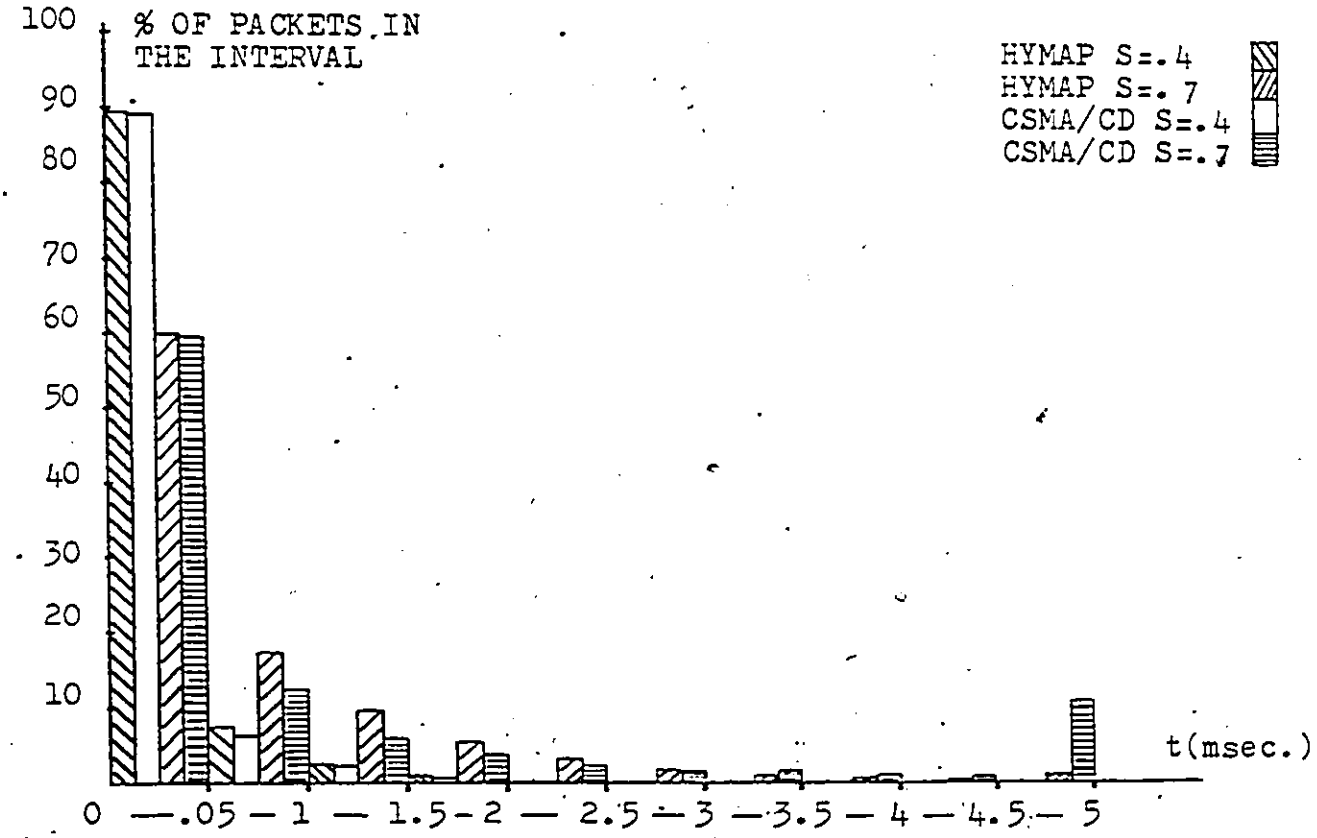


Fig4.13 Delay Distribution of HYMAP and CSMA/CD. (a/t_p=0.4)

4.5 VOICE PERFORMANCE OF HYMAP

A simulation analysis was done to evaluate the voice performance of HYMAP [2,4,12]. We assume that 2N voice stations are connected to a cable of length L (km). The stations are uniformly distributed along the cable.

The voice information is coded using a 64 kb/sec PCM scheme at each terminal and the voice information is framed into fixed length packets of 100 bytes (90 bytes correspond to voice information and 10 bytes to control information). So the voice information in each packet is 720 bits long and to feed the receiver with 64 kb/sec, each voice terminal must then transfer 89 packets/sec. It follows then that the interpacket arrival time equals 11.24 milisec.

The statistics for voice conversations [19], indicate the following averages in a typical call:

Talkspurt duration= 1.366 sec.
Silence duration = 1.802 sec.
Talking = 43.53 %
Double Talking = 6.58 %
Mutual Silence = 18.97 %

We can then assume that, in a 2N voice terminal system, only N voice stations are active, at any given time, talking to the rest (all this at the peak hour).

Let us now consider a 10 Mb/sec cable system. Depending on the multiplexing scheme we can conclude the following:

- A classical time division multiplex (TDM) system is able of supporting up to 156 half-duplex 64 kb/sec voice circuits. This translates to 78 full-duplex voice circuits.

- A synchronous packet system can support fewer half-duplex voice circuits because of the control overhead on each packet. For 100 bytes packets, as defined before, a synchronous packet system can support up to 140 half-duplex voice circuits.

It can be concluded then that HYMAP will support fewer than 140 half-duplex voice circuits, due to the non-synchronous nature of the protocol.

The program designed to simulate the voice performance of HYMAP, generates 89N packets/sec using an exponential distribution. This is a valid assumption if silence detection is being performed at each voice terminal. All packets with delay greater than 11.24 millisec. are considered lost. Hence, depending on the cable capacity and the number of stations, a different loss is obtained.

Usually a voice information loss of 2% or less is well tolerated by an average listener, so that value was used to determine the number of voice terminals that can be actively using the system.

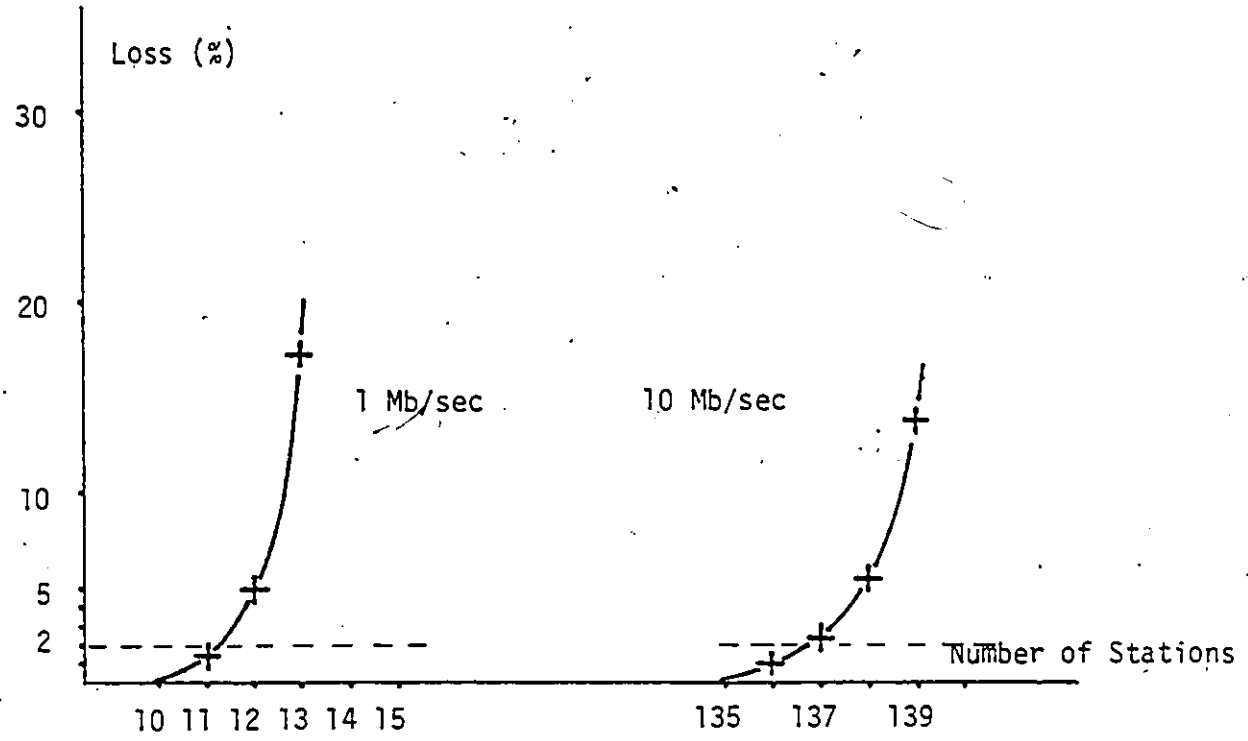


Fig 4.14 Voice Packets Loss Rate for Hymap.

The simulation results, shown in fig 4.14, indicate that for a system with a capacity of 1 Mb/sec. up to 11 voice terminals, at a throughput of .78, can be supported. At 10 Mb/sec., up to 137 voice terminals, at a throughput of .97, can be supported by HYMAP.

The actual number of voice terminals connected to the system, can be increased by considering the fact that no

station uses the channel 100 % of its active time, due to silences among words. So it is possible to connect roughly up to 2.5 N telephone circuits to the system, that is, 340 telephone circuits approximetely at 10 Mb/sec. Furthermore, using concentrators at each voice terminal, around 1000 telephones could be operated with a low blocking probability. [16]

These results compare favorably with the ones reported for CSMA/CD and GBRAM [15,16]. Those results indicate for CSMA/CD a capacity of 94 voice circuits at S= .66 and for GBRAM a capacity of 125 voice circuits at S= .85, both in 10 Mb/sec systems.

Chapter V

CONCLUSIONS

The thesis has addressed the problem of designing bus protocols in Local Area Networks. We have made extensive use of a graphical approach, introduced in chapter III, to develop these protocols and also to analyze others.

The graphical approach allowed us to understand very closely what the different protocols do when used in a cable system. It also helped us to design very reliable simulation programs, since an extensive use of practical information such as finite propagation delays, non zero turn-around time, uniform distribution of stations, etc., has been utilized on the programs.

In chapter II we analyzed and simulated CSMA/CD and GBRAM, two well known protocols that served us as starting points to develop the protocols described in chapter III and IV. It was concluded that CSMA/CD has a better performance than GBRAM at a light load, whereas GBRAM leads at high values of the load. In terms of complexity, CSMA/CD is simpler to implement and has the advantage of not require from the stations to be on at all times.

The protocols we developed in chapter III: DBRAM, protocol C, protocol D, protocol E and protocol F greatly improve the performance of GBRAM by using more efficiently the information available on the channel.

Protocol F is without question the best performer of all those protocols, if the delay-throughput characteristic is considered. However its complexity, particularly the problem of synchronization at light loads, is also bigger.

One particular disadvantage of most of the protocols described in chapter III (especially protocols D and E), is that all stations in the system, having or not ready packets, must be active all the time, sending reservation information. DBRAM, on the other hand, must have available at each station the position of all stations in the network, preventing so an easy change of the system's configuration.

Protocol C is an attractive alternative, since is a very simple scheme that only requires from the stations to wait the amount of time just necessary to determine if another station has started transmitting. It does not require that all stations be active at one time. Only the first station in the network must be active at all times, sending dummy packets, in order to achieve synchronization in the network.

All the previous protocols share a common disadvantage, with respect to CSMA/CD, that is their delay at light loads, is greater due to the scheduling scheme used.

HYMAP, the protocol analyzed in chapter IV, solves the problem mentioned before, by sharing the best features of CSMA/CD and the collision-free protocols. At light load it has even a better response than CSMA/CD, whereas at medium and heavy traffic it closely performs like protocol C.

It should be clear that the collision-free procedure is not protocol C, but a modified version of it that introduces bit indication sequences. As we commented in chapter IV, the main problem of HYMAP is for the stations to synchronize with the transmission of station 1 after a collision has taken place, and also when the traffic has decreased, to return to the CSMA/CD mode of operation. This is the reason bit indication sequences are used.

The performance of HYMAP should improve by first deleting the bit indication sequences in all but station 1 (by using a time-out counter to return to CSMA/CD) and secondly by using a protocol F modified procedure. This approach would also free all stations from the burden of being on at all times. The approach mentioned has been left for future work.

The voice performance of HYMAP was studied considering a system with only voice stations. The simulation results indicate an improved performance as compared with the one of CSMA/CD or GBRAM. Future work will have to deal with combined voice and data traffic and the use of a more accurate voice traffic model.

Summing up, the main contributions of this thesis we think have been:

- Use of graphical methods as an aid in the design of Local Area Networks protocols;
- Improved simulation analysis of CSMA/CD and GBRAM;
- Development and simulation analysis of five other protocols and
- Development and simulation analysis of HYMAP, a new hybrid multiple access protocol.

Appendix A

PROGRAM LISTINGS

A.1 DEFINITIONS OF TERMS AND VARIABLES

A.1.1 Packet and Channel Times

TJAM = Jamming time after a collision in CSMA/CD.

TSLOT = Slot-time, equals twice the end-to-end prop. time.

TPACK = Packet size.

TDEL = Turn around time.

TBIT = Bit-time duration.

TMAX = Simulated time.

TINI = Initial time.

TTRU = Final time

TFIN = Time of the last packet arrival.

TTDELY = Time window for obtaining the delay distribution.

A.1.2 Station and Event List Parameters

NDEVIC = Number of stations connected to the system.

ND(k) = Array containing the station number of packet k in the event list.

T(k) = Array containing the time value of packet k in the event list.

DELAY(k) = Array containing the delay value of packet k in the event list.

NCOLLI(k) = Array containing the No of collisions

of packet k in the event list.

NPACKT(k) = Array containing the No of packets outputed by station k.

TC(k) = Array containing the collision detection time of packet k in the event list.

STIMGE(k) = Array containing the summation of packet arrival times of station k.

STIME(k) = Array containing the summation of packet transmission times of station k.

LBUFF = Length of the event list.

A.1.3 Random Number Generator Parameters

M5,M6 = Random number generator seeds for the arrival process.

M7,M8 = Random number generator seeds for the backoff process.

LSEED = Random number generator offset.

A.2 CSMA/CD SIMULATION PROGRAM

The following pages show the program listing of the CSMA/CD simulation program. The other programs share most of CSMA/CD routines, so those routines will not be presented for sake of saving space. However, the organization of the programs will be included.

```
C       ************************************************
C       ***  CSMA/CD 2.1  SIMULATION  21/DEC/,       ****
C       ************************************************
        DIMENSION T(100),ND(100),NPACKT(100)
        DIMENSION DELAY(100),NCOLLI(100),LFCOL(100),LFDEF(100)
        DIMENSION TC(100),NUPA(100),STIMGE(100),STIME(100),SDELAY(100)
        DIMENSION SQUARE(100),NDEL(100),TS(100)
        COMMON T,ND,DELAY,NCOLLI,SWI,TC
C
C       **********  INPUT SIMULATION DATA *************
C
        KSIM=1
        KMAX=2
        TYPE 5
5       FORMAT(' INPUT TJAM,TSLOT,TPACK,TDEL,M5,M6,SEED')
        ACCEPT 6,TJAM,TSLOT,TPACK,TDEL,M5,M6,LSEED
6       FORMAT(E12.3,E12.3,E12.3,E12.3,I6,I6,I6)
        TYPE 666,TJAM,TSLOT,TPACK,TDEL,LSEED
666     FORMAT(' TJAM=',F12.8,' TSLOT=',F12.8,' TPACK=',F12.8,' TDEL=',
     N  F12.8,' LSEED=',I4)
        TYPE 1
1       FORMAT( ' INPUT NDEVIC,LBUFF,NPAC,L1 (0=PRI),NINTER,BACK.,TDEL')
        ACCEPT 2,NDEVIC,LBUFF,IN7,L1,L2,KS,TTDELY
2       FORMAT(I4,I4,I6,I4,I4,I4,F12.8)
        TYPE 222,NDEVIC,LBUFF,KS
222     FORMAT(' NDEVIC=',I4,' LBUFF=',I4,'BACKOFF..LIN=0,EXP=1',I3)
        TYPE 3
3       FORMAT(' INPUT ARRIVAL RATE (PAC/SEC), NUMBER OF ITERATIONS')
        ACCEPT 4,XINIC,LSTEP
4       FORMAT(F10.0,I6)
        VF=TSLOT/2/(NDEVIC-1)
        NINTER=L2
        TDIST=TTDELY/NINTER
        LSUP=LBUFF-1
```

```
      M7=0
      M8=0
      IF(LSEED.EQ.0)GOTO 90
      DO 2550 LD=0,LSEED
      X=RAN(M5,M6)
2550  CONTINUE
90    XLAMDA=XINIC
95    TMAX=IN7/XLAMDA
      SWI=0
      TYPE 91,XLAMDA,TMAX
91    FORMAT(' XLAMDA=',F9.2,' TMAX=',F12.6)
      TYPE 91,XLAMDA,TMAX
C
C     *** GENERATE THE EVENT LIST ***
C
      DO 155 KA=1,LBUFF
      CALL GENERA(KA,NDEVIC,0.,L1,TMAX,XLAMDA,M5,M6)
      STIMGE(ND(KA))=STIMGE(ND(KA))+T(KA)
155   CONTINUE
      NGPAC=LBUFF
      TINI=T(1)
C
C     *** START THE SIMULATION. STOP AT TMAX ***
C
17    IF(T(1).GE.TMAX)KSIM=KMAX
16    IF(T(1).EQ.999)GOTO 10
      TF=T(1)+TPACK
      IF(T(2).GT.T(1)+TSLOT/2)GOTO 175
      IF(ND(2).EQ.ND(1))GOTO 175
      DO 170 K=2,LBUFF
        KCOL=K
190     IF(T(K).GT.T(1)+TSLOT/2)GOTO 172
195     IF(T(K).LT.T(1)+ABS(ND(K)-ND(1))*VF+1.E-8)LFCOL(K)=1
        IF(LFCOL(K).EQ.1)LCOL=1
170   CONTINUE
      PAUSE 'BUFCOL'
      GOTO 70
172   IF(LCOL.EQ.0)GOTO 175
      LCOL=0
      GOTO 70
175   IF(T(2).GT.TF+TSLOT/2)GOTO 49
      DO 180 K=2,LBUFF
        KDEF=K
        IF(T(K).GT.TF+TSLOT/2)GOTO 60
        IF(T(K).LT.TF+ABS(ND(K)-ND(1))*VF+1.E-8)LFDEF(K)=1
180   CONTINUE
      PAUSE 'BUFDEF'
      GOTO 60
C
C     ===STATISTICS FOR OUTPUT PACKET===
C
```

```
49      LGAP=0
50      NF=ND(1)
        X=T(1)
        STIME(NF)=STIME(NF)+X
        KSEG=INT(DELAY(1)/TDIST)+1
        IF(KSEG.GT.NINTER)KSEG=NINTER
        NDEL(KSEG)=NDEL(KSEG)+1
        NPAC=NPAC+1
        NPACKT(NF)=NPACKT(NF)+1
        IF(DELAY(1).LT.0)PAUSE 'DEL<0'
        NSUMCO=NSUMCO+NCOLLI(1)
        SUMDEL=SUMDEL+DELAY(1)
        SDELAY(NF)=SDELAY(NF)+DELAY(1)
        X1=X-DELAY(1)
572     IF(NPAC.GT.KL*500)GO TO 57
        IF(L1.EQ.1)GO TO 59
        GO TO 58
57      TYPE 571,KL,INT(IN7*1./500)
571     FORMAT(' PROGRAM IN PROGRESS. SEGMENT # ',I4,' OF ',I4)
        KL=KL+1
        GOTO 572
58      TYPE 55,NF,NPACKT(NF),T(1),DELAY(1),NCOLLI(1),NUPA(NF)
55      FORMAT(' PA.',I4,' #PA.',I5,' OUT.',F12.8,' DEL.',F12.8,
     N  ' #CO.',I4,' QUEUE',I4)
59      NCOLLI(1)=0
        DELAY(1)=0
54      KTOP=1
        CALL DELETE(LBUFF,KTOP)
616     IF(KSIM.EQ.KMAX)GOTO 56
        NSPAC=NPAC
        TTRU=X+TPACK
52      CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
        IF(T(LBUFF).EQ.999)GOTO 250
        NGPAC=NGPAC+1
        TFIN=T(LBUFF)
        N1=ND(LBUFF)
        STIMGE(N1)=STIMGE(N1)+T(LBUFF)
        DO 245 KB=LSUF,1,-1
                IF(ND(KB).EQ.N1.AND.T(KB).GE.T(LBUFF))GOTO 249
245     CONTINUE
        GOTO 250
249     X=T(KB)-T(LBUFF)
        T(LBUFF)=T(KB)+TPACK+TDEL
        DELAY(LBUFF)=X+TPACK+TDEL
        GOTO 250
56      T(LBUFF)=999
        GOTO 17
250     IF(I8.NE.0)GOTO 601
        IF(LFLAG1.EQ.1)GOTO 251
        GOTO 17
251     LFLAG1=0
```

```
        GOTO 254
10      CONTINUE
C
C       *** OUTPUT FINAL STATISTICS ***
C
        TRUPUT=(NSPAC*TPACK)/(TTRU-TINI)
        AVIPAT=(TFIN-TINI)/NGPAC
        APACSE=1/AVIPAT
        DO 1010 KG=1,LBUFF
        TDELT=STIME(KG)-STIMGE(KG)
        TOTDEL=TOTDEL+TDELT
        IF(KG.EQ.NDEVIC+1)L9=1
        IF(L9.EQ.1)GO TO 1010
        TYPE 1020,KG,NPACKT(KG),TDELT,SDELAY(KG)
1020    FORMAT(' DEVICE',I6,' OUTPUT',I6,' DELAY',F12.8,F12.8)
1010    CONTINUE
        L9=0
        TYPE 1000,NPAC,SUMDEL,TRUPUT
1000    FORMAT(' NPAC=',I6,' CHDEL=',F12.8,' TROUGHPUT=',F6.4)
        TYPE 1001,AVIPAT,APACSE
1001    FORMAT(' AV.ARRIV.TIME',F12.8,' AV.PACK PER SEC.',F12.2)
        X=TOTDEL/NGPAC
        Y=SUMDEL/NPAC
        TYPE 1003,X,Y
1003    FORMAT(' AV.DELAY',F12.8,' CHAN/TOT DEL',F12.8)
        TYPE 1004,NGPAC,NSPAC
1004    FORMAT(' OVER A TOTAL OF ',I6,' PAC. ',I6,' PAC. OUTPUT ON TIME')
        DO 1050 K=1,NINTER
          TYPE 1055,K,NDEL(K)
1055      FORMAT(' INTERVAL ',I4,' FREQUENCY ',I4)
          J=INT(NDEL(K)*100./NPAC)
1056      FORMAT(F<J+2>.<J+1>)
          NDEL(K)=0
1050    CONTINUE
C
C       *** CLEAR COUNTERS AND CONTINUE ***
C
        DO 92 L0=1,LBUFF
        NCOLLI(L0)=0
        DELAY(L0)=0
92      CONTINUE
        KL=0
        DO 93 L7=1,NDEVIC
        NUPA(L7)=0
        STIME(L7)=0
        STIMGE(L7)=0
        NPACKT(L7)=0
93      CONTINUE
        TF=0
        KSIM=1
        NPAC=0
```

```
      SDEL=0
      TYPE 3333,M5,M6
3333  FORMAT(' M5=',I6,' M6=',I6)
      TYPE 3333,M5,M6
      TOTDEL=0
      SUMDEL=0
      LSTEP=LSTEP-1
      IF(LSTEP.EQ.0)STOP 'END OF RUN'
      GO TO 95
C
C     ===DEFERENCE ROUTINE===
C
60    DO 100 J=2,KDEF
        IF(LFDEF(J).EQ.0)GOTO 100
        TX=TF+ABS(ND(J)-ND(1))*VF+TDEL
        DELAY(J)=DELAY(J)+TX-T(J)
        T(J)=TX
        LFDEF(J)=0
        IF(L1.EQ.1)GOTO 100
        TYPE 1009,J,ND(J),T(J)
1009    FORMAT(' DEFERENCE',I4,I4,F12.8)
100   CONTINUE
      LINDIC=KDEF
      LFLAG1=1
      GOTO 3000
C
C     ===COLLISION ENFORCEMENT ROUTINE===
C
70    DO 500 L=1,KCOL
        DO 510 J=L,KCOL
              IF(J.EQ.L)GOTO 510
              IF(LFCOL(J).EQ.0)GOTO 510
              IF(ABS(T(J)-T(1)).GT.ABS(ND(J)-ND(1))*VF+1.E-8)GOTO 510
              X=ABS(ND(J)-ND(L))*VF
              TC1=T(L)+X
              TC2=T(J)+X
              IF(TC(J).EQ.0.OR.TC1.LT.TC(J))TC(J)=TC1
              IF(TC(L).EQ.0.OR.TC2.LT.TC(L))TC(L)=TC2
              TS1=TC(L)+TJAM+X
              TS2=TC(J)+TJAM+X
              IF(TS1.GT.TS(J))TS(J)=TS1
              IF(TS2.GT.TS(L))TS(L)=TS2
      IF(L1.EQ.1)GOTO 510
      TYPE 521,L,J,TC1,TC2,TS1,TS2
521   FORMAT(' L=',I4,'J=',I4,'TC1=',F12.8,'TC2=',F12.8,'TS1=',F12.8,
     N 'TS2=',F12.8)
510   CONTINUE
      IF(T(L).GT.T(1)+TSLOT/2)GOTO 550
500   CONTINUE
550   DO 600 J=1,LBUFF
              IF(TC(J).EQ.0)GOTO 650
```

```
                IF(NCOLLI(J).NE.15)GOTO 605
                KSEG=INT(DELAY(J)/TDIST)+1
                IF(KSEG.GT.NINTER)KSEG=NINTER
                NDEL(KSEG)=NDEL(KSEG)+1
                X=T(J)-DELAY(J)
                SUMDEL=SUMDEL+DELAY(J)
                SDELAY(ND(J))=SDELAY(ND(J))+DELAY(J)
                STIME(ND(J))=STIME(ND(J))+TC(J)
                TYPE 606,X,ND(J),TC(J),NUPA(ND(J))
606             FORMAT(' PACKET ',F12.8,'BY DEV.',I4,' AT 'F12.8,I5)
                GOTO 615
605             NCOLLI(J)=NCOLLI(J)+1
                TX1=TC(J)+TJAM+TSLOT*LBAKOF(ND(J),NCOLLI(J),K5,M7,M8)
                IF(TX1.LE.TS(J))TX1=TS(J)+TDEL
                DELAY(J)=DELAY(J)+TX1-T(J)
          IF(DELAY(J).LT.0)PAUSE 'COL DEO'
                T(J)=TX1
          LFLAG1=0
          IF(L1.EQ.1)GOTO 600
          TYPE 6669,J,ND(J),T(J),NCOLLI(J)
6669      FORMAT('--->',I4,I4,F12.8,I4)
600       CONTINUE
          PAUSE 'ERROR 1'
          LINDIC=LBUFF
          GOTO 254
615       CALL DELETE(LBUFF,J)
          I8=I8+1
          GOTO 616
601       I8=I8-1
          GOTO 600
C
C         ===ADDITIONALDEFERENCE ROUTINE===
C
650       LFLAG3=1
          GOTO 3000
630       LFLAG3=0
          DO 800 M=2,LBUFF
            IF(LFCOL(M).EQ.1.OR.T(M).GT.T(1)+TSLOT+TJAM)GOTO 800
            TX=T(M)
            LUP=KCOL
            DO 850 L=1,LUP
                  IF(LFCOL(L).EQ.1)GOTO 850
                  T1=TC(L)+TJAM+ABS(ND(M)-ND(L))*VF+TDEL
                  IF(T1.GT.TX)TX=T1
850         CONTINUE
            DELAY(M)=DELAY(M)+TX-T(M)
            T(M)=TX
800       CONTINUE
2500      DO 2510 K8=1,LBUFF
          LFCOL(K8)=0
          TS(K8)=0
```

```
      TC(KS)=0
2510  CONTINUE
      LINDIC=LBUFF
      GOTO 254
C
C     ===ABSORBING ROUTINE===
C
3000  DO 3010 KD=1,LSUP
        DO 3020 KE=KD+1,LBUFF
        IF(T(KE).EQ.999.OR.ND(KE).NE.ND(KD))GOTO 3020
        IF(T(KE).GT.T(KD))GOTO 3020
        X4=T(KE)
3025    T(KE)=T(KD)+TBACK+TDEL
        DELAY(KE)=DELAY(KE)+T(KE)-X4
3020    CONTINUE
3010  CONTINUE
      IF(LFLAG3.EQ.1)GOTO 630
      IF(LFLAG1.EQ.1)GOTO 50
C
C     ===REORDER ROUTINE SHELL SORT===
C
254   LGAP=LINDIC
255   IF(LGAP.LE.1)GOTO 260
      LGAP=LGAP/2
270   NSWIT=0
      MAX=LINDIC-LGAP
      DO 200 I=1,MAX
            IF(T(I).LE.T(I+LGAP))GOTO 290
            TEMP=T(I)
            T(I)=T(I+LGAP)
            T(I+LGAP)=TEMP
            LTEMP=ND(I)
            ND(I)=ND(I+LGAP)
            ND(I+LGAP)=LTEMP
            TEMP=DELAY(I)
            DELAY(I)=DELAY(I+LGAP)
            DELAY(I+LGAP)=TEMP
            LTEMP=NCOLLI(I)
            NCOLLI(I)=NCOLLI(I+LGAP)
            NCOLLI(I+LGAP)=LTEMP
            NSWIT=1
290   CONTINUE
200   CONTINUE
      IF(NSWIT.NE.0)GOTO 270
      GOTO 255
260   CONTINUE
      IF(KSIM.EQ.KMAX)GOTO 16
      GOTO 17
      END
C
      SUBROUTINE DELETE(LBUFF,I1)
```

```
C
      COMMON T(100),ND(100),DELAY(100),NCOLLI(100),SWI,TC(100)
      DO  700 J=I1,LBUFF
      DELAY(J)=DELAY(J+1)
      NCOLLI(J)=NCOLLI(J+1)
      T(J)=T(J+1)
      TC(J)=TC(J+1)
      ND(J)=ND(J+1)
 700  CONTINUE
      RETURN
      END
C
      SUBROUTINE GENERA(LBUFF,NDEVIC,X3,L1,TMAX,XLAMDA,M5,M6)
C
      COMMON T(100),ND(100),DELAY(100),NCOLLI(100),SWI
      IF(SWI.EQ.1)GO TO 67
      X1=X3
      SWI=1
 67   CONTINUE
      X=RAN(M5,M6)*NDEVIC
      X=X+1.
      ND(LBUFF)=INT(X)
      X=RAN(M5,M6)
      X2=X1-ALOG(X)/XLAMDA
      X1=X2
      T(LBUFF)=X2
      IF(T(LBUFF).GT.TMAX)T(LBUFF)=999
      IF(L1.EQ.1)RETURN
      TYPE 43, LBUFF,ND(LBUFF),T(LBUFF)
 43   FORMAT(I4,I6,F12.8)
      RETURN
      END
C
      FUNCTION LBAKOF(J0,K0,KS,M7,M8)
C
      IF(KS.NE.0)GOTO 910
C
C     ===LINEAR BACKOFF===
C
      IF(K0.LT.8)GOTO 920
      N=50
      GOTO 930
920   N=2+6*K0
930   LBAKOF=INT(RAN(M7,M8)*N)
      RETURN
C
C     ===EXPONENTIAL BACKOFF===
C
910   IF(K0.LT.8)GOTO 915
      N=255
      GOTO 930
915   N=2**K0-1
      GOTO 930
      END
```

A.3 GBRAM SIMULATION PROGRAM

The simulation program for GBRAM consists of the following routines, some of them have already been presented in the previous section:

- Input Simulation Data;
- Generate the Event List;
- Group and Position Routine;
- Start Simulation, Stop at TMAX;
- Output Final Statistics;
- Clear Counters and Continue;
- Subroutine SCHEDU;
- Subroutine DELETE;
- Subroutine GENERA.

A.4 DBRAM SIMULATION PROGRAM

In this case the organization of the program is:

- Input Simulation Data;
- Generate the Event List;
- Start the Simulation, Stop at TMAX;
- Output Final Statistics;
- Clear Counters and Continue;
- Subroutine SCHEDU;
- Subroutine DELETE and
- Subroutine GENERA.

A.5 PROTOCOL D SIMULATION PROGRAM

Here the organization of the program is:

- Input Simulation Data;
- Generate the Event List;
- Start the Simulation, Stop at TMAX;
- Output Final Statistics;
- Clear Counters and Continue;
- Subroutine SCHEDU;
- Subroutine DELETE;
- Subroutine GENERA.

```
C       *** START THE SIMULATION. STOP AT TMAX ***
C                                                                  GBRAM
        NF=1
        CALL SCHEDU(1,1,TDEL)
        IF(T(1).LT.TS(ND(1)))GO TO 1330
1310    X=T(1)-TS(ND(1))
        DO 1340 K=2,LBUFF
          Y=T(K)-TS(ND(K))
          IF(Y.LT.X)X=Y
1340    CONTINUE
        LX=INT(X/DPER)
1350    DO 1300 K=1,NDEVIC
          TS(K)=TS(K)+LX*DPER
1300    CONTINUE
        LX=1
        IF(T(1).GE.TS(ND(1)))GO TO 1350
1320    IF(T(1).GE.TS(ND(1)))GO TO 1310
1330    DELAY(1)=TS(ND(1))-T(1)
        TF=TS(ND(1))+TPACK
        NF=ND(1)
        STIME(NF)=STIME(NF)+TS(NF)
        NPAC=NPAC+1
        NPACKT(NF)=NPACKT(NF)+1
        SUMDEL=SUMDEL+DELAY(1)
        X1=TS(NF)-DELAY(1)
        IF(NPAC.GT.KL*500)GO TO 57
        IF(L1.EQ.1)GO TO 59
        GO TO 58
57      IF(NPAC.GT.KL*500+2)KL=KL+1
58      TYPE 55,NF,TS(NF),DELAY(1),NPACKT(NF),X1
55      FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC ',I4,
     N  F12.8)
59      CALL DELETE(LBUFF,1)
        IF(KSIM.EQ.KMAX)GO TO 56
        NSPAC=NPAC
        TTRU=TF
56      CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
        IF(T(LBUFF).EQ.999)GO TO 250
        NGPAC=NGPAC+1
        TFIN=T(LBUFF)
        STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
250     LGROUP=LGR(NF)
        LPOSIT=LPO(NF)
        CALL SCHEDU(LGROUP,LPOSIT,TDEL)
        NS=1
        DO 1410 J=2,LBUFF
          IF(T(J).GT.TS(ND(J)))GO TO 1410
          IF(TS(ND(J)).LT.TS(ND(NS)))NS=J
1410    CONTINUE
        IF(TS(1).GE.TMAX)KSIM=KMAX
        IF(T(1).EQ.999)GO TO 10
        IF(NS.EQ.1)GO TO 1320
        TTEMP=T(NS)
        NTEMP=ND(NS)
1430    T(NS)=T(NS-1)
        ND(NS)=ND(NS-1)
        NS=NS-1
        IF(NS.EQ.1)GO TO 1420
        GO TO 1430
1420    T(1)=TTEMP
        ND(1)=NTEMP
        GO TO 1330
10      CONTINUE
```

```
C
C       *** START THE SIMULATION. STOP AT TMAX ***
C
        NF=1
        CALL SCHEDU(1,TDEL)
        IF(T(1).LT.TS(ND(1)))GO TO 1330
1310    X=T(1)-TS(ND(1))
        DO 1340 K=2,LBUFF
          Y=T(K)-TS(ND(K))
          IF(Y.LT.X)X=Y
1340    CONTINUE
        LX=INT(X/DPER)
1350    DO 1300 K=1,NDEVIC
          TS(K)=TS(K)+LX*DPER
1300    CONTINUE
        LX=1
        IF(T(1).GE.TS(ND(1)))GO TO 1350
1320    IF(T(1).GE.TS(ND(1)))GO TO 1310
1330    DELAY(1)=TS(ND(1))-T(1)
        TF=TS(ND(1))+TPACK
        NF=ND(1)
        STIME(NF)=STIME(NF)+TS(NF)
        NPAC=NPAC+1
        NPACKT(NF)=NPACKT(NF)+1
        SUMDEL=SUMDEL+DELAY(1)
        X1=TS(NF)-DELAY(1)
        IF(NPAC.GT.KL*500)GO TO 57
        IF(L1.EQ.1)GO TO 59
        GO TO 58
57      IF(NPAC.GT.KL*500)KL=KL+1
58      TYPE 55,NF,TS(NF),DELAY(1),NPACKT(NF)
55      FORMAT(I4,F12.8,F12.8,I6)
59      CALL DELETE(LBUFF,1)
        IF(KSIM.EQ.KMAX)GO TO 56
        NSPAC=NPAC
        TTRU=TF
56      CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAM01,M5,M6)
        IF(T(LBUFF).EQ.999)GO TO 250
        NGPAC=NGPAC+1
        TFIN=T(LBUFF)
        STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
250     CALL SCHEDU(NF,TDEL)
        NS=1
        DO 1410 J=2,LBUFF
          IF(T(J).GT.TS(ND(J)))GO TO 1410
          IF(TS(ND(J)).LT.TS(ND(NS)))NS=J
1410    CONTINUE
        IF(TS(1).GE.TMAX)KSIM=KMAX
        IF(T(1).EQ.999)GO TO 10
        IF(NS.EQ.1)GO TO 1320
        TTEMP=T(NS)
        NTEMP=ND(NS)
1430    T(NS)=T(NS-1)
        ND(NS)=ND(NS-1)
        NS=NS-1
        IF(NS.EQ.1)GO TO 1420
        GO TO 1430
1420    T(1)=TTEMP
        ND(1)=NTEMP
        GO TO 1330
10      CONTINUE
```

*** PROTOCOL D ***

```
C
C       *** START THE SIMULATION. STOP AT TMAX ***
C
        NF=1
1320    CALL SCHEDU(TBIT,TDEL)
        IF(T(1).LT.TF)GO TO 1330
1310    X=T(1)-TF
        LX=INT(X/DPER)
        IF(DPER*FRAC(X/DPER).GT.(ND(1)-1)*(VF+TBIT))LX=LX+1
        TF=TF+LX*DPER
1350    DO 1300 K=1,NDEVIC
          TS(K)=TS(K)+LX*DPER
1300    CONTINUE
1330    DO 5100 K=NDEVIC,1,-1
          DO 5110 J=1,LBUFF
            IF(ND(J).NE.K)GO TO 5110
              IF(T(J).GT.TF+(K-1)*(VF+TBIT))GO TO 5100
                DELAY(K)=TS(K)-T(J)
                SQUARE(K)=SQUARE(K)+DELAY(K)*DELAY(K)
                TOUT=TS(K)
                DO 5200 L=K-1,1,-1
                  TS(L)=TS(K)+TPACK+TDEL+(K-L)*VF
5200            CONTINUE
                NF=K
                STIME(K)=STIME(K)+TS(K)
                NPAC=NPAC+1
                NPACKT(NF)=NPACKT(NF)+1
                SUMDEL=SUMDEL+DELAY(K)
                X1=TS(NF)-DELAY(K)
                IF(NPAC.GT.KL*500)GO TO 57
                IF(L1.EQ.1)GO TO 59
                GO TO 58
57              IF(NPAC.GT.KL*500)KL=KL+1
58              TYPE 55,NF,TS(NF),DELAY(K),NPACKT(NF),X1
55              FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC ',
     N          I4,F12.8)
59              CALL DELETE(LBUFF,J)
                IF(KSIM.EQ.KMAX)GO TO 56
                NSPAC=NPAC
                TTRU=TF
56              CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
                IF(T(LBUFF).EQ.999)GO TO 250
                NGPAC=NGPAC+1
                TFIN=T(LBUFF)
                STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
250             IF(TS(1).GE.TMAX)KSIM=KMAX
                IF(T(1).EQ.999)GO TO 10
                GO TO 5100
5110        CONTINUE
5100    CONTINUE
        TF=TOUT+TPACK+TDEL+(NF-1)*VF
        GO TO  1320
10      CONTINUE
```

```
C       *** GROUP AND POSITION ROUTINE ***
        DO 1200 K=1,NDEVIC
          LGR(K)=INT((K-1)*NGR*1./NDEVIC)+1
          LPO(K)=K-NDEVIC/NGR*(LGR(K)-1)
1200    CONTINUE
C
C
        SUBROUTINE SCHEDU(LG,LP,TDEL)
C
        COMMON NDEVIC,NGR,NF,VF,TF,TSLOT,TSLOT1,LGR(100),LPO(100)
        COMMON T(100),ND(100),SWI,TS(100)
        DO 2100 K=1,NDEVIC
          IF(LGR(K).EQ.LG)GO TO 2110
            TS(K)=TF+IABS(NF-K)*VF+TSLOT*FLOAT(LGR(K)-LG+NGR)*1./NGR
     N            TSLOT1*(LPO(K)-1)+TDEL
            GO TO 2100
2110        IF(LPO(K).GT.LP)GO TO 2120
              TS(K)=TF+IABS(NF-K)*VF+TSLOT*NGR+TSLOT1*(LPO(K)-1)+TDEL
              GO TO 2100
2120          TS(K)=TF+IABS(NF-K)*VF+TSLOT1*(LPO(K)-LP)+TDEL
2100    CONTINUE
        RETURN
        END
```

**** GBRAM ROUTINES ****

```
C
        SUBROUTINE SCHEDU(LAST,TDEL)
C
        COMMON NDEVIC,NGR,NF,VF,TF,TSLOT,TSLOT1,LGR(100),LPO(100)
        COMMON T(100),ND(100),SWI,TS(100)
        DO 2100 K=1,NDEVIC
          IF(K.LE.LAST)GO TO 2110
            TS(K)=TF+2*VF*(K-LAST)+TDEL
            GO TO 2100
2110        TS(K)=TF+(4*(NDEVIC-1)+2*(K-LAST))*VF+TDEL
2100    CONTINUE
        RETURN
        END
```

**** DBRAM SCHEDULING ROUTINE ****

```
C
        SUBROUTINE SCHEDU(TEST,TDEL)
C
        COMMON NDEVIC,NGR,NF,VF,TF,TSLOT,TSLOT1,LGR(100),LPO(100)
        COMMON T(100),ND(100),SWI,TS(100)
        DO 2100 J=1,NDEVIC
          TS(J)=(2*NDEVIC-J-1)*VF+NDEVIC*TSLOT+TDEL+TF
2100    CONTINUE
        RETURN
        END
```

**** PROTOCOL D SCHEDULING ROUTINE ****

A.6 PROTOCOL C SIMULATION PROGRAM

Here the program is as follows:

- Input Simulation Data;
- Generate the Event List;
- Start the Simulation, Stop at TMAX;
- Output Final Statistics;
- Clear Counters and Continue;
- Subroutine SCHEDU;
- Subroutine DELETE;
- Subroutine GENERA.

A.7 PROTOCOL F SIMULATION PROGRAM

This program differs from the previous one only in the main body (Start the Simulation, Stop at TMAX) and in the Subroutine SCHEDU.

A.8 HYMAP SIMULATION PROGRAM

The organization is as follows:

- Input Simulation Data;
- Generate the Event List;
- Start the Simulation, Stop at TMAX;
- Deference Routine;
- Collision Enforcement Routine;
- Start Collision-Free Procedure;
- Reorder the Event List;
- Subroutine SCHEDU;

- Subroutine DELETE;
- Subroutine GENERA.

```
                ****   PROTOCOL C ****
C
C       *** START THE SIMULATION. STOP AT TMAX ***
C
        NF=1
        CALL SCHEDU(1,TDEL)
        IF(T(1).LT.TS(ND(1)))GO TO 1330
1310    X=T(1)-TS(ND(1))
        DO 1340 K=2,LBUFF
          Y=T(K)-TS(ND(K))
          IF(Y.LT.X)X=Y
1340    CONTINUE
        LX=INT(X/DPER)
1350    DO 1300 K=1,NDEVIC
          TS(K)=TS(K)+LX*DPER
1300    CONTINUE
        LX=1
        IF(T(1).GE.TS(ND(1)))GO TO 1350
1320    IF(T(1).GE.TS(ND(1)))GO TO 1310
1330    DELAY(1)=TS(ND(1))-T(1)
        SQUARE(ND(1))=SQUARE(ND(1))+DELAY(1)*DELAY(1)
        TF=TS(ND(1))+TPACK
        NF=ND(1)
        STIME(NF)=STIME(NF)+TS(NF)
        NPAC=NPAC+1
        NPACKT(NF)=NPACKT(NF)+1
        SUMDEL=SUMDEL+DELAY(1)
        X1=TS(NF)-DELAY(1)
        IF(NPAC.GT.KL*500)GO TO 57
        IF(L1.EQ.1)GO TO 59
        GO TO 58
57      IF(NPAC.GT.KL*500+2)KL=KL+1
58      TYPE 55,NF,TS(NF),DELAY(1),NPACKT(NF),X1
55      FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC ',I4,F12.8)
59      CALL DELETE(LBUFF,1)
        IF(KSIM.EQ.KMAX)GO TO 56
        NSPAC=NPAC
        TTRU=TF
56      CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
        IF(T(LBUFF).EQ.999)GO TO 250
        NGPAC=NGPAC+1
        TFIN=T(LBUFF)
        STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
250     CALL SCHEDU(NF,TDEL)
        NS=1
        DO 1410 J=2,LBUFF
          IF(T(J).GT.TS(ND(J)))GO TO 1410
          IF(TS(ND(J)).LT.TS(ND(NS)))NS=J
1410    CONTINUE
        IF(TS(1).GE.TMAX)KSIM=KMAX
        IF(T(1).EQ.999)GO TO 10
        IF(NS.EQ.1)GO TO 1320
        TTEMP=T(NS)
        NTEMP=ND(NS)
1430    T(NS)=T(NS-1)
        ND(NS)=ND(NS-1)
        NS=NS-1
        IF(NS.EQ.1)GO TO 1420
        GO TO 1430
1420    T(1)=TTEMP
        ND(1)=NTEMP
        GO TO 1330
10      CONTINUE
```

```
C
C       *** START THE SIMULATION. STOP AT TMAX ***
C
        NF=1
        LDIRJ=1
1320    CALL SCHEDU(LDIRJ,TBIT,TDEL)
        IF(T(1).LT.TS(LFIN))GO TO 1330
1310    X=T(1)-TS(LINI)
        LX=INT(X/DPER)
1350    DO 1300 K=1,NDEVIC
          TS(K)=TS(K)+LX*DPER
1300    CONTINUE
1330    IF(LDIRJ.EQ.0)GO TO 5100
          LINI=1
          LFIN=NDEVIC
          LSTE=1
          GO TO 5125
5100      LINI=NDEVIC
          LFIN=1
          LSTE=-1
5125      DO 5110 K=LINI,LFIN,LSTE
            DO 5120 J=1,LBUFF
              IF(ND(J).NE.K)GO TO 5120
              IF(T(J).GE.TS(K))GO TO 5110
                DELAY(K)=TS(K)-T(J)
                SQUARE(K)=SQUARE(K)+DELAY(K)*DELAY(K)
                TF=TS(K)+TPACK
                NF=K
                STIME(K)=STIME(K)+TS(K)
                NPAC=NPAC+1
                NPACKT(NF)=NPACKT(NF)+1
                SUMDEL=SUMDEL+DELAY(K)
                X1=TS(NF)-DELAY(K)
                IF(NPAC.GT.KL*500)GO TO 57
                IF(L1.EQ.1)GO TO 59
                GO TO 58
57              IF(NPAC.GT.KL*500)KL=KL+1
58              TYPE 55,NF,TS(NF),DELAY(K),NPACKT(NF),X1
55              FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC ',
     N          I4,F12.8)
59              IF(LDIRJ.EQ.0)GO TO 5210
                LFN1=NDEVIC
                LST1=1
                GO TO 5220
5210            LFN1=1
                LST1=-1
5220            DO 5200 L=K,LFN1,LST1
                  TS(L)=TS(L)+TPACK
5200            CONTINUE
                CALL DELETE(LBUFF,J)
                IF(KSIM.EQ.KMAX)GO TO 56
                NSPAC=NPAC
                TTRU=TF
56              CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
                IF(T(LBUFF).EQ.999)GO TO 250
                NGPAC=NGPAC+1
                TFIN=T(LBUFF)
                STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
250             IF(TS(1).GE.TMAX)KSIM=KMAX
                IF(T(1).EQ.999)GO TO 10
                GO TO 5110
5120        CONTINUE
5110      CONTINUE
          IF(LDIRJ.EQ.0)GO TO 5130
          LDIRJ=0
          GO TO 1320
5130      LDIRJ=1
          GO TO 1320
10        CONTINUE
```

PROTOCOL F

```
C
      SUBROUTINE SCHEDU(LAST,TDEL)
C
      COMMON NDEVIC,NGR,NF,VF,TF,TSLOT,TSLOT1,LGR(100),LPO(100)
      COMMON T(100),ND(100),SWI,TS(100)
      DO 2100 K=1,NDEVIC
        IF(K.LE.LAST)GO TO 2110
          TS(K)=TF+(K-LAST)*VF+TDEL*(K-LAST)
          GO TO 2100
2110      TS(K)=TF+(NDEVIC-LAST+K-1)*(VF+TDEL)+(NDEVIC-1)*(VF+TDEL)
2100  CONTINUE
      RETURN
      END
```

**** PROTOCOL C SCHEDULING ROUTINE ****

```
C
      SUBROUTINE SCHEDU(LDIRJ,TBIT,TDEL)
C
      COMMON NDEVIC,NGR,NF,VF,TF,TSLOT,TSLOT1,LGR(100),LPO(100)
      COMMON T(100),ND(100),SWI,TS(100)
      IF(LDIRJ.EQ.0)GO TO 2150
      DO 2100 J=1,NDEVIC
        TS(J)=TS(1)+(J-1)*(VF+TBIT+TDEL)
2100  CONTINUE
      RETURN
2150  DO 2160 J=NDEVIC,1,-1
        TS(J)=TS(NDEVIC)+(NDEVIC-J)*(VF+TBIT+TDEL)
2160  CONTINUE
      RETURN
      END
```

**** PROTOCOL F SCHEDULING ROUTINE ****

```
C
C       *** START THE SIMULATION. STOP AT TMAX ***
C
17      IF(T(1).GE.TMAX)KSIM=KMAX
16      IF(T(1).EQ.999)GO TO 10
C       *** CALCULATE FREE TIME FOR NEXT PACKET ***
        TF=T(1)+ABS(ND(2)-ND(1))*VF+TPACK
C       *** IF SUCCESS PRINT STATISTICS OF PACKET ***
        IF(TF.LT.T(2)) GO TO 50
C       *** IF DEFERENCES OCCUR MANAGE THEM ***
        IF(T(2)-T(1).GT.ABS(ND(2)-ND(1))*VF) GO TO 60
C       *** ELSE A COLLISION HAS TAKEN PLACE ***
        GO TO 70


C
C       *** STATISTICS ROUTINE FOR PACKET ***
C
 50     K=ND(1)
        X=T(1)
        STIME(K)=STIME(K)+X
        SQUARE(K)=SQUARE(K)+DELAY(1)*DELAY(1)
C       *** UPDATE COUNTERS ***
        KSEG=INT(DELAY(1)/TDIST)+1
        IF(KSEG.GT.NINTER)KSEG=NINTER
        NDEL(KSEG)=NDEL(KSEG)+1
        NPAC=NPAC+1
        NPACKT(K)=NPACKT(K)+1
        SUMDEL=SUMDEL+DELAY(1)
        X1=X-DELAY(1)
53      IF(NPAC.GT.KL*500)GO TO 57
        IF(L1.EQ.1)GO TO 59
        GO TO 58
57      TYPE 571,KL,INT(IN7*1./500)
571     FORMAT(' PROGRAM IN PROGRESS.  SEGMENT # ',I4,' OF ',I4)
        KL=KL+1
        GO TO 59
58      TYPE 55,K,T(1),DELAY(1),NPACKT(K),X1
55      FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC',
     N  ' NP=',I4,' OR.',F12.8)
59      DELAY(1)=0
C       *** DELETE PACKET FROM EVENT LIST ***
        CALL DELETE(LBUFF,1)
        IF(KSIM.EQ.KMAX)GO TO 56
        NSPAC=NPAC
        TTRU=X+TPACK
C       *** GENERATE A NEW EVENT ***
        CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
        IF(T(LBUFF).EQ.999)GO TO 250
        NGPAC=NGPAC+1
        TFIN=T(LBUFF)
        STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
        GO TO 250
56      T(LBUFF)=999
C
```

```
C
C       *** DEFERENCE ROUTINE ***
C
 60     DO 610 J=2,LBUFF
          TF=T(1)+ABS(ND(J)-ND(1))*VF+TPACK+TDEL
          IF(T(J).GT.TF)GO TO  50
C         *** UPDATE COUNTERS ***
          DELAY(J)=DELAY(J)+TF-T(J)
          T(J)=TF
 610    CONTINUE
        GO TO 250
C
C       *** COLLISION ENFORCEMENT ROUTINE ***
C
 70     DO 710 K=1,LBUFF
          DO 720 J=1,LBUFF
            IF(J.EQ.K)GOTO 720
            IF(ABS(T(J)-T(I)).GT.ABS(ND(J)-ND(1))*VF)GOTO 710
            TFIT=T(K)+ABS(ND(J)-ND(K))*VF
            IF(TC(J).EQ.0)TC(J)=TFIT
            IF(TFIT.LT.TC(J))TC(J)=TFIT
 720        CONTINUE
 710    CONTINUE
        IF(L1.EQ.1)GOTO 751
        DO 752 K=1,LBUFF
        IF(TC(K).EQ.0)GOTO 751
        TYPE 753,K,ND(K),TC(K)
 753    FORMAT(I4,I4,F12.8)
 752    CONTINUE
 751    DO 750 K=1,LBUFF
          IF(TC(K).EQ.0)GOTO 740
          TXND=TC(K)+ABS(ND(K)-1)*VF
          IF(TXND.GT.TEND)TEND=TXND
        IF(L1.EQ.1)GOTO 750
 750    CONTINUE
C
C       *** START PROTOCOL C ***
C
 740    CALL SCHEDU(TEND,TBIT,TDEL,NDEVIC,VF)
        LCOUNT=0
        DO 7410 K=1,NDEVIC
          DO 7420 J=1,LBUFF
            IF(ND(J).NE.K)GO TO 7420
            IF(T(J).GT.TS(K))GO TO 7410
                DELY=TS(K)-T(J)+DELAY(J)
                SQUARE(K)=SQUARE(K)+DELY*DELY
                TF=TS(K)+TPACK
                KSEG=INT(DELY/TDIST)+1
                IF(KSEG.GT.NINTER)KSEG=NINTER
                NDEL(KSEG)=NDEL(KSEG)+1
                LCOUNT=LCOUNT+1
```

```
                STIME(K)=STIME(K)+TS(K)
                NPAC=NPAC+1
                NPACKT(K)=NPACKT(K)+1
                SUMDEL=SUMDEL+DELY
                X1=TS(K)-DELY
                IF(NPAC.GT.KL*500)GO TO 7430
                IF(L1.EQ.1)GO TO 7432
                GO TO 7431
7430            TYPE 7435,KL,INT(IN7*1./500)
7435            FORMAT(' PROGRAM IN PROGRESS. SEGMENT# ',I4,' OF ',I4)
                KL=KL+1
                GO TO 7432
7431            TYPE 7434,K,TS(K),DELY,NPACKT(K),X1
7434            FORMAT(' DEV.',I3,' OUT AT ',F12.8,' AFTER ',F10.8,' SEC',
               ' NP=',I4,' OR.',F12.8)
7432            DO 7440 L=K,NDEVIC
                  TS(L)=TS(L)+TPACK
7440            CONTINUE
                CALL DELETE(LBUFF,J)
                IF(KSIM.EQ.KMAX)GO TO 7450
                NSPAC=NPAC
                TTRU=TF
7450            CALL GENERA(LBUFF,NDEVIC,T(LBUFF-1),L1,TMAX,XLAMDA,M5,M6)
                IF(T(LBUFF).EQ.999)GO TO 7460
                NGPAC=NGPAC+1
                TFIN=T(LBUFF)
                STIMGE(ND(LBUFF))=STIMGE(ND(LBUFF))+T(LBUFF)
7460            IF(TS(1).GE.TMAX)KSIM=KMAX
                IF(T(1).EQ.999)GO TO 10
                GO TO 7410
7420          CONTINUE
7410        CONTINUE
            IF(LCOUNT.NE.0)GO TO 755
            DO 7500 K=1,LBUFF
              TC(K)=0
7500        CONTINUE
            TEND=0
            GO TO 17
755         TEND=TS(NDEVIC)+(NDEVIC-1)*VF+TBIT
            GO TO 740
            END
C
            SUBROUTINE SCHEDU(TEND,TBIT,TDEL,NDEVIC,VF)
C
            COMMON T(100),TS(100)
            DO 2100 J=1,NDEVIC
              TS(J)=TEND+TBIT+TDEL+(J-1)*(VF+TBIT+TDEL)
2100        CONTINUE
            RETURN
            END
```

REFERENCES

1. Wood D.C., Holmgren S.F., Skelton A.P., "A Cable-bus Protocol Architecture",CH1405-0/79/0000-0137,1979,pp 137.

2. Coviello G.J., Lake O.L., Redinho G.R., "System Design Implications of Packetized Voice", ICC-77, pp 38.3.49

3. Melvin D.K., "Voice on Ethernet...Now!", NTC-81.

4. Tobagi F.A., "Multiaccess Protocols in Packet Communications Systems", IEEE Trans. on Comm., Vol Com-28, No 4, Apr 1980, pp 468.

5. Nutt G.J., Bayer D.L., "Performance of CSMA/CD Networks Under Combined Voice and Data Loads", IEEE Trans. on Comm., Vol Com-30, No 1, Jan 1982, pp 6.

6. Kiesel W.M., Kuehn P.J., "CSMA-CD-DR: a New Multi-access Protocol for Distributed Systems", IEEE CH 1679-0/81/0000-0010,1981,pp A2.4.1.

7. Gruber J., "Performance Considerations for Integrated Voice and Data Networks", Computer Communications, Vol 4, No 3, Jun 1981, ppl06.

8. Lam S.S., "A Carrier Sense Multiple Access Protocol for Local Area Networks", Computer Networks, No 4, Apr 1980, pp 21.

9. Crane R.C., Taft E.A., "Practical Considerations in Ethernet Local Network Design", Proc. 13 th Hawaii Int'l Conf. Systems Sciences, Jan 1980, pp 166.

10. Johnson D.H., O'Leary G.C., "A Local Access Network for Packetized Digital Voice Communications", IEEE Trans. on Comm., Vol Com-29, No 5, May 1981, pp 679.

11. Shoch J.F., Hupp J.A., "Measured Performance of an Ethernet Local Network", Comm. of the ACM, Vol 23, No 12, Dec 1980,
pp 711.

12. Shoch J.F., "Carrying Voice Traffic Through an Ethernet Local Network..a General Overview", Inter. Workshop on LACN, Aug 1980.

13. Baxter L.A., Baugh C.R., "A Comparison of Architectural Alternatives for Local Voice/Data Communications", IEEE Comm. Magazine, Jan 1982, pp 44.

14. Li S., "Performance Analysis of a Conflict-free Multiple Access LAN for Voice and Data", M Sc. Thesis U. of Waterloo, 1981

15. Liu T.T., Li L., "A Virtual Token Scheme, GBRAM, for Local Data and Voice Distribution", GTE Labs., Waltham, MA 02254.

16. Musser J.M., "Packet-voice Performance on a CSMA/CD LAN", ISSLS, Sep 1982.

17. Kleinrock L., "Queueing Systems Vol II", John Wiley, 1976.

18. Chlamtac I., Franta W.R., Levin K.D., "BRAM: The Broadcast Recognizing Access Method", IEEE Trans. on Comm., Vol Com-27, No 8, Aug 1979, pp 1183.

19. Brady P.T., "A Statistical Analysis of On-Off Patterns in 16 Conversations", BSTJ, Vol 47, 1968, pp 73.

20. Kleinrock L., Tobagi F.A., "Packet Switching in Radio Channels: Part I, CSMA Modes and their Throughput-Delay Characteristics", IEEE Trans. on Comm., Vol Com-23, No 12, Dec 1976, pp 1400.

21. Capetanakis J.I., "Generalized TDMA: The Multiaccessing Tree Protocol", IEEE Trans. on Comm., Vol Com-27, No 10, Oct 1979, pp 1476.

22. Lam S.S., Kleinrock L., "Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation", IEEE Trans. on Comm., Vol Com-23, No 4, Apr 1975, pp 410.

23. Beltrao-Moura J.A., "Loops and Ethernet: Evaluation and Comparison of Performance and Complexity", CCNG T-Report T-80, U. of Waterloo, Apr 1979.

24. Bux W., "Local Area Subnetworks: a Performance Comparison", Local Networks for Computer Communications, North-Holland Pub., 1981, pp 157.

25. Shoch J.F. et al, "Evolution of the Ethernet Local Computer Network", IEEE Computer, Aug 1982, pp 10.

26. Li L., Hughes H.D., Greenberg L.H., "Performance Analysis of a Shortest-Delay Protocol", Computer Networks, Jul 1982, pp 189.

27. Liu T.T., Li L., Franta W.R., "The Analysis of a Conflict-free Protocol Based on Node Clusters", IEEE Computer Networking Symposium 1981, pp 61.

28. Li L., Hughes H.D., "Definition and Analysis of a New Shortest-delay Protocol", IEEE Computer Networking Symposium 1981, pp 21.

29. Gold Y.I., Franta W.R., "An Efficient Scheduling Function for Distributed Multiplexing of a Communication Bus Shared by a Large Number of Users", ICC-82, pp 5.C.5.1.

30. Tobagi F.A., Hunt V.B., "Performance Analysis of CSMA/CD", Computer Networks, Vol 4, pp 245, 1980.

31. Martin J., "System Analysis for Data Transmission", Prentice-Hall, 1972.

32. Guenther W.C., "Analysis of Variance", Prentice-Hall, 1964.

33. Tanenbaum A.S., "Computer Networks", Prentice-Hall, 1981.

34. Kleinrock L., Scholl M., "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes for a Small Number of Data Users", ICC Conf. Proc., Chicago IL, June 1977, pp 22.1-105.

35. Rios M., Georganas N.D., "A New Hybrid Multiple-Access Protocol for Data and Packet-Voice over Local Area Networks", Third International Congress on Computer Science, Santiago, Chile, June 1983.