

8

An Investigation of Selected

Local Area Network

Access Control Protocols

by

Alan Jacobsen

Submitted in Partial Fulfillment

of the Requirements for the

Degree of

Master of Engineering

in the

Department of Electrical Engineering

at

McGill University

© April, 1984

Abstract

The present industry effort in defining standards for local area networks is discussed, especially the access control protocols used to share the local communications channel. Either selection or contention access control protocols are typically used in local area networks, Ethernet (CSMA-CD) being a well-known example of a contention scheme, a token ring an example of a selection scheme. Results of a simulation study of the access control protocols of Ethernet and of a new protocol combining aspects of both contention and selection techniques, decentralized adaptive probing, are shown. Simulation allows these protocols to be modelled in detail, providing meaningful estimates of the performance indicators of network throughput, mean waiting time, mean queue size, mean rejection rate, mean failure rate, and mean number of attempts required for successful transmission, all as functions of the traffic intensity. The relative advantages and disadvantages of contention and selection access control techniques are thus elucidated. Finally, an analysis based on queueing theory treating CSMA-CD with a finite buffer capacity is presented, illustrating the analytical complexity required to address even simplified protocol models.

Résumé

Les efforts actuels de l'industrie pour standardiser les réseaux locaux sont discutés, eu égard aux techniques employées pour contrôler l'accès aux canaux de communications partagés. Les techniques d'accès par sélection (comme << token rings >>) ou par contention (comme << Ethernet (CSMA-CD) >>) sont typiquement employées dans les réseaux locaux. Les résultats d'une étude par simulation d'Ethernet et d'une nouvelle technique (<< decentralized adaptive probing >>) incorporant des caractéristiques de techniques de contention et de sélection dans un façon adaptatif sont exhibités. La simulation permet la modélisation de ces algorithmes de contrôle en détail et rend compte les indicateurs de performance suivants comme une fonction de la densité du trafic: le taux moyen d'utilisation du réseau, le temps moyen d'attente, la longueur moyenne de la file d'attente, le taux moyen de rejet, le taux moyen d'échec, et le nombre moyen des essais requis pour une transmission réussie. Dans cette manière les différences entre les deux techniques d'accès sont elucidées. Pour terminer, une analyse fondée sur la théorie des files d'attente est présentée au sujet de CSMA-CD avec des stations de capacité limitée. La technique d'accès est simplifiée pour faire l'analyse, mais, malgré cela, l'analyse devient compliquée, montrant les difficultés en analysant une technique d'accès.

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. Local Area Networks.....	1
1.1 Introduction.....	2
1.2 The Open Systems Interconnection Reference Model.....	8
1.3 The IEEE 802 Committee Proposed Standards.....	14
2. A Simulation Study of Ethernet and Decentralized Adaptive Probing.....	16
2.1 Introduction.....	17
2.2 Simulation Modelling.....	19
2.2.1 Ethernet.....	20
2.2.2 Decentralized Adaptive Probing.....	24
2.3 Simulation Techniques.....	49
2.3.1 Regenerative Simulation.....	50
2.3.2 Stopping Rules.....	55
2.3.3 Ratio Estimation — The Jackknife Estimator.....	58
2.4 A Comparison of Adaptive Probing and CSMA's Contention Resolution Efficiencies Results.....	67
2.5 Simulation Results.....	78
2.5.1 Network Throughput.....	80
2.5.2 Mean Waiting Time.....	83
2.5.3 Mean Queue Size.....	97
2.5.4 Mean Rejection Rate.....	98
2.5.5 Mean Failure Rate.....	107
2.5.6 Mean Number of Attempts Required.....	114
2.5.7 Mean Probe Level.....	115
3. An Analysis of CSMA-CD With a Finite Number of Buffered Users.....	122
3.1 Introduction.....	123
3.2 The Analytical Model.....	125
3.3 Analysis.....	128
3.3.1 The Channel Markov Chain.....	129
3.3.2 The User Markov Chain.....	141

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page</u>
3.3.3 Stability Considerations and User Markov Chain Verification.....	147
3.3.4 Coupling of the Markov Chains.....	149
3.3.5 Delay Analysis.....	156
4. Conclusion.....	163
5. References.....	166
Appendix	172

LIST OF FIGURES

<u>Figures</u>	<u>Page</u>
1.1	Network Layering (adapted from [5]).....12
1.2	The OSI Reference Model Architecture.....13
2.2.1	Timing Considerations in Decentralized Adaptive Probing.....31
2.2.2	The Probing Algorithm.....33
2.2.3	Full-Scale System Configuration.....39
2.2.4	Level-Dependent Maximum Propagation Delays for the Single-Cable System.....47
2.2.5	Level-Dependent Maximum Propagation Delays for the Full-Scale System.....48
2.3.1	Simulation Performance As a Function of Load.....54
2.4.1	Mean Time to Clear versus the Probability a Station Has a Message.....75
2.4.2	Mean Time to Clear versus the Probability a Station has a Message.....76
2.4.3	Mean Time to Clear versus the Probability a Station Has a Message.....77
2.5.1a-d	Network Throughput versus Load ρ and Arrival Rate λ 83-86
2.5.2a-h	Mean Waiting Time versus Load ρ and Arrival Rate λ 89-96
2.5.3a-h	Mean Queue Size versus Load ρ 98-105
2.5.4a-d	Mean Rejection Rate versus Load ρ and Arrival Rate λ ..107-110
2.5.5a-b	Mean Failure Rate versus Load ρ and Arrival Rate λ ...112-113
2.5.6a-d	Mean Number of Attempts Required versus Load ρ and Arrival Rate λ 115-118
2.5.7a-b	Mean Probe Level versus Load ρ and Arrival Rate λ 120-121
3.1	User Terminal Model.....127
3.2	Imbedded Slots in the Channel Markov Chain.....139
3.3	State Transition Diagram for the Channel Markov Chain.....140
3.4	State Transition Diagram for the User Markov Chain.....146

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1.1	User Concerns.....	7
2.2.1	Simulation System Parameters.....	23
2.2.2	256-Station Network.....	40
2.2.3	100-Station Network.....	45
2.3.1	Simulation Techniques Used in This Study.....	57
2.3.2	Performance Indicators (Classical Estimators).....	65
2.3.3	Ratio Estimation with the Jackknife Estimator.....	66

1. Local Area Networks

1.1 Introduction

With the advent of widespread computing power, the desire and need to interconnect devices in a local environment has grown. Local area communication networks have been and are being developed to satisfy that requirement, but, considering the number of potential users, it is still a nascent industry. The term "local area network" today suggests a high data rate system extending throughout an office building (or buildings) providing packet-switched communications to a number of users. The intent of a local area network is seen as information transfer rather than a more industrial function such as process control. The market for such a capability is large, but its growth depends on the timely and effective satisfaction of potential network users' wants and needs. The technical problems concerning the efficient shared use of a common communications channel in a local environment have received attention over the last decade; the pragmatic problems of developing the market for local area networks are just beginning to be addressed.

The ability to interconnect devices depends not just on the existence of a physical link, but also on the compatibility of the devices and their capability to meaningfully communicate. The International Organization for Standardization has attempted to address this issue by identifying the levels of communication required between information processing devices. The concern is not just for local area networks, but for all networks. The resultant Open Systems Interconnection (OSI) Reference Model, discussed in Section 1.2 of this study, proposes a layered architecture for each system to thereby define the areas of common responsibility over the inter-communication. At each layer, the communicating systems must possess a common set of procedures and functions; i.e., a common protocol, if the systems are to "understand" one another.

If communications between disparate information-handling systems are to become commonplace, the development of standard protocols observed by the manufacturers of such equipment is a necessity. While the OSI Reference Model has not been endorsed as an industry standard, some protocols in keeping with the Reference Model have been made standard. The efforts of

the Institute of Electrical and Electronic (IEEE) 802 Local Network Standards Committee, discussed in Section 1.3, have resulted in the proposal of standard protocols for the lowest two of the Reference Model's seven layers. Protocols corresponding to other layers of the Reference Model have also been proposed or adopted as standards, but it is the work of the IEEE 802 committee which has most excited the market for local area networks.

The 802 committee, after much work in collaboration with representatives from industry, has proposed or adopted standards concerning the transmission media used to support local area networks and the techniques by which access to such media is controlled. The acceptance of such standards allows the market for local area networks to finally develop beyond the embryonic stages of the last ten years. Even lacking the standardization of higher layer protocols, the existence of standard media access control protocols and the accompanying proliferation of vendors permits users to build local area networks without the fear of future incompatibility or obsolescence. Insuring comprehension and compatibility between the interconnected devices remains the responsibility of the user, but at least the risks involved in investing in a local area network are reduced.

From the users' point of view, the IEEE 802 committee has sorted out some of the confusion and uncertainty surrounding local area networks. The 802 committee, however, has not proposed just one standard, but has acted to reduce the number of possible choices. The issues of performance and cost are most important to the potential investor in a local area network, as summarized in Table 1.1, but the solution to these concerns depends on a user's specific needs and wants and on the technology supplied by the vendors of local area networks — the 802 standards in themselves only attempt to narrow the options. Depending on a user's requirements, the local area network may use twisted wire pair or coaxial or fibre-optic cable, may be a baseband or broadband system, may be configured in a variety of topologies, and may use any of a number of schemes to control access to the communications channel(s). These choices may not be made completely independent of one another, though, since one particular selection may narrow the remaining options.

The choice of transmission medium, however, is perhaps a decision which does not influence the other options. In local area networks, data rates are typically 1-10 Mbits/s. This range may be supported by all three of the communications media discussed here, but there are relative advantages and disadvantages. Coaxial cable is a reliable transmission medium, but considered to be costly and cumbersome to install. Fibre optic cable has the greatest bandwidth of these media, capable of supporting even higher data rates, but is sensitive to physical abuse and may impose strict limitations to the number of stations that can be supported by the network. Finally, use of twisted pair as the local area network medium is often attractive since it may already be installed in an existing facility, but it may not provide reliable transmission at high data rates.

Broadband networks offer wide bandwidths, allowing the transmission of graphics and even video, and may support more than one channel on the medium. Baseband networks, on the other hand, admit only one transmission channel but do not require the modulation techniques needed in a broadband system. Local area networks offered by Wang Laboratories, Inc., System, Inc., and Ungermann-Bass, Inc., for example, are broadband systems while Xerox Corporation's Ethernet is a baseband network. The IEEE 802 Ethernet committee has adopted the access control protocol used in Ethernet as one of its proposed standards, but has specified both baseband and broadband versions.

The selection of network topology may be intimately related to the choice of access control mechanism; indeed, the operation of some access protocols relies upon specific network organizations. The 802 committee treats a local network as being based on either broadcast or sequential media. On a broadcast medium, a transmission by one station is received by all stations of the network; on a sequential medium, data packets are transmitted from one station to another about the network. One may thus consider broadcast and sequential media to define bus and ring topologies, respectively. The actual topology as defined by the organization of the cable network is at the discretion of the user; further classifications beyond "bus" and "ring" are of course possible, but, for the purposes of the 802 committee and this discussion, the simple distinction between broadcast and

sequential media suffices. Note, however, that a star topology, for example, is not excluded by this scheme — one may conceive of either broadcast or sequential systems configured as a star network. In networks supported by bus media, the failure of one or more stations does not affect network operation, assuming decentralized systems. In networks supported by sequential media, however, the issue of network reliability becomes significant and must be specifically addressed.

The problem of regulating access to a shared channel extends beyond just the field of local area networks; satellite channels, for example, require the same sort of control. Luczak [46] has classified the access control techniques advanced in the research literature for shared broadcast channels, distinguishing between reservation, selection, and contention protocols. An example of a reservation scheme would be TDMA, with possible variations including the static or dynamic, centralized or distributed allocation and reservation of channel capacity. Selection protocols, rather than reserving capacity, effectively choose which users may access the channel at a given time, examples being token passing and polling. Again, a selection protocol may be either centralized or decentralized. In contention access control protocols, users of the shared channel transmit randomly, resolving contention through the protocol's specific collision resolution technique. Ethernet, for example, employs a contention access protocol, a form of CSMA-CD (Carrier Sense Multiple Access with Collision Detection).

The mechanisms used in local area networks to control channel access are generally selection or contention protocols — reservation schemes are not an efficient use of capacity given the bursty nature of typical local network users. Other aspects specific to local networks; for example, the relatively short propagation times through the network and the possibility of sequential rings, have naturally led to the development of access control protocols that are uniquely suited to local area networks. The following local area networks are representative of those being currently considered, as evidenced by a recent IEEE Special Issue on Local Area Networks [47]: the token ring, slotted ring, buffer insertion ring, CSMA-CD, and the token passing bus. The IEEE 802 committee has further narrowed its proposed standards to the token ring, CSMA-CD, and the token passing bus.

This study, then, attempts to reveal aspects of local area network performance in the context of the above discussion. The effort focusses on access control on broadcast media, evaluating performance in terms of the access control protocol's effectiveness in sharing channel capacity with minimal user impact. Section 2 approaches the problem through simulation, modelling two particular access control protocols in detail and assessing a variety of performance indicators. Section 3, on the other hand, applies the techniques of queueing theory to a simplified model of an access control protocol, gauging performance by the network throughput and the mean message delay.

The simulations presented in Section 2 are of Ethernet, as specified by the 802 committee, and of decentralized adaptive probing, a decentralized version of an adaptive protocol incorporating aspects of both contention and selection access control techniques. Simulation permits the detailed modelling of the access protocols and allows the observation and collection of any number of performance indicators. Comparison of the simulation results for the pure contention scheme of Ethernet and the adaptive hybrid of both contention and selection techniques illustrates the limitations of a pure contention access technique. The difference in performance, however, proves not to be as great as one might expect.

Section 3 presents an analysis of CSMA-CD where the stations have a finite buffer capacity. The mathematical analysis of access control protocols may permit parametric studies and/or comparison against similar analyses, hence their utility. Often, however, the analytical model of the access control algorithm is necessarily simplified in order to effect the analysis. Extrapolation to actual performance of the access control protocol is thus not always possible. Here, the model of CSMA-CD is one considered elsewhere in the research literature, but the introduction of finite buffer size leads to a set of coupled non-linear equations that do not readily admit numerical solution. Nonetheless, the analysis demonstrates the techniques available to solve such problems.

Finally, Section 4 summarizes this work; Section 5 lists the references used; and the Appendix provides a sample of the simulation output and listings of the simulation programs.

- **Cost**
- **Compatibility between connected equipment**
 - Higher level protocols
 - Efficient use of available resources
- **Network loading/network capacity providing acceptable performance, especially response times**
- **Network reliability and management**
 - Performance monitoring and fault isolation
 - Error recovery
- **Network connectivity**
 - Installation of the network
 - Efficient distribution of network access points
 - Capacity for network expansion
 - Gateway capability to systems outside the local network

Table 1.1 User Concerns

1.2 The Open Systems Interconnection Reference Model

Cognizant of the need for international standardization among manufacturers, the International Organization for Standardization (ISO) began in 1977 to define a framework for the development of standard protocols. The result, the International Standard Reference Model of Open Systems Interconnection [4], establishes a layered architecture concept to model the interconnection of systems. The Reference Model defines a system as "a set of one or more computers, associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer." A system is open in the sense that conformance with international standards permits interconnection with all other systems also in compliance. The Open Systems Interconnection (OSI) architecture thus serves as a model of how conforming systems should appear externally to other open systems; the internal organization and functioning of individual open systems is not within the scope of the document. Indeed, the standard emphasizes that it is not meant to precisely define the services and protocols required, but merely provides an international basis for the development of such definitions.

The OSI architecture models a network of interconnected systems as a hierarchy of layers of procedural and functional responsibility. As suggested by Figure 1.1, the layers span across individual systems, working outwards from the lowest to the highest layer, building upon each other. Each layer augments the services provided by the lower layers so that, ultimately, the highest layer possesses the capability to administer the desired distributed process.

The intersection of a layer and a system may be seen as a subsystem; a layer is then simply all subsystems of the same rank in the interconnected network. Cooperation within the same layer is governed by that layer's protocols, thereby establishing the manner in which the services of lower layers are used to provide the capabilities required by higher layers. This representation of the layered architecture is employed by the OSI to illustrate the seven-layer Reference Model, as shown in Figure 1.2. The layered

architecture allows individual protocols and service layers to be replaced or modified as needed without requiring changes in other parts of either the network or its individual components. The following principles, excerpted from References [4] and [5], were used as guidelines in the creation of the Reference Model:

- do not create so many layers as to make difficult the system engineering task describing and integrating these layers;
- create a boundary at a point where the services description can be small and the number of interactions across the boundary is minimized;
- create separate layers to handle functions which are manifestly different in the process performed or the technology involved;
- collect similar functions into the same layer;
- select boundaries at a point which past experience has demonstrated to be successful;
- create a layer of easily localized functions so that the layer could be totally redesigned and its protocols changed in a major way to take advantage of new advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers;
- create a boundary where it may be useful at some point in time to have the corresponding interface standardized;
- create a layer when there is a need for a different level of abstraction in the handling of data, e.g., morphology, syntax, semantics;
- enable changes of functions or protocols within a layer without affecting the other layers;
- create for each layer interfaces with its upper and lower layer only;
- create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communication services need it;

- create, where needed, two or more sublayers with a common, and therefore minimum, functionality to allow interface operation with adjacent layers; and,
- allow bypassing of sublayers.

The resulting Reference Model comprises seven layers: from highest to lowest, the Application, Presentation, Session, Transport, Network, Data Link, and Physical Layers.

As the highest layer of the OSI architecture, protocols of the Application Layer directly serve the end users in enabling their exchange of meaningful information. Application Layer protocols thus act to define the character of the interconnection between systems.

The Presentation Layer provides a common representation of information for the Application Layer, furnishing the required transformations between the syntaxes used by the communicating applications. Future work will be directed to develop standard Virtual Terminal and File Transfer protocols within this layer. Already, an ISO standard for "Extended Control Characters of I/O Imaging Devices" exists.

The Session Layer supports the Presentation Layer in establishing a dialogue and in organizing and synchronizing the resulting data exchange. Currently, Session Layer functions have been incorporated into existing Presentation Layer protocols, but a standard Session Layer protocol should be readily extractible.

The Transport Layer, in coordination with services provided by lower layers, provides a transparent and universal data transfer service, relieving all higher layers from any concern with the detailed mechanism by which the reliable and cost effective transfer of data is accomplished. The Transport Layer is thus concerned with end-to-end performance, independent of the underlying communications media in all respects other than the quality of service. Several proposed standard Transport Layer protocols are available, the most widely known being the IFIP-sponsored INWG 96-1.

The Network Layer provides a consistent network service, masking from higher layers the differences in the characteristics of disparate transmission and subnetwork technologies. The functions of routing and relaying are performed by the Network Layer. An enhanced version of level 3 of the CCITT X.25 interface potentially affords a standard Network Layer protocol.

The Data Link Layer enables the Network Layer to control the interconnection of the communication paths within the lowest layer, the Physical Layer, thus providing the functional and procedural means to establish, maintain, and release the link. Furthermore, the Data Link Layer detects and possibly corrects errors originating in the Physical Layer. Standard Data Link protocols have already been developed by the ISO, the most well-known being HDLC.

The Physical Layer, then, provides the mechanical, electrical, functional, and procedural means to activate, maintain, and deactivate the physical transmissions required for bit transmission on a variety of communications media. CCITT standards such as X.21, V.24, V.25, etc. currently exist defining the interfaces with various physical media and the protocols for establishing, controlling, and releasing switched data circuits.

The American National Standards Institute (ANSI), however, has recently decided not to endorse the OSI reference model as an ISO standard [29]. Acknowledging the model's utility in defining an abstract architecture and thereby facilitating discussion among equipment designers, the American delegates to the ISO objected to the possible ramifications of mandating the OSI model as a standard. In addition to the predictable vendor opposition to enforced conformance to the OSI model, the US delegation's reservations concerned the possibility of freezing the technology prematurely, of legislating a vague outline in force without adequate provision for continual updating. The impact of ANSI's decision on the OSI reference model remains to be seen, although ANSI has emphasized its support of the OSI architecture as a model rather than as a standard.

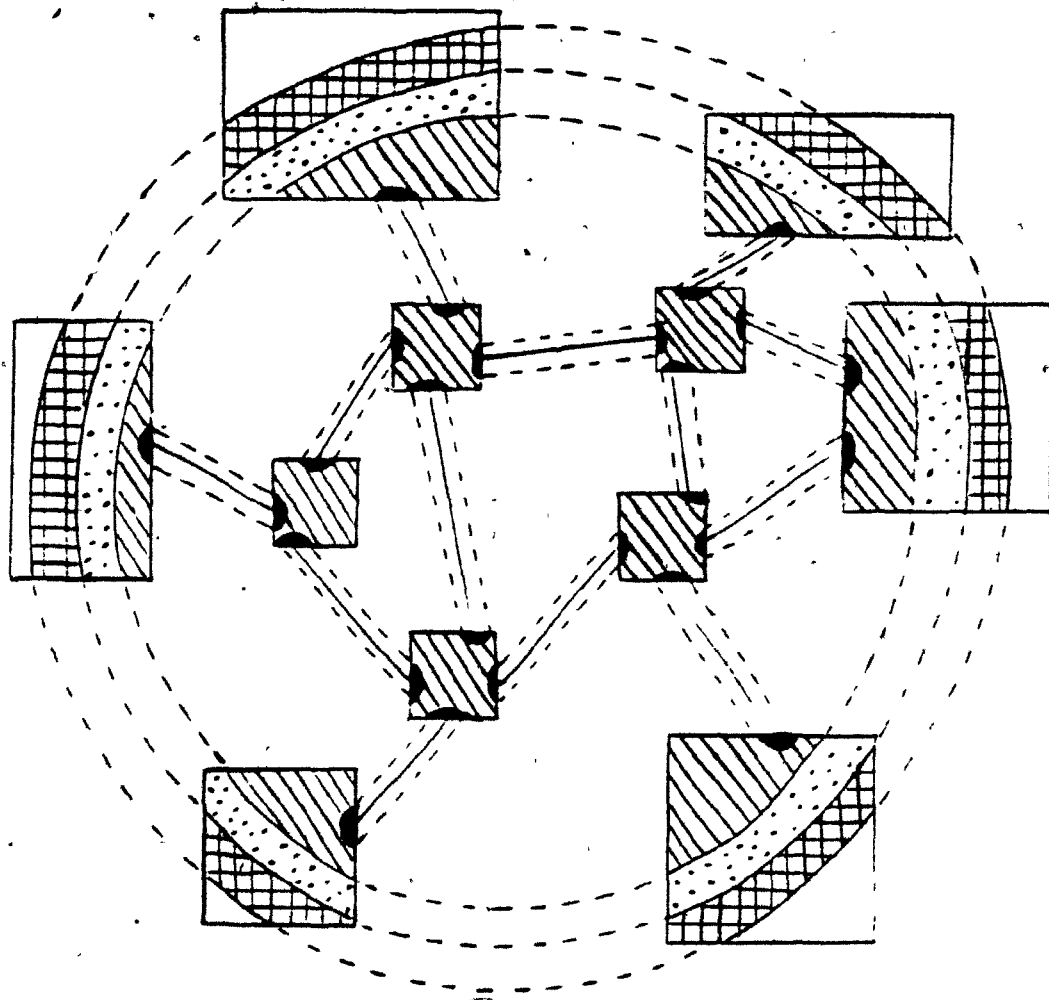


Figure 1.1 Network Layering (adapted from [5])

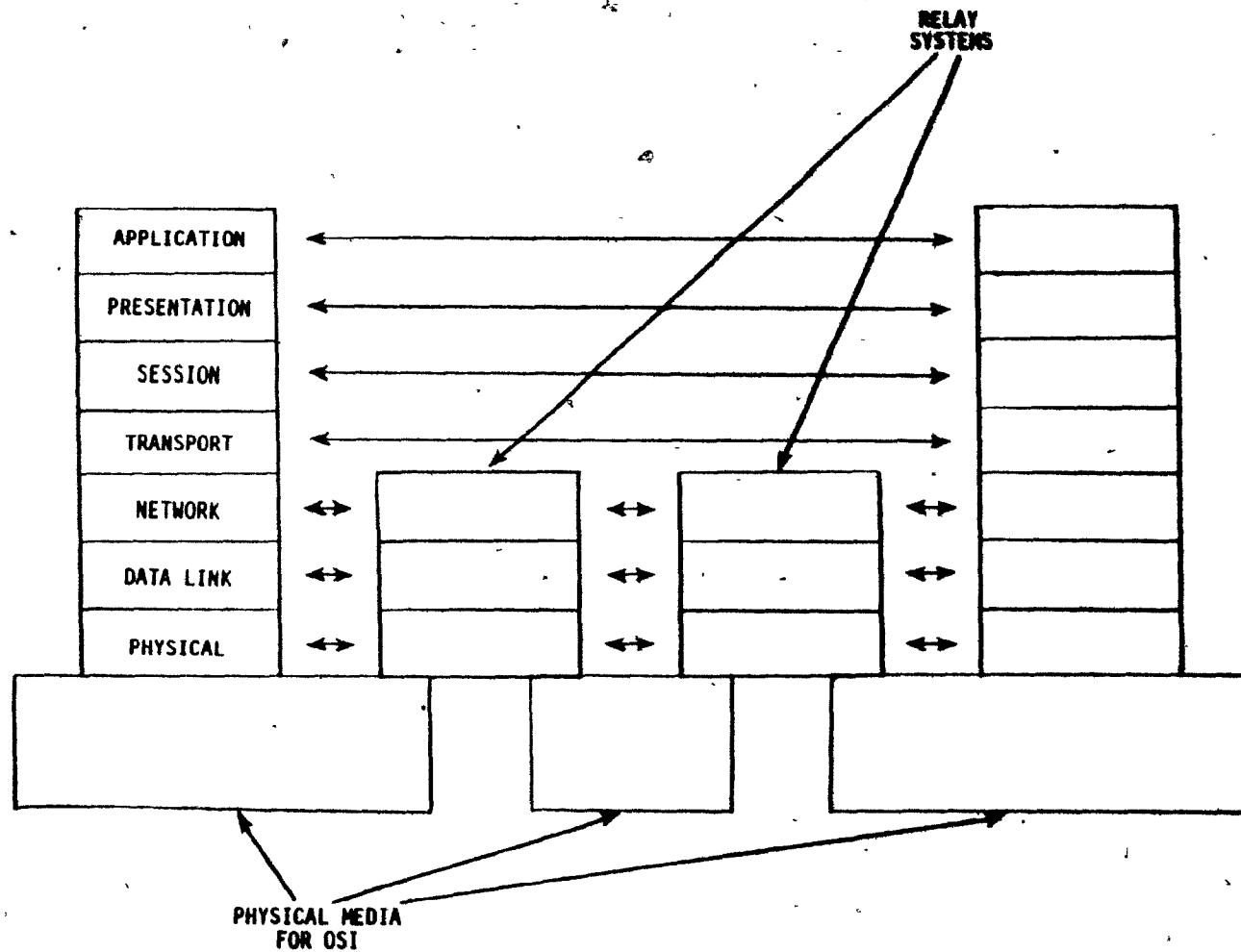


Figure 1.2 The OSI Reference Model Architecture

1.3 The IEEE 802 Committee Proposed Standards

Acting within the context of the ISO's reference model, the IEEE 802 committee issued in July, 1981 its first draft [9] of a proposed local network standard governing the two lowest layers of the OSI architecture. The proposed IEEE standards restrict themselves to local area networks where stations are on the order of one kilometer of each other and engaged in commercial or light industrial applications. Computer-controlled voice/data PBX networks, military local networks, and real-time, high reliability industrial applications such as process control are not specifically addressed.

A shared channel configuration is defined, with low cost and ease of incremental network expansion considered to be design goals. The IEEE standard subdivides the OSI Data Link Layer into two layers, the Logical Link Control and Media Access Control Layers. The Logical Link Control layer draws upon the Asynchronous Balanced Mode link control procedures specified by ISO 6256-1979 and ANSI X3.66-1979 to define a standardized frame format and error recovery procedure. The Media Access Control Layer defines the mechanisms by which the shared channel is successfully utilized by the stations composing the network. Since throughput and message delay characteristics are dependent upon the employed channel access protocol, the IEEE 802 committee has developed proposed standards for both token-passing and carrier sense multiple access with collision detection (CSMA-CD).

The IEEE specification also applies to the OSI Physical Layer and the physical medium itself (which is beyond the scope of the OSI model). Characteristics of a physical interface are thus defined which allow it to be independently implemented by different vendors and still prove compatible. Several media are currently envisioned to support local area networks: twisted pair, optical fiber, and baseband or broadband coaxial cable. The most recent revision, Draft D of the proposed standards, defines specific functional, electrical, and mechanical characteristics of a baseband coaxial cable system supporting CSMA-CD [10] and a coaxial cable system supporting a "token-passing single channel phase-continuous FSK bus" [11]; other media will be standardized at some future date.

Most of the attention received by the IEEE 802 committee's proposed standards has been directed towards the Media Access Control Layer and its definition of the procedures governing access to the media. The committee's selection of CSMA-CD as an access method was motivated by Xerox's early work in developing Ethernet [26]; the choice of token-passing as a second distinct access method was similarly inspired by industry's activities in that area. Indeed, IBM has contributed its research efforts in token ring topologies, system design, architecture, and protocols to the IEEE 802 committee [12] in order to maintain a voice in the development of local area network standards. The influence of manufacturers on the work of the 802 committee is sizeable: Computerworld [13] reports that when Draft D of the CSMA-CD standards appeared likely to deviate significantly from the DEC-Intel-Xerox Ethernet specification, "tremendous" pressure from the European Computer Manufacturer Association led to Draft D's present version, closely resembling the Ethernet specification. As a result, as of mid-December 1982, thirteen companies announced their endorsement of Draft D of IEEE Standard P802.3 (CSMA-CD): Bridge Communications, Inc., Data General Corp., Digital Equipment Corp., Fujitsu America, Inc., Hewlett-Packard Co., Intel Corp., Interlan, Inc., National Semiconductor Corp., Siemens AG, Tektronix, Inc., 3Com Corp., Ungermann-Bass, Inc., and Xerox Corp. Such corporate lobbying is of course essential in developing an industry-wide standard. Maris Graube, the chairman of the IEEE 802 committee, has endeavored to open the formulation of local area network standards to all concerned parties, with the acceptability of the standard determined by consensus among representative vendors [14].

2. A Simulation Study of Ethernet and Decentralized Adaptive Probing

2.1 Introduction

As part of this study, a software simulation of the IEEE 802 committee's CSMA-CD access control protocol has been developed and used to investigate performance aspects of a typical local area network. Such a simulation study has been performed elsewhere (Reference [44], for example), but in this work simulation of a new access control protocol, decentralized adaptive probing, has also been developed. By comparing the performance of the two protocols, insight into the workings of a local area network is obtained.

Ethernet's access control mechanism; i.e., CSMA-CD, is well known, but the adaptive probing access protocol has received attention only in the research literature. "Adaptive probing" is a term used by Hayes [16] to describe use of a binary tree search algorithm as a mechanism to control access to the communications channel. Hayes considers the protocol to be directed by a central controller, but here the algorithm is modelled as a decentralized system, each node of the network independently observing and effectively directing the access protocols.

The two leading area network access control protocols, Ethernet (CSMA-CD) and token passing, represent contention and selection access techniques, respectively. The conventional wisdom is that, when network traffic is light, a contention access protocol will allow the transmission of a data frame with less delay than a selection protocol because the contention protocol permits random access to the channel while the selection protocol requires some time to determine which users desire access. At high traffic levels, however, a contention access protocol will delay a data frame's successful transmission longer than a selection protocol as those users with messages to transmit contend for access. To efficiently handle both low and high traffic levels, the decentralized adaptive probing protocol is designed to behave as a contention access protocol under light traffic loads and progressively become more and more like a selection access protocol under the highest loads. One expects, then, comparable performance between the two protocols at the lowest loads and then a divergence of performance as traffic intensifies. The simulation results of Ethernet compared with those of decentralized adaptive probing help elucidate such intuitions.

A description of the two protocols may be found in Sections 2.2.1 and 2.2.2; the network configuration that was simulated is described in Section 2.2.3; simulation techniques are discussed in Section 2.3; a short analysis attempting to predict the relative performance of the two protocols is given in Section 2.4; and Section 2.5 discusses the simulation results.

2.2 Simulation Modelling

The simulations conducted here are of Ethernet and decentralized adaptive probing; the simulation programs, then, must accurately model these access control protocols and the network on which they operate. Section 2.2.1 briefly describes the algorithm of Ethernet, 1-persistent CSMA-CD; Section 2.2.2 develops in detail a decentralized version of the adaptive probing algorithm; and Section 2.2.3 defines the network topologies used for the simulation studies.

2.2.1 Ethernet

Ethernet is an implementation of the CSMA-CD access protocol which has been developed by Xerox and is now gaining widespread market acceptance. As discussed in Section 1.3, the IEEE 802 committee's specification of a proposed standard CSMA-CD access protocol draws heavily upon aspects of the DEC/Intel/Xerox system. The first draft of the IEEE proposed standards [9] was used to define the simulation model used in this study. Subsequent drafts have altered certain physical parameters, but the specified CSMA-CD algorithm has remained unchanged.

The algorithm, known in the research literature as 1-persistent CSMA-CD, is described by both DEC/Intel/Xerox and the IEEE 802 committee through a procedural model written in the form of Concurrent Pascal processes. While that exposition is the most unambiguous, a word description of the algorithm will suffice here. All stations supported by the network share a broadcast channel; i.e., a channel where all stations receive any signal present, and monitor the channel to recognize data frames destined to them. Stations with a frame ready to transmit sense the channel before beginning their transmission. If the channel is sensed idle, transmission begins; if sensed busy, transmission is deferred until the channel is again sensed idle. At that time, transmission is again attempted, after a fixed-duration interframe gap, regardless of whether the channel is then sensed idle or busy. It is this behavior upon deferring transmission which is called 1-persistent: after deferring its transmission, a station persists in its attempt, trying again with probability one.

Due to propagation delays, two or more stations may sense the channel idle and attempt concurrent transmissions, resulting in a collision on the broadcast channel. Such collisions are detected by the colliding stations, who then continue their transmission for a fixed number of bits, jamming the channel, to insure that all stations sense the channel as busy. Retransmission is scheduled according to the value determined by "truncated binary exponential backoff": a delay of r slot times is observed — one slot equalling a fixed number of bit-times — before the n -th retransmission

attempt. The value of r is chosen as a uniformly distributed random integer in the range of $0 \leq r < 2k$, where $k = \min(n, \text{backoff limit})$. Transmission is attempted a limited total number of times; if all allowed attempts fail, an error is reported to higher protocol layers.

The parameter of slot time mentioned above is not just arbitrarily selected. The slot time must be an upper bound on the acquisition time of the medium and on the length of the frame fragment generated by a collision as well as the unit of time used in scheduling retransmissions. Clearly, the slot time is determined by the network implementation: it must be larger than the sum of the maximum round-trip propagation time and jam time. The system parameters chosen for this simulation study appear in Table 2.2.1; they stem from the 1980 DEC/Intel/Xerox Ethernet specification [15] and/or are cited in the first draft of the IEEE 802 Local Area Network Standards [9] as either recommended or worst-case values for a 10 Mbps implementation.

Since Jackson and Fuchs' work in characterizing the user-computer interaction in multiaccess computer communications (specifically, in time-sharing systems) [28], the Poisson process has been generally accepted as a reasonable description of the underlying arrival processes for "bursty" users. To model system traffic in this simulation, then, the generation of data frames is assumed to obey a Poisson arrival process with λ frames/second arriving to each station. Symmetric, balanced loading is assumed for simplicity and to ease the interpretation of results. In an actual system, traffic patterns would be defined by the type of stations supported by the network and the functional role of the network itself. Furthermore, higher level protocols introduce their own demands on the network capacity. For example, Marathe and Hawe's simulation of Ethernet performance [27] used measurements of existing installations to characterize the system loads resulting from user transactions in a program development environment where file edits, links, compiles, executes, copies, deletes, etc. are commonplace. They then parameterized the load by the number of system users and simulated an Ethernet system to obtain mean waiting times, mean number of attempts, and the percent idle time on the cable as a function of the number

of users. Here, system traffic is modelled by assuming a Poisson arrival rate to each station and a specific frame size distribution, as discussed in Section 2.5.

Transmission rate:	10 Mbps
Slot time:	512 bits
Minimum frame size:	512 bits
Maximum frame size:	12144 bits
Jam size:	32 bits
Attempt limit:	16 attempts
Backoff limit:	10
Interframe gap size:	9.6 μ s
Time required for a station to assert its collision detect signal following the actual reception of a collision:	9 bit-times
Time required for a station to assert its carrier sense signal following the onset of carrier presence information	2 bit-times
Time required for a station to disable its carrier sense signal following the loss of carrier presence information:	1.6 bit-times
Total duration of collision detection and jamming:	9 + 32 bit-times

Table 2.2.1 Simulation System Parameters

2.2.2 Decentralized Adaptive Probing

In 1978, Hayes advanced the technique of adaptive probing [16]. Conceived as a centralized access control protocol, adaptive probing is here considered in a decentralized form on the same sort of broadcast channel as described in the previous section on Ethernet. The probing mechanism grants access to the channel to groups of stations at a time, resolving contention between stations through a variation of the tree algorithms of Capetanakis [17,18]. Adaptive probing thus provides the respective advantages of random access protocols at low traffic levels and deterministic access protocols at high traffic levels.

Initially, assuming a quiescent channel, all stations operate in a random access mode, accessing the channel according to the CSMA-CD algorithm of Ethernet. In the event of a collision, however, individual colliding stations do not schedule their retransmissions randomly. Each station on the channel possesses a unique binary address for the purposes of this algorithm. After detecting a collision, jamming the channel, and finally sensing the channel again idle, colliding stations whose addresses begin with '0' attempt their first retransmission after observing the necessary inter-frame gap. Colliding stations whose addresses begin with '1' schedule their retransmissions to a future instant where it is assured that any transmission by stations of the first group will have been detected, thus avoiding collision between the two groups. Collisions arising within a group are similarly resolved, with each probed group of stations defined by their common address prefix; i.e., defined by having the same first n binary digits in their address. Stations continually monitor the channel, independently assessing the status of the probing process and rescheduling their transmissions accordingly. At the end of the tree search, all stations make independent estimates of the network's traffic level and determine an optimum point in the tree search algorithm at which to begin the next cycle. It is in this sense that the probing algorithm is adaptive — adapting itself to network traffic levels.

The algorithm is perhaps best explained through example. Figure 2.2.1 illustrates the worst-case situation which determines the adaptive probing scheduling delays. In Figure 2.2.1, the position of a station on the channel is represented by its horizontal position; time progresses vertically down the page. With the system initially in the random access mode, station 011 begins its transmission. Just prior to the elapse of t_p , the propagation delay between stations 011 and 000, and t_1 , the time required for a station to recognize the transmission, station 000 begins its own transmission. Almost immediately, station 000 has enough information to detect the collision and, upon doing so after an elapse of t_2 , jams the channel for a duration of t_3 . Station 011, however, does not receive information concerning the collision until t_p seconds after the start of station 000's transmission. At that time, station 011 is able to detect the collision and jam the channel. After a propagation delay t_p and the time t_4 required to disable its carrier sense signal, station 000 senses the channel idle. Similarly, station 011 senses the channel idle after the sum of the propagation delay t_p and the time t_4 required to disable the carrier sense signal has elapsed from the end of station 000's jam. When many stations are attempting transmission, of course, the beginning of each station's jam depends on the earliest instant at which collision information is received, and the time at which each station finally senses the channel idle depends on the latest instant at which the end of all stations' transmissions is received. In the simulation model of both Ethernet and adaptive probing, this minimization/maximization over all transmitting stations is performed for each station upon every initial transmission.

Having collided, the stations now begin the adaptive probing binary tree search strategy. Since the addresses of both stations in this example begin with '0', they both start transmitting after an interframe gap t_5 has elapsed from the time they sensed the channel idle. Again, a collision results. After ceasing their transmissions and finally sensing the channel idle, all stations with addresses beginning with '00' transmit after an interframe gap; stations with addresses beginning with '01' schedule their transmissions to avoid colliding with any of the other group. As seen in Figure 2.2.1, the minimum scheduling delay must be equal to the sum of twice

the propagation delay t_p , the interframe gap t_5 , and the time t_4 required for a station to recognize the presence of a transmission on the channel.

The preceding example shows the worst-case situation between stations that are members of adjacent probe groups, where "adjacent" means that their address prefixes differ by 1 in the least significant place. The scheduling delays employed by all ready stations must be sufficient to insure that all stations in a given probe group will sense activity on the channel generated by any preceding probe group and thus properly defer their transmissions. If addresses are assigned randomly to stations throughout the system, then the propagation delay used in the above scheduling delay calculation must be the network's maximum end-to-end propagation delay to thus account for the worst eventuality. Alternatively, if station addresses are assigned in an ordered fashion according to the physical location of each station, then the probing algorithm may take advantage of this grouping by associating a different delay with each probe level.

The probe level at any instant is given by the number of bits in the prefix currently used to identify the probed group of stations. A probe level is also identified for the entire probe cycle, designating the top level of the tree search algorithm. For a probe level n , there are 2^n groups of stations to be probed and thus $2^n - 1$ pairs of adjacent groups of stations. The delay associated with probe level n must therefore be the maximum propagation delay between any two stations which are elements of adjacent probe groups. To reduce scheduling delays, sequential station addresses should be assigned to stations local to each other to thus minimize the delay between adjacent probe groups at any probe level.

The scheduling delay that each ready station computes upon sensing the channel idle is based upon its knowledge of the probe cycle's current progress. Since each bit of the probe prefix corresponds to a probe level, the scheduling delay can be calculated by operating upon the prefix in the following manner. If the station's first n address bits correspond to the present n -bit probe prefix, then transmission is scheduled after the elapse of the interframe gap. Otherwise, the probe prefix is incremented by 1 and

the scheduling delay, as shown in Figure 2.2.1, includes as an initial offset the interframe gap, the time required to detect activity on the channel, and two maximum propagation delays associated with the probe level n . Two propagation delays rather than one are required to account for the worst-case sequence of events shown in Figure 2.2.1, where the timing is such that a station of the currently probed group senses the channel idle a full propagation delay after the instant that a station of the next probe group does. In incrementing the prefix, least significant zero bits extending beyond the number of prefix bits designated by the cycle's probe level are dropped: the tree search effectively climbs back up to its initial level. If, after this first step, the resulting prefix does not correspond to either the station's address or the start of the next probe cycle as defined by a prefix of k 0's, where k is the current probe cycle's probe level, then the prefix is again incremented until either of these states is reached. With each increment, the propagation delay associated with the resultant number of bits in the prefix; i.e., associated with that probe level, is added to the scheduling delay. Only a single propagation delay is added since, at this point, one is merely scheduling probe groups out of each other's way so that any transmissions by previous probe groups will be sensed prior to the scheduled transmission time.

When this algorithm stops at the beginning of the next probe cycle, the station is effectively scheduling itself to estimate system traffic, evaluate a new probe level for the next probe cycle, and then schedule its transmission in that new cycle. Since all stations that are ready to transmit are continually monitoring the channel and rescheduling their transmission, ultimately all ready stations who do not receive access to the channel during the current probe cycle schedule themselves to the beginning of the next cycle. The last calculations made as the current probe cycle ends are the ones that finally and accurately determine the beginning of the next cycle with respect to each ready station.

The above scheduling algorithm assumes that all stations know the value of the current probe prefix upon sensing the channel idle. The probe prefix, however, reflects both the cycle's probe level and the current status

of tree search. Again, an example perhaps best illustrates the algorithm. In Figure 2.2.2, the probe level has been established as 3 for the probe cycle. Initially, then, all stations with addresses beginning with '000' are authorized to transmit. A collision results, so all stations desiring to transmit a frame schedule their transmissions according to the above procedure and a new prefix of '0000'. No stations prefixed by '0000' are ready to transmit. Because all stations are continually monitoring the channel, those stations who scheduled themselves in the next probe group; i.e., those stations prefixed by '0001', are aware of the elapsed idle time since the channel was last sensed busy. Clearly, if a collision occurred among stations prefixed by '000', yet no transmissions were attempted by those stations prefixed by '0000', then at least two ready stations are in the probe group identified by '0001'. To therefore avoid a needless collision, level-skipping is employed whereby all ready stations reschedule their transmissions according to a new prefix of '00010'. Similarly, if no station prefixed by '00010' were to transmit, then, upon reaching their scheduled transmission time without sensing activity on the channel, ready stations prefixed by '00011' and all other ready stations would level-skip and reschedule their transmissions according to a new prefix of '000110'.

In this example, however, some station prefixed by '00010' does successfully transmit its data frame. The next probe group, prefixed by '00011', also yields a successful transmission. Incrementing this prefix to calculate the next set of scheduling delays results in '00100'. Trailing zeroes beyond the cycle's probe level of 3 are dropped: the probing algorithm has finished granting access to stations prefixed by '000' and now authorizes access to those stations prefixed by '001'. As this example shows, knowledge of the probing algorithm and its scheduling delays and observation of activity on the channel allow any station to independently determine the present status of the tree search and act accordingly.

While the foregoing discussion has explained the workings of decentralized probing, the adaptive response of the algorithm to system traffic levels has yet to be discussed. At low traffic levels, the algorithm should allow random access to the channel; at high traffic levels, excessive con-

attention for the channel should be avoided by deterministically allocating access. The probing technique readily provides the means to control this type of behavior through the selection of the probe level used at the beginning of a probe cycle. Hayes [16] has shown that the mean duration of a probe cycle is minimized if the cycle begins by probing groups of 2^j stations at a time, where j is the maximum integer such that $(1-p)^{2^j} \geq .5$. The parameter p is the probability that a station has a data frame to transmit and is assumed to be equal to $1-e^{-\lambda T}$, where λ is the Poisson arrival rate to each station and T is the duration of the previous probe cycle. Besides assuming that message generation is a Poisson process, this result assumes that a station, once having gained access to the channel, transmits all of its buffered data frames except for those frames which were generated during the current probe cycle. The expression for p is thus valid, and the above choice of j is intuitively appealing as that choice which selects the largest size initial probe group that has a greater than 50% chance of possessing no data frames at all to transmit.

The decentralized probing algorithm modelled here, however, allows new packets to enter into the ongoing probe cycle and, most importantly, allows a station to transmit only one frame per probe cycle. This last policy is reasonable in itself, but is in fact expedient given that an interframe gap must be maintained between frames. These two features, though, render the preceding expression for p incorrect. Nonetheless, the rule outlined above is used to provide a means, however suboptimum, to direct the adaptive setting of the probe cycle's initial probe level. To implement an adaptive strategy in this decentralized context, all stations are assumed to know the number of stations, N , currently in the system. This information would perhaps be sent to all stations by a system supervisor every time the network configuration changed, subject to any uncertainty due to the failure of a station or stations. Since all stations are continually monitoring the channel, they each may time the length of the previous probe cycle, T , and estimate the mean arrival rate per station, $\hat{\lambda}$, as the inverse of the average time between successful transmissions divided by the number of stations. To account for possible changes in system traffic with time, the average time between successful transmissions should be computed as a moving average,

where the number of intervals sampled should be large enough to provide an accurate estimate of the mean, yet small enough to be sensitive to any recent change in traffic levels. In the simulation model, an arbitrary window of the last 65 successful transmissions was used although traffic levels were never considered to be time-varying.

The number of stations in the system is not constrained to be an integer power of 2; in a real network, such a restriction would be ludicrous. Let the integer m be such that $2^m \geq N > 2^{m-1}$. At the end of each probe cycle, then, each station independently evaluates $p = (1 - e^{-\hat{\lambda}T})$ and determines the probe level of the next probe cycle as $m - j$, where j is, again, the maximum integer such that $(1 - p)^{2^j} \geq .5$, $0 \leq j \leq m$. Due to propagation delays, it is conceivable that individual estimates at each station of $\hat{\lambda}$ and T could vary enough to yield different results. To avoid any ambiguity, it is assumed that every packet includes a data field giving the probe cycle's probe level as well as the originating station's probe address and a prefix pointer indicating the least significant bit position of the prefix with respect to that address. Any confusion resulting from different interpretations of the initial probe level will eventually be dispelled when, finally, the probing mechanism enables a particular station to successfully transmit its packet and all stations then accept that station's conception of the current probe cycle.

The decentralized adaptive probing algorithm detailed here exhibits all the features of centralized adaptive probing except that control of the algorithm is distributed among stations in the network. The intelligence and decision-making capability required at each station is thus significantly greater than in the centralized algorithm where stations need only respond to interrogation by the central processor. With the current technology of very large scale integrated circuitry, however, the demands of the decentralized algorithm could be readily met.



Event

- A Station begins transmission
- B Station detects collision and begins jamming channel
- C Station senses channel idle

Parameters

- t_p Maximum propagation delay between station 000 and 011
- t_1 Time required for a station to assert its carrier sense signal following the onset of carrier presence information
- t_2 Time required for a station to assert its collision detect signal following the reception of collision information
- t_3 Jam time
- t_4 Time required for a station to disable its carrier sense signal following the loss of carrier presence information
- t_5 Interframe gap

Figure 2.2.1 Timing Considerations in Decentralized Adaptive Probing
(continued)

Probe Level 3:

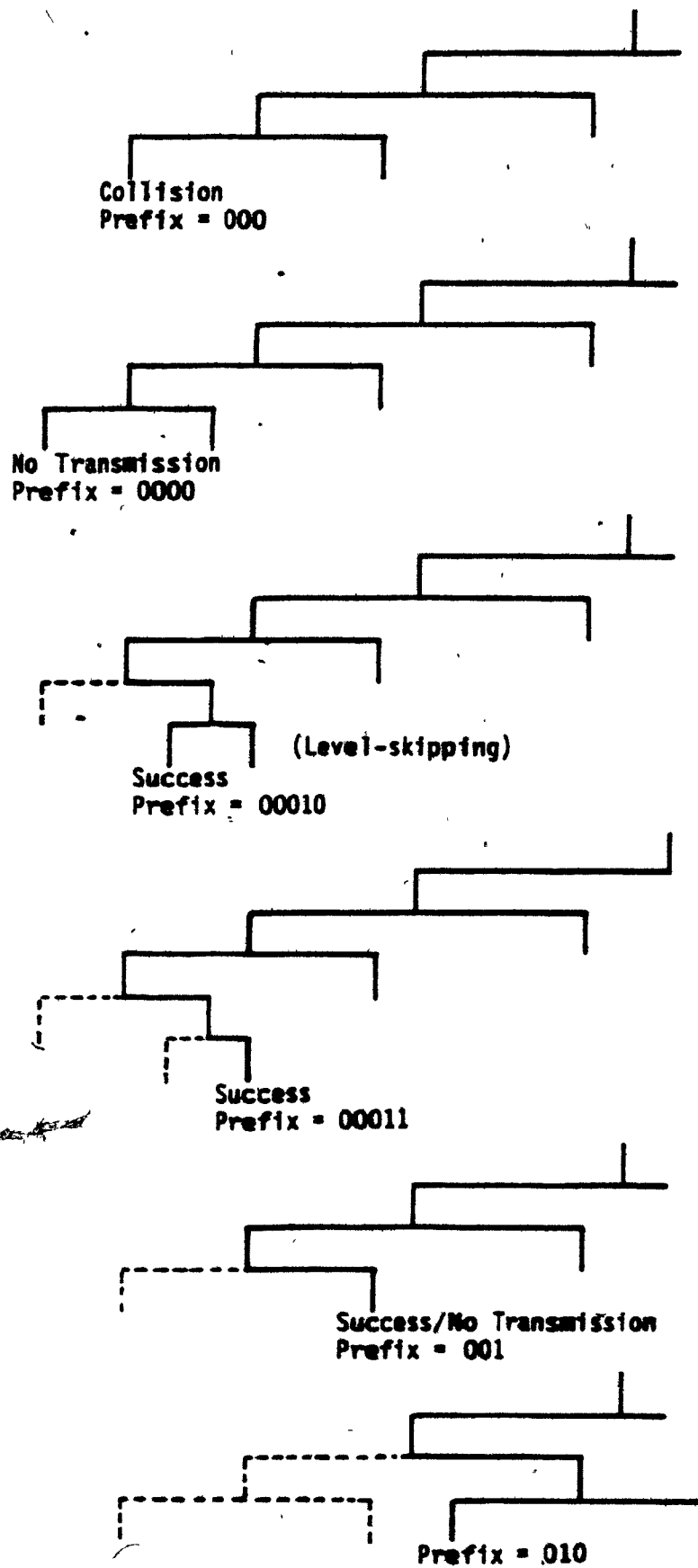


Figure 2.2.2 The Probing Algorithm

2.2.3 System Configuration

In the simulations of both Ethernet and decentralized adaptive probing, a network topology was assumed consistent with the first draft of the IEEE 802 Committee's proposed local network standards [9]. A baseband coaxial cable transmission system operating at 10 Mbps is specified, with the connection between a station and the cable accomplished by a media access unit. The media access unit (MAU), an implementation of the OSI Physical Layer, connects directly to the coaxial cable and provides the hardware to send and receive the encoded signals, detect collisions on the cable, and provide the necessary electrical isolation. The MAU capability is fully specified by the IEEE standard to guarantee system compatibility.

The IEEE standard applies physical limits to the transmission system configuration to constrain maximum propagation times to support proper CSMA-CD access protocol performance. The following restrictions were observed in the simulation model:

- A coaxial cable terminated by its characteristic impedance at each end constitutes a coax cable segment. A coax segment may have a maximum length of 500 meters; the cable must support a minimum propagation velocity of $.77 c$, where $c = 300,000$ kilometers per second.
- A point-to-point link constitutes a link segment. A maximum of 1000 meters may be used anywhere in the system, supporting a minimum propagation velocity of $.65 c$.
- The interconnection cable between a station and its associated MAU or between a repeater and coax or link segment may have a maximum length of 50 meters; the cable must support a minimum propagation velocity of $.65 c$.
- A maximum total number of 100 MAU's and repeaters may be connected to each coax cable segment. Since the connections to the cable cause signal reflections due to their finite bridging impedance, the coaxial cable jackets are marked every 2.5 meters and connections

are required to be made at these marks. This policy guarantees a minimum spacing between MAU's/repeaters and controls their relative spacing, insuring their non-alignment on fractional wavelength boundaries so that reflections do not add in phase to a significant degree.

- * ● A maximum of two repeaters may exist in the transmission path between any two stations. Repeaters need not be located at the ends of cable segments, nor must there only be one repeater per cable segment. A point-to-point link is assumed to encompass one repeater set. In the maximum size system configuration, then, a signal may travel the full length of 3 coax cable segments, 1000 meters of point-to-point link, and the lengths of 6 interconnection cables (one at the sending station, one at the receiving station, and two at each of two repeaters). A repeater is assumed to impose a delay of 2 bit times.

The above specifications are identical to the 1980 DEC-Intel-Xerox Ethernet specifications [15]. The most recent revision of the IEEE standards [10], however, permits a somewhat larger configuration. The asterisked specifications above are there redefined as follows:

- A point-to-point link constitutes a link segment. A link segment may contain a maximum end-to-end propagation delay of 2570 nanoseconds and must terminate in a repeater set at each end. Two adjacent link segments may be joined without a repeater set between them.
- The maximum transmission path permitted between any two stations is five segments (coax or link), four repeater sets and their interconnection cables, two MAU's, and two interconnection cables at the sending and receiving stations. Of the five segments, a maximum of three may be coax segments; the remainder are link segments.

Notwithstanding the above revisions, the following conclusions can be made regarding the maximum, worst-case system configuration permitted by the

initial specifications. A single cable system will possess a maximum end-to-end propagation delay of 2.677 μ s due to a maximum 500 meters of coaxial cable (supporting a propagation velocity of .77 c) plus 100 meters of interconnection cable (supporting a propagation velocity of .65 c) between the MAU's and the stations at the cable ends. A full-scale system will, in the worst case, involve 1500 meters of coaxial cable, two repeaters each interposing a delay of two bit-times (.2 μ s for a 10 Mbps system) and each requiring a maximum 100 meters of interconnection cable, 1000 meters of point-to-point link, and 100 meters of interconnection cable for the two stations at the system's extreme ends. The resultant maximum end-to-end propagation delay is thus 13.560 μ s.

In this simulation study, both a single cable system supporting 100 stations and a full-scale system supporting 256 stations were considered. Additional simulations were also performed of systems comprising 48 and 64 stations, but those results are not formally presented here. The full-scale system is shown in Figure 2.2.3, with Table 2.2.2 listing the 256 stations. Table 2.2.3 lists the 100 stations supported on the single cable system. From these topologies and the preceding requirements, the matrix of station-to-station propagation delays was generated and used throughout the simulation programs.

As discussed in Section 2.2.2, the decentralized adaptive probing algorithm relies upon scheduling delays to control access to the channel, where the scheduling delays depend upon the propagation delays between stations. If the scheduling delays are made probe level-dependent to thereby minimize their impact, the local stations are assigned consecutive addresses, as shown in Tables 2.2.2 and 2.2.3.

For a single-cable system, the optimum assignment of addresses is obvious: the cable is partitioned linearly by the station addresses, the first bit of the address dividing the cable into approximate halves, the second bit dividing the cable into quarters, and so on. Figure 2.2.4 shows, for the network simulated, the correspondences between station address prefixes and the actual station location. The maximum end-to-end propagation delays between adjacent probe groups help define the level-dependent

scheduling delays and are indicated in the figure. For systems composed of more than one cable, however, the topology need not be one-dimensional, making a linear partition impossible. Figure 2.2.5 shows the partitioning used for the full-scale system considered here. The task of assigning addresses to stations in a given configuration to minimize the delays between adjacent probe groups can be solved as a linear programming problem, but, for the purposes of this study, the addresses were distributed in an empirical fashion. The resulting partitioning is thus not ideal, but it does permit level-dependent scheduling delays: In an operational system, one could conceive of a system supervisor dynamically assigning addresses with each change in network configuration to take best advantage of the topology.

Transmission rate:	10 Mbps
Slot time:	512 bits
Minimum frame size:	512 bits
Maximum frame size:	12144 bits
Jam size:	32 bits
Attempt limit:	16 attempts
Backoff limit:	10
Interframe gap size:	9.6 μ s
Time required for a station to assert its collision detect signal following the actual reception of a collision:	9 bit-times
Time required for a station to assert its carrier sense signal following the onset of carrier presence information	2 bit-times
Time required for a station to disable its carrier sense signal following the loss of carrier presence information:	1.6 bit-times
Total duration of collision detection and jamming:	9 + 32 bit-times

Table 2.2.1 Simulation System Parameters

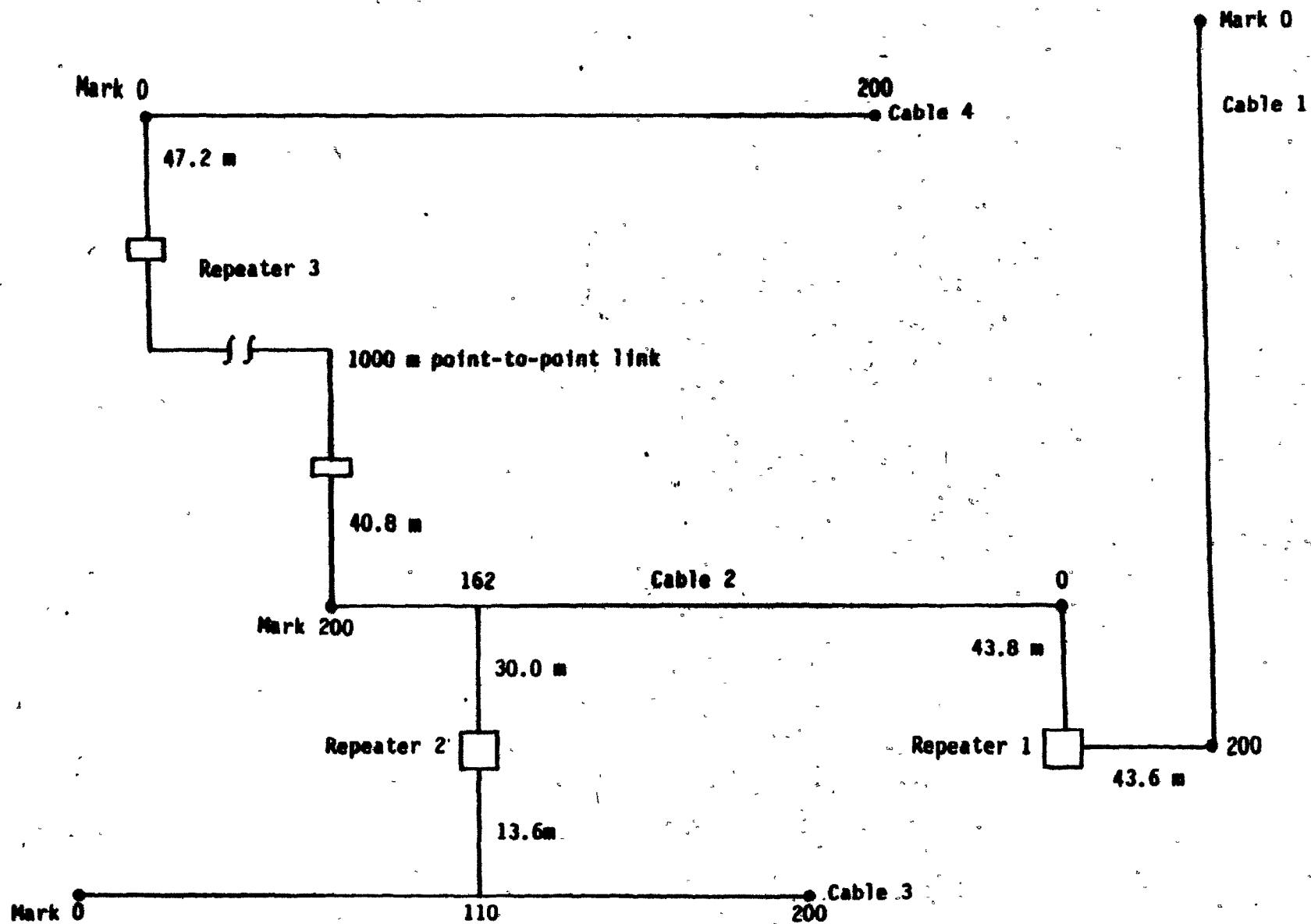


Figure 2.2.3 Full-Scale System Configuration

Number of stations = 256

Cable:	Mark:	Address:	NWU cable length:	metre
1	0	00000000	0.1	
1	2	00000001	47.7	
1	3	00000010	21.7	
1	5	00000011	27.0	
1	8	00000100	38.0	
1	9	00000101	18.9	
1	11	00000110	2.9	
1	12	00000111	39.6	
1	13	00001000	0.4	
1	15	00001001	28.1	
1	20	00001010	9.5	
1	24	00001011	19.9	
1	26	00001100	31.7	
1	29	00001101	15.0	
1	30	00001110	38.9	
1	31	00001111	27.8	
1	32	00010000	41.6	
1	33	00010001	29.1	
1	34	00010010	48.0	
1	37	00010011	19.7	
1	38	00010100	34.5	
1	40	00010101	48.2	
1	42	00010110	3.8	
1	45	00010111	48.7	
1	46	00011000	28.6	
1	47	00011001	18.1	
1	48	00011010	12.7	
1	53	00011011	32.4	
1	55	00011100	18.2	
1	56	00011101	24.3	
1	58	00011110	7.7	
1	61	00011111	38.3	
1	64	00100000	42.7	
1	65	00100001	34.6	
1	67	00100010	18.0	
1	69	00100011	23.0	
1	72	00100100	15.0	
1	74	00100101	42.2	
1	79	00100110	43.0	
1	82	00100111	16.6	
1	85	00101000	16.9	
1	87	00101001	4.2	
1	90	00101010	7.5	
1	93	00101011	9.5	
1	94	00101100	34.1	
1	98	00101101	18.1	
1	99	00101110	48.0	
1	100	00101111	8.2	
1	102	00110000	11.5	
1	104	00110001	26.7	
1	106	00110010	23.2	
1	111	00110011	39.0	
1	112	00110100	24.5	
1	115	00110101	24.0	
1	116	00110110	12.2	
1	120	00110111	28.9	
1	121	00111000	18.1	

Table 2.2.2 256-Station Network

Number of stations = 256

Cable:	Mark:	Address:	HWU cable length:
1	123	001110010	43.6 metre
1	123	001110100	40.0
1	130	001110110	32.3
1	132	001111000	24.2
1	134	001111010	23.9
1	139	001111100	11.3
1	140	001111110	21.1
1	141	010000000	23.3
1	147	010000010	20.6
1	150	010000100	42.9
1	154	010000110	3.7
1	157	010001000	25.0
1	160	010001010	30.4
1	169	010001100	42.9
1	170	010001110	44.0
1	172	010010000	1.9
1	176	010010010	40.6
1	182	010010100	26.3
1	184	010010110	46.9
1	189	010011000	3.2
1	192	010011010	47.9
1	194	010011100	34.3
1	196	010011110	14.3
1	197	010100000	39.4
1	198	010100010	47.3
2	4	010100100	0.0
2	5	010100110	23.0
2	7	010101000	40.3
2	8	010101010	32.9
2	10	010101100	3.0
2	11	010101110	0.6
2	12	010110000	23.3
2	14	010110010	1.0
2	15	010110100	14.0
2	24	010110110	37.6
2	25	010111000	23.4
2	26	010111010	26.6
2	37	010111100	47.0
2	38	010111110	44.2
2	39	011000000	10.3
2	40	011000010	7.6
2	41	011000100	16.3
2	42	011000110	30.1
2	43	011001000	12.3
2	44	011001010	14.7
2	46	011001100	20.9
2	49	011001110	22.2
2	50	011010000	0.6
2	53	011010010	1.1
2	70	011010100	41.2
2	73	011010110	3.6
2	84	011011000	32.9
2	102	011011010	42.1
2	103	011011100	30.0
2	105	011011110	0.2
2	106	011100000	30.7

Table 2.2.2 256-Station Network (continued)

Number of stations = 256

Cable:	Mark:	Address:	NRU cable length:
2	120	011100010	22.7
2	124	011100100	35.1
2	126	011100110	46.4
2	128	0111001000	32.6
2	140	011101010	1.9
2	141	011101100	0.6
2	142	011101110	40.4
2	143	011110000	11.1
2	149	011110010	4.2
2	154	011110100	12.3
2	156	011110110	1.3
2	157	011111000	6.6
2	158	011111010	44.4
2	159	011111100	9.1
2	160	011111110	20.4
2	174	100000000	20.2
2	176	100000010	14.1
2	177	1000000100	19.1
2	178	1000000110	3.1
2	179	1000001000	40.7
2	180	1000001010	49.4
2	181	1000001100	36.1
2	184	1000001110	13.7
2	185	1000010000	15.2
2	186	1000010010	34.1
2	189	1000010100	1.3
2	190	1000010110	2.0
2	192	1000011000	22.3
2	194	1000011010	7.0
2	196	1000011100	4.9
3	3	1000011110	30.0
3	4	1001000000	20.1
3	5	1001000010	15.8
3	6	1001000100	0.4
3	7	1001000110	1.7
3	12	1001010000	39.9
3	20	1001010100	24.0
3	22	1001011000	3.6
3	27	1001011100	42.7
3	30	1001100000	32.3
3	31	1001100100	26.0
3	32	1001101000	15.3
3	33	1001101100	0.6
3	40	1001110000	0.1
3	42	1001110100	7.2
3	43	1001111000	12.9
3	42	1001111100	11.0
3	44	1010000000	27.4
3	46	1010000100	24.0
3	44	1010000110	19.3
3	46	1010000111	10.2
3	45	1010001000	9.9
3	47	1010001010	0.0
3	48	1010001100	7.2
3	49	1010001110	19.3
3	50	1010100000	0.7

Table 2.2.2 256-Station Network (continued)

Number of stations = 256

Cable:	Mark:	Address:	MAU cable length:
J	91	101010010	11.1 metre
J	92	101010100	10.1
J	93	101010110	22.0
J	94	101011000	32.0
J	95	101011010	23.6
J	109	101011100	44.6
J	111	101011110	17.6
J	120	101100000	12.2
J	122	101100010	43.3
J	123	101100100	4.3
J	126	101100110	30.3
J	129	101101000	40.9
J	140	101101010	13.4
J	141	101101100	40.7
J	148	101101110	31.2
J	150	101110000	30.6
J	154	101110010	34.3
J	159	101110100	22.1
J	165	101110110	22.7
J	172	101111000	46.8
J	173	101111010	20.9
J	180	101111100	9.6
J	186	101111110	23.7
J	190	110000000	10.1
J	194	110000010	27.7
J	199	110000100	16.9
4	1	110000110	40.0
4	2	110001000	19.0
4	4	110001010	19.9
4	5	110001100	13.9
4	6	110001110	41.0
4	8	110010000	3.0
4	12	110010010	2.0
4	18	110010100	20.6
4	20	110010110	11.0
4	23	110011000	27.1
4	27	110011010	27.0
4	29	110011100	24.7
4	30	110011110	40.3
4	33	110100000	3.7
4	34	110100010	46.7
4	35	110100100	43.7
4	40	110100110	10.3
4	60	110101000	10.0
4	62	110101010	27.9
4	63	110101100	40.3
4	80	110101110	30.2
4	93	110110000	39.0
4	101	110110010	43.0
4	102	110110100	37.4
4	103	110110110	29.0
4	104	110111000	44.2
4	110	110111010	36.7
4	111	110111100	13.2
4	112	110111110	20.7
4	119	111000000	17.1
4	121	111000010	1.6

Table 2.2.2 256-Station Network (continued)

Number of stations = 256

Cable:	4	Mark:	122	Address:	111000100	MAU cable length:	13.8	metre
	4		138		111000110		29.7	
	4		139		111001000		35.1	
	4		140		111001010		17.5	
	4		141		111001100		47.1	
	4		142		111001110		28.4	
	4		144		111010000		38.8	
	4		146		111010010		17.7	
	4		147		111010100		26.1	
	4		148		111010110		15.5	
	4		160		111011000		46.5	
	4		161		111011010		41.4	
	4		162		111011100		24.7	
	4		163		111011110		9.8	
	4		164		111100000		29.9	
	4		165		111100010		13.8	
	4		176		111100100		8.8	
	4		177		111100110		22.9	
	4		178		111101000		14.3	
	4		179		111101010		45.1	
	4		180		111101100		34.8	
	4		184		111101110		45.5	
	4		185		111110000		7.3	
	4		190		111110010		43.9	
	4		193		111110100		22.1	
	4		194		111110110		36.5	
	4		195		111111000		44.7	
	4		196		111111010		35.5	
	4		199		111111100		24.2	
	4		200		111111110		8.6	

Number of repeaters = 3

Repeater 1 links cable 1 to cable 2

Cable:	1	Mark:	200	MAU cable length:	43.6	metres
Cable:	2	Mark:	0	MAU cable length:	43.8	metres

Repeater 2 links cable 2 to cable 3

Cable:	2	Mark:	162	MAU cable length:	38.8	metres
Cable:	3	Mark:	110	MAU cable length:	13.6	metres

Repeater 3 links cable 2 to cable 4

Cable:	2	Mark:	200	MAU cable length:	48.8	metres
Cable:	4	Mark:	0	MAU cable length:	47.2	metres

Length of internal point-to-point link = 1888.8 metres

Table 2.2.2 256-Station Network (continued)

Number of stations = 100

Cable:	1	Mark:	0	Address:	0000000	MAN cable length.	0.1 metres
	1		2		0000001		23.3
	1		3		0000010		47.7
	1		4		0000100		21.7
	1		5		0000101		28.6
	1		6		0000110		27.0
	1		8		0000111		42.9
	1		9		0001000		38.0
	1		12		0001001		3.7
	1		19		0001010		18.9
	1		20		0001100		2.9
	1		21		0001101		26.8
	1		23		0001110		39.6
	1		24		0001111		38.4
	1		25		0010000		0.4
	1		26		0010010		20.1
	1		27		0010100		9.3
	1		38		0010101		42.9
	1		44		0010110		19.9
	1		45		0011000		31.7
	1		46		0011010		13.8
	1		47		0011100		38.9
	1		48		0011110		27.8
	1		49		0100000		41.6
	1		50		0100001		44.8
	1		58		0100010		29.1
	1		59		0100011		1.9
	1		60		0100100		45.0
	1		62		0100101		48.6
	1		63		0100110		19.7
	1		64		0101000		34.3
	1		70		0101001		26.3
	1		72		0101010		48.2
	1		76		0101011		46.9
	1		78		0101100		3.8
	1		80		0101101		3.2
	1		83		0101110		48.7
	1		84		0101111		47.9
	1		86		0110000		20.6
	1		87		0110010		18.1
	1		88		0110100		12.7
	1		90		0110101		34.3
	1		92		0110110		32.4
	1		93		0110111		14.3
	1		94		0111000		18.2
	1		95		0111010		26.3
	1		96		0111011		39.4
	1		97		0111100		7.7
	1		98		0111110		38.3
	1		100		0111111		47.3
	1		101		1000000		42.7
	1		103		1000001		43.6
	1		104		1000010		34.6
	1		105		1000011		43.8
	1		106		1000100		18.0
	1		107		1000101		8.8
	1		108		1000110		23.8

Table 2.2.3 100-Station Network

Number of stations = 100

Cable:	1	Mark:	100	Address:	1001000	MMU cable length:	13.0 metres
	1		110		1001001		23.0
	1		112		1001010		42.2
	1		116		1001011		40.3
	1		118		1001100		43.0
	1		120		1001101		33.9
	1		122		1001110		16.6
	1		124		1010000		16.9
	1		125		1010001		3.0
	1		126		1010010		4.2
	1		130		1010011		0.6
	1		134		1010100		7.5
	1		135		1010110		9.5
	1		136		1010111		29.3
	1		137		1011000		34.1
	1		140		1011001		1.0
	1		140		1011010		10.1
	1		149		1011100		40.8
	1		150		1011110		8.2
	1		153		1011111		14.8
	1		160		1100000		11.5
	1		162		1100001		37.6
	1		163		1100011		23.4
	1		164		1100010		26.7
	1		165		1100111		26.6
	1		167		1100100		23.2
	1		160		1100110		39.0
	1		170		1101000		26.5
	1		171		1101010		24.8
	1		173		1101100		12.2
	1		170		1101101		47.0
	1		180		1101110		20.9
	1		181		1110000		18.1
	1		182		1110010		43.6
	1		183		1110100		40.8
	1		184		1110101		44.2
	1		186		1110110		33.3
	1		190		1111000		24.2
	1		192		1111010		23.9
	1		194		1111100		11.3
	1		196		1111011		10.3
	1		190		1111101		7.6
	1		200		1111110		21.1

Table 2.2.3 100-Station Network (continued)

Probe Level 1
2.677 μ s (200 marks)



Probe Level 2
1.649 μ s (105 marks)



Probe Level 3
1.162 μ s (60 marks)



Probe Level 4
.946 μ s (40 marks)



Probe Level 5
.783 μ s (25 marks)



Mark

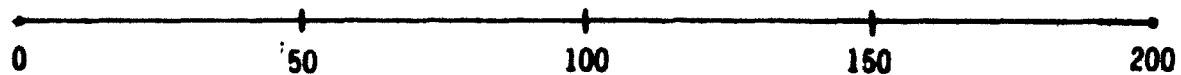


Figure 2.2.4 Level-Dependent Maximum Propagation Delays for the Single-Cable System

Probe Level 1: 13.560 μ s

Probe Level 2: 11.367 μ s

Probe Level 3: 11.151 μ s

Probe Level 4: 10.285 μ s

Probe Level 5: 9.365 μ s

Probe Level 6: 9.041 μ s

Probe Level 7: 9.041 μ s

Probe Level 8: 9.041 μ s

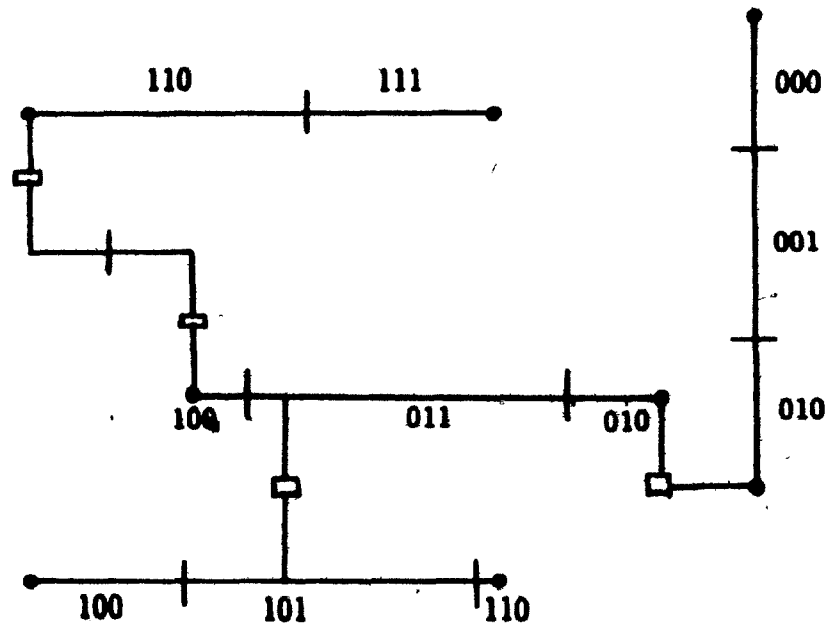


Figure 2.2.5 Level-Dependent Maximum Propagation Delays for the Full-Scale System

2.3 Simulation Techniques

Simulation of access control protocols implies more than just the programming of the access control algorithms and the artificial generation of network traffic — statistically valid estimates of the various performance indicators are desired. Sections 2.3.1 through 2.3.3 discuss the techniques employed here to obtain point estimates and confidence intervals from the simulations of Ethernet and decentralized adaptive probing. Section 2.3.2 explains regenerative simulation and how a simulation run may be considered to comprise a number of independent and identically distributed subruns; Section 2.3.2 considers the criteria used to halt a simulation run; and Section 2.3.3 treats the jackknife estimator, the ratio estimator used to obtain point estimates and confidence intervals in these simulations.

2.3.1 Regenerative Simulation

The simulation of the access control protocols modelled here (or of any stochastic system) defines a statistical experiment. Statistically valid statements regarding the simulation results are required if meaningful conclusions are to be drawn. For that reason, not only is a point estimate of a particular performance indicator desired, but so too is a confidence interval about such an estimate. Although simulation techniques have been used to study transient behavior [19], in this study the estimated performance indicators are meant to reflect stationary or steady state behavior of the system. Statistics-taking from the simulation, however, is made difficult by the statistical dependence between successive observations and the inability to initialize the simulation with steady state conditions.

Such difficulties are avoided, though, if it is possible to find a means of dividing a simulation run into independently and identically distributed (i.i.d.) subruns (or tours). The methods of classical statistical analysis (e.g., the estimation of confidence intervals, hypothesis testing, regression, and sequential estimation) may then be applied [37]. Stochastic processes known as regenerative processes fulfill the above need by definition. A rigorous mathematical presentation of regenerative processes may be found in Reference [39], but, informally, a regenerative process may be defined as a k -dimensional vector stochastic process $\{Z(t); t \geq 0\}$ which possesses an increasing sequence of random regeneration times $\{s_n; n \geq 1\}$ such that between any two consecutive regeneration times s_n and s_{n+1} the portion $\{Z(t); s_n \leq t \leq s_{n+1}\}$ of the regenerative process is an independent and identically distributed replica of the portion between any two other consecutive regenerative times. The expected time between successive regeneration points is assumed to be finite. In a typical situation, then, s_n represents the n -th entrance to a particular fixed state upon which the process (and the simulation) proceeds as if "from scratch" [37]. The points $\{s_n\}$ effectively demarcate an imbedded Markov chain.

The choice of state as a regenerative point is somewhat arbitrary. Crane and Iglehart [38], for example, note that for discrete- or continuous-valued Markov chains the expected width of the confidence intervals about

the various point estimates is independent of which state is chosen, given a fixed simulation run time. The regeneration state may thus be chosen so that the tours will be short or so that it is easy to initialize the system in that state.

From the procedural definitions of the CSMA-CD and adaptive probing access control protocols modelled here, it is apparent that a regenerative point may be easily defined by the state when the channel is idle and no station has a frame in its buffer to be transmitted. In the case of the adaptive probing algorithm, the probe level must also be at zero; i.e., the system must be in contention mode with no ongoing probe cycle. The computer simulations were therefore initialized in this idle state with the first and successive tours defined by the system's return to this state. Since decentralized adaptive probing involves the estimation by each station of the system load, the tours defined by the idle state are not truly independent, but, because the system load is held constant throughout a simulation run, the tours are nearly independent. To allow the stations' estimates to stabilize, an initial number of tours in the adaptive probing simulation runs was discarded and not used for statistics-taking.

It may happen, however, that, for example, at high levels of network traffic in these access control simulations, the system returns only infrequently to the regeneration point. The mean time between regeneration points is thus inordinately long, and the achievement of the desired number of tours requires an excessive amount of computer simulation time. Alternatives to the regenerative method are then possible. Sauer [42] discusses what he terms the aggregate state and the batches of events methods.

In the aggregate state method, a set of states rather than just one specific state is chosen as the regeneration point. A tour is then defined as the interval between the instants when the process initially enters into the aggregate state and, after having left the aggregate state, then returns to an element of the aggregate state. The rationale for choosing a particular set of states as the aggregate state is typically the desire to obtain shorter tour durations. If, however, the aggregate state is appropriately chosen so that it defines, loosely speaking, an "interval" about a true

regeneration point, then, as Crane and Iglehart note in their discussion of approximation techniques in the simulation of stable stochastic systems [40], the tours thus obtained may be approximately i.i.d.

In the batches of events method, each tour simply consists of a fixed number of events with the initial state arbitrarily chosen. The hope is that the tours so defined will be i.i.d. if the number of events per tour is sufficiently large. The difficulty lies in determining the appropriate batch size. In an empirical study of such simulation techniques, Sauer [42] reports that both the aggregate state and batches of events methods may yield valid confidence intervals given an appropriate selection of the relevant parameters. While there is no firm theoretical basis to determine either the number of events that comprise a tour in the batches of events method or the states that define the aggregate state in the aggregate state method, Sauer prefers the aggregate state method as an alternative to regenerative simulation.

On the basis of Sauer's experience, the aggregate state method was used here in certain cases to obtain the desired simulation results. As noted previously, using the idle state as a regeneration point leads to excessive run times in obtaining the requisite number of tours when the simulated system experiences high traffic levels. The difficulty is simply that, at high levels, the system does not often fall idle. The decrease in the number of tours achieved as system load increases is illustrated by Figure 2.3.1. The problem is exacerbated when the buffer size is greater than one, since then the stations effectively store demands made upon the capacity of the channel, and when the protocol governing access to the channel is adaptive probing. In that case, as described in Section 2.2.2, the protocol responds to a history of activity on the channel by setting the probe level to a non-zero value, thereby anticipating the likelihood of contention. Even if the channel should become quiet and the buffers at all stations empty, at high load levels the adaptive probing algorithm would probably maintain a non-zero probe level. To obtain simulation results in such instances, then, the state defining a regeneration point was typically expanded into an aggregate state by incorporating states with any value of probe level (but only at the end of a probe cycle) and stations with less

than or equal to, say, one frame in their buffers. Even with this application of the aggregate state method, the number of events occurring per tour was great enough that this choice of aggregate state may be justified by the same reasoning used in the discussion of the batches of events method. The effect of using such approximations to regenerative simulation may be seen in some of the results presented in Section 2.5: at high loads, some of the confidence intervals shown are not as small as one would prefer.

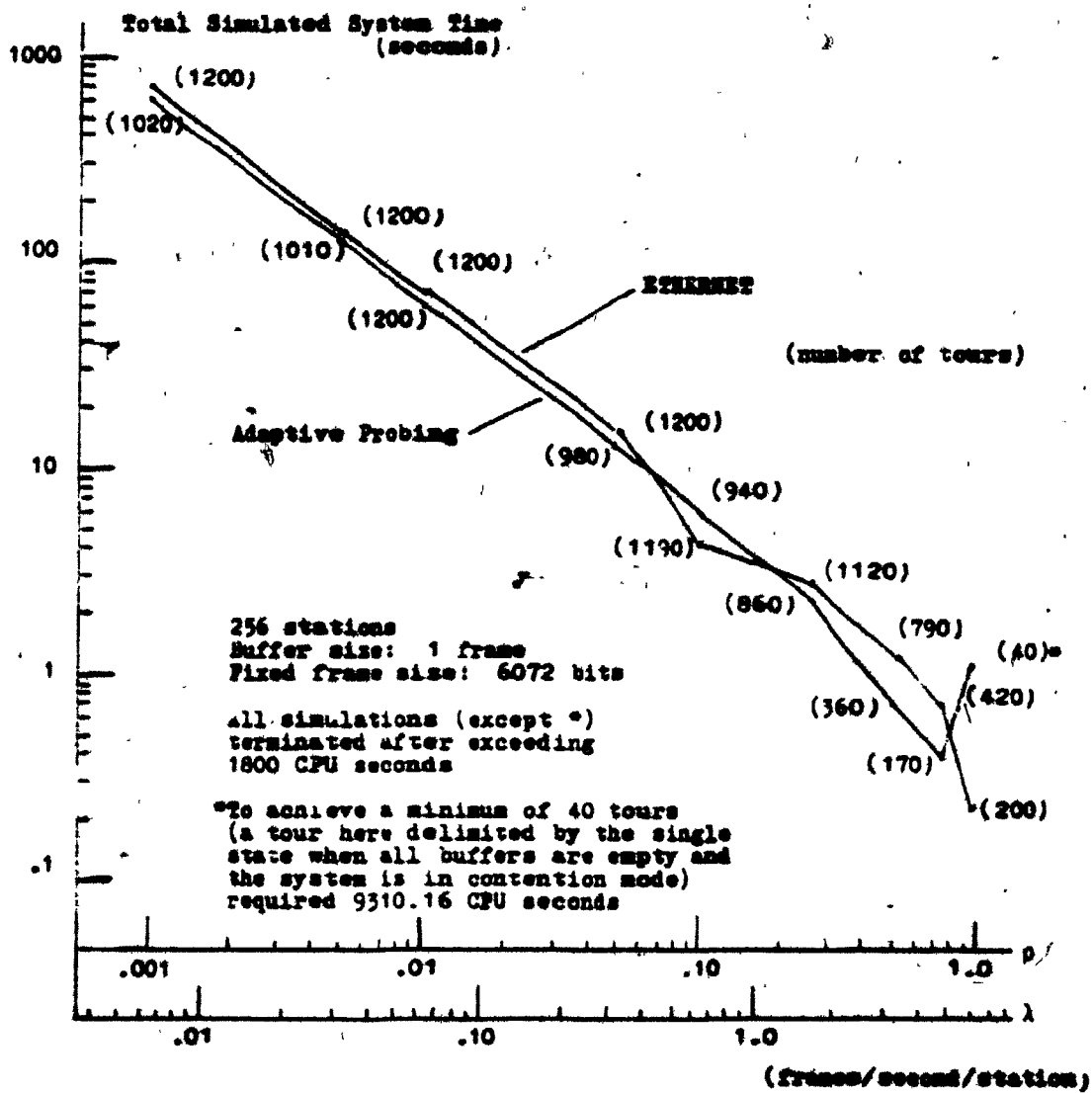


Figure 2.3.1 Simulation Performance As a Function of Load

2.3.2 Stopping Rules

Given that a simulation run may be divided into i.i.d. (or perhaps only approximately i.i.d.) tours, there remains the question of determining when to stop the simulation. Several stopping rules have been discussed in the literature, distinguishable either as fixed or sequential rules. A fixed rule ends the simulation after a fixed limit is reached, be it the simulated time, CPU time, or number of tours; a sequential rule halts the simulation after every K tours, evaluates a stopping criterion, and then, depending on whether the criterion is satisfied, either ends the simulation or continues it for yet another K tours. A typical stopping criterion inspects the relative width of the confidence interval about a particular point estimate. In other words, if the estimate of the parameter θ is given by $\hat{\theta}$ with a confidence interval defined by $(\hat{\theta}-\delta, \hat{\theta}+\delta)$, then the sequential stopping criterion demands that the relative width $2\delta/\hat{\theta}$ be within prescribed limits.

Lavenberg and Sauer [43] have conducted a variety of computer simulation experiments to assess the relative merits of fixed and sequential stopping rules. The disadvantage of a fixed stopping rule, of course, is that the final relative width of the estimated confidence interval is unknown upon starting the simulation. A sequential stopping rule allows the ultimate relative width to be specified initially, and, in their experiments, it was found that a relative width of 0.05 was small enough to yield valid confidence intervals. For larger relative widths and correspondingly smaller numbers of tours, however, the confidence intervals estimated with fixed stopping rules were valid more frequently than those obtained with sequential stopping rules. Sequential stopping rules, they conclude, have worse small sample behavior than fixed stopping rules. The authors advise a fixed stopping rule as appropriate if one wishes to obtain a valid confidence interval about a point estimate with an unspecified and possibly large relative width. To obtain precise valid estimates, a sequential stopping rule with a small relative width criterion is recommended.

In this simulation study, then, the relative width of the confidence interval about the estimated network throughput was monitored every $K=10$ tours and a width of 0.05 used as the criterion for the sequential stopping

rule. The choice of K was made arbitrarily; Lavenberg and Sauer affirm that different values of K have little effect on the results. A minimum of 40 tours was required before the simulation was allowed to stop in the hope that this minimum would allow the desired appearance of asymptotic properties of the employed ratio estimator. After 40 tours the sequential stopping rule was used, but with an additional fixed stopping rule imposing a maximum of either 2000 tours or 30 minutes of CPU time. These rules are summarized in Table 2.3.1, but variations were sometimes used to obtain results when, at high traffic levels, the simulation of a single tour consumed a relatively large amount of CPU time. In such cases, the parameter K and the minimum number of tours required before ending the simulation, for example, would be decreased so that computer time would not be spent simulating tours beyond that needed to meet the stopping criterion. The stopping criterion itself would, in the name of expediency, be relaxed; even so, it was often the limit on CPU time which finally halted such simulation runs.

Regenerative simulation:

- Regeneration point = Idle state
 - No activity on channel
 - No frames in any station buffer
 - Probe level = 0 for adaptive probing

Stopping rules:

Combination of sequential and fixed rules

1st 10 tours discarded for adaptive probing

Minimum 40 tours

Sequential stopping rule:

0.05 relative width of 90% confidence interval about estimated network throughput

Inspected every $K=10$ tours.

Fixed stopping rule:

Maximum 2000 tours

Maximum 30 minutes CPU time

Table 2.3.1 Simulation Techniques Used in This Study.

2.3.3 Ratio Estimation — The Jackknife Estimator

In simulation studies, many of the performance indicators that one wishes to estimate are defined as the ratio of two random variables, the denominator typically being some function of the simulation duration. Here, all the performance indicators — network throughput, mean waiting time, mean queue size, mean rejection rate, mean failure rate, mean number of attempts required for successful transmission, mean probe level, and system load — are defined as such ratios, as shown in Table 2.3.2. In general, then, for a sample (X_i, Y_i) ($i=1, \dots, n$) of paired random variables with $E[Y_i] = \eta$ and $E[X_i] = \mu$, one wishes to obtain a point estimate and confidence interval for the ratio η/μ .

The most obvious ratio estimator is the classical estimator, $\bar{Y}/\bar{X}$, where

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$$

and

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i.$$

This estimator, however, is generally biased; i.e., $\bar{Y}/\bar{X} \neq \eta/\mu$. Let $r = \bar{Y}/\bar{X}$ and $\rho = \eta/\mu$. Knowing that $E[\bar{Y}] = \eta$ and $E[\bar{X}] = \mu$ (in other words, knowing that $\bar{Y}$ and $\bar{X}$ are unbiased estimators of the means of their respective underlying distributions), then the following statements are true:

$$\rho = \frac{E[\bar{Y}\bar{X}]}{E[\bar{X}]}$$

and

$$E[r] = \frac{E[r]E[\bar{X}]}{E[\bar{X}]}$$

Consequently,

$$E[r] - \rho = \frac{1}{E[\bar{X}]} [E[r]E[\bar{X}] - E[r\bar{X}]]$$

or, equivalently,

$$E[r] = \rho - \frac{\text{cov}(r, \bar{X})}{E[\bar{X}]}.$$

The estimator $r = \bar{Y}/\bar{X}$ is thus only unbiased if r and $\bar{X}$ are uncorrelated.

Other statistical estimators for the ratio ρ/μ are known to reduce the bias of the classical estimator. Iglehart [41] has compared five ratio estimators in the simulations of stable stochastic systems using the regenerative method. The ratio estimators tested include the classical estimator, $\bar{Y}/\bar{X}$, and four others called the Fieller, Beale, Tin, and jackknife estimators. In comparing the performance of these estimators for simulations of an M/M/1 queue, a periodic review inventory model, and the classical repairman problem, Iglehart finds that, for short runs, the jackknife estimator provides the smallest bias and best confidence interval coverage of the estimated parameter, and, for long runs, the estimators are generally comparable. He concludes by recommending the jackknife estimator, especially when the number of tours used in the regenerative method is limited. In this study, as noted previously, the CPU time spent simulating each tour becomes exorbitant at high system load levels. For that reason, the jackknife estimator was chosen as the ratio estimator in these simulations.

The generalized jackknife estimate $G(\hat{\theta}_1, \hat{\theta}_2)$ be defined as

$$G(\hat{\theta}_1, \hat{\theta}_2) = \frac{\hat{\theta}_1 - R\hat{\theta}_2}{1 - R},$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are estimators of the parameter θ and R is any real number except 1 [30]. The generalized jackknife statistic is one of several robust statistical methods that do not depend on the assumption of normality yet possess many of such an assumption's desirable features. The generalized jackknife, in particular, provides an estimator with a reduced bias and allows some measure of the estimator's variance.

The estimator employed in this study is the jackknife estimator $J(\hat{\theta})$ introduced by Quenouille [32]. Let $\hat{\theta}$ be an estimate for θ defined on the random sample $X_1, X_2, \dots, X_n$. This sample is partitioned into N subsets, each of size M , so that $NM = n$. A new random sample is then obtained by arbitrarily deleting a subset of size M from the original sample. Let $\hat{\theta}_i$ designate the estimator $\hat{\theta}$ defined on the subsample arising when the i -th subset of size M is deleted. The jackknife estimator is then

$$J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^N J_i(\hat{\theta})$$

where

$$J_i(\hat{\theta}) = N\hat{\theta} - (N-1)\hat{\theta}_i, \quad i=1, \dots, N.$$

The jackknife estimator is thus of the form

$$J(\hat{\theta}) = N\hat{\theta} - (N-1)\bar{\hat{\theta}}_i,$$

where

$$\bar{\hat{\theta}}_i = \frac{1}{N} \sum_{i=1}^N \hat{\theta}_i,$$

revealing that $J(\hat{\theta})$ is a special case of the generalized jackknife where $R = (N-1)/N$, $\hat{\theta}_1 = \hat{\theta}$, and $\hat{\theta}_2 = \bar{\hat{\theta}}_i$.

The motivation which led to this definition of the jackknife estimator $J(\hat{\theta})$ can be seen when the estimate $\hat{\theta}$ is biased [30]. Let

$$E(\hat{\theta}) = \theta + \sum_{i=1}^{\infty} \frac{a_i}{n^i},$$

where, again, n is the number of samples and $\{a_i\}$ may be functions of θ but not of n . The average $\bar{\theta}_i$ of the M estimates formed on the subsamples remaining when the i -th subset of M samples is deleted will then have an expected value

$$E[\bar{\theta}_i] = \theta + \sum_{i=1}^{\infty} \frac{a_i}{(n-M)^i}.$$

From the definition of $J(\hat{\theta})$, its expectation is then

$$E[J(\hat{\theta})] = \theta - \frac{a_2}{n(n-M)} - \frac{a_3(2n-M)}{n^2(n-M)^2} - \dots$$

Assuming $a_1 = 0$, the bias of $J(\hat{\theta})$ is thus of order n^{-2} while that of $\hat{\theta}$ is of order n^{-1} . Furthermore, if $a_2 = a_3 = \dots = a_n \dots = 0$, then $J(\hat{\theta})$ is unbiased.

Durbin [33] has investigated Quenouille's jackknife estimator (with $M=2$) as a method of ratio estimation. Assuming that the regression of Y on X is linear and X is normally distributed, the jackknife not only reduces the bias but also the variance when compared to the classical ratio estimator, $\bar{Y}/\bar{X}$. When X has a gamma distribution, Durbin shows that, although the variance of the jackknife with $M=2$ is slightly increased compared to the variance of the classical estimator, the jackknife's reduction in bias is such that the mean square error of the jackknife is less than that of the classical estimator. Later, it has been shown that both the bias and variance of the jackknife estimator are decreasing functions of M when X has a normal [35] or gamma [36] distribution, thus suggesting that $M=n$ is the optimum choice for use with the jackknife.

While the preceding discussion reveals the usefulness of the jackknife estimator in reducing bias, the jackknife further possesses asymptotic properties which allow robust interval estimation. In an abstract, Tukey [34] proposed that the terms $\{J_i(\hat{\theta})\}$ could be treated as independent identically

distributed (i.i.d.) random variables, thus allowing one to obtain approximate confidence intervals for the jackknife. In subsequent unpublished work, Tukey called the terms $\{J_i(\hat{\theta})\}$ "pseudovalues" and, indeed, it was Tukey who coined the name "jackknife" for the form of Quenouille's estimator shown here [31].

Tukey's conjecture has been proven valid for a variety of general problems [31]. For example, Gray and Schucany [30] consider the case where the samples $X_1, \dots, X_n$ are from a distribution with mean μ and finite variance. The parameter θ is assumed to be a function of the mean; i.e., $\theta = f(\mu)$, where f is a real-valued function which possesses a bounded second derivative in a neighborhood of μ , and the estimator $\hat{\theta}$ is defined as $\hat{\theta} = f(\bar{X})$. They then show that, for the generalized jackknife $G(\hat{\theta}_1, \hat{\theta}_2)$, terms analogous to the pseudovalues of the jackknife are effectively i.i.d.

The consequence of assuming the pseudovalues $\{J_i(\hat{\theta})\}$ to be i.i.d. can be seen by the following reasoning, adapted from a discussion in Reference [37]. The jackknife estimator $J(\hat{\theta})$, defined as

$$J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^N J_i(\hat{\theta}),$$

is the sample mean of $\{J_i(\hat{\theta})\}$. The underlying distribution for $\{J_i(\hat{\theta})\}$ has mean θ and an assumed finite variance σ^2 . By the central limit theorem, then,

$$\lim_{N \rightarrow \infty} \Pr \left[\frac{\sqrt{N}(J(\hat{\theta}) - \theta)}{\sigma} \leq z \right] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-u^2/2} du = I(z), \quad -\infty \leq z \leq \infty.$$

Define $s^2(N)$ as

$$s^2(N) = \frac{1}{N-1} \sum_{i=1}^N (J_i(\hat{\theta}) - J(\hat{\theta}))^2,$$

so that $s^2(N)$ is an unbiased estimator of the variance σ^2 . By the strong law of large numbers, $s(N) \rightarrow \sigma$ with probability one, and, hence, $\sigma/s(N)$ converges in probability to one. Through a version of the continuous mapping theorem, it can then be shown that

$$\lim_{N \rightarrow \infty} \Pr \left[\frac{\sqrt{N(N-1)} (J(\hat{\theta}) - \theta)}{\sqrt{\sum_{i=1}^N (J_i(\hat{\theta}) - J(\hat{\theta}))^2}} \leq z \right] = \Phi(z) .$$

In other words, if the N pseudovalues $\{J_i(\hat{\theta})\}$ are assumed to be approximately i.i.d. random variables, then the statistic

$$\frac{\sqrt{N(N-1)} (J(\hat{\theta}) - \theta)}{\sqrt{\sum_{i=1}^N (J_i(\hat{\theta}) - J(\hat{\theta}))^2}} = \frac{\sqrt{N} (J(\hat{\theta}) - \theta)}{s(N)}$$

has an approximate t distribution with $N-1$ degrees of freedom.

If one defines $z_{\xi} = \Phi^{-1}(\xi)$, $0 < \xi < 1$, then, for large N ,

$$\Pr \left[-z_{1-\gamma/2} \leq \frac{\sqrt{N(N-1)} (J(\hat{\theta}) - \theta)}{\sqrt{\sum_{i=1}^N (J_i(\hat{\theta}) - J(\hat{\theta}))^2}} \leq z_{1-\gamma/2} \right] = 1-\gamma .$$

A $100(1-\gamma)$ percent approximate confidence interval about the point estimate $J(\hat{\theta})$ is thus given by

$$\left(J(\hat{\theta}) - \frac{s(N)z_{1-\gamma/2}}{\sqrt{N}}, J(\hat{\theta}) + \frac{s(N)z_{1-\gamma/2}}{\sqrt{N}} \right) .$$

In this study, 90% confidence intervals are employed so that $\gamma = .1$ and $z_{1-\gamma/2} = 1.645$. Table 2.3.3 summarizes the jackknife estimator as used here. Note that the estimator $\hat{\theta}$ is taken to be the classical ratio estimator, $\bar{y}/\bar{x}$, and that $N=n$, the total number of samples.

$$\text{Network throughput} = \frac{\text{Total time channel carries successful transmissions}}{\text{Total simulation time}}$$

$$\text{Mean waiting time} = \frac{\sum_{\text{All stations}} \sum_{\text{All successfully transmitted frames}} \text{Waiting time}}{\text{Total number of successful transmissions}}$$

$$\text{Mean queue size} = \frac{\sum_{\text{All stations}} \sum_{\text{Queue size } k} k \times \text{Elapsed time spent with queue size } = k}{\text{Total simulation time} \times \text{Number of stations}}$$

$$\text{Mean rejection rate} = \frac{\sum_{\text{All stations}} \text{Number of frames turned away from a full buffer}}{\text{Total simulation time} \times \text{Number of stations}}$$

$$\text{Mean failure rate} = \frac{\sum_{\text{All stations}} \text{Number of frames abandoned after attempt limit reached}}{\text{Total simulation time} \times \text{Number of stations}}$$

$$\text{Mean number of attempts required for successful transmission} = \frac{\sum_{\text{All stations}} \sum_{k \text{ attempts}} \text{Number of successful } k \times \text{transmissions requiring } k \text{ attempts}}{\text{Total simulation time} \times \text{Number of stations}}$$

$$\text{Mean probe level} = \frac{\sum_{\text{Probe level } k} k \times \text{Elapsed time network is at-probe level } k}{\text{Total simulation time}}$$

Table 2.3.2 Performance Indicators (Classical Estimators)

$$\text{Jackknife Estimator } J(\hat{\theta}) = \frac{1}{n} \sum_{i=1}^n J_i(\hat{\theta}) = n\hat{\theta} - (n-1)\hat{\theta}_1$$

Given paired samples (X_i, Y_i) , $i=1, \dots, n$, where $E[Y_i] = \eta$ and $E[X_i] = \mu$, the jackknife estimator $J(\hat{\theta})$ provides an estimate of the ratio $\theta = \eta/\mu$ biased as n^{-2}

Let $\hat{\theta} = \sum_{i=1}^n Y_i / \sum_{i=1}^n X_i = \bar{Y}/\bar{X}$ = the classical ratio estimator

and $\hat{\theta}_1 = \sum_{j=1}^n Y_j / \sum_{j=1}^n X_j$, then

$$J\left(\frac{\bar{Y}}{\bar{X}}\right) = n \left[\frac{\sum_{i=1}^n Y_i}{\sum_{i=1}^n X_i} \right] - (n-1) \sum_{j=1}^n \left[\frac{Y_j}{X_j} \right]$$

Define the pseudovalues $J_i(\hat{\theta})$ as $J_i(\hat{\theta}) = n\hat{\theta} - (n-1)\hat{\theta}_i$

$$\text{Let } s^2(n) = \frac{1}{n-1} \sum_{i=1}^n (J_i(\hat{\theta}) - J(\hat{\theta}))^2$$

90% confidence interval about $J(\hat{\theta})$:

$$\left(J(\hat{\theta}) - \frac{1.645 s(n)}{\sqrt{n}}, J(\hat{\theta}) + \frac{1.645 s(n)}{\sqrt{n}} \right)$$

Table 2.3.3 Ratio Estimation with the Jackknife Estimator

2.4 A Comparison of Adaptive Probing and CSMA's Contention Resolution Efficiencies

Given the description of the Ethernet and decentralized adaptive probing access control protocols, one wishes to compare their relative performance and anticipate the simulation results. The access control algorithms described in Sections 2.2.1 and 2.2.2, however, discourage analysis. Consider instead algorithms that have been well examined in the research literature: non-persistent carrier sense multiple access (CSMA) and adaptive probing. As one possible basis for comparison, the efficiency of the two algorithms in resolving contention may be investigated under controlled conditions. Let each of $N = 2^j$ stations have, at some initial point in time, a probability p of possessing a single message to be transmitted. Messages are assumed to be of fixed duration m , measured in units of slot times, a single message possible per station. One wishes to determine how long it will take the two schemes to successfully transmit all the system's messages, assuming further message generation is suspended during the process.

The adaptive probing technique is here used in the centralized random access context where an inquiry by the central processor is answered by either silence, a successful message transmission, or a garbled transmission resulting from a conflict between two or more users. Level-skipping, as described in Section 2.2.2, is employed. Each inquiry requires one slot, the inquiries effecting a binary search of, for example, 2^j users. It has been shown by J. F. Hayes, A. Grami, and K. Sohraby [45] that a recursive relationship for the generating function $Q_j(z)$ of a random access probing cycle of a tree of 2^j users is given by

$$Q_j(z) = zQ_{j-1}^2(z) + (z-z^2) \left[N_{0j-1}^2 + N_{0j-1} Q_{j-1}(z) \right] \\ + N_{0j-1} N_{1j-1} (2z - z^2 - z^3)z^m ;$$

where

2^j = the number of users in a group,

$$N_{0j} = \text{Pr}[\text{no messages in } 2^j \text{ stations}] = (1-p)^{2^j}.$$

$$N_{1j} = \text{Pr}[\text{one message among } 2^j \text{ stations}] = 2^j p (1-p)^{2^j-1}.$$

m = the number of slots/messages ;

and

$$Q_1(z) = (1-p)^2 z + 2p(1-p)z^{m+1} + p^2 z^{2m+3},$$

where

p = Pr[a station has a message].

From the above it is a simple matter to find the mean and mean-square of the time needed to transmit all messages:

$$E[\text{cycle}_j] = \left. \frac{d}{dz} Q_j(z) \right|_{z=1} = 1 + 2Q'_{j-1}(1) + N_{0j-1} [N_{0j-1} + 3N_{1j-1} + 1].$$

with

$$E[\text{cycle}_{j=1}] = \left. \frac{d}{dz} Q_1(z) \right|_{z=1} = 1 + 2p^2 + 2mp$$

and

$$\begin{aligned}
 E[\text{cycle}_j^2] - E[\text{cycle}_j] &= \frac{d^2}{dz^2} Q_j(z) \Big|_{z=1} \\
 &= 4Q'_{j-1}(1) + 2Q''_{j-1}(1) + 2[Q'_{j-1}(1)]^2 - 2[N_{0j-1}^2 + N_{0j-1}] \\
 &\quad - 2 N_{0j-1} Q'_{j-1}(1) - 8 N_{0j-1} N_{1j-1} - 6 N_{0j-1} N_{1j-1} m
 \end{aligned}$$

with

$$\begin{aligned}
 E[\text{cycle}_1^2] - E[\text{cycle}_1] &= \frac{d^2}{dz^2} Q_1(z) \Big|_{z=1} \\
 &= 2m^2 p^2 + 8p^2 m + 2pm^2 + 2pm + 6p^2.
 \end{aligned}$$

The probing technique is adaptive in the sense that the initial 2^n users may be divided into 2^{n-k} groups of 2^k , each group to be probed separately. The number of groups is chosen so as to minimize the duration of a cycle. The mean total cycle time is thus $2^{n-k} Q'_k(1)$. To minimize the mean cycle time, one chooses $k^* = \max\{0, 1, \dots, j, \dots, n\}$ such that

$$N_{0j} + N_{0j-1} + 3N_{0j-1} N_{1j-1} - 1 > 0.$$

Note that $k^* = 0$ implies, as a result of the relatively high probability that each station has a message, that conventional polling, sequentially querying each user, will minimize the mean cycle time. Adaptive probing

here reverts to polling whenever the probability p that a user has a message exceeds $1/\sqrt{2} = .707$.

In contrast to the systematic procedure of adaptive probing, the decentralized non-deterministic scheme of slotted non-persistent CSMA operates in the following fashion:

- (1) at the beginning of each time slot, stations with messages ready to transmit sense the channel and transmit if it is sensed idle;
- (2) if the channel is sensed busy or if a transmission conflict has occurred, then the user, now backlogged, schedules retransmission according to the retransmission delay distribution.

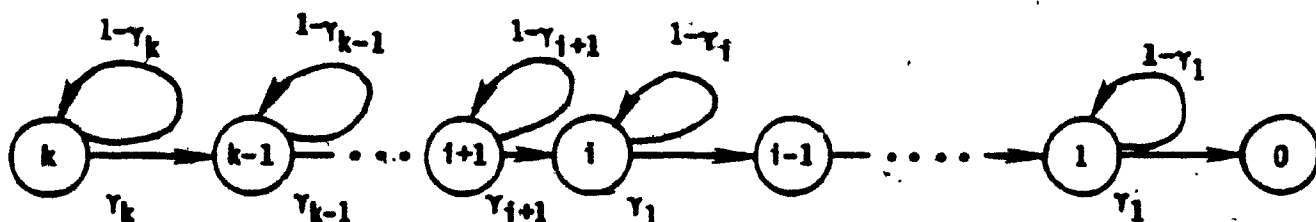
Assume here that the retransmission delay is geometrically distributed with mean $1/v$ slots; i.e., each backlogged user senses the channel with probability v . Messages are of fixed length m slots, where one slot, being the length of a probing inquiry, is assumed to be at least as long as the maximum propagation delay τ between stations. In accordance with the probing analysis, assume that the message length m incorporates τ and thus in itself represents the full amount of time that the channel is sensed busy after a transmission. No collision detect and abort capability is assumed — conflict between two or more users lasts one message length — yet the acknowledgement indicating a foiled transmission is assumed to arrive in a negligible amount of time.

Although CSMA is an ongoing process, with each idle station having a probability of generating a new message at every slot, for the purposes of this comparison one considers here just the operation of the CSMA scheme in clearing the messages initially present among the N users. If only one station has a message, then its transmission is immediate and successful. If $k > 1$ stations have a message, then all k transmit and collide in the first m slots. All k stations are then backlogged and must resolve their conflict through the random retransmission policy.

Let γ_i be the probability of a successful message transmission given that i stations are backlogged. From the above description, γ_i must equal the probability that only one of the i backlogged stations senses the channel:

$$\gamma_i = i v (1-v)^{i-1}.$$

The resolution process is represented by the accompanying transition diagram:



The time spent in each state of the transition diagram is geometrically distributed and statistically independent of the time spent in other states; the probability of $(n-1)$ collisions, each of length m slots, finally followed by a successful transmission, given that there are i backlogged stations, is $\gamma_i (1-\gamma_i)^{n-1}$. The generating function of the number of slots spent in state i is thus

$$\sum_{n=1}^{\infty} \gamma_i (1-\gamma_i)^{n-1} z^{nm} = \frac{\gamma_i z^m}{1 - (1-\gamma_i) z^m}.$$

The generating function $G_k(z)$ of the number of slots required until all k packets are transmitted without collision is simply the product of the above generating functions:

$$G_k(z) = \prod_{i=1}^k \frac{\gamma_i z^m}{1 - (1-\gamma_i) z^m} = z^{km} \sum_{i=1}^k \frac{\gamma_i}{1 - (1-\gamma_i) z^m}.$$

From $G_k(z)$, one can obtain $E[T|k] = G'_k(1)$, the mean number of slots required given k stations are backlogged:

$$E[T|k] = km + m \sum_{j=1}^k \frac{1-\gamma_j}{\gamma_j} ; \quad k \geq 2 .$$

Similarly, $E[T^2|k] - E[T|k] = G''_k(1)$ is given by the expression

$$\begin{aligned} E[T^2|k] - E[T|k] &= km(km-1) + 2km^2 \sum_{i=1}^k \frac{1-\gamma_i}{\gamma_i} \\ &+ \sum_{i=1}^k \frac{\gamma_i(1-\gamma_i)m(m-1) + 2m^2(1-\gamma_i)^2}{\gamma_i^2} + \sum_{i=1}^k \frac{(1-\gamma_i)}{\gamma_i} \sum_{\substack{j=1 \\ j \neq i}}^k \frac{(1-\gamma_j)}{\gamma_j} m^2 ; \quad k \geq 2 . \end{aligned}$$

With p equal to the probability that each station has a message, the probability that k of N stations have a message is given by

$$\pi_k = \binom{N}{k} p^k (1-p)^{N-k} .$$

From the discussion of the CSMA scheme, it is clear that the mean cycle time $E[T]$ (in slots) required to clear the system is

$$E[T] = 1 - \pi_0 m + \sum_{k=2}^N E[T|k] \pi_k ,$$

and the variance of the cycle time is given by

$$\text{Var}(T) = (1 - \pi_0)\pi_0 m^2 + \sum_{k=2}^N \text{Var}(T|k)\pi_k.$$

The mean and variance of the cycle time depend on the value of ν , the probability that a backlogged user senses the channel. Just as the adaptive probing algorithm takes advantage of knowledge of p , the probability that a station initially has a message, to minimize the mean probing cycle time, so too can the mean CSMA cycle time be minimized by judicious selection of the parameter ν , as happens, for example, in Ethernet's exponential backoff algorithm. While an analytic expression for such an optimum value of ν^* is not readily obtained, it is easy to see that ν^* hinges on the value of p as well as on N and m . To make the desired comparison with adaptive probing, then, ν was chosen as that value minimizing $E[T]$ for $\nu = .0025, .005, .0075, \dots$. This coarse tuning is seen reflected in the irregularities of the resulting curves for CSMA performance.

The results obtained from this comparison of random retransmission (CSMA) and adaptive probing are exhibited in Figures 2.4.1-3, showing the mean number of slots required to clear the N terminals as a function of p , the probability of each of the N terminals initially having a message. For fixed values of p , the probability of more than one message in the system increases as the number of stations increases. The differences in the curves for $N=32$ and $N=128$ are thus due to the differences in effective system loads.

Figures 2.4.1 and 2.4.2 compare CSMA versus adaptive probing when the message length is equal to one slot. When the system is lightly loaded; i.e., for low levels of p , CSMA requires less time to clear the system of messages than does adaptive probing. In CSMA, only those of the total N stations with messages ready to transmit are in competition for the channel while the centralized adaptive probing algorithm may waste inquiries of length one slot on groups of stations that have no messages whatsoever. Indeed, Figure 2.4.1 shows the mean time to clear for adaptive probing

approaching an asymptote of one slot for low values of p . This behavior is a direct result of the analysis, which assumes a centralized form of adaptive probing requiring at least one slot time to make the initial inquiry determining whether there are any messages in the system at all. If one assumes that contention resolution begins only after the need has been established, then the mean clear time for adaptive probing derived previously may be reduced by the probability that zero or more than two messages are in the system. The resulting curves are shown in Figure 2.4.2 against the same CSMA curves as in Figure 2.4.1. Some advantage to CSMA at low values of p is still seen in the curves for networks of 32 stations, but no advantage exists for networks of 128 stations at the same values of p since, as commented previously, the effective system load is greater.

Figure 2.4.3 shows the comparative performance of adaptive probing and CSMA in clearing 32 and 128 stations when the message length is 100 slots. Neither scheme is here considered to possess a collision detect and abort capability, so both algorithms suffer a full message length delay when a collision occurs. The overhead of centralized adaptive probing — the one slot inquiry time — is thus relatively negligible. The curves show the adaptive probing algorithm to be superior to CSMA for all values of p . At higher values of p this result is to be expected since, while CSMA requires an ever-dwindling number of stations to compete probabilistically for the channel until only one finally remains, the mean time to clear for adaptive probing is always less than or equal to that required by the deterministic system of conventional probing.

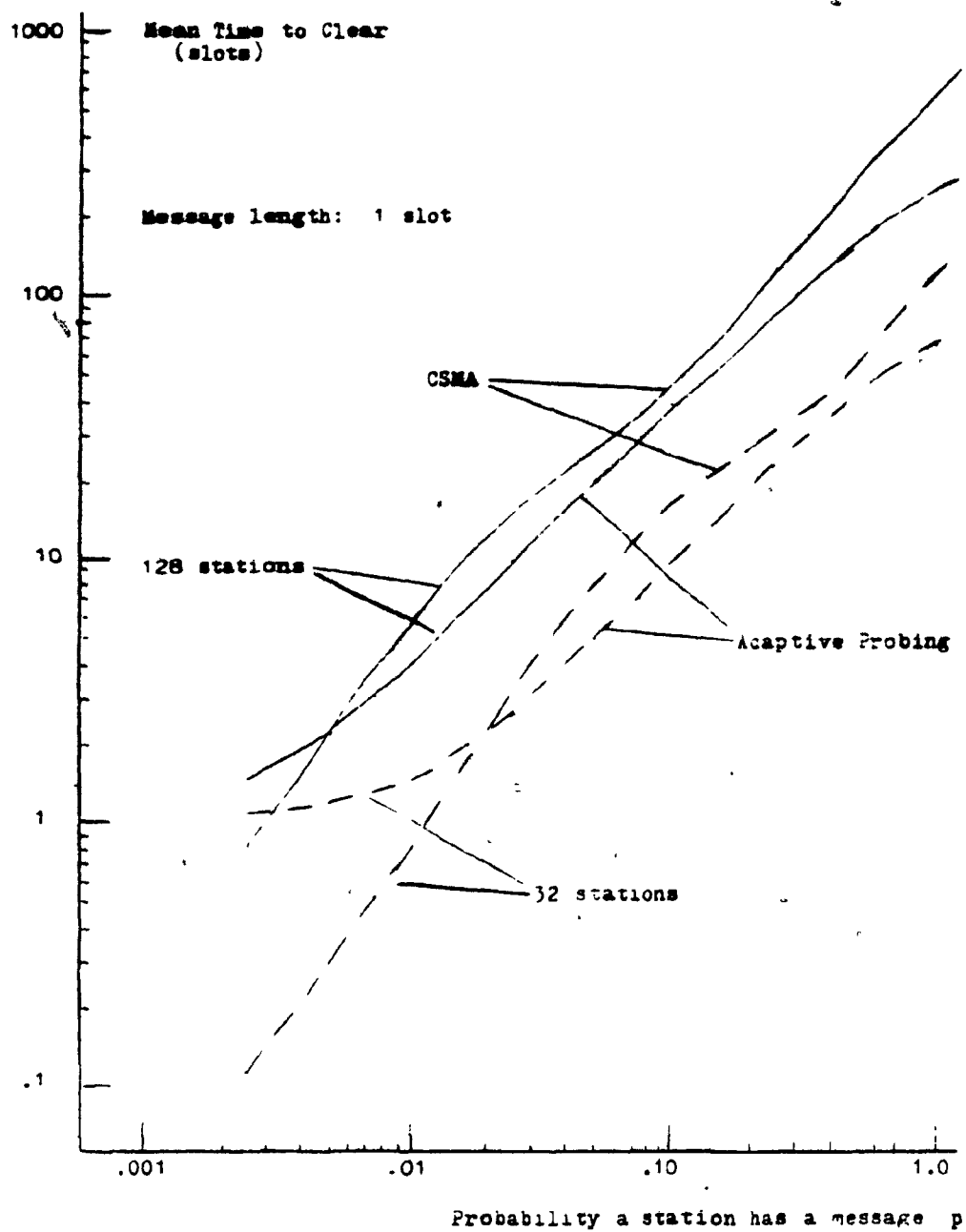


Figure 2.4.1 Mean Time to Clear versus the Probability a Station Has a Message

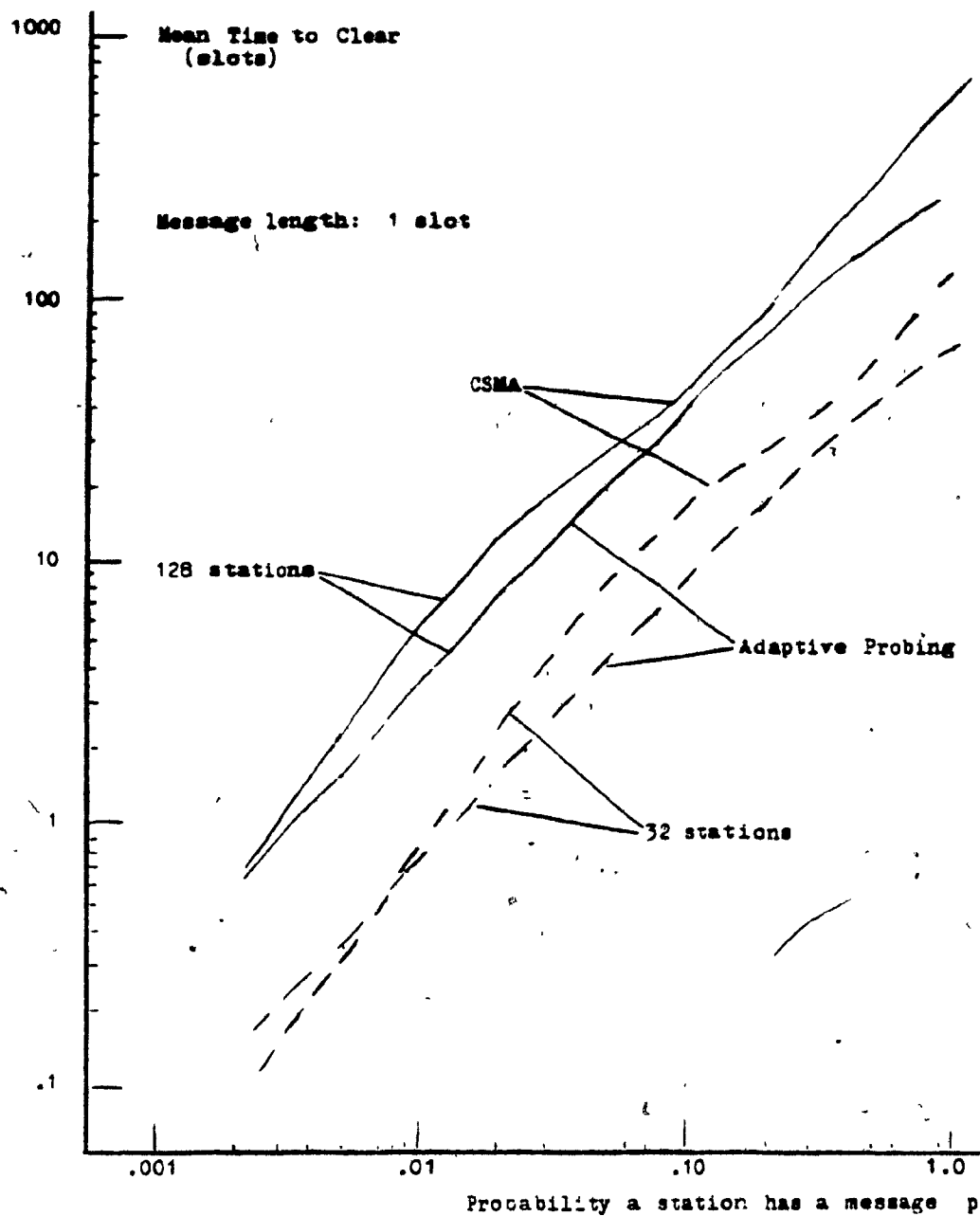


Figure 2.4.2 Mean Time to Clear versus the Probability a Station Has a Message

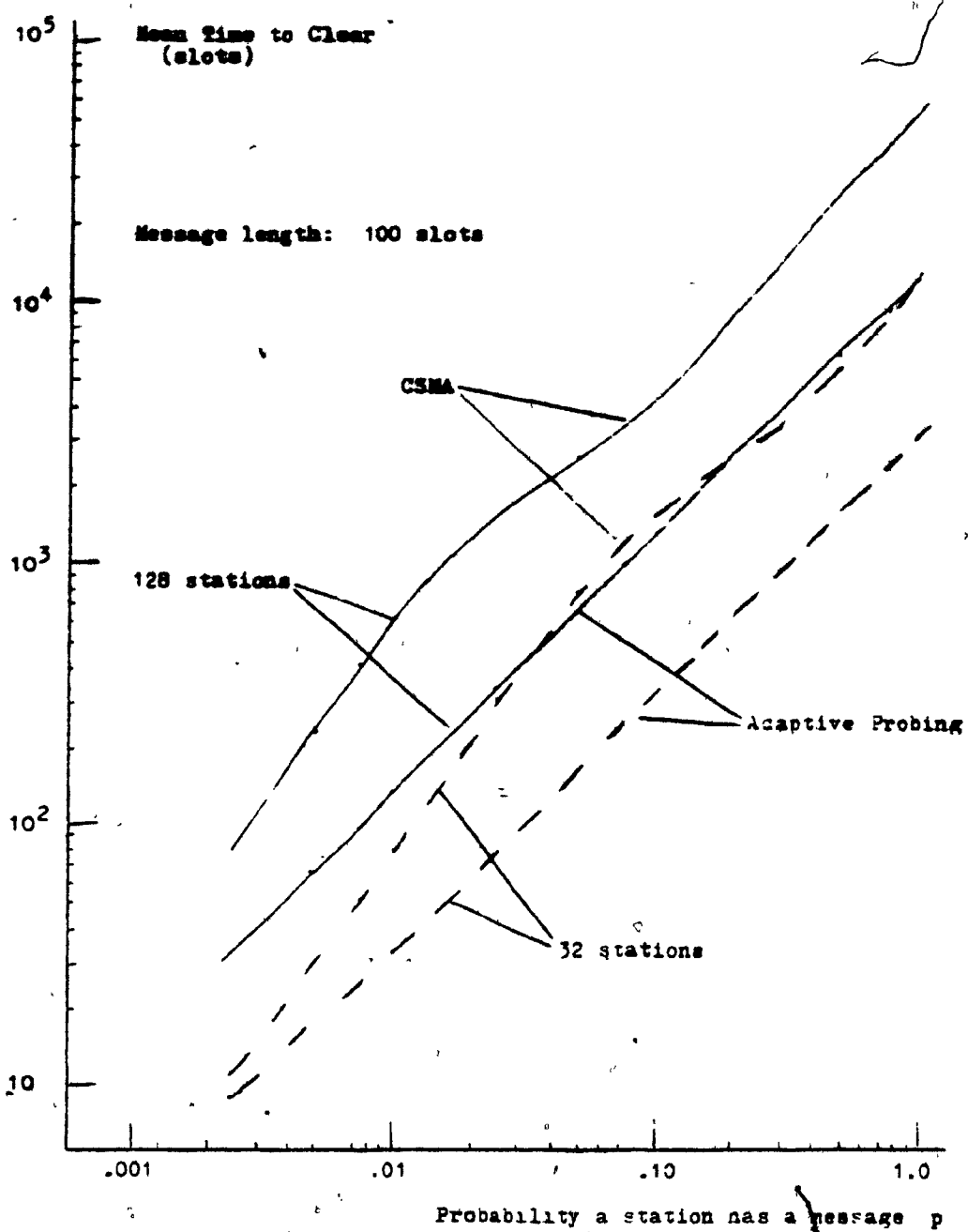


Figure 2.4.3 Mean Time to Clear versus the Probability a Station Has a Message

2.5 Simulation Results

The simulation programs modelling Ethernet and decentralized adaptive probing serve to provide data on the performance of these access protocols. Performance is assessed through various indicators considered as functions of ρ , where ρ is known as the traffic intensity, load, or utilization factor. Kleinrock [2] defines ρ as the utilization factor, "the ratio of the rate at which 'work' enters the system to the maximum rate (capacity) at which the system can perform this work." The local area networks considered here all hinge on many users sharing a common bus, effectively comprising a single-server system. As Kleinrock notes, for single-server systems the utilization factor defined above is equal to the traffic intensity of the system, the rate at which work enters the system. Indeed, he concludes that for G/G/1 queueing systems; i.e., single-server systems with independent and completely arbitrary interarrival time and service time distributions, the utilization factor ρ is the fraction of time the server is busy. The parameter ρ thus acts as a fundamental independent variable quantifying the demands made on the capability of a local area network. The load ρ — "load" being the preferred term throughout this study — is calculated as the product of the arrival rate to each station (λ , in frames/second/station), the number of stations, and the mean frame size (in seconds, including the 64-bit preamble appended to all frames).

Parameterized by the type of access protocol, the number of stations, the message-length distribution, and the buffer size of each station, simulation statistics were gathered on the following performance indicators as functions of ρ :

- network throughput;
- mean waiting time;
- mean queue size;
- mean rejection rate;
- mean failure rate;
- mean number of attempts required for successful transmission; and,
- mean probe level.

Note that symmetric 90% confidence intervals about the data points appear asymmetric due to the logarithmic scale of the y-axis. Also, confidence intervals on ρ appear for some data points as a result of difficulties in the regenerative simulation of adaptive probing at high loads, as discussed in Section 2.3.1.

2.5.1 Network Throughput

Network throughput is here defined as the fraction of time which the system is successfully transmitting data, including all overhead information. Data throughput, on the other hand, does not include the time spent in transmitting overhead information and thus reflects only that fraction of the channel capacity used to transmit the ultimately desired data. Overhead in the sense used here refers to those portions of each data frame required by the various protocol layers for their proper operation. No attempt, of course, was made in this study to model anything but the media access techniques of Ethernet and adaptive probing, so the dependence of data throughput on higher layer protocols is not evident. Indeed, overhead in the form of additional channel traffic may be generated by higher layers as, for example, when the Data Link Layer of a particular station transmits a frame to acknowledge a correctly received packet. The overhead in each data frame required solely for the multiaccess techniques of Ethernet and adaptive probing is modest: a 64-bit preamble for synchronization purposes, and, in the case of decentralized adaptive probing, an address field, prefix pointer, and current probe level word. Source and destination addresses, the cyclic redundancy check code used for error-checking, frame delimiters, etc. all represent overhead associated with other protocol layers.

Shoch and Hupp's study [3] of an operational Ethernet local area network provides some basis for comparison with the model used here. In their system, a 550-meter cable supported over 120 stations generating traffic typical of that particular network's operations. If one equates network throughput with load — a reasonable approximation supported by the results obtained in this study — then Shoch and Hupp's measurements of the channel utilization indicate that, averaged over a full twenty-four hours, system loads ranged from .006 to .0084. In the busiest second, a channel utilization/load of .37 was observed. Study of a local area network over a full range of loads is thus reasonable, since one can imagine instantaneous loads even exceeding unity. As part of Shoch and Hupp's work, traffic levels were artificially raised to extremely high levels; the observed throughput remained at capacity levels for loads greater than unity, indicating the stability of the Ethernet algorithm. They reported throughputs approaching

.96-.98, greater than any analysis of CSMA-CD has suggested possible. They did not, however, investigate the effects of such high loads on other performance aspects; e.g., the delays suffered by individual data packets.

Since the systems modelled here assume an interframe gap of $9.6 \mu\text{s}$ between each frame (as seen by the transmitting station), the maximum network throughput is simply the ratio of the maximum frame length, including the 64-bit preamble appended to all frames, to the maximum frame length plus the interframe gap. Two different message length distributions were used in this study: a fixed frame size of 6072 bits and a bimodal frame size distribution where 20% of all messages have the maximum allowed length of 12144 bits and the remainder have the minimum allowed length of 512 bits. This bimodal distribution was chosen to approximate typical Ethernet traffic characteristics [3] where most packets are short ones containing terminal traffic, acknowledgements, etc., and the remaining traffic comprises long packets such as would be required by file transfers. At the assumed transmission rate of 10 Mbps, the maximum network throughput possible with the fixed frame format is .985; the maximum throughput for the bimodal frame size distribution is .968. These values represent maximum achievable throughputs; clearly, contention between stations for access to the channel will tend to prevent the achievement of these maximums.

As seen in the accompanying graphs, Figures 2.5.1.a-d, network throughput for both adaptive probing and Ethernet corresponds directly to the system load up until a load of approximately .5 is reached. At loads greater than .5, the network throughput begins to lag slightly behind the offered load ρ , with the maximum observed throughputs approaching the limits suggested in the previous paragraph. The observed throughput can, of course, never reach the values cited above. In the case of Ethernet, some channel capacity is necessarily wasted as transmissions collide. For adaptive probing, even if it is operating at its highest probe level and effectively polling all stations, there will be gaps exceeding $9.6 \mu\text{s}$ between frames due to the propagation delays between stations. Given that the propagation delays are so small, adaptive probing possesses a maximum network throughput greater than that of Ethernet, but, as seen in these results, the Ethernet contention resolution mechanism is so efficient that the difference in

maximum throughput between the two access control protocols is insignificant. Comparison between the two protocols must therefore be made on the basis of other performance indicators.

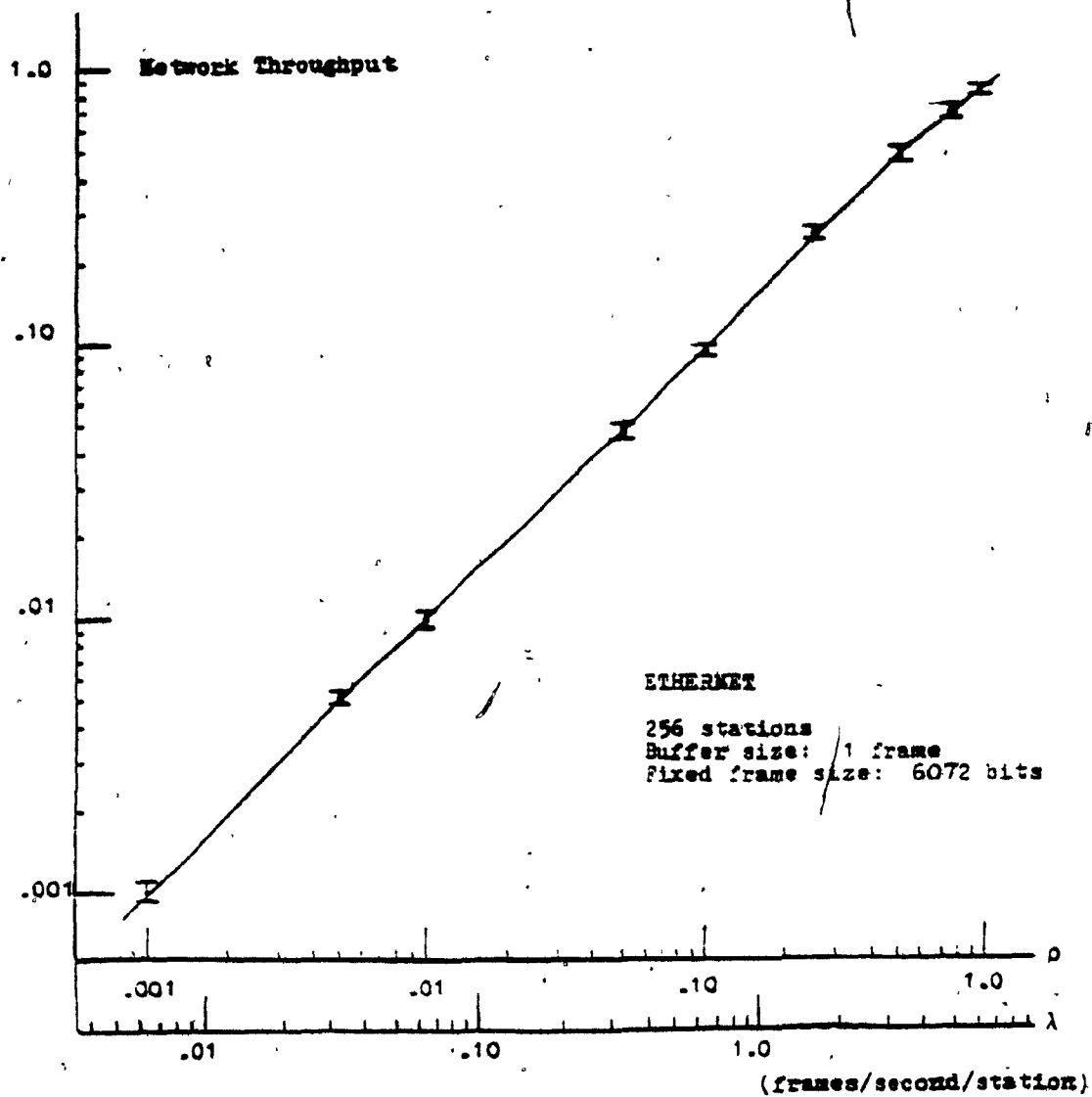


Figure 2.5.1.a Network Throughput versus Load ρ and Arrival Rate λ

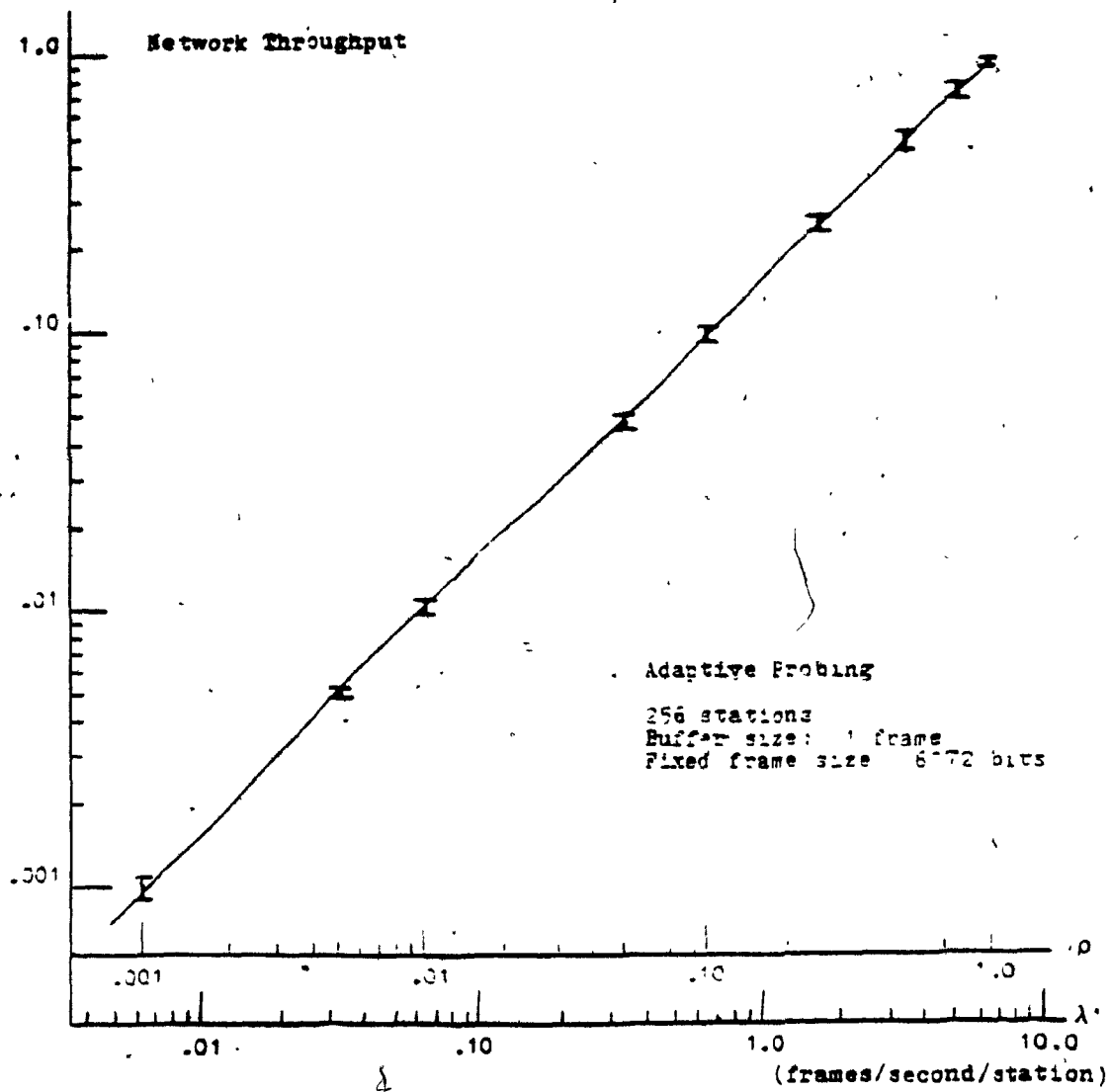


Figure 2.5.1.b Network Throughput versus Load ρ and Arrival Rate λ

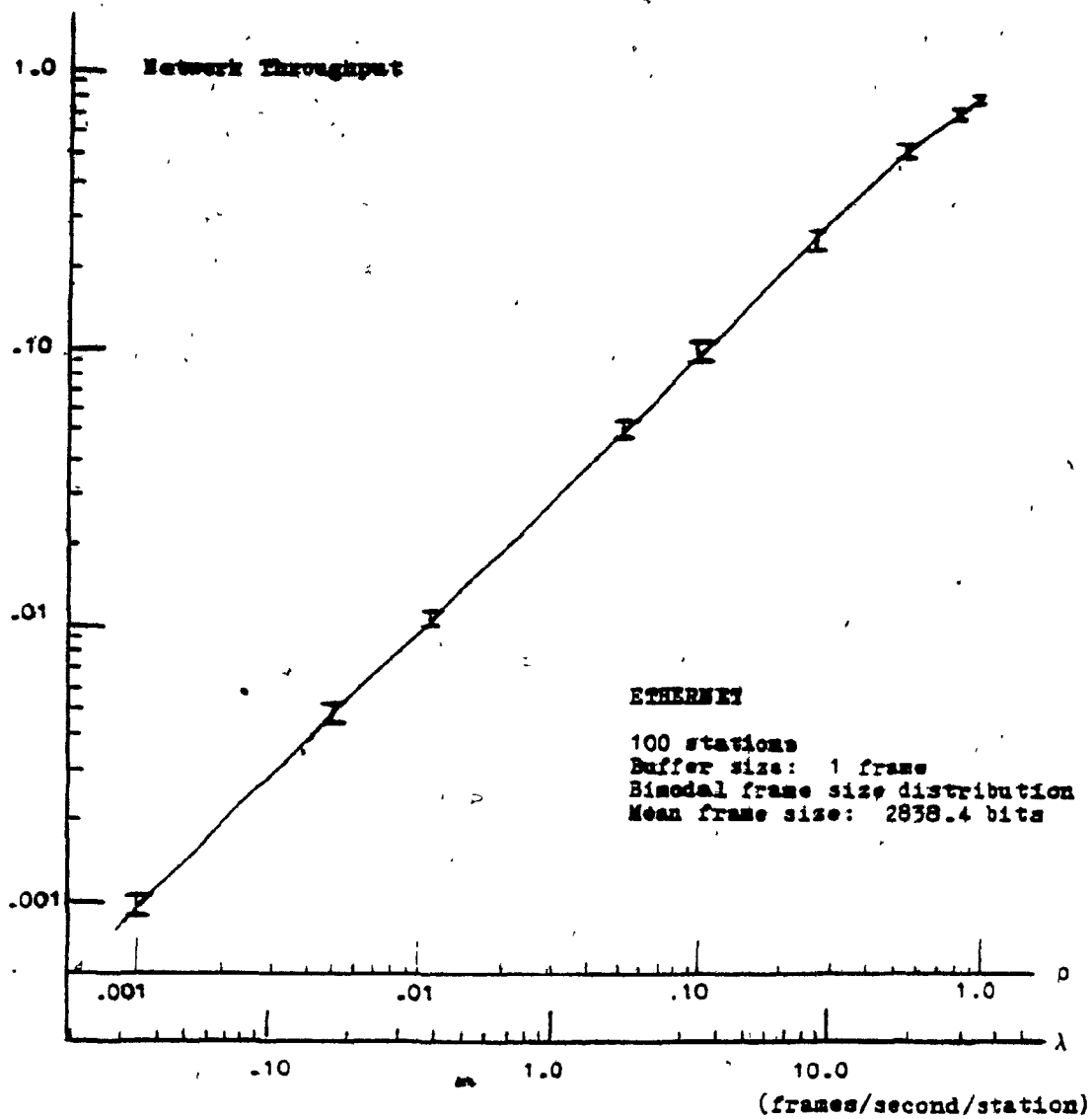


Figure 2.5.1.c Network Throughput versus Load ρ and Arrival Rate λ

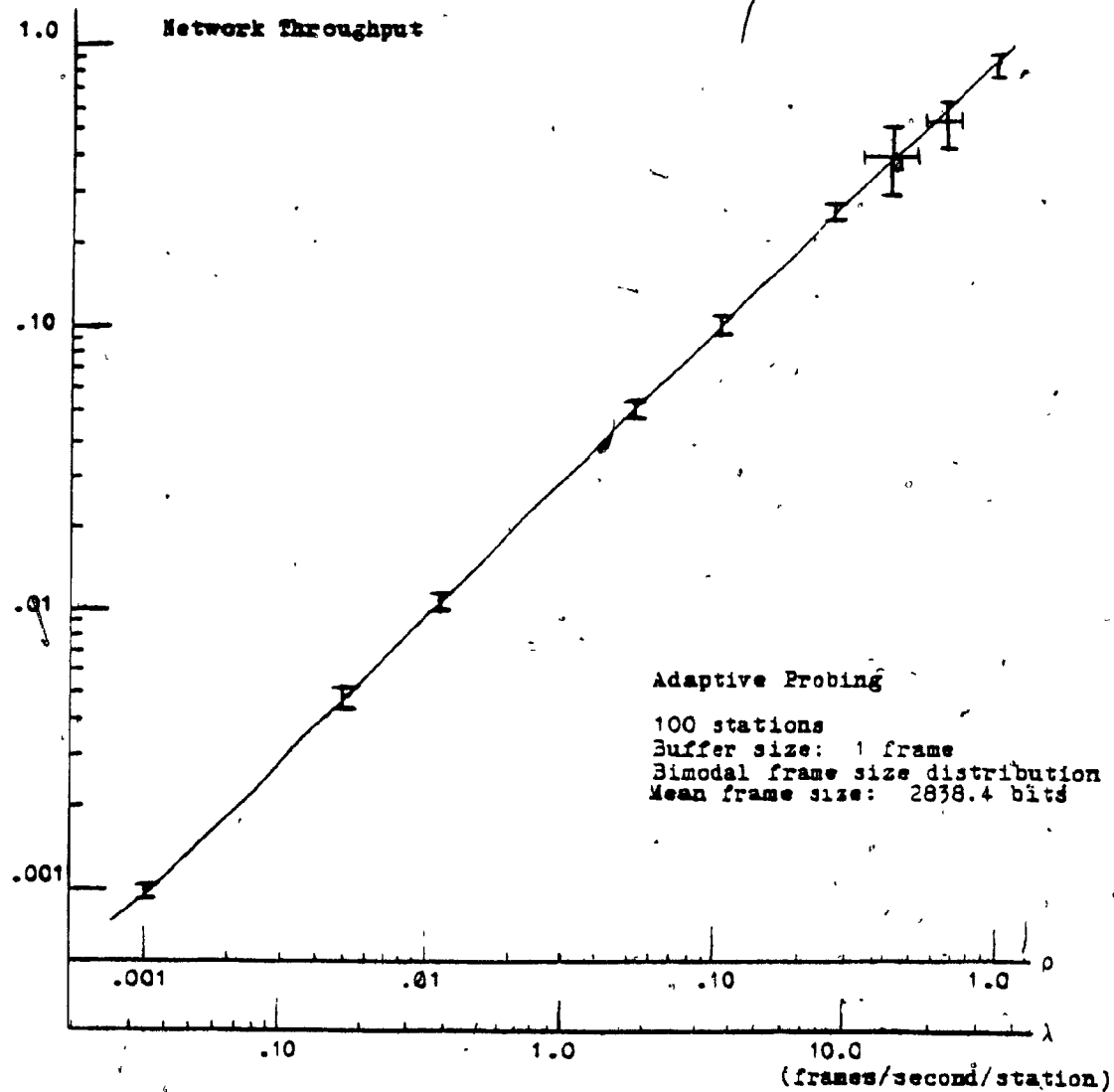


Figure 2.5.1.d Network Throughput versus Load ρ and Arrival Rate λ

2.5.2 Mean Waiting Time

Besides throughput, perhaps the most important indicator of an access protocol's performance is the mean delay a frame incurs in the process of being successfully transmitted. At high loads, individual frames suffer relatively greater delays in return for high system throughput. Excessive delays may limit the utility of the network and impede the operation of individual users. The transmission of digitized voice packets, for example, cannot be successfully supported when lengthy delays prohibit real-time re-assembly of the packets.

The mean frame delay is defined as the sum of the mean frame duration and the mean waiting time, where the waiting time is the interval between a frame's arrival and its successful departure from the originating station's buffer. Only the term dependent on the system load; i.e., the mean waiting time, is shown here.

The simulation results of Figures 2.5.2.a-h show little difference in the delay performance between Ethernet and adaptive probing, except, perhaps, at the very highest load levels, $\rho=.95$. At the lowest loads, of course, decentralized adaptive probing and Ethernet operate identically until a collision occurs and the two schemes' different contention resolution mechanisms go into effect. The simple analysis of Section 2.3.3 suggests that adaptive probing should impose lesser delays than Ethernet, although that model is inexact and does not even consider the effect of collision detection. Apparently, the parameters of Ethernet's collision detection procedure and the retransmission scheduling algorithm are sufficiently well-tuned to provide delay performance comparable to adaptive probing. At the highest load, adaptive probing is most likely to poll all stations or all pairs of stations. The resulting overhead incurred accounts for adaptive probing's greater delay compared to Ethernet. This polling overhead is, of course, exacerbated when the number of stations is not a power of two: time is wasted as non-existent stations are polled for messages.

The use of level-dependent scheduling delays in adaptive probing represents an attempt to mitigate the effects of polling overhead at high loads. By using their addresses to group those stations in close proximity to one another, the scheduling delays required by the decentralized adaptive probing algorithm may be reduced as the binary search progresses. The potential advantage, however, is not realized because of the requirement that a gap of $9.6 \mu s$ be maintained between the transmission of data frames. In decentralized adaptive probing, all stations with packets to transmit schedule their transmissions according to the adaptive probing algorithm immediately upon sensing the channel idle. Those stations in the next probe group ready to transmit will wait $9.6 \mu s$ before attempting transmission; all other ready stations schedule their transmissions according to the probing cycle. To insure no conflict with those stations transmitting in $9.6 \mu s$, the other ready stations must wait $9.8 \mu s$ (the interframe gap plus the two bit times required for a station to sense a signal following the onset of carrier presence information) in addition to the adaptive probing scheduling delays. The scheduling delays are derived from the local area network topology, as discussed in Section 2.2.3, and are on the order of microseconds or tens of microseconds. The initially required $9.8 \mu s$ delay dominates all others and thereby limits performance. The curves obtained from simulation support this conclusion (Figures 2.5.2.c and 2.5.2.h), showing no discernible advantage in using level-dependent scheduling delays. A practical requirement — the $9.6 \mu s$ interframe gap — thus foils this attempt to improve adaptive probing delay performance.

Intuitively, one expects a frame's mean waiting time to be sensitive to the buffer size of each station: as longer queues are allowed by a greater buffer capacity, the waiting times in those queues should consequently increase. This intuition is corroborated by the simulation results obtained for a 100-station single cable system where, for high loads ($\rho \geq .5$), mean waiting times do increase with buffer size. No change is evident at these loads for buffer sizes greater than three, reflecting the ability of the access protocol to efficiently service the station buffers. Similarly, the waiting time is the same for all buffer sizes at loads less than .5: buffer sizes greater than one at these arrival rates provide excess capacity with little impact on performance.

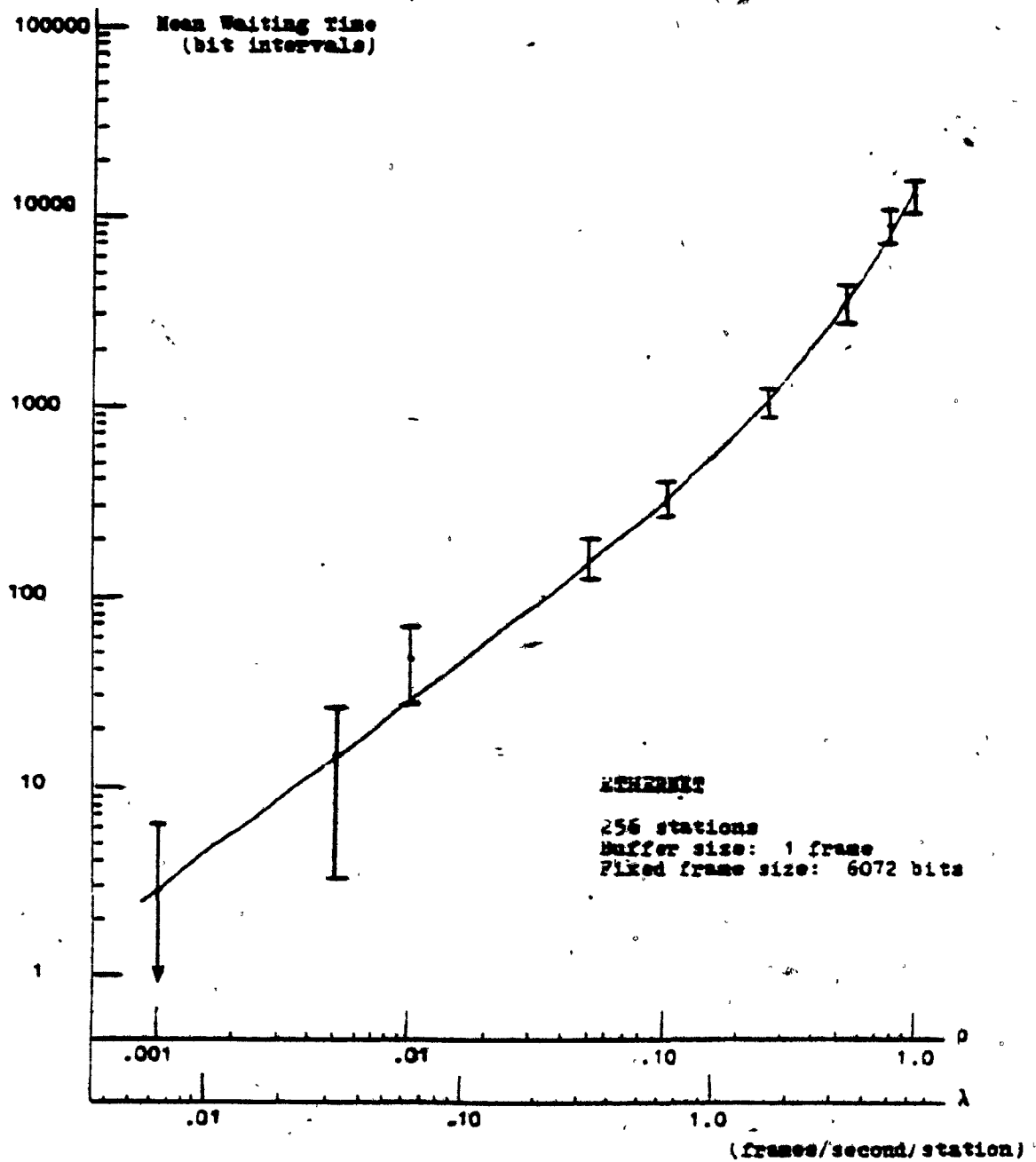


Figure 2.5.2.a Mean Waiting Time versus Load ρ and Arrival Rate λ

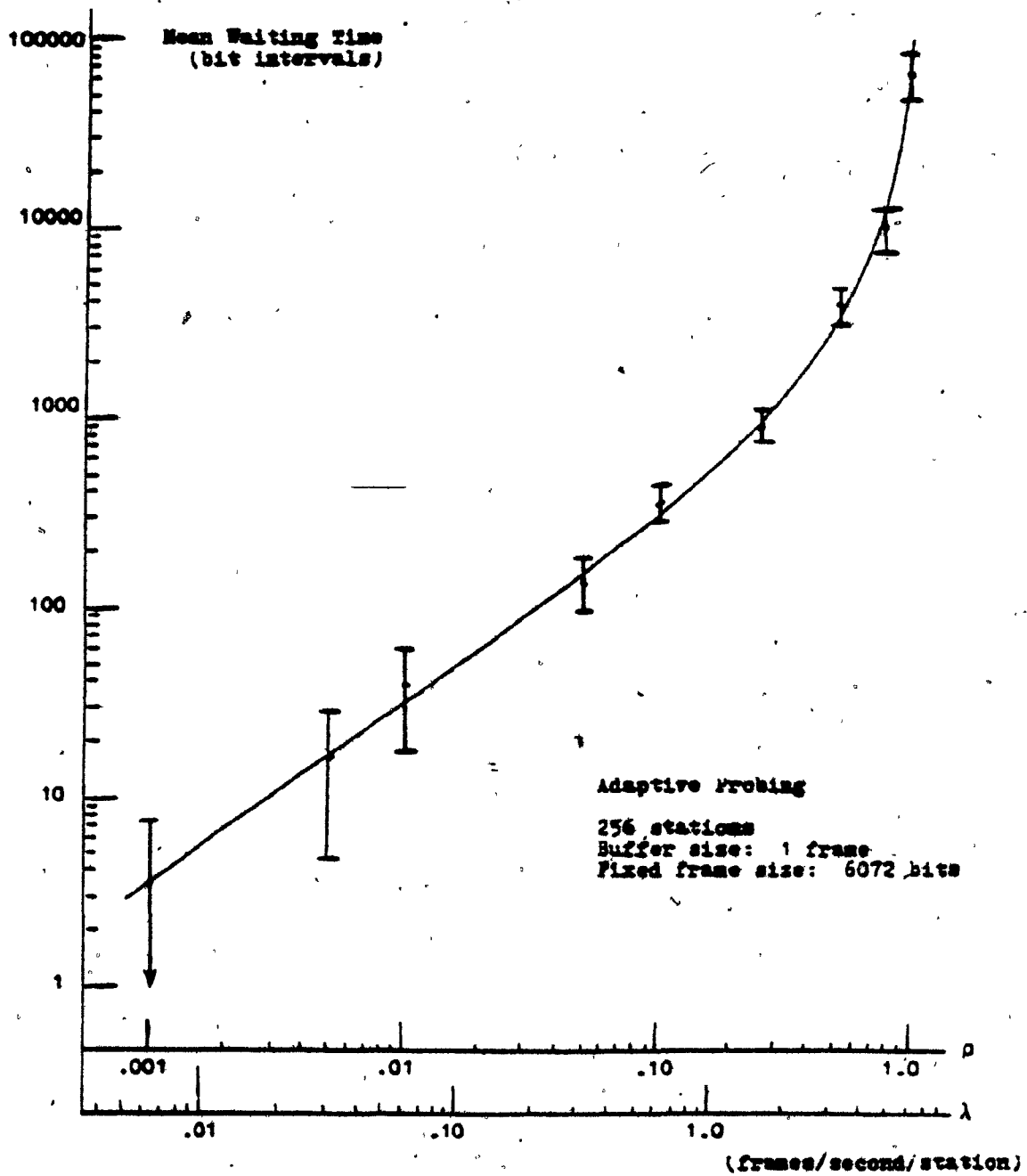


Figure 2.5.2.b Mean Waiting Time versus Load ρ and Arrival Rate λ

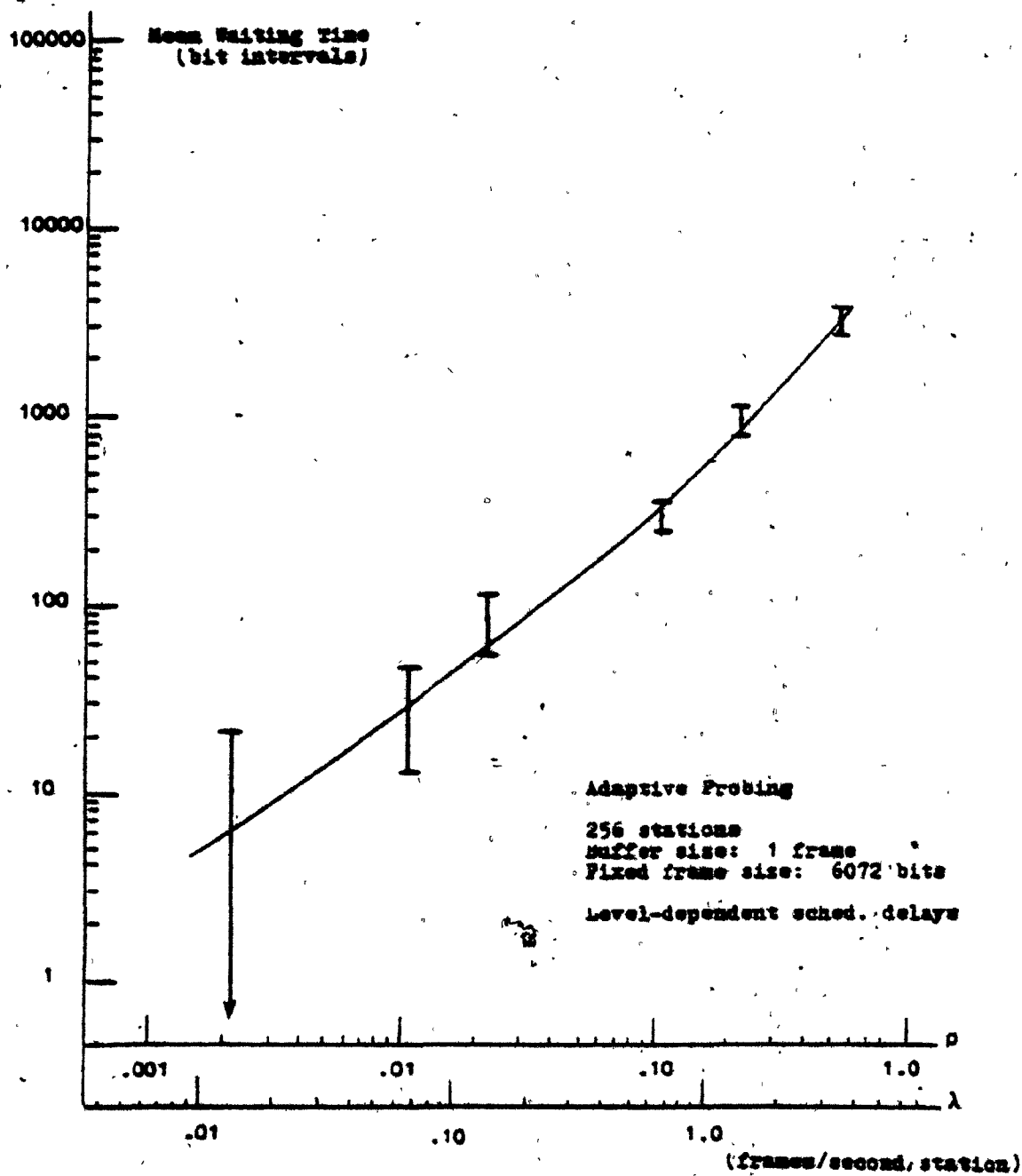


Figure 2.5.2.c Mean Waiting Time versus Load ρ and Arrival Rate λ

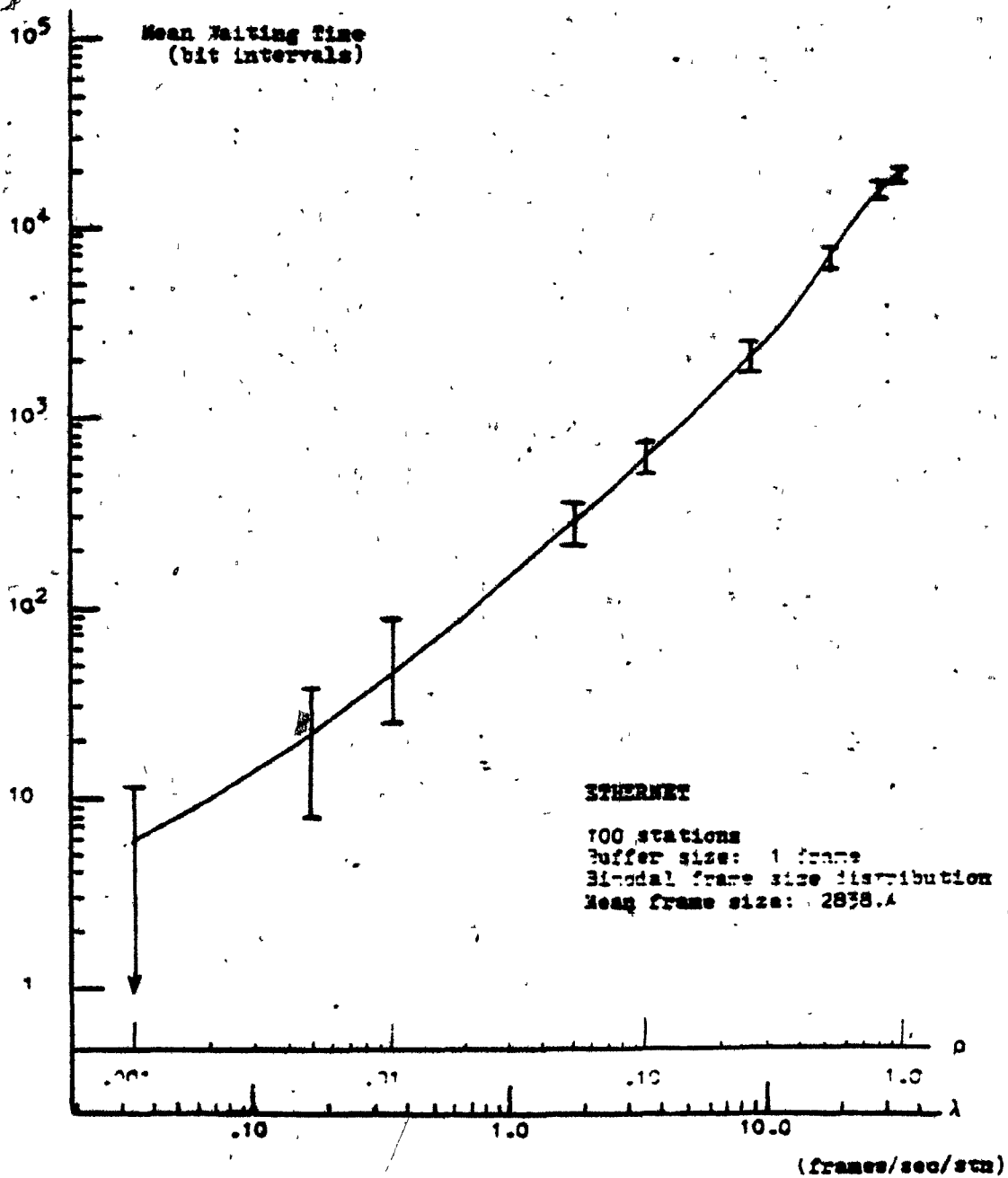


Figure 2.5.2.d Mean Waiting Time versus Load ρ and Arrival Rate λ

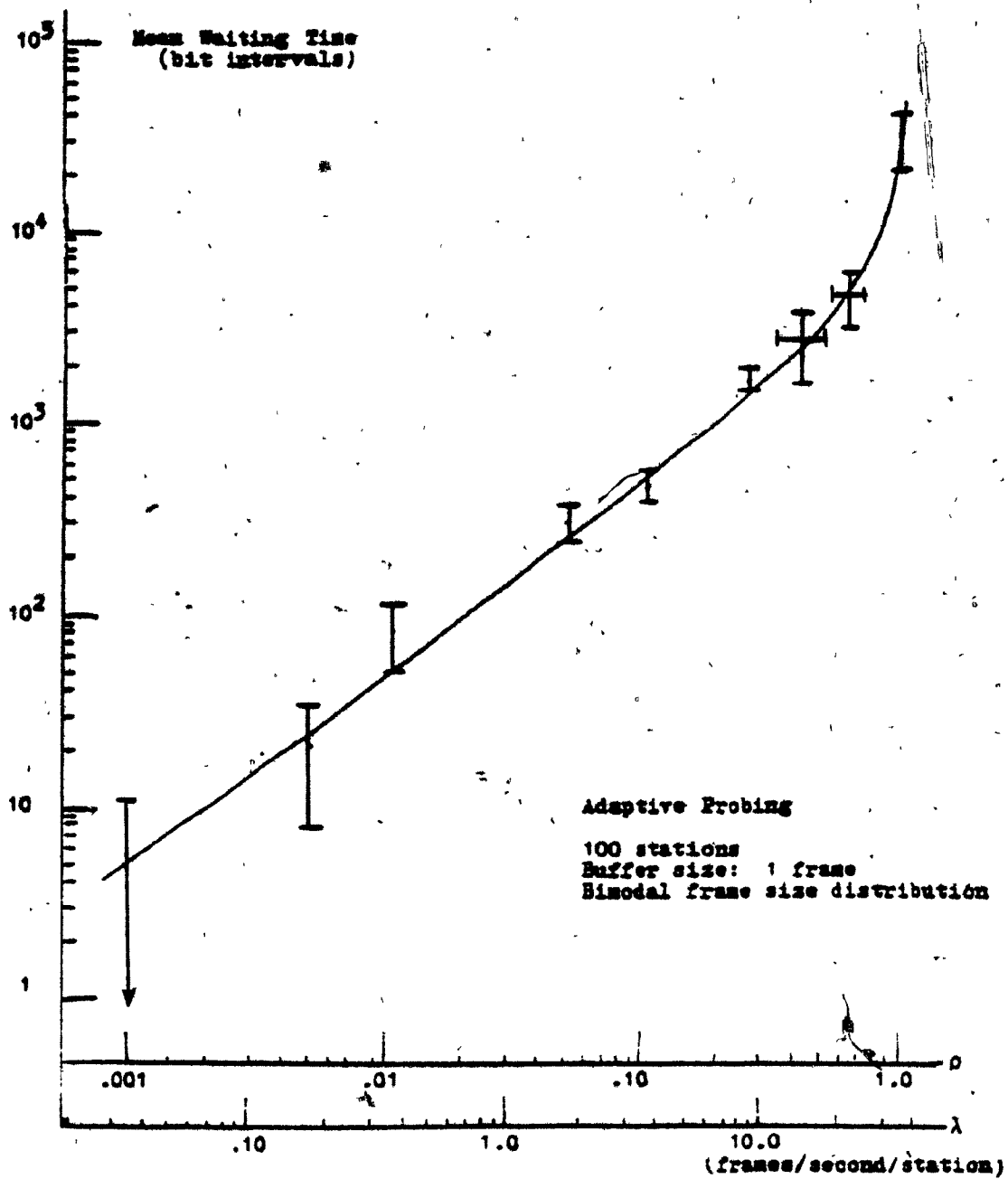


Figure 2.5.2.f Mean Waiting Time versus Load ρ and Arrival Rate λ

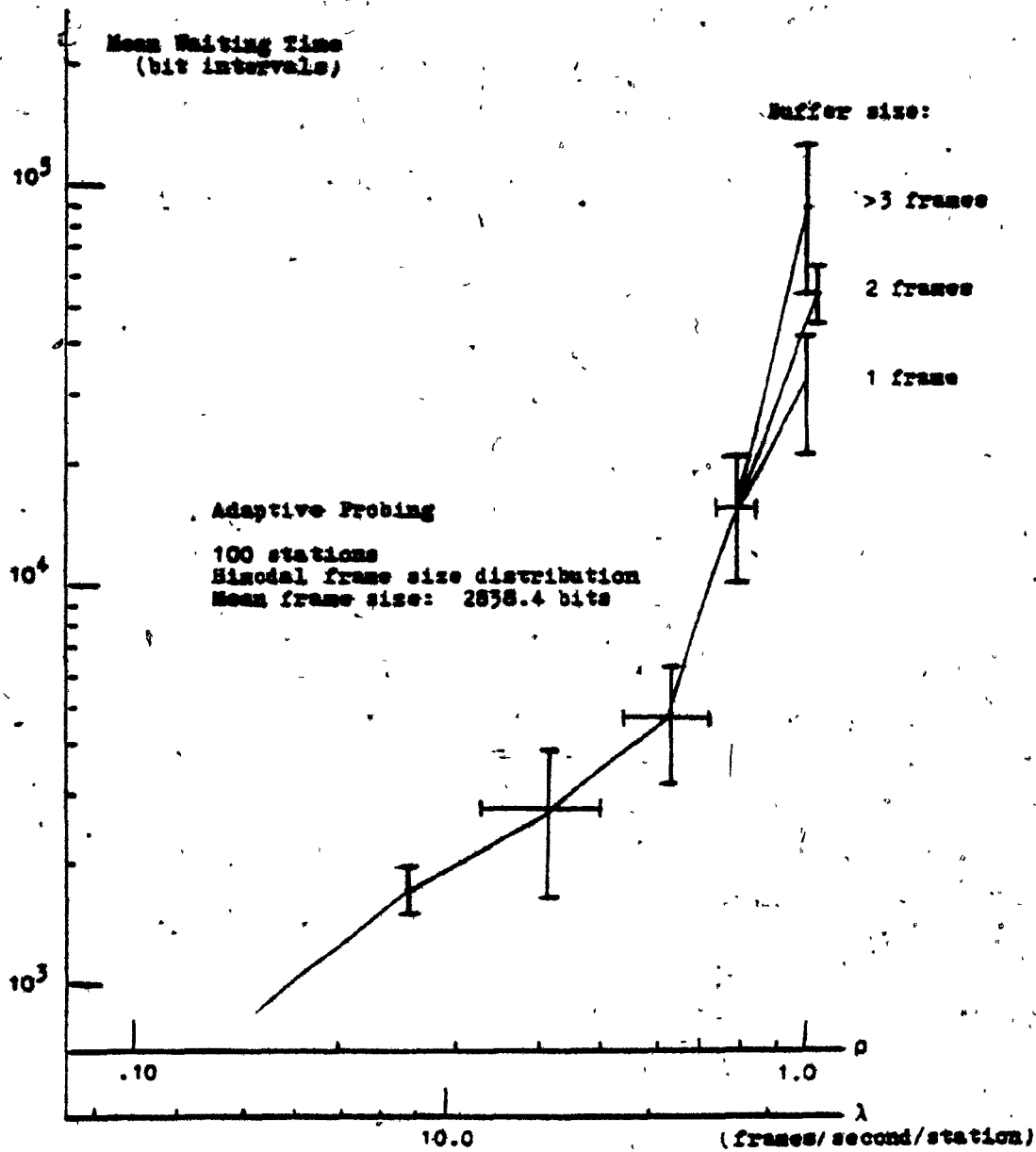


Figure 2.5.2.g Mean Waiting Time versus Load ρ and Arrival Rate λ

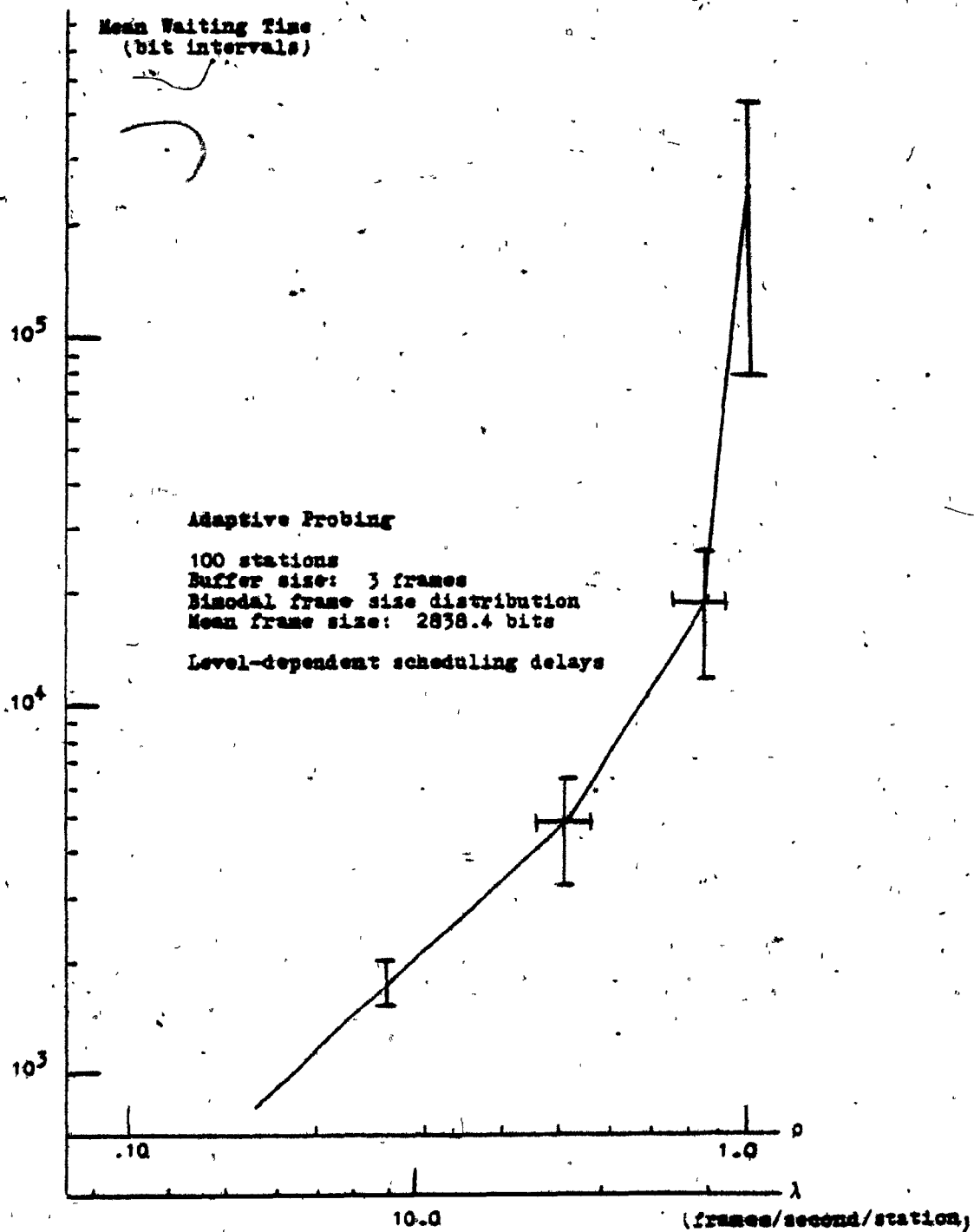


Figure 2.5.2.h Mean Waiting Time versus Load ρ and Arrival Rate λ

2.5.3 Mean Queue Size

Not all of the performance indicators under consideration here are solely functions of the load ρ . Those indicators referring to an individual station's performance rather than system performance reveal a dependence on the arrival rate λ . The mean queue size, for example, reflects the arrival rate of packets to each station's queue. Figures 2.5.3.a and 2.5.3.b vividly demonstrate this effect for Ethernet: as a function of load, Figure 2.5.3.a, the system with less stations and therefore a higher arrival rate shows a greater mean queue size. As a function of arrival rate, Figure 2.5.3.b, the curves of mean queue size for the two systems are very similar until they begin to diverge at higher arrival rates. At those high arrival rates, the system's performance in servicing frames in each station's queue — a function of load, not arrival rate — begins to be the major factor affecting queue size. Indeed, Figure 2.5.3.a may be interpreted to suggest that, at the highest loads, the mean queue size tends to a common value independent of arrival rate.

The remaining graphs of mean queue size, Figures 2.5.3.c-h, show the performance of 256- and 100-station Ethernet and adaptive probing systems. Waiting time and queue size are intimately related by Little's result [25] so it is not surprising to see that the findings here parallel those obtained for the mean waiting time: the mean queue size is identical for both systems except at the highest load, where adaptive probing has a greater mean queue size than Ethernet. Again, one surmises that the overhead incurred as adaptive probing operates farther down the binary search tree accounts for the difference: more frames arrive to each queue and must wait as the current probe cycle continues. The dependence of mean queue size on buffer size is also observable in these results and has already been discussed in the context of the mean waiting time.

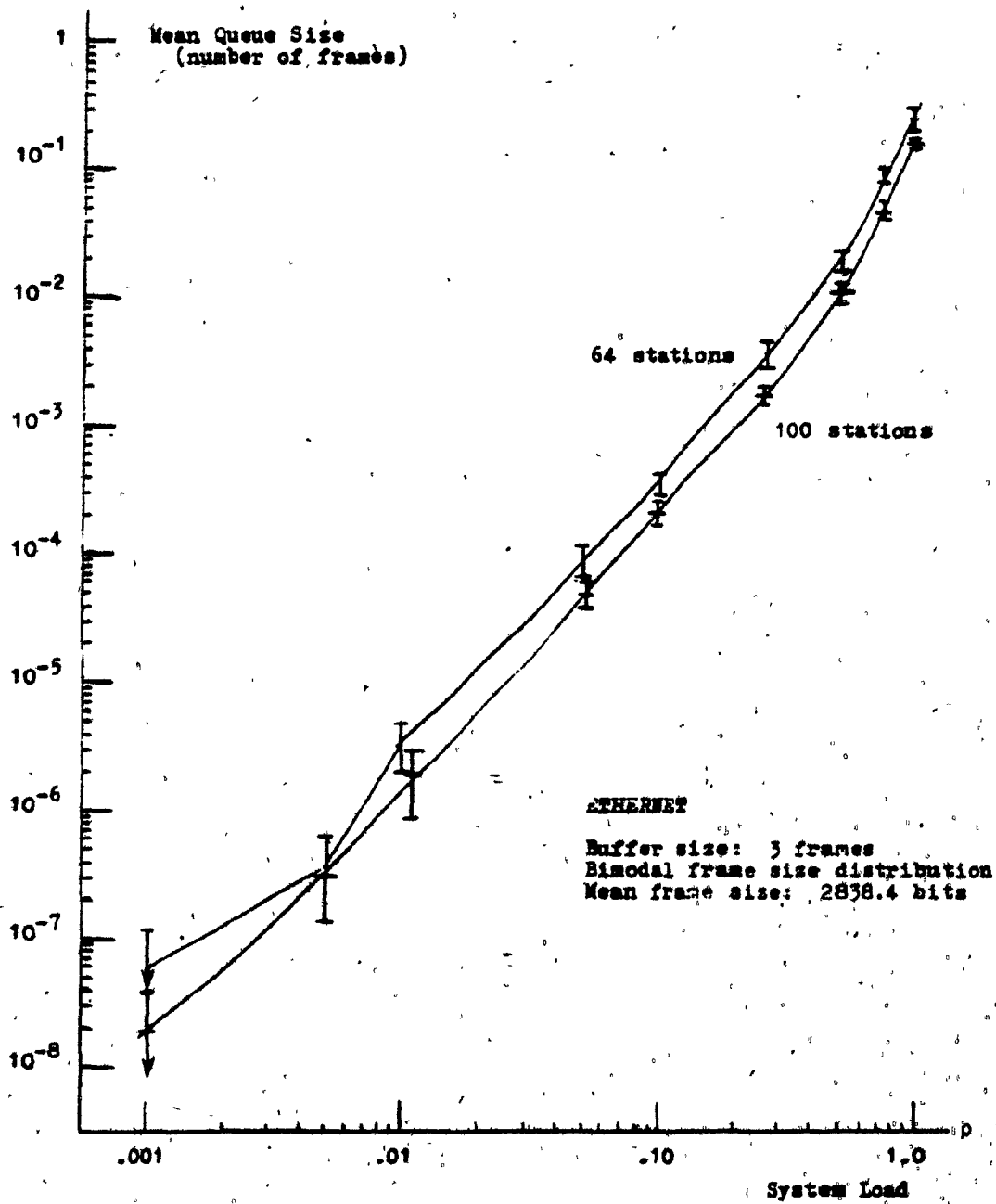


Figure 2.5.3.a Mean Queue Size versus Load ρ

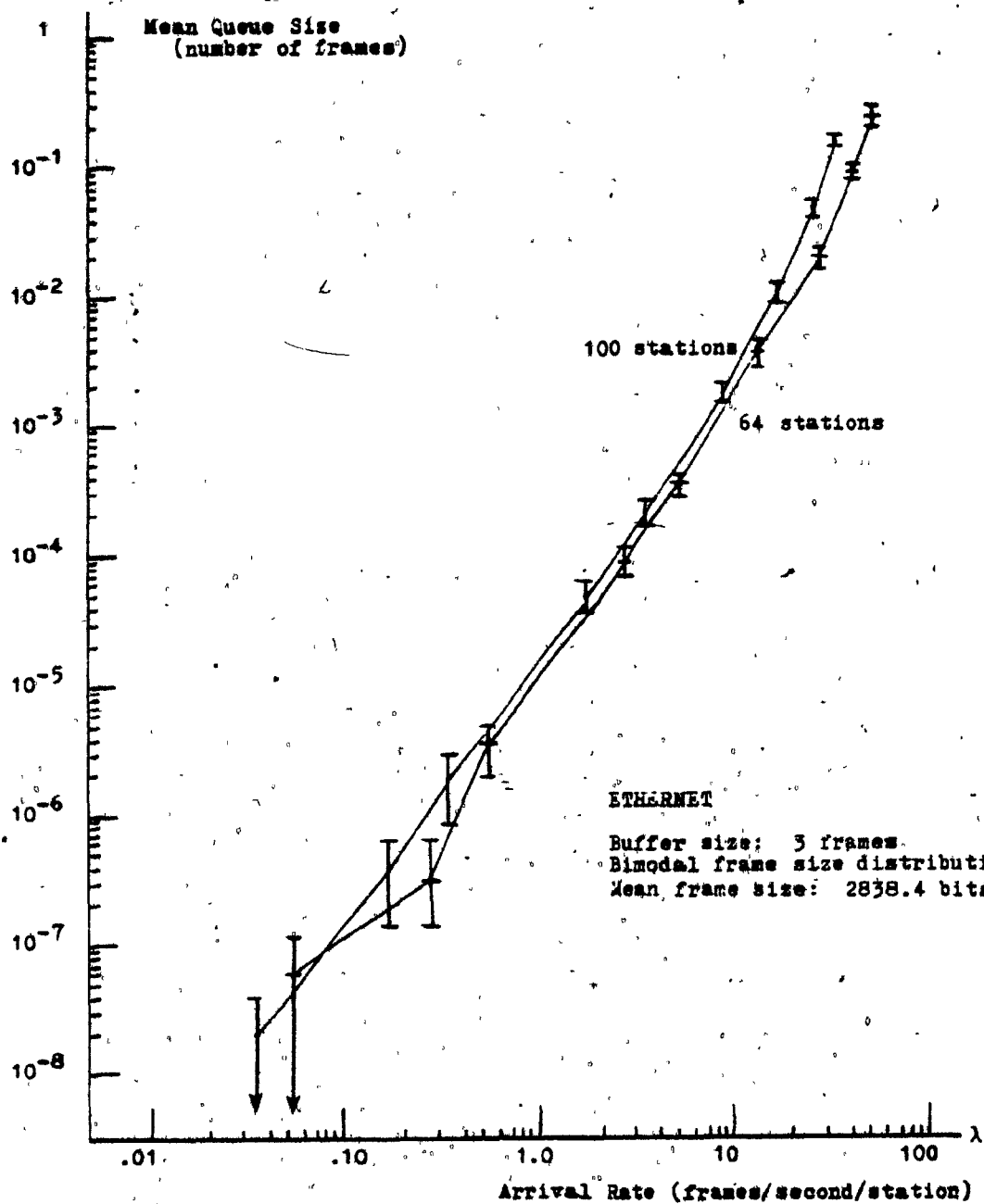


Figure 2.5.3.b Mean Queue Size versus Arrival Rate λ

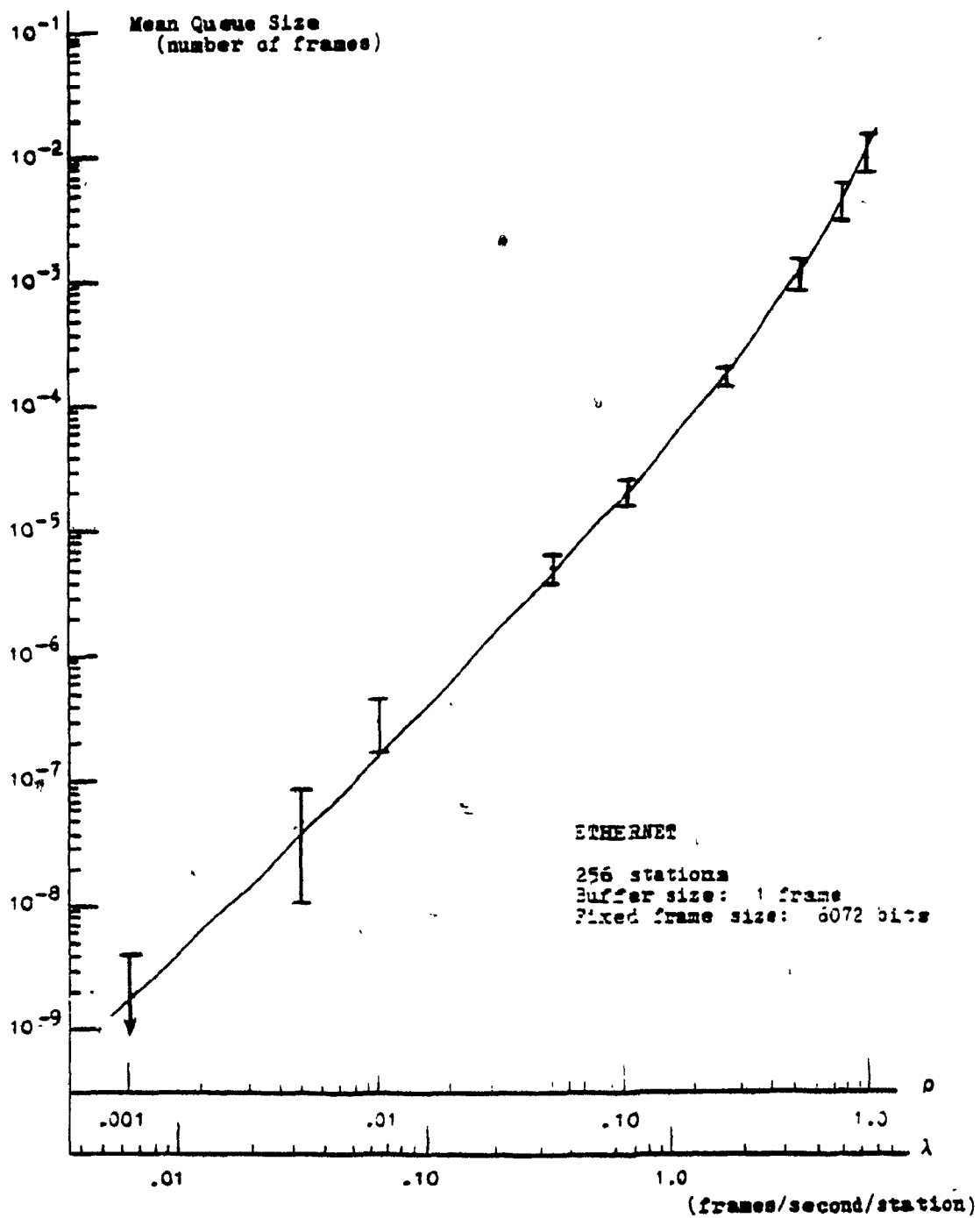


Figure 2.5.3.c Mean Queue Size versus Load ρ and Arrival Rate λ

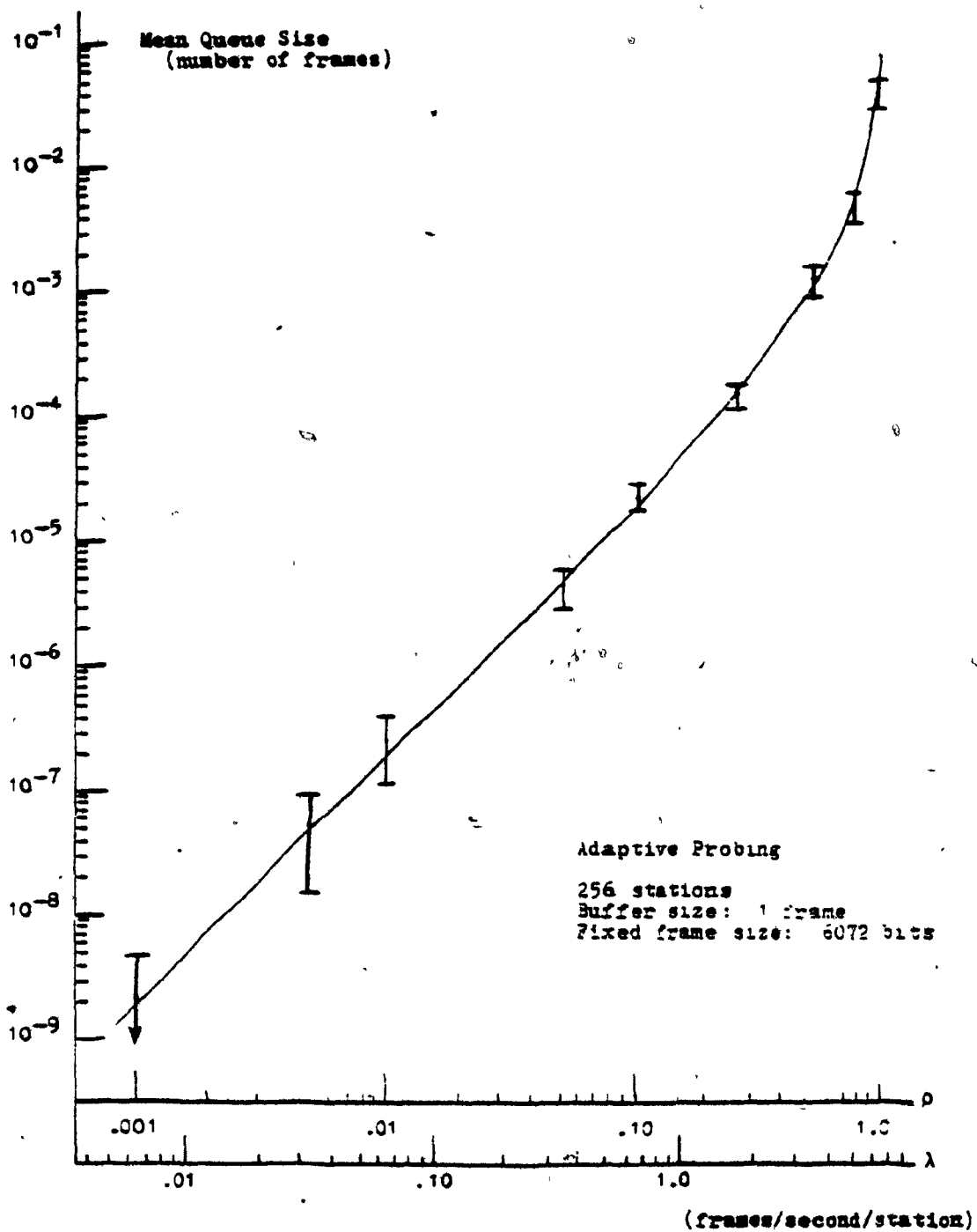


Figure 2.5.3.d Mean Queue Size versus Load ρ and Arrival Rate λ

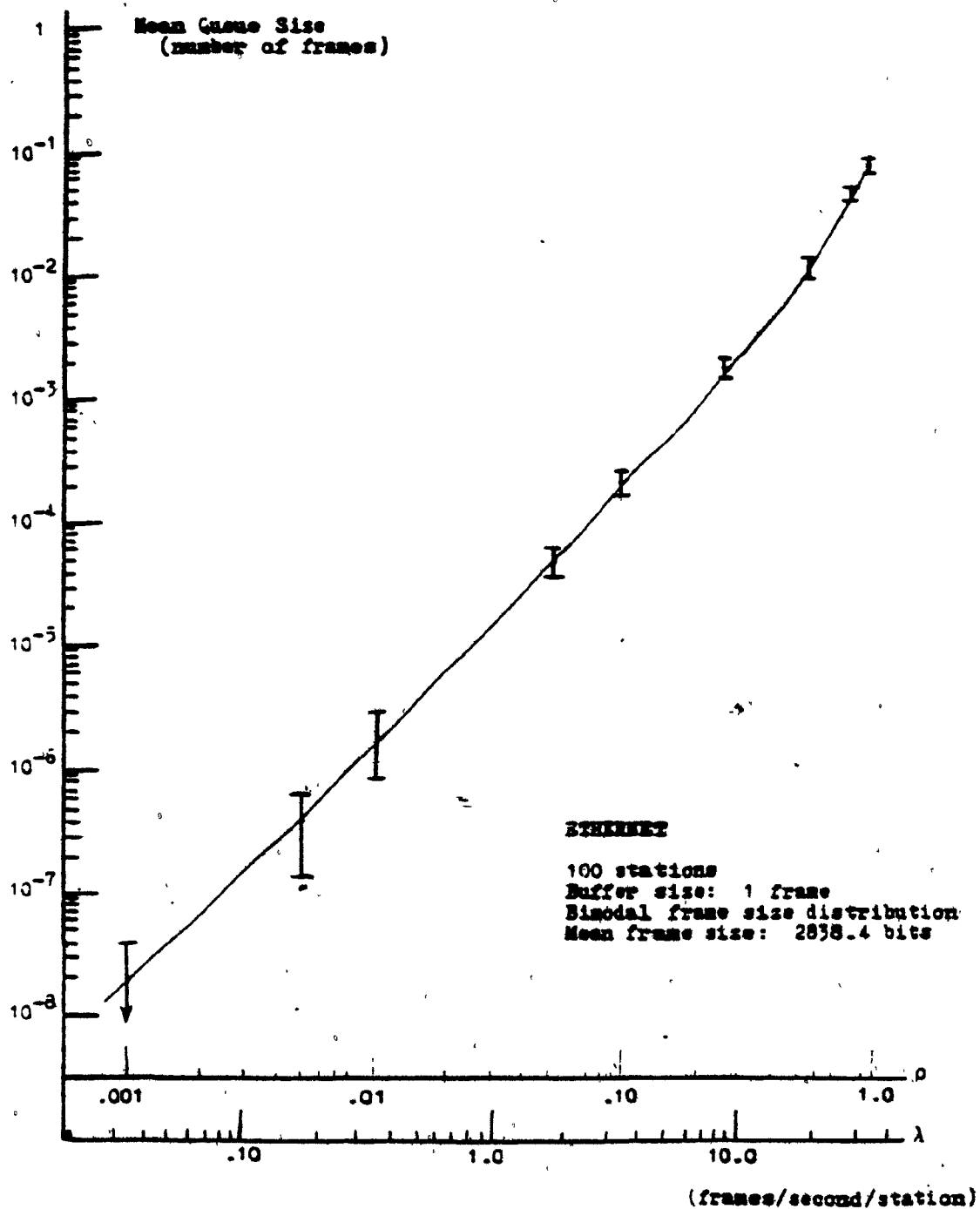


Figure 2.5.3.e Mean Queue Size versus Load ρ and Arrival Rate λ

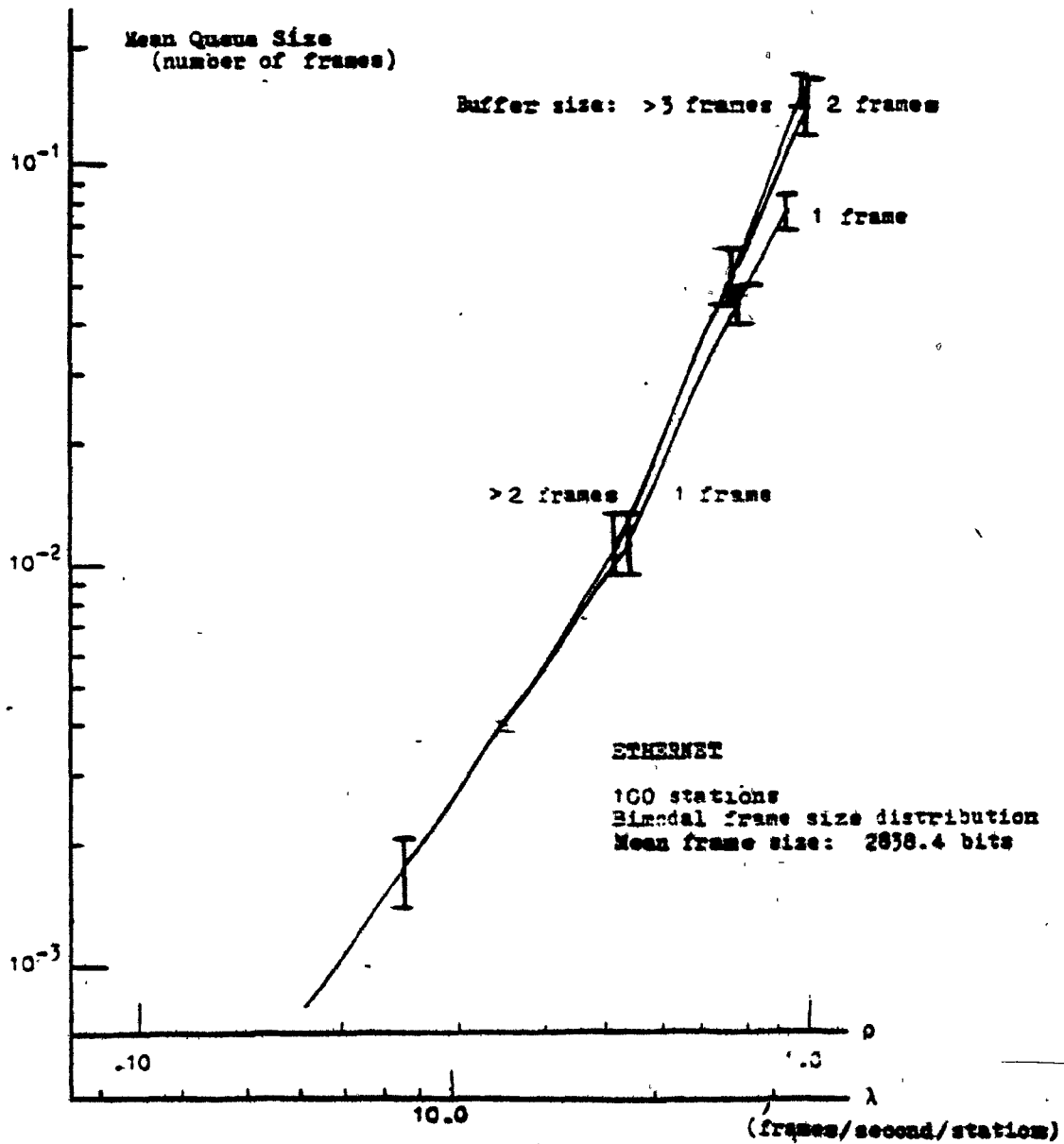


Figure 2.5.3.f Mean Queue Size versus Load ρ and Arrival Rate λ

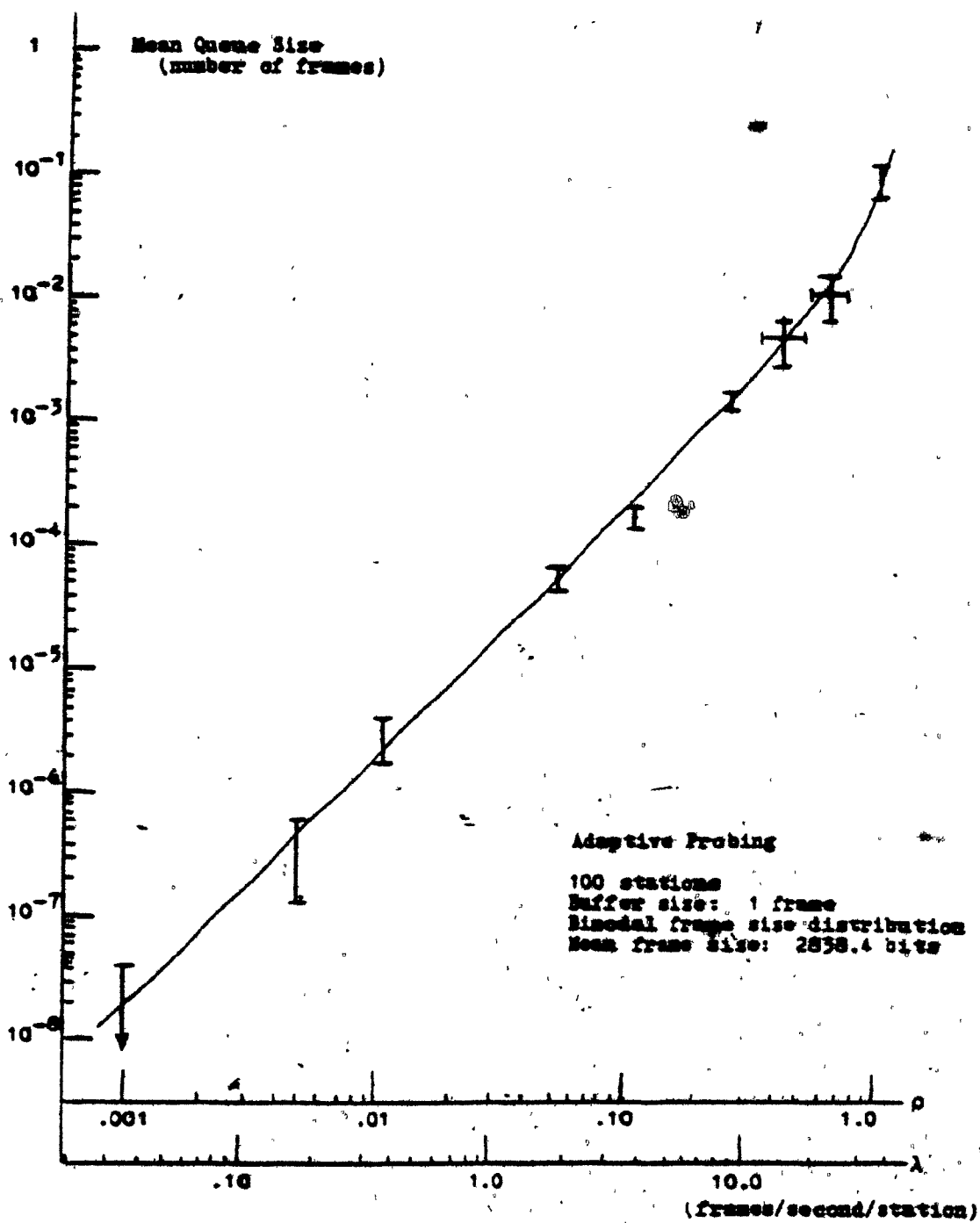


Figure 2.5.3.g Mean Queue Size versus Load ρ and Arrival Rate λ

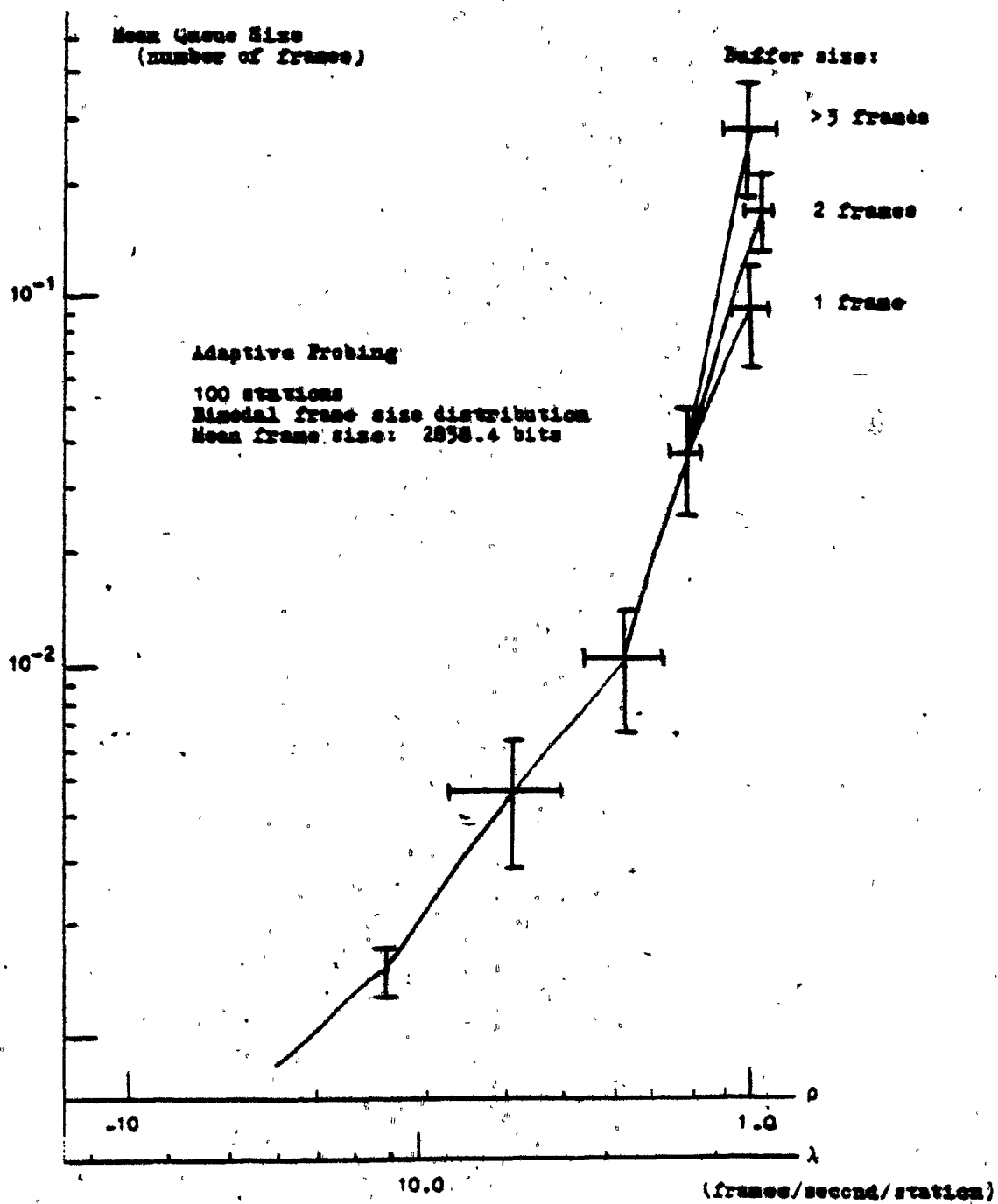


Figure 2.5.3.h Mean Queue Size versus Load ρ and Arrival Rate λ

2.5.4 Mean Rejection Rate

The most serious impact of a station's buffer size is demonstrated by the mean rejection rate, the mean rate at which frames are turned away from full station buffers. A non-zero rejection rate represents the failure of the access protocol and the station buffers to accommodate the offered load. Like the mean queue size, the mean rejection rate can be expected to depend not just on system load, but also on the arrival rate of frames to each station. On the other hand, since rejection rates only become apparent at the highest loads when station buffers are most likely to be full, perhaps system performance alone determines the rejection rate. The available data, including data not shown here, is, unfortunately, inconclusive.

In Figures 2.5.4.a-d, the mean rejection rate is seen to remain zero for loads up to and including .95 for buffer sizes greater than or equal to four frames. This observation is true for both Ethernet and adaptive probing and for systems of 100, 64, and 48 stations. Since $\rho = N\lambda T$, one may conclude that, subject to the conditions of this simulation, a buffer size of four or more frames will provide sufficient capacity to keep the mean rejection rate to zero for systems larger than 48 stations whether the rejection rate is dependent on either arrival rate or load. The results obtained for the mean queue size and mean waiting time are in agreement, exhibiting no change with buffer sizes greater than three: a buffer size greater than or equal to four frames provides sufficient excess buffer capacity to handle extremely high loads. The curves for the mean waiting time, however, show that, at the highest load, as the buffer size is increased from one to two to three, each increment costs on the order of tens of thousands of bit times in mean frame delay. Since a rejected frame will probably be rescheduled for transmission at some later time, thus effectively introducing additional delay, the best design policy would seem to be to provide each station with an adequate buffer size to handle surges of high traffic levels. The model employed here, of course, assumes identical arrival rates to all stations. When load is not balanced evenly throughout the system, one expects the best performance to be realized when the more active stations are provided with greater buffer capacity than less active ones.

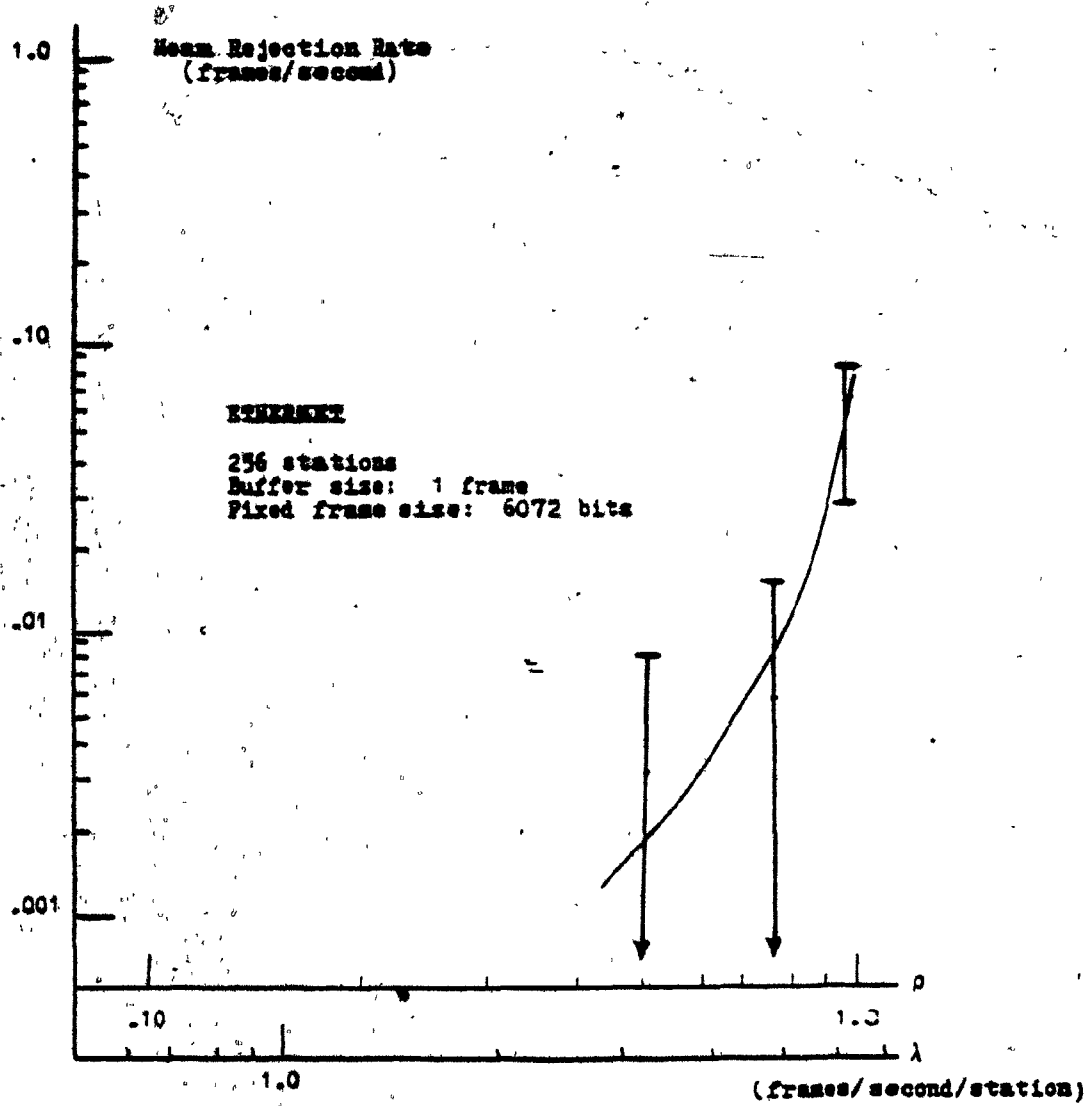


Figure 2.5.4.a Mean Rejection Rate versus Load ρ and Arrival Rate λ

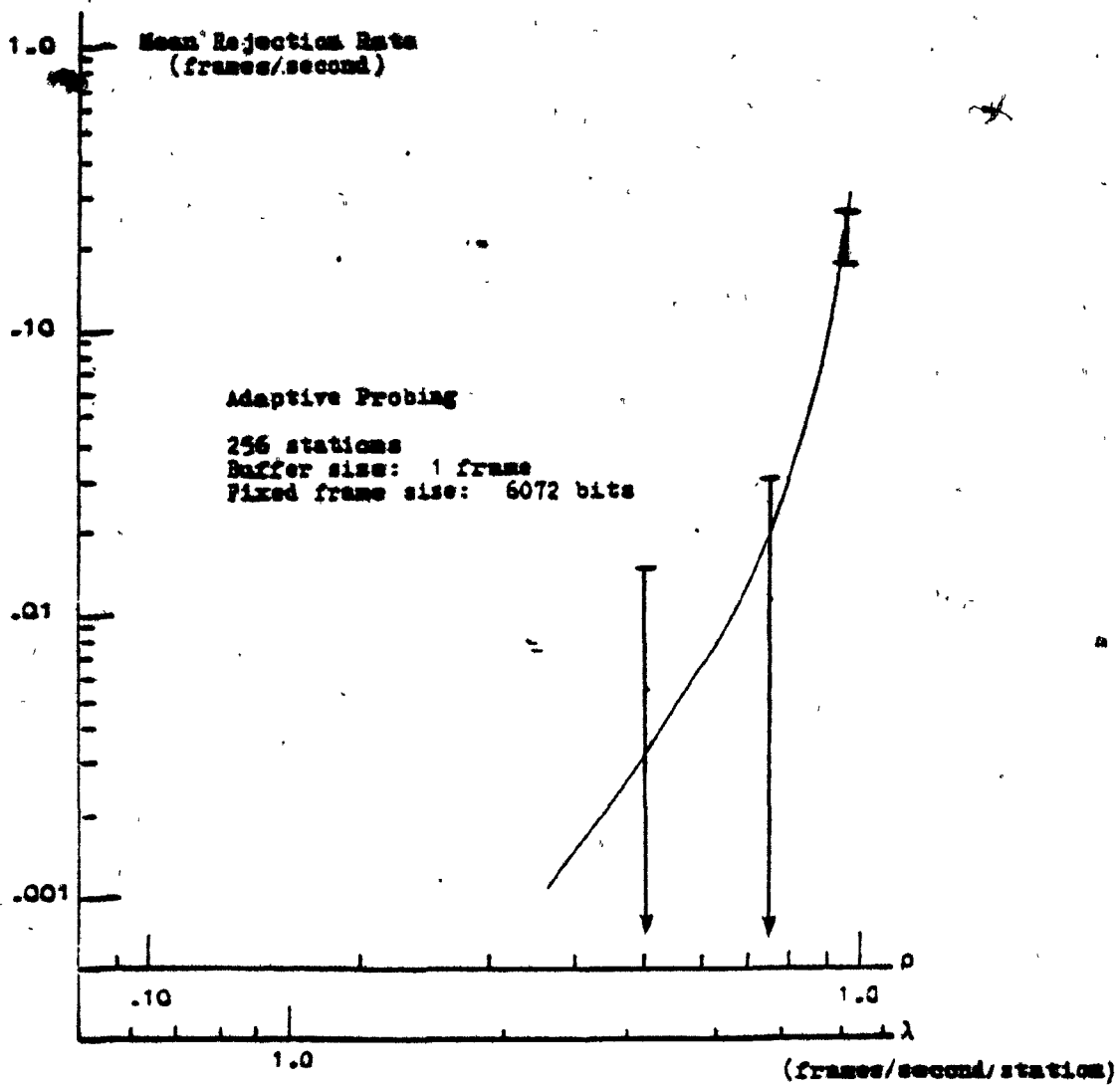


Figure 2.5.4.b Mean Rejection Rate versus Load ρ and Arrival Rate λ

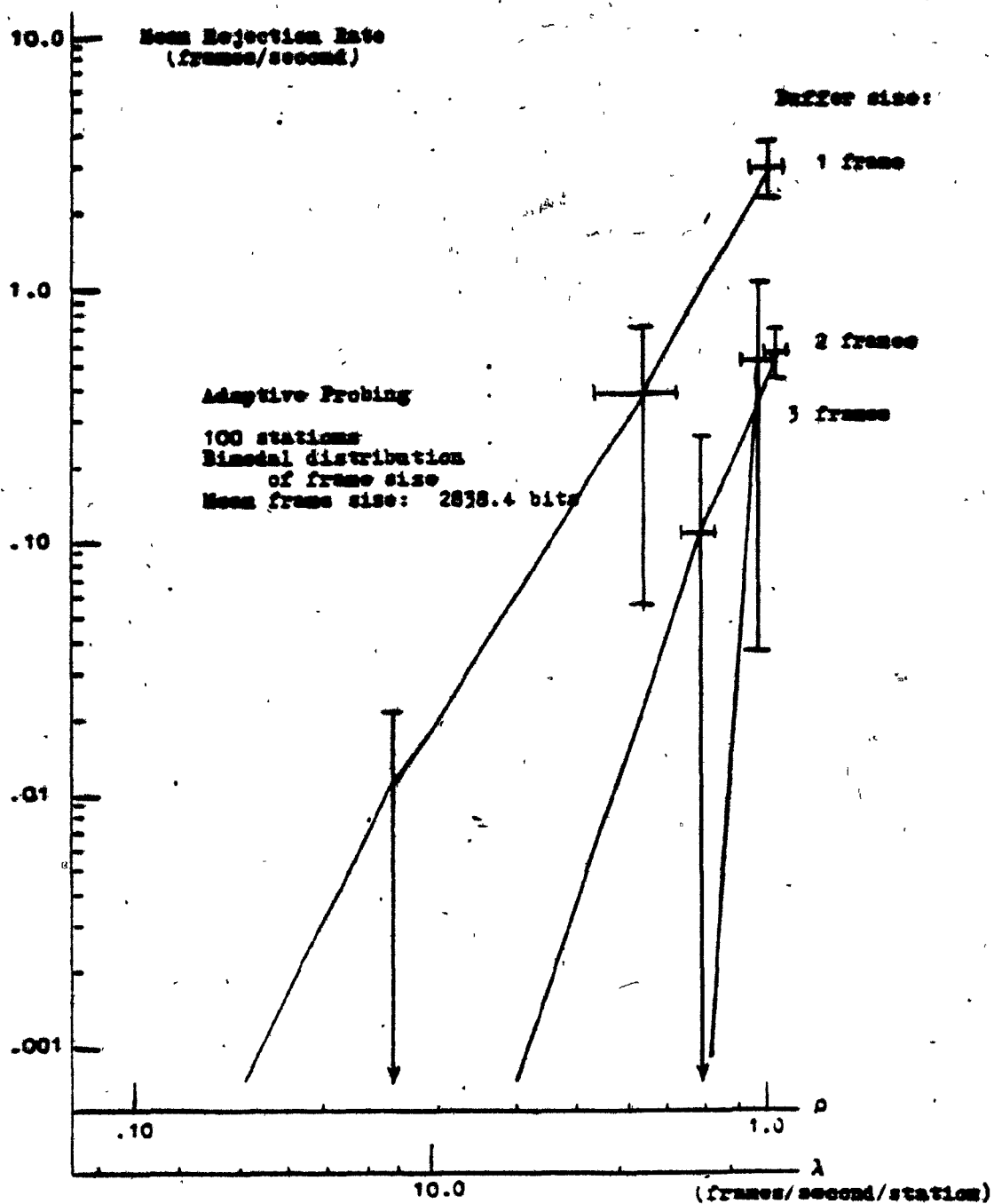


Figure 2.5.4.c Mean Rejection Rate versus Load ρ and Arrival Rate λ

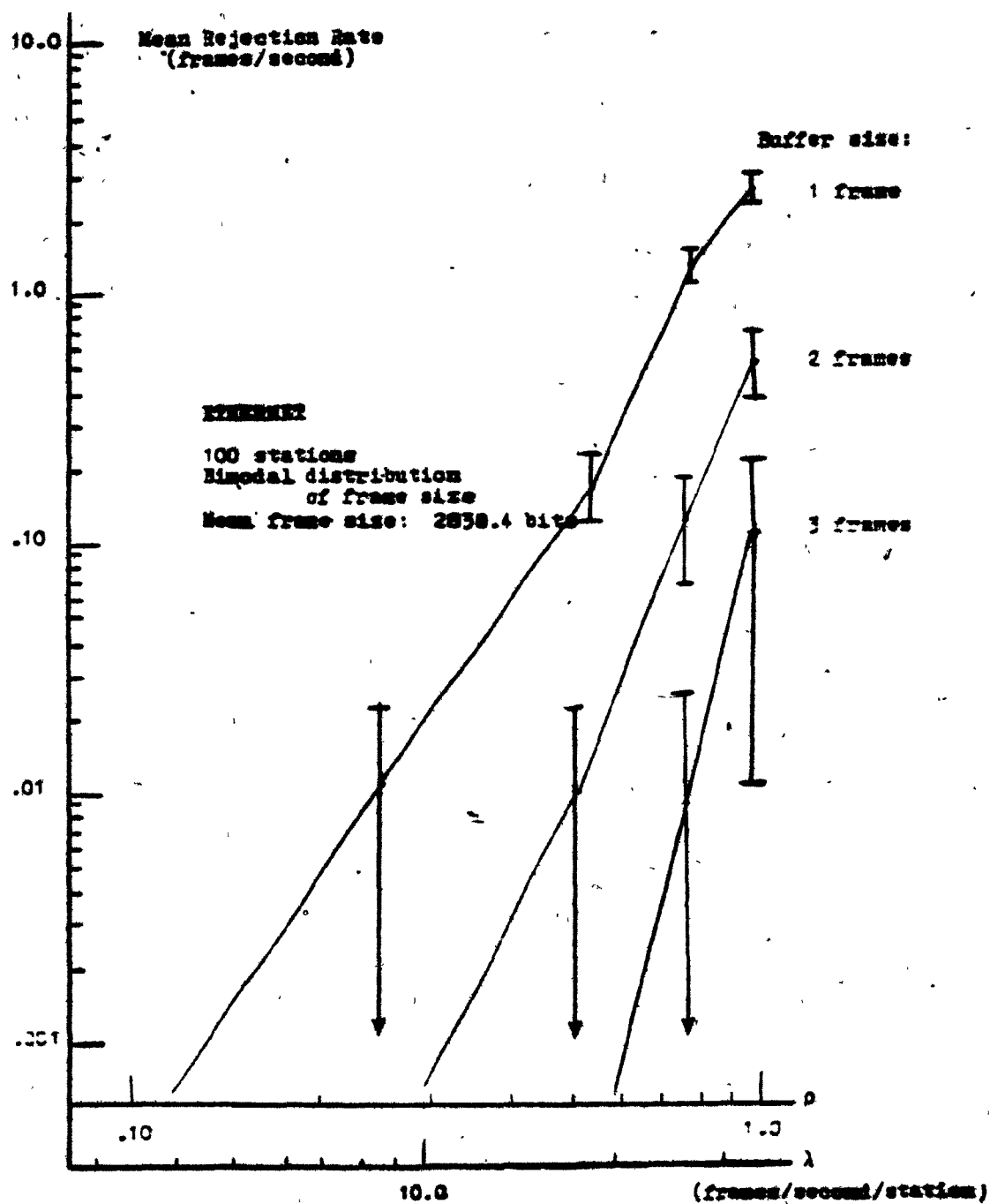


Figure 2.5.4.d Mean Rejection Rate versus Load ρ and Arrival Rate λ

2.5.5 Mean Failure Rate

The mean failure rate, the mean rate at which data frames fail to gain access to the channel in the required number of attempts — here, sixteen — results from the access protocols inability to efficiently resolve contention. Figures 2.5.5.a and 2.5.5.b are for Ethernet only; no curves are shown for adaptive probing since that algorithm guarantees successful transmission by ultimately polling all stations. The limit of sixteen attempts means that, in the worst case, it would require a population of more than 2^{15} stations for adaptive probing to evince a non-zero failure rate. Ethernet, however, can fail to successfully transmit data packets at high loads ($\rho \geq .5$). Note that failure rates at the same load levels are different for systems of 256 or 100 stations: it is not the system failure rate which is shown, but the average failure rate observed at each station. Here, then, is the tradeoff between selection and contention access control techniques: while providing reduced waiting times at low loads in comparison to token rings or conventional polling systems, a contention technique such as Ethernet cannot give equivalent performance at the highest load levels. Adaptive probing provides a happy compromise, operating as a purely random access protocol at the lowest loads and as a purely deterministic polling system at the highest loads.

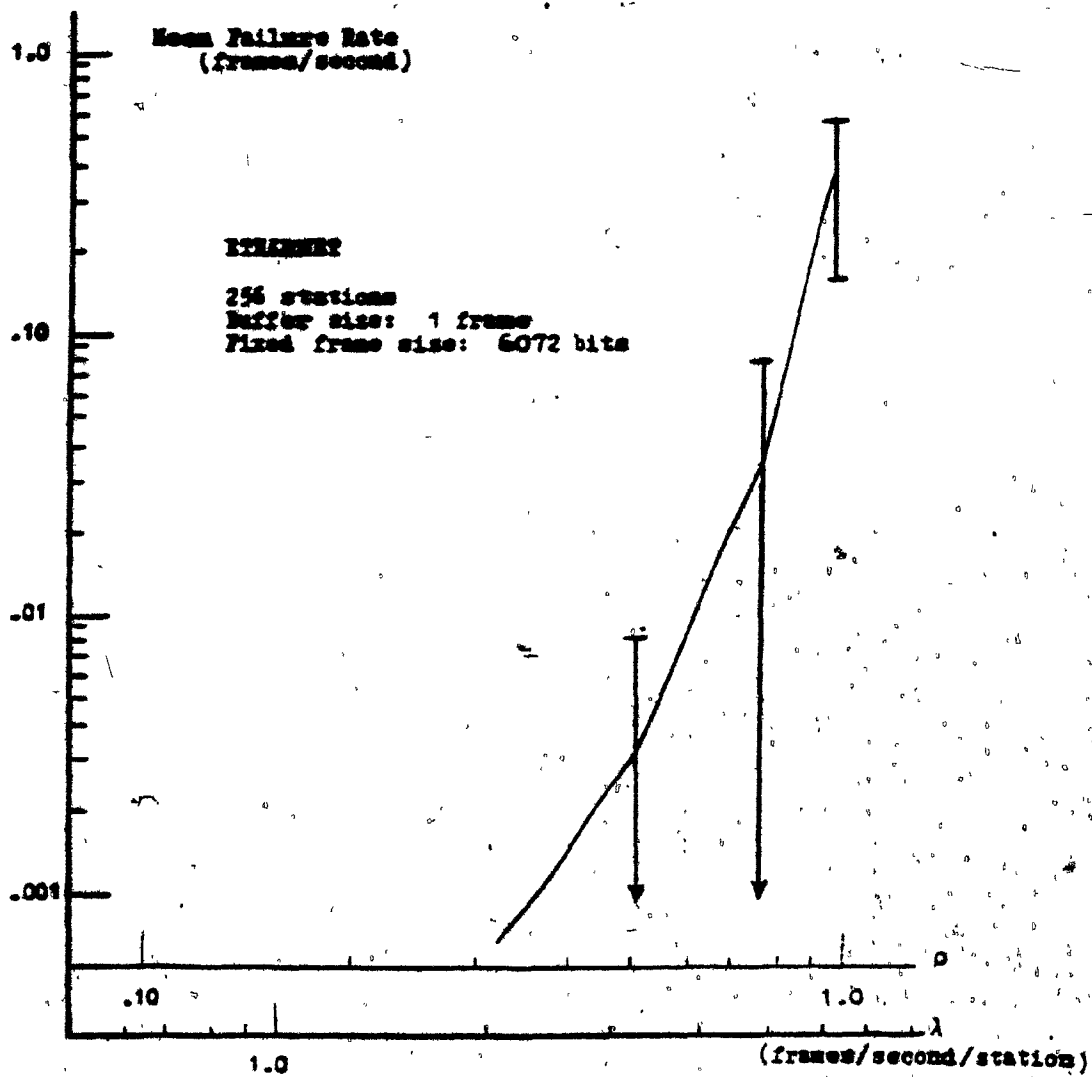


Figure 2.5.5.a Mean Failure Rate versus Load ρ and Arrival Rate λ

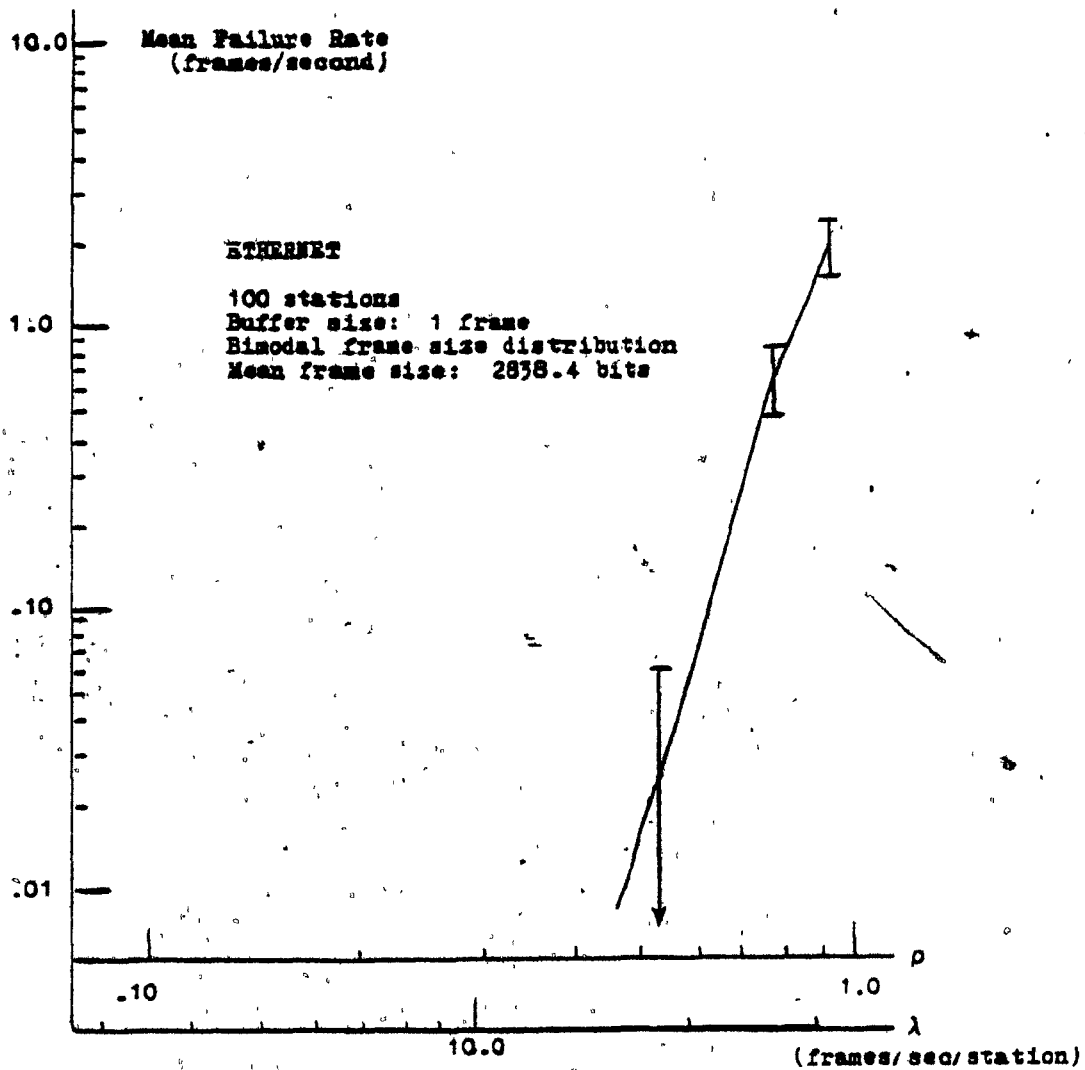


Figure 2.5.5.b Mean Failure Rate versus Load ρ and Arrival Rate λ

2.5.6 Mean Number of Attempts Required

The mean number of attempts a frame requires for its successful transmission reveals the functioning of the access protocol in its effort to resolve contention among stations. As seen in Figures 2.5.6.b and 2.5.6.d, the decentralized adaptive probing algorithm assesses system traffic conditions accurately enough to limit the mean number of attempts required. In fact, Figure 2.5.6.d seems to show that, at a load of .95, the adaptive probing algorithm has successfully anticipated the load level and is able to reduce the number of failed transmissions used in directing the binary search. The CSMA-CD algorithm of Ethernet, however, requires more and more transmission attempts as the system load increases, as shown in Figures 2.5.6.a and 2.5.6.c.

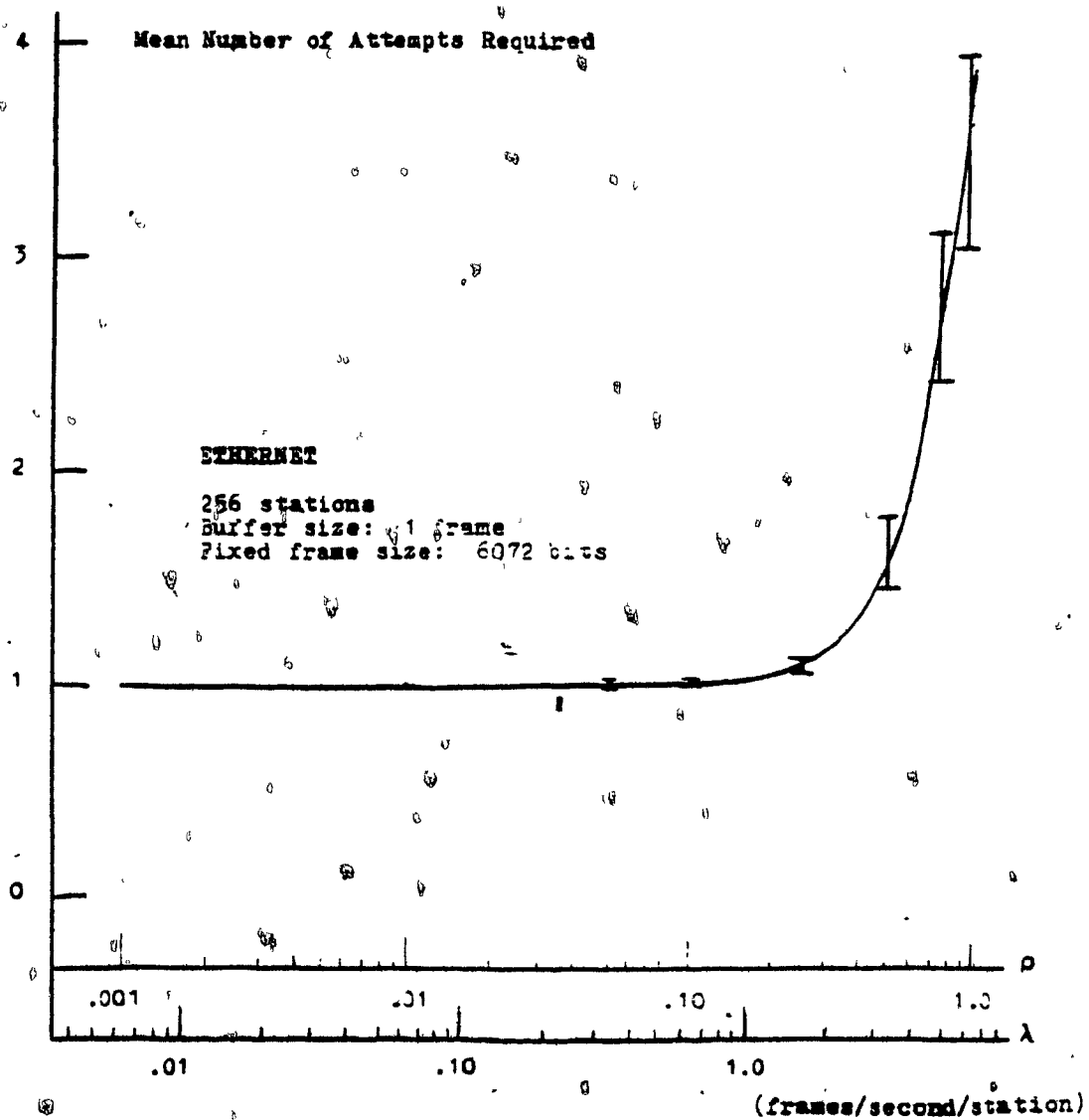


Figure 2.5.6.a Mean Number of Attempts Required versus Load ρ and Arrival Rate λ

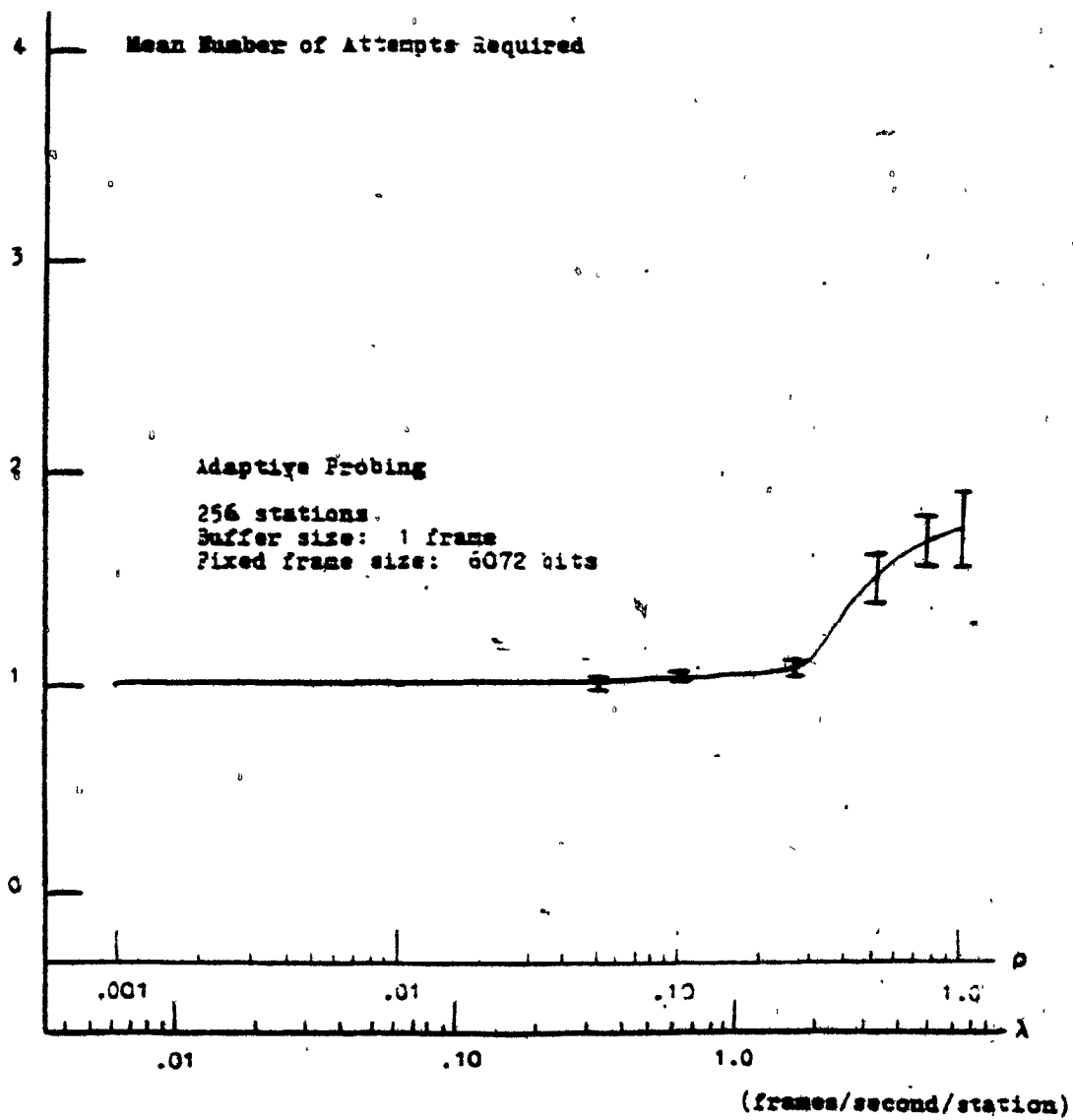


Figure 2.5.6.b Mean Number of Attempts Required versus Load ρ and Arrival Rate λ

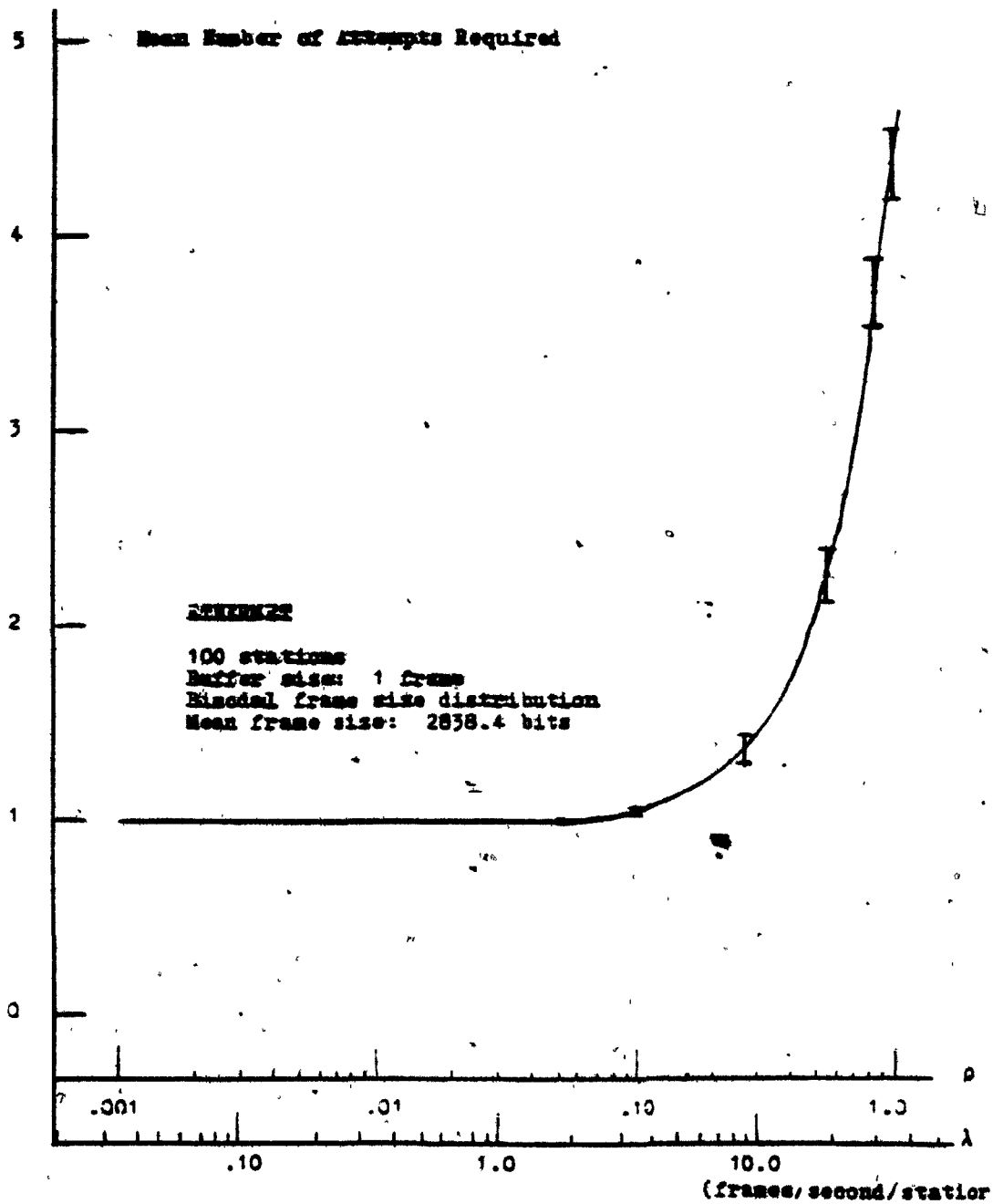


Figure 2.5.6.c Mean Number of Attempts Required versus Load ρ and Arrival Rate λ .

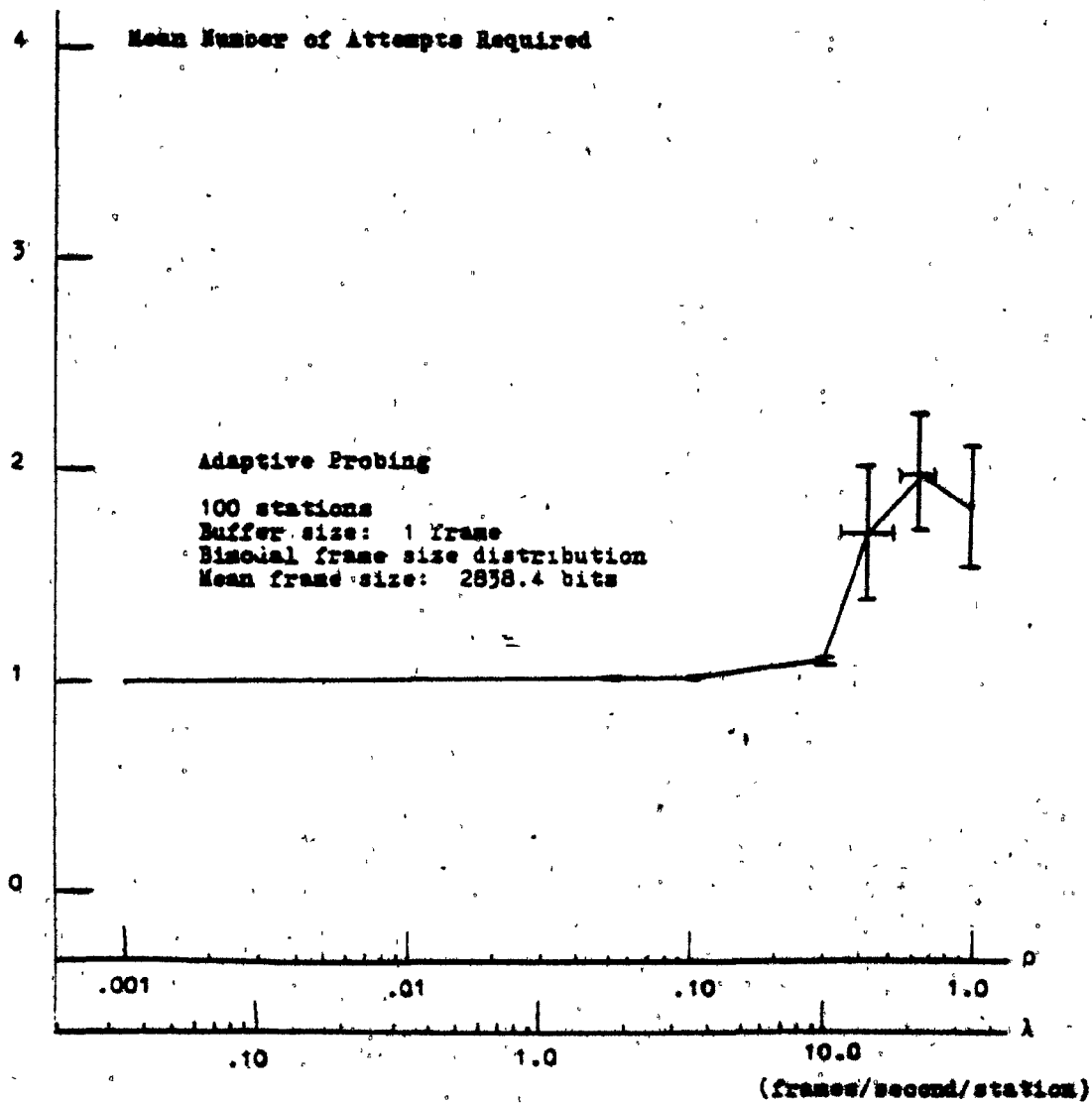


Figure 2.5.6.d Mean Number of Attempts Required versus Load ρ and Arrival Rate λ

2.5.7 Mean Probe Level

The functioning of the adaptive probing algorithm may be monitored by observing the mean probe level maintained by the system. As explained in Section 2.2.2, a collision or a history of collisions on the channel directs the binary search used to allocate access to the channel. After the opportunity to access the channel has been granted to the last of the stations on the current probe cycle, the algorithm begins again, initiating a new cycle. The level down the binary search tree at which the probe cycle begins is based on each station's estimation of the probability that other stations have a message. Possible discrepancies among individual estimates are finally resolved by the probe level word contained within the next successfully transmitted frame. If the cycle begins with 2^n groups to be probed, the probe level is said to be n . Contention within each group will cause that group to be probed further; i.e., the binary search will progress down the tree until all ready stations within the initial group have successfully transmitted. Access to the channel will then pass to all ready stations in the next of the 2^n groups. Figures 2.5.7.a and 2.5.7.b demonstrate that the mean probe level increases to adjust to increasing system loads, as expected.

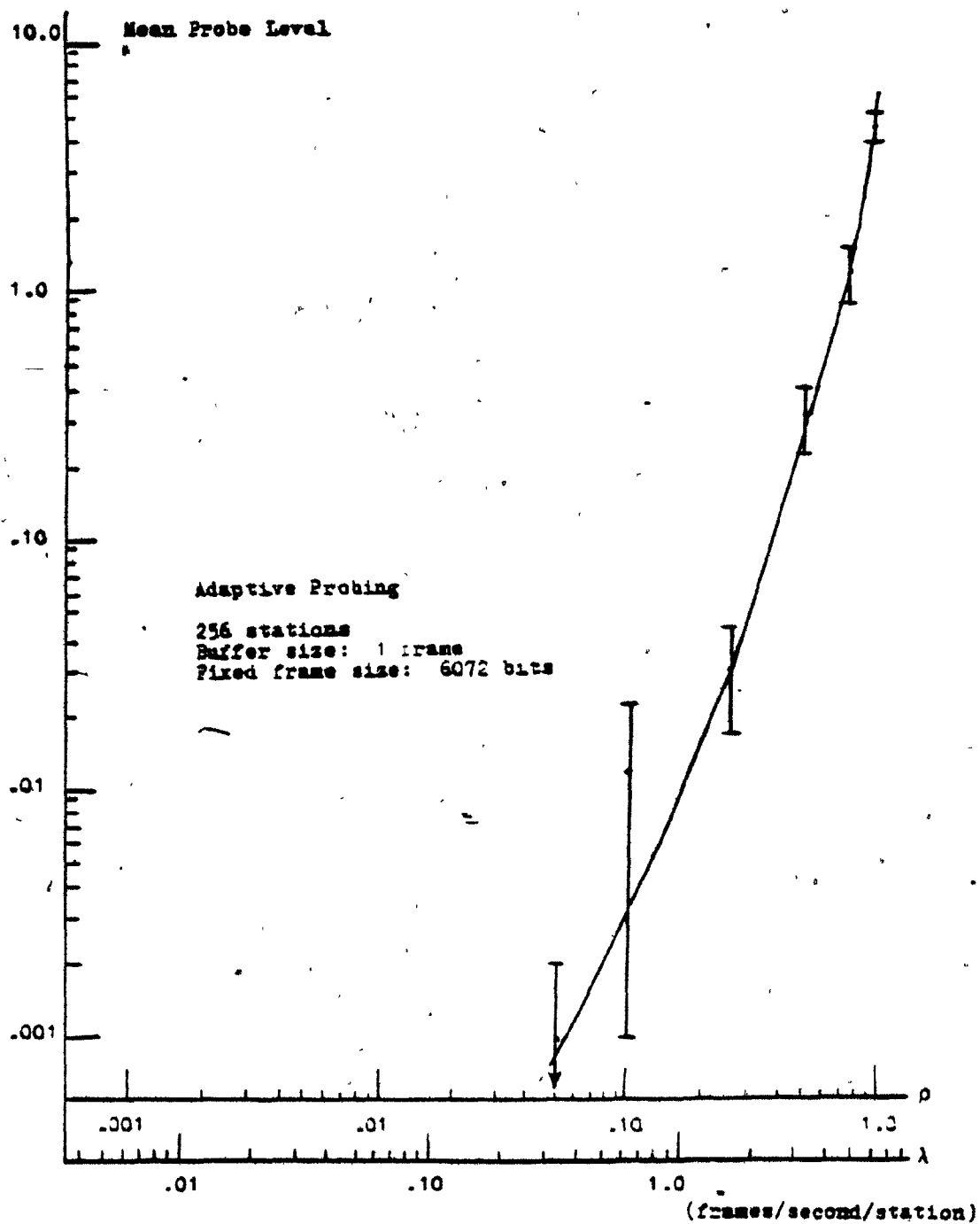


Figure 2.5.7.a Mean Probe Level versus Load ρ and Arrival Rate λ

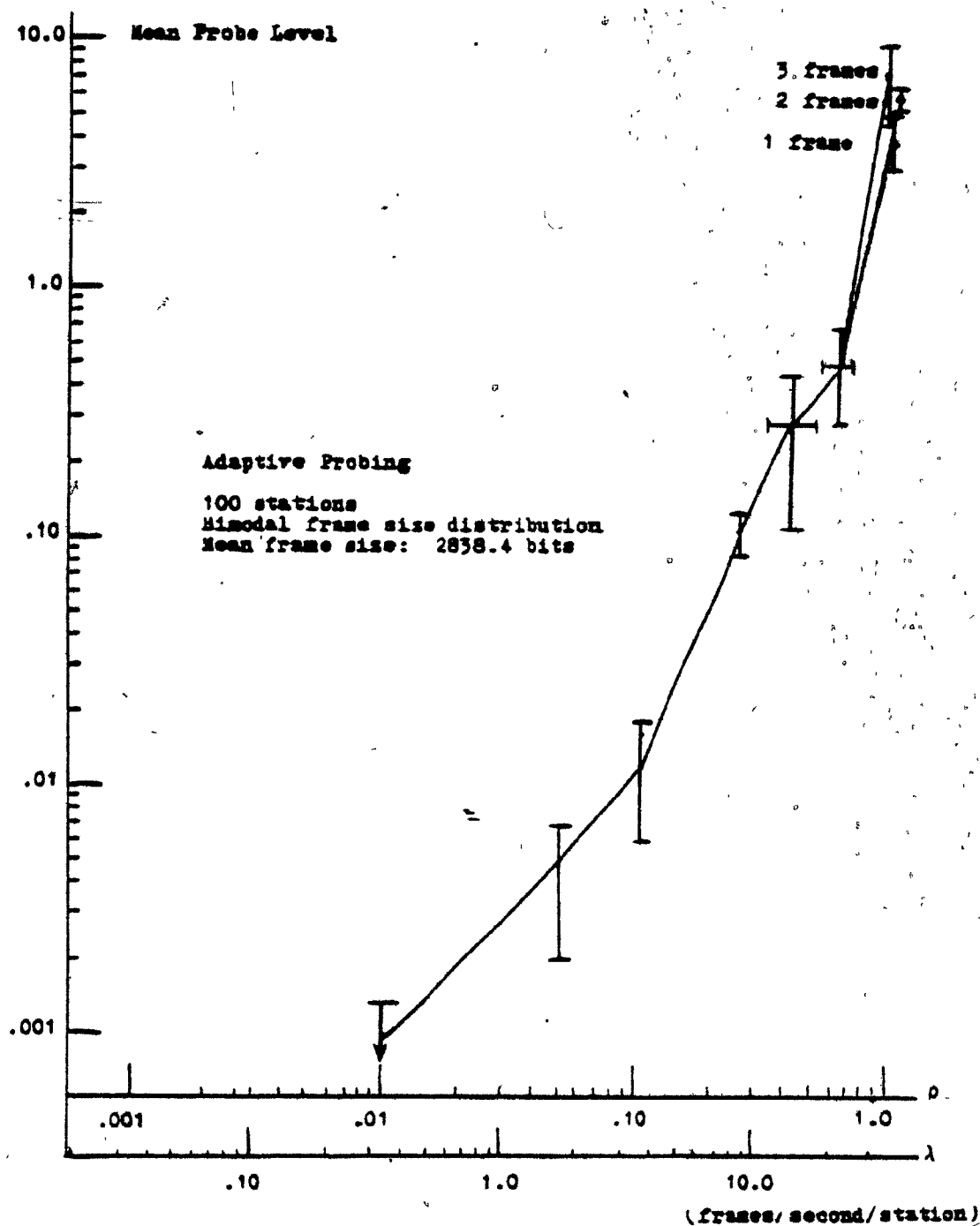


Figure 2.5.7.b Mean Probe Level versus Load ρ and Arrival Rate λ

3. An Analysis of CSMA-CD With a Finite Number of Buffered Users

3.1 Introduction

A number of analyses of Carrier Sense Multiple Access (CSMA) or Carrier Sense Multiple Access with Collision Detection (CSMA-CD) have appeared in the literature, but few, if any, consider the case of stations with limited buffer capacity. This section, then, details an analysis of CSMA-CD with a finite number of buffered users.

Section 3.2 presents the model of non-persistent CSMA-CD used in this analysis. "Non-persistent" implies that a station, upon first generating a packet for transmission and sensing the channel busy, reschedules its transmission as if it had experienced a collision. Contrast this behavior to that of Ethernet's algorithm, 1-persistent CSMA-CD, where the station would transmit with probability one upon next sensing the channel idle. Consideration of non-persistent as opposed to 1-persistent CSMA-CD allows the identification of an imbedded Markov chain and is thus crucial to the analysis. Also note that time is assumed to be slotted, somewhat simplifying the analysis.

Section 3.3 develops the analysis per se, applying the technique of Saadawi and Ephremides [20] of coupling a Markov chain describing the channel with another Markov chain describing a particular user. In that way the interaction of the users on the channel (directed by the access control protocol) as it impacts a particular user's buffer may be assessed. Their work, however, concerns slotted ALOHA and not CSMA-CD, resulting in different state transition diagrams than presented here. The Markov chain describing the channel is adapted from work by Tobagi and Hunt [6]; the Markov chain describing a particular user is derived from the analytical model. Analysis of the two Markov chain leads to a set of nonlinear equations coupled by common parameters; solution then gives the steady-state probabilities and allows quantities such as the throughput and mean waiting time to be calculated. Unlike the example given by Saadawi and Ephremides, however, this analysis leads to two coupled parameters rather than one. Numerical solution by Wegstein's iteration method as employed by Saadawi and Ephremides is thus not suitable. A multivariable form of Wegstein's iteration method exists, but, since one is solving for probabilities and not

arbitrary numbers, the iteration procedure leads to numerically unstable situations. Other methods such as bounded optimization must be employed if numerical results are required, but a presentation of the analysis was considered sufficient for the purposes of this work.

3.2 The Analytical Model

One assumes a finite population of M user terminals, each consisting of an input buffer, a transmission unit, and a retransmission unit, as shown in Figure 3.1. It is assumed that time is slotted, where the slot size is the end-to-end propagation delay τ . To simplify the analysis, all devices are assumed to be synchronized and are required to start packet transmission only at the beginning of a slot. The input buffer is assumed to provide space for an unlimited number of message packets. The arrival process to each user's input buffer is modelled as a Bernoulli process, with the probability of an arrival in a given slot equal to σ . The packet at the head of the input buffer's queue enters the transmitter unit one slot after the preceding packet's transmission has been successfully completed. Transmission then begins at the beginning of the next slot, effectively providing a two-slot interpacket gap so that a user does not monopolize the channel by unceasingly emptying its buffer. The packet remains in the transmitter until it is known whether the transmission has been successful.

The protocol under consideration here is commonly referred to as non-persistent carrier sense multiple access with collision detection (CSMA-CD). Each user terminal with a packet ready to transmit senses the channel prior to transmission. If the channel is sensed busy or if a collision occurs, the terminal is said to be blocked and the packet enters the retransmission unit. Transmission is rescheduled according to a geometrical distribution with mean $1/\nu$ slots; thus, the packet re-enters the transmission unit with probability ν at each slot. In the event of success, the terminal remains unblocked, where two substates are distinguished: idle, when the buffer is empty, and active, when the buffer is not empty. An active terminal sends the packet at the head of the buffer to the transmitter unit with probability one. A packet that arrives to an idle terminal is sent immediately to the transmitter unit without incrementing the buffer.

Given that a transmission is initiated on an empty channel, it requires at most one end-to-end propagation delay τ for the packet transmission to reach all other devices. When the terminals are not synchronized and free to transmit at any time, a collision can occur only if another device

initiates its transmission before the current transmission is sensed, in which case it will take at most one additional end-to-end delay before all devices receive the interference. Let ξ denote the time required for a device to detect a collision once an interference signal reaches it; ξ is clearly implementation dependent and, for example, has been stipulated in the first draft of the proposed IEEE 802 Local Networks Standards [9] to be no more than 9 bit times. Once the collision is detected, an interfering device aborts its transmission and jams the channel to ensure all other devices detect the collision. This collision consensus re-enforcement mechanism is implemented through the jam time ζ . The above considerations imply that, in the worst case, given that a collision occurs, the time required for all devices to stop transmission is

$$\gamma' = 2\tau + \xi + \zeta.$$

The time until all devices sense the channel idle is thus $\gamma' + \tau$. All times are now expressed in units of slots, so that, for example, γ' seconds is normalized to an integer number γ slots, and the time required for all devices to sense the channel idle is $\gamma + 1$ slots.

The state of any individual user terminal may be specified by the number of packets in its input buffer plus its status as blocked or unblocked. Clearly, the service time of a packet at a particular terminal is dependent on the status of the other queues. While the determination of the joint state probabilities of all M users remains a daunting problem, Saadawi and Ephremides [20] have suggested a technique of coupling the Markov chain describing the channel with the Markov chain describing a particular user. The result, though approximate, should provide insight into the actual behavior of the complete system.

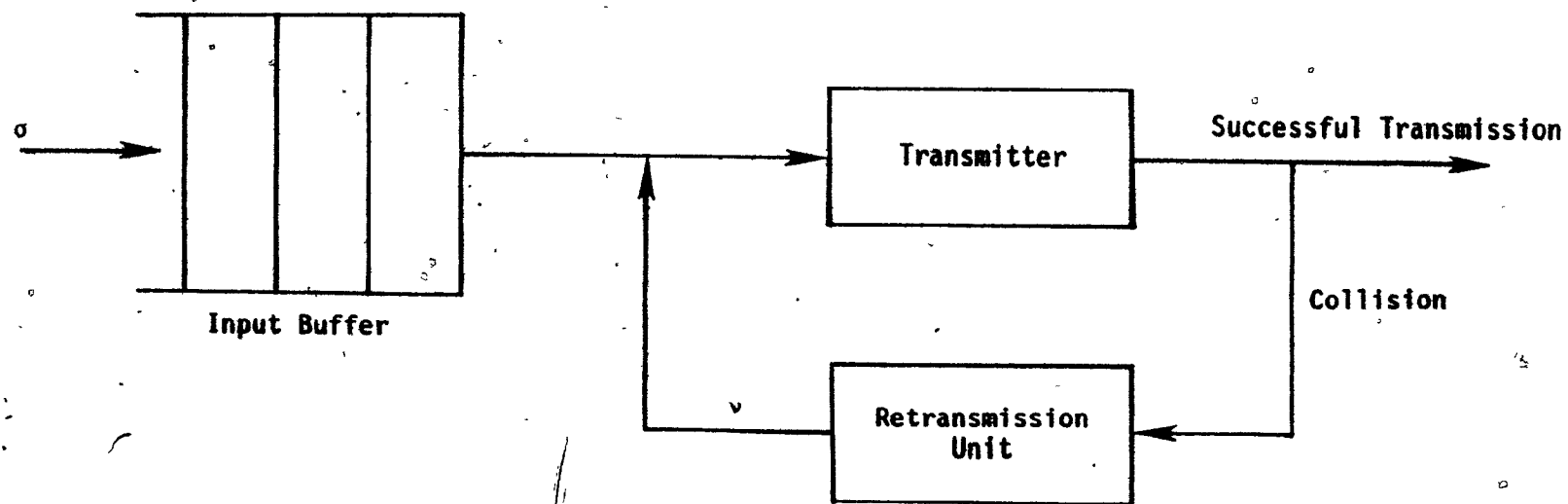


Figure 3.1 User Terminal Model

3.3 Analysis

The details of the analysis of CSMA-CD with a finite number of buffered users are developed in the following pages. Section 3.3.1 considers the channel Markov chain, constructing the necessary state transition matrix that leads to the stationary probabilities of the channel states. Analysis of the channel Markov chain leads to an expression for the steady-state network throughput, although numerical values for two particular parameters depend on the analysis of the user Markov chain, outlined in Section 3.3.2. Section 3.3.3 examines the user Markov chain on the basis of flow conservation, independently verifying the user Markov chain analysis; Section 3.3.4 then develops the equations for the parameters coupling the two Markov chains, making necessary approximations. Finally, Section 3.3.5 solves for the mean message delay in terms of values determined in the course of the analysis.

3.3.1 The Channel Markov Chain

The state of the channel is described by the number of terminals that are in each of the three states, blocked, active, and idle. Following the analysis of Tobagi and Hunt [6], one first observes that the channel experiences cycles of alternate idle and busy periods. A busy period ends one slot after either a successful transmission or a collision consensus reinforcement period; i.e., it ends once all stations again sense the channel inactive. An idle period then begins, lasting at least until the beginning of the next slot. The idle period will last only one slot if a device is active, however, since in that case the device has successfully completed a transmission in the last cycle and, with at least one packet remaining in its buffer, will attempt another transmission after an assumed two slot interpacket gap (as seen from its point of view). It is because of this guaranteed transmission attempt on the part of an active user that one distinguishes between active and idle devices.

Let m equal the number of active terminals and n equal the number of blocked terminals. Note that m may only be 0 or 1 since at most one successful transmission is possible in a cycle. The number of idle terminals may be found as $M-(n+m)$. Consequently, the state of the channel at any given slot may be described by (m,n) , where $n=0,1,\dots,M$ if $m=0$ and $n=0,1,\dots,M-1$ if $m=1$.

To accomplish the desired analysis, one considers the imbedded Markov chain identified by the slot immediately following the busy period, as shown in Figure 3.2. Although the model has yielded a two-dimensional finite state space defined by (m,n) , one may appropriately index the states so as to represent them by a one-dimensional vector. The transition probability matrix P between consecutive imbedded points may then be found, allowing the determination of the stationary probability distribution of the $2M+1$ states. P may be constructed in steps by examining the transitions leading to either a successful or unsuccessful transmission period plus the changes in the channel state that may occur during the busy period. These transitions are represented by the single-slot transition matrices S , F , QS , and QF , defined below. Additional information is required, however,

regarding the size of a terminal's input buffer contents in order to determine whether or not a terminal becomes or remains active. Let

$$E = \text{Pr} [\text{buffer size} = 1 \mid \text{user is active}], \text{ and}$$

$$E' = \text{Pr} [\text{buffer is empty} \mid \text{user is blocked}].$$

Consider the state transition diagram of Figure 3.3, which depicts the state transitions resulting in a transmission attempt. Construct the matrix S from the five distinct state transitions which lead to a successful transmission:

$$S1 \quad (1,n) \longrightarrow (0,n) \quad = \text{Probability that only the active terminal (with buffer contents} = 1) \text{ transmits}$$

$$= E (1 - \sigma)^{M-(n+1)} (1 - v)^n$$

$$\text{where } n = 0, 1, \dots, M-1$$

$$S2 \quad (0, n+1) \longrightarrow (0, n) \quad = \text{Probability that only one of } n+1 \text{ blocked terminals (with buffer contents} = 0) \text{ transmits, given that the busy period begins}$$

$$= \frac{E' (n+1) v (1 - v)^n (1 - \sigma)^{M-(n+1)}}{1 - (1 - v)^{n+1} (1 - \sigma)^{M-(n+1)}}$$

$$\text{where } n = 0, 1, \dots, M-1$$

S3

$(1,n) \rightarrow (1,n)$

= Probability that only the active terminal
(with buffer contents > 1) transmits

$$= (1 - E) (1 - v)^n (1 - \sigma)^{M-(n+1)}$$

where $n = 0, 1, \dots, M-1$

S4

$(0,n+1) \rightarrow (1,n)$

= Probability that only one of $n+1$ blocked
terminals (with buffer contents > 1)
transmits, given that the busy period
begins

$$= \frac{(1 - E') (n + 1) v (1 - v)^n (1 - \sigma)^{M-(n+1)}}{1 - (1 - v)^{n+1} (1 - \sigma)^{M-(n+1)}}$$

where $n = 0, 1, \dots, M-1$

S5

$(0,n) \rightarrow (0,n)$

= Probability that none of n blocked terminals
transmit, but one of $M-n$ idle stations
generates and successfully transmits a
message, given that the busy period begins

[note that the terminal remains idle]

$$= \frac{(1 - v)^n (M - n) \sigma (1 - \sigma)^{M-n-1}}{1 - (1 - v)^n (1 - \sigma)^{M-n}}$$

where $n = 0, 1, \dots, M-1$

All other terms of the S-matrix are equal to zero. The matrix F is similarly derived from the five state transitions resulting in a collision:

F1
 $(1, n-1) \rightarrow (0, n)$ = Probability that the active terminal and one or more of $n-1$ blocked terminals transmit

$$= [1 - (1 - v)^{n-1}] (1 - \sigma)^{M-n}$$

where $n = 2, \dots, M$

F2
 $(1, n-i) \rightarrow (0, n)$ = Probability that the active terminal transmits and packets are generated and transmitted at $i-1$ of $M-1-(n-i)$ idle terminals

$$= \binom{M+i-n-1}{i-1} \sigma^{i-1} (1 - \sigma)^{M-n}$$

where $n = 2, \dots, M$ and $i = 2, \dots, n$

F3
 $(0, n) \rightarrow (0, n)$ = Probability that more than one of n blocked terminals transmits, given that a busy period begins

$$= \frac{(1 - \sigma)^{M-n} [1 - (1 - v)^n - n(1 - v)^{n-1} v]}{1 - (1 - v)^n (1 - \sigma)^{M-n}}$$

where $n = 2, \dots, M$

F4
 $(0, n-1) \rightarrow (0, n)$

= Probability that more than one of $M-(n-1)$ idle terminals generate and transmit messages, given that the busy period begins

$$= \frac{\binom{M-(n-1)}{1} \sigma^1 (1-\sigma)^{M-n}}{1 - (1-\nu)^{n-1} (1-\sigma)^{M-(n-1)}}$$

where $n = 2, \dots, M$ and $i = 2, \dots, n$

F5
 $(0, n-1) \rightarrow (0, n)$

= Probability that one of $M-(n-1)$ idle terminals and at least one of $n-1$ blocked terminals transmit, given that the busy period begins

$$= \frac{[M-(n-1)] \sigma (1-\sigma)^{M-n} [1 - (1-\nu)^{n-1}]}{1 - (1-\nu)^{n-1} (1-\sigma)^{M-(n-1)}}$$

where $n = 2, \dots, M$

The matrices defined above are not, strictly speaking, transition matrices; the sum $S + F$, however, is a stochastic matrix and is the true transition matrix leading to a transmission period.

There is a distinction in the transitions of S - and F -matrices which must be noted. In the case of S -matrix transitions, a final state $(0, n)$ signifies the ongoing transmission of a user which, at the beginning of its transmission, has no packets in its input buffer and thus does not become active. The transmitting user is then one among the $M-n$ unblocked stations. During the transmission, however, subsequent message arrivals to this particular user will change its status to active so that the user will attempt, with probability one, a second transmission one slot immediately after the next imbedded point. Arrivals to any of the other $M-(n+1)$ idle stations during the transmission result in their becoming blocked as they sense the ongoing transmission and refrain from interfering. Assuming all

message packets to be of fixed duration T slots, then the $T + 1$ slots after an S -matrix transition may each be characterized by the slot transition matrix QS , where, from the above discussion,

$$QS$$

$$(0, n-i) \rightarrow (1, n-1)$$

- = Probability that i of $M-(n-i+1)$ unblocked stations (including the transmitting station) generate new messages
- = [Probability that i of $M-(n-i)$ unblocked stations generate new messages]
- x [Probability that the transmitting station generates a new message, given that i of $M-(n-i)$ unblocked stations generate new messages]

and

$$QS$$

$$(0, n-i) \rightarrow (0, n)$$

- = Probability that i of $M-(n-i)$ unblocked stations (not including the transmitting station) generate new messages
- = [Probability that i of $M-(n-i)$ unblocked stations generate new messages]
- x [Probability that the transmitting station does not generate a new message, given that i of $M-(n-i)$ unblocked stations generate new messages].

The probability that a message arrives to the transmitting station, given that arrivals occur at i of the $M-(n-i)$ unblocked users, is found through

the hypergeometric distribution [21]: the probability p_k that k of z identifiable elements are among a group of r elements chosen at random from a population of N is

$$p_k = \frac{\binom{z}{k} \binom{N-z}{r-k}}{\binom{N}{r}}$$

Here, then, $k=1$, $z=1$, $r=1$, and $N=M-(n-1)$, so that

$$\begin{aligned} \text{QS} \\ (0, n-1) \longrightarrow (1, n-1) &= \binom{M-(n-1)}{1} \sigma^1 (1-\sigma)^{M-(n+1)} \frac{1}{M-(n-1)} \\ &\text{where } n = 0, 1, \dots, M \text{ and } i = 0, \dots, n \end{aligned}$$

and

$$\begin{aligned} \text{QS} \\ (0, n-1) \longrightarrow (0, n) &= \binom{M-(n-1)}{1} \sigma^1 (1-\sigma)^{M-(n+1)} \frac{M-n}{M-(n-1)} \\ &\text{where } n = 0, 1, \dots, M-1 \text{ and } i = 0, \dots, n \end{aligned}$$

A final state $(1, n)$ identifies the transmitting user as active, leaving $M-(n+1)$ idle users, any of which will become blocked if they generate new messages during the transmission in progress. Hence,

$$\begin{aligned} \text{QS} \\ (1, n-1) \longrightarrow (1, n) &= \text{Probability that } i \text{ of } M-(n-i+1) \text{ idle} \\ &\quad \text{stations generate new messages} \\ &= \binom{M-(n-i+1)}{i} \sigma^i (1-\sigma)^{M-(n+1)} \\ &\text{where } n = 0, 1, \dots, M-1 \text{ and } i = 0, \dots, n \end{aligned}$$

F-matrix transitions, however, all end in state $(0, n)$, correctly reflecting the number of non-idle terminals at the end of the transition

slot as well as at the end of the transmission period. Each of the $\gamma+1$ slots of an unsuccessful transmission period thus has state transition matrix QF , where QF is given by

$$\begin{aligned}
 QF &= \text{Probability that } i \text{ of } M-(n-i) \text{ idle stations} \\
 (0, n-i) \longrightarrow (0, n) &\text{ generate new messages} \\
 &= \binom{M-(n-i)}{i} \sigma^i (1-\sigma)^{M-n} \\
 &\text{where } n = 0, \dots, M \text{ and } i = 0, \dots, n
 \end{aligned}$$

The QS - and QF -matrices are completed by providing transitions of probability one cycling on those states which do not change during the transition period:

$$\begin{aligned}
 QS &= 1, \\
 (0, M) \longrightarrow (0, M)
 \end{aligned}$$

$$\begin{aligned}
 QS &= 1, \text{ and} \\
 (1, M-1) \longrightarrow (1, M-1)
 \end{aligned}$$

$$\begin{aligned}
 QF &= 1, \text{ where } n = 0, \dots, M-1. \\
 (1, n) \longrightarrow (1, n)
 \end{aligned}$$

The above single-slot transition matrices allow one to construct the transition matrix P between consecutive imbedded points:

$$P = S QS^{T+1} + F QF^{T+1}.$$

Let the row vector π_C denote the stationary probability distribution of the $2M+1$ channel states, where, for example, π_C may be written as

$$\pi_C = [\pi_C(0,0), \pi_C(0,1), \dots, \pi_C(0,M), \pi_C(1,0), \dots, \pi_C(1,M-1)].$$

With the matrix P accordingly indexed, π_C is found through solution of the matrix equation $\pi_C = \pi_C P$.

Knowledge of π_C allows the computation of other system statistics. For example, the stationary distribution of the idle period is easily found. When a user is active, it attempts transmission immediately (i.e., one slot) after the channel is sensed idle by all users. The idle period is thus always at least one slot long, with the probability that the idle period is exactly one slot equal to the probability that a user in the system is active plus the probability that a transmission begins because either new messages arrive to one or more idle stations or one or more blocked users sense the idle channel and attempt transmission:

$$\text{Pr}[\text{idle period} = 1 \text{ slot}] =$$

$$\sum_{n=0}^{M-1} \pi_C(1, n) + \sum_{n=0}^M \pi_C(0, n) [1 - (1-v)^n (1-\sigma)^{M-n}].$$

The probability that the idle period is of duration $k > 1$ slots depends only on the actions of the idle and blocked stations:

$$\text{Pr}[\text{idle period} = k \text{ slots}] =$$

$$\sum_{n=0}^M \pi_C(0, n) [(1-v)^n (1-\sigma)^{M-n}]^{k-1} [1 - (1-v)^n (1-\sigma)^{M-n}].$$

The mean idle period duration is therefore

$$\bar{I} = \sum_{n=0}^M \pi_C(0, n) \frac{1}{[1 - (1-v)^n (1-\sigma)^{M-n}]} + \sum_{n=0}^{M-1} \pi_C(1, n) \text{ slots.}$$

From consideration of the five transitions leading to a successful transmission period, the probability of a successful transmission, P_S , is calculated as

$$P_S = \sum_{n=0}^M \pi_C(1,n) (1-\sigma)^{M-(n+1)} (1-\nu)^n$$

$$+ \sum_{n=0}^M \pi_C(0,n) \frac{\nu(1-\nu)^{n-1}(1-\sigma)^{M-n} + (1-\nu)^n(M-n)\sigma(1-\sigma)^{M-(n+1)}}{1 - (1-\nu)^n(1-\sigma)^{M-n}} .$$

allowing the busy period distribution to be expressed as

$$\text{Pr}[\text{busy period} = k+1 \text{ slots}] = P_S \delta(k-T) + (1-P_S) \delta(k-\gamma) ,$$

where, again, T is the fixed duration of a message in integer units of slots. The mean cycle time is simply the sum of the mean idle period, $\bar{I}$, and the mean busy period, $\bar{B}$, where $\bar{B}$ is obviously equal to $1 + P_S T + (1-P_S)\gamma$.

Since the state variables at each imbedded point define a regenerative process, the average stationary channel throughput, S , is readily computed as the ratio of the average time per cycle that the channel is successfully used to transmit a message to the average cycle length [22]; i.e.,

$$S = P_S T / (\bar{I} + \bar{B}).$$

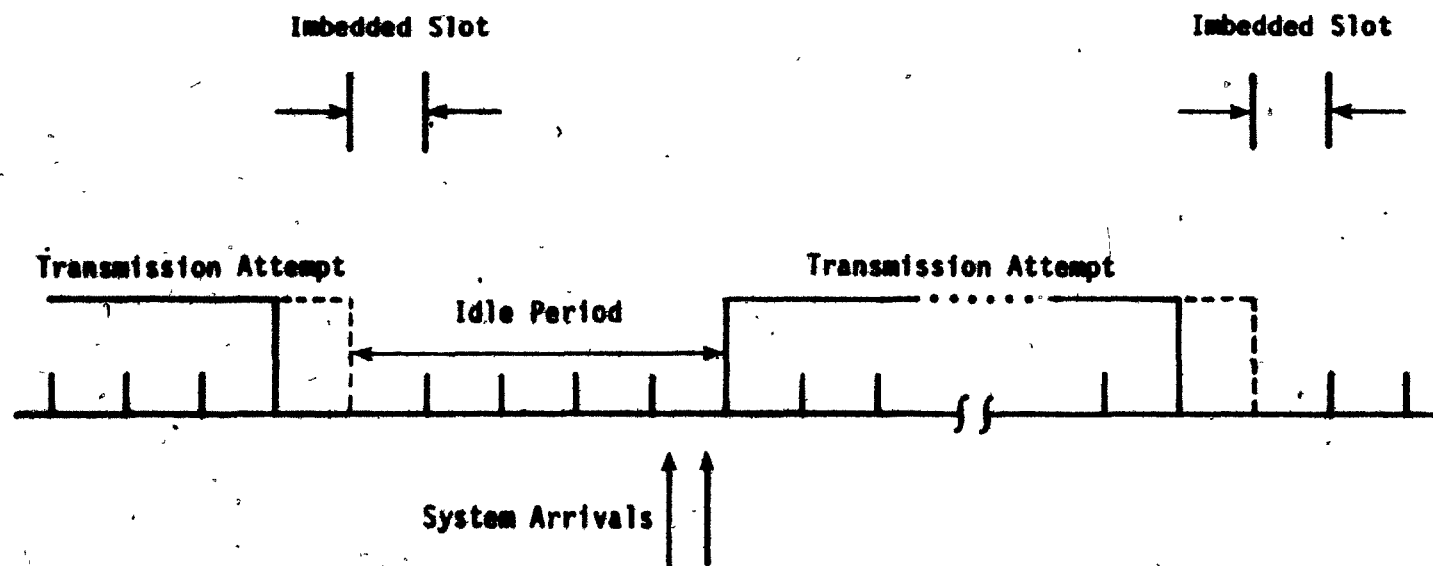


Figure 3.2 Imbedded Slots in the Channel Markov Chain

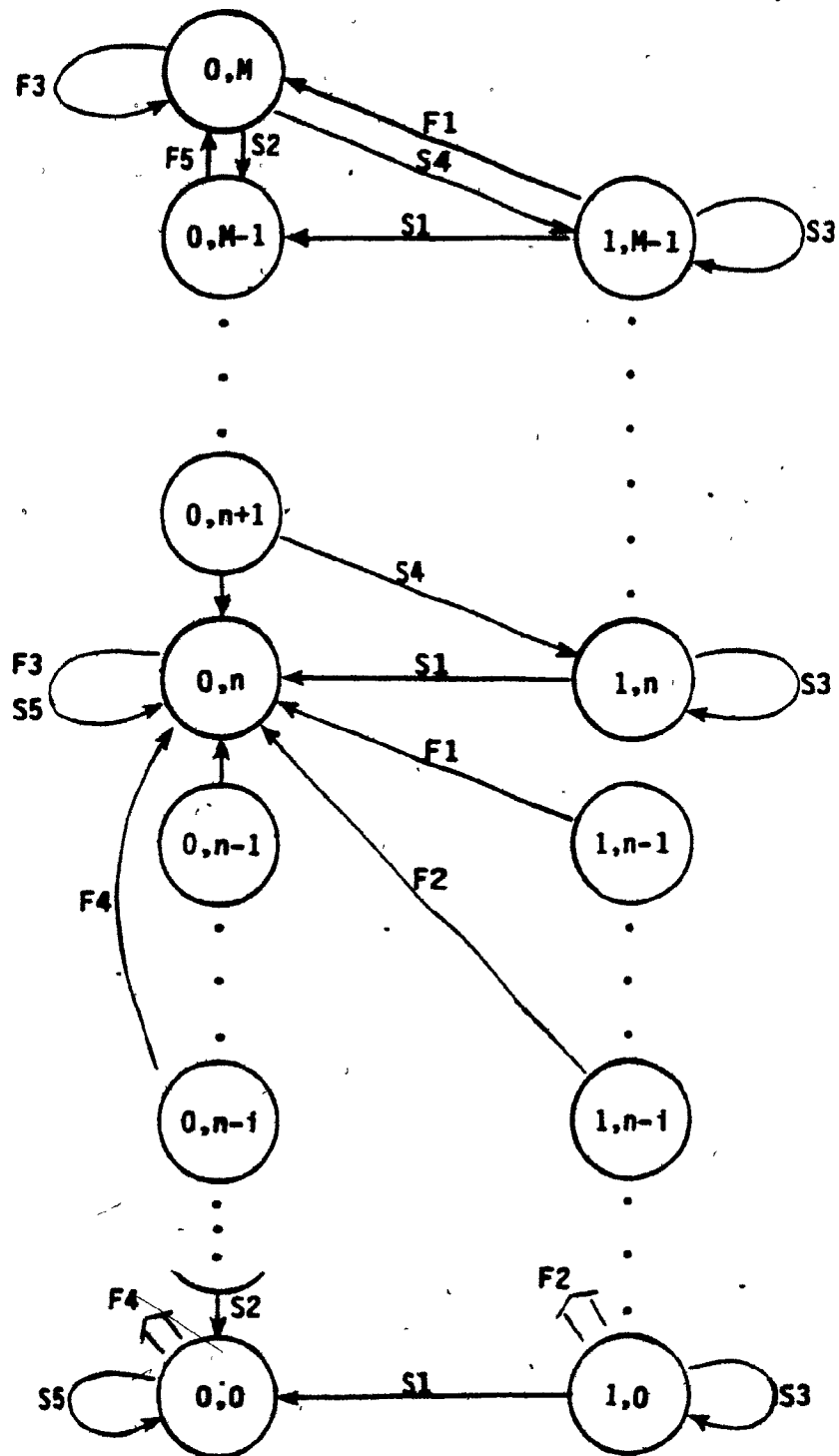


Figure 3.3 State Transition Diagram for the Channel Markov Chain

3.3.2 The User Markov Chain

The state of a particular terminal is described by the size of its buffer contents and by its status as blocked or unblocked. Let (i,j) denote the state of the user, where i equals 0 if the station is blocked or 1 if it is not blocked (either idle or active), and $j=0,1,2,\dots$ indicates the number of packets in the buffer. There is no limit to the possible values of j since one assumes the user to have an unlimited buffer capacity.

The user Markov chain, like the channel Markov chain, is defined by the imbedded points occurring at the end of each transmission attempt. From the analysis of the channel Markov chain, one has a probabilistic description of the idle and busy periods composing a cycle. The probability distribution of a cycle may thus be expressed as

$$\begin{aligned} \text{Pr} [\text{cycle} = n \text{ slots}] &= (1 - P_S) \text{Pr}[\text{idle period} = n-(\gamma+1) \text{ slots}] \\ &+ P_S \text{Pr}[\text{idle period} = n-(T+1) \text{ slots}]. \end{aligned}$$

The probability A_j that j messages arrive to a station during a cycle is therefore given by

$$A_j = \sum_{n=j}^{\infty} \binom{n}{j} \sigma^j (1-\sigma)^{n-j} \text{Pr}[\text{cycle} = n \text{ slots}], \text{ where } j = 0,1,\dots$$

Just as the probabilities E and E' of the channel Markov chain analysis express the influence of a user's state on channel transitions, so too must

additional information here be provided to reflect a user's performance in gaining access to the channel. Let

$$r = \text{Pr}[\text{successful transmission} | \text{user is blocked}],$$

$$q_1 = \text{Pr}[\text{collision} | \text{user is idle}],$$

and

$$q_2 = \text{Pr}[\text{collision} | \text{user is active}].$$

The state transition diagram of Figure 3.4 may then be constructed, keeping in mind the verbal description of the protocol and the respective roles of the input buffer, transmission unit, and retransmission unit. In particular, the packet currently attempting transmission is not counted as part of the input buffer contents; this distinction is made to allow separate delay calculations for the time a packet spends queueing in the input buffer and the time spent gaining access to the channel.

Let π_{ij} denote the steady-state probability of the state (i, j) . From Figure 3.4, the equilibrium equations are found:

$$\pi_{1,0} = A_0 r \pi_{0,0} + [A_0 + A_1(1-q_1)] \pi_{1,0} + A_0(1-q_2) \pi_{1,1}, \quad (1)$$

$$\pi_{1,i} = (1-q_2) \sum_{k=1}^{i+1} A_{i-k+1} \pi_{1,k} + (1-q_1) A_{i+1} \pi_{1,0} + r \sum_{k=0}^i A_{i-k} \pi_{0,k} \quad (2)$$

where $i=1,2,\dots$

and

$$\pi_{0,i} = (1-r) \sum_{k=0}^i A_{i-k} \pi_{0,k} + q_2 \sum_{k=1}^{i+1} A_{i-k+1} \pi_{1,k} + A_{i+1} q_1 \pi_{1,0} \quad (3)$$

where $i=0,1,\dots$

To continue the analysis, let $G_i(z)$ denote the z-transform of the probabilities $\{\pi_{ij}\}$; i.e., let

$$G_i(z) = \sum_{j=0}^{\infty} \pi_{ij} z^{-j}, \quad i=0 \text{ or } 1.$$

Consequently,

$$G_0(1) = \Pr[\text{user is blocked}] = \sum_{j=0}^{\infty} \pi_{0,j},$$

$$G_1(1) = \Pr[\text{user is not blocked}] = \sum_{j=0}^{\infty} \pi_{1,j}.$$

and

$$G_0(1) + G_1(1) = 1. \quad (4)$$

The distinction between idle and active unblocked users made in the channel Markov chain analysis may also be made here:

$$\Pr[\text{user is idle}] = \pi_{1,0}$$

and

$$\Pr[\text{user is active}] = G_1(1) - \pi_{1,0}.$$

Applying the operator $\sum_{i=1}^{\infty} z^{-i}$ to equation (2) yields:

$$\begin{aligned} G_1(z) - \pi_{1,0} &= (1 - q_2) \left[A(z) z (G_1(z) - \pi_{1,0}) - A_0 \pi_{1,1} \right] \\ &+ (1 - q_1) \pi_{1,0} z \left[A(z) - A_1 z^{-1} - A_0 \right] + r \left[A(z) G_0(z) - A_0 \pi_{0,0} \right]. \end{aligned}$$

where $A(z)$ is the z -transform of the probabilities A_j ,

$$A(z) = \sum_{j=0}^{\infty} A_j z^{-j}.$$

Adding equation (1) to the above results in

$$\begin{aligned} G_1(z) = & (1 - q_2) A(z) z [G_1(z) - \pi_{1,0}] + r A(z) G_0(z) \\ & + (1 - q_1) \pi_{1,0} z [A(z) - A_0] + A_0 \pi_{1,0} \end{aligned} \quad (5)$$

In a similar manner, application of the operator $\sum_{z=0}^{\infty} z^{-k}$ to equation (3) gives

$$G_0(z) = (1-r) A(z) G_0(z) + q_2 A(z) z [G_1(z) - \pi_{1,0}] + \pi_{1,0} q_1 z [A(z) - A_0] \quad (6)$$

By differentiating equations (5) and (6) with respect to z , evaluating at $z=1$, and then adding the results together, one obtains the equation

$$[A'(1) + 1] G_1(1) + A'(1) G_0(1) - A_0 \pi_{1,0} = 0. \quad (7)$$

A final equation is found by evaluating either equation (5) or (6) at $z=1$:

$$r G_0(1) = q_2 [G_1(1) - \pi_{1,0}] + \pi_{1,0} q_1 (1 - A_0). \quad (8)$$

Equations (4), (7), and (8) may then be solved for $G_0(1)$, $G_1(1)$, and $\pi_{1,0}$:

$$G_0(1) = \frac{[A'(1) + 1][q_1(1 - A_0) - q_2] + q_2 A_0}{r A_0 + (1 - A_0)(q_1 - q_2)}, \quad (9)$$

$$G_1(1) = \frac{r A_0 - A'(1)[q_1(1 - A_0) - q_2]}{r A_0 + (1 - A_0)(q_1 - q_2)}, \quad (10)$$

and

$$\pi_{1,0} = \frac{q_2 A'(1) + r[A'(1) + 1]}{r A_0 + (1 - A_0)(q_1 - q_2)}. \quad (11)$$

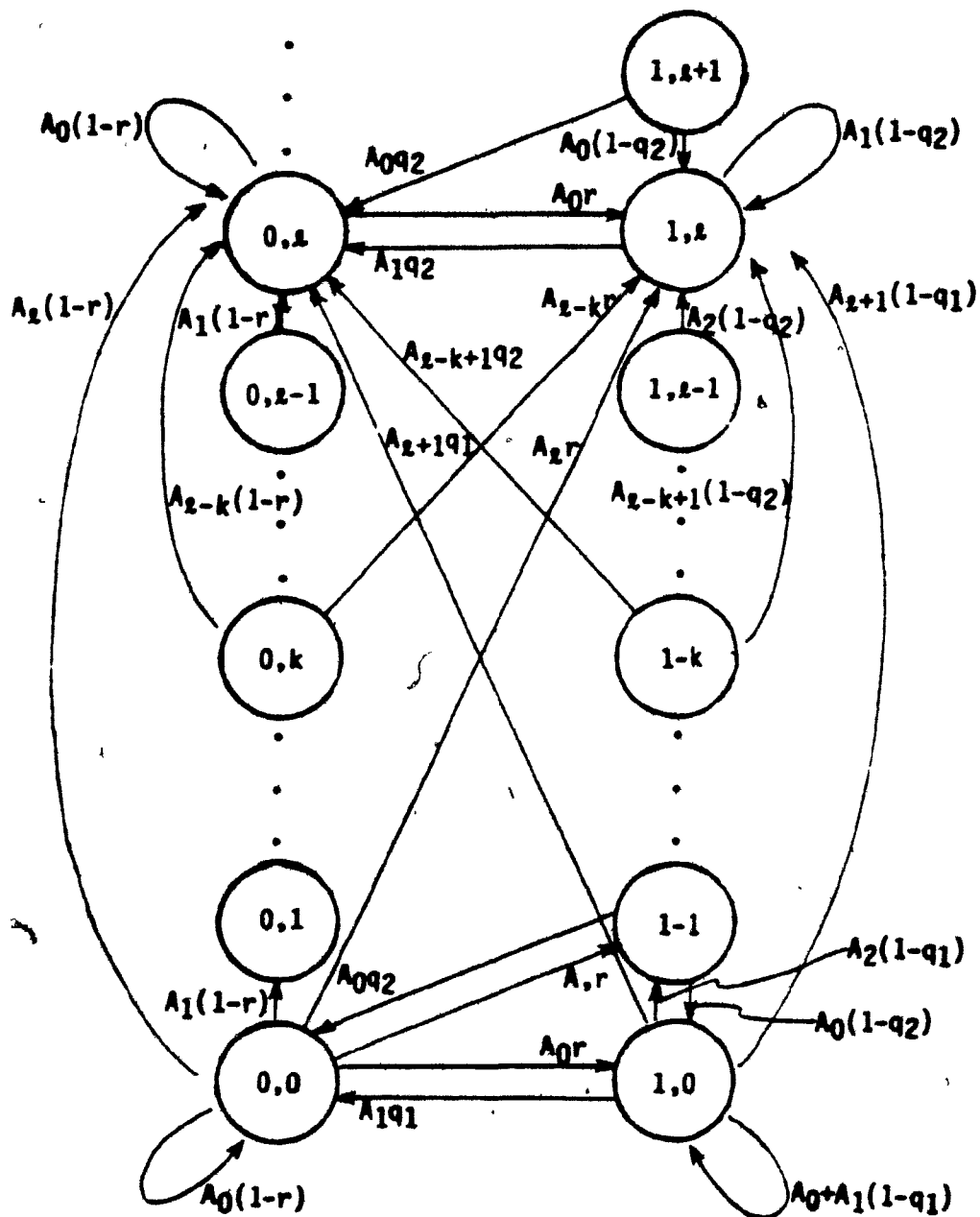


Figure 3.4 State Transition Diagram for the User Markov Chain

3.3.3 Stability Considerations and User Markov Chain Verification

The model constructed here parallels that of many queueing systems in the sense that message packets are buffered in a queue at each user awaiting service. A queueing system is said to be stable when, under steady-state conditions, packets suffer finite duration mean delays. Stability of the current model can be investigated by inspecting flow conservation equations [2].

If one considers an entire terminal — input buffer, transmission unit, and retransmission unit — to constitute a system in itself, then flow conservation implies that the average flow into a terminal per cycle must be equal to the average flow out per cycle. The average flow into a terminal per cycle is the mean number of arrivals per cycle, $\bar{A}$, where

$$\bar{A} = -A'(1) = \sum_{j=0}^{\infty} j A_j.$$

The average flow out of a terminal per cycle is given by

$$\begin{aligned} & 1 \times \Pr \left[\begin{array}{c} \text{successful} \\ \text{transmission} \end{array} \middle| \text{user is active} \right] \times \Pr[\text{user is active}] + \\ & 1 \times \Pr \left[\begin{array}{c} \text{successful} \\ \text{transmission} \end{array} \middle| \text{user is blocked} \right] \times \Pr[\text{user is blocked}] + \\ & 1 \times \Pr[\text{at least one arrival}] \\ & \times \Pr \left[\begin{array}{c} \text{successful} \\ \text{transmission} \end{array} \middle| \text{user is idle} \right] \times \Pr[\text{user is idle}] \end{aligned}$$

Flow equilibrium thus requires that

$$\bar{A} = (1 - q_2) [G_1(1) - \pi_{1,0}] + r G_0(1) + (1 - A_0) (1 - q_1) \pi_{1,0}. \quad (12)$$

Alternatively, if one considers just a user's input buffer as a queueing system, then the average flow into the system per cycle is again $\bar{A}$, but the average flow out per cycle is given by

$$1 \times \text{Pr}[\text{at least one arrival}] \times \text{Pr}[\text{user is idle}] \\ + 1 \times \text{Pr}[\text{user is active}] + 0 \times \text{Pr}[\text{user is blocked}] .$$

Flow equilibrium here requires that

$$\bar{A} = (1 - A_0) \pi_{1,0} + G_1(1) - \pi_{1,0} \quad (13)$$

Equations (12) and (13) define stability regions for the independent system parameter values M , σ , and v , although the complexity of the model obscures the relationship. Nonetheless, these equations, together with equation (6), may be solved for $G_0(1)$, $G_1(1)$, and $\pi_{1,0}$, yielding

$$G_0(1) = \frac{[A'(1) + 1] [q_1(1 - A_0) - q_2] + q_2 A_0}{r A_0 + (1 - A_0) (q_1 - q_2)} ,$$

$$G_1(1) = \frac{r A_0 - A'(1) [q_1(1 - A_0) - q_2]}{r A_0 + (1 - A_0) (q_1 - q_2)} ,$$

and

$$\pi_{1,0} = \frac{q_2 A'(1) + r [A'(1) + 1]}{r A_0 + (1 - A_0) (q_1 - q_2)} .$$

The above results are, of course, identical to those obtained in the user Markov chain analysis, thus providing an independent verification.

3.3.4 Coupling of the Markov Chains

The parameters E and E' appearing in the channel Markov chain and r , q_1 , and q_2 of the user Markov chain analysis provide links between the two Markov chains; the value of a parameter of one particular chain is found in terms of quantities derived from the other.

Consider the parameter r , previously defined as the probability a user's transmission is successful, given that it is blocked. One's viewpoint is from that of a particular user; e.g., user A, who is the subject of the user Markov chain analysis. Thus

$$r = \sum_{n=1}^M \Pr \left[\begin{array}{c} \text{user A makes} \\ \text{a successful} \\ \text{transmission} \end{array} \middle| \begin{array}{c} \text{channel} \\ \text{state} \\ (0,n) \end{array}, \text{user A is} \right. \left. \Pr \left[\begin{array}{c} \text{channel} \\ \text{state} \\ (0,n) \end{array} \middle| \text{user A is} \right. \right]$$

Channel states $(1,n)$ are not involved in the above expression since in that case the active user transmits with probability one, implying that $r=0$. The conditional probability of user A's successful transmission given that the channel state is $(0,n)$ and user A is blocked is simply $\frac{1}{n} v(1-v)^{n-1} (1-\sigma)^{M-n}$. Observe that the other term appearing in the summation may be rewritten as

$$\Pr \left[\begin{array}{c} \text{channel} \\ \text{state} \\ (0,n) \end{array} \middle| \text{user A is} \right. \left. \Pr \left[\begin{array}{c} \text{user A is} \\ \text{blocked} \end{array} \middle| \begin{array}{c} \text{channel state} \\ (0,n) \end{array} \right] \right] = \frac{\pi_C(0,n) \Pr \left[\begin{array}{c} \text{user A is} \\ \text{blocked} \end{array} \middle| \begin{array}{c} \text{channel state} \\ (0,n) \end{array} \right]}{\Pr[\text{user A is blocked}]}$$

The probability that user A is blocked given that the channel state is $(0,n)$ is found with the help of the hypergeometric distribution cited previously, where here $k=1$, $i=1$, $r=n$, and $N=M$:

$$\Pr \left[\begin{array}{c} \text{user A is} \\ \text{blocked} \end{array} \middle| \begin{array}{c} \text{channel state} \\ (0,n) \end{array} \right] = \frac{\binom{1}{1} \binom{M-1}{n-1}}{\binom{M}{n}} = \frac{n}{M}$$

To find the last unknown probability, an approximation must be made: assume that the probabilities that each user is blocked are statistically independent. Consequently, one may make the following observation:

$$\Pr \begin{bmatrix} \text{no stations} \\ \text{are blocked} \end{bmatrix} = \pi_C(0,0) + \pi_C(1,0) = [1 - \Pr[\text{user A is blocked}]]^M.$$

The unconditioned probability that user A is blocked is thus

$$\Pr[\text{user A is blocked}] = 1 - [\pi_C(0,0) + \pi_C(1,0)]^{1/M}.$$

As a result, r is approximated as

$$r = \sum_{n=1}^M \frac{v(1-v)^{n-1} (1-\sigma)^{M-n} \frac{1}{M} \pi_C(0,n)}{1 - [\pi_C(0,0) + \pi_C(1,0)]^{1/M}}.$$

In a similar fashion, one derives q_1 as follows:

$$q_1 = \Pr[\text{collision} | \text{user is idle}]$$

$$= 1 - \Pr[\text{successful} | \text{user is idle}]$$

$$= 1 - \sum_{n=0}^{M-1} \Pr \left[\begin{array}{c|c} \text{user A can make} & \text{channel} \\ \text{a successful} & \text{state} \\ \text{transmission} & (0,n) \end{array} \middle| \begin{array}{c} \text{user} \\ \text{is} \\ \text{idle} \end{array} \right] \Pr \left[\begin{array}{c|c} \text{channel} & \text{user} \\ \text{state} & \text{is} \\ (0,n) & \text{idle} \end{array} \right].$$

Note that $n=M$ is not considered since there is no idle user in that case. The conditional probability that user A can make a successful transmission given that the channel state is $(0,n)$ and user A is idle is equal to the probability that the n blocked stations do not sense the channel and attempt a transmission and that the remaining $M-(n+1)$ users other than A do not generate messages:

$$\Pr \left[\begin{array}{c|c} \text{user A can make} & \text{channel} \\ \text{a successful} & \text{state} \\ \text{transmission} & (0,n) \end{array} \middle| \begin{array}{c} \text{user} \\ \text{A is} \\ \text{idle} \end{array} \right] = (1-v)^n (1-\sigma)^{M-(n+1)} \left(\frac{1}{M-n} \right).$$

The conditional probability that the channel state is (0,n) given that A is idle is first rewritten as

$$\Pr \left[\begin{array}{c|c} \text{channel state} & \text{user A is} \\ (0,n) & \text{idle} \end{array} \right] = \frac{\pi_C(0,n) \Pr \left[\begin{array}{c|c} \text{user A} & \text{channel state} \\ \text{is idle} & (0,n) \end{array} \right]}{\Pr[\text{user A is blocked}]}$$

where

$$\Pr \left[\begin{array}{c|c} \text{user A} & \text{channel state} \\ \text{is idle} & (0,n) \end{array} \right] = \frac{(M-n)}{M}$$

and the unconditional probability that user A is idle is found through the following approximation:

$$\Pr \left[\begin{array}{c} \text{no station} \\ \text{is idle} \end{array} \right] = \pi_C(0,M) + \pi_C(1,M-1) + [1 - \Pr[\text{user A is idle}]]^M.$$

The result is thus

$$q_1 = 1 - \frac{\sum_{n=0}^{M-1} (1-\nu)^n (1-0)^{M-(n+1)} \frac{1}{M} \pi_C(0,n)}{1 - (\pi_C(0,M) + \pi_C(1,M-1))^{1/M}}.$$

Finally, the last of the three user Markov chain parameters, q_2 , is found through the following reasoning:

$$q_2 = \Pr[\text{collision} | \text{user A is active}]$$

$$= 1 - \Pr[\text{success} | \text{user A is active}]$$

$$= 1 - \sum_{n=0}^{M-1} \Pr \left[\begin{array}{c} \text{user A makes} \\ \text{a successful} \\ \text{transmission} \end{array} \middle| \begin{array}{c} \text{channel} \\ \text{state} \\ (1,n) \end{array}, \text{user A is active} \right] \Pr \left[\begin{array}{c} \text{channel} \\ \text{state} \\ (1,n) \end{array} \middle| \text{user A is active} \right]$$

$$= 1 - \sum_{n=0}^{M-1} (1-v)^n (1-\sigma)^{M-(n+1)} \Pr \left[\begin{array}{c} \text{channel} \\ \text{state} \\ (1,n) \end{array} \middle| \text{user A is active} \right].$$

The conditional probability that the channel state is $(1,n)$ given that user A is active is restated as

$$\Pr \left[\begin{array}{c} \text{channel} \\ \text{state} \\ (1,n) \end{array} \middle| \text{user A is active} \right] = \frac{\pi_C(1,n) \Pr \left[\begin{array}{c} \text{user A} \\ \text{is active} \end{array} \middle| \begin{array}{c} \text{channel} \\ \text{state} \\ (1,n) \end{array} \right]}{\Pr[\text{user A is active}]}$$

where

$$\Pr[\text{user A is active} | \text{channel state } (1,n)] = 1/M$$

and the unconditional probability that user A is active results from the following approximation:

$$\Pr[\text{no active user}] = 1 - \sum_{n=0}^{M-1} \pi_C(1,n) = [1 - \Pr[\text{user A is active}]]^M,$$

implying that

$$\text{Pr}(\text{user A is active}) = 1 - \left[1 - \sum_{n=0}^{M-1} \pi_C(1,n) \right]^{1/M}.$$

The parameter q_2 is thus given by

$$q_2 = 1 - \frac{\sum_{n=0}^{M-1} (1-\nu)^n (1-\sigma)^{M-(n+1)} \pi_C(1,n) \frac{1}{M}}{1 - \left[1 - \sum_{n=0}^{M-1} \pi_C(1,n) \right]^{1/M}}.$$

In the above expressions for r , q_1 , and q_2 , the following approximations may be made. For the denominator of r , for example,

$$\pi_C(0,0) + \pi_C(1,0) = 1 - M \text{Pr}(\text{user A is blocked}) + \dots,$$

implying that

$$\text{Pr}(\text{user A is blocked}) = \frac{1}{M} \left[1 - (\pi_C(0,M) + \pi_C(1,0)) \right].$$

Similarly,

$$\text{Pr}(\text{no station is idle}) = \frac{1}{M} \left[1 - (\pi_C(0,M) + \pi_C(1, M-1)) \right]$$

and

$$\text{Pr}(\text{user A is active}) = \frac{1}{M} \left[1 - \left(1 - \sum_{n=0}^{M-1} \pi_C(1,n) \right) \right].$$

The appropriate expressions for r , q_1 , and q_2 may thus be written as

$$r = \frac{\sum_{n=1}^M v(1-v)^{n-1} (1-\sigma)^{M-n} \pi_C(0,n)}{1 - (\pi_C(0,0) + \pi_C(1,0))}$$

$$q_1 = 1 - \frac{\sum_{n=0}^{M-1} (1-v)^n (1-\sigma)^{M-(n+1)} \pi_C(0,n)}{1 - (\pi_C(0,M) + \pi_C(1,M-1))}$$

and

$$q_2 = 1 - \frac{\sum_{n=0}^{M-1} (1-v)^n (1-\sigma)^{M-(n+1)} \pi_C(1,n)}{\sum_{n=0}^{M-1} \pi_C(1,n)}$$

These approximations are, of course, the same as used by Saadawi and Ephremides [20].

The parameters E and E' of the channel Markov chain are easily derived in terms of the quantities obtained from the user Markov chain:

$$E = \text{Pr}[\text{buffer size} = 1 | \text{user is active}] = \frac{\pi_{1,1}}{G_1(1) - \pi_{1,0}}$$

and

$$E' = \text{Pr}[\text{buffer is empty} | \text{user is blocked}] = \frac{\pi_{0,0}}{G_0(1)}$$

To find the values of $\pi_{1,1}$ and $\pi_{0,0}$, first extract the following equation from equation (3) by setting $z=1$:

$$\pi_{1,0} = A_0 (1-r) \pi_{0,0} + A_1 q_1 \pi_{1,0} + A_0 q_2 \pi_{1,0}.$$

Solving the above equation and equation (1) for $\pi_{1,1}$ and $\pi_{0,0}$ yields

$$\pi_{1,1} = \frac{\pi_{1,0} [1 + A_0 r - 2A_0 + A_0^2 (1-r) + A_0 A_1 (1-q_1-r) - A_1 (1-q_1)]}{A_0 (1-q_2) - A_0^2 (1-q_2-r)}$$

and

$$\pi_{0,0} = \frac{\pi_{1,0} [A_0 q_2 (1-A_0) + A_0 A_1 (q_1-q_2)]}{A_0 (1-q_2) - A_0^2 (1-q_2-r)},$$

thereby allowing E and E' to be evaluated. One is thus dealing with a set of simultaneous coupled nonlinear equations since, as shown above, the quantities E and E' depend on q_1 , q_2 , and r and are required in the channel Markov chain analysis to find the channel state probabilities $\pi_C(m,n)$, in turn, are needed to determine q_1 , q_2 , and r .

3.3.5 Delay Analysis

In many analyses of queueing systems, the waiting time distribution of a customer in the system cannot be obtained, yet the mean waiting time may be found via the fundamental queueing formula $L = \lambda W$, where L is the average number of customers in the system, λ the mean arrival rate, and W the desired mean waiting time. This relation is often identified as Little's result, since it was Little [23] who first provided a general proof free of specific assumptions about the arrival and service distributions, the queue discipline, the independence of interarrival times, etc. The only assumption to his proof is the strict stationarity of all the involved processes. Jewell [24] later gave a somewhat more general proof assuming only the existence of regeneration points in the arrival and waiting time processes, but, finally, Stidham [25] has proved the relation assuming only that the relevant limiting averages exist. The mathematical legitimacy of an intuitive and often heuristically proven relation is thus secure.

In the context of the present analysis, the average total delay per packet, D , is the sum of the average waiting time in a user's input buffer, W_B , plus the average time elapsed from the first attempted transmission up to and including the final successful one, W_S . The waiting time W_B is easily found by Little's result. The average number $\bar{N}$ in the input buffer is obtained from the user Markov chain analysis as

$$\bar{N} = - \left. \frac{d}{dz} G_0(z) \right|_{z=1} - \left. \frac{d}{dz} G_1(z) \right|_{z=1} .$$

The required derivatives are most easily obtained by examining the two equations that result by differentiating equation (5) once and the sum of

equations (5) and (6) twice with respect to z , evaluating at $z=1$. Solving for $G'(1)$ and $G'(1)$ yields

$$\bar{N} = \left\{ (q_2 + r) \left[A''(1) G_0(1) + (A''(1) + 2A'(1)G_1(1)) \right] \right. \\ \left. - 2 \left[(1-r) A'(1) G_0(1) + q_2 A'(1) (G_1(1) - \pi_{1,0}) \right] \right. \\ \left. + q_2 G_1(1) - \pi_{1,0} + \pi_{1,0} q_1 (1 - A_0) + \pi_{1,0} q_1 A'(0) \right\} \\ \times \frac{1}{2r(1 + A'(1)) + 2q_2 A'(1)}$$

The mean arrival rate is here $\bar{A}/(\bar{B} + \bar{T})$, the mean number of arrivals per cycle divided by the mean cycle duration, so that

$$W_B = \frac{\bar{N} (\bar{B} + \bar{T})}{\bar{A}}$$

The analysis of Saddawi and Ephremides [20], modified somewhat here, provides a means to determine W_B . Consider first the number of cycles (idle and busy periods) required for success given that the first cycle resulted in a collision. The probability that success is achieved in the k -th cycle given that the user was blocked is $(1-r)^{k-1}r$, by the assumed statistical independence of cycles. Let the random variable C^F represent the length, in slots, of a cycle resulting in failure; C^S the length of a cycle resulting

in success. The z-transform of the number of cycles required for success given that the first attempt ends in failure is thus

$$\begin{aligned}
 S(z) &= E \left[z^{-(C_1^F + \dots + C_{k-1}^F + C_k^S)} \right] \\
 &= \sum_{k=1}^{\infty} \Pr(K=k) E \left[z^{-(C_1^F + \dots + C_{k-1}^F + C_k^S)} \middle| K=k \right] \\
 &= \sum_{k=1}^{\infty} \Pr(K=k) C_F^{k-1}(z) C_S(z) \\
 &= \sum_{k=1}^{\infty} r(1-r)^{k-1} C_F^{k-1}(z) C_S(z) \\
 &= \frac{r C_S(z)}{1 - (1-r) C_F(z)} ,
 \end{aligned}$$

where $C_F(z)$ and $C_S(z)$ denote the z-transforms of the random variables C^F and C^S , respectively.

The probability that a cycle lasts n slots, given that it ultimately ends in a failed transmission, is equivalent to the probability that the

idle period is of length $n - (\gamma + 1)$. The z-transform $C_F(z)$ is therefore given by

$$\begin{aligned}
 C_F(z) &= \sum_{m=1}^{\infty} z^{-(\gamma+1)} z^{-m} \Pr\{\text{idle period} = m\} \\
 &= z^{-(\gamma+1)} \left[z^{-1} \sum_{n=0}^{M-1} \pi_C(1, n) + \right. \\
 &\quad \left. \sum_{m=1}^{\infty} z^{-m} \sum_{n=0}^M \pi_C(0, n) \left[(1-\nu)^n (1-\sigma)^{M-n} \right]^{m-1} \times \left[1 - (1-\nu)^n (1-\sigma)^{M-n} \right] \right] \\
 &= z^{-(\gamma+1)} \left[z^{-1} \sum_{n=0}^M \pi_C(1, n) \right. \\
 &\quad \left. + \sum_{n=0}^M \pi_C(0, n) \frac{1 - (1-\nu)^n (1-\sigma)^{M-n}}{(1-\nu)^n (1-\sigma)^{M-n}} \underbrace{\sum_{m=1}^{\infty} z^{-m} \left[(1-\nu)^n (1-\sigma)^{M-n} \right]^{m-1}}_{\frac{(1-\nu)^n (1-\sigma)^{M-n} z^{-1}}{1 - (1-\nu)^n (1-\sigma)^{M-n}}} \right].
 \end{aligned}$$

so that

$$C_F(z) = z^{-(\gamma+1)} \left[z^{-1} \sum_{n=0}^M \pi_C(1, n) + \sum_{n=0}^M \pi_C(0, n) \left[\frac{1 - (1-\nu)^n (1-\sigma)^{M-n}}{z - (1-\nu)^n (1-\sigma)^{M-n}} \right] \right].$$

In a similar manner, the z-transform $C_S(z)$ is found as:

$$C_S(z) = \sum_{m=1}^{\infty} z^{-(T+1)} z^{-m} \text{Pr}[\text{idle period} = m]$$

$$= z^{-(T+1)} \left[z^{-1} \sum_{n=0}^M \pi_C(1,n) + \sum_{n=0}^M \pi_C(0,n) \left[\frac{1 - (1-v)^n (1-\sigma)^{M-n}}{z - (1-v)^n (1-\sigma)^{M-n}} \right] \right]$$

The mean values of C^F and C^S may be obtained from the z-transforms $C_F(z)$ and $C_S(z)$ as

$$\bar{C}^F = - \frac{d}{dz} C_F(z) \Big|_{z=1} = \sum_{m=1}^{\infty} (\gamma + 1 + m) z^{-m-\gamma-1} \Big|_{z=1} \text{Pr}[\text{idle period} = m]$$

$$= \gamma + 1 + \bar{I}$$

and

$$\bar{C}^S = \dots = T + 1 + \bar{I}.$$

where $\bar{I}$ is the mean idle period duration.

The mean number of slots $\bar{S}$ required for success given that the first attempt ended in failure may now be derived as follows:

$$\bar{S} = - \frac{d}{dz} S(z) \Big|_{z=1} = - \frac{d}{dz} \left[\frac{r C_S(z)}{1 - (1-r) C_F(z)} \right] \Big|_{z=1}$$

$$= - \frac{r^2 C_S'(z) + r(1-r) C_F'(z)}{r^2}$$

$$= \bar{C}^S + \frac{1}{r} (1-r) \bar{C}^F$$

To remove the conditioning, consider P_{coll} , the probability of a first collision occurring:

$$\begin{aligned}
 P_{\text{coll}} &= \Pr[\text{user is active} | \text{user is not blocked}] \Pr[\text{collision} | \text{user is active}] \\
 &\quad + \Pr[\text{user is idle} | \text{user is not blocked}] \Pr[\text{collision} | \text{user is idle}] \Pr[\text{arrival to user}] \\
 &= \frac{G_1(1) - \pi_{1,0}}{G_1(1)} q_2 + \frac{\pi_{1,0}}{G_1(1)} q_1 \sigma .
 \end{aligned}$$

The condition that the user not be blocked provides that the collision occur on the first attempt. In the event that a successful transmission occurs on the first attempt, the mean elapsed time is simply C^S , with P_{succ} , the probability of success on the first attempt, given by

$$\begin{aligned}
 P_{\text{succ}} &= \Pr[\text{user is active}] \Pr[\text{success} | \text{user is active}] \\
 &\quad + \Pr[\text{user is idle}] \Pr[\text{success} | \text{user is idle}] \Pr[\text{arrival to user}] \\
 &= \frac{G_1(1) - \pi_{1,0}}{G_1(1)} (1 - q_2) + \frac{\pi_{1,0}}{G_1(1)} (1 - q_1) \sigma .
 \end{aligned}$$

Combining terms, W_s , the mean elapsed time from the first attempt up to and including the ultimate successful transmission, is found to be

$$W_s = \bar{S} P_{\text{coll}} + \bar{C}^S P_{\text{succ}}$$

$$= \frac{G_1(1) - \pi_{1,0}}{G_1(1)} \left[q_2 \frac{1}{r} (1 - r) \bar{C}^F + \bar{C}^S \right]$$

$$+ \frac{\pi_{1,0}}{G_1(1)} \left[\bar{C}^S + q_1 \frac{1}{r} (1 - r) \bar{C}^F \right]$$

4. Conclusion

In conclusion, this study has presented a brief review of the current status of efforts to standardize local area networks and has assessed the performance of selected access control protocols through the techniques of computer simulation and mathematical analysis. Emphasis has been placed both on the employed techniques themselves and on the resulting assessment of access control protocol performance.

The simulations of Ethernet and decentralized adaptive probing model these protocols in detail; in the case of decentralized adaptive probing, the algorithm has been developed for the first time through this simulation study. The simulation methods used here to obtain performance results — regenerative simulation, sequential and/or fixed stopping rules, and ratio estimation with the jackknife estimator — are generally applicable and may be used, for example, in the simulation of a variety of queueing systems.

Access control protocols, of course, may be seen as queueing systems and, besides being simulated, may be modelled in such a way as to allow their analysis by the techniques of queueing theory. The technique used here to solve for the throughput and mean waiting time of CSMA-CD with a finite number of buffered users involves the coupling of two Markov chains describing activity for a particular user and for the communications channel itself.

While the analysis of CSMA-CD was not extended to produce numerical results, the simulation studies of Ethernet and decentralized adaptive probing yield point estimates and their associated confidence intervals for the following network performance indicators as functions of traffic intensity (system load): network throughput, mean waiting time, mean queue size, mean rejection rate, mean failure rate, and mean number of attempts required for successful transmission. Simulation allows the detailed modelling of these two protocols, permitting the measurement of quantities not generally accessible through analysis. Here, for example, detailed simulation of Ethernet's CSMA-CD access control protocol reveals network throughput almost approaching unity at high load levels, yet the analyses of CSMA-CD appearing in the literature rarely, if ever, indicate such performance. Similarly, the mean waiting time has been shown to be dependent on the buffer size at

each station until the buffer size becomes greater than three when performance no longer shows such a dependence. The mean rejection rate; i.e., the rate at which messages are turned away from full station buffers, also becomes negligible with buffer sizes greater than three. Inspection of the results for mean queue size reveal how queue size depends on the arrival rate at each station at low levels of traffic, but depends on the system load at high traffic levels when the access control protocol's effectiveness in servicing users affects the ability to empty station buffers. The relationship between mean queue size and mean waiting time given by Little's result is borne out of the findings of these simulations.

The simulation, by providing a means of developing and gauging the access control technique of decentralized adaptive probing, has highlighted the differences in performance between a pure contention protocol like Ethernet's and a technique incorporating aspects of selection protocols. At high traffic levels, the CSMA-CD access control technique requires that stations contend with each other for access to the channel, incurring delays in the process. Additionally, Ethernet's specifications allow only sixteen transmission attempts — after sixteen attempts, a failure is reported to higher protocol layers. In decentralized adaptive probing, however, sixteen attempts are never required to achieve a successful transmission — success is guaranteed after fewer attempts. Furthermore, no penalty in performance is incurred for this benefit since the decentralized adaptive probing protocol operates as a CSMA-CD system at low traffic levels.

This study, then, has revealed aspects of the performance of local area network access control protocols and has presented a variety of techniques suitable for the study of such protocols. Possible future work stemming from this effort includes use of the simulation programs in parametric studies, application of the analysis of CSMA-CD with a finite number of buffered users to obtain numerical results, and perhaps the implementation of a network supported by the decentralized adaptive probing access control technique.

5. References

- [1] F.A. Tobagi, Mario Gerla, Richard W. Peebles, and Eric G. Manning, "Modeling and Measurement Techniques in Packet Communication Networks," Proc. of the IEEE, Vol. 66, No. 11, November 1978, pp. 1423-1447.
- [2] Leonard Kleinrock, Queueing Systems, Vol. I: Theory, John Wiley & Sons, New York, 1975.
- [3] John F. Shoch and Jon A. Hupp, "Measured Performance of an Ethernet Local Network," Comm. of the ACM, Vol. 23, No. 12, December 1980, pp. 711-721.
- [4] International Organization for Standardization Draft International Standard ISO/DIS 7498, "Information Processing Systems — Open Systems Interconnection — Basic Reference Model," 22 April 1982.
- [5] Hubert Zimmerman, "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection," IEEE Trans. Communications, Vol. COM-28, No. 4, April 1980, pp. 425-432.
- [6] F.A. Tobagi and V.B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Proc. Local Area Communications Network Symposium, Boston, MA, May 1979, pp. 217-245; also in Computer Networks, Vol. 4, No. 5, November 1980, pp. 245-259.
- [7] S.S. Lam, "A Carrier Sense Multiple Access Protocol for Local Networks," Computer Networks, Vol. 4, No. 1, January 1980, pp. 21-32.
- [8] Leonard Kleinrock and F.A. Tobagi, "Packet Switching in Radio Channels: Part 1 — Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," IEEE Trans. Communications, Vol. COM-23, No. 12, December 1975, pp. 1400-1416.

- [9] IEEE 802 Local Network Standard, Draft 1, IEEE Computer Society, 4 July 1981.
- [10] IEEE Project 802, Draft IEEE Standard 802.3: CSMA/CD Access Method and Physical Layer Specification, Revision D, December 1982.
- [11] IEEE Project 802, Draft IEEE Standard 802.4: Token-Passing Bus Access Method and Physical Layer Specifications, Draft D, December 1982.
- [12] Ware Myers, "Toward a Local Network Standard," IEEE Micro, Vol. 2, No. 3, August 1982, pp. 28-45.
- [13] "Thirteen Vendors Endorse IEEE Local-Net Standard," Computerworld, Vol. XVI, No. 50, 13 December 1982.
- [14] Maris Graube, "Local Area Nets: A Pair of Standards," IEEE Spectrum, June 1982, pp. 60-64.
- [15] The Ethernet: A Local Area Network; Data Link Layer and Physical Layer Specifications, Version 1.0, Technical Manual, DEC-Intel-Xerox, 30 September 1980.
- [16] Jeremiah F. Hayes, "An Adaptive Technique for Local Distribution," IEEE Trans. Communications, Vol. COM-26, No. 8, August 1978, pp. 1178-1186.
- [17] John I. Capetanakis, "Generalized TDMA: The Multi-Accessing Tree Protocol," IEEE Trans. Communications, Vol. COM-27, No. 10, October 1979, pp. 1476-1484.
- [18] John I. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," IEEE Trans. Information Theory, Vol. IT-25, No. 5, September 1975, pp. 505-515.

- [19] Steven Schoemaker, "On Simulation," in Computer Networks and Simulation, edited by S. Schoemaker, North-Holland Publishing Co., 1978, pp. 69-84.
- [20] T.N. Saadawi and A. Ephremides, "Analysis, Stability, and Optimization of Slotted ALOHA with a Finite Number of Buffered Users," IEEE Trans. Automatic Control, Vol. AC-26, No. 3, June 1981, pp. 680-689.
- [21] William Feller, An Introduction to Probability Theory and Its Applications, Volume I, Third Edition, John Wiley & Sons, New York, 1968.
- [22] S. Karlin and H.M. Taylor, A First Course in Stochastic Processes, Second Edition, Academic Press, New York 1975.
- [23] J.D.C. Little, "A Proof of the Queueing Formula $L=\lambda W$," Operations Research 9, 1961, pp. 383-387.
- [24] W.S. Jewell, "A Simpler Proof of $L=\lambda W$," Operations Research 15, 1967, pp. 1109-1116.
- [25] S. Stidham, Jr., "A Last Word on $L=\lambda W$," Operations Research 22, 1974, pp. 417-421.
- [26] Robert M. Metcalfe and David R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Comm. of the ACM, Vol. 19, No. 2, July 1976, pp. 395-404.
- [27] Madhav Marathe and Bill Hawe, "Performance of a Simulated Ethernet Programming Environment," NETwords: the NETWORKS SIG newsletter, Vol. 8, No. 2, February 1983, pp. 19-29.
- [28] E. Fuchs and P.E. Jackson, "Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models," ACM Symposium on Problems in the Optimization of Data Communications Systems, Callaway Gardens, Pine Mountain, Georgia, 13-16 October 1969, pp. 201-226; also in CACM, Vol. 13, No. 16, 1970, pp. 752-757.

- [29] "ANSI Rejects OSI Model as Standard," Infocomm Standards, Vol. 1, No. 8, December 1982, pp. 121-122.
- [30] H.L. Gray and W.R. Schucany, The Generalized Jackknife Statistic, Marcel Dekker, Inc., New York, 1972.
- [31] Rupert G. Miller, "The Jackknife — A Review," Biometrika 61, 1974, pp. 1-15.
- [32] M.H. Quenouille, "Notes on Bias in Estimation," Biometrika 43, 1956, pp. 353-360.
- [33] J. Durbin, "A Note on the Application of Quenouille's Method of Bias Reduction to the Estimation of Ratios," Biometrika 46, 1959, pp. 477-480.
- [34] John W. Tukey, "Bias and Confidence in Not-Quite Large Samples (Abstract)," Ann. Math. Statist., Vol. 19, 1958, p. 614.
- [35] J.N.K. Rao, "A Note on the Estimation of Ratios by Quenouille's Method," Biometrika 52, 1965, pp. 647-649.
- [36] J.N.K. Rao and J. Webster, "On Two Methods of Bias Reduction in the Estimation of Ratios," Biometrika 53, 1966, pp. 571-577.
- [37] Michael A. Crane and Donald L. Iglehart, "Simulating Stable Stochastic Systems, I: General Multiserver Queues," J. Assoc. Comput. Mach., Vol. 21, No. 1, January 1974, pp. 103-113.
- [38] Michael A. Crane and Donald L. Iglehart, "Simulating Stable Stochastic Systems, II: Markov Chains," J. Assoc. Comput. Mach., Vol. 21, No. 1, January 1974, pp. 114-123.
- [39] Michael A. Crane and Donald L. Iglehart, "Simulating Stable Stochastic Systems, III: Regenerative Processes and Discrete-Event Simulations," Operations Research, Vol. 23, No. 1, January-February 1975, pp. 33-45.

- [40] Michael A. Crane and Donald L. Iglehart, "Simulating Stable Stochastic Systems, IV: Approximation Techniques," Management Science, Vol. 21, No. 11, July 1975, pp. 1215-1224.
- [41] Donald L. Iglehart, "Simulating Stable Stochastic Systems, V: Comparison of Ratio Estimators," Naval Res. Logist. Quart., Vol. 22, 1978, pp. 553-565.
- [42] Charles H. Sauer, "Confidence Intervals for Queueing Simulations of Computer Systems," Performance Evaluation Review, Vol. 8, Nos. 1 and 2, Spring-Summer 1979, pp. 45-55.
- [43] S.S. Lavenberg and C.H. Sauer, "Sequential Stopping Rules for the Regenerative Method of Simulation," IBM J. Res. Develop. 21, November 1977, pp. 545-558.
- [44] Imrich Chlemtac and Raj Jain, "A Methodology for Building a Simulation Model for Efficient Design and Performance Analysis of Local Area Networks," Simulation, Vol. 42, No. 2, February 1984, pp. 57-66.
- [45] Jeremiah F. Hayes, personal communications, McGill University, Montreal, 1982.
- [46] Edward C. Luczak, "Global Bus Computer Communication Techniques," Proc. Computer Networking Symposium, Gaithersburg, MD, December 1978, pp. 58-71.
- [47] "Special Issue on Local Area Networks," IEEE Journal on Selected Areas in Communications, Vol. SAC-1, No. 5, edited by K. Kuemmerle, B.W. Stuck, and F.A. Tobagi, November 1983.

Appendix

This appendix provides listings of the programs used in the simulations of adaptive probing and Ethernet. Additionally, a sample page of output is shown. The programs were written for private use and are not intended to be distributed; they are included here simply as documentation. Aspects of the code are not as efficient as possible, but reflect the design and debugging process. Furthermore, the programs were not created to be exceptionally user-friendly. In particular, many of the parameters that are varied as part of the simulation study (e.g., the number of stations, etc.) may only be changed by editing and again compiling the program.

The first program, labelled here CSMA5.FOR, simulates Ethernet in accordance with the model discussed in Section 2.2.1. Points of the program that might be altered to produce desired results are underscored and/or annotated. The second program, APROBE.FOR, written using the same structure as CSMA5.FOR but necessarily more complex, simulates decentralized adaptive probing as detailed in Section 2.2.2. Annotations to the code indicate the manner in which the method of aggregate states was used to obtain a sufficient number of tours when the system was simulated at high load levels.

The third program, NTOPO.FOR, was originally written as an interactive program to allow manual entry of the network topology. The user specifies the cable number, location on the cable in units of 2.5 meter marks, and the type of connection; i.e., whether it is a station or a repeater. If a station is specified, the program requires its binary address; if a repeater is specified, the program requires the other end of the connection in terms of the cable number and location on that cable. All MAU cable lengths are arbitrarily generated by the program. The network topology that is thus defined is then checked to insure that there are no violations of the rules cited in Section 2.2.3; e.g., that there are no loops in the system, that no two stations are assigned to the same location or given the same address, that the number of connections per cable is not excessive, etc.

To easily enter the number of stations desired for the simulation study, NTOPO.FOR, rather than operating interactively, has been modified to receive answers to its queries from a separate data file. The program provides as output a listing of the network's stations and repeaters and a data

file providing the number of stations, the address of each station, and a matrix of the propagation delays between each station. This data file is all that is used by the simulation programs since, from the point of view of the access control protocols, the only effect of the network configuration is the propagation delays between stations.

Sample Page of Simulation Output

ETHERNET (CSMA-CD)

System parameters:

Number of stations = 100
Transmission rate = 10 Mbits per second
Buffer size at all stations = 20 frame(s)

System statistics (90% confidence intervals):

System load = 0.2533 (0.2395, 0.2671)
Poisson arrival rate = 0.614E+00 frames/second
Mean xmitted frame size (w/o preamble) = 2076.69 (2716.33, 3037.04)
Data throughput = 0.2358 (0.2200, 0.2516)
Network throughput = 0.2545 (0.2384, 0.2705)
Mean frame delay (incl. preamble) = 4.963E-04 (4.643E-04, 5.284E-04)
Mean waiting time = 2.024E-04 (1.756E-04, 2.291E-04) secs
Mean queue size = 1.750E-03 (1.493E-03, 2.007E-03) frames
Mean number of attempts required = 1.323E+00 (1.268E+00, 1.377E+00)
Mean departure rate = 0.652E+00 (0.356E+00, 0.940E+00) frames/second
Mean rejection rate = 0.000E+00 (0.000E+00, 0.000E+00) frames/second
Mean abandonment rate = 0.000E+00 (0.000E+00, 0.000E+00) frames/second

Simulation statistics:

Total time simulated = 280 seconds
Mean duration of each tour = 1.299E-03 seconds
Number of tours = 2000
Total CPU seconds used = 763.94 seconds

Program CSMA5.FOR

CSMA5.FOR

A simulation program of Ethernet— a Carrier Sense Multiple Access
local network with Collision Detection (CSMA-CD).

NSTAT terminals are connected to the network, each with buffer
space NLIM and individual Poisson arrival rate ALAN(NSTAT).

DIMENSION nspace(400), natt(400), alan(400), iqsize(400),
1 naban(400), irej(400), wtotal(400), tidle(400),
1 packet(400), delay(400), sked(800), tk(400, 51),
1 wtime(400, 50), pdel(400), tdel(400, 400),
1 ndep(400), aqsize(400),
1 nat(400, 40), suatt(400), rho1(9),
1 theta_time(2000), theta_rej(2000),
1 theta_aban(2000), theta_timeon(2000),
1 theta_timeon1(2000), theta_size(2000),
1 theta_pmean(2000), theta_deprate(2000),
1 theta_tdep(2000), theta_thru1(2000),
1 theta_thru(2000), theta_rejrate(2000),
1 theta_abanrate(2000), theta_taqsize(2000),
1 theta_tdelay(2000), theta_twait(2000),
1 theta_tatt(2000)

EQUIVALENCE (theta_thru, theta_timeon),
1 (theta_rejrate, theta_rej),
1 (theta_abanrate, theta_aban),
1 (theta_taqsize, theta_size)

DATA (rho1(i), i=1, 9)/.001, .005, .01, .05, .1, .25, .5, .75, .95/

COMMON jseed
1 /blk1/ mean, maxfr, minfr, rate, frgap, limbko, islot
1 /blk15/ icarr, carroff
1 /blk2/ sked, tidle, packet, wtime
1 /blk25/ iqsize, natt
1 /blk3/ ndep, wtotal, delay, naban
1 /blk35/ tdel
1 /blk4/ nchek(400)
1 /blk45/ defer
1 /blk5/ irej
1 /blk55/ nat
1 /blk6/ pmean

INTEGER*4 cpu_seconds, start_cpu_clock, elapsed_cpu
EXTERNAL cpu_seconds

LOGICAL sstate, defer(400), end_of_tour, stop

CHARACTER line*80, reply*1, address(400)*16

*Notiv of station & station
propagation delays*

C Access topology file

OPEN (UNIT=2, FILE='user [alan]dat100.dat', STATUS='old',
1 FORM='unformatted', ERR=999)

READ(2, ERR=101) nstat, ((tdel(i, j), j=1, nstat), i=1, nstat),
1 (address(j), j=1, nstat)

101 CLOSE(UNIT=2)

C The following parameters are vestiges of an earlier incarnation
C of the simulation and are here rendered ineffectual:

ssstate= TRUE.

C =state of system; set to TRUE in this
C regenerative simulation study

DO j=1, nstat
nchek(j)=0
END DO

C Set simulation parameters:

C nlim=1 =limit to all stations' frame buffers (buffer capacity)

C System initialization:

C islot=512 =number of bits in a slot

C rate=10. e6 =transmission rate in bits/second

C limatt=16 =attempt limit

C limbko=10 =back-off limit

C jsize=32 =jam size in bits

C frgap=9.6e-6 =interframe gap size in seconds

minfr=islot

```

C                                     =minimum frame size in bits
maxfr=12144
C                                     =maximum frame size in bits
icdet=9
C                                     =number of bits required for collision detection
icarr=2
C                                     =number of bits required for a station to sense
C                                     a signal following the onset of carrier
C                                     presence information.
carroff=1.5
C                                     =number of bits required for a station to disable
C                                     the carrier sense signal following the
C                                     loss of carrier presence information
tjam=(icdet+jsize)/rate
C                                     =total duration of collision detection and jamming
jseed=117
C                                     =initial value for seed integer of random variate
C                                     generator
C
C Supreme loop control.
prate=0.0
DO iruns=1, 3
C                                     May be adjusted to provide runs of specific
C                                     load levels
prate = rho1(iruns)/(nstat*(2838.4 + 64.)/rate))
ktours = 10
C                                     must equal mean # of b's/frame
C                                     =number of tours elapsed before statistics
C                                     are again evaluated in stopping rule
C                                     May be adjusted
C                                     for high load runs
C Initialize parameters for start-up of first tour:
nlim1=nlim+1
nstat2=2*nstat
DO i=1, nstat
alam(i)=prate
C                                     =Poisson arrival rate of frames to i-th queue
tidle(i)=-frgap
C                                     =time from last event upon which station i hears
C                                     the channel idle
packet(i)=pleng*jseed/mean*maxfr/minfr/rate
C                                     =length of station j's current packet yet
C                                     to be transmitted (in seconds)
IF (iqsize(i) .GT. 0) THEN
sked(i)=ran(jseed)*.00001
ELSE

```

sked(1)=1.7e38

END IF

=time of station i's next transmission attempt

IN=INT(TOTVL

sked: IN=-1 /alam(1) * alog(ran(jseed))

=time of next arrival to station i

END DO

ntours=0

=number of times system returns to idle state

ltour=0

=index used to determine whether the number of
tours has increased by ktours

time=0.0

=total simulation time

Initialize accumulators for statistics-taking:

sumtime=0.0

sumrej=0.0

sumaban=0.0

sumtimeon=0

sumtimeon1=0.0

sumsize=0.0

sumpmean=0.0

sumtdep=0.0

sumdelay=0.0

sumwait=0.0

sumatt=0.0

stop = FALSE.

=flag indicating whether simulation run is over

start_cpu_clock = cpu_seconds(0)

=initial value of clock of elapsed cpu time

Begin new tour:

DO WHILE (.NOT. stop)

Initialize tour statistics:

tourtime=0.0

=time spent in current tour

timeon=0.0

=time channel is active during steady-state period
(minus all overhead)

timeon1=0.0

=time channel is active during steady-state period

mean=0.0

=accumulator for calculating mean size of packets
successfully transmitted

nlim1=nlim+1
nstat2=2*nstat

DO i=1,nstat

defer(i)=.false.

=no stations are presently deferring transmission

natt(i)=0

=number of transmission attempts for the frame at
the head of the i-th station's buffer

ndep(i)=0

=number of successful departures from station i,
compiled during steady-state period

iqsize(i)=0

=number of frames in the i-th station's buffer

naban(i)=0

=number of frame transmissions abandoned at
station i due to excessive attempts;
compiled during steady-state period

irej(i)=0

=number of frames rejected due to a full^o buffer
at station i; compiled during steady-state
period

wtotat(i)=0.0

=total time frames spend in station i's frame buffer
before being transmitted; compiled during
steady-state period for frames that are
successfully transmitted

delay(i)=0.0

=total time frames spend in station i's frame buffer
before being transmitted, plus frame duratio
compiled during steady-state period for
successfully transmitted frames

DO j=1,nlim

tk(i,j)=0.0

=amount of time station i has length (j-1);
compiled during steady-state period

wtime(i,j)=0.0

=elapsed waiting time of the j-th element of
station i's buffer; compiled during
steady-state period

END DO

tk(i,nlim1)=0.0

```
DO j=1,nstat
  IF i=j THEN
```

```
    number of station i's successful
    transmissions requiring j attempts; compiled
    only for packets having arrived during steady-state
    period
```

```
  END IF
```

```
END DO
```

```
Simulation of one tour
```

```
end_of_tour = FALSE
```

```
DO WHILE ( NOT end_of_tour)
```

```
Find next event (arrival or transmission attempt)
```

```
del=sched(1)
```

```
j=1
```

```
DO i=2,nstat2
```

```
  IF (sched(i) LT del) THEN
```

```
    del=sched(i)
```

```
    =next event time
```

```
  j=i
```

```
    =index of that next event; j > nstat implies
    a frame arrival to station (j - nstat);
    otherwise, station j is attempting a
    transmission
```

```
  END IF
```

```
END DO
```

```
Update simulation and tour clock
```

```
time = time + del
```

```
tourtime = tourtime + del
```

```
Update schedule times:
```

```
DO i=1,nstat2
```

```
  sched(i)=sched(i)-del
```

```
END DO
```

```
Update time until each station hears the channel idle:
```

```
DO i=1,nstat
```

```
  tidle(i)=tidle(i)-del
```

```
END DO
```

```
Compile steady-state statistics:
```

=, TRUE, always; see the section on Representative simulation

IF (sstate) THEN

Increment steady-state clock

tsted=tsted+del

Increment amount of time station 1 has length iqsize(1)

DO i=1,nstat

ij=iqsize(i)+1

tk(i,ij)=tk(i,ij)+del

Increment waiting time of all elements of station 1's buffer

DO k=1,iqsize(1)

wtime(1,k)=wtime(1,k)+del

END DO

END DO

END IF

IF (j GT nstat) THEN

CALL arrive(j,nstat,nlim,sstate,alam)

ELSE

CALL xmit(j,nstat,sstate,timeon,timeon1,nlim,tjam,limatt)

END IF

Test for end of tour

end_of_tour = TRUE

DO i=1,nstat

IF (iqsize(i) NE 0) end_of_tour = FALSE

END DO

IF (end_of_tour) THEN

Update number of tours

ntours = ntours + 1

Sum over all stations:

taqsize=0.

trej=0.

tnaban=0.

tdep=0

tdelay=0.

twait=0.

tatt=0.

DO j=1,nstat

aqsize(j)=0 @

DO k=1,nlim1

k1=k-1

aqsize(j) = aqsize(j) + k1*tk(j,k)

=weighted sum of time elapsed at

```

C                                     queue size k1 at station J
END DO
taqsize = taqsize + aqsize(J)

avatt(J) = 0
DO k=1,limatt
  avatt(J) = avatt(J) + k*nat(J,k)
                                     =weighted sum of attempts required
                                     at station J
END DO

trej = trej + irej(J)
                                     =number of transmission requests
                                     turned away from all stations

tnaban = tnaban + naban(J)
                                     =number of transmission attempts
                                     abandoned by all stations

tdep = tdep + ndepw(J)
                                     =total number of successful transmissions

tdelay = tdelay + delay(J)
                                     =total frame delay

twait = twait + wtotal(J)
                                     =total waiting time

tatt = tatt + avatt(J)
                                     =total number of attempts

END DO

Average over all stations

taqsize = taqsize/nstat
                                     =average queue size (prior to averaging over time)

tnaban = tnaban/nstat
                                     =number of transmission requests that fail per station

trej = trej/nstat
                                     =number of transmission requests turned
                                     away from the (full) station buffer per station

Update pseudovalue elements for jackknife estimators:

theta_time(ntours) = sumtime
theta_rej(ntours) = sumrej
theta_aban(ntours) = tnabantnaban
theta_timeonl(ntours) = sumtimeonl
theta_timeonl1(ntours) = sumtimeonl1
theta_size(ntours) = sumsize
theta_pmean(ntours) = sumpmean
theta_tdep(ntours) = sumtdep
theta_tdelay(ntours) = sumdelay

```

```

theta_twait(ntours) = sumwait
theta_tatt(ntours) = sumatt

```

```

sumtime = sumtime + tourtime
sumrej = sumrej + trej
sumaban = sumaban + tnanban
sumtimeon = sumtimeon + timeon
sumtimeon1 = sumtimeon1 + timeon1
sumsize = sumsize + taqsize
sumpmean = sumpmean + pmean
sumtdep = sumtdep + tdep
sumdelay = sumdelay + tdelay
sumwait = sumwait + twait
sumatt = sumatt + tatt

```

```

IF (ntours GT 1) THEN

```

```

  DO i=1,(ntours-1)

```

```

    theta_time(i) = theta_time(i) + tourtime
    theta_rej(i) = theta_rej(i) + trej
    theta_aban(i) = theta_aban(i) + tnanban
    theta_timeon(i) = theta_timeon(i) + timeon
    theta_timeon1(i) = theta_timeon1(i) + timeon1
    theta_size(i) = theta_size(i) + taqsize
    theta_pmean(i) = theta_pmean(i) + pmean
    theta_tdep(i) = theta_tdep(i) + tdep
    theta_tdelay(i) = theta_tdelay(i) + tdelay
    theta_twait(i) = theta_twait(i) + twait
    theta_tatt(i) = theta_tatt(i) + tatt

```

```

  END DO

```

```

END IF

```

```

IF (ntours GE 40 AND ntours/ktours NE 1tour) THEN

```

```

  Reset index of last cycle of ktours

```

```

  1tour = ntours/ktours

```

```

  Evaluate network throughput via jackknife estimator

```

```

  thrupt1 = 0 0

```

```

  DO i=1,ntours

```

```

    Evaluate pseudovalues

```

```

    theta_thru1(i) = ntours*sumtimeon1/sumtime -
    (ntours-1)*theta_timeon1(i)/theta_time(i)

```

```

    Sum pseudovalues

```

```

    thrupt1 = thrupt1 + theta_thru1(i)

```

```

  END DO

```

```

  thrupt1 = thrupt1/ntours

```

```

  Evaluate variance of jackknifed thrupt1.

```

```

  s2thru1 = 0 0

```

```

  DO i=1,ntours

```



```

1      theta_abanrate(i) = ntours*sumaban/sumtime -
      (ntours-1)*theta_aban(i)/theta_time(i)
1      theta_taqsize(i) = ntours*sumsize/sumtime -
      (ntours-1)*theta_size(i)/theta_time(i)
1      theta_pmean(i) = ntours*sumpmean/sumtdep -
      (ntours-1)*theta_pmean(i)/theta_tdep(i)
1      theta_deprate(i) = ntours*sumtdep/sumtime -
      (ntours-1)*theta_tdep(i)/theta_time(i)
1      theta_tdelay(i) = ntours*sumdelay/sumtdep -
      (ntours-1)*theta_tdelay(i)/theta_tdep(i)
1      theta_twait(i) = ntours*sumwait/sumtdep -
      (ntours-1)*theta_twait(i)/theta_tdep(i)
1      theta_tatt(i) = ntours*sumatt/sumtdep -
      (ntours-1)*theta_tatt(i)/theta_tdep(i)

```

Sum pseudovalues

```

thruput = thruput + theta_thru(i)
rejrate = rejrate + theta_rejrate(i)
abanrate = abanrate + theta_abanrate(i)
taqsize = taqsize + theta_taqsize(i)
pmean = pmean + theta_pmean(i)
deprate = deprate + theta_deprate(i)
frdelay = frdelay + theta_tdelay(i)
frwait = frwait + theta_twait(i)
fratt = fratt + theta_tatt(i)

```

END DO

```

thruput = thruput/ntours
      =ratio of time channel is spent successfully
      transmitting data less overhead to
      total time

```

```

rejrate = rejrate/ntours
      =number of arrivals to a station's buffer
      not accepted per second

```

```

abanrate = abanrate/ntours
      =number of transmission attempts abandoned
      per second

```

```

taqsize = taqsize/ntours
      =mean queue size

```

```

pmean = pmean/ntours
      =mean frame size (less preamble)

```

```

deprate = deprate/ntours
      =mean departure rate of successful frames from
      all stations

```

```

frdelay = frdelay/ntours
      =mean frame delay of successful frames

```

```

frwait = frwait/ntours
      =mean frame waiting time

```

```

fratt = fratt/htours
      =mean number of attempts required for successful
      transmission

```

```

Evaluate variance of jackknifed estimates

```

```

s2thru=0.0
s2rejrate=0.0
s2abanrate=0.0
s2taqsize=0.0
s2pmean=0.0
s2deprate=0.0
s2frdelay=0.0
s2frwait=0.0
s2fratt=0.0

```

```

DO 1=1, ntours

```

```

      s2thru = s2thru + (theta_thru(1) - thrupt)**2
      s2rejrate = s2rejrate +
1          (theta_rejrate(1) - rejrate)**2
      s2abanrate = s2abanrate +
1          (theta_abanrate(1) - abanrate)**2
      s2taqsize = s2taqsize +
1          (theta_taqsize(1) - taqsize)**2
      s2pmean = s2pmean +
1          (theta_pmean(1) - pmean)**2
      s2deprate = s2deprate +
1          (theta_deprate(1) - deprate)**2
      s2frdelay = s2frdelay +
1          (theta_tdelay(1) - frdelay)**2
      s2frwait = s2frwait +
1          (theta_twait(1) - frwait)**2
      s2fratt = s2fratt +
1          (theta_tatt(1) - fratt)**2

```

```

END DO

```

```

s2thru = s2thru/(ntours-1)
s2rejrate = s2rejrate/(ntours-1)
s2abanrate = s2abanrate/(ntours-1)
s2taqsize = s2taqsize/(ntours-1)
s2pmean = s2pmean/(ntours-1)
s2deprate = s2deprate/(ntours-1)
s2frdelay = s2frdelay/(ntours-1)
s2frwait = s2frwait/(ntours-1)
s2fratt = s2fratt/(ntours-1)

```

```

delta = 1.645*sqrt(s2pmean/ntours)
upper_pmean = pmean + delta
tlower_pmean = pmean - delta

```

```

rho = prate*(pmean+64)*nstat/rate
upper_rho = prate*(upper_pmean+64)*nstat/rate
tlower_rho = prate*(tlower_pmean+64)*nstat/rate

```

```

WRITE(6, '(///, " System statistics (90% confidence",
1      " intervals): ', //, 3x,

```

```

1      ''Poisson arrival rate = '', 1pe10. 3,
1      '' frames/second'', //, 5x,
1      ''Mean transmitted frame size'',
1      ''xw/o preamble) = '', 0pf8. 2, 2x, ''( '', 0pf8. 2,
1      ''( '', 0pf8. 2, '' ) bits'', //, 5x,
1      ''System load = '', 0pf6. 4, 2x,
1      ''( '', 0pf6. 4, '' , '', 0pf6. 4, '' )''''''
1      prate, pmean, tlower_pmean, upper_pmean,
1      rho, tlower_rho, upper_rho

delta = 1.645*sqrt(s2thru/ntours)
upper = thrupt + delta
tlower = thrupt - delta
upper1 = thrupt1 + del
tlower1 = thrupt1 - del

WRITE(6, '(5x, ''Data throughput = '', 0pf6. 4, 2x,
1      ''( '', 0pf6. 4, '' , '', 0pf6. 4, '' )''''', //, 5x,
1      ''Network throughput = '', 0pf6. 4, 2x,
1      ''( '', 0pf6. 4, '' , '', 0pf6. 4, '' )''''')
1      thrupt, tlower, upper, thrupt1, tlower1,
1      upper1

delta = 1.645*sqrt(s2frdelay/ntours)
upper = frdelay + delta
tlower = frdelay - delta

WRITE(6, '(5x,
1      ''Mean frame delay (incl. preamble) = '',
1      1pe10. 3, 2x, ''( '', 1pe10. 3, '' , '', 1pe10. 3,
1      '' ) secs''''')
1      frdelay, tlower, upper

delta = 1.645*sqrt(s2frwait/ntours)
upper = frwait + delta
tlower = frwait - delta

WRITE(6, '(5x,
1      ''Mean waiting time = '', 1pe10. 3, 2x, ''( '',
1      1pe10. 3, '' , '', 1pe10. 3, '' ) secs''''')
1      frwait, tlower, upper

delta = 1.645*sqrt(s2taqsize/ntours)
upper = taqsize + delta
tlower = taqsize - delta

WRITE(6, '(5x, ''Mean queue size = '', 1pe10. 3,
1      2x, ''( '', 1pe10. 3, '' , '', 1pe10. 3, '' ) frames''''')
1      taqsize, tlower, upper

delta = 1.645*sqrt(s2fratt/ntours)
upper = fratt + delta

```

1
1
1

1
1
1

1
1
1

1
1
1

1
1
1
1
1
1
1

```

END DO
C      test on end_of_tour

END DO
C      test on stop (end of simulation on rho1)

END DO
C      i=1,9

```

```

WRITE(6, '(1H1)')

```

```

STOP

```

```

999  WRITE(6,*) 'Error in OPENing data file'
STOP

```

```

END

```

```

C      Function PLENG provides a bimodal distribution for
C      the frame lengths

```

```

FUNCTION pleng(jseed, mean, maxfr, minfr)

```

```

IF (ran(jseed) .LT. .2) THEN

```

```

    pleng=maxfr+64.

```

```

ELSE

```

```

    pleng=minfr+64.

```

```

END IF

```

```

RETURN

```

```

END

```

*Bimodal
Distribution*

```

C      FUNCTION pleng(jseed, mean, maxfr, minfr)

```

```

C      pleng=maxfr/2 + 64.

```

```

C      RETURN

```

```

C      END

```

Fixed length frame

```

C      FUNCTION pleng(jseed, mean, maxfr, minfr)

```

```

C10  pleng=-1. *mean*log(ran(jseed))

```

```

C      packet length is derived first as an
C      exponentially distributed
C      variate with the given mean.

```

*Exponential
distribution*

```

C      IF (pleng .GT. maxfr) THEN

```

```

C      pleng=maxfr

```

```

C ELSE IF (pleng .LT. minfr) THEN
C   pleng=minfr
C END IF

```

but the length must be within the limits established by minfr and maxfr.

```

C   pleng=int(pleng/8.)*8. + 64

```

and a packet must equal an integer number of octets plus the 64 bit preamble

```

C RETURN
C END

```

Subroutine IDLE computes the time all stations sense the channel idle with respect to the n transmitting stations listed in index(m). Their transmissions end at end(index(m)).

```

SUBROUTINE idle(nstat, index, end, m)
DIMENSION index(m), end(400)
COMMON jseed
1 /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot
1 /blk15/ icarr, carroff
1 /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)
1 /blk35/ tdel(400, 400)

DO k=1, nstat
  tidle(k)=0.0
  DO kn=1, m
    temp=end(index(kn))+tdel(index(kn), k)
    IF (index(kn) .NE. k) temp=temp+carroff/rate
    IF (temp .GT. tidle(k)) tidle(k)=temp
  END DO
END DO

RETURN
END

```

Station j is attempting a transmission; note that when subroutine XMIT is called, all elapsed times have been updated to the time of j's attempt

```

SUBROUTINE xmit(j, nstat, sstate, timeon, timeon1, nlim, tjam, limatt)
DIMENSION index1(400), index2(400), index3(400), e_jam(400)
COMMON jseed
1 /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot
1 /blk15/ icarr, carroff
1 /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)
1 /blk25/ lqsize(400), natt(400)
1 /blk3/ ndep(400), wtotal(400), delay(400), naban(400)
1 /blk35/ tdel(400, 400)
1 /blk4/ nchek(400)

```

```

1      /blk45/ defer
1      /blk55/ nat(400)(40)
1      /blk6/ pmean
LOGICAL sstate,defer(400),cont

```

C Reset relevant values:

```

DO k=1,nstat
  ejam(k)=0.0
  tidle(k)=0.0
END DO

```

C Search for transmission attempts scheduled to occur before
C J's transmission is sensed:

```

n=0
m=0
DO k=1,nstat
  cont=.true.

```

C =contention status; .false. implies a station
C is either deferring transmission or
C is scheduled for transmission much later

```

DO l=1,nstat
  IF (sked(k) .GT. (sked(l)+tdel(k,l)+(icarr/rate)))
1                                                           cont=.false.
END DO

```

```

IF (cont .EQ. .true. .OR. k .EQ. j) THEN

```

C m=m+1
C =number of stations in contention, including station j

```

  defer(k)=.false.

```

C defer(k) of contending station and
C station j reset

```

  index1(m)=k

```

C =station number of a contending station

```

END IF
END DO

```

```

IF (m .EQ. 1) THEN

```

C no collision, unless a deferring
C station transmits

C Test for possible collision due to deferring stations:

C Reset the time at which j's jam ends (to be used in the
C event of deferring station collision):

```

  ejam(j)=1.7e38

```

C Test for stations who have a transmission scheduled during the
C ongoing transmission:

```
DO k=1,nstat
  IF (sked(k) .LT. (packet(j)+tdel(k,j)+(carroff/rate))
1   .AND. (defer(k))) THEN
    temp=sked(k)+tdel(k,j)+tjam
    IF (temp .LT. ejam(j)) THEN
      ejam(j)=temp
      kd=k
    END IF
  END IF
END DO
```

IF (ejam(j) .LT. 1.7e38) THEN

C a deferring station has caused
C a collision

```
mtemp=m+1
DO WHILE (mtemp .NE. m)
```

C until no more deferring stations are in contention

```
mtemp=m
```

C Compute time until all stations sense the channel idle:

```
CALL idle(nstat,index1,ejam,m)
```

C Find the deferring stations that are thus in conflict:

```
DO k=1,nstat
  IF (sked(k) LT tidle(k) .AND. (defer(k))) THEN
```

C Reset status of transmitting station:

```
defer(k)=.false.
```

```
m=m+1
```

```
index1(m)=k
```

C =index of all transmitting stations in contention

```
ejam(k)=sked(k)+tjam
```

C =time at which the k-th station's jam ends

```
END IF
```

```
END DO
```

```
END DO
```

C Compute time all stations sense channel idle:

```
CALL idle(nstat,index1,ejam,m)
```

C Update number of attempts and reschedule transmissions
C for colliding stations:

```

DO k=1, m
  CALL att(index1(k), limatt, nlim, sstatq, ejan, nstat)
END DO

```

ELSE

successful transmission by station j

```
natt(j) = natt(j) + 1
```

success on (new) natt(j)-th attempt

```
IF (sstate) THEN
```

compile steady-state statistics

Increment time channel is successfully active:

```
timeon = timeon + (packet(j)-64./rate - 152./rate)
```

(subtracting off the time spent transmitting
the 64 bit preamble and an
assumed 19 octets of overhead)

```
timeon1 = timeon1 + packet(j)
```

Test whether successful transmission had arrived during
steady-state period:

```
IF (nchek(j) .NE. 0) THEN
```

nchek(j) = 0 @ j

update counter

```
nchek(j)=nchek(j)-1
```

ELSE

compile statistics

```
pmean = pmean + (packet(j)*rate - 64.)
```

```
ndep(j)=ndep(j)+1
```

```
wtotat(j)=wtotat(j)+utime(j,1)
```

```
delay(j)=delay(j)+utime(j,1)+packet(j)
```

```
nat(j,natt(j)) = nat(j,natt(j)) + 1
```

END IF

Advance the sequencing of waiting times in j's queue:

```
DO i=2, nlin
```

```
  ii=i-1
```

```
  utime(j, ii)=utime(j, i)
```

```
END DO
```

```
utime(j, nlin)=0.0
```

END IF

Update station j's queue size:

iqlsize(j)=iqlsize(j)-1

Reset number of transmission attempts:

natt(j)=0

Compute time all stations hear channel idle:

ejam(j)=packet(j)

CALL idle(nstat, index1, ejam, m)

IF (iqlsize(j) GE. 1) THEN

Schedule next transmission:

sked(j)=packet(j) + frgap

Generate packet length of next message to be transmitted:

packet(j)= pleng(jseed, mean, maxfr, minfr)/rate

update frame size for packet now
to be transmitted

ELSE

sked(j)=1.7e38

with no frames in station j's queue,
the next transmission is
scheduled to an improbably long
interval

END IF

END IF

Are any stations deferring transmission?

DO k=1, nstat

IF (sked(k) .LT. tidle(k) .AND. k .NE. j) THEN
defer(k)=.true.

i. e., station k's transmission
is deferred

sked(k)=tidle(k)+frgap

END IF

END DO

ELSE

collision!

Determine the time all stations in contention end their transmissions, with respect to their own locations

DO k=1, n

ejam(index1(k))=1.7e38

DO l=1, m

temp=sked(index1(l))+tdel(index1(l), index1(k))+tjam

IF (temp .LT. ejam(index1(k)) .AND. l .NE. k)

1

ejam(index1(k))=temp

=time from transmission
event to end of
jam for contending
station index1(k)

i.e., a contending station jams upon
discovering contention

END DO

END DO

Look for deferring stations who decide to transmit:

ntemp=n+1

n1=n

DO WHILE (ntemp .NE. m)

ntemp=m

Compute time until all stations sense the channel idle as
a result of jamming stations:

CALL idle(nstat, index1, ejam, m)

Find the deferring stations that are thus in conflict:

DO k=1, nstat

IF (sked(k) .LT. tidle(k) .AND. (defer(k))) THEN

Reset status of transmitting station:

defer(k)=.false.

m=m+1

index1(m)=k

=index of all transmitting stations in contention

ejam(k)=sked(k)+tjam

=time at which the k-th station's jam ends

END IF

END DO

END DO

C Compute time all stations hear channel idle

CALL idle(nstat, index1, ejam, m)

C Update number of attempts and reschedule transmissions
C for colliding stations

DO k=1, m

CALL att(index1(k), limatt, nlim, sstate, ejam, nstat)

END DO

C Find new deferring stations

DO k=1, nstat

IF (sked(k) LT tidle(k)) THEN

defer(k)= true

sked(k)=tidle(k)+frgap

END IF

END DO

END IF

RETURN

END

C Unsuccessful transmission attempt by station 1 is rescheduled
C or abandoned by subroutine ATT

SUBROUTINE att(i, limatt, nlim, sstate, ejam, nstat)

DIMENSION ejam(nstat)

COMMON jseed

1 /blk1/ mean, maxfr, minfr, nste, frgap, limbko, islot

1 /blk2/ sked(800), tidle(400), packet(400), wtime(400, 50)

1 /blk25/ lqsize(400), natt(400)

1 /blk3/ ndep(400), wtotal(400), delay(400), naban(400)

1 /blk4/ nchek(400)

LOGICAL sstate

C Increment number of transmission attempts

natt(i)=natt(i)+1

C Has the attempt limit been reached?

IF (natt(1) EQ 1) THEN

abandon the packet

natt(1)=0

IF (iqsize(1) GT 1) THEN

sked(1)=tidle(1)+frgap

packet(1)=pleng(jseed, mean, maxfr, minfr)/rate

ELSE

sked(1)=1.7e38

END IF

iqsize(1)=iqsize(1)-1

IF (sstate) THEN

IF (nchek(1) NE 0) THEN

nchek(1)=nchek(1)-1

ELSE

naban(1)=naban(1)+1

END IF

DO J=2, nlim

J1=J-1

utime(1, J1)=utime(1, J)

END DO

utime(1, nlim)=0

END IF

ELSE

prepare next attempt

sked(1)=ejam(1)

1 +ntime(natt(1), limbko, islot, rate)

END IF

RETURN

END

SUBROUTINE arrive(j, nstat, nlim, sstate, alam)

DIMENSION alam(nstat)

COMMON jseed

1 /blk1/ mean, maxfr, minfr, rate, frgap, limbko, islot

1 /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)

1 /blk25/ iqsize(400), natt(400)

1 /blk45/ defer(400)

1 /blk5/ irej(400)

LOGICAL sstate, defer

J1=j-nstat

sked(J)=-1./alam(J1) * alog(nan(jseed))

J=J1

IF (iqsize(J) EQ nlim) THEN

IF (sstate) THEN

irej(J)=irej(J)+1

END IF

ELSE

iqsize(J)=iqsize(J)+1

```

      wtime(j, lqsize(j))=0
END IF

```

```

IF (lqsize(j) EQ 0) THEN
  packet(j)=pleng(jseed, mean, maxfr, minfr)/rate
  IF (tidle(j) GE -frgap) THEN
    IF (tidle(j) GT 0.0) defer(j)= TRUE
    sked(j)=tidle(j)+frgap
  ELSE
    sked(j)=0.0
  END IF
ENDIF

RETURN
END

```

```

FUNCTION *time(n, limbko, islot, rate)
COMMON jseed

```

```

      n = total number of attempts

```

```

      nk=n
      IF (n GT limbko) nk=limbko
      rtime=(int(ran(jseed)*2 *nk)) * islot/rate
      RETURN
      END

```

```

SUBROUTINE print(filename)
CHARACTER filename *(*), line*80

```

```

      INTEGER * 4 remove_tail
      EXTERNAL remove_tail

```

```

      OPEN (UNIT=1, FILE=filename, READONLY, STATUS='old', ERR=1)

```

```

      DO WHILE ( TRUE )
        READ(1, '(a)', ERR=2) line
        WRITE(6, '(x,a)') line(:remove_tail(line))
      END DO

```

```

1      WRITE(6, *) 'Error in OPENing text file'
      STOP

```

```

2      CLOSE(UNIT=1)
      RETURN

```

```

      END

```

Program APROBE.FOR

APROBE.FOR-----

NSTAT terminals are connected to the network, each with buffer
space NLIM and individual Poisson arrival rate ALAM(NSTAT).

```
DIMENSION nspace(400), natt(400), alam(400), iqsize(400),
1          naban(400), irej(400), wtotal(400), tidle(400),
1          packet(400), fdelay(400), sked(800), tk(400, 31),
1          wtime(400, 30), pdel(400), tdel(400, 400),
1          ndep(400), aqsize(400), await(400), adelay(400),
1          nat(400, 40), avatt(400), plevel(17), rho1(9),
1          theta_time(2000), theta_rej(2000),
1          theta_aban(2000), theta_timeon(2000),
1          theta_timeon1(2000), theta_size(2000),
1          theta_level(2000), theta_pmean(2000),
1          theta_tdep(2000), theta_thru1(2000),
1          theta_thru(2000), theta_rejrate(2000),
1          theta_abanrate(2000), theta_taqsize(2000),
1          theta_alevel(2000), theta_deprate(2000),
1          theta_tdelay(2000), theta_twait(2000),
1          theta_tatt(2000)
```

```
EQUIVALENCE (theta_thru, theta_timeon),
1            (theta_rejrate, theta_rej),
1            (theta_abanrate, theta_aban),
1            (theta_taqsize, theta_size),
1            (theta_alevel, theta_level)
```

DATA (rho1(i), i=1, 9)/. 001, . 005, . 01, . 05, . 1, . 25, . 5, . 75, . 95/

```
COMMON jseed
1      /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot
1      /blk15/ icarr, carroff
1      /blk2/ sked, tidle, packet, wtime
1      /blk25/ iqsize, natt
1      /blk3/ ndep, wtotal, fdelay, naban
1      /blk35/ tdel
1      /blk4/ nchek(400)
1      /blk45/ defer
1      /blk5/ irej
1      /blk55/ nat
1      /blk6/ delay(16)
```

```

1      /blk65/ crash
1      /blk7/  prefix(400), address(400)
1      /blk75/ iflag(400)
1      /blk8/  level(400), tcycle(400)
1      /blk85/ index, movavg, time, elam, atime, xmission(65)
1      /blk9/  pmean

```

```

INTEGER*4 cpu_seconds, start_cpu_clock, elapsed_cpu
EXTERNAL  cpu_seconds

```

```

LOGICAL sstate, defer(400), iflag, iflag1,
1      crash, end_of_tour, stop, warn_up

```

```

CHARACTER line*80, reply*1, prefix*16, address*16

```

```

Access topology file:

```

*File giving station & station
matrix & propagation delays*

```

OPEN (UNIT=2, FILE='user:[alan]dat100/dat', STATUS='old',
1      FORM='unformatted', ERR=999)

```

```

READ(2, ERR=101) nstat, ((tdei(i, j), j=1, nstat), i=1, nstat),
1      (address(j), j=1, nstat)

```

```

101 CLOSE(UNIT=2)

```

from file

```

C      The following parameters are vestiges of an earlier incarnation
C      of the simulation and are here rendered ineffectual:

```

```

sstate=.TRUE.

```

*=state of system; set to .TRUE. in this
regenerative simulation study*

```

DO j=1, nstat
  nchek(j)=0
END DO

```

```

Set simulation parameters:

```

```

nlim=1

```

=limit to all stations' frame buffers (buffer capacity)

```

movavg=64

```

*=the number of past successful transmissions
(dating from the most recent one)
used to compute the estimated
(system) arrival rate*

```

Evaluation of nlevel (for function JLEVEL):

```

```

nlevel=0

```

```

num=1

```

```

DO UNTIL (num .LT. nstat)

```

```

nlevel=nlevel+1
num=num+2
END DO

```

System initialization:

```

rate=18.e6           =transmission rate in bits/second

limatt=16            =attempt limit

jsize=32             =jam size in bits

frgap=9.6e-6         =interframe gap size in seconds

minfr=512            =minimum frame size in bits

maxfr=12144          =maximum frame size in bits

icdet=9              =number of bits required for collision detection

icarr=2              =number of bits required for a station to sense
                      a signal following the onset of carrier
                      presence information

carroff=1.6          =number of bits required for a station to disable
                      the carrier sense signal following the
                      loss of carrier presence information

tjam=(icdet+jsize)/rate
                      =total duration of collision detection and jamming

jseed=117            =initial value for seed integer of random variate
                      generator

```

If station addresses are assigned according to physical location, then the probing algorithm may take advantage of this fact by associating a different delay with each probing level (propinquity):

Propinquity delays for 64 and 100 single cable system:

```

delay(1) = 5.3546454e-6      (200 marks)
delay(2) = 3.2983683e-6      (185 marks)

```

```

delay(3) = 2.3243423e-6      (68 marks)
delay(4) = 1.8914419e-6      (48 marks)
delay(5) = 1.5667666e-6      (25 marks)

```

```

DO i=6,16
  delay(i)=delay(5)
END DO

```

Proximity delays for 256 and 345 station multi-cable system:

```

delay(1) = 2.7128346e-5
delay(2) = 2.2734732e-5
delay(3) = 2.2381832e-5
delay(4) = 2.857823e-5
delay(5) = 1.8738403e-5
delay(6) = 1.8881852e-5
delay(7) = delay(6)
delay(8) = delay(6)
delay(9) = 5.6988353e-6

```

```

DO i=10,16
  delay(i)=delay(9)
END DO

```

```

delaymax=1.3368173e-5
      =maximum possible end-to-end propagation delay
      (seconds) in full size system

```

```

delaymax1=2.6773227e-6
      =maximum possible end-to-end propagation delay
      (seconds) in one cable system

```

```

delay(1) = 2.*delaymax1
      =the time offset necessary at level(1) [=1]
      in scheduling those transmissions
      not currently probed;
      does not include the initial offsets
      frgap and icarr/rate

```

No proximity:

```

DO i=2,16
  delay(i) = delay(1)
END DO

```

Supreme loop control:

(controls the various simulation runs)

prate=rho1(9)/(nstat*((maxfr/2. + 64.)/rate))

ktours = 10

=number of tours elapsed before statistics
are again evaluated in stopping rule

Initialize parameters for start-up of first tour:

Zero out the vector of inter-departure times:

DO i=1, noutug
 xmission(i)=0.0
END DO

index=1

=pointer to the vector of inter-departure times
(i.e., xmission(...))

atime=0.0

=time of last successful transmission

nlim=nlim+1
nstat2=2*nstat
DO i=1, nstat

 alam(i)=prate

=Poisson arrival rate of frames to i-th queue

 tcycle(i)=0.0

=station i's estimate of the duration of the
most recent probing cycle

 prefix(i)=

=station i's understanding of the binary search
probing process is represented by its
value of the address prefix...

 level(i)=0

...and the probe level

 level(i)=0 indicates contention operation;
 level(i)>0 indicates the probing level

 iflag(i)=.FALSE.

=an indicator that station i's scheduled event
is a cue for the re-evaluation of
station i's appraisal of the probing
level, level(i)

 crash=.FALSE.

=an indicator that the last activity sensed on the
channel was a collision; used in level
skipping

 tidle(i)=-frgap

=time from last event upon which station i hears
the channel idle

packet(i)=pleng(jseed,mean,maxfr,minfr)/rate
=length of station j's current packet yet
to be transmitted (in seconds)

IF (lqsize(i) .GT. 0) THEN
sked(i)=ran(jseed)*.00001
ELSE
sked(i)=1.7e38
END IF

=time of station i's next transmission attempt

in=nstat+i
sked(in)=-1./alan(i) * alog(ran(jseed))
=time of next arrival to station i

END DO

ntours=0
=number of times system returns to idle state

ltour=0
=index used to determine whether the number of
tours has increased by ktours

time=0.0
=total simulation time

Initialize accumulators, sample means, variances, and covariances:

sumtime=0.0
sumrej=0.0
sumaban=0.0
sumtimeon=0.
sumtimeon1=0.0
sumsize=0.0
sumlevel=0.0
sumpmean=0.0
sumtdep=0.0
sumdelay=0.0
sumwait=0.0
sumatt=0.0

invar=10
ware-up = .TRUE.

=flag indicating that initial 10 tours are
being used to allow each station's
estimate of the departure rate to
stabilize

stop = .FALSE.
=flag indicating whether simulation run is over

start_cpu_clock = cpu_seconds(0) -205-

```

C          =initial value of clock of elapsed cpu time
C
C  Begin new tour:
C  DO WHILE (. NOT. stop)
C
C  Initialize tour statistics:
C  tourtime=0.0
C          =time spent in current tour
C
C  timeon=0.0
C          =time channel is active during steady-state period
C          (minus all overhead)
C
C  timeon1=0.0
C          =time channel is active during steady-state period
C
C  pmean=0.0
C          =accumulator for calculating mean size of packets
C          successfully transmitted
C
C  DO i=1,17
C      plevel(i) = 0.0
C          =elapsed time spent at probe level (j-1);
C          compiled during steady-state
C  END DO
C
C  nlim1=nlim+1
C  nstat2=2*nstat
C
C  DO i=1,nstat
C      defer(i)=.false.
C          =no stations are presently deferring transmission
C
C      natt(i)=0
C          =number of transmission attempts for the frame at
C          the head of the i-th station's buffer
C
C      ndep(i)=0
C          =number of successful departures from station i;
C          compiled during steady-state period
C
C      iqsize(i)=0
C          =number of frames in the i-th station's buffer
C
C      naban(i)=0
C          =number of frame transmissions abandoned at
C          station i due to excessive attempts;
C          compiled during steady-state period
C
C      irej(i)=0
C          =number of frames rejected due to a full buffer
C          at station i; compiled during steady-state
C          period

```

utotal(i)=0.0

=total time frames spend in station i's frame buffer before being transmitted; compiled during steady-state period for frames that are successfully transmitted

fdelay(i)=0.0

=total time frames spend in station i's frame buffer before being transmitted, plus frame duration compiled during steady-state period for successfully transmitted frames

DO j=1,nlim

tk(i,j)=0.0

=amount of time station i has length (j-1); compiled during steady-state period

utime(i,j)=0.0

=elapsed waiting time of the j-th element of station i's buffer; compiled during steady-state period

END DO

tk(i,nlim1)=0.0

DO j=1,limatt

nat(i,j)=0

=number of station i's successful transmissions requiring j attempts; compiled only for packets having arrived during steady-state period

END DO

END DO

Simulation of one tour:

end_of_tour = .FALSE.

DO WHILE (.NOT. end_of_tour)

Find next event (arrival or transmission attempt):

del=sked(1)

j=1

DO i=2,nstat2

IF (sked(i) .LT. del) THEN

del=sked(i)

=next event time

j=i

=index of that next event; j > nstat implies a frame arrival to station (j - nstat); otherwise, station j is attempting a transmission

END IF
END DO

C Update simulation and tour clock:

time = time + del
tourtime = tourtime + del

C Update schedule times:

DO i=1,nstat2
sked(i)=sked(i)-del
END DO

C Update time until each station hears the channel idle:

DO i=1,nstat
tidle(i)=tidle(i)-del
END DO

C Compile steady-state statistics:

IF (sstate) THEN - sstate = TRUE.

C Increment steady-state clock:

tsted=tsted+del

C Increment time spent at current probe level:

plevel(level(1)+1) = plevel(level(1)+1) + del

C Increment amount of time station i has length iqsize(i):

DO i=1,nstat
ij=iqsize(i)+1
tk(i,ij)=tk(i,ij)+del

C Increment waiting time of all elements of station i's buffer:

DO k=1,iqsize(i)
utime(i,k)=utime(i,k)+del
END DO

END DO

END IF

C Increment each station's clock of the probe cycle:

DO i=1,nstat
tcycle(i)=tcycle(i)+del

END DO

IF (J .GT. nstat) THEN

arrival to station J

CALL arrive(J, nstat, nlim, sstate, xlam, elam, nlevel)

ELSE IF (lflag(J)) THEN

end of probing cycle as seen
by station J

DO i=1, nstat

lflag1 = lflag(i)

tcycle1 = tcycle(i)

All stations reset probe cycle clock tcycle(i):

Note that the probe cycle can end only after a successful
transmission, which in turn implies that
all stations have been scheduled
to transmit after that transmission.

etime = -tidle(i)

CALL lskip(prefix(i), etime, level(i), etime1)

etime1 is the elapsed time
the station was updated to
the current value

IF (prefix(i) .LE. '0000000000000000') THEN

according to station i, the prefix
is now '00..00' and etime1 represents
time elapsed in the new probe cycle

tcycle(i) = etime1

ELSE

station i still sees itself in the previous
probe cycle

tcycle(i) = etime1 -

toffset(prefix(i), '0000000000000000',
lflag(i), level(i))

lflag(i) = .FALSE.

END IF

All stations re-evaluate the probe level---this is
necessary in the event of a new arrival to an empty
station before any other successful transmissions or
in case the probe level reverts to 0 and all ready
stations will be scheduling their transmissions
immediately.

Past cycle time is adjusted to its proper end:

tcycle1 = tcycle1 - tcycle(i)

Station i re-evaluates the probe level:

level(i) = jlevel(elam, tcycle1, nstat, nlevel)

Set prefix accordingly.

IF (level(i) .EQ. 0) THEN

prefix(i) = ' '

ELSE

prefix(i) = '0'

m = 1

DO WHILE (m .LT. level(i))

prefix(i) = prefix(i)(1:m) // '0'

m = m + 1

END DO

END IF

IF (lflag1) THEN

Reschedule i's transmission:

delta = toffset(prefix(i), address(i), lflag(i),
level(i))

note that toffset resets lflag(i)

Remember that tidle(i) is as of sked(j)=0.0; therefore,
in the following calculations tidle(i) is adjusted
to be correct as of sked(i)=0.0

Any transmissions between this instant in time and sked(i)
will result in tidle(i)'s update and the rescheduling
of i, so the following may be done with impunity

IF ((tidle(i) - sked(i)) .GE. -frgap) THEN

IF (delta .EQ. 0.0) delta = frgap

sked(i) = tidle(i) + delta

ELSE

sked(i) = sked(i) + delta

END IF

END IF

Reset tidle(i) to avoid rescheduling in PROBE:

tidle(i) = -tcycle(i)

this should also properly align new arrivals
to the new probe cycle

END DO

* GO TO 5 *[used for aggregate states method at high load;]
allows test for end-of-twr when probe level > 0*

ELSE IF (level(j) .GT. 0) THEN

station j attempts transmission

via the probing mechanism

CALL probe(j, nstat, sstate, timeon, timeon1, nlim, tjam, limatt)

ELSE

station j's transmission is via contention

CALL contend(j, nstat, sstate, timeon,
timeon1, nlim, tjam, limatt)

¹ * Go-To 5 [aggregate state method]
* over it

Test for end of tour

end_of_tour = TRUE

DO i=1, nstat

IF (lqsize(i) NE 0) end_of_tour = FALSE

END DO

.GT. m for aggregate state method

IF (end_of_tour) THEN

IF (NOT warm_up) THEN

compile statistics

Update number of tours

ntours = ntours + 1

Look at all the stations

Sum over all stations

taqsize=0

trej=0

tnaban=0

tdep=0

tdelay=0

twait=0

tatt=0

DO j=1, nstat

aqsize(j)=0

DO k=1, nlim1

k1=k-1

aqsize(j) = aqsize(j) + k1*tk(j,k)

=weighted sum of time elapsed at

queue size k1 at station j

END DO

taqsize = taqsize + aqsize(j)

avatt(j)=0

DO k=1, limatt

avatt(j) = avatt(j) + k*nat(j,k)

=weighted sum of attempts required

at station j

END DO

trej = trej + irej(j)

=number of transmission requests

turned away from all stations

```

tnaban = tnaban + naban(j)
           =number of transmission attempts
           abandoned by all stations

```

```

tdep = tdep + tdep(j)
           =total number of successful transmissions

```

```

tdelay = tdelay + fdelay(j)
           =total frame delay

```

```

twait = twait + wtotal(j)
           =total waiting time

```

```

tatt = tatt + att(j)
           =total number of attempts

```

```

END DO

```

```

Average over all stations

```

```

taqsize = taqsize/nstat
           =average queue size (prior to averaging over time)

```

```

tnaban = tnaban/nstat
           =number of transmission requests that fail per stat

```

```

trej = trej/nstat
           =number of transmission requests turned
           away from the (full) station buffer per station

```

```

alevel = 0.0

```

```

DO i=1,17

```

```

    alevel = alevel + plevel(i)*(i-1)
           =weighted time spent at probe level (i-1)

```

```

END DO

```

```

Update pseudovalue elements for jackknife estimators

```

```

theta_time(ntours) = sumtime
theta_rej(ntours) = sumrej
theta_aban(ntours) = sumaban
theta_timeon(ntours) = sumtimeon
theta_timeon1(ntours) = sumtimeon1
theta_size(ntours) = sumsize
theta_level(ntours) = sumlevel
theta_pmean(ntours) = sumpmean
theta_tdep(ntours) = sumtdep
theta_tdelay(ntours) = sumdelay
theta_twait(ntours) = sumwait
theta_tatt(ntours) = sumatt

```

```

sumtime = sumtime + tourtime

```

```

sumrej = sumrej + trej

```

```

sumaban = sumaban + tnanban
sumtimeon = sumtimeon + timeon
sumtimeon1 = sumtimeon1 + timeon1
sumsize = sumsize + taqsize
sumlevel = sumlevel + alevel
sumpmean = sumpmean + pmean
sumtdep = sumtdep + tdep
sumdelay = sumdelay + tdelay
sumwait = sumwait + twait
sumatt = sumatt + tatt

```

```

IF (ntours GT 1) THEN
  DO i=1, (ntours-1)
    theta_time(i) = theta_time(i) + tourtime
    theta_rej(i) = theta_rej(i) + trej
    theta_aban(i) = theta_aban(i) + tnanban
    theta_timeon(i) = theta_timeon(i) + timeon
    theta_timeon1(i) = theta_timeon1(i) + timeon1
    theta_size(i) = theta_size(i) + taqsize
    theta_level(i) = theta_level(i) + alevel
    theta_pmean(i) = theta_pmean(i) + pmean
    theta_tdep(i) = theta_tdep(i) + tdep
    theta_tdelay(i) = theta_tdelay(i) + tdelay
    theta_twait(i) = theta_twait(i) + twait
    theta_tatt(i) = theta_tatt(i) + tatt
  END DO
END IF

```

END DO
END IF

minimum # of tours

```

IF (ntours GE. 40 AND ntours/ktours .NE. 1tour) THEN

```

Reset index of last cycle of ktours:

```
1tour = ntours/ktours
```

Evaluate network throughput via jackknife estimator:

*may be deleted if
statistics to be checked
every end-of-tour*

```
thrput1 = 0.0
```

```
DO i=1, ntours
```

Evaluate pseudovalues:

```
theta_thru1(i) = ntours*sumtimeon1/sumtime -  
                  (ntours-1)*theta_timeon1(i)/theta_time(i)
```

Sum pseudovalues:

```
thrput1 = thrput1 + theta_thru1(i)
```

```
END DO
```

```
thrput1 = thrput1/ntours
```

Evaluate variance of jackknifed thrput1:

```
s2thru1 = 0.0
```

```
DO i=1, ntours
```

```

      s2thru1 = s2thru1 + (theta_thru1(i) - thruput1)**2
END DO
s2thru1 = s2thru1/(ntours-1)

```

Evaluate 90% confidence interval on thruput1:

```
del = 1.645*sqrt(s2thru1/ntours)
```

Evaluate elapsed cpu time:

```
elapsed_cpu = cpu_seconds(start_cpu_clock)
```

```
IF (abs((2.*del)/thruput1) .LE. 0.05) THEN
```

desired relative width of 0.05

achieved: end simulation run

```
stop = TRUE.
```

```
ELSE IF (ntours EQ. 2000 OR.
```

```
elapsed_cpu .GE. 360000) THEN
```

end simulation run

```
stop = TRUE.
```

~ 1 hr. CPU time

```
END IF
```

(later reduced to other values)

```
IF (stop) THEN
```

calculate and output system statistics

```
WRITE(6, '(1h1, //, //, // Adaptive probing', //,
```

```
  // System parameters: ', //, //, 5x,
```

```
  // Number of stations = ', //, //, 13, //, 5x,
```

```
  // Transmission rate = 10 Mb/its per second',
```

```
  // 5x, // Buffer size at all stations = ', //, //, 12,
```

```
  // frame(s)')')
```

```
nstat, nlin
```

```
itest=0
```

```
DO i=2, 16
```

```
  IF (delay(i) NE. delay(i-1)) itest=1
```

```
END DO
```

```
IF (itest EQ. 1) THEN
```

```
  WRITE(6, '(5x, // Propinquity mode: ', //, //, 9x,
```

```
    // Delay(level 1) = ', //, //, 1pe10. 3, // seconds',
```

```
    // 9x, // Delay(level 2) = ', //, //, 1pe10. 3,
```

```
    // seconds')')
```

```
    delay(1), delay(2)
```

```
  DO i=3, 16
```

```
    IF (delay(i) NE. delay(i-1)) THEN
```

```
      WRITE(6,
```

```
        // 9x, // Delay(level ', //, //, i1, //) = ', //, //, 1pe10. 3,
```

```
        // seconds', //')')
```

```
        i, delay(i)
```

```
    END IF
```

```
  END DO
```

```
ELSE
```

```

1      WRITE(6, '(5x, ''Probing delay = '', 1pe10.3,
1      '' seconds'')')
                                     delay(1)
END IF

```

C Compute jackknife estimators and 90% confidence intervals:

```

thruput=0.0
rejrate=0.0
abanrate=0.0
taqsize=0.0
alevel=0.0
pmean=0.0
deprate=0.0
frdelay=0.0
frwait=0.0
fratt=0.0

```

DO i=1, ntours

 Calculate pseudovalues:

```

1      theta_thru(i) = ntours*sumtimeon/sumtime -
      (ntours-1)*theta_timeon(i)/theta_time(i)
1      theta_rejrate(i) = ntours*sumrej/sumtime -
      (ntours-1)*theta_rej(i)/theta_time(i)
1      theta_abanrate(i) = ntours*sumaban/sumtime -
      (ntours-1)*theta_aban(i)/theta_time(i)
1      theta_taqsize(i) = ntours*sumsize/sumtime -
      (ntours-1)*theta_size(i)/theta_time(i)
1      theta_alevel(i) = ntours*sumlevel/sumtime -
      (ntours-1)*theta_level(i)/theta_time(i)
1      theta_pmean(i) = ntours*sumpmean/sumtdep -
      (ntours-1)*theta_pmean(i)/theta_tdep(i)
1      theta_deprate(i) = ntours*sumtdep/sumtime -
      (ntours-1)*theta_tdep(i)/theta_time(i)
1      theta_tdelay(i) = ntours*sumdelay/sumtdep -
      (ntours-1)*theta_tdelay(i)/theta_tdep(i)
1      theta_twait(i) = ntours*sumwait/sumtdep -
      (ntours-1)*theta_twait(i)/theta_tdep(i)
1      theta_tatt(i) = ntours*sumatt/sumtdep -
      (ntours-1)*theta_tatt(i)/theta_tdep(i)

```

C Sum pseudovalues:

```

thruput = thruput + theta_thru(i)
rejrate = rejrate + theta_rejrate(i)
abanrate = abanrate + theta_abanrate(i)
taqsize = taqsize + theta_taqsize(i)
alevel = alevel + theta_alevel(i)
pmean = pmean + theta_pmean(i)
deprate = deprate + theta_deprate(i)
frdelay = frdelay + theta_tdelay(i)
frwait = frwait + theta_twait(i)
fratt = fratt + theta_tatt(i)

```

END DO

thrput = thrput/ntours
=ratio of time channel is spent successfully
transmitting data less overhead to
total time

rejrate = rejrate/ntours
=number of arrivals to a station's buffer
not accepted per second

abanrate = abanrate/ntours
=number of transmission attempts abandoned
per second

taqsize = taqsize/ntours
=mean queue size

alevel = alevel/ntours
=mean probe level

pmean = pmean/ntours
=mean frame size (less preamble)

deprate = deprate/ntours
=mean departure rate of successful frames from
all stations

frdelay = frdelay/ntours
=mean frame delay of successful frames

frwait = frwait/ntours
=mean frame waiting time

fratt = fratt/ntours
=mean number of attempts required for successful
transmission

Evaluate variance of jackknifed estimates:

s2thru=0.0
s2rejrate=0.0
s2abanrate=0.0
s2taqsize=0.0
s2alevel=0.0
s2pmean=0.0
s2deprate=0.0
s2frdelay=0.0
s2frwait=0.0
s2fratt=0.0

DO i=1,ntours
s2thru = s2thru + (theta_thru(i) - thrput)**2
s2rejrate = s2rejrate +
(theta_rejrate(i) - rejrate)**2
s2abanrate = s2abanrate +

```

1          (theta_abanrate(i) - abanrate)**2
      s2taqsize = s2taqsize +
1          (theta_taqsize(i) - taqsize)**2
      s2alevel = s2alevel +
1          (theta_alevel(i) - alevel)**2
      s2pmean = s2pmean +
1          (theta_pmean(i) - pmean)**2
      s2deprate = s2deprate +
1          (theta_deprate(i) - deprate)**2
      s2frdelay = s2frdelay +
1          (theta_tdelay(i) - frdelay)**2
      s2frwait = s2frwait +
1          (theta_twait(i) - frwait)**2
      s2fratt = s2fratt +
1          (theta_tatt(i) - fratt)**2

```

END DO

```

s2thru = s2thru/(ntours-1)
s2rejrate = s2rejrate/(ntours-1)
s2abanrate = s2abanrate/(ntours-1)
s2taqsize = s2taqsize/(ntours-1)
s2alevel = s2alevel/(ntours-1)
s2pmean = s2pmean/(ntours-1)
s2deprate = s2deprate/(ntours-1)
s2frdelay = s2frdelay/(ntours-1)
s2frwait = s2frwait/(ntours-1)
s2fratt = s2fratt/(ntours-1)

```

```

delta = 1.645*sqrt(s2pmean/ntours)
upper_pmean = pmean + delta
tlower_pmean = pmean - delta

```

```

rho = prate*(pmean+64.)*nstat/rate
upper_rho = prate*(upper_pmean+64.)*nstat/rate
tlower_rho = prate*(tlower_pmean+64.)*nstat/rate

```

```

1      WRITE(6, '(//, // System statistics (90% confidence',
1          ' intervals) //, //, 5x,
1          ' System load = ', 0pf6. 4, 2x,
1          ' (', 0pf6. 4, ' ', 0pf6. 4, ') //, //, 5x,
1          ' Poisson arrival rate = ', 1pe10. 3,
1          ' frames/second //, //, 5x,
1          ' Mean xmitted frame size ',
1          ' (w/o preamble) = ', 0pf8. 2, 2x, ' (', 0pf8. 2,
1          ' ', 0pf8. 2, ') bits //')
1          rho, tlower_rho, upper_rho,
1          prate, pmean, tlower_pmean, upper_pmean

```

```

delta = 1.645*sqrt(s2thru/ntours)
upper = thruput + delta
tlower = thruput - delta
upper1 = thruput1 + del
tlower1 = thruput1 - del

```

```

1      WRITE(6, '(5x, ''Data throughput = '', 0pf6. 4, 2x,
1      '(', 0pf6. 4, ' ', 0pf6. 4, ')', 5x,
1      ''Network throughput = '', 0pf6. 4, 2x,
1      '(', 0pf6. 4, ' ', 0pf6. 4, ')')')
1      thruput, tlower, upper, thruput1, tlower1,
1      upper1

      delta = 1.645*sqrt(s2alevel/ntours)
      upper = alevel + delta
      tlower = alevel - delta

      WRITE(6, '(5x, ''Mean probe level = '', 0pf6. 3,
1      2x, '(', 0pf6. 3, ' ', 0pf6. 3, ')')')
1      alevel, tlower, upper

      delta = 1.645*sqrt(s2frdelay/ntours)
      upper = frdelay + delta
      tlower = frdelay - delta

      WRITE(6, '(5x,
1      ''Mean frame delay (incl. preamble) = '',
1      1pe10. 3, 2x, '(', 1pe10. 3, ' ', 1pe10. 3,
1      '' secs')')
1      frdelay, tlower, upper

      delta = 1.645*sqrt(s2frwait/ntours)
      upper = frwait + delta
      tlower = frwait - delta

      WRITE(6, '(5x,
1      ''Mean waiting time = '', 1pe10. 3, 2x, '(',
1      1pe10. 3, ' ', 1pe10. 3, '' secs')')
1      frwait, tlower, upper

      delta = 1.645*sqrt(s2taqsize/ntours)
      upper = taqsize + delta
      tlower = taqsize - delta

      WRITE(6, '(5x, ''Mean queue size = '', 1pe10. 3,
1      2x, '(', 1pe10. 3, ' ', 1pe10. 3, '' frames')')
1      taqsize, tlower, upper

      delta = 1.645*sqrt(s2fratt/ntours)
      upper = fratt + delta
      tlower = fratt - delta

      WRITE(6, '(5x, ''Mean number of attempts required = '',
1      1pe10. 3, 2x, '(', 1pe10. 3, ' ', 1pe10. 3,
1      ''')')
1      fratt, tlower, upper

```

```

      delta = 1.645*sqrt(s2deprate/ntours)
      upper = (deprate + delta)/nstat
      tlower = (deprate - delta)/nstat
      deprate = deprate/nstat

      WRITE(6, '(5x, "Mean departure rate = ", 1pe10.3,
1          2x, "(", 1pe10.3, " ", 1pe10.3, " frames",
1          " //second")')
1          deprate, tlower, upper

```

```

      delta = 1.645*sqrt(s2re_rate/ntours)
      upper = rejrate + delta
      tlower = rejrate - delta

      WRITE(6, '(5x, "Mean rejection rate = ", 1pe10.3,
1          2x, "(", 1pe10.3, " ", 1pe10.3, " frames",
1          " //second")')
1          rejrate, tlower, upper

```

```

      delta = 1.645*sqrt(s2abanrate/ntours)
      upper = abanrate + delta
      tlower = abanrate - delta

      WRITE(6, '(5x, "Mean abandonment rate = ", 1pe10.3,
1          2x, "(", 1pe10.3, " ", 1pe10.3, " frames",
1          " //second")')
1          abanrate, tlower, upper

```

```

      simtime = sumtime/ntours
      cpu = elapsed_cpu/100

      WRITE(6, '(/, " Simulation statistics:", //, 5x,
1          "Total time simulated (incl. discarded",
1          " initial 10 tours) = ", 0pf8.2,
1          " seconds", //, 5x,
1          "Mean duration of each tour = ", 1pe10.3,
1          " seconds", //, 5x, "Number of tours = ",
1          14, //, 5x, "Total CPU seconds used = ",
1          0pf8.2, " seconds", //, //)')
1          time, simtime, ntours, cpu

```

END IF

END IF last tour of simulation

END IF

END testing for end of simulation

ELSE

still warming up

iwarm = iwarm - 1

IF (iwarm .EQ. 0) warm_up = .FALSE.

END IF

END warm_up test

END IF

```

C          END, IF end of tour
C
C      * END IF          END IF arrival, end of probe cycle, probing, or
C                          contention
C      deleted if aggregate states method employed
C      END DO
C          test on end_of_tour
C
C      END DO
C          test on stop (end of simulation on rho1)
C
C      END DO
C          i=i,9

```

```

C      WRITE(6, '(1H1)')

```

```

C      STOP

```

```

C 999      WRITE(6,*) 'Error in OPENing data file'
C      STOP

```

```

C      END

```

```

C      FUNCTION pleng(jseed, mean, maxfr, minfr)

```

```

C      pleng=maxfr/2 + 64.

```

```

C      RETURN

```

```

C      END

```

Fixed length frames

```

C      FUNCTION pleng(jseed, mean, maxfr, minfr)

```

```

C10      pleng=-1. *mean*alog(ran(jseed))

```

```

C          packet length is derived first as an
C          exponentially distributed
C          variate with the given mean.

```

```

C      IF (pleng .GT. maxfr) THEN
C          pleng=maxfr
C      ELSE IF (pleng .LT. minfr) THEN
C          pleng=minfr
C      END IF

```

```

C          but the length must be within the limits
C          established by minfr and maxfr,

```

```

C      pleng=int(pleng/8.)*8. + 64.

```

*Exponential
dist.
frames*

and a packet must equal an integer number
of octets plus the 64 bit preamble

RETURN
END

Function PLENG provides a bimodal distribution for
the frame lengths

FUNCTION pleng(jseed, mean, maxfr, minfr)

IF (ran(jseed) .LT. .2) THEN
 pleng=maxfr+64.
ELSE
 pleng=minfr+64.
END IF

RETURN
END

bimodal
distribution

FUNCTION jlevel(ela, tcycle, nstat, nlevel)
LOGICAL stop

p=1. -exp(-ela*tcycle)

ilevel=0
nexp=1
stop=.FALSE.

DO WHILE ((1.-p)**nexp .GE. 0.5 .AND. .NOT. stop)
 IF (nexp .GE. nstat) stop=.TRUE.
 ilevel=ilevel+1
 nexp=nexp*2
END DO

IF (ilevel .NE. 0) ilevel=ilevel-1

jlevel=nlevel-ilevel

RETURN
END

SUBROUTINE lskip(prefix, etime, level, etime1)
CHARACTER*16 prefix, prefix1
COMMON /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot
1 /blk15/ icarr, carroff
1 /blk6/ delay(16)
1 /blk65/ crash

LOGICAL stop, crash, skip

slack is less than the propagation delay between two adjacent marks on the coax line and should provide some degree of insurance against numerical errors:

slack=1. e-8

etime1=etime

i=1

DO WHILE (prefix(1:1) .NE. ' ')

i=i+1

END DO

prefix1=prefix

m=i-1

stop = .FALSE.

skip = .FALSE.

IF (etime1 .LT. (fngap + icarr/rate)) THEN

stop = .TRUE.

ELSE

etime1 = etime1 - fngap - icarr/rate

END IF

DO WHILE (.NOT. stop)

IF (prefix1(m:m) EQ '0') THEN

prefix1(m:m)='1'

IF (etime1 GE. (delay(m)-slack)) THEN

skip = .TRUE.

etime1 = etime1 - delay(m)

prefix(m:m) = prefix1(m:m)

ELSE

stop = .TRUE.

END IF

ELSE IF (m .GT. level) THEN

prefix1(m:m)=' '

k=m

m=m-1

etime1 = etime1 - delay(k)/2.

CALL add1(prefix1,m)

DO WHILE (prefix1(m:m) .EQ. '0' .AND. m .GT. level)

prefix1(m:m)=' '

m=m-1

END DO

IF (etime1 .GE. (delay(m)/2. -slack)) THEN

skip = .TRUE.

etime1 = etime1 - delay(m)/2

prefix = prefix1

ELSE

stop = .TRUE.

etime1 = etime1 + delay(k)/2

m=k

END IF

ELSE

CALL add1(prefix1,m)

IF (etime1 .GE. (delay(m)-slack)) THEN

```

        skip = TRUE
        etime1 = etime1 - delay(m)
        prefix = prefix1
    ELSE
        stop = TRUE
    END IF
END IF
END DO

```

C Upon return, etime1 gives the elapsed time since the
C prefix was updated to the current new value

```

IF (crash .AND. skip) prefix = prefix(1:m)//'0'

```

```

RETURN
END

```

```

SUBROUTINE update(prefix, address, level)
CHARACTER*16 prefix, address, prefix1

```

```

i=1
DO WHILE (prefix(i:i) NE. ' ')
    i=i+1
END DO

```

```

m=i-1

```

```

IF (m EQ. 0) THEN
    WRITE(6,*) 'Subroutine UPDATE called improperly'
    STOP
END IF

```

```

DO WHILE (prefix(1:m) NE. address(1:m) .AND. m NE. 0)
    IF (prefix(m:m) EQ. '0') THEN
        prefix(m:m)='1'
    ELSE IF (m GT. level) THEN
        prefix(m:m)=' '
        m=m-1
        CALL add1(prefix, m)
        DO WHILE (prefix(m:m) EQ. '0' .AND. m GT. level)
            prefix(m:m) = ' '
            m=m-1
        END DO
    ELSE
        CALL add1(prefix, m)
    END IF
END DO

```

```

IF (m NE. 0) THEN
    IF (prefix(m:m) EQ. '0') THEN
        prefix(m:m)='1'
    ELSE IF (m GT. level) THEN
        prefix(m:m)=' '
        m=m-1
    END IF
END IF

```

```

DO WHILE (prefix(m:m) .EQ. '0' .AND. m .GT. level)
    prefix(m:m) = ' '
    m=m-1
END DO
ELSE
    CALL add1(prefix,m)
END IF
END IF

RETURN
END

```

Given a station's address and the station's value for the current prefix, subroutine TOFFSET computes the delay required until transmission

1) if the station's address is greater than the prefix, then the prefix value is incremented to match the address. The least significant zero bits of the prefix vanish with each increment that results in trailing zeroes until the number of bits is equal to the probing level. A delay associated with the number of bits currently in the prefix is added with each increment. If any least significant zeroes are removed from the prefix, the delay added is that corresponding to one half the delay accorded to the original number of bits in the prefix plus half the delay associated with the resultant number of bits in the prefix.

2) if the station's address is less than the prefix and probing is in effect (i.e., if level .GT. 0), then the above procedure is used to provide the delay until the cycle begins anew at '0000000000000000' (or less). The flag LFLAG is then set to indicate that at the scheduled time, the station must re-evaluate the probe level and again schedule transmission.

```

FUNCTION toffset(prefix,address,lflag,level)
CHARACTER*16 prefix,address,prefix1
LOGICAL lflag
COMMON /blk1/ mean,maxfr,minfr,rate,frgap,limbo,islot
1      /blk15/ icarr,carroff
1      /blk6/ delay(16)

lflag=.FALSE.

toffset=0 0

i=1
DO WHILE (prefix(i:i) .NE. ' ')
    i=i+1
END DO

```

prefix1=prefix

m=1-1

IF (m EQ 0) RETURN

C If the station is not to transmit immediately, a delay of
C frgap and icarr/rate is introduced to account for
C the possibility of the currently probed station(s)
C transmitting:

IF (prefix1(1:m) NE address(1:m)) toffset=frgap+icarr/rate

DO WHILE (prefix1(1:m) NE address(1:m) AND NOT lflag)

IF (prefix1(m:m) EQ '0') THEN

prefix1(m:m)='1'

toffset=toffset+delay(m)

ELSE IF (m GT level) THEN

prefix1(m:m)=' '

m=m-1

toffset = toffset + delay(m+1)/2.

CALL add1(prefix1,m)

DO WHILE (prefix1(m:m) EQ '0' AND m GT level)

prefix1(m:m)=' '

m=m-1

END DO

toffset = toffset + delay(m)/2

ELSE

CALL add1(prefix1,m)

toffset = toffset + delay(m)

END IF

IF (prefix1(1:m) LE '0000000000000000') lflag= TRUE.

END DO

RETURN

END

SUBROUTINE add1(prefix,m)

CHARACTER prefix*(*)

m1=m

DO WHILE (m1 NE 0)

IF (prefix(m1:m1) EQ '0') THEN

prefix(m1:m1)='1'

RETURN

ELSE

prefix(m1:m1)='0'

m1=m1-1

END IF

END DO

RETURN

END

```
SUBROUTINE traffic(nstat)
COMMON /blk85/ index, movavg, time, elam, atime, xmission(65)

xmission(index) = time - atime

atime = time

IF (index EQ. movavg) THEN
  index=1
ELSE
  index=index+1
END IF

i=1
sum=0.0
DO WHILE (xmission(1) NE. 0.0 AND i LE. movavg)
  sum = sum + xmission(1)
  i=i+1
END DO

i=i-1

elam = 1 / (sum./i)

Normalize to find the rate per station

elam = elam/nstat

RETURN
END
```

Subroutine IDLE computes the time all stations sense the channel idle with respect to the m transmitting stations listed in index(m). Their transmissions end at end(index(m)).

```
SUBROUTINE idle(nstat, index, end, m)
DIMENSION index(m), end(400)
COMMON jseed
1 /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot
1 /blk13/ icarr, carroff
1 /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)
1 /blk33/ tdel(400, 400)

DO k=1, nstat
  tidle(k)=0.0
  DO kn=1, m
    temp=end(index(kn))+tdel(index(kn), k)
    IF (index(kn) NE. k) temp=temp+carroff/rate
    IF (temp GT tidle(k)) tidle(k)=temp
  END DO
END DO
```

RETURN
END

Subroutine CONTENTD is called when the system is operating
in its 1-persistent CSMA mode.

Station J is attempting a transmission; note that when
subroutine CONTENTD is called, all elapsed times have been
updated to the time of J's attempt

```
SUBROUTINE contend(J, nstat, sstate, timeon,
1          timeon1, nlim, tjam, limatt)
DIMENSION index1(400), index2(400), index3(400), ejam(400)
COMMON jseed
1      /blk1/ mean, maxfr, minfr, rate, frgap, limbko, izlot
1      /blk15/ icarr, carroff
1      /blk2/ sked(800), tidle(400), packet(400), wtime(400, 50)
1      /blk25/ lqsize(400), natt(400)
1      /blk3/ ndep(400), wtotal(400), fdelay(400), naban(400)
1      /blk35/ tdel(400, 400)
1      /blk4/ nchek(400)
1      /blk45/ defer
1      /blk55/ nat(400, 40)
1      /blk65/ crash
1      /blk7/ prefix(400), address(400)
1      /blk75/ lflag(400)
1      /blk8/ level(400), tcycle(400)

LOGICAL sstate, defer(400), cont, lflag, crash
CHARACTER*16 prefix, address
```

Reset relevant values:

```
DO k=1, nstat
    ejam(k)=0.0
    tidle(k)=0.0
END DO
```

Search for transmission attempts scheduled to occur before
J's transmission is sensed:

```
n=0
m=0
DO k=1, nstat
    cont=.true.
1      =contention status; .false. implies a station
        is either deferring transmission or
        is scheduled for transmission much later
    DO l=1, nstat
        IF (sked(k) .GT. (sked(l)+tdel(k, l)+(icarr/rate)))
            cont=.false.
1    END DO
```

```

IF (cont.EQ. .true. .OR. k.EQ. j) THEN
  m=m+1
  =number of stations in contention, including station j
  defer(k)=.false.
                                defer(k) of contending station and
                                station j reset

  index1(m)=k
                                =station number of a contending station

END IF
END DO

IF (m EQ. 1) THEN
                                no collision, unless a deferring
                                station transmits

Test for possible collision due to deferring stations:

  Reset the time at which j's jam ends (to be used in the
  event of deferring station collision):

  ejam(j)=1.7e38

  Test for stations who have a transmission scheduled during the
  ongoing transmission.

  DO k=1,nstat
    IF (sked(k) LT (packet(j)+tdel(k,j)+(carroff/rate))
    1 AND (defer(k))) THEN
      temp=sked(k)+tdel(k,j)+tjam
      IF (temp LT ejam(j)) THEN
        ejam(j)=temp
        kd=k
      END IF
    END IF
  END DO

  IF (ejam(j) LT 1.7e38) THEN
                                a deferring station has caused
                                a collision

  Set collision indicator:

  crash = .TRUE.

  mtemp=m+1
  DO WHILE (mtemp.NE. m)
                                until no more deferring stations are in contention

  mtemp=m

  Compute time until all stations sense the channel idle:

  CALL idle(nstat,index1,ejam,m)

```

Find the deferring stations that are thus in conflict:

DO k=1,nstat

IF (sked(k) .LT. tidle(k) .AND. (defer(k))) THEN

Reset status of transmitting station:

defer(k)=.false.

m=m+1

index1(m)=k

=index of all transmitting stations in contention

ejam(k)=sked(k)+tjam

=time at which the k-th station's jam ends

END IF

END DO

END DO

Compute time all stations sense channel idle:

CALL idle(nstat, index1, ejam, m)

All stations enter probing mode and reset their prefixes:

DO i=1,nstat

level(i)=1

prefix(i)='0'

defer(i)=.FALSE.

All stations reset their probe cycle clocks:

tcycle(i)=-tidle(i)

END DO

Update number of attempts and reschedule transmissions
for colliding stations:

DO k=1, m

CALL att(index1(k), linatt, nlin, sstate, ejam, nstat)

END DO

Reschedule all other transmissions:

k=1

DO i=1,nstat

IF (index1(k) .EQ. i) THEN

IF (k .LT. m) k=k+1

ELSE IF (sked(i) .LT. 1.7e30) THEN

delta = toffset(prefix(i), address(i),

iflag(i), level(i))

IF (delta .EQ. 0.0) delta=frgap

sked(i) = tidle(i) + delta

```
END IF  
END DO
```

```
ELSE
```

```
    successful transmission by station j
```

```
    CALL success(j, sstate, timeon, timeon1, nlim, nstat)
```

```
    All stations reset their probe levels:
```

```
    DO k=1, nstat
```

```
        prefix(k)=' '
```

```
        level(k)=0
```

```
    END DO
```

```
    Compute time all stations hear channel idle:
```

```
    ejam(j)=packet(j)
```

```
    CALL idle(nstat, index1, ejam, m)
```

```
    IF (iqsize(j) .GE. 1) THEN
```

```
        Schedule next transmission:
```

```
        sked(j)=packet(j) + frgap
```

```
        Generate packet length of next message to be transmitted:
```

```
        packet(j)= pleng(jsqed, mean, maxfr, minfr)/rate
```

```
        update frame size for packet now  
        to be transmitted
```

```
    ELSE
```

```
        sked(j)=1.7e38
```

```
        with no frames in station j's queue,  
        the next transmission is  
        scheduled to an improbably long  
        interval
```

```
    END IF
```

```
Are any stations deferring transmission?
```

```
DO k=1, nstat
```

```
    IF (sked(k) .LT. tidle(k) .AND. k .NE. j) THEN  
        defer(k)=.true.
```

```
        i.e., station k's transmission  
        is deferred
```

```
    sked(k)=tidle(k)+frgap
```

```
END IF
```

END DO

END IF

ELSE

collision!

Set collision indicator:

crash = .TRUE.

Determine the time all stations in contention end their transmissions, with respect to their own locations

DO k=1, n

ejam(index1(k))=1.7e38

DO l=1, n

temp=sked(index1(l))+tdel(index1(l), index1(k))+tjam

IF (temp .LT. ejam(index1(k)) .AND. l .NE. k)

1 ejam(index1(k))=temp

 =time from transmission
 event to end of
 jam for contending
 station index1(k)

i. e., a contending station jams upon
discovering contention

END DO

END DO

Look for deferring stations who decide to transmit:

mtemp=m+1

m1=m

DO WHILE (mtemp .NE. m)

 mtemp=m

 Compute time until all stations sense the channel idle as
 a result of jamming stations:

 CALL idle(nstat, index1, ejam, m)

 Find the deferring stations that are thus in conflict:

DO k=1, nstat

 IF (sked(k) .LT. tidle(k) .AND. (defer(k))) THEN

 Reset status of transmitting station:

```

      m=m+1
      index1(m)=k
          =index of all transmitting stations in contention

      ejam(k)=sked(k)+tjam
          =time at which the k-th station's jam ends

      END IF
    END DO
  END DO

  Compute time all stations hear channel idle:
  CALL idle(nstat, index1, ejam, m)

  All stations enter probing mode and reset their prefixes:

  DO i=1, nstat
    level(i)=1
    prefix(i)=0
    defer(i)=.FALSE.

    All stations reset their probe cycle clocks:

    tcycle(i)=-tidle(i)

  END DO

  Update number of attempts and reschedule transmissions
  for colliding stations:

  DO k=1, m
    CALL att(index1(k), limatt, nlim, sstate, ejam, nstat)
  END DO

  Reschedule all other transmissions:

  k=1
  DO i=1, nstat
    IF (index1(k) .EQ. i) THEN
      IF (k .LT. m) k=k+1
    ELSE IF (sked(i) .LT. 1.7*38) THEN
      delta = toffset(prefix(i), address(i),
1          lflag(i), level(i))
      IF (delta .EQ. 0.0) delta=frgap
      sked(i) = tidle(i) + delta
    END IF
  END DO

END IF

RETURN
END

```

C Subroutine PROBE is called when the system is operating
C in its adaptive probing mode

C Station J is attempting a transmission; note that when
C subroutine PROBE is called, all elapsed times have been
C updated to the time of J's attempt

```
SUBROUTINE probe(j, nstat, sstate, timeon,  
1 timeon1, nlim, tjam, limatt)  
DIMENSION index1(400), index2(400), index3(400), ejam(400)  
COMMON jseed  
1 /blk1/ mean, maxfr, minfr, rate, frgap, limbo, islot  
1 /blk15/ icarr, carroff  
1 /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)  
1 /blk25/ lqsize(400), natt(400)  
1 /blk3/ ndep(400), utotal(400), fdelay(400), naban(400)  
1 /blk35/ tdel(400, 400)  
1 /blk4/ nchek(400)  
1 /blk45/ defer  
1 /blk55/ nat(400, 40)  
1 /blk65/ crash  
1 /blk7/ prefix(400), address(400)  
1 /blk75/ lflag(400)  
1 /blk8/ level(400), tcycle(400)
```

```
LOGICAL sstate, defer(400), cont, lflag, crash  
CHARACTER*16 prefix, address
```

C Search for transmission attempts scheduled to occur before
C J's transmission is sensed:

```
DO k=1, nstat  
cont=.true.
```

=contention status; .false. implies a station
is scheduled for transmission much later

```
IF (lflag(k)) THEN  
cont=.FALSE.
```

```
ELSE
```

```
DO l=1, nstat
```

```
IF (.NOT. lflag(l) .AND.
```

```
1 sked(k) .GT. (sked(l)+tdel(k, l)+(icarr/rate)))
```

```
1 cont=.FALSE.
```

```
END DO
```

```
END IF
```

```
IF (cont .EQ. TRUE. OR. k .EQ. j) THEN
```

```
etime = -tidle(k) + sked(k)
```

=elapsed time that station k will have sensed the
channel idle at the start of its collision

Update station k's prefix as according to etime:

```
CALL lskip(prefix(k), etime, level(k), etime1)
```

IF (etime1 .NE. etime) THEN

the prefix has been changed
due to the elapsed idle time;
therefore, reschedule the
transmission

Reset tidle(k) to avoid further rescheduling:

tidle(k) = -etime1

sked(k) = sked(k) - etime1 + toffset(prefix(k),
address(k), iflag(k), level(k))

Note how -etime1 refers the offset value to the
instant when the current prefix was determined

END IF

END IF
END DO

IF (sked(j) .EQ. 0.0) THEN

the re-evaluation of j's prefix
due to the elapsed idle time
has not affected j's scheduled
transmission (and may not have
changed that of the other stations)

Search for transmission-attempts scheduled to occur before
j's transmission is sensed:

n=0

m=0

DO k=1, nstat

cont=.true.

=contention status, false. implies a station
is scheduled for transmission much later

IF (iflag(k)) THEN

cont=.FALSE.

ELSE

DO l=1, nstat

IF (sked(k) .GT. (sked(l)+tidle(k,l)+(icurr/rate)))
cont=.FALSE.

END DO

END IF

IF (cont .EQ. TRUE. .OR. k .EQ. j) THEN

m=m+1

=number of stations in contention, including station j

defer(k)=.false.

defer(k) of contending station and
station j reset

index1(m)=k

```

C          =station number of a contending station

      END IF
      END DO

      ELSE

      RETURN

      END IF

C      Reset relevant values:

      DO k=1,nstat
        ejam(k)=0.0
        tidle(k)=0.0
      END DO

      IF (m .EQ. 1) THEN
C          no collision;
C          successful transmission by station J

        CALL success(j, sstate, timeon, timeon1, nlim, nstat)

C      Compute time all stations hear channel idle:

        ejam(j)=packet(j)
        CALL idle(nstat, index1, ejam, m)

C      Update all stations' prefixes and probe levels:

        CALL update(prefix(j), address(j), level(j))
        DO k=1, nstat
          prefix(k)=prefix(j)
          level(k)=level(j)
        END DO

        IF (lqsize(j) .GE. 1) THEN

C      Schedule next transmission:

          IF (prefix(j) .LE. '0000000000000000') THEN
C              this was the last
C              transmission of the probe
C              cycle; station J schedules
C              itself to re-evaluate the
C              probe cycle immediately

            lflag(j)=.TRUE.
            sked(j)=packet(j)

          ELSE

C              schedule re-evaluation whenever possible

            sked(j)=packet(j) +

```

```

1      toffset(prefix(j), address(j), iflag(j), level(j))

      END IF

      Generate packet length of next message to be transmitted:

      packet(j) = pleng(jseed, mean, maxfr, minfr)/rate

      update frame size for packet now
      to be transmitted

ELSE
      sked(j) = 1.7e38

      with no frames in station j's queue,
      the next transmission is
      scheduled to an improbably long
      interval

END IF

Reschedule all other transmissions (effectively cancelling
any currently scheduled probe level re-evaluation):

DO i=1, nstat
      IF (i .NE. j AND sked(i) .LT. 1.7e38) THEN

          If updating the prefix to the next value has led to
          a new probe cycle, then set iflag(i) to have
          the probe level re-evaluated:

          IF (prefix(i) .LE. '0000000000000000') THEN
              iflag(i) = .TRUE.

              Schedule ASAP

              sked(i) = tidle(i)

          ELSE

              schedule according to present probe cycle

              delta = toffset(prefix(i), address(i),
              iflag(i), level(i))

              Guarantee frgap delay between transmissions:

              IF (delta .EQ. 0.0) delta = frgap

              sked(i) = tidle(i) + delta

          END IF

      END IF

END DO

```

ELSE

collision

Set collision indicator

crash = TRUE.

Determine the time all stations in contention end their transmissions, with respect to their own locations:

DO k=1,m

ejam(index1(k))=1.7e38

DO l=1,m

temp=sked(index1(l))+tdel(index1(l),index1(k))+tjam

IF (temp LT ejam(index1(k)) AND l.NE.k)

1

ejam(index1(k))=temp

=time from transmission
event to end of
jam for contending
station index1(k)

i.e., a contending station jams upon
discovering contention

END DO

END DO

Compute time all stations hear channel idle:

CALL idle(nstat,index1,ejam,m)

All stations continue probing

DO i=1,nstat

k=1

DO WHILE (prefix(i)(k:k).NE.'')

k=k+1

END DO

k=k-1

IF (k.EQ.0) THEN

prefix(i)='0'

ELSE

prefix(i) = prefix(i)(1:k) //'0'

END IF

END DO

Update number of attempts and reschedule transmissions
for colliding stations:

DO k=1,m

CALL att(index1(k),limatt,nlim,estate,ejam,nstat)

Reschedule all other transmissions

```

k=1
DO i=1,nstat
  IF (index1(k) EQ 1) THEN
    IF (k LT m) k=k+1
  ELSE IF (sked(i) LT 1.7e38) THEN
    delta = toffset(prefix(i),address(i),
1          iflag(i),level(i))
    IF (delta EQ 0.0) delta=frgap
    sked(i) = tidle(i) + delta
  END IF
END DO

END IF

```

RETURN
END

Subroutine SUCCESS updates station j's status after a
successful transmission

```

SUBROUTINE success(j, sstate, timeon, timeon1, nlim, nstat)
COMMON /blk1/ mean, maxfr, minfr, rate, frgap, limbko, islot
1      /blk2/ sked(800), tidle(400), packet(400), wtime(400, 50)
1      /blk25/ lqsize(400), natt(400)
1      /blk3/ ndep(400), wtotal(400), fdelay(400), naban(400)
1      /blk4/ nchek(400)
1      /blk55/ nat(400, 40)
1      /blk65/ crash
1      /blk85/ index, moveavg, time, elam, atime, xmission(65)
1      /blk9/ pmean

```

LOGICAL sstate, crash

CALL traffic(nstat)

Reset collision indicator:

crash = FALSE

natt(j) = natt(j) + 1

success on (new) natt(j)-th attempt

IF (sstate) THEN sstate = .TRUE.

compile steady-state statistics

Increment time channel is successfully active:

```

timeon = timeon + (packet(j)-64 /rate - 152 /rate)
               (subtracting off the time spent transmitting
               the 64 bit preamble and an
               assumed 19 octets of overhead)

```

```

timeon1 = timeon1 + packet(j)

```

```

Test whether successful transmission had arrived during
steady-state period

```

```

IF (nchek(j) NE. 0) THEN
                                nchek(j) = 0 0 j
                                update counter

```

```

nchek(j)=nchek(j)-1

```

```

ELSE
    compile statistics

```

```

pmean = pmean + (packet(j)*rate - 64 )
ndep(j)=ndep(j)+1
wtotat(j)=wtotat(j)+wtime(j,1)
fdelay(j)=fdelay(j)+wtime(j,1)+packet(j)
nat(j,natt(j)) = nat(j,natt(j)) + 1

```

```

END IF

```

```

Advance the sequencing of waiting times in j's queue

```

```

DO i=2,nlim
    ii=i-1
    wtime(j,ii)=wtime(j,i)
END DO
wtime(j,nlim)=0 0

```

```

END IF

```

```

Update station j's queue size

```

```

iqsize(j)=iqsize(j)-1

```

```

Reset number of transmission attempts

```

```

natt(j)=0

```

```

RETURN
END

```

```

Unsuccessful transmission attempt by station 1 is rescheduled
or abandoned by subroutine ATT

```

```

SUBROUTINE att(1,limatt,nlim,ssstate,ejam,nstat)
DIMENSION ejam(nstat)

```

```

COMMON jseed
1 /blk1/ mean,maxfr,minfr,rate,frgap,limbko,islott
1 /blk2/ sked(300),ticle(400),packet(400),wtime(400,50)
1 /blk25/ iqsize(400),natt(400)
1 /blk3/ ndep(400),wtotat(400),fdelay(400),naban(400)

```

```

1      /blk7/ prefix(400), address(400)
1      /blk75/ iflag(400)
1      /blk8/ level(400), tcycle(400)

```

```

LOGICAL sstate, l+lag
CHARACTER*16 prefix, address

```

```

C      Increment number of transmission attempts
      natt(1)=natt(1)+1

```

```

C      Has the attempt limit been reached?

```

```

      IF (natt(1) EQ limatt) THEN
C      abandon the packet

```

```

      natt(1)=0
      IF (lqsize(1) GT 1) THEN

```

```

1      delta = toffset(prefix(1), address(1),
                        iflag(1), level(1))
      IF (delta EQ 0 0) delta=frgap
      sked(1) = tidle(1) + delta

      packet(1)=pleng(jsced, mean, maxfr, minfr)/rate

```

```

      ELSE
      sked(1)=1 7e38
      END IF

```

```

      lqsize(1)=lqsize(1)-1

```

sstate = true.

```

      IF (sstate) THEN
      IF (nchek(1) NE 0) THEN
        nchek(1)=nchek(1)-1

```

nchek(1) = 0. 0

```

      ELSE
      naban(1)=naban(1)+1

```

```

      END IF
      DO j=2, nlim
      j1=j-1
      utime(1, j1)=utime(1, j)

```

```

      END DO
      utime(1, nlim)=0
      END IF

```

```

      ELSE
C      prepare next attempt

```

```

1      delta = toffset(prefix(1), address(1),
                        iflag(1), level(1))
      IF (delta EQ 0 0) delta=frgap
      sked(1) = tidle(1) + delta

```

```

      END IF

```

```

      RETURN
      END

```

```

SUBROUTINE arrive(j, nstat, nlim, sstate, alam, elam, nlevel)
DIMENSION alam(nstat), elam(nstat)
COMMON jseed
1      /blk1/ mean, maxfr, minfr, rate, frgap, limbko, islot
1      /blk2/ sked(800), tidle(400), packet(400), utime(400, 50)
1      /blk25/ iqsize(400), natt(400)
1      /blk45/ defer(400)
1      /blk5/ irej(400)
1      /blk65/ crash
1      /blk7/ prefix(400), address(400)
1      /blk75/ lflag(400)
1      /blk8/ level(400), tcycle(400)

```

```

LOGICAL sstate, lflag, defer, crash
CHARACTER*16 prefix, address

```

```

j1=j-nstat
sked(j)=-1 /alam(j1) * alog(ran(jseed))
j=j1

```

```

IF (iqsize(j) EQ nlim) THEN
  IF (sstate) THEN
    irej(j)=irej(j)+1
  END IF
ELSE
  iqsize(j)=iqsize(j)+1
  utime(j, iqsize(j))=0.
END IF

```

```

IF (iqsize(j) EQ 1) THEN

```

C Generate new packet length

```

packet(j)=pleng(jseed, mean, maxfr, minfr)/rate

```

```

IF (tidle(j) GE -fgap) THEN

```

```

1      IF (prefix(j) LE. '0000000000000000' AND NOT crash
          AND. level(j) NE. 0) THEN

```

C Arrival results in scheduling of re-evaluation of
C probe level.

C Schedule re-evaluation of level(j) ASAP:

```

sked(j) = tidle(j) + frgap
lflag(j) = TRUE.

```

```

ELSE

```

C Calculate transmission time

```

1      delta = toffset(prefix(j), address(j), lflag(j),
                      level(j))

```

```

IF (delta EQ. 0.0) delta = frgap

```

```

        IF (tidle(j) GT 0.0 AND level(j) EQ 0)
            defer(j) = TRUE.
1       sked(j) = tidle(j) + delta
        END IF

    ELSE IF (level(j) .NE. 0 AND
1       -tidle(j) .GT. toffset(prefix(j), '0000000000000000',
1       iflag(j), level(j))) THEN

        the arrival is transmitted immediately,
        in the contention mode
        (arrival outside of
        probe cycle)

        sked(j) = 0.0
        level(j) = 0
        prefix(j) = ' '
        iflag(j) = FALSE

    ELSE
        arrival occurs during ongoing probe cycle (level .GE. 0)

        etime = -tidle(j)
        etime1 = 0.0
        IF (level(j) .NE. 0) THEN
            CALL lskip(prefix(j), etime, level(j), etime1)

            tidle(j) reset to align with prefix boundary:
            tidle(j) = -etime1

        END IF

        delta = toffset(prefix(j), address(j),
1       iflag(j), level(j))

        IF (etime1 GT 0.0 AND delta .EQ. 0.0) THEN
            arrival occurs during probing
            of its own prefix; since it
            may have arrived very close to
            the time when the next prefix
            gains access to the channel...

            schedule re-evaluation of probe level:

            sked(j) = -etime1 + toffset(prefix(j), '0000000000000000',
1       iflag(j), level(j))

            iflag(j) = TRUE.

        ELSE

            sked(j) = -etime1 + delta

        END IF
    
```

END IF

END IF

RETURN

END

Program TOPO.FOR

Program TOPO.FOR uses as input the file TOPO.DAT to prepare a file MATSTOR.DAT for use by the adaptive probing and Ethernet simulations. The input file must answer the questions appearing in TOPO's interactive mode, as seen in the display programs ETHER.FOR and APROBE.FOR. Ultimately, TOPO provides the number of stations, the matrix of propagation delays, and the list of station addresses.

```
DIMENSION itopo(201,50), rept(400), stat(1500), itest(50),  
1      tdel(600,600), address(600)  
CHARACTER reply*1, reply1*1, more*1, line*80, address*16
```

```
CALL print('user:[alan]csma1.txt')  
READ(5, '(a)') line
```

```
CALL print('user:[alan]csma1b.txt')  
READ(5, '(a)') line  
CALL print('user:[alan]csma1c.txt')
```

```
OPEN (UNIT=5, FILE='user:[alan]topo.dat', STATUS='old', ERR=999)
```

```
jseed=117  
rate=18.06  
nbits=9
```

Initialization.

```
maxnum=2.*nbits
```

```
DO j=1, 50  
  itest(j)=0  
  DO i=1, 201  
    itopo(i, j)=0  
  END DO  
END DO
```

```
DO j=1, 600  
  DO i=1, 600  
    tdel(i, j)=0.0  
  END DO  
END DO
```

```
ncable=0  
nrept=0  
nstat=0
```

Begin program:

```
more='Y'  
DO WHILE (more.EQ. 'Y')
```

```
WRITE(6, '(1h0, "Enter cable number: ")')
```

READ(5,*) 1cable

IF (1cable .GT. 50) THEN

WRITE(6,*) 'Simulation assumes maximum of 50 cables'

GO TO 1

ELSE IF (1cable .LT. 1) THEN

WRITE(6,*) 'Improper cable designation'

GO TO 1

END IF

IF (1cable .GT. ncable) ncable=1cable

WRITE(6, '(1h0, ''Enter mark number (0-200): ''')')

READ(5,*) mark

IF (mark .GT. 200 .OR. mark .LT. 0) THEN

WRITE(6,*) 'Improper mark designation'

GO TO 2

END IF

WRITE(6, '(1h0, /, '' Connection type: ''')')

CALL yesno('Station?', reply)

Data Structures:

itopo(mark+1, 1cable) = the station index number
or
- (the repeater index number)
or
zero if no connection is made

Let istat = the station index number:

stat((istat*3)-2) = 1cable, the station's cable

stat((istat*3)-1) = mark, the station's location on 1cable

stat(istat*3) = the length of cable between the station
and its MRU

Let irept = the repeater index number:

rept((irept*8)-7) = 1cable1, one of the repeater's
linked coax cables

rept((irept*8)-6) = mark1, the repeater's connection
point on 1cable1

rept((irept*8)-5) = the length of cable between the
repeater and coax cable 1cable1

rept((irept*8)-4) = icable2, the other of the repeater's
linked coax cables

rept((irept*8)-3) = mark2, the repeater's connection
point on icable2

rept((irept*8)-2) = the length of cable between the
repeater and coax cable icable2

rept((irept*8)-1) = the length of any internal point-to-point
link; also acts as a flag indicating
whether loop testing of this repeater
has been performed (in that case, value
equals length of point-to-point link
minus 1100.)

rept(irept*8) = the total propagation delay between
icable1 and icable2

IF (reply .EQ. 'Y') THEN

IF (itopo(mark+1, icable) .EQ. 0) THEN
nstat=nstat+1

WRITE(6, '(1H0, "Enter binary station address",
(in quotes):'))

READ(5, *) address(nstat)

k=1

DO WHILE (address(nstat)(k:k) .NE. ' ' .AND. k .LE. 16)

IF (address(nstat)(k:k) .NE. '1' .AND.
address(nstat)(k:k) .NE. '0') THEN

WRITE(6, '(1H0, "Address must be binary"))
GO TO 7

END IF

k=k+1

END DO

k=k-1

IF (k .NE. nbits) THEN

WRITE(6, '(1H0, "Station ", i3)') nstat

WRITE(6, '(1H0, "Incorrect number of",
bits in address'))

STOP

END IF

DO i=1, nstat

DO j=1, nstat

IF (i .NE. j) THEN

IF (address(i)(1:nbits) .EQ. address(j)(1:nbits))
THEN

WRITE(6, '(1H0, "Stations ", i3, " and ", j3,
have been assigned the same address')) i, j

STOP

END IF

END IF

END DO

END DO

```

1      itopo(mark+1,icable)=nstat
      stat((nstat*3)-2)=icable
      stat((nstat*3)-1)=mark
      stat(nstat*3)=ran(jseed)*50.e0
      WRITE(6, '(1h0, "Station: ", //, 6x, "mark: ", //, 13, //, 6x,
1      "cable: ", //, 13)') mark, icable
      ELSE
1      WRITE(6, '(1h0, "Indicated cable position has already",
      " been assigned"')')
      WRITE(6, *) 'icable, mark: ', icable, mark
      STOP
      END IF
      ELSE
      IF (itopo(mark+1,icable) .EQ. 0) THEN
          nrept=nrept+1
          itopo(mark+1,icable)=-nrept
          rept((nrept*8)-7)=icable
          rept((nrept*8)-6)=mark
          rept((nrept*8)-5)=ran(jseed)*50.e0

          WRITE(6, '(1h0, "Repeater: ", //, 6x, "mark: ", //, 13, //, 6x,
1          "cable ", //, 13,
1          ///, " Repeater links to (mark, cable) ", //, " Enter",
1          " cable number: ")') mark, icable
          READ(5, *) icable2

          IF (icable2 .GT. 50) THEN
              WRITE(6, *) 'Simulation assumes maximum of 50 cables'
              GO TO 3
          ELSE IF (icable2 .LT. 1) THEN
              WRITE(6, *) 'Improper cable designation'
              GO TO 3
          ELSE IF (icable2 .EQ. icable) THEN
              WRITE(6, '(1h0, "Topology error: repeater connects",
1              " to same cable at both ends"')')
              GO TO 3
          END IF

          IF (icable2 .EQ. icable) THEN
              WRITE(6, '(1h0, "Topology error: repeater connects",
1              " to same cable at both ends"')')
              WRITE(6, *) 'icable: ', icable
              STOP
          END IF

          IF (icable2 .GT. ncable) ncable=icable2

          rept((nrept*8)-4)=icable2

          WRITE(6, '(1h0, "Enter mark number (0-200): ")')
          READ(5, *) mark2

          IF (mark2 .GT. 200 .OR. mark2 .LT. 0) THEN
              WRITE(6, *) 'Improper mark designation'
              GO TO 4
          END IF

```

```

rept((nrept*8)-3)=mark2

IF (itopo(mark2+1,icable2) EQ 0) THEN
    ptplen=0.
    CALL yesno('Does the repeater have an internal point-to-
1point link?',reply1)
    IF (reply1 EQ 'Y') THEN
        WRITE(6, '(1h0, "Enter length of point-to-point",
1          " link (in metres): "')
        READ(5, *) ptplen
        IF (ptplen .GT. 1000.) THEN
            WRITE(6, '(1h0, "Point-to-point link may not",
1          " exceed 1000 metres"')
            GO TO 5
        ELSE IF (ptplen .LT. 0.) THEN
            WRITE(6, *) 'Improper Link length'
            GO TO 5
        END IF
    END IF

    rept((nrept*8)-2)=ran(jseed)*50.e0
    rept((nrept*8)-1)=ptplen
    rept(nrept*8)=2./rate + (ptplen+rept((nrept*8)-2)
1      + rept((nrept*8)-5))/(.65e0*3.e0)

    itopo(mark2+1,icable2)=-nrept

ELSE
    WRITE(6, '(1h0, "Indicated cable position has already",
1      " been assigned"')
    WRITE(6, *) 'icable2,mark2:', icable2,mark2
    STOP
END IF

ELSE
    WRITE(6, '(1h0, "Indicated cable position has already",
1      " been assigned"')
    WRITE(6, *) 'icable,mark:', icable,mark
    STOP
END IF

END IF

IF (nstat EQ 600 .OR. nstat EQ maxnum) THEN
    WRITE(6, '(1h0, "Limit of ",13," stations has",
1      " been reached"') nstat
    WRITE(6, '(1h0, "Press RETURN",
1      " for more"')
    READ(5, '(a)') line
    more='N'
ELSE
    CALL yesno('Are there any more connections?', more)

    WRITE(6, *) 'icable,mark:', icable,mark

END IF

END DO

WRITE(6, '(1h1, "Number of stations: ",14)') nstat

```

```

DO i=1,nstat
  icable=stat((i*3)-2)
  mark=stat((i*3)-1)

  WRITE(6, '(1h0, //, 5x, 'Station ', i4, //, 9x,
1      'Address: ', a, //, 9x, 'Mark: ', i3, //, 9x,
1      'Cable: ', i3, //, 9x, 'MAU cable length = ', f4.1,
1      ' metres', //) i, address(i), mark, icable, stat(i*3)

  WRITE(6, '(1h0, //, 57('' ''), 'Press RETURN for more'')')
  READ(6, '(a)') line

```

END DO

```

WRITE(6, '(1h1, 'Number of repeaters: ', i4)') nrept

```

```

DO i=1,nrept
  icable1=rept((i*8)-7)
  mark1=rept((i*8)-6)
  icable2=rept((i*8)-4)
  mark2=rept((i*8)-3)

  WRITE(6, '(1h0, //, 5x, 'Repeater ', i4, //, 9x,
1      'links cable ', i3, ' to cable ', i3, //, 9x,
1      'Mark: ', i3, //, 9x, 'Cable: ', i3, //, 9x,
1      'MAU cable length = ', f4.1, ' metres', //, 9x,
1      'Mark: ', i3, //, 9x, 'Cable: ', i3, //, 9x,
1      'MAU cable length = ', f4.1, ' metres', //, 9x,
1      'Length of internal point-to-point link = ', f6.1,
1      ' metres', //, 9x,
1      'Total propagation delay between cables = ',
1      1pe10.3, //)')
1      i, icable1, icable2, mark1, icable1, rept((i*8)-5), mark2,
1      icable2, rept((i*8)-2), rept((i*8)-1), rept(i*8)

```

```

WRITE(6, '(1h0, 57('' ''), 'Press RETURN for more'')')
READ(6, '(a)') line

```

END DO

```

CALL yesno('Is this the desired topology?', reply)
IF(reply .EQ. 'N') STOP

```

Check input topology:

```

DO icable=1, ncable
  nmau=0
  DO mark1=1, 201
    IF (itopo(mark1, icable) .NE. 0) nmau=nmau+1
  END DO

  IF (nmau .GT. 100) THEN
    WRITE(6, '(1h0, 'More than 100 connections to cable ', i3)')
1      icable
    STOP
  ELSE IF (nmau .EQ. 0) THEN
    WRITE(6, '(1h0, 'No connections to cable ', i3)') icable

```

STOP
END IF

END DO

tptplen=0.
DO i=1,nrept
tptplen = tptplen + rept((i*8)-1)
END DO

IF (tptplen .GT. 1000.) THEN
WRITE(6, '(1h1, "Total length of point-to-point links in",
1 " system exceeds 1000 metres"'))
STOP
END IF

C Check for loops in topology on more than two repeaters between
C stations

icable=1
itest(1)=1
icnt=1
iend=0
igodo=0

DO WHILE (iend .EQ. 0)

DO mark1=1,201

IF (itopo(mark1,icable) LT 0) THEN

irept = -itopo(mark1,icable)

IF (rept((irept*8)-1) GE. 0) THEN
repeater has not yet been checked

IF (rept((irept*8)-7) NE. icable) THEN

icable1 = rept((irept*8)-7)

ELSE

icable1 = rept((irept*8)-4)

END IF

IF (itest(icable1) .EQ. 0) THEN

itest(icable1) = icnt+1

ELSE

WRITE(6,*) 'Topology error: loop in system'

STOP

END IF

C Set flag on this repeater by subtracting 1100. from the
C point-to-point link length

rept((irept*8)-1) = rept((irept*8)-1) - 1100.

END IF

END IF

END DO

```

IF (icnt EQ 1) icnt=2

igodo=0
DO WHILE (igodo EQ 0)
  icable=icable+1

  IF (icable GT ncable) THEN
    icnt=icnt+1
    icable=1
    iend=1
    igodo=1

    DO i=1,ncable

      WRITE(6,*) i, itest(1)

      IF (itest(1) EQ 0) THEN
        iend=0
        igodo=0
      END IF
    END DO

    ELSE IF (itest(icable) EQ icnt) THEN
      igodo=1
    END IF

  END DO

END DO

icnt2=0
icnt3=0

DO i=1,ncable
  IF (itest(i) EQ 2) THEN
    icnt2=icnt2+1
  ELSE IF (itest(i) EQ 3) THEN
    icnt3=icnt3+1
  ELSE IF (itest(i) GT 3) THEN
    WRITE(6, '(1h0, "More than two repeaters exist between",
1 " two or more stations" )')
    STOP
  END IF
END DO

IF (icnt2 GT 1 .AND. icnt3 GE 1) THEN
  WRITE(6, '(1h0, "More than two repeaters exist between",
1 " two or more stations" )')
  STOP
END IF

```

Restoration of point-to-point line length

```

DO i=1,nrept
  rept((i*8)-1) = rept((i*8)-1) + 1100.
END DO

```

C Evaluation of station-to-station delays

DO 1stat=1, nstat

icable = stat((1stat*3)-2)

DO mark1=1, 201

IF (itopo(mark1, icable) GT 1stat) THEN

1 tdel(1stat, itopo(mark1, icable)) = (stat(1stat*3) +
1 stat(3*itopo(mark1, icable)))/(65*3. e8)
1 + (abs(mark1-stat((1stat*3)-1))*2. 5)/(77*3. e8)

1 tdel(itopo(mark1, icable), 1stat) =
1 tdel(1stat, itopo(mark1, icable))

END IF

END DO

DO mark1=1, 201

IF (itopo(mark1, icable) LT 0) THEN

irept = -itopo(mark1, icable)

IF (rept((irept*8)-7) NE. icable) THEN

icable2 = rept((irept*8)-7)

markr = rept((irept*8)-6)

ELSE

icable2 = rept((irept*8)-4)

markr = rept((irept*8)-3)

END IF

DO mark2=1, 201

IF (itopo(mark2, icable2) GT 1stat) THEN

1 tdel(1stat, itopo(mark2, icable2)) =
1 (stat(1stat*3) + stat(3*itopo(mark2, icable2)))/
1 (65*3. e8) +
1 rept(irept*8) +
1 (abs(mark1-stat((1stat*3)-1)) +
1 abs(markr-mark2))*2. 5)/(77*3. e8)

1 tdel(itopo(mark2, icable2), 1stat) =
1 tdel(1stat, itopo(mark2, icable2))

END IF

END DO

DO mark2=1, 201

```

      IF (1topo(mark2, 1cable2) LT 0) THEN
        IF (1topo(mark2, 1cable2) NE 1topo(mark1, 1cable))
1                                     THEN
          1rept3 = -1topo(mark2, 1cable2)

          IF (rept((1rept2*8)-7) NE 1cable2) THEN
            1cable3 = rept((1rept2*8)-7)
            mark2 = rept((1rept2*8)-6)
          ELSE
            1cable3 = rept((1rept2*8)-4)
            mark2 = rept((1rept2*8)-3)
          END IF

          DO mark3=1, 201

            IF (1topo(mark3, 1cable3) GT 1stat) THEN

              tdel(1stat, 1topo(mark3, 1cable3)) =
1              (stat(1stat*3) + stat(3*1topo(mark3, 1cable3)))
1              ( 65*3 e8) +
1              rept(1rept*8) + rept(1rept2*8) +
1              (abs(mark1-stat(1stat*3)-1)) +
1              abs(mark1-mark2) + abs(mark2-mark3))*2.5)/
1              ( 77*3 e8)

              tdel(1topo(mark3, 1cable3), 1stat) =
1              tdel(1stat, 1topo(mark3, 1cable3))

            END IF

          END DO

        END IF

      END IF

    END DO

  END DO

CLOSE(UNIT=5)

OPEN (UNIT=2, FILE='user [alan]matstor1.dat', STATUS='new',
1      FORM='unformatted', ERR=999)
WRITE(2, '(nstat, (tdel(1, j), j=1, nstat), 1=1, nstat),
1      (address(j), j=1, nstat))' )
CLOSE(UNIT=2)

OPEN (UNIT=3, FILE='user [alan]topo lis', STATUS='new', ERR=999)
WRITE(3, '(1h1, ///, ' Number of stations = ', i3)' ) nstat

```

```

ifirst=0
DO icable=1,nrpt
  IF (1+topo(mark1,icable) GT 0) THEN

    mark = mark1 - 1
    istat = 1+topo(mark1,icable)
    IF (ifirst EQ 0) THEN
      ifirst=1
      WRITE(3, '(//,5x, " Cable ",i1,4x, " Mark: ",
1          13,4x, " Address ",a,4x,
1          " MAU cable length ",f4,1, " metres"))'
1          icable,mark,address(istat),stat(istat*3)
    ELSE
      WRITE(3, '(14,i1,4x,7x,13,4x,10x,a,4x,19x,f4,1))'
1          icable,mark,address(istat),(1+nbits),
1          stat(istat*3)
    END IF
  END IF
END DO
END DO

IF (nrpt GT 0) THEN
  WRITE(3, 1n1) Number of repeaters = '(,i1)' nrpt
  DO i=1,nrpt
    icable1=rept((i*8)-7)
    mark1=rept((i*8)-6)
    icable2=rept((i*8)-4)
    mark2=rept((i*8)-3)

    WRITE(3, '(//,5x, " Repeater ",i1, " links cable ",
1          11, " to cable ",i1, //,9x, " Cable ",
1          11,4x, " Mark ",i1,13,4x, " MAU cable length ",
1          2,f4,1, " metres", //,9x, " Cable ",
1          11,4x, " Mark ",i1,13,4x, " MAU cable length. ",
1          2,f4,1, " metres"))'
1          icable1,icable2,icable1,mark1,rept((i*8)-5),
1          icable2,mark2,rept((i*8)-2)

    IF (rept((i*8)-1) GT 0) THEN
      WRITE(3, '(//,9x, " Length of internal point-to-point",
1          1          link = ',f6,1, " metres"))'
1          rept((i*8)-1)
    END IF
  END DO
END IF

CLOSE(UNIT=3)

STOP

999 WRITE(3, ' Error in OPENing data file')
STOP

END

```

