INFORMATION TO USERS

This reproduction was made from a copy of a manuscript sent to us for publication and microfilming. While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. Pages in any manuscript may have indistinct print. In all cases the best available copy has been filmed.

The following explanation of techniques is provided to help clarify notations which may appear on this reproduction.

1. Manuscripts may not always be complete. When it is not possible to obtain missing pages, a note appears to indicate this.

2. When copyrighted materials are removed from the manuscript, a note appears to indicate this.

3. Oversize materials (maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or in black and white paper format.*

4. Most photographs reproduce acceptably on positive microfilm or microfiche but lack clarity on xerographic copies made from the microfilm. For an additional charge, all photographs are available in black and white standard 35mm slide format.*

***For more information about black and white slides or enlarged paper reproductions, please contact the Dissertations Customer Services Department.**

U·M·I Dissertation Information Service

University Microfilms International
A Bell & Howell Information Company
300 N. Zeeb Road, Ann Arbor, Michigan 48106

Chiou, Ian Yiing-Shyang

DESIGN AND ANALYSIS OF A VOICE/DATA INTERNET TRANSPORT SYSTEM

The Ohio State University PH.D. 1986

DESIGN AND ANALYSIS OF A VOICE/DATA INTERNET TRANSPORT SYSTEM

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Ian Yiing-Shyang Chiou, B.S.E.E., M.S.

* * * * *

The Ohio State University

1986

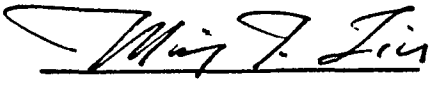
Reading Committee:

Prof. Ming T. Liu

Prof. Jerome Rothstein

Prof. Neelam Soundararajan

Approved By



Adviser
Department of Computer
and Information Science

Dedicated To My Parents

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my advisor Prof. Ming T. Liu for his continuous guidance, advice and encouragement during the course of my research. His dedication, scholarship and wisdom have always made me feel humble in the pursuit of my future career.

I am also grateful to Prof. Jerome Rothstein and Prof. Neelam Soundararajan for serving on my reading committee and for their many helpful suggestions and comments on my research.

Thanks also go to the members of our Advanced Communication Systems research group. In particular, I would like to single out Nien-Chen Liu for his many helpful discussions and friendship.

I have benefitted from interactions with my colleagues at the IRCC/CIS Computing Laboratory. I am especially grateful to Delbert Waggoner, Dennis Slaggy and Frank Northrup for their support, consideration and friendship. It has been a pleasure to work with them.

Finally, no dissertation could ever be completed without the support of one's loved ones. Above all, I thank my wife, Mei-Mei, and my son, Albert, for their patience, understanding and endurance during my tenure at OSU. They have always been the source of the inspiration that moves me forward.

The work reported herein was supported in part by U. S. Army CECOM, Ft. Monmouth, NJ, under Contract No. DAAB07-83-K-K542. The views, opinions and/or findings contained in this paper are those of the author and should not be construed as an official Department of the Army position, policy or decision.

VITA

Sept. 14, 1954	Born, Tainan, Taiwan, Republic of China
1977	B.S., Department of Electrical Engineering National Taiwan University, Taipei, Taiwan, Republic of China
1979	Graduate Teaching Assistant, Dept. of Computer and Information Science The Ohio State University, Columbus, Ohio
1980	Graduate Research Associate, Dept. of Physiology The Ohio State University, Columbus, Ohio
1981	M.S., Dept. of Computer and Information Science The Ohio State University, Columbus, Ohio
1981 – 1983	System Programmer, AMF Logic Sciences, Inc., Houston, Texas & Chemical Abstracts Service, Columbus, Ohio
1983 – Present	Graduate Research Associate, IRCC/CIS Computing Laboratory The Ohio State University, Columbus, Ohio

PUBLICATIONS

- "CAMPUSNET: A Gateway-Network Approach To Interconnecting A Campus-Wide Internet,"
  in *Proc. of IEEE INFOCOM85*, pp. 168-177, Washington DC, March, 1985. Co-author: Ming T. Liu

- "GATENET: A Voice/Data Internet Transport System,"
  in *Proc. of IEEE INFOCOM86*, pp. 39-46, Miami, Florida, April, 1986. Co-author: Ming T. Liu

FIELDS OF STUDY

- Major Field: Computer Architectures and Organizations
- Minor Field: Theory of Automata and Formal Languages
- Minor Field: Programming Languages

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

Due to rapid advances in computing and communications technology and its potential role in the areas of office automation and distributed processing, computer networking has drawn considerable attention over the past decade. As more and more computer networks are installed in government agencies, universities, factories, corporations and many other areas, the desirability of interconnecting computer networks to obtain more versatile and extensive network services will become even more critical.

While at present the technology of constructing individual networks is well understood, the problems associated with network interconnections are just beginning to receive attention. According to the characteristics of the networks involved, network interconnection can be classified into three types:

1. local area network (LAN) to long haul network communications;

2. long haul network to long haul network communications; and

3. local area network to local area network communications.

For type (1) network communications, CCITT (International Telegraph and Telephone Consultative Committee) recommendation X.25 has been adopted as the international standard interface for individual node to packet-mode Public Data Networks (PDNs). Recommendation X.25 specifies the interface between the customer's equipment (called DTE - data terminal equipment) and the network equipment (called DCE - data circuit-terminating equipment). A virtual circuit approach is implied in Recommendation X.25.

For type (2) network communications, TCP/IP of DoD DARPA is one of the most popular internet protocols accepted in user communities. The IP (Internet Protocol) is a datagram protocol designed to transmit blocks of data from a source to a destination. The IP does not provide a reliable communication facility and thus has no provision for flow control and error control. The TCP (Transmission Control Protocol) is a transport protocol built on top of the IP and uses end-to-end mechanisms (e.g., flow control, positive acknowledgments with timeout and retransmission, sequence numbers, etc.) to ensure reliable sequenced data delivery over a logical connection. For networks which are PDNs, CCITT has adopted Recommendation X.75 to define the interface

between the PDNs. This recommendation is quite similar to Recommendation X.25. The equipment on either side of this interface is called a signaling terminal (STE). The STE-STE interface is much like the DTE-DCE interface and consists of a split gateway with each gateway-half in a physical device controlled by each connecting PDN [24].

For type (3) network communications, depending on the distances between the LANs, a LAN can communicate with another LAN through either a long haul network (which also falls into the type (1) communications class) or some dedicated software/hardware switching device. It is this type of network communications that we are specifically addressing in this dissertation.

1.1 Characteristics of Campus-wide Internet

Local area networks have been a major driving force in office automation from which users at a single location can access a wide variety of computational resources and communication services. However, a LAN normally is restricted to a relatively small area ranging in a distance from several hundred meters to one or two kilometers. The number of nodes that can be attached to a single LAN is also limited to

an upper bound. All of these limitations severely handicap many present-day and potential user applications.

On a typical university campus, many separate buildings are spread over an area too wide for coverage by a single LAN, but the buildings are not so geographically dispersed as to justify using long haul network technology. On such a campus, there usually exist many LANs that are developed over several years using different technologies to cover various buildings. The interconnection of LANs in this environment is called a campus-wide internet, which can be characterized by the following properties [25, 6]:

1. Geographically, it normally spans more than a single building, but administratively, it still belongs to one organization, thereby allowing inter-communications to be achieved over its own privately installed equipment without resort to a public data network. This property is the most essential one because communicating over privately installed equipment can be much more economical than using public data network facilities (The cost difference may be a factor ranging from 10 to 100.).

2. Within this boundary, numerous nodes (e.g., computers, data sources and data sinks) must be interconnected. At the present time, there may be fewer than a hundred nodes to be interconnected, but as hardware costs keep dropping and personal computers and workstations gain more popularity, the number of nodes to be interconnected may soon reach into hundreds or even thousands.

The first property duly justifies the economic incentive to construct a privately owned internet within a campus-wide area, while the second property advocates the necessity to construct such an internet to cope with the ever-growing number of computing devices. It should be noted that the term "campus-wide internet," which will be used frequently in the following discussions, actually stands for any internet with the above properties. Hence, it is equally applicable to any industrial or corporate internets which have similar organizational and geographical characteristics.

Several approaches to constructing a campus-wide internet have been proposed. Saltzer [25] suggests that under a relatively loose administration, with no single user or user group responsible for coordination and maintenance, source routing can be a good choice for a campus-wide internet environment. (Under natural growth conditions, meshed topology seems the most likely internet configuration.) Although this approach is relatively simple and economical, it has several limitations. First, the internet users are held responsible for making route selection, which may become cumbersome and time-consuming as the internet expands. Second, source routing requires a static path to be selected before transmission; hence, a packet would be lost if there is any faulty condition en route. Third, because a longer packet header is needed to specify the transmission path, source routing is not suitable for handling integrated voice/data traffic.

Danthine [6] recommends that an internet be constructed by connecting each LAN to a backbone network through a dedicated gateway. In this approach, a gateway is only used to mediate between a LAN and a backbone network, thus the functionality of the gateway is very simple. The backbone network runs throughout the whole internet area, and broadband technology is most likely to be deployed. While this approach can provide satisfactory integrated voice/data services, it is definitely a very costly solution. Furthermore, since all the internet traffic is routed via the backbone network, transmission control and reliability issues must be properly addressed to prevent internet performance degradation or service interruptions.

1.2 Motivation, Objectives and Contributions of Research

In network communication, the traffic bottleneck cycles between transmission elements and switching elements. In the past several decades, communication links have always been the bottlenecks; thus, a great deal of research has been oriented toward optimizing the utilization of communication links. Many sophisticated adaptive routing schemes have been proposed to maximize the link utilization at the cost of increased processing time. However, due to rapid advances in microelectronics and fiber optic systems in recent years, the situation has

changed dramatically. The scarcity of transmission bandwidths no longer exists; however, the processing elements now become too slow to cope with their tasks. As a consequence of this trend, the installation of very fast and effective gateways at network interconnection points is necessary so that the gateways will not become the internet traffic bottlenecks.

Furthermore, because of advances in voice digitization techniques and the potential economic benefits, interest and demand for integrated voice and data services through the same communication system have grown rapidly. Substantial research and experimental work has demonstrated the feasibility of LANs supporting voice/data integrated services. Most of the work, however, has concentrated on the extent of a LAN boundary. To make the integrated services even more valuable and extensive, it would be highly desirable for users to be able to receive voice/data services beyond a single LAN's boundary.

One of the central issues involved in providing internet voice communication is the fact that voice communication requires stringent transmission delays to facilitate smooth conversations among distant users. Unfortunately, due to the dynamics of packet switched environments, carrying voice traffic across the boundaries of different LANs is likely to be subject to various delay and throughput conditions

en route. Several internetwork protocols have been designed and used to achieve internetwork communications. However, all of these existing protocols are oriented toward data communications and as a consequence do not support well for the voice traffic. Thus, a new design of communication protocols becomes necessary in order to meet the requirements of integrated voice and data communication systems.

In light of the above considerations, we feel that the current network interconnection technology is both insufficient and inefficient to serve the existing and future voice/data communication needs. Therefore, the main objective of this research has been to design an internet transport system so that satisfactory voice and data communication services can be achieved in a cost-effective way. As a result, a new network interconnection architecture is proposed, together with its supporting communication protocols. Our approach, instead of following the conventional ad hoc approaches to interconnecting LANs, merges the roles of the backbone network and gateways into a single unit called GATEway-NETwork (GATENET) in order to facilitate the design of a voice/data internet transport system. As the performance evaluations show, GATENET is a feasible and effective approach to meet the future communication needs.

1.3 Organization of Dissertation

This dissertation is concerned with the system design and performance evaluations of a voice/data internet transport system within a campus-wide environment. Each chapter addresses a distinct topic involved in the design of such an internet transport system.

Chapter 2 serves as the basis for this research. It first discusses various characteristics of local area networks, and then briefly describes the various technologies involved in interconnecting local area networks. In particular, several possible methods to implement a gateway are explored. Next, it introduces the ISO seven-layered protocol hierarchies and the standardization efforts in LAN protocols. Also presented are the various features needed to support packetized voice communications.

In Chapter 3, a detailed discussion of and comparisons between the conventional internetworking approach and the GATENET approach are presented. Further, the chapter discusses GATENET's topology as well as its addressing and routing schemes. Three different ways to implement GATENET are also discussed in addition to a summary of the advantages and disadvantages of the GATENET approach.

On the basis of the GATENET hierarchical structure, Chapter 4

defines various communication protocols to support the internet data and voice traffic. Data Transport Protocol (DTP) and Voice Transport Protocol (VTP) are treated separately. Two levels of flow and congestion control mechanisms are also introduced in order to prevent internet performance degradation when the internet becomes overloaded. Also discussed is the enhanced transport layer protocol support to facilitate resolving incompatibilities among connecting LANs.

Chapter 5 is concerned with the reliability aspect of the GATENET design. To avoid the internet partitioning problems under faulty conditions, a "buddy link" scheme is presented as a cost-effective means to improve GATENET reliability, followed by several case studies with respect to various link failure conditions.

Chapter 6 discusses the performance evaluations of GATENET. The GATESIM network communication simulator together with some of the assumptions and parameters are briefly described. Next, a thorough simulation study with respect to GATENET delay and throughput characteristics and the impact of flow and congestion control are conducted and discussed.

Chapter 7 summarizes the results of this research, and directions for future research are also suggested.

CHAPTER II
BACKGROUND

This chapter discusses various subjects related to network interconnections and packetized voice communication. These subjects include LAN architectures, internetworking techniques, communication protocols and voice communication aspects.

Although many different technologies have been used in LAN implementations, LANs in general can be classified according to three distinct features: topologies, transmission media and transmission control mechanisms. Section 2.1 gives a brief discussion of each of these aspects. Depending on the characteristics of individual LANs, network interconnections can be implemented through repeaters, bridges or gateways. Section 2.2 describes the differences among these various techniques. Also discussed are four distinct ways to implement a gateway. Since protocols are the kernel of any communication systems and due to the complexity of modern communication requirements, the

layering approach is utilized in order to decompose complex communication systems into a number of more manageable layer protocols. Section 2.3 provides a brief overview of the ISO seven-layered protocol hierarchies. It also summarizes the activities related to the standardization process of LAN protocols. Several components are needed to realize digital voice communication over packet switched networks. Section 2.4 contains discussions for each of those components. Finally, a summary is given in Section 2.5.

2.1 Basic Architectures of Local Area Networks

Due to the deployment of different hardware technologies, LANs and conventional long haul networks show many different characteristics with respect to topological layout, transmission bandwidth and network protocols. In particular, since long haul networks usually have a wide geographical scope and since the processing time of the switching processors is fast enough when compared to the traverse time across the communication subnetwork, long haul networks tend to implement complex protocols (which thus result in more processing time) to optimize the link utilization. In contrast, local area networks, due to their high channel bandwidths, tend to implement simple protocols to minimize the processing time.

Although a number of definitions of LANs have been proposed in the field, because of the divergent applications and design technologies, it is difficult to determine a single definition that can be universally accepted. As discussed in [26], however, a LAN generally has the following features:

- geographically confined to a distance of up to a few miles;
- multiple services often possible on a single LAN, including voice, data and video;
- high-speed transmission media normally in the range of 50 Kb/s to 150 Mb/s;
- some form of topological layout and access control;
- owned by a single organization.

Although most of the existing LANs are primarily used for data communications, recent studies have demonstrated the feasibility of using current technology to support a mixture of voice and data traffic. In view of the recent advances made in microelectronics and fiber optics, one can safely claim that in the near future LANs will become the core of office automation, and integrated information services, including data, voice, video, graphic and facsimile, will then all be available on a single network system.

In general, the basic architectures of LANs can be classified

according to their topologies, transmission media and transmission control mechanisms. These areas are briefly discussed below.

2.1.1 Topology

Topology means the interconnection strategy of a network in which a node can communicate with other nodes of the same network. Generally, a LAN can be constructed as one of the following topologies [27]:

- Star topology: one node forms the center, with a separate link to each of the remaining nodes. All traffic is directed to and from the center node.

- Ring topology: nodes are interconnected into a closed loop, within which each node is connected to exactly two adjacent nodes.

- Bus topology: nodes are connected to a common channel, through which all the nodes transmit and receive messages.

- Meshed topology: nodes are connected in an arbitrary pattern, and there may be multiple paths between each pair of nodes.

- Hierarchical topology: nodes are connected as a tree structure; each node, except the root node, has a unique parent and possibly some children.

All of these topologies have been used by various networks, and each has its pros and cons, depending on the particular applications and

environments. Ring and bus topologies are most popular in commercial systems, and many analytical and simulation performance studies over a broad spectrum of parameters can be found in the literature.

2.1.2 Transmission Media

Transmission media are the physical connections between the source and the destination nodes. These may differ in the characteristics of bandwidth, geographical dispersion, connectivity, immunity to noise and cost. The transmission media often used in local area networks include the following [28, 21]:

- Twisted pairs: typically used for low speed transmission; but, with properly spaced repeaters, data rates of up to 10 Mbps are achievable. Twisted pairs have long been used as a relatively inexpensive means of data communication and are most cost-effective when used in low traffic and single building environments.

- Coaxial cables: can provide higher throughput and support a large number of devices. Two transmission methods, baseband and broadband, can be employed on a coaxial cable. Baseband coaxial cables can provide data rates from one to ten Mbps and are generally limited to a single building. Because of their simplicity and low interfacing cost, baseband cables have been widely used in many LAN implementations. The bandwidth of broadband cables is somewhere between that of baseband and fiber optic cables and can be in the

order of up to 300 Mbps. Unlike the baseband's single data path, broadband cables can have many data paths supporting simultaneous transmissions of data, voice and video. Broadband technology is more expensive when compared to baseband technology, but due to its wider bandwidth and greater geographical coverage, several commercial LANs using off-the-shelf CATV (Community Antenna Television) hardware have recently begun to appear on the market.

- Fiber optic cables: a very attractive medium for future communication systems. These cables can run for several miles without a repeater and provide extremely high data rates of up to a few Giga bits per second. However, because of the technical difficulties and high costs involved in cable tapping and signal extracting, the present use of fiber optics is limited to point-to-point communication, while multidrop mode communication still needs further exploration.

2.1.3 Transmission Control Mechanisms

Quite a few transmission control mechanisms have been proposed for use in building local area networks. Most of these mechanisms can be categorized in one of the following classes [20, 19]:

- Fixed Assignment: the channel bandwidth is allocated to each node of the network according to a predefined pattern. Two well-known examples are time-domain multiplexing (TDM) and frequency-domain multiplexing (FDM). Both TDM and FDM work well under heavy buffered traffic when the number of nodes is small and static. But if the traffic is bursty and

the number of nodes is large, much of the bandwidth may be wasted.

- Random assignment: no strict rule regulates the utilization of the channel bandwidth; thus, nodes on the same network need to compete with one another for channel access. As a result, collisions are unavoidable, and some traffic control strategies are required in order to avoid or recover from collisions. Examples include ALOHA, CSMA and CSMA/CD. These schemes perform well when traffic is light, but their performance declines rapidly under heavy traffic loads which greatly increase the possibility of collision.

- Demand Assignment: channel bandwidth is allocated upon demand; hence, there is no bandwidth waste due to collisions or unnecessary allocation to idle nodes. Examples are token-ring, token-bus and register-insertion. These schemes perform well under heavy traffic conditions, and their performance is predictable, although it suffers from some overhead because of bandwidth reservation when traffic is light.

2.2 Network Interconnections

In a truly distributed computing environment, user processes needing to communicate should be able to do so whether they are in the same network or not. Thus, in order to accommodate the growing demands for geographically distributed processing, efficient resource utilization and secure voice and data communications, the issues of network interconnections have in recent years drawn considerable attention and interest.

One of the objectives of designing a network interconnection strategy is to preserve freedom in the design of future computer networks and still be able to interconnect with existing ones. Although many design principles and the experience gained in developing computer networks can be applied, with slight adaptation, to network interconnections, there are still many problems which suggest that different treatments must be devised in order to achieve internetworking. In particular, the lack of a single controlling authority can make the internet design problems more difficult to solve.

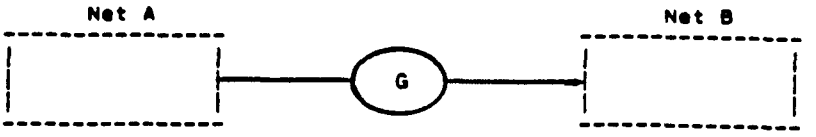
Depending on the characteristics of individual networks (or subnetworks), there are currently three different approaches to interconnecting them as an extended network [12, 3]:

1. Repeaters. These are used to interconnect several cable segments within a LAN using identical software protocols and hardware technologies. Being the simplest among the three approaches, a repeater is usually used to extend the length of the cable, amplifying and transmitting whatever signals it receives (including collisions). No filtering function is performed by a repeater.

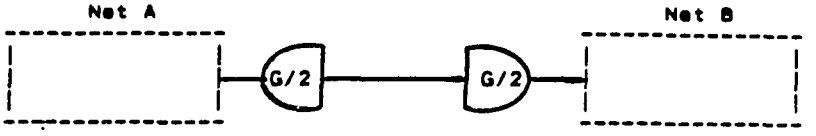
2. Bridges (also called Data Link Relays). These are used to interconnect several networks using different hardware technologies (e.g., network topologies, data transmission rates, etc.) but compatible software protocols (e.g., maximum packet size, addressing scheme, available services, etc.). A bridge performs the filtering function so that only selective packets are forwarded to appropriate networks which it connects. The bridge makes no attempt to modify the contents nor add any additional headers to the packets.

3. Gateways. These are used to interconnect networks using different hardware technologies and incompatible software protocols. Since it is the most general and complicated technology among the three approaches, a gateway may have to address not only the protocol incompatibility problems (by either translating the software protocols from one network into another or by adding an extra internetwork header) but also route selections based upon the network (or internet) layer address supplied by the source node.

As shown in Figure 1, a gateway, depending on the economical and

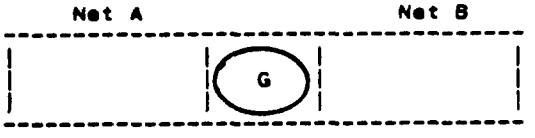


a. single dedicated gateway



b. two dedicated gateway halves

c. two gateway halves residing on separate hosts



d. single shared gateway

Figure 1. Implementation of Gateway

performance requirements, may be implemented in one of the following ways [18]:

1. A gateway is implemented as a physically isolated node, which is equipped with appropriate software protocols and hardware interfaces (Figure 1.a). This approach often incurs higher installation costs; however, due to the continuing downward trend in hardware cost and the increasing demand for voice/data integrated services, its high performance features will outweigh the extra cost. This approach, however, may raise administrative issues when the interconnecting networks belong to different organizations.

2. A gateway is split into two gateway halves, and each gateway half is implemented as a physically isolated node (Figure 1.b). This approach eliminates the administrative problems of the gateway but with the penalty of additional hardware cost.

3. A gateway is split into two gateway halves, and each gateway half resides on a host node of each connecting network (Figure 1.c). This approach provides a tradeoff between the performance and the hardware cost. If the internetwork traffic is not intense and performance requirements are not stringent, this approach can be a cost-effective method for internetworking.

4. A gateway is implemented using a host node shared by connecting networks (Figure 1.d). All internetwork communication software is placed on this single node; hence, any changes on any connecting networks need to be properly monitored and updated. Being the simplest and least

expensive way to achieve internetwork communication, the performance aspects are heavily dependent on the work load of the residing host.

2.3 Protocol Hierarchies and Standardization

A protocol is a set of rules that govern the exchange of information between communicating entities. Due to the inherent complexity in protocol design, the "layering" technique has been widely adopted as an effective means of decomposing a large communication system into a series of layers, each performing a well-defined set of functions to support the communication activities.

As shown in Figure 2, each layer N provides certain services to layers N+1 and higher, shielding the details of how the offered services are actually constructed. Layer N is, in turn, constructed using the services provided through interfaces with layers N-1 and lower. With the services provided by each corresponding Layer N-1, layer N processes on different communication systems can communicate with each other through some communication paths. The rules used by layer N processes in the communication are collectively called (N) **protocol**, and the boundary between layers N and N-1 is called an (N-1) **interface**.

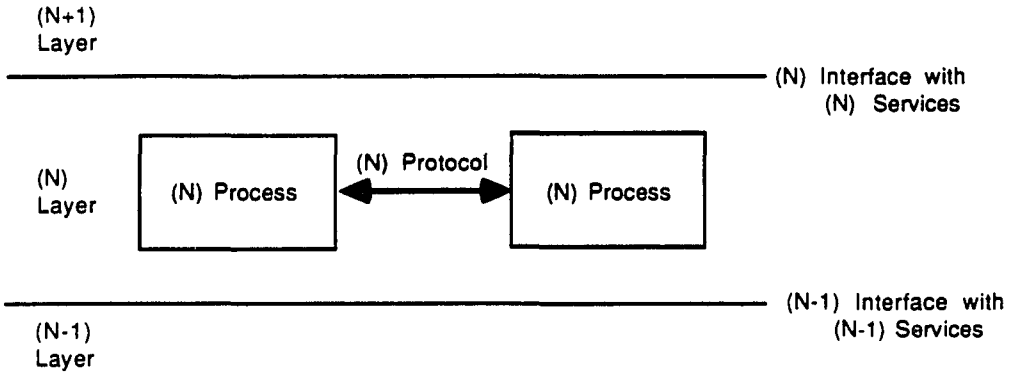


Figure 2. Layers, interfaces and protocols

An (N-1) **interface** defines a set of services which layer N can request from layer N-1.

2.3.1 Protocol Hierarchies

Given a complex communication system without proper coordination and guidelines, each designer will form a layered structure with different layer hierarchies, each with a distinct name and functions. This unfortunate situation has proliferated for decades, thereby causing great difficulty when different systems try to communicate with one another. To overcome this problem, the International Standards Organization (ISO) has defined the Reference Model of Open System Interconnection (OSI) as a conceptual framework, based upon which an end system of one design is able to interconnect and communicate with any other end systems as long as the associated peer protocols follow the same OSI standards.

The OSI Reference Model is divided into seven layers [7]. The lower three layers handle the transmission of data among communicating entities and, thus, are dependent on the hardware technology used for transmission media. The higher three layers provide direct functional support to the end users of the OSI environment and, thus, are independent of the underlying network technology. The Transport Layer is a liaison between the upper and lower layers to ensure that the

services provided by the lower layers fulfill the requirements of the upper layers. The functions of each individual layer are described as follows:

1. Physical Layer: concerned with mechanical, electrical, functional and procedural interfacing so that unstructured bit streams can be transmitted over physical media.

2. Data Link Layer: responsible for framing and possible error detection and error recovery over a point-to-point communication link so that raw bits can appear free of transmission errors to the network layer.

3. Network Layer: responsible for multiplexing, routing, error control and congestion control in order to ensure that data units are correctly routed to their destinations. Network Layer provides the upper layers with independence from concerns about the underlying transmission media and switching technologies used to connect two end systems. It should be noted here that internetwork data transport is part of the function of this layer.

4. Transport Layer: responsible for providing reliable, transparent end-to-end transport services so that session entities are free from the details of how reliable and optimal transfer of data can be achieved. This layer also handles the end-to-end connection establishment and termination.

5. Session Layer: responsible for supporting the interactions between two cooperating presentation entities, including binding and unbinding them into a relationship and synchronization of data operations.

6. Presentation Layer: responsible for the representations of information to facilitate the data exchange between two application entities. In other words, the Presentation Layer deals with the syntax selection and conversion of information so that applications in an OSI environment need only concentrate on the semantic aspects of data operations.

7. Application Layer: the highest layer in the OSI protocol hierarchies. It is responsible for directly providing the distributed information services to the end users of the OSI environment.

It must be noted here that although the OSI Reference Model was originally motivated by and defined for end systems using long haul network technology, it is also applicable to LAN environments. The only exception is that in many LAN environments, due to the inherent "broadcast" capability, route selections are normally not needed, thereby resulting in small or even empty network layer protocols.

2.3.2 Standardization of LAN Protocols

There have been many government agencies (e.g., NBS,[1] DoD), organizations (e.g., ISO, ANSI and ECMA) and companies (e.g., GM,

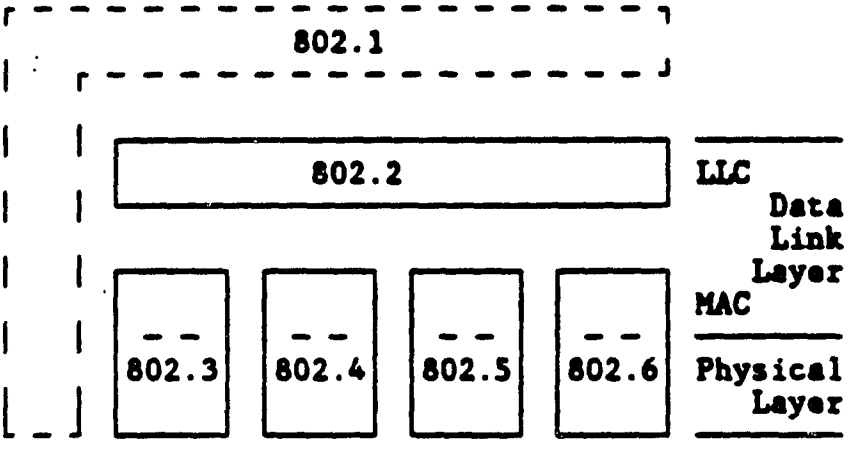
---

[1]NBS: National Bureau of Standards; DoD: Department of Defense; ISO: International Standards Organization; ANSI: American National Standards Institute; ECMA: European Computer Manufacturers Association.

Xerox) involved in standardizing local area network protocols. It is impossible for this dissertation to discuss this standardization process thoroughly; therefore, only the work of the IEEE 802 Project (see Figure 3) will be summarized [9, 23, 29, 1].

An effort to develop local network standards was first initiated by the Institute of Electrical and Electronics Engineers (IEEE) 802 committee in February 1980. Interest in this area quickly became a concern both nationally and internationally. The major goal of the IEEE 802 committee is to deal with protocols for accessing and controlling local network media of different technologies. As a result, its local network reference model corresponds to the two lowest layers of the OSI reference model.

The IEEE physical layer is concerned with bit transmission, device attachment and electrical signaling over various types of local network media. Since all devices in a LAN are connected to a common transmission medium, the Medium Access Control (MAC) sublayer, which constitutes the lower part of the IEEE data link layer, is defined to deal with channel access among various devices connected to the same local network medium. The Logical Link Control (LLC) sublayer, which constitutes the higher part of the IEEE data link layer, is functionally independent of the underlying MAC and physical layers and is



802.1 Higher Layer Interfaces
802.2 Logical Link Control
802.3 CSMA/CD Bus
802.4 Token Bus
802.5 Token Ring
802.6 Metropolitan Area Networks (MAN)

Figure 3. IEEE 802 Project [23]

responsible for establishing, maintaining and terminating a logical connection between communicating devices. The IEEE subcommittee 802.2, which is responsible for LLC standards, has defined three types of services (i.e., connectionless, connection oriented and acknowledged connectionless) for the upper layers.

The IEEE 802 committee, after recognizing the fact that no single standard would be suitable for all LAN applications and traffic patterns, decided to adopt multiple standards. To date, three sets of MAC-physical protocols have been accepted as IEEE standards:

- IEEE 802.3 CSMA/CD
- IEEE 802.4 Token Bus
- IEEE 802.5 Token Ring

The IEEE 802.1 Higher Layer Interface Standard subcommittee, responsible for issuing recommendations and guidelines, is now actively looking into a variety of higher layer design issues such as overall organization of the standards, network management and internetworking. To date, no protocol standards related to internetworking have been brought up. The IEEE 802.6 subcommittee is responsible for Metropolitan Area Networks (MAN) standards, although no standards have yet been issued.

2.4 Packet Voice Communication

Integrated packet switched networks have drawn considerable interest from the research community in recent years because of a number of potential benefits which they offer. These include:

- reduced installation and operation costs through sharing of transmission and switching facilities;
- improved performance by dynamically sharing bandwidths between voice and data traffic and by transmitting voice packets only during talkspurts;
- enhanced network services for users who need access to both data and voice communications;
- capability to support multiplicity of the variable bandwidth services of future communication systems;
- more secure voice communication by applying data security measures developed for data communication.

For voice signals to be carried over a packet switched network, they must first be encoded and packetized at a source voice terminal. Next, an underlying transport system is used to deliver the voice packets within a reasonable time limit to a destination voice terminal which can then depacketize and decode the received voice packets [22]. Some of the elements related to packet voice communication are briefly discussed below.

2.4.1 Functionality of Packet Voice Terminal

As shown in Figure 4, a packet voice terminal (PVT), which serves as the interface between the user and the network, can be conceptually decomposed into four functional modules [31]:

- The voice processor performs conversions between analog/digital signals at speeds ranging from 2 Kbps to 64 Kbps and the marking of each parcel (which normally contains 20-50 ms of speech) as either active or silent.

- The protocol processor is the control center of the PVT, which must generate and interpret packets for call setup and provide buffering and synthesis algorithms to ensure smooth voice playout to the users.

- The network interface processor is responsible for network-dependent hardware and software interfaces to access the packet switched network.

- The telephone instrument serves as the user interface to the PVT. It may be similar to the conventional telephone set but usually provides more signaling capabilities. Computer terminals may also be used to enhance the user interface.

In earlier experimental work on packet voice communication, PVTs were usually implemented on large general-purpose computers. But with the advance of VLSI technology, one can expect that affordable compact microprocessor-based PVTs will soon be on the market.

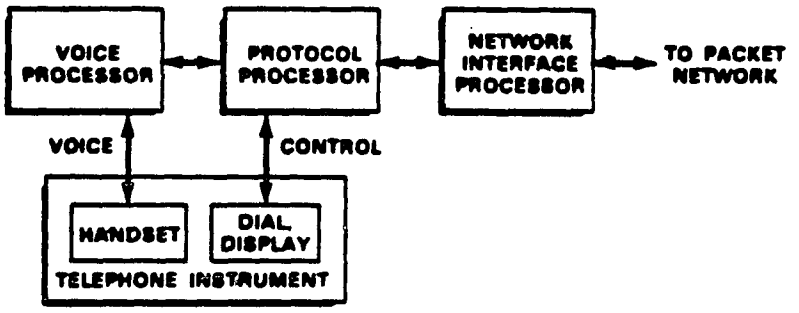


Figure 4. Functionality of packet voice terminal [31]

2.4.2 Vocoding Techniques

Many vocoding (or voice encoding) techniques are available to convert speech to proper digital forms, but they differ greatly in data rates, processing requirements, hardware complexity, and quality of output voice. In general, these vocoding techniques can be classified into time and frequency domain classes [5, 8]. The former class, called waveform coding, is designed to reconstruct voice signals that "look" as much as possible like the original input signals. Examples include PCM (Pulse Code Modulation), DPCM (differential PCM), ADPCM (Adaptive Differential PCM), and CVSD (Continuously Variable Slope Delta Modulation). The latter class, called vocoder, is designed to reconstruct voice signals that "sound" as much as possible like the original input signals. An example is LPC (Linear Predictive coding). (A third class can also be formed by combining the above two techniques.) Waveform coding normally requires data rates in the range of 8 to 64 kb/s, whereas vocoder requires data rates from 1 to 16 kb/s. Generally, the fidelity of the output speech is proportional to the data rates; however, for a given fidelity, the required data rates can be reduced at the cost of more computational processing. The PCM (Pulse Code Modulation) method, which has been widely used in digital telephony, produces good voice quality with high data rates and low processing complexity. However, due to the scarcity of channel bandwidths in long haul

networks, such techniques as CVSD and LPC have been adopted for earlier experimental work on packetized voice communication to make use of their low data rates.

2.4.3 Protocol Functions for Voice Communication

Interactive data communication tends to be bursty in nature, while voice communication tends to be stream-like with a sustained duration for each voice call. For data traffic, transmissions must be very reliable, but occasional variations of transmission delay and throughput can be tolerated. For voice traffic, however, the situation is quite the opposite; transmission delay is very stringent, but a small percentage of packet loss is harmless. As a consequence of these differences, separate communication protocols need to be developed to support voice communication.

In the early seventies, when packet voice communication was first experimented over ARPANET, a separate Network Voice Protocol (NVP) was designed to support the high throughput, low delay requirements of voice communication. Later on, the NVP was revised and enhanced to support internetwork communication, and the protocol functions were separated into two levels. The higher level protocol, called NVP (2nd generation), is concerned with call establishment, packetization and reconstruction of digital voice signals, and dynamic conference control

features. The lower level protocol, called ST (STream protocol), is concerned with internet transport functions for both point-to-point and conference communications.

ST, similar to IP, is an end-to-end internet transport protocol, but it utilizes the virtual circuit approach instead of the datagram approach. Hence for each voice call, a connection setup process must be carried out before a speech conversation begins. NVP (2nd generation) calls on both IP and ST to support voice communication: IP is used primarily for voice control packet delivery and ST is used for voice packet delivery.

2.5 Summary

In this chapter, three major design aspects of local area networks were first presented, followed by a brief description of various internetworking technologies and methods of implementing gateways. Also discussed were ISO seven-layered protocol hierarchies together with the functions of each individual layer. The LAN standardization process was next reported. Finally, several components needed to support packetized voice communication were discussed.

CHAPTER III
GATENET: AN INTERNET TRANSPORT SYSTEM

This chapter is concerned with the architectural aspects of the GATENET internet transport system. In Section 3.1, we describe the scenarios of an internet transmission using the tra[illegible]tional gateway approach and then compare and contrast them with our GATENET approach. A detailed discussion of the GATENET structure with respect to its topology, addressing and routing schemes is next presented in Section 3.2. Advantages and disadvantages of the GATENET approach are then identified in Section 3.3. Finally, Section 3.4 summarizes the chapter.

3.1 Internetworking Approaches

Internetworking generally means interconnecting computer networks, whether they are of similar types or not. As stated earlier, here we are primarily interested in interconnecting LANs within a campus-wide area. (For the sake of clarity, a single dedicated gateway approach is assumed in the following discussions.)

3.1.1 The Conventional Gateway Approach

Figure 5 shows a typical internet interconnected with gateways. Assume that Host 1 in LAN A tries to communicate with Host 2 in LAN D. Normally, an internet packet will be transmitted across the internet as follows:

1. Host 1 first prepares an internet packet (consisting of an internet header, a data packet and possibly some trailer), encapsulates it with a header of LAN A, looks up the routing table, and then forwards it to the proper gateway (in this example, gateway G1) en route to Host 2.

2. Upon reception of the packet, gateway G1 will decapsulate the header of LAN A, examine the internet address (contained in the internet header) and then decide that the next stop will be gateway G2. Since LAN B lies between gateway G1 and gateway G2, gateway G1 will encapsulate the internet packet with a header of LAN B and forward it.

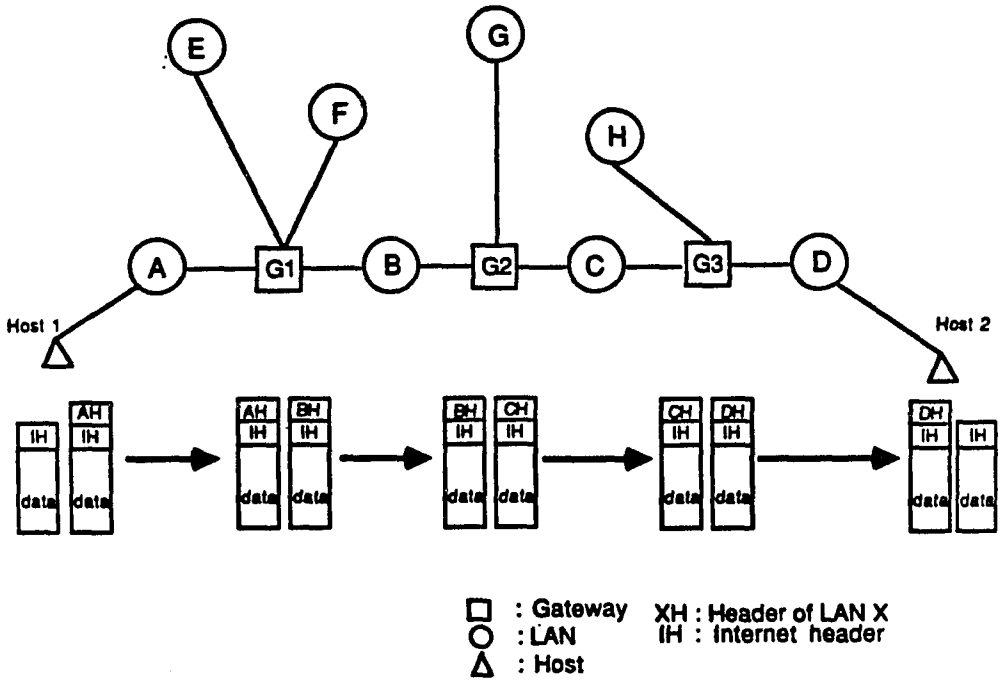


Figure 5. The conventional gateway approach

(In general, this step can be repeated as many times as the number of intermediate nodes between the source host and the destination host.)

3. When the packet finally reaches Host 2, Host 2 will decapsulate the header of LAN D and then process the packet according to the information specified in the internet header.

During an internet transmission, if the size of an internet packet exceeds the maximum packet size of the intermediate network, fragmentation and reassembly processes must be invoked. For each fragmented packet, proper internet headers and trailers also need to be created. In general, two basic approaches can be used to handle such problems:

- Internetwork fragmentation and reassembly: a packet may be fragmented prior to the entry of a LAN (e.g., at a gateway) and reassembled only when it gets to its destination. If the maximum packet sizes are different in LANs en route, the internet packets may have to be broken into even smaller packets several times before they finally reach their destination.

- Intranetwork fragmentation and reassembly: once an internet packet is fragmented before the entry to a LAN, the fragmented pieces will be reassembled immediately after being delivered to another LAN boundary. In a huge internet, as the internet packets travel through various LAN boundaries, fragmentation and reassembly processes may be invoked several times, thus incurring many overheads.

In brief, there are three major functions that a gateway must perform during an internet transmission:

1. The gateways are responsible for encapsulation and decapsulation of LAN headers over the internet packets.

2. The gateways need to look up the routing tables and decide on which path to forward the transit packets.

3. If the size of an arriving internet packet exceeds the maximum allowable packet size, the gateways need to perform fragmentation and subsequent reassembly processes properly.

Another important task of the gateway is monitoring and controlling the internet traffic. However, since it is not relevant to our present discussions, this task will not be elaborated on.

3.1.2 GATENET Approach

Traditionally, a gateway is used to mediate between different networks. However, in our approach, the role of gateways is further enhanced so that a gateway may be used to mediate not only between LANs but also between a LAN and another gateway or even between gateways.

After the interconnections among LANs and gateways are reorganized (see Figures 6 and 7), the internet is logically separated into

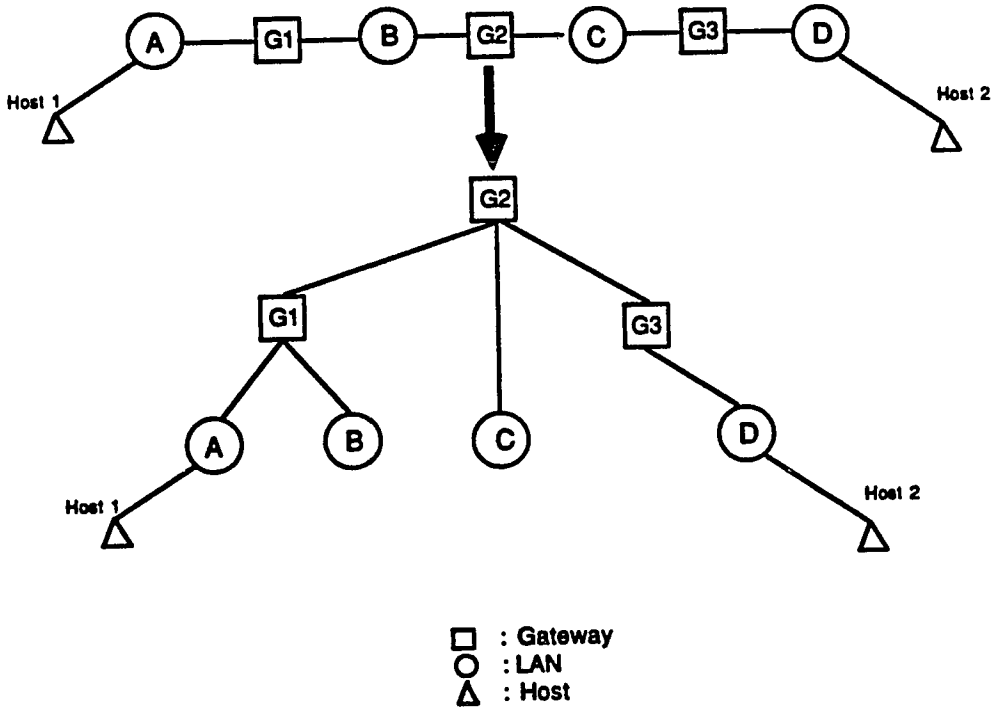


Figure 6. Internet Reorganization

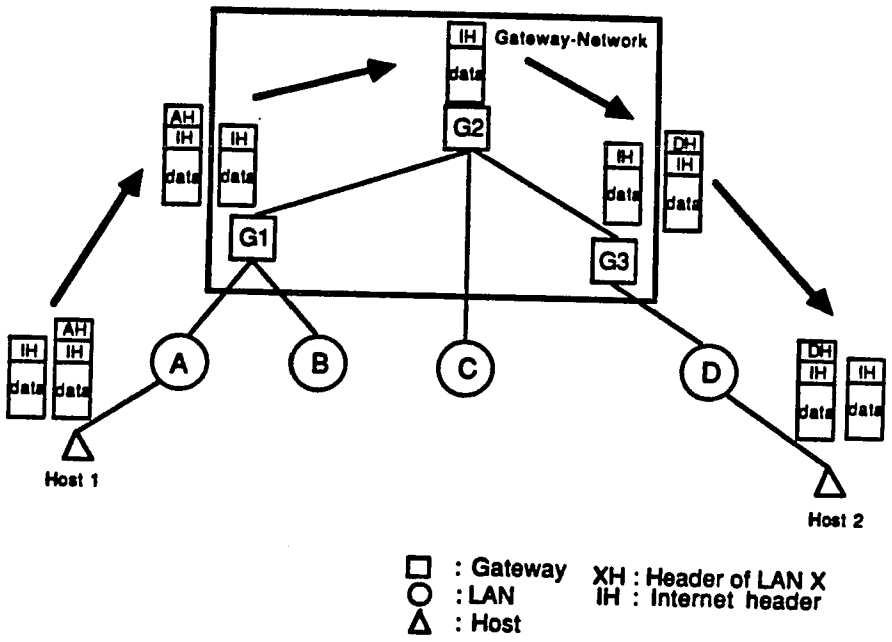


Figure 7. GATENET internetworking approach

two parts: one consists of all LANs and the other consists of all gateways which form a GATEway-NETwork (GATENET). Comparing Figures 5 and 7, one can see that our approach has the following advantages:

1. The internet packet sent from Host 1 to Host 2 essentially bypasses LANs B and C. In other words, although the internet packet goes through the boundaries of LANs B and C via gateways G1 and G2, the local header processing for these LANs is not needed during the internet communication. Consequently, the internet packet transport is essentially independent of the characteristics of the intermediate LANs through which the packet passes, thereby eliminating many processing overheads at the gateways.

2. With the integration of internet gateways into a gateway-network, encapsulation/decapsulation of LAN headers and possible fragmentation/reassembly processing for each internet packet can only occur when the packet is at an entry or an exit gateway.

3. Because of the hierarchical approach of GATENET, each gateway maintains only the cluster routing information for its descendants. Hence, the routing table is small and remains essentially fixed as the size of the internet grows. Further, as a result of the GATENET architecture, simple protocols can be applied at the gateways, thereby reducing the possibility that gateways may become internet traffic bottlenecks (to be further explored in Section 3.2.3).

4. All gateways are essentially interconnected into a gateway-network. As a result, the GATENET structure can facilitate the exercise of internet flow control as well as the dynamic adjustment if there are any changes in the internet topology or traffic characteristics (see Section 4.2.3).

As the number of nodes in the internet grows, our approach will result in fewer processing overheads and a smaller routing table at each gateway when compared with those of the conventional approaches. These advantages will be discussed in more detail in section 3.3.

3.2 GATENET Architectures

In this section, we first show the general topology of GATENET and discuss three different methods to implement it. Based upon the GATENET structure, we then define the addressing and routing schemes to support the internet transport functions. For ease of explanation, the end gateway of LAN X is defined as the gateway by which LAN X is connected to the gateway-network.

3.2.1 Topology of GATENET

GATENET (see Figure 8) is a rooted hierarchically-structured network consisting of gateways and LANs. Logically, all LANs are on the same level (level 1), and all gateways are above the LANs (level 2 and up) forming a gateway-network. Physically, the gateway-network can be built in three different ways:

1. distributed approach: gateways are spread over the campus area and are interconnected through communication links.

2. centralized approach: all gateways are located in one location, and all LANs are connected to the gateway-network by remote access links. This approach is similar to the approach used in the current Private Automated Branch eXchanges (PABXs), where all incoming and outgoing traffic is first routed to a centralized hub.

3. hybrid approach: gateways are clustered into several sub-gateway-networks according to a hierarchical or organizational structure, and only these sub-gateway-networks are interconnected by remote access links.

Each gateway can connect two or more gateways or LANs. Each link is a full-duplex transmission line, one for transmitting packets upward and the other for transmitting packets downward.

The highest level gateway is called the root gateway which is

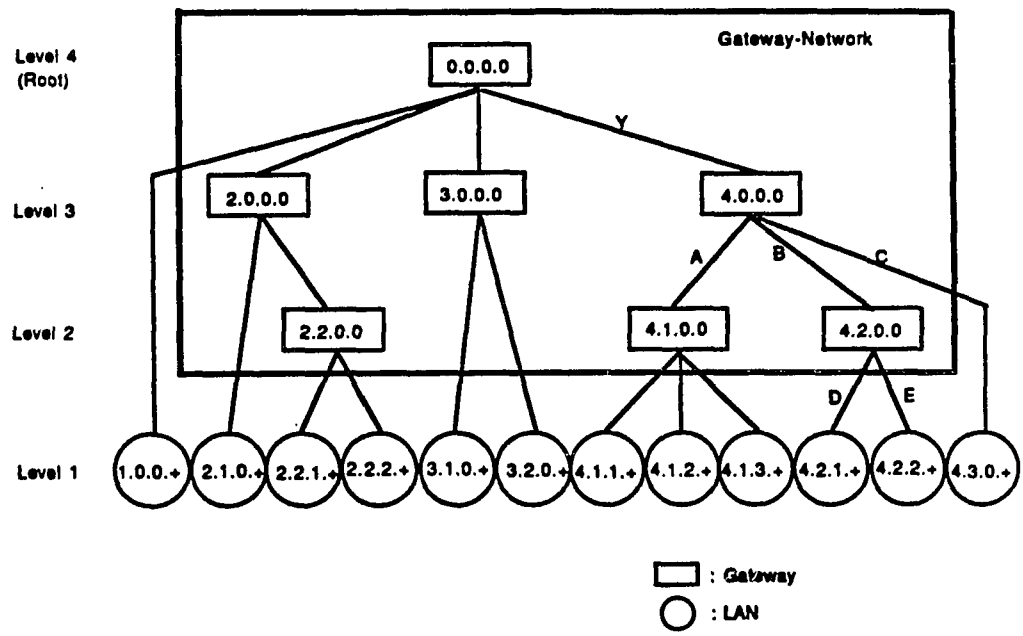


Figure 8. Topology of GATENET

equipped with the capability to communicate with other long haul networks or LANs outside the campus boundary. Since GATENET is designed to fulfill the inter-office data/voice communication needs within a campus or a corporate boundary, special GATENET communication protocols (to be discussed in Chapter 4) are thus specified to provide a simple and efficient way for internetwork communication. Hence, the root gateway may have to perform some translation of the internet headers for packets going outside the internet boundary to ensure compatibility with the internetwork protocols (e.g., TCP/IP) adopted by the majority of the user community. The root gateway is also the Internet Route Server for maintaining and providing internet addressing and routing information.

3.2.2 Addressing

Various studies have shown that in many cases LANs are primarily used for local traffic, with only a very small proportion of the traffic routed outside the LAN boundary. As a result, although the flat global address scheme has been adopted in several major national internets, it is not particularly suitable in the GATENET environment. If the population of the internet is large, the flat global address scheme requires that a huge routing table be maintained at each gateway, and route selections may be time-consuming. Further, the routing table at

each gateway needs to be updated whenever there is any configuration change within the internet, thereby incurring many overheads.

Since an organizational hierarchy often exists within a university campus or any industrial corporation, it seems logical to choose a hierarchical addressing scheme for internet addressing. (In [10, 15], the authors have concluded that, for a huge network, a hierarchical addressing scheme is required to keep the routing capacity of each switching node within bounds.) In the following, a hierarchical addressing algorithm using Kleene's notations is presented, with this algorithm the internet address for each gateway or host in GATENET can be established recursively from its immediate parent.

(Note that here a unit denotes either a gateway or a LAN, and it is assumed that there are four levels in the hierarchy.)

1. (Basis) The root gateway is 0.0.0.0.

2. (Recursion: for each unit in the internet)
   a. Inherit the address from its immediate parent.

   b. Obtain the sequence number of this unit under its immediate parent and substitute the value of the sequence number for the first zero subfield (from left to right).

      Subfield 0 can then be used to identify host addresses

within a LAN. Each LAN is free to choose its own topology and addressing scheme, except that the addresses assigned must be nonzero. (Subfield 0 is zero for all gateways.)

It must be noted that the height of the hierarchy in a campus-wide environment is expected to remain essentially fixed even when the internet continues to grow. Consequently, the number of subfields required for specifying an internet address does not need to be changed in most cases.

As an example, the internet address for Host H with local address A (where A is nonzero) within Unit U (in this case the unit is a LAN) can be generated as follows (see Figure 9):

1. UNIT U inherits address 4.1.0.0 from its immediate parent node.

2. UNIT U's address now becomes 4.1.2.0 since unit U is the second son (from left to right) of its parent node.

3. Host H address becomes 4.1.2.A since its local address is A and it is within UNIT U.

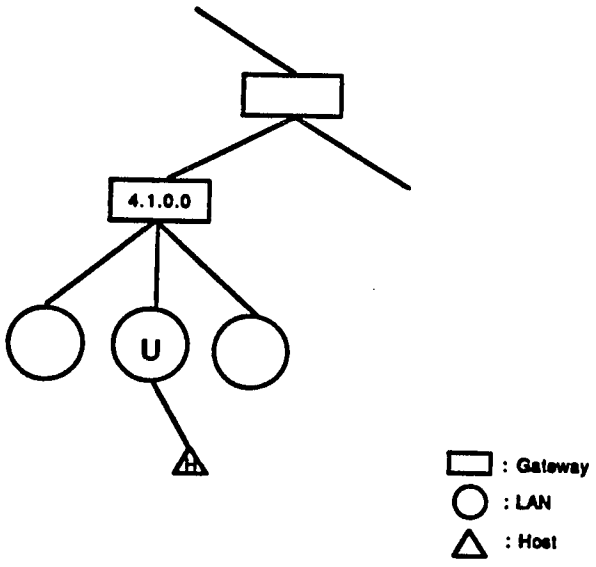


Figure 9. Example of GATENET addressing scheme

3.2.3 Routing

Because of GATENET's hierarchical structure and hierarchical addressing scheme, all addresses corresponding to the descendants of a gateway can be described in just a few patterns. Accordingly, each gateway maintains only such cluster routing information in its routing table (see Figure 10). The routing table has an entry associated with each outgoing link from the gateway. Each entry consists of an address pattern and its associated link. Since the number of entries is the same as that of outgoing links from the gateway, the size of the routing table is small and remains essentially fixed even as the internet grows.

If a node is created or removed, only the routing table of its parent node needs to be updated. No broadcast is needed in this case. The parent node will be responsible for reporting this change to the Internet Route Server, if such a server exists.

When an internet packet arrives, the gateway will first try to look for a matching entry in the routing table (from the top down), according to the destination address in the internet header. To be specific,

1. if the packet belongs to one of the gateway's descendants, the packet will be routed downward to the next stop via the corresponding link;

(i) For gateway 4.0.0.0

4.1.*.*	A
4.2.*.*	B
4.3.0.+	C
..*.*	Y

(ii) For gateway 4.2.0.0

4.2.1.+	D
4.2.2.+	E
..*.*	B

\+ : Nonzero integer
* : Any integer

Figure 10. Routing tables of GATENET

2. otherwise, the packet will simply be routed upward to its immediate parent. Undeliverable packets at the root gateway will be discarded.

Therefore, during an internet transmission, the internet packet will, in general, first climb zero or more levels up the gateway-network until it arrives at the youngest ancestor of the destination node. The internet packet will then descend zero or more levels from this ancestor to reach its destination node.

It is possible that an internet packet might be hopping in a closed loop (due to transmission errors or incorrect specifications of the destination address) during an internet transmission. To avoid such a problem, a special subfield MARK (part of the transport control option) is specified in the internet header. Specifically, for each packet sent downward from a level n gateway (i.e., to level n-1 or below), this subfield will be marked as n. Thus, whenever a level n gateway receives from its son a packet whose MARK subfield is already marked n, the gateway can discard the packet accordingly to prevent such a packet from circulating in a loop.

3.3 Features of GATENET

To summarize, the features of GATENET that distinguish it from the conventional internet approaches are as follows.

3.3.1 Advantages

1. Since the gateway-network is responsible for all the inter-LAN routing, the encapsulation/decapsulation of LAN headers for each internet packet may only occur when the packet is at an end gateway. For traffic between gateways, the internet packets are the "universal" packets, which can travel in the gateway-network without incurring extra local header processing.

2. When fragmentation/reassembly services are needed by the internet, only the end gateways are required to perform such services (In GATENET, all the packets have the same maximum packet size.). Consequently, such services are essentially independent of the characteristics of the intermediate nodes through which the packet passes.

3. Since only cluster routing information is maintained at each gateway, the routing table is small and remains essentially fixed as the size of the internet grows. As a consequence, the processing time for route selections can be significantly reduced.

4. Since the gateway-network is responsible for all the internet

routing, inter-LAN communications become as easy as intra-LAN communications and are transparent to LAN users. As long as the address of a destination node is known (This can be obtained from Internet Route Server.), one can always initiate the inter-LAN communications. This relieves each LAN host from the burden of maintaining a huge routing information table, which can be a problem when the number of internet nodes becomes very large.

5. All gateways are essentially interconnected into a gateway-network. Therefore, effective schemes to control the internet congestion problems can be devised to prevent internet performance from degrading under overload conditions.

6. Communications with networks outside the internet normally involve charging activities and require special procedures, such as converting the protocols to make them compatible with those of public data networks. This can be conveniently handled by the root gateway, which will be responsible for such functions as management, maintenance and accounting of external traffic. However, this does not exclude the possibility that some nodes may still be able to communicate directly with any host outside the internet when it is appropriate to do so.

3.3.2 Disadvantages

1. A few extra gateways and communication links need to be installed, which means higher implementation costs.

2. A potential weakness of the hierarchical structure is the fact that when one or more gateways or links become faulty, the whole internet may be partitioned into several isolated parts which cannot communicate with each other.

3. For communications among neighboring LANs, one or two more hops may be needed in GATENET, as compared with the conventional gateway approach.

Initially, the first problem might seem undesirable. But as hardware costs continue to decline, we believe that the high-performance features and the versatile integrated voice/data services provided by the GATENET approach will justify the extra costs of constructing such a network.

The second problem of internet reliability can be improved through redundancy. The "buddy link" solution to be discussed in Chapter 5 can be used to overcome the link failures. Gateway failures can also be overcome by using backup processors. However, since these processors may be expensive, the installation of such backup components can be limited to only those gateways whose functions are critical to the internet communications.

3.4 Summary

In this chapter, we have shown the superiority of the GATENET approach over the conventional gateway approaches in supporting the voice and data internet transport functions. Based upon GATENET's hierarchical structure and hierarchical addressing scheme, internet routing becomes very simple and efficient. Under normal conditions, routing between each source and destination pair is static, thereby eliminating the need for sequencing. This characteristic is most favorable in the support of the voice traffic. However, if there should be any faulty conditions in the path, the internet communications might be interrupted. In Chapter 5, we present a "buddy link" solution to alleviate such problems.

CHAPTER IV
INTERNET COMMUNICATION PROTOCOLS

In this chapter, several sets of protocols are defined to support internet voice and data communications. In Section 4.1, we first examine several possible approaches to resolve protocol differences in interconnecting incompatible LANs and discuss their respective advantages and disadvantages. Next, three sets of protocols supporting GATENET internet transport functions are presented in Section 4.2. In particular, Section 4.2.1 deals with the data transport protocol and Section 4.2.2 deals with the voice transport protocol. Section 4.2.3 presents two levels of flow and congestion control protocols to ensure that satisfactory GATENET performance is maintained even under overload conditions. Section 4.2.4 suggests several enhanced transport layer protocols that might be helpful in constructing an internet. Finally, Section 4.3 presents a summary of the chapter.

4.1 Interconnecting Incompatible LANs

An internet normally consists of a wide variety of LANs implemented with different hardware technologies and incompatible protocols. Thus, in order for internetwork voice/data communications to be supported, some common rules must be agreed upon and obeyed by every LAN sharing the common resources. It is the function of internet communication protocols to achieve this goal.

In general, there are three common ways for incompatible LANs to communicate with one another within a campus-wide internet:

1. Augment the functionality of each LAN so that every LAN in the internet supports equivalent protocols and services.

2. Augment the functionality of gateways so that gateways alone resolve all the internet incompatibilities.

3. Augment the functionality of both gateways and LANs so that there is a uniform internetwork service level (mostly below the transport layer) across the overall internet.

In the first method, each LAN provides equivalent network services; thus, the internet becomes simply an extended network, and internet communication can be achieved via installation of physical links. However, the development of appropriate software for all the

participating LANs can be very complicated and costly, making this method unattractive. This problem is further compounded when the individual LANs are of heterogeneous types.

The second method may be a good choice when only a few LANs are to be interconnected with one another and no further expansion is needed. When there are many LANs to be interconnected, (e.g., within a campus area), this method becomes impractical since each gateway must be specially coded to handle the protocol incompatibilities of each connecting LANs.

The third method, which is widely used, has the potential to interconnect a vast number of LANs with a reasonable development cost. It basically requires that each participating node add an internet header (IH) for each internet packet, based upon which the switching nodes can make routing decisions and take appropriate actions. The source and the destination internet addresses as well as other control information needed for internet communication are contained in the internet header. This approach allows the protocol complexity to be reduced to a more manageable level; hence, the same set of protocols developed for one gateway can be migrated to another gateway with only a minor adaptation to the environment of each connecting LAN. It is this approach that we have adopted in the GATENET protocol design.

4.2 GATENET Communication Protocols

In GATENET, each gateway may encounter three types of incoming traffic: voice packets, data packets and control packets. Due to the timeliness constraint imposed on voice communication, voice packets receive the highest non-preemptive processing priority at gateway processors, control packets the next highest and data packets the lowest priority. Within each priority class, packets are processed according to the order of arrivals.

Due to the varying performance and reliability requirements, two different sets of protocols have been developed to handle voice and data traffic separately. Since control packets share the same reliability requirement as data packets, the control packets are transmitted using the same data transport protocol, but they have a higher processing priority at the gateway processors. Further, in order to minimize GATENET transmission delays, inter-gateway transmissions would not perform the retransmission function [13]. Should there be any error detected during inter-gateway transmissions, the packet would simply be discarded. As a remedy, the entry-gateway-to-exit-gateway (EGTEG) positive acknowledgment and retransmission schemes are used to ensure data integrity during the internet transmissions (The EGTEG positive acknowledgment scheme also serves the purpose of flow control, which will be discussed in Section 4.2.3.2.).

4.2.1 GATENET Data Transport Protocol (DTP)

Data communication in GATENET is basically datagram-based, and each packet is treated as an independent unit. Both data packets and control packets are handled by GATENET Data Transport Protocol (DTP).

4.2.1.1 DTP For Data Packets

To deliver a message via GATENET to a destination host located at a different LAN boundary, the source host need first prepare an internet data packet (i.e., a data segment plus an internet header), encapsulates it with a local network's header, and then routes it to the entry gateway connected to the same LAN. If the message size exceeds the maximum packet size allowed in GATENET, the packetization process must be invoked. Packetization can be accomplished either at the source host or the entry gateway (see Section 4.2.4.2).

After receiving the packet, the entry gateway will decapsulate the local network's header, examine the internet address, and then route it to the next gateway en route. Once the packet enters the gateway-network via the entry gateway, no more local header's encapsulation/decapsulation is needed until the packet reaches the exit gateway. The entry gateway will keep a copy of the internet packet

until a positive acknowledgment is returned from the exit gateway. Otherwise, it will retransmit the packet after a timeout period (up to some predefined number of times). This inter-gateway transmission step will, in general, be repeated several times until the internet packet finally reaches its exit gateway.

When the internet packet is routed to the exit gateway, the exit gateway returns a positive acknowledgment to the entry gateway. Since the internet packet leaves the gateway-network at this point, the exit gateway encapsulates the packet with a local network's header and then forwards it to the destination host. This completes an internet transmission.

4.2.1.2 DTP For Control Packets

There are two types of control packets in GATENET:

- voice control packets: These are mainly concerned with call setup, arrangement of communication options (including choices of vocoding techniques), monitoring, interrupts and call termination.

- data acknowledgment packets: Most EGTEG positive acknowledgments are piggybacked via data packets. However, if, after a waiting period, no data packet is heading for the same destination, a stand-alone acknowledgment will be sent out as a control packet.

Since the loss of control packets may cause more severe problems than that of voice packets and since the timeliness of the control packets is less critical than that of the voice packets, control packets and voice packets are handled separately [4, 5]. Control packets are transmitted across the internet the same way as data packets. However, before control packets are forwarded to the next gateway in the path, the receiving gateway may invoke some extra processing (see Section 4.2.2).

4.2.2 GATENET Voice Transport Protocol (VTP)

While interactive data users can respond to internet traffic congestion by slowing down their data exchange activities, voice users require a smoother internet services. In order to minimize the dispersion of internet transit delays, which is inherent because of the dynamics of packet switching environments, a virtual circuit approach is thus adopted for GATENET Voice Transport Protocol (VTP). With the virtual circuit approach, a voice call, once accepted, is guaranteed to continue the voice session without suffering from any significant delay fluctuation due to other internet activities. Further, through the pre-established path, the virtual circuit approach allows the use of abbreviated headers which can lead to reduced header overheads. This approach is most favorable in supporting the continuous long-lived high-bandwidth voice packet streams.

Furthermore, due to the nature of voice traffic, in which short delay variation is more desirable than speech integrity, no acknowledgment is needed. When transmission errors are detected by any receiving gateway en route, the voice packets will simply be dropped from the internet. A timestamp is associated with each voice packet so that in case of occasional packet loss out-of-order arrivals of voice packets may still be accepted as long as the timestamps are in an upward direction. Another function of the timestamp (used by the gateways) is to discard voice packets in transit whenever their lifetime exceeds a predetermined period.

When two users on different hosts in the internet wish to initiate a voice communication, an initial call setup is required to reserve buffers (one for either direction) at each gateway en route. Only when all such buffers are reserved can a voice call be allowed to proceed. A disconnection request can be issued later by either participating user to free all the resources reserved for such a voice session.

If a call setup request arrives at a gateway in the path but cannot be immediately accepted, the request will be held for a given period before a rejection message is initiated. If such a call setup request cannot be honored until the holding time expires, a rejection message will then be issued by the rejecting gateway. Such a rejection message

will also release those buffers reserved for this unsuccessful call setup request.

A reserved buffer has two states: AVAIL means it is ready for the next voice packet, and BUSY means it is holding a voice packet pending a transmission. During a call setup process, data rates between the source and destination ends must be properly established so that voice packets arriving at a BUSY reserved buffer will be extremely rare. The flow and congestion control mechanisms to be discussed in Section 4.2.3 can help prevent such undesirable situations. New voice packets will overrun the old ones if such a situation does arise.

Based upon VTP, some voice-related higher level protocols, such as the conference protocol, can then be added on top of it. For such higher level protocols, the portion dealing with the transport of voice contents still remains the same as VTP; the portion dealing with the control aspects, however, needs to be further enhanced. In particular, a control scheme must be devised to ensure fair floor assignment among the conferencees. Interested readers are referred to [4] for a more detailed discussion.

4.2.3 GATENET Flow and Congestion Control Protocols

Flow and congestion control protocols are protocols used to regulate network traffic flows so that the network can still provide satisfactory data and voice services even under overload conditions. Rigorously speaking, flow control is distinguished from congestion control. Flow control is generally applied on an end-to-end basis to prevent the sender from sending packets at a rate faster than the receiver can process it, while congestion control is applied in the communication subnet to deal with situations where there are more arriving packets than the available buffers at the switching nodes. However, for the purposes of this discussion, the term "flow control" will be used in a loose sense to stand for flow and congestion control.

As shown in Figure 11, in an uncontrolled network, as the offered load increases, the network throughput usually increases correspondingly up to a maximum level, and then the throughput degrades rapidly to a very low level as the input traffic exceeds the network carrying capacity. At first thought, one might suggest that the network be designed in such a way that under no circumstances can the maximum allowable offered load exceed the underlying network capacity. But due to the bursty nature of the packet switched environments, such an approach will result in an undesirable situation in which the network resources are

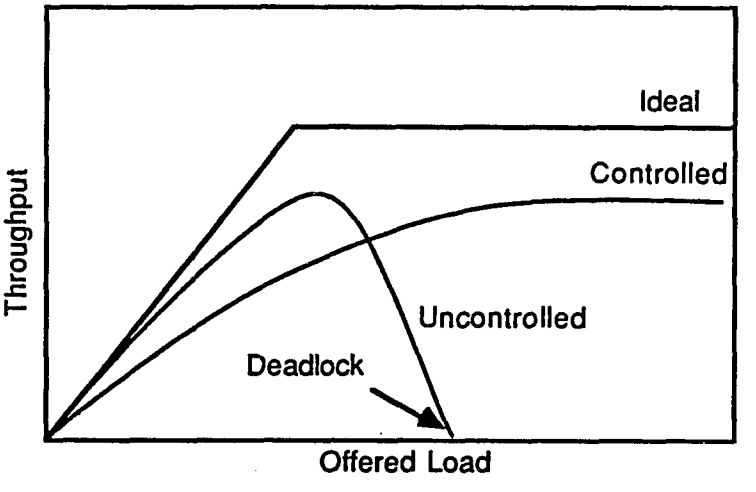


Figure 11. Throughput vs Flow Control [11]

underutilized most of the time. Hence, to avoid the shortcoming of the aforementioned conservative approach, most networks are designed and tuned to achieve the best performance under the projected average traffic. As a direct consequence of this design principle, congestion control mechanisms are thus needed to prevent network performance degradation when the offered load occasionally exceeds the network capacity.

In general, most of the congestion control mechanisms fall into the following categories [30]:

1. Preallocating network resources before the communication activities take place.

2. Allowing the switching nodes to drop packets when certain predefined threshold conditions are met.

3. Setting up an upper bound for the total number of packets that can be in the subnet at any given time.

4. Restricting or throttling off new input traffic when the network is congested.

5. Applying flow control mechanisms to control the traffic between a sender and a receiver. Note that flow control is a means of congestion control, but using flow control alone is not sufficient to prevent network congestion since end-to-end traffic control can only affect a small portion of total network traffic.

Congestion control is further compounded in an internet environment where there may be many LANs of varying flow and congestion control policies interconnected and interacting with one another. Furthermore, if voice and data integrated services are to be supported through the same internet, due to the differing performance and reliability requirements of data and voice traffic, new flow and congestion control schemes, other than those conventional ones for either data or voice traffic, must be designed in order to accommodate such integrated internetwork services.

In most of the current internet technology, a gateway is normally used to mediate between different networks; hence, gateways are spread among the connecting LANs in an unstructured and uncoordinated way. As a result, it is extremely difficult to apply an effective overall internet congestion control, and the internet performance becomes unpredictable and may degrade sharply when network overloading occurs. Therefore, one of the goals of this research is to identify and propose a suitable internet architecture that can facilitate the exercise of internet congestion control. In our GATENET approach, through the unique hierarchical integration of internet gateways, all gateways are essentially interconnected into a gateway-network. Such a gateway-network is essentially separated from the connecting LANs and, thus, can provide a better environment for internet congestion control.

During the course of this research work, several versions of flow and congestion control mechanisms were studied using GATESIM network communication simulator (which will be further discussed in Chapter 6). On the basis of these simulation studies, we have concluded that two levels of distributed flow and congestion control mechanisms are needed to ensure the proper function of GATENET under heavy load conditions. These two levels are described as follows:

1. hop level: regulates the traffic between LANs and a gateway as well as between a gateway and another gateway;

2. entry-gateway-to-exit-gateway (EGTEG) level: regulates the traffic between an entry gateway and an exit gateway.

4.2.3.1 Hop Level Flow Control

GATENET hop level flow control uses the principle of the Input Buffer Limit (IBL) strategy [11], which classifies incoming traffic into several priority classes and throttles the lower priority traffic based upon buffer utilization at each individual entry node. IBL, which is a distributed congestion control method, keeps track of the local congestion rather than the global congestion at each entry gateway, thus providing a simple and cost-effective way to achieve congestion control. The rationale behind this strategy is that, through the backpressure effect, entry node congestion can often provide reliable indication of internal network congestion conditions.

There have been several versions of IBL strategy proposed to achieve congestion control over data communication networks. All of these methods basically distinguish between input traffic and transit traffic and give priority to transit traffic when congestion conditions occur at an entry node. In Lam's work [16, 17], an input packet is dropped whenever the total number of input packets exceeds a predefined quota. Kamoun's scheme [14], on the other hand, drops an input packet once the total number of input and transit packets exceeds a given threshold. Both of their studies show that with respect to a given network topology and traffic pattern an optimal input buffer limit can always be found to maximize network throughput under heavy load conditions. Input buffer limits higher or lower than the optimal value will lead to a substantial throughput degradation.

In GATENET, due to its support of both data and voice traffic, hop level congestion control is more complicated than the aforementioned IBL schemes. The hop level congestion control applied at each GATENET gateway can be summarized as follows:

- At any given time, no more than a predetermined number of buffers (say, MAXVOICE) can be allocated for voice calls.

- When the total buffer utilization exceeds a predefined threshold, no more input data traffic will be accepted unless the total number of input packets in the gateway is less than a given quota (say, MAXINDATA).

Due to the nature of voice communication, as mentioned earlier, voice packets are given the highest processing priority. However, since the number of buffers provided by a gateway usually exceeds the number of calls a gateway can handle, to avoid voice traffic tying up a gateway processor, the number of simultaneous voice calls through a gateway must be restricted so that data traffic can have a fair share of the processor time.

Further, in contrast to Lam's scheme, the gateway limits the amount of input traffic only as the threshold level is reached so as to eliminate unnecessary flow constraints under light traffic conditions (see Chapter 6 for further discussions of GATENET's performance).

4.2.3.2 Entry-Gateway-To-Exit-Gateway (EGTEG) Level Flow Control

One of the most common problems in network operation is buffer congestion at the exit point. When different LAN technologies are involved, the exit gateway must resolve the speed mismatch between the source and the destination ends so that a single source will not overload the corresponding exit gateway or the global gateway-network. Since voice calls are controlled through preallocation measures, the EGTEG flow control is primarily designed to regulate internet data traffic.

The basic concept of the EGTEG flow control method is as follows:

1. EGTEG flow control is exercised on an entry-gateway-to-exit-gateway basis, and a path between each entry gateway and exit gateway is considered a logical pipe.

2. Each pipe is individually flow controlled by a window mechanism. Namely, at any given time, no more than a fixed number of unacknowledged data packets can exist in each pipe.

To be specific, each entry gateway will keep track of the status of each outgoing pipe and maintain a copy of each admitted new data packet. The exit gateway will return an acknowledgment for each correctly received data packet. On receipt of the acknowledgment, the entry gateway then removes the copy of the acknowledged data packet and adjusts the transmission window of the associated pipe accordingly. If the acknowledgment is not received within a certain time-out period, the entry gateway will retransmit the packet (up to some predetermined number of times).

When the internet becomes congested, the transit delays of the acknowledgments will be prolonged. Hence, in addition to regulating the traffic generated at a single entry gateway so as not to overload a corresponding exit gateway, the EGTEG flow control also has the effect of slowing down the traffic rate that the entry gateway will send into the congested areas.

4.2.4 Enhanced Protocol Support at Gateways

Most of the current internetwork protocol research has focused on the network layer (see Figure 12). Therefore, any two application layer users who wish to communicate with each other must either choose common protocols for layers 4 through 7 or find some way to perform the protocol translation [2]. This requirement often needs substantial development work and may impose unnecessary restrictions over many user applications.

As mentioned in Chapter 2, several worldwide organizations (e.g., ISO, ANSI, NBS, and ECMA) have been working on standardization issues. However, no worldwide agreement concerning LAN's organizational structures, layer functions or internetworking strategies has yet been reached. Quite a few different standards have been proposed (e.g., IEEE Project 802), each one having its pros and cons both technically and politically. Thus, it is commonly felt that the current emerging diversity of local area networks will continue for a while [25].

As a result, gateways will continue to play an important role in achieving internetworking. However, there are still many controversial design issues related to the functionality of gateways in internet communications. When the gateway was first built in the DARPA internet, the idea was to make it as simple as possible; consequently, it

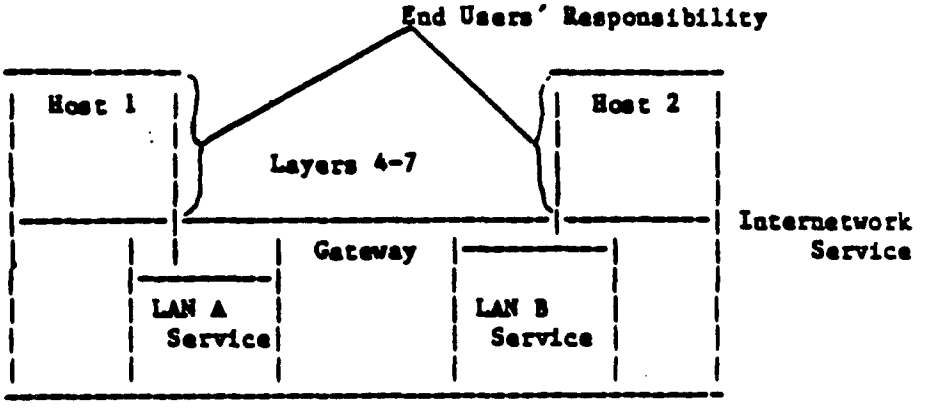


Figure 12. The functionality of a gateway

was basically implemented as a store and forward unit for datagrams. In Saltzer's source routing approach, gateways were also envisioned as simple store and forward units (In a campus environment, these gateways can be implemented by using specially designed microprocessor devices.). However, because of the high bandwidth nature and differing characteristics of LANs, a simple gateway seems inadequate to provide the functionality needed for interconnecting many LANs within a campus or corporate boundary. Hence, the traditional functionality of gateways must be enhanced so as to facilitate internetwork communications.

Nevertheless, a gateway can not provide all the services needed by each connecting LAN, since these services require a complicated set of protocols which will transform the gateway itself into a bottleneck in the internet system. Hence, when deciding if a service should be included or not, one needs to carefully study the discretion criteria, such as whether such a service is needed by a majority of connecting LANs (if not by all) or to what degree such a service would degrade the gateway performance.

In the GATENET design, in order to resolve the internet incompatibilities without greatly sacrificing performance, gateways are enhanced with some optional transport protocol support in addition to current network layer protocol support. Such support will necessitate

additional LAN customized specifications be kept at each end gateway (An end gateway table is used for maintaining these specifications.).

In the following, we present two examples which demonstrate that this enhanced protocol support at gateways can be useful in achieving internet communications in an incompatible environment.

4.2.4.1 End-to-End Acknowledgment

End-to-end acknowledgment[2] schemes are normally provided by the transport layer protocol. However, due to the diversity of LANs, some reliability-oriented LANs may choose to support it, some performance-oriented LANs may choose not to, while most LANs may choose to allow either. If an internet is designed to allow either option (a subfield in the internet header specifies this option), some problems might occur when two LANs with different options wish to communicate with each other (say, LAN A with an acknowledgment option and LAN B with a no-acknowledgment option).

Our solution to this problem is to enhance the functionality of the

---

[2]An end-to-end acknowledgment is different from an entry-gateway-to-exit-gateway acknowledgment, since an end-to-end acknowledgment is usually a message exchange between two end users instead of two end gateways.

end gateways so that they will resolve such an incompatibility. Consider the following two cases (see Figure 13):

1. Host 1 is the receiver, and Host 2 is the sender.

   When Host 2 sends a packet to Host 1, Host 2 will expect an acknowledgment from Host 1, but Host 1 will not acknowledge. The solution is to enhance the gateway functions so that the internet header will be checked when the end gateway of LAN A receives a packet from LAN B. If such a packet requires an acknowledgment and if LAN A has set the no-acknowledgment option (this information is kept in the end gateway table), the end gateway of LAN A will complete the transaction by first forwarding the packet to Host 1 and then issuing an acknowledgment to LAN B (The end gateway is considered an end user under such a situation.).

2. Host 1 is the sender, and Host 2 is the receiver.

   When Host 1 sends a packet to Host 2, Host 2 will always return an acknowledgment. When the acknowledgment arrives at the end gateway of LAN A, since LAN A has a no-acknowledgment option, the end gateway will simply drop such an acknowledgment.

As shown in the first strategy, a real end-to-end significance of acknowledgment is not preserved as it is in most transport layer protocols; consequently, this approach is sometimes controversial.

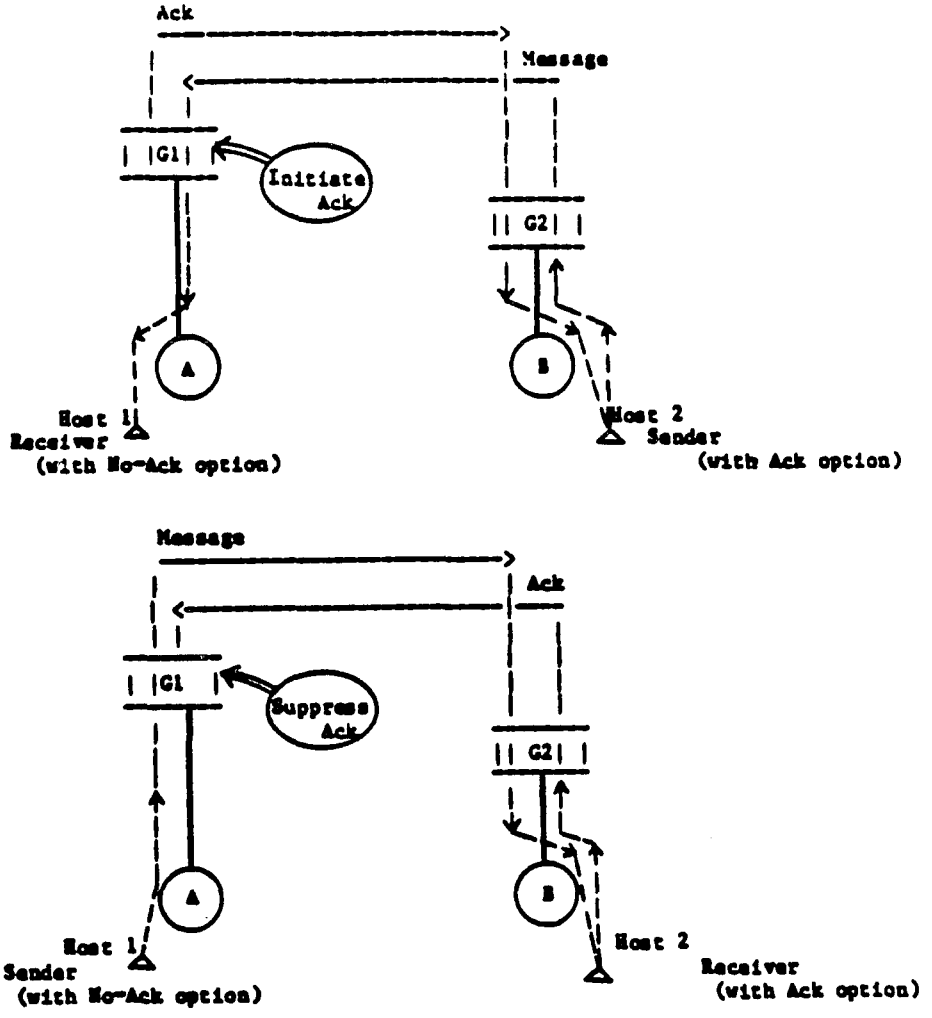


Figure 13. Communication between LANs with different end-to-end acknowledgment schemes

However, we think this approach is a good design tradeoff and should be able to satisfy most LAN applications without incurring costly software development efforts.

4.2.4.2 Packetization and Reassembly Services

In an internet environment with the interconnection of many different LANs, packetization and reassembly services are often needed for long messages. It is very likely that some LANs do not have such capabilities but would like to receive such services from the internet. Naturally, the end gateway is the best choice for providing this service (If the connecting LAN is already equipped with such a function, the end gateway will incur no overhead.).

- Packetization: In GATENET, all the gateways have the same maximum packet size (MAXSIZE). Whenever a gateway receives a packet with a size exceeding MAXSIZE, a packetization process will first be invoked and then a reassembly service request will be initiated. The fragmented packets can not be delivered until the destination end permits the reassembly request.

- Reassembly: Under heavy load conditions, packet reassembly at the destination end often leads to deadlocks if buffer management is not properly designed. Hence, when a reassembly service is needed, the source end (which can be a host or a gateway depending on where packetization is performed) needs first to send a reasonable buffer allocation request to the destination end before transmission starts.

Once the destination end gateway receives the request, it checks the end gateway table.

- If the destination LAN has set the reassembly option, the end gateway will honor the allocation request and take proper action according to its current buffer utilization status. If the end gateway's buffer usage is high for the moment, the request can be rejected, and the source end can then retransmit the request after a pre-established time-out period.

- Otherwise, the end gateway simply passes the request to the destination host and lets the destination host handle such a request.

4.3 Summary

In this chapter, separate protocols for data and voice traffic have been defined. Two levels of flow and congestion control schemes have been introduced to handle internet congestion problems. Enhanced transport layer protocol support has also been suggested to reduce protocol incompatibility problems. Based upon the support of these protocols, GATENET is capable of supporting many high level user activities, such as data/voice file transfer, voice conferencing, data base applications, etc.

CHAPTER V
RELIABILITY OF INTERNET TRANSPORT SYSTEM

An internet usually consists of a large number of computing and communication resources spread over some extended geographical area. The reliability of such an internet transport system thus becomes an important design concern. In this chapter, a "buddy link" scheme is presented to improve the reliability of the GATENET design. Section 5.1 gives a description of the "buddy link" scheme. Based on the "buddy link" scheme, Section 5.2 then elaborates several case studies of various link failure conditions. Section 5.3 presents a summary.

5.1 Buddy Link Reliability Scheme

As discussed in Chapter 3, the GATENET design adopts the hierarchical structure and hierarchical addressing scheme to support the internet voice and data transport functions. While this hierarchical approach reduces the size of the routing tables and minimizes the processing time for route selections, it suffers from a potential problem of internet partitioning under faulty conditions.

In general, reliability problems in GATENET can be classified into two types: gateway failures and link failures. Both types of failures can be overcome by using backup components. However, a complete redundancy solution is often not economically justified in a campus-wide internet environment. Hence, a "buddy link" scheme is presented as a cost-effective way to improve the internet reliability. Based upon such a scheme, a single buddy link, if properly installed, can serve as the backup link for any tree link in the associated "buddy loop."

Depending on vitality, traffic load, and distance, a gateway may choose to have certain gateways (other than its parent and sons) as its "buddy gateways" connected by extra "buddy links" which need not be on the same hierarchical level (See Figure 14). The regular links, which preserve the GATENET hierarchical structure, are termed "tree links" to

distinguish them from buddy links. Assume that G2 and G3 are buddy gateways connected by a buddy link BL and that GA is the youngest common ancestor of G2 and G3. A "buddy loop" is then defined as a closed path passing through G2, BL, G3 and GA. In general, incorporating such buddy links will change the gateway-network from a hierarchical structure to a meshed structure. However, with the introduction of buddy routing tables, effective routing procedures can still be maintained.

At each gateway, in addition to a normal routing table, a buddy routing table (BRT) is needed to support route alternations under link failure conditions. Since a gateway may be associated with several buddy loops, the buddy routing table maintains a sub-buddy routing table (SBRT) for each associated buddy loop. Each SBRT keeps an entry for each tree link within the associated buddy loop. Each entry in turn consists of four subfields:

- ENABLE, which shows if this entry is enabled or not;
- FLINK, which shows the ID of the possible faulty link;
- ADDR, which gives a description of the cluster addresses to be detoured;
- RLINK, which shows the link to be used for detouring.

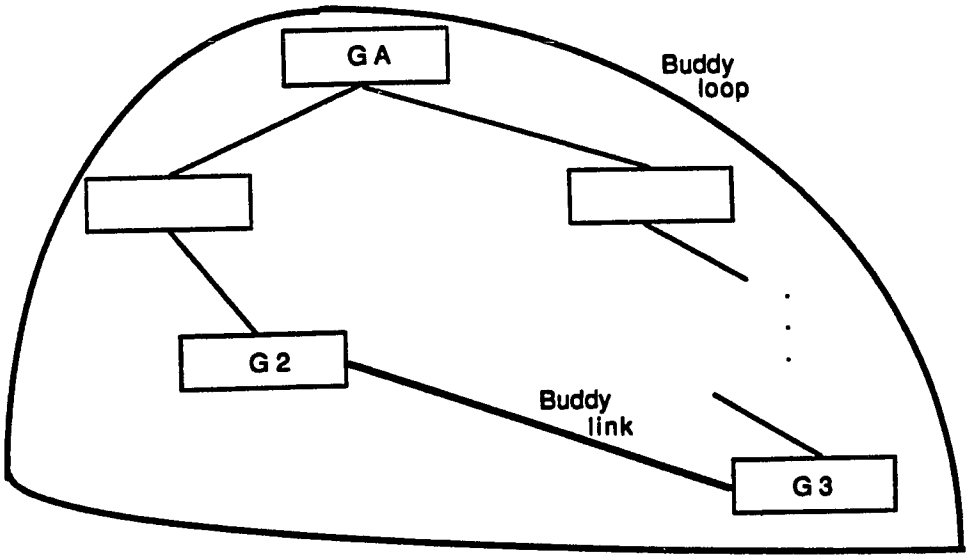


Figure 14. Buddy gateways, buddy link and buddy loop

When a buddy link is installed, the youngest common ancestor of the gateways on either side of the buddy link (i.e., GA) will be notified. Appropriate detour information will then be sent by GA to each gateway in the associated buddy loop to update its buddy routing table to reflect such a change. Under normal conditions, the buddy link can only be used to deliver those packets whose destination nodes are the descendants of the buddy gateways (i.e., G2 or G3).

A special flag RED is maintained at each gateway to signal if route alternations are in effect. When the RED flag is set to the ON state, the gateway is reminded that when making route selections the buddy routing table should be checked before the normal routing table. Since the buddy link is used by both buddy gateways even under normal conditions, the RED flag at either buddy gateway is set to the ON state after the buddy gateway is installed.

When a link failure occurs, two situations may arise. First, if the faulty link happens to be a buddy link, the only action that needs to be taken is to disable the associated entry (making use of such a buddy link) in the buddy routing table of either buddy gateway. The RED flag is set to the OFF state if no more entries are enabled in the buddy routing table. Second, if the faulty link is a tree link, the upper gateway of the faulty link first chooses a proper buddy loop for

subsequent route alternations. Usually, the buddy loop with the smallest number of links will be chosen, and an arbitration can be made if there is a tie. A "detour" control packet containing the ID of the faulty link and the chosen buddy loop is then initiated and passed to each gateway in the chosen buddy loop. If, unfortunately, the detour control packet should encounter any further faulty conditions en route, a "detour cancellation" would be returned all the way back to the upper gateway to cancel such a detour action. The upper gateway then picks up the next smaller buddy loop and repeats the same process until either the detour announcement is successfully carried out or there is no more available buddy loop.

Upon receiving a "detour" control packet, the gateway then sets the RED flag to the ON state (if it is not on yet) and enables the associated entry in the buddy routing table. It should be noted that some packets may be lost during the detour transition, but they can be recovered through the EGTEG positive acknowledgment and retransmission schemes.

When a detour action is in effect, route selections at the gateway on either side of the buddy link (i.e., G2 or G3) are more complicated, since a nonlocal packet may be routed to either its parent gateway or its buddy gateway. Unfortunately, since only local cluster address

patterns are maintained, the gateway is unable to make a choice between the two alternate routes; thus, a nonlocal packet is always forwarded to its parent gateway. If the parent gateway and its associated ancestors and descendants cannot deliver such a packet to its destination, the packet will eventually be returned to the gateway for detouring. As a result, it is possible that a packet may visit a gateway the second time. To facilitate handling such a detouring process, a DETOUR subfield is introduced in the internet header to provide information pertaining to whether a packet has been detoured. The DETOUR subfield will be set to a value of YES when a packet is detoured. However, since the DETOUR subfield is used only to decide if a nonlocal packet should be routed through a buddy link at a buddy gateway, its value will be reset to NO once a packet is detoured through a buddy link.

When the faulty link has been repaired, depending on whether it is a buddy link or tree link, each associated gateway will be notified, and proper actions will be taken to restore normal operating conditions.

For data traffic, the above scheme is sufficient for route alternations. For voice traffic, however, some refinements must be added because of the virtual circuit approach. Since a buffer reservation must be made at each gateway en route for each voice session, after the link

failures and subsequent route alternations, a voice packet may be routed to some gateway where no buffer reservation is made for such a voice session. If this situation occurs and there are buffers available, the gateway may accept such a voice packet and automatically initiate a buffer reservation for this voice session. If the buffer reservation cannot be honored at that time because of the limitations of the allowed active voice sessions, a special request will be queued by the gateway. As soon as the gateway has a free slot, the reservation will be honored with the highest priority. This process continues until either such a voice session finally secures a buffer reservation or a call termination request of this voice session arrives.

5.2 Case Studies of Buddy Link Scheme

In this section, scenarios of route alternations are presented with respect to various link failure conditions. As shown in Figure 15, Gateways G2 and G3 become buddy gateways through Buddy Link BL (the youngest common ancestor being G4). The buddy loop associated with buddy link BL is the closed path Y-X-BL-A. The SBRT of each gateway associated with the buddy link BL is shown in Figure 16 where NLOC denotes any nonlocal transit packets and NLOCDET denotes any nonlocal transit packets with a value of YES in the subfield DETOUR (of the internet header).

Since there are three tree links in the buddy loop, three distinct link failure conditions may occur. Each of these conditions is discussed below.

- Case 1: Link A fails.
    1. Nonlocal transit packets at G2 (from its descendants) are detoured via Link BL.

    2. Transit packets at G1 with the address pattern 4.1.*.* are detoured via Link Y.

    3. Transit packets at G4 with the address pattern 4.1.*.* are detoured via Link X.

    4. Transit packets at G3 with the address pattern 4.1.*.* are detoured via Link BL.

- Case 2: Link X fails. This situation is similar to Case 1; hence, it will not be detailed here.

- Case 3: Link Y fails.
    1. Transit packets at G4 with the address pattern 4.*.*.* are detoured via Link X.

    2. Transit packets at G3 with the address pattern 4.*.*.* are detoured via Link BL.

    3. Nonlocal transit packets at G1 are detoured via Link A.

    4. At G2, nonlocal transit packets can be classified into two different classes. The first class consists of those packets whose destinations can be reached through Link

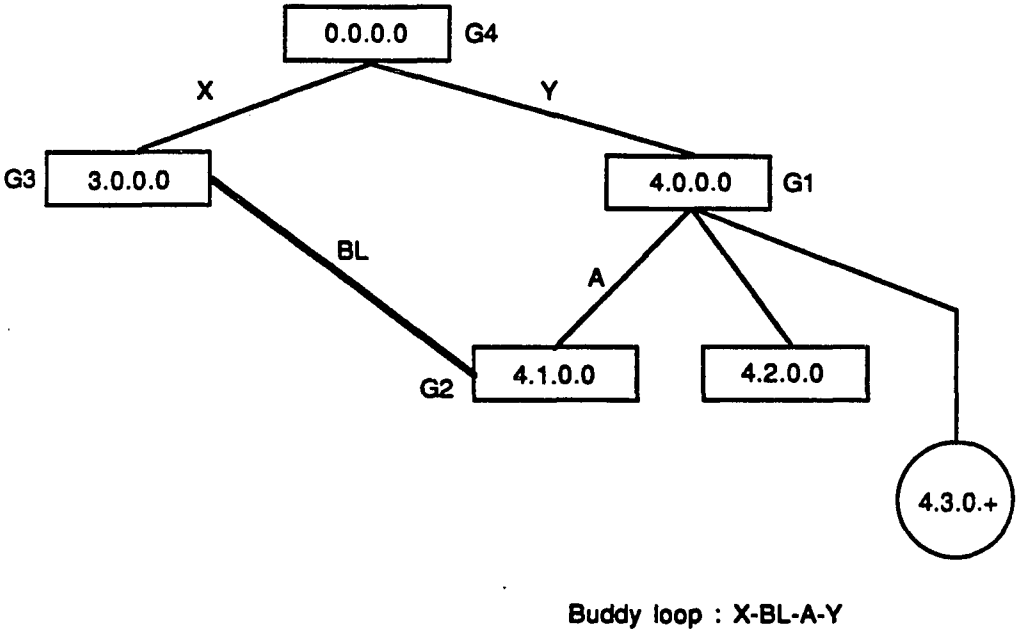


Figure 15. Example of buddy link scheme

0	X	3.*.*.*	A
0	Y	NLOC	A
0	A	4.1.*.*	Y

a. at Gateway 4.0.0.0 (G1)

1	X	3.*.*.*	BL
0	Y	NLOCDET	BL
0	A	NLOC	BL

b. at Gateway 4.1.0.0 (G2)

0	X	NLOC	BL
0	Y	4.*.*.*	BL
1	A	4.1.*.*	BL

c. at Gateway 3.0.0.0 (G3)

0	X	3.*.*.*	Y
0	Y	4.*.*.*	X
0	A	4.1.*.*	X

d. at Gateway 0.0.0.0 (G4)

Figure 16. Sub-buddy routing tables (under normal conditions)

A. The second class consists of those packets whose destinations can be reached through Link BL. The subfield DETOUR in the internet header can be used to help distinguish these two different classes of packets.

When a nonlocal transit packet P (say, with the destination address 3.2.1.4) arrives at G2 with a value of NO in the DETOUR subfield, there is insufficient routing information for G2 to decide to which link it must forward such a packet. Thus, packet P will be sent through Link A to G2's parent node G1.

When G1 receives packet P, the packet will be delivered to a proper destination node if it is a local packet for G1; otherwise, the packet will be detoured via Link A (instead of going through faulty Link Y), setting a value of YES in the DETOUR subfield.

When packet P arrives at G2 a second time, since it is a nonlocal packet with a value of YES in the DETOUR subfield, it will be detoured via Link BL to reach its final destination.

Hence, a single buddy link BL can be used as the backup link for tree links X, Y and A. The internet transport system may still be able to maintain its integrity without suffering from any partitioning should any one of the tree links in the associated buddy loop fail.

5.3 Summary

In this chapter, a "buddy link" scheme has been shown to be a cost-effective means to improve the reliability of the GATENET internet transport system. When properly installed, a buddy link can be used to provide route alternations for any faulty tree link in the associated buddy loop, thereby significantly reducing the number of extra links needed to prevent internet partitioning.

CHAPTER VI
PERFORMANCE EVALUATIONS OF GATENET

This chapter presents performance characteristics of the GATENET design utilizing the architectures and protocols discussed in Chapters 3 and 4. Section 6.1 briefly describes the motivations for conducting such a performance study. Section 6.2 outlines a simulation model of GATENET, and Section 6.3 shows the parameters and assumptions used in the simulation runs. Performance indices of interest are described in Section 6.4. Section 6.5 presents a detailed discussion of GATENET performance under various operating and traffic conditions. Finally, a summary is presented.

6.1 Motivations

Flow and congestion control has been a major research topic in the design of local area networks because of its substantial impact on the performance of the underlying network operations. Due to the varying network technologies and traffic patterns involved with each installation, it is very difficult to obtain a good control strategy which will solve the network congestion problems. The congestion problem is further complicated in an internet environment where there are many LANs of different characteristics interconnected and interacting with one another. In addition, if both data and voice communication services are to be supported by the same internet, special mechanisms must be devised to accommodate the different requirements of data and voice traffic.

Performance optimization is often the most important goal when designing a communication system. Many features, such as throughput, delay, reliability, fairness, cost and expansibility, can be used as the evaluation indices of network performance. Frequently, however, a design strategy chosen to optimize a performance index may lead to the degradation of other performance indices. It is therefore unrealistic to expect that a single design choice will optimize the performance in all ways. Thus, a tradeoff is unavoidable in the process of reaching a design decision.

Due to the inherent complexity of modern communication systems, the interactions among different design parameters are often difficult to analyze or predict without conducting a detailed simulation study or actually measuring the performance of the target system. Thus, to aid our design work during the course of this research, a network communication simulator, called GATESIM, was developed as a tool to study the performance of GATENET.

GATESIM is written in SIMSCRIPT using discrete event-driven simulation techniques. Although GATESIM has been used primarily to evaluate the performance of GATENET, it is, in fact, a rather general purpose network communication simulator. In GATESIM, each major network feature is defined as a separate routine; hence, with some modifications, GATESIM can easily be reconfigured to model different network topologies, routing, or flow and congestion control mechanisms.

Initially, GATESIM was used to investigate and compare the various flow and congestion control mechanisms. After thoroughly studying the behavior of GATENET, we determined that two levels of flow and congestion control mechanisms would be required to improve GATENET's performance under heavy load conditions (see Section 4.2.3). Later, GATESIM was further used to evaluate GATENET's performance under various traffic patterns and to show how the

parameters related to the proposed flow and congestion control mechanisms affect the performance. It is the latter part of the performance study that will be covered in this chapter.

6.2 A Simulation Model of GATENET

As shown in Figure 17, our simulation model of GATENET consists of seven gateways, 12 regular links and two buddy links, with each link in a simplex mode. Gateways and links are modeled as follows:

Each gateway G (see Figure 18) is modeled as a single server with three priority FCFS queues and a bounded buffer pool. The first queue, which has the highest processing priority, contains the voice packets. The second queue, which has the next highest processing priority, contains the control packets. The third queue, having the lowest processing priority, contains the data packets. The bounded buffer pool is shared by all types of packets and is subject to the specified flow and congestion control schemes.

Each gateway is also associated with a source generator, voice generators, and a sink. At a specified rate, the source generator

generates new incoming traffic, including data packets and voice setup control packets. Once a voice session is established, each speaker on either side is modeled with a voice generator, which generates voice packets at a fixed rate. The sink consumes all the packets delivered to it.

Each communication link (in simplex mode) is modeled as a single server with one FCFS queue. No buffer pool is associated with a link.

6.3 Simulation Assumptions and Parameters

This section describes the assumptions and the parameters used in the simulation.

1. The buffer pool at each gateway is structured into a number of segments, each consisting of 72 bytes. Packets arriving at a gateway without allocation of sufficient segments are discarded.

2. The length of a data packet is determined according to the distribution shown in Figure 19. The maximum length of a data packet is 512 bytes, each with an extra internet header of 26 bytes. The length of a voice packet is 128 bytes, each with an extra abbreviated internet header of 16 bytes. The length of a control packet is 32 bytes, including the internet header.

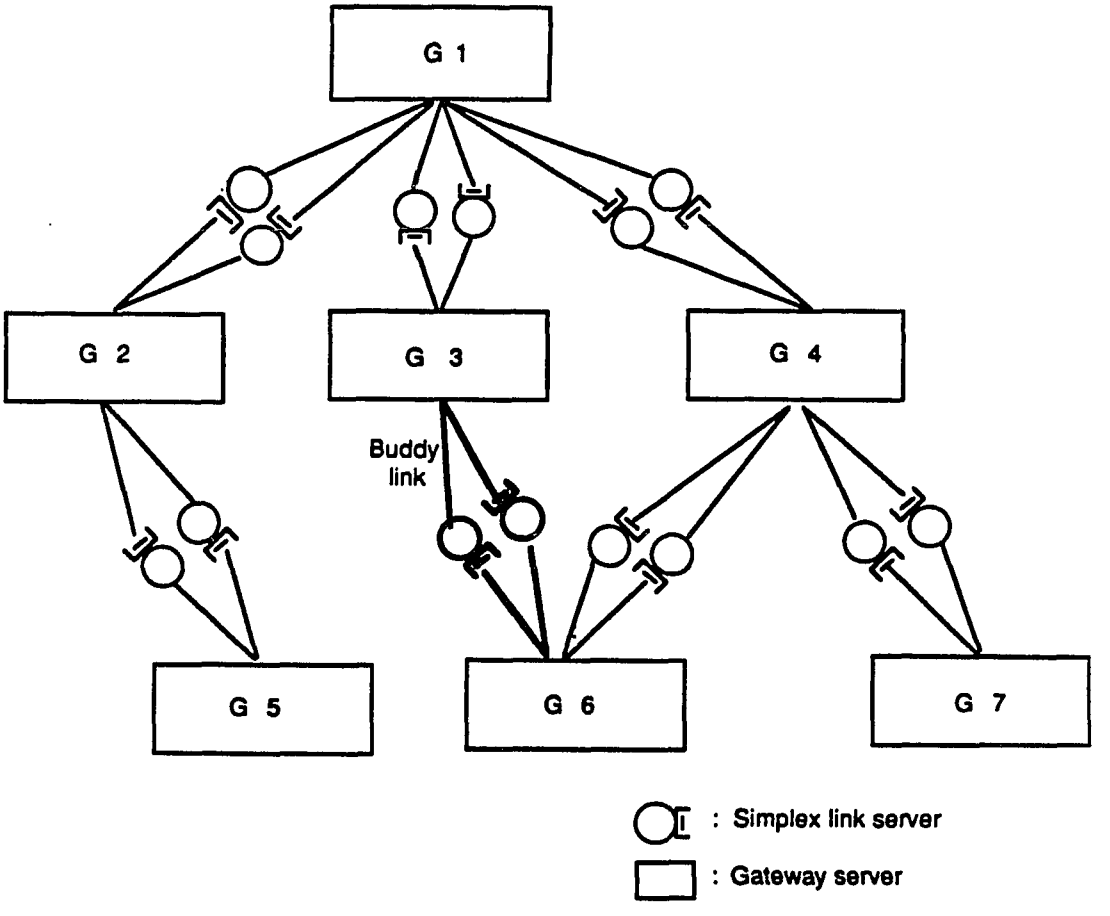


Figure 17. Simulation model of GATENET

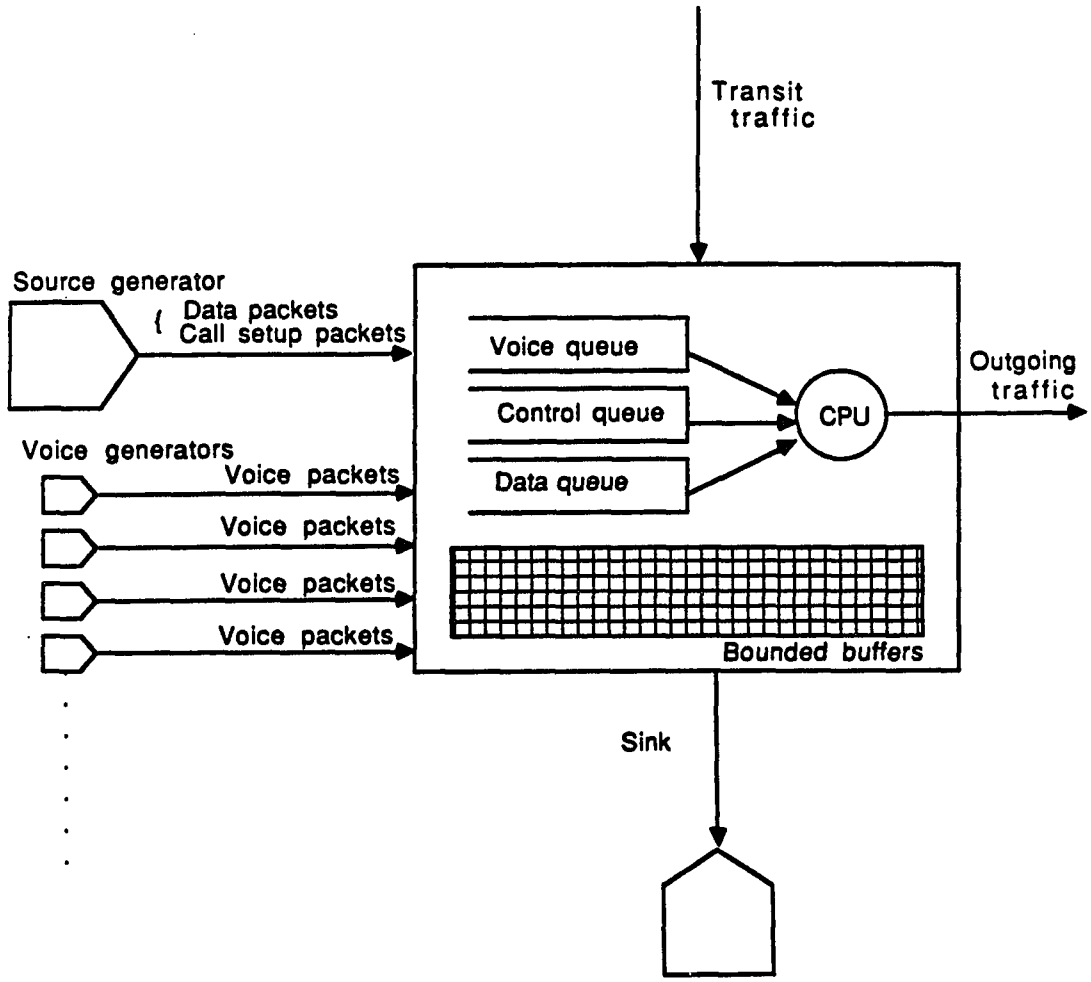


Figure 18. Simulation model of a gateway

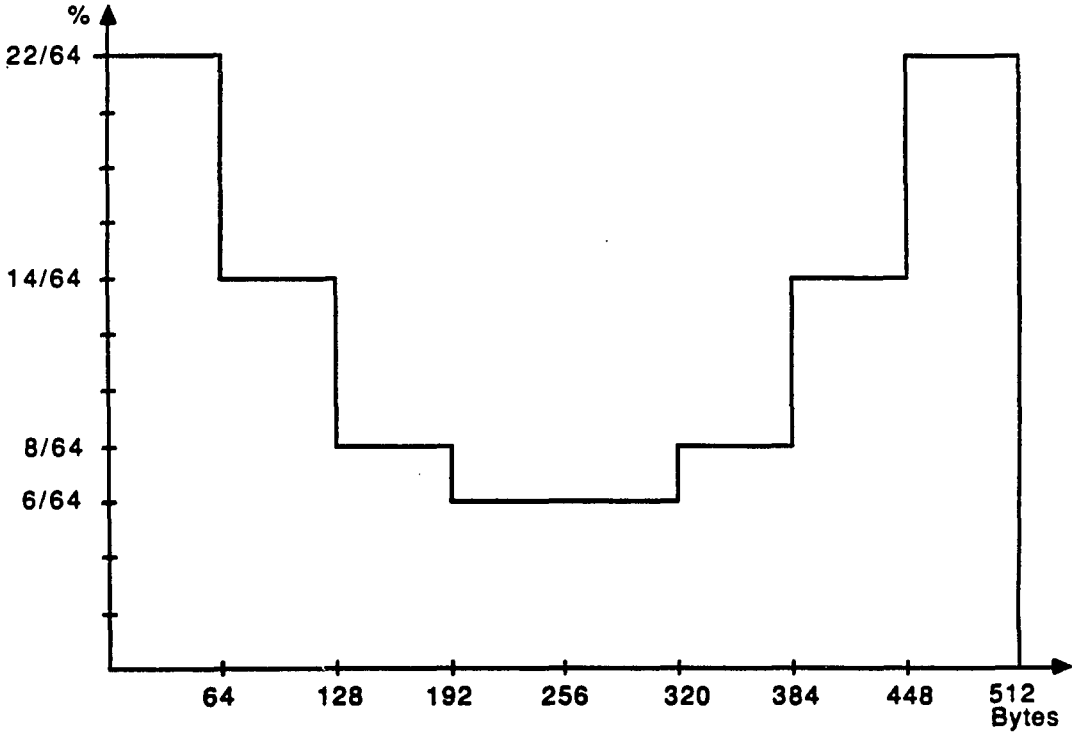


Figure 19. Distribution of data packet length

4. Let the total number of segments in the buffer pool at each gateway be TOTALBUF. According to the flow and congestion control mechanisms specified in Section 4.2.3, no more than MAXVOICE segments can be allocated to voice traffic at any given time. When the buffer utilization exceeds THRESHOLD, no more input data packets will be accepted unless the total number of input data packets is less than MAXINDATA.

5. Since the size of the control packets is rather small, it is assumed that the control packets are always accepted by the gateways, accounting for no usage of the aforementioned buffer pool.

6. The window size for the EGTEG flow control is WINDOW.

7. The rate of the new incoming packets generated by a source generator is assumed to be a poisson process with a mean IR. Among the new packets, 0.14% are voice setup control packets; the rest are data packets. It is also assumed that the destinations of the generated traffic are distributed uniformly across the GATENET.

8. With proper vocoding and silence detection techniques, each voice generator is assumed to generate voice data at the rate of 16 Kbps. Since the length of a voice packet is 128 bytes, 16 voice packets per second will be generated by each voice generator.

9. In real environments, an EGTEG positive acknowledgment is

usually piggybacked through a data packet; hence, no extra processing time is incurred at the intermediate gateways. In order to account for delayed acknowledgments, we have treated an EGTEG acknowledgment as a special data packet which requires no processing time. Such a data packet is placed in the third queue waiting to be serviced when it arrives at a busy gateway. By doing so, we can get a more accurate estimate of the transit delay of the EGTEG acknowledgment.

10. The gateway processes a packet at a speed of 1/1800 of a second. Each interrupt for either an acknowledgment timeout or a call holding timeout takes 1/10,000 of a second. Note that due to the amount of the traffic load, the root gateway is assumed to have twice the processing speed and twice the size of the buffer pool as do the rest of the gateways.

11. It is assumed that all the packets will be consumed at their respective exit gateways.

12. The length of a voice call is assumed to be exponentially distributed with a mean of 150 seconds.

13. The transmission speed at each link is 1,500,000 bps. Transmission errors are assumed to be negligible.

14. The timeout period is 3.5 seconds. A data packet may be retransmitted up to three times.

15. The maximum holding period for each voice connection request is 0.5 seconds.

6.4 Performance Measurements

The performance indices of concern in this study are as follows:

1. Throughput (packets/sec): the number of packets per second delivered to all the sinks in GATENET. Throughputs for data traffic and voice traffic are measured separately.

2. Transit delay (sec): the time period between the arrival of a packet at the source gateway and its delivery to the exit gateway. Transit delays for data, control, and voice packets are measured separately.

3. Transit data blocking probability: the ratio of the number of dropped transit data packets to the total number of data packets admitted in the GATENET.

4. Incoming data blocking probability: the ratio of the number of admitted incoming data packets to the total number of incoming data packets.

5. Incoming call blocking probability: the ratio of the number of rejected call requests to the total number of incoming call requests in the GATENET.

6.5 Simulation Results

In this section, the results of five sets of experiments are examined. Section 6.5.1 describes GATENET's performance with respect to various offered loads, including normal and faulty conditions. Section 6.5.2 shows the effect of different sized buffer pools on the performance of GATENET, while section 6.5.3 discusses the impact of different threshold values on the system. The effect of using different MAXINDATA values with respect to a fixed TOTALBUF is presented in Section 6.5.4. Next, Section 6.5.5 shows GATENET's performance under different window sizes. A summary is given in Section 6.5.6.

During each simulation run, many statistics were gathered, but only the indices described in Section 6.4 will be discussed in the following experiments. Samples collected for both data and voice traffic statistics are in the order of five; thus, the results of the simulation should be sufficiently accurate.

6.5.1 The Effect of Increasing the Offered Load

In this experiment, we studied GATENET's behavior under various offered loads. Cases of normal and faulty conditions are addressed. The parameters used are as follows: WINDOW = 5, THRESHOLD = 0.75, TOTALBUF = 440, MAXVOICE = 180 and MAXINDATA = 40. The values used for the offered load range from 875 to 14,000 packets per second. One thing to be noted here is that the offered load refers to all of the load applied to the GATENET, and it is assumed that each gateway has an equal share of the total offered load.

6.5.1.1 Normal conditions

As shown in Figure 20, GATENET's throughput grows as the offered load increases; but as the offered load goes beyond a certain amount, the throughput gradually saturates and maintains a rather steady level thereafter. This behavior conforms to the requirements of an ideal flow and control mechanism which is applied to prevent throughput degradation from occurring as a result of overloading.

Figure 21 shows the average transit delay of the voice traffic, classified according to the number of hops traversed during the internet transmission. It shows that the voice transit delay fluctuates within a rather small range and is insensitive to the increase of the offered load

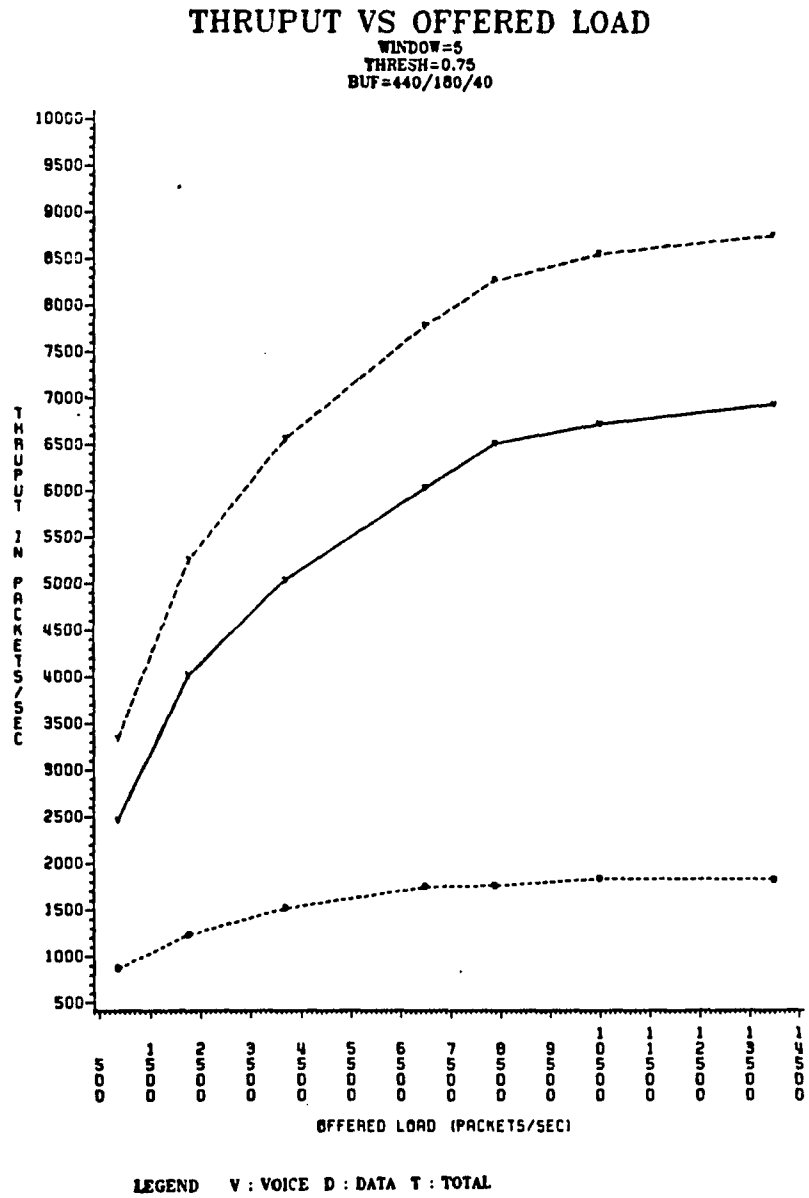


Figure 20. GATENET throughput vs the offered load (normal conditions)

once the offered load exceeds a certain limit (Standard deviation of the voice transit delay is shown in Table 1.). As discussed in previous chapters, minimization of the voice transit delay and its associated variance are the most critical performance requirements for the support of voice communication in a packet switched system. The results shown here support our claim that the congestion and control measures proposed in Section 4.2.3 can very effectively handle the GATENET congestion conditions.

Table 2 shows the transit delay of the control traffic, and Figure 22 shows the transit delay of the data traffic. Due to the lowest processing priority assigned to the data traffic, the data transit delay increases as a result of the increased offered load; however, since the delay increases at such a relatively slow rate, it should be able to fulfill most of the data communication requirements.

Because of the flow and congestion control, most of the excessive incoming data packets are rejected at the entry gateways. As a result, the probability of discarding the transit data packets at busy gateways is reduced, and waste of internet resources is also avoided. Figure 23 shows the transit data blocking probability, which remains at an extremely low and stable level irrespective of the amount of the offered load. This is also a good indication of the effectiveness of our proposed flow and congestion control mechanisms.

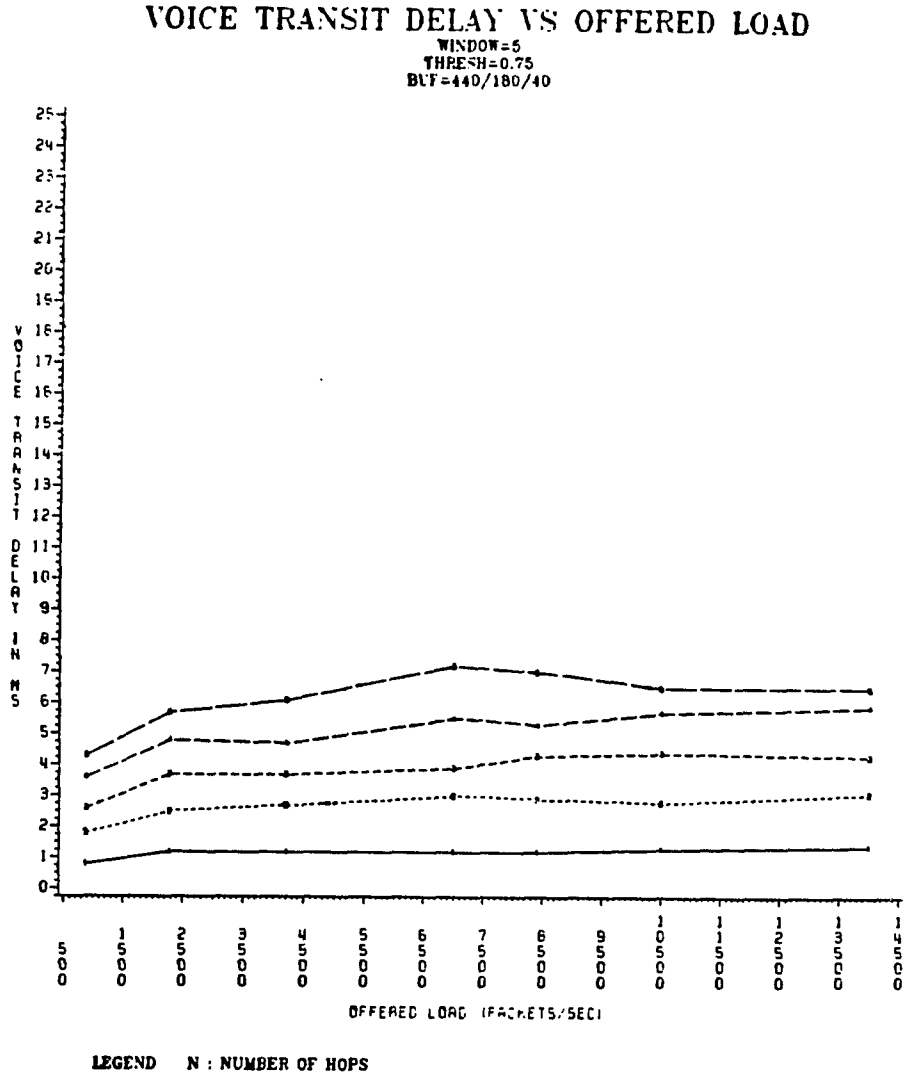


Figure 21. GATENET voice transit delay vs the offered load (normal conditions)

Table 1. Standard deviation of voice transit delay under various offered load (normal conditions)

Load	1 hop	2 hops	3 hops	4 hops	5 hops
875	0.5	0.7	0.8	1.1	0.7
2275	0.7	1.0	1.1	1.3	1.6
4200	0.8	1.1	1.1	1.2	1.3
7000	0.8	1.2	1.2	1.3	1.7
8400	0.8	1.1	1.5	1.3	1.3
10500	0.8	1.0	1.3	1.4	1.3
14000	0.8	1.2	1.3	1.5	1.9

Table 2. GATENET control transit delay vs the offered load (normal conditions)

Load	1 hop	2 hops	3 hops	4 hops	5 hops
875	2	4	5	6	7
2275	11	17	16	18	17
4200	17	24	19	25	26
7000	23	32	25	37	25
8400	27	31	26	33	31
10500	27	38	32	31	29
14000	32	36	37	49	33

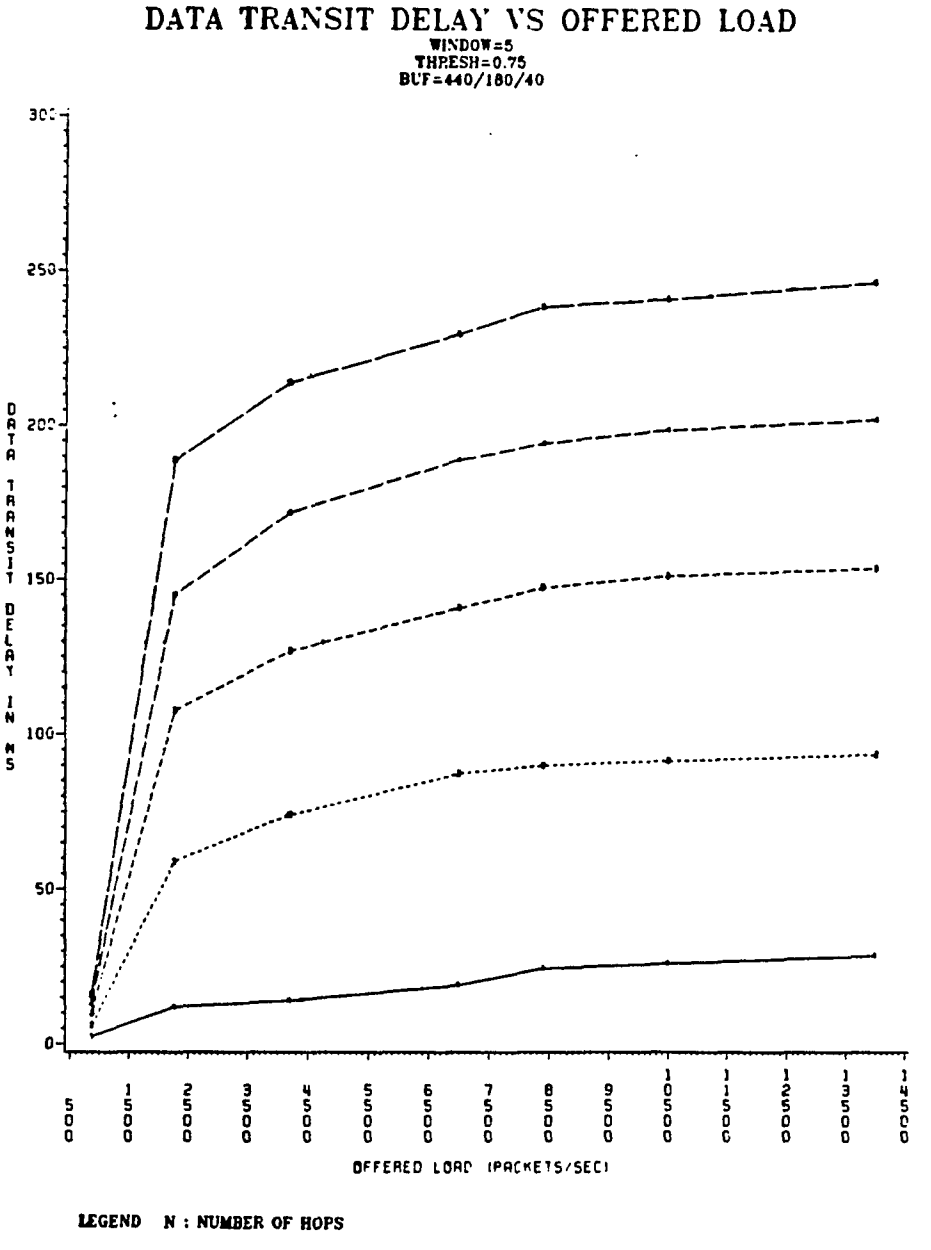


Figure 22. GATENET data transit delay vs the offered load (normal conditions)

Blocking probabilities for incoming data packets and incoming voice calls are shown in Figures 24 and 25, respectively.

6.5.1.2 Faulty conditions

As discussed in Chapter 5, with properly installed buddy links, the integrity of internet communication can still be maintained even when there are link failures. This section describes GATENET's performance under various link failure conditions.

1. Link A fails. At light traffic, GATENET's throughput appears to not be affected by the failure of Link A (see Figure 26), since the buddy link can provide appropriate alternate routing for those packets that should have travelled through Link A. However, as the offered load continues to grow, the adverse effect of losing Link A becomes more apparent, resulting in lower data and voice throughput, higher data and voice transit delay, higher incoming data and incoming call blocking probability (see Figures 26-30). However, the transit data blocking probability is lower than that of normal conditions (see Figure 31). Because of the higher incoming data and incoming call blocking probabilities, less traffic is admitted to the GATENET. As a result, the probability of discarding the transit data packets will be relatively lower when compared with that of normal conditions.

2. Link Y fails. Because Link Y is directly connected to the root gateway, the throughput (especially the voice throughput)

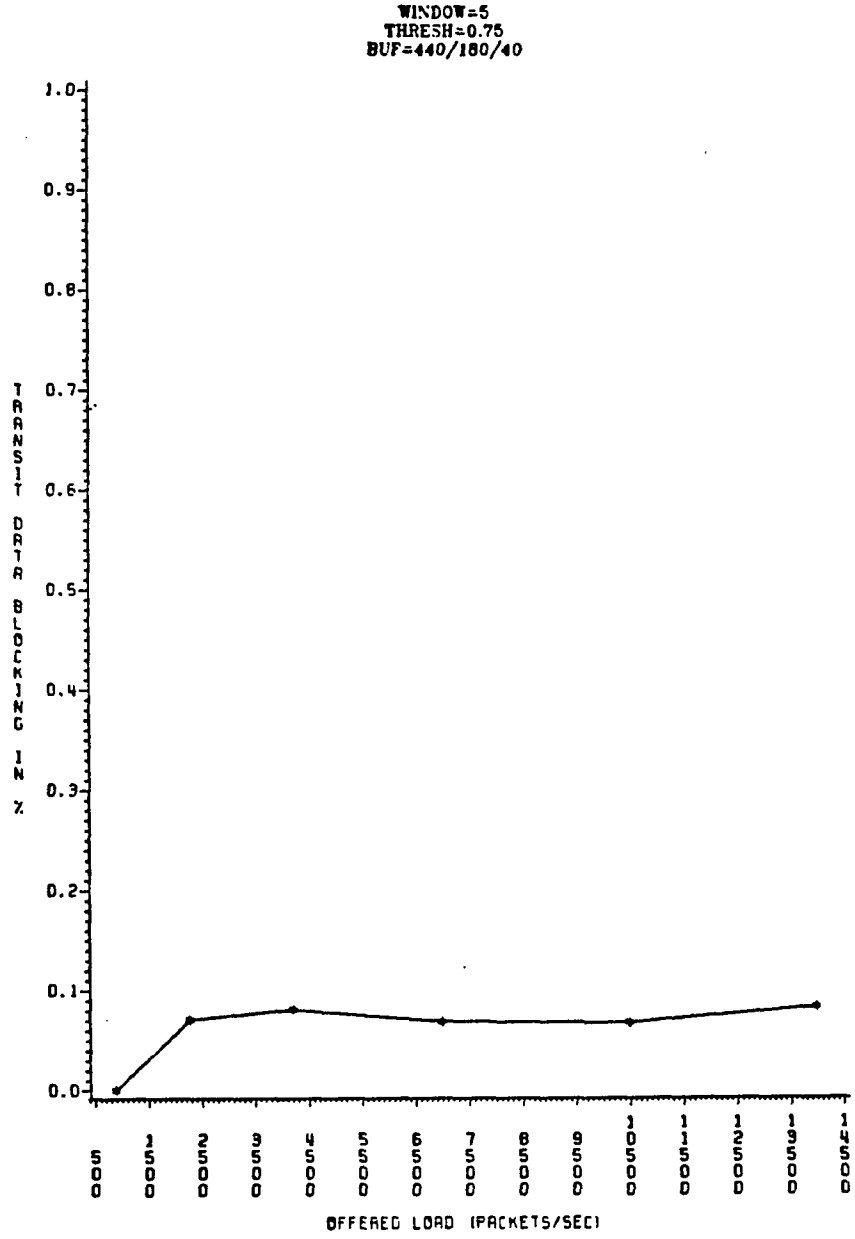


Figure 23. GATENET transit data blocking vs the offered load (normal conditions)

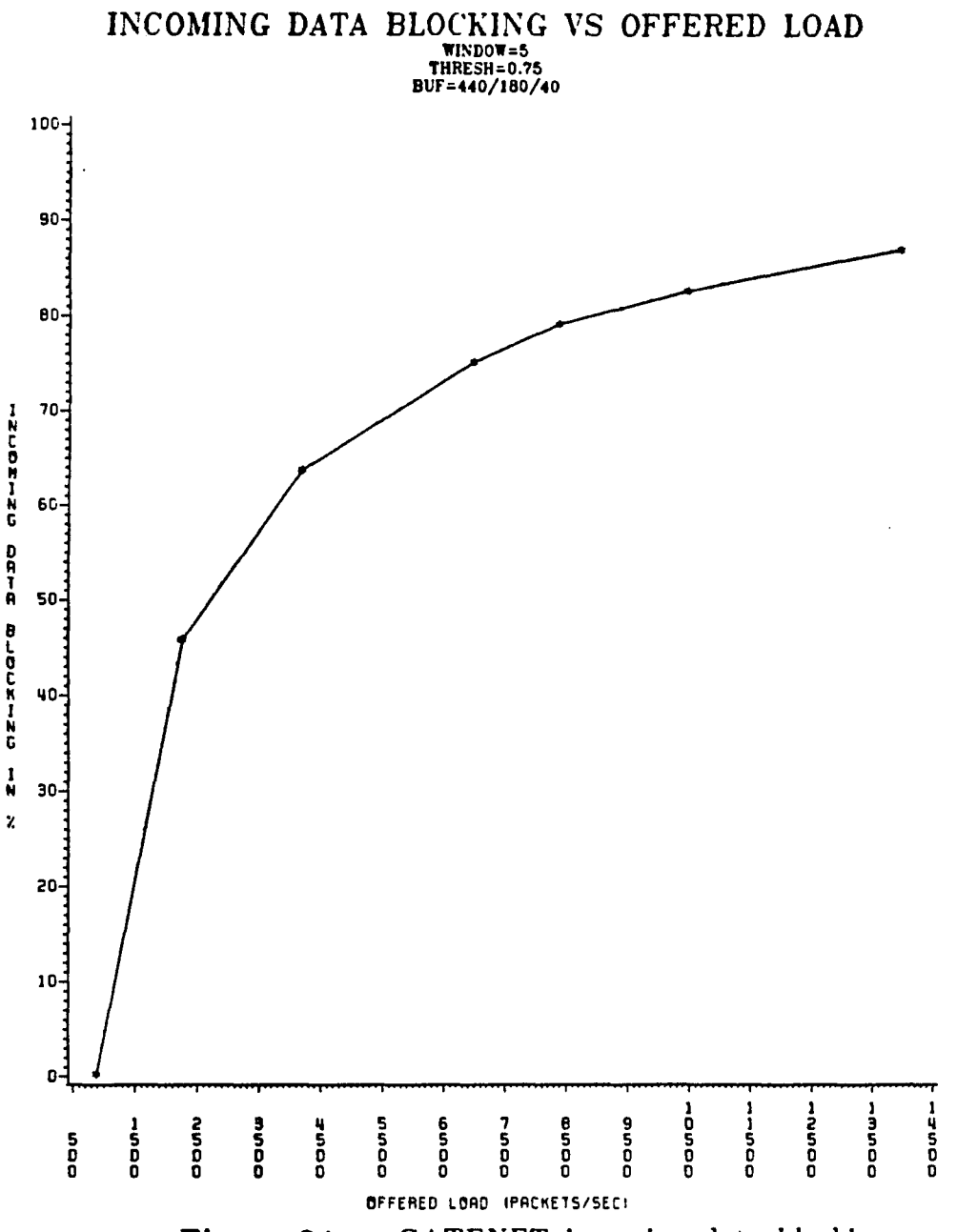


Figure 24. GATENET incoming data blocking vs the offered load (normal conditions)

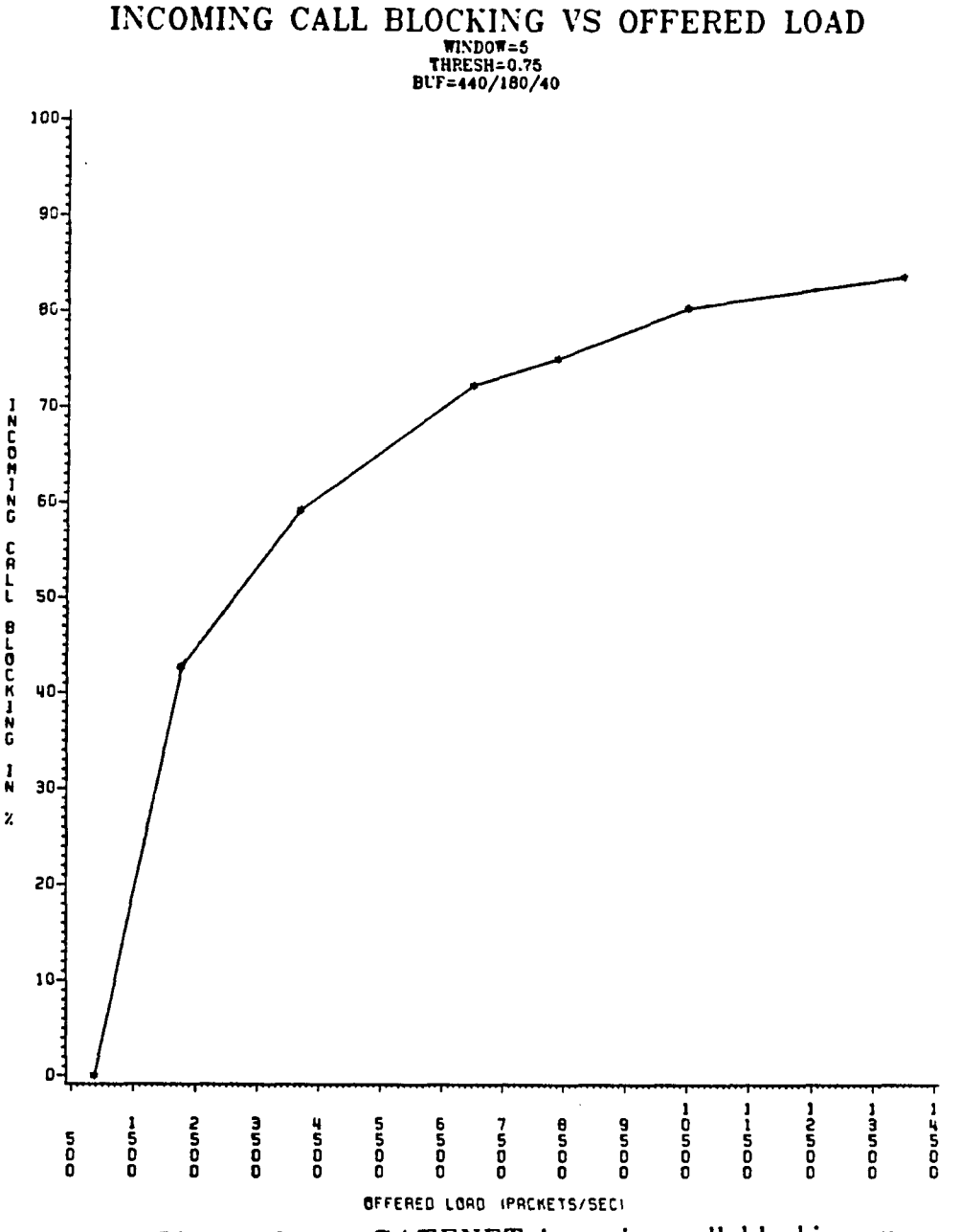


Figure 25. GATENET incoming call blocking vs the offered load (normal conditions)

in the previous case (see Figure 26). Because of the dramatic decrease in the amount of voice traffic, the data throughput increases, and the incoming data blocking probability is lower (see Figure 29). Also, because of significantly reduced traffic in the GATENET, both the data and voice transit delays are lower than those under normal conditions (see Figures 27 and 28). Further, due to the large amount of traffic that needs to be rerouted via the buddy link, the transit data blocking probability is much higher (see Figure 31).

6.5.2 The Effect of Increasing the Size of the Buffer Pool

This experiment studied GATENET's performance with respect to different sizes of the buffer pool. The parameters used in this experiment are as follows: WINDOW = 5, THRESHOLD = 0.75, MAXVOICE = 180 and the offered load = 1575 packets per second. The values used for TOTALBUF range from 260 to 560, and MAXINDATA is assumed to be 15% of (TOTALBUF-MAXVOICE).

As the size of the buffer pool increases, more incoming data packets will be admitted to the GATENET, thus leading to the increase of the data throughput (see Figure 32). However, the rate of throughput increase quickly slows down as the pool size exceeds a certain value. The voice throughput is not affected by this change. This is in accordance with the GATENET design, which gives voice

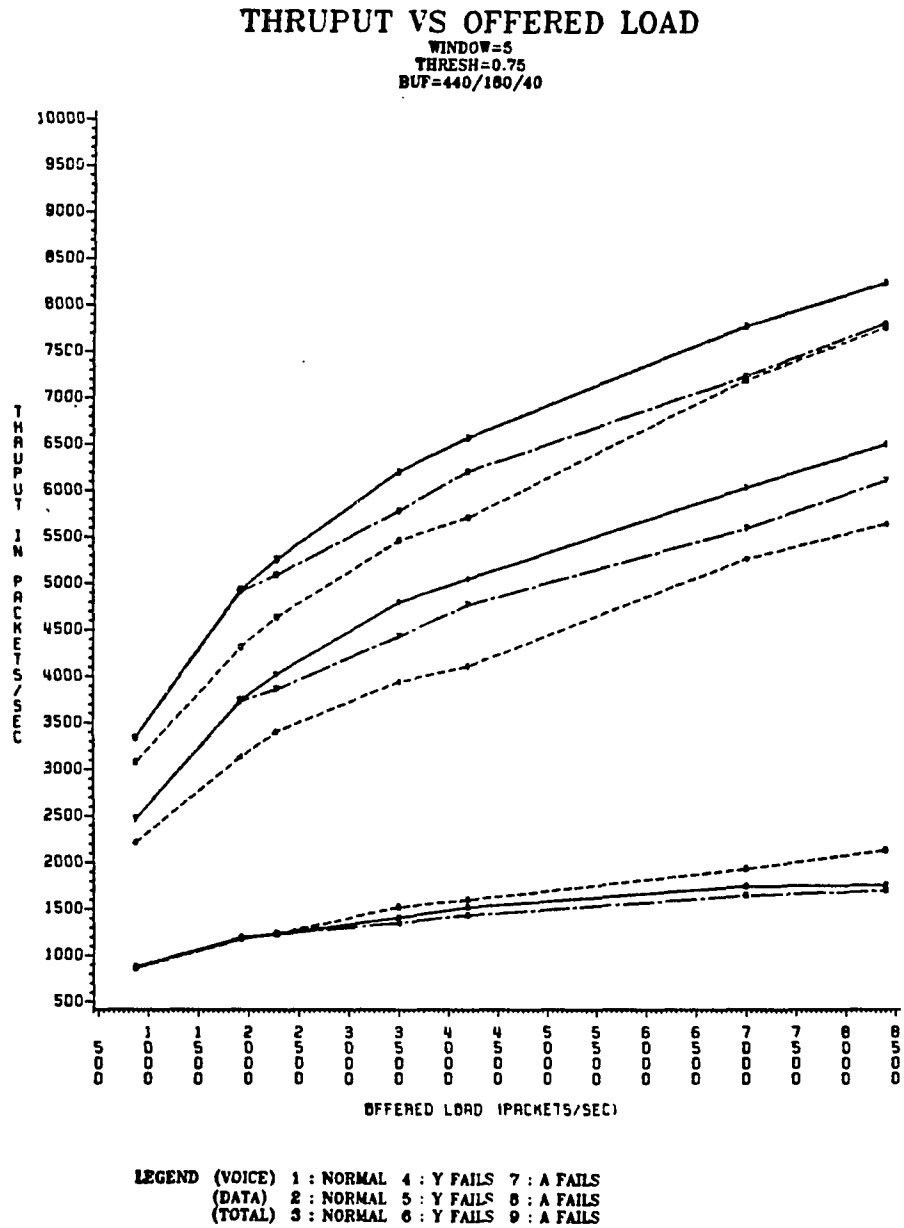


Figure 26. GATENET throughput vs the offered load (faulty conditions)

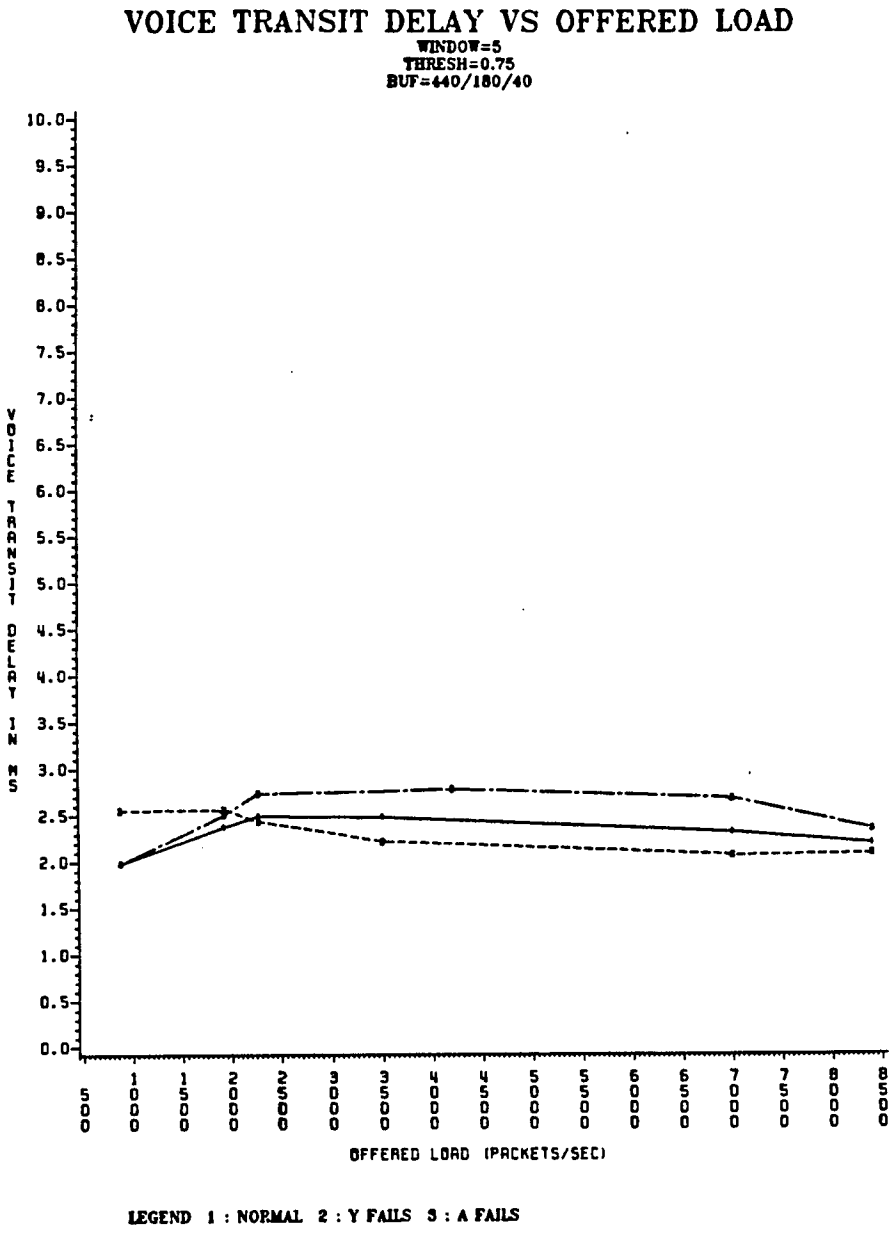


Figure 27. GATENET voice transit delay vs the offered load (faulty conditions)

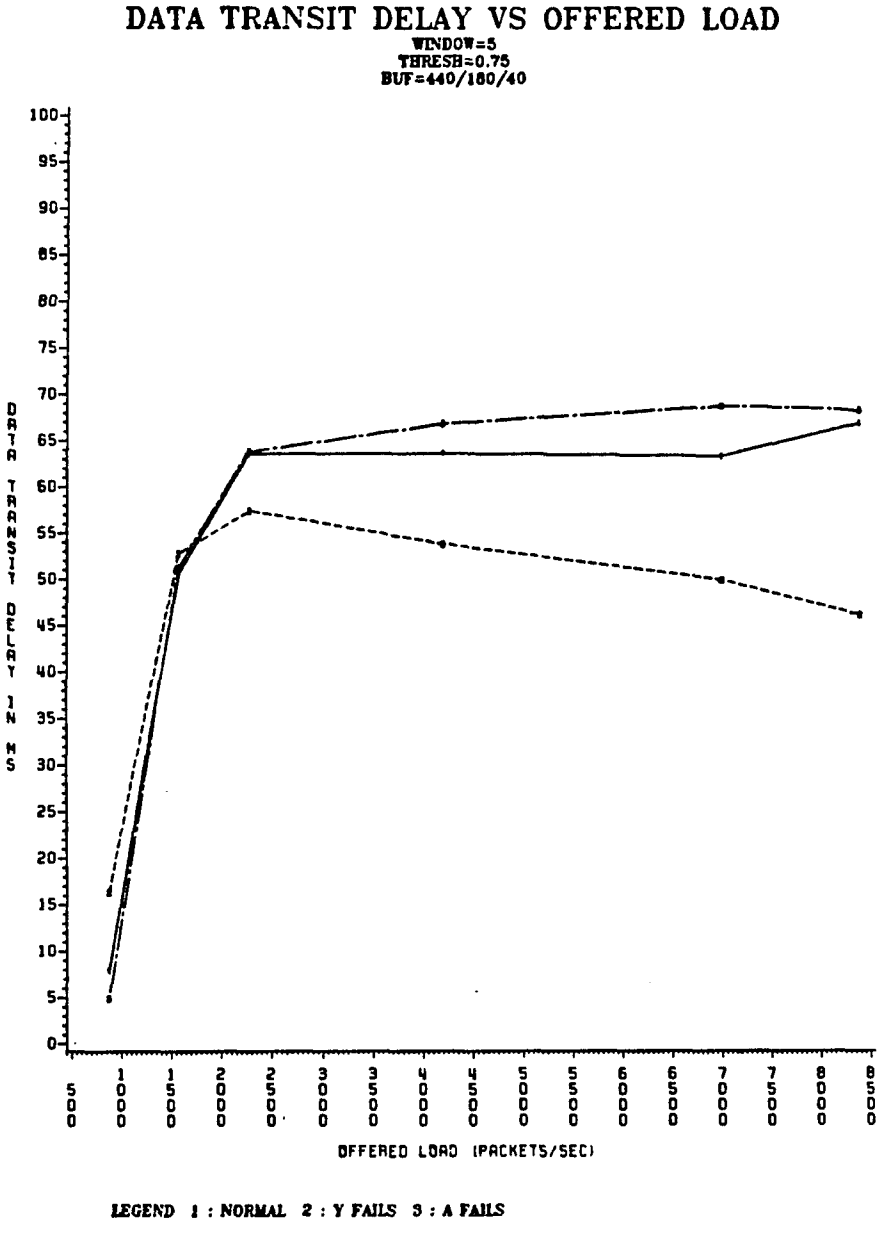


Figure 28. GATENET data transit delay vs the offered load (faulty conditions)

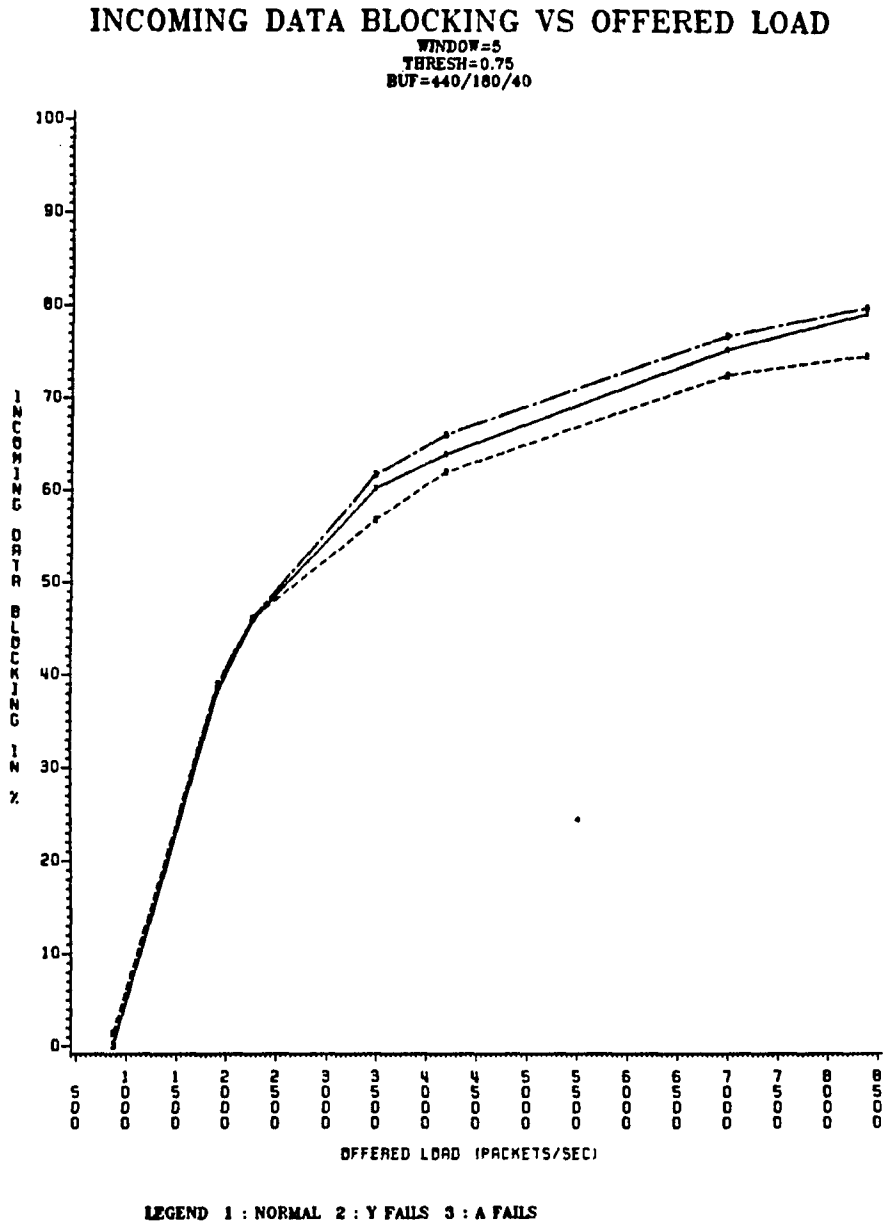


Figure 29. GATENET incoming data blocking vs the offered load (faulty conditions)

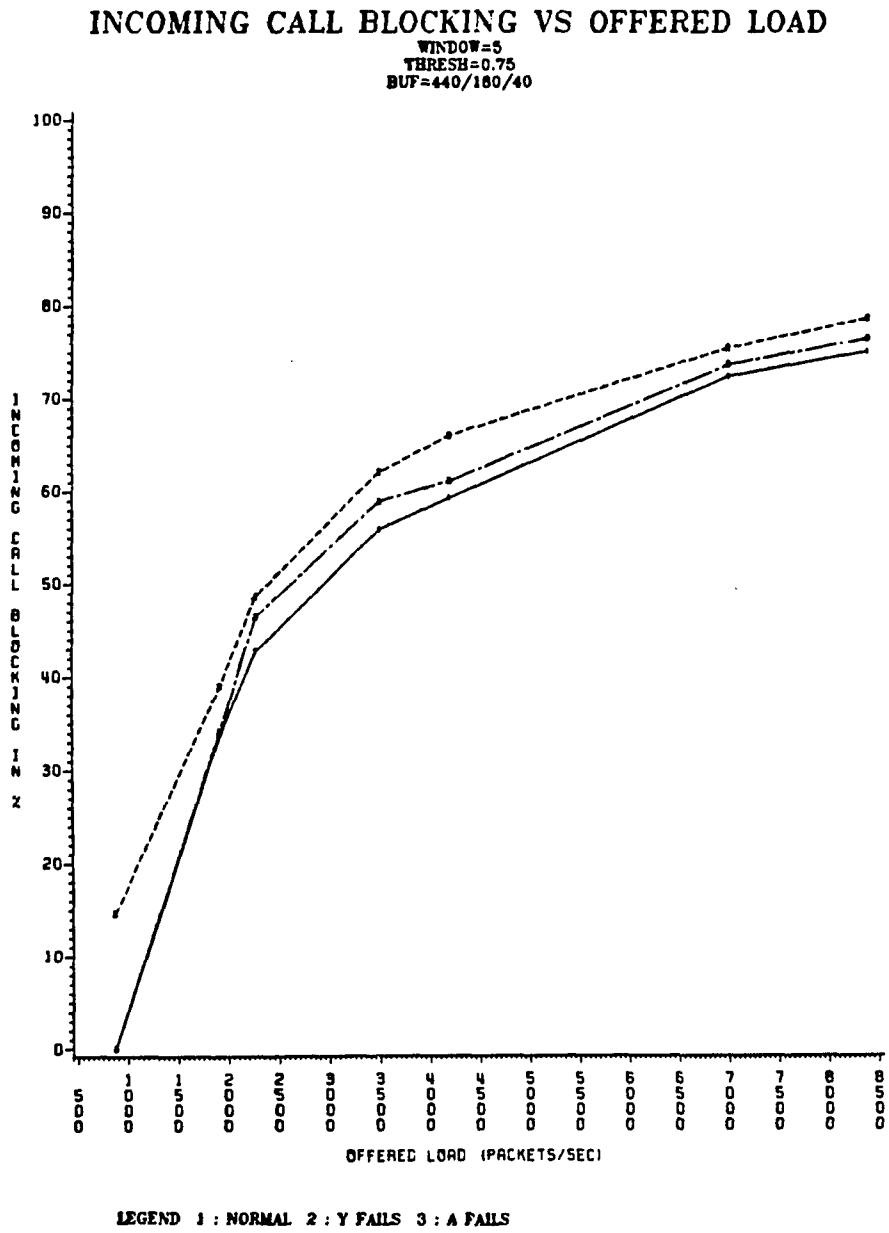


Figure 30. GATENET incoming call blocking vs the offered load (faulty conditions

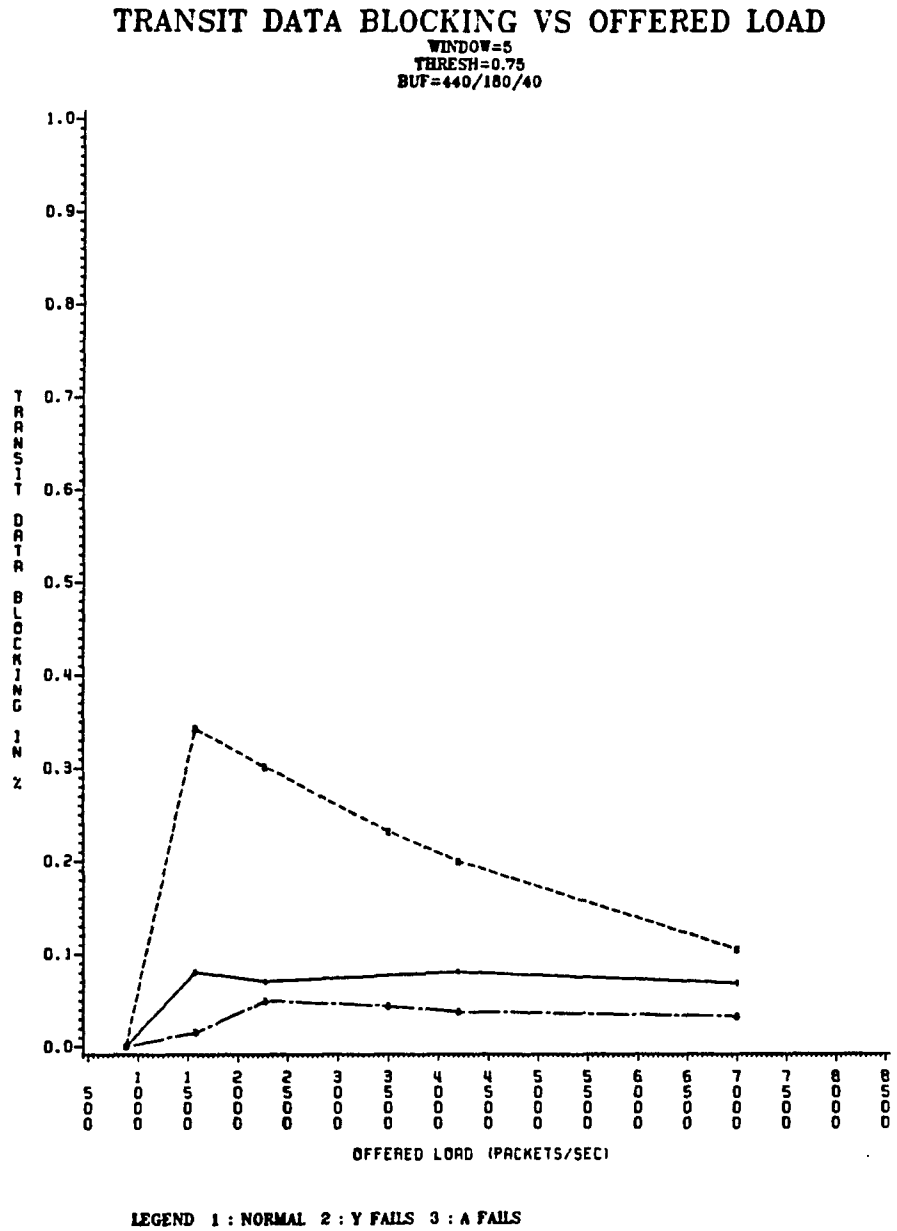


Figure 31. GATENET transit data blocking vs the offered load (faulty conditions)

traffic the highest processing priority and subjects it to an upper bound of buffer utilization at any given time.

Figure 33 shows that the average data transit delay increases as the size of the buffer pool becomes larger. This situation is also as expected, since the larger the buffer size, the more incoming data packets will be admitted into the GATENET. However, the processing speed at the gateways remains the same. Hence, on the average, more queueing time is needed to service each data packet.

Observations from Figures 32 and 33 suggest that once the size of the buffer pool has grown to a certain value increasing the buffer size is no longer a good strategy to improve GATENET's performance. This is because such a move produces only a small gain in throughput and significantly increases the data transit delay.

Figures 34 and 35 show the respective blocking probabilities for the transit data and incoming data traffic.

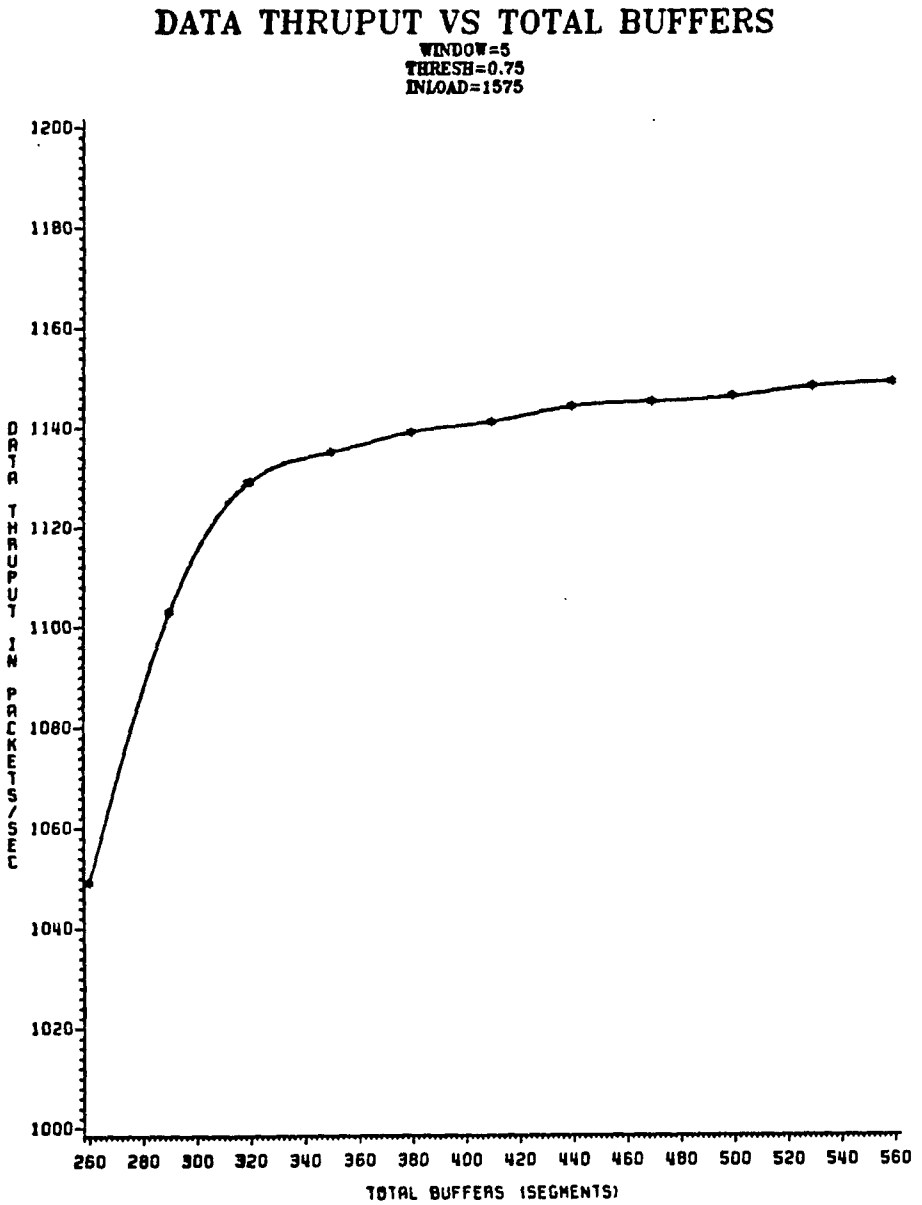


Figure 32. GATENET throughput vs size of the buffer pool

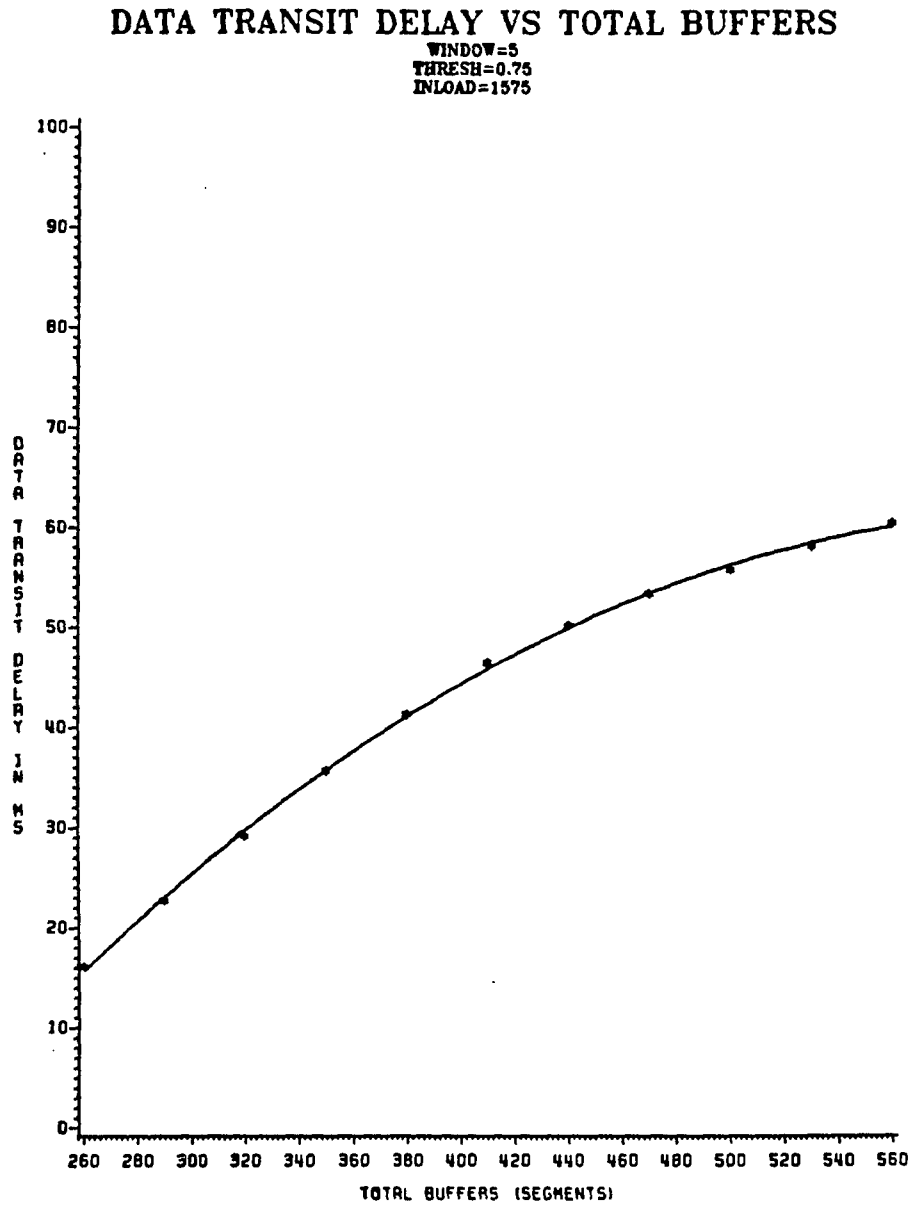


Figure 33. GATENET data transit delay vs size of the buffer pool

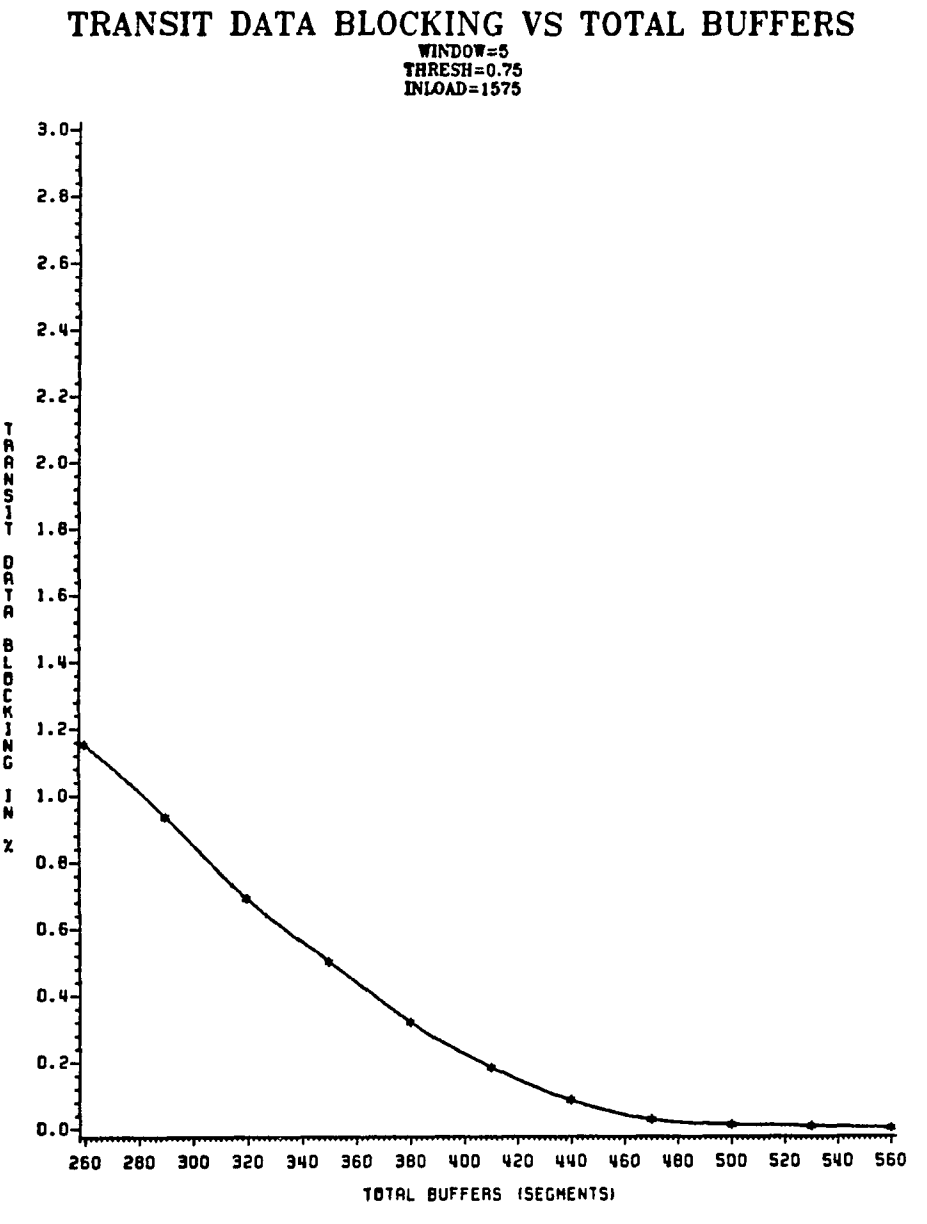


Figure 34. GATENET transit data blocking vs size of the buffer pool

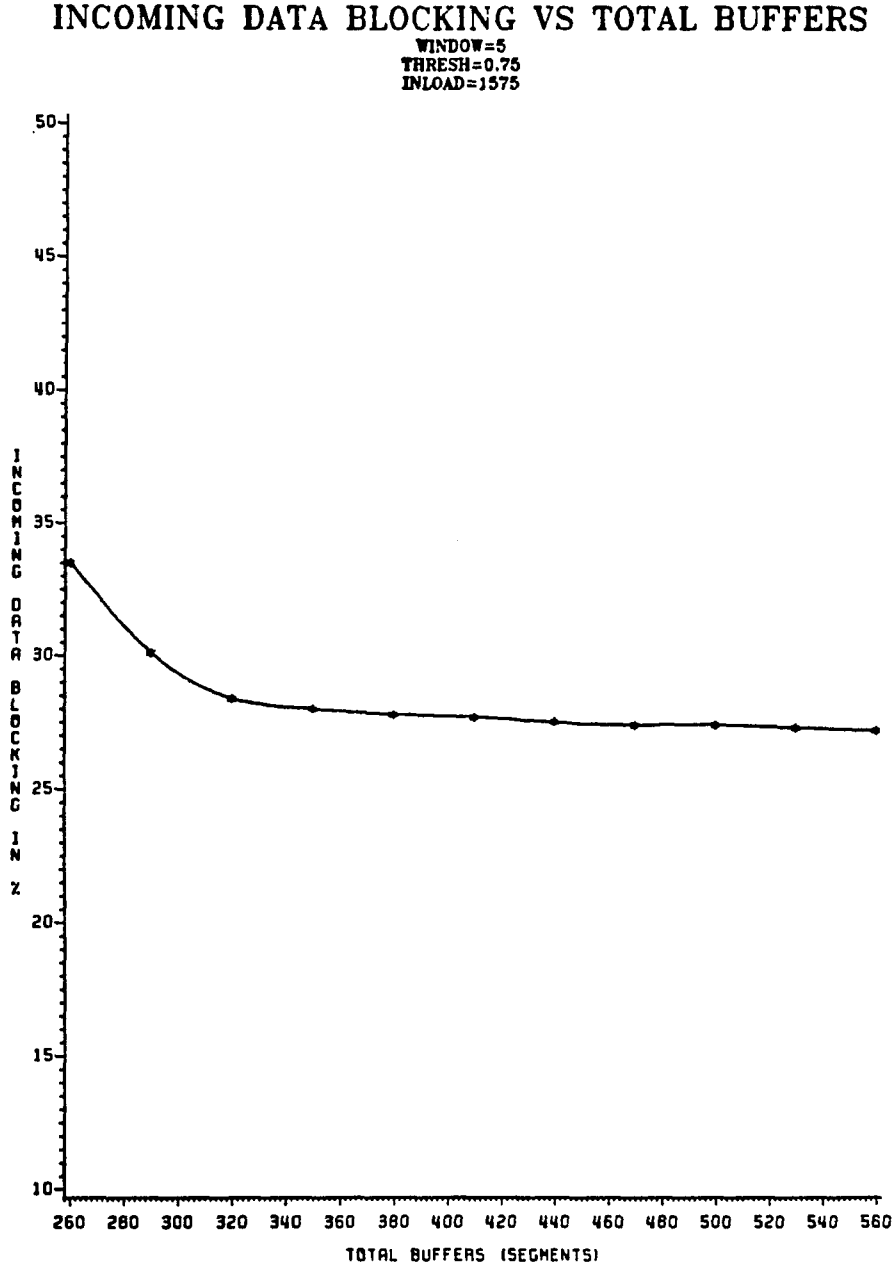


Figure 35. GATENET incoming data blocking vs size of the buffer pool

6.5.3 The Effect of Increasing the Threshold Value

In this experiment, the effect of increasing the threshold value on GATENET's performance was studied. The parameters used in the experiment are as follows: WINDOW = 5, TOTALBUF = 440, MAXVOICE = 180, MAXINDATA = 40 and the offered load = 1575 packets per second. The values used for THRESHOLD range from 0.60 to 1.00.

Figure 36 shows that the data throughput increases slightly as the threshold value becomes larger. This is due to the fact that a larger threshold value allows more incoming data packets to be admitted to the GATENET. The voice throughput is not affected when the threshold value is changed.

Figure 37 shows the data transit delay. Since more incoming data traffic will be admitted into the GATENET as a result of the increased threshold value, the data transit delay will increase accordingly. But as the threshold value approaches a certain limit, the transit delay decreases rapidly. This situation may look strange at first glance, but it, in fact, highlights a serious problem which often appears in a congested network. Due to the lack of proper congestion control, when congestion occurs, only the data packets requiring fewer hops can be delivered to their destination hosts, and the data packets requiring more

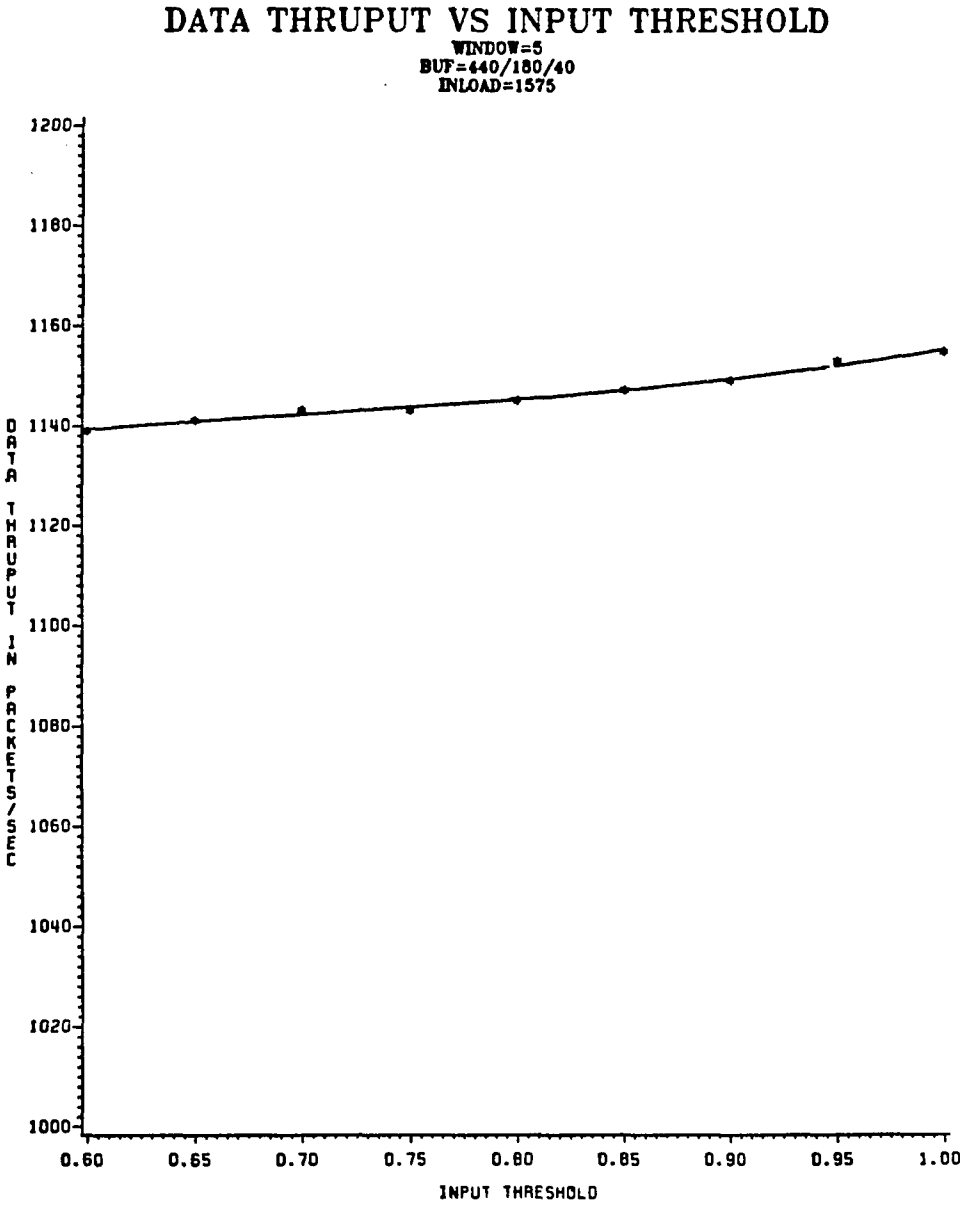


Figure 36. GATENET throughput vs the threshold value of input buffer limit

hops to get to their destination are more likely to be discarded en route. As a result, the average transit data delay decreases as the network congestion is aggravated. Figure 38 shows the blocking probability of the transit data traffic.

Observations from this experiment show that the data throughput can be slightly increased if the threshold value is set higher. However, such a threshold value should be carefully chosen so that it will not go beyond a limit at which the data transit delay begins to deteriorate.

6.5.4 The Effect of Increasing the Window Size

In this experiment, we studied the effect on GATENET's performance of increased window size. The parameters used in this experiment are as follows: THRESHOLD = 0.75, TOTALBUF = 440, MAXVOICE = 180, MAXINDATA = 40 and the offered load = 1575 packets per second. The values used for the window size range from one to 12.

Initially, GATENET's throughput increases as the window size becomes larger (see Figure 39). However, after the window size reaches a certain value, the throughput begins to decline. The explanation for this behavior is the fact that more incoming data traffic (than the amount the GATENET can handle) will be admitted to GATENET as a

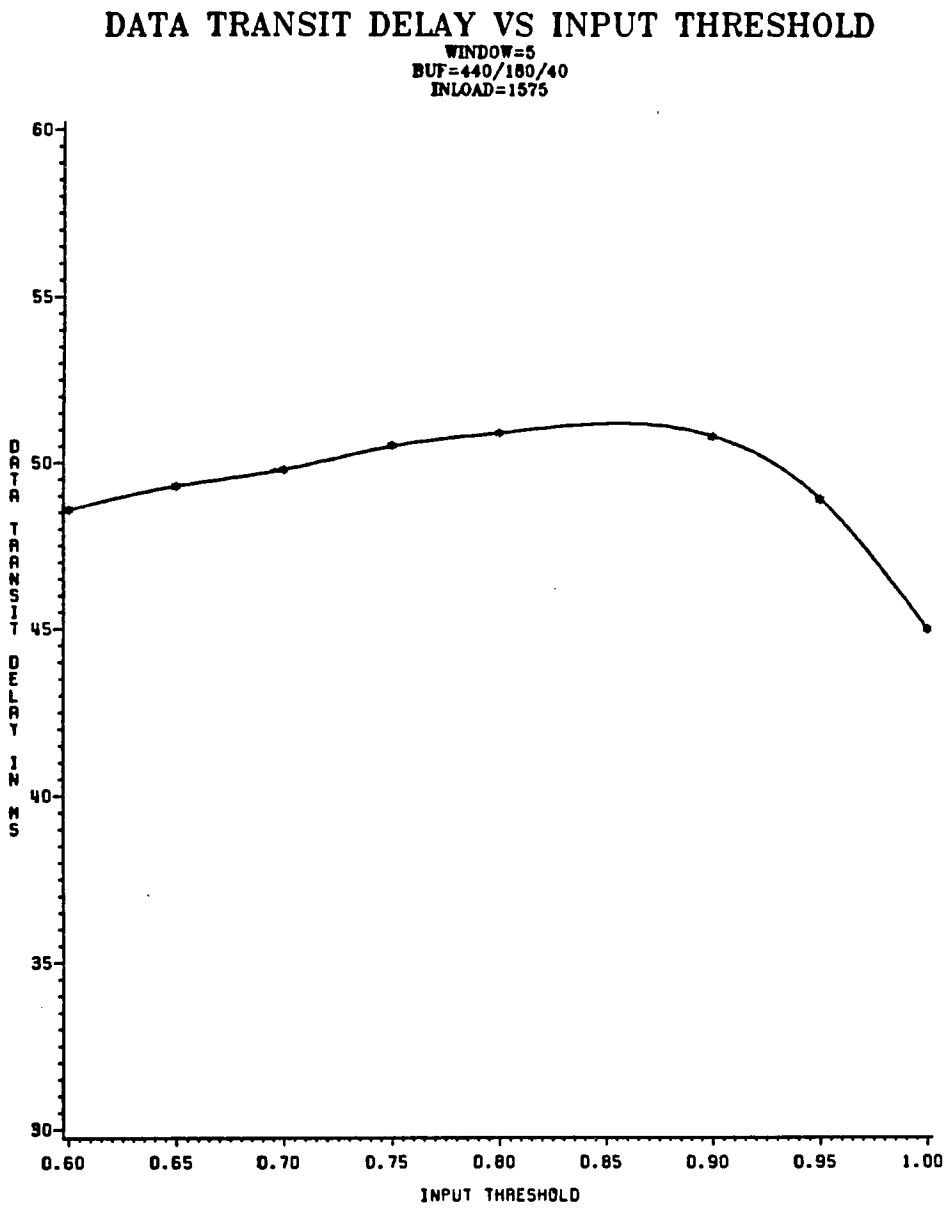


Figure 37. GATENET transit data delay vs the threshold value of input buffer limit

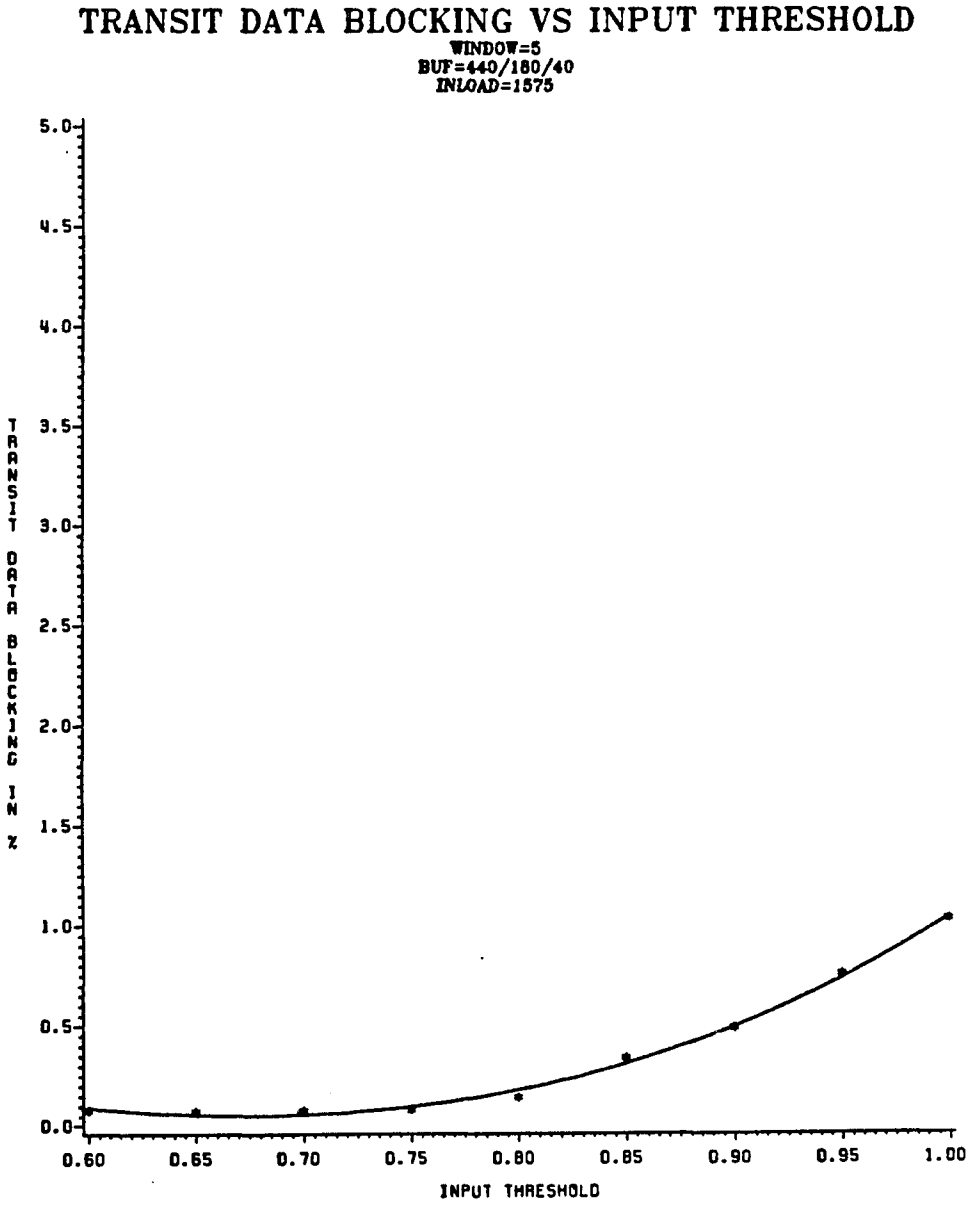


Figure 38. GATENET transit data blocking vs the threshold value of input buffer limit

result of the larger window size. The larger window subsequently leads to significantly higher blocking probability for the transit data traffic (see Figure 40).

The data transit delay increases as the window size becomes larger (see Figure 41) because the larger size allows more incoming data traffic to be admitted to GATENET. Thus, on the average, more queueing time is needed for each data packet to be serviced. Figure 42 shows the blocking probability of the incoming data traffic.

6.5.5 The Effect of Increasing Incoming Data Limit

This experiment studied the impact on GATENET's performance of increased incoming data limit. The parameters used in this experiment are as follows: WINDOW = 5, THRESHOLD = 0.75, TOTALBUF = 440, MAXVOICE = 180 and the offered load = 1575 packets per second. The values used for MAXINDATA range from 0 to 260.

As shown in Figure 43, GATENET's throughput grows slightly as the value of MAXINDATA increases. The voice throughput is not affected by this change until MAXINDATA increases to a value that will affect the buffer utilization for voice traffic. Such a case is not encountered with the range chosen for this experiment.

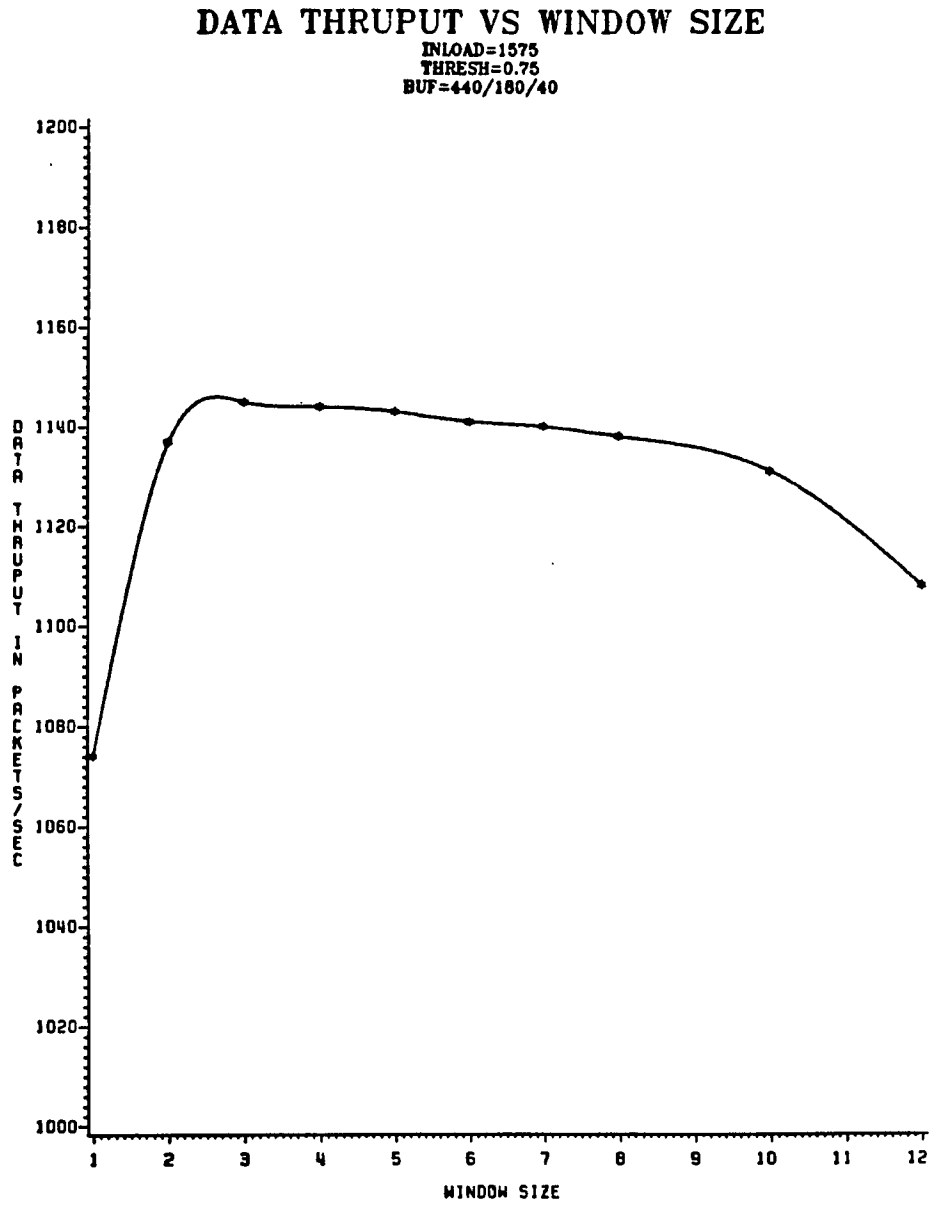


Figure 39. GATENET throughput vs the window size

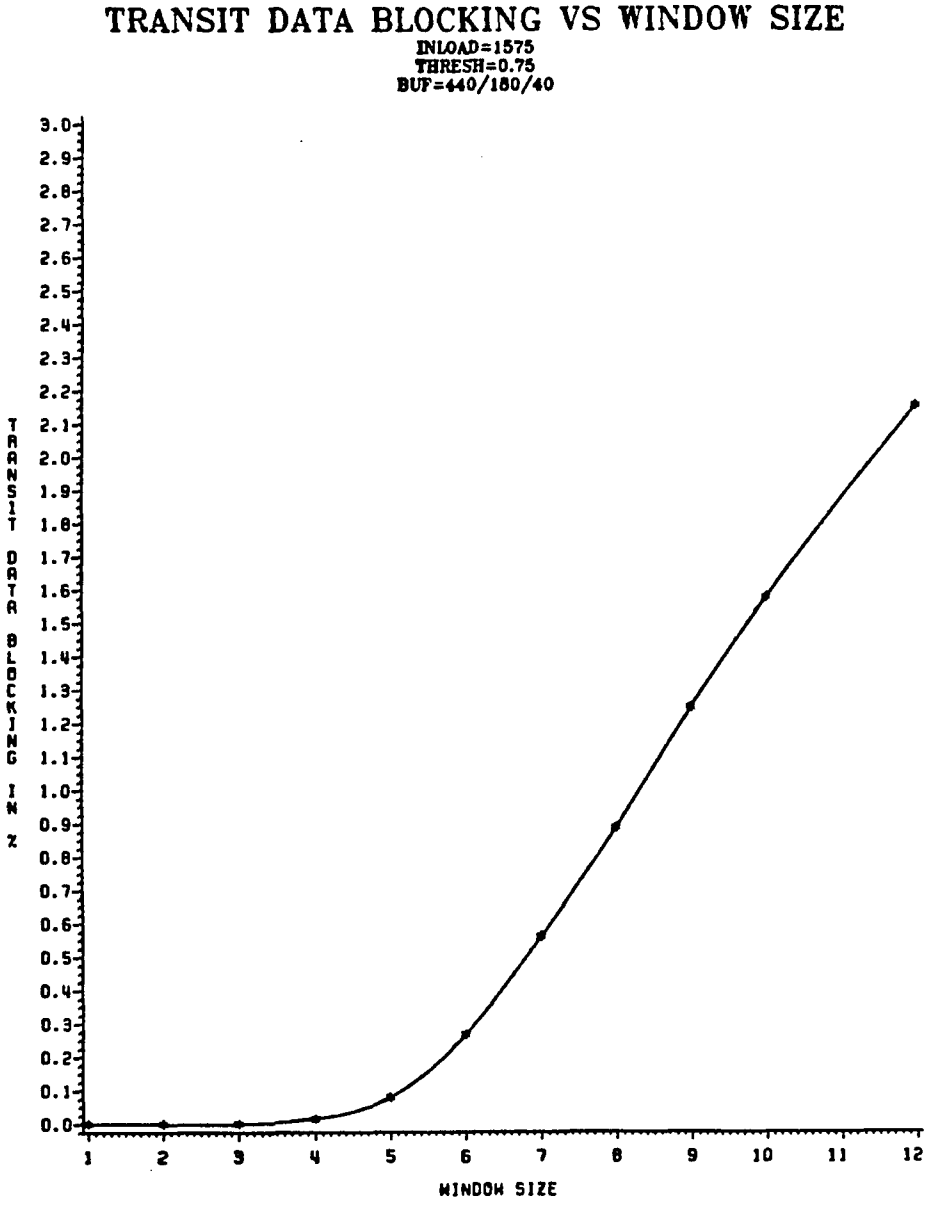


Figure 40. GATENET transit data blocking vs the window size

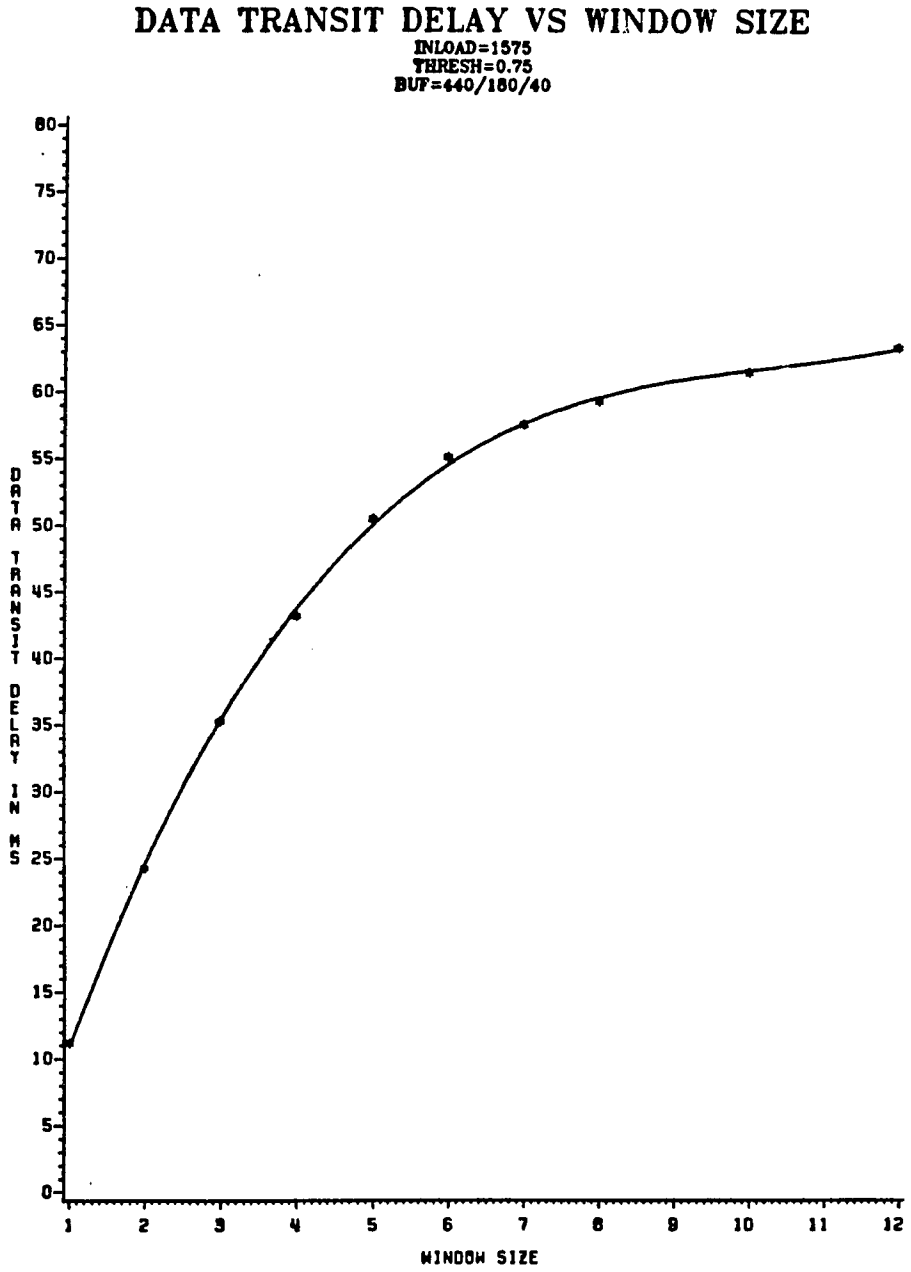


Figure 41. GATENET data transit delay vs the window size

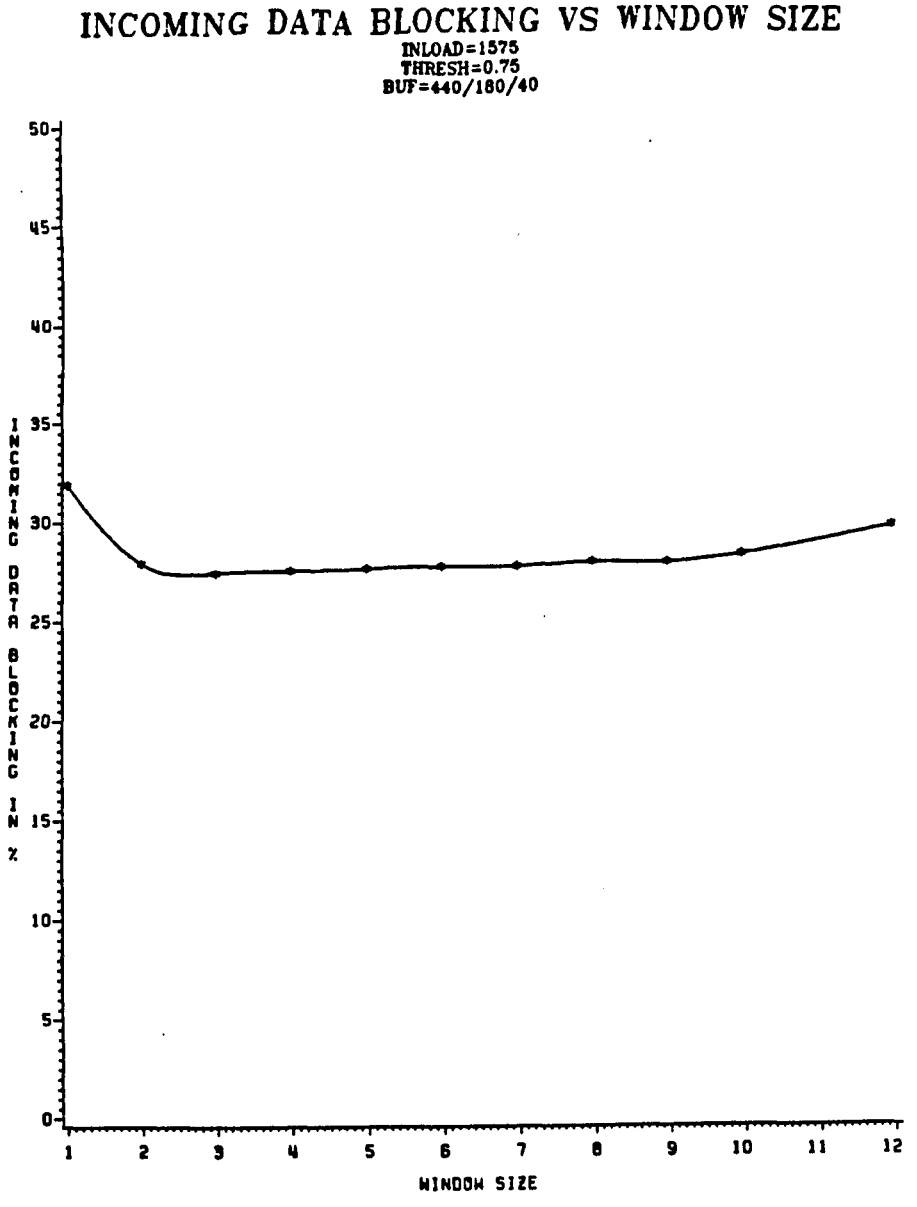


Figure 42. GATENET incoming data blocking vs the window size

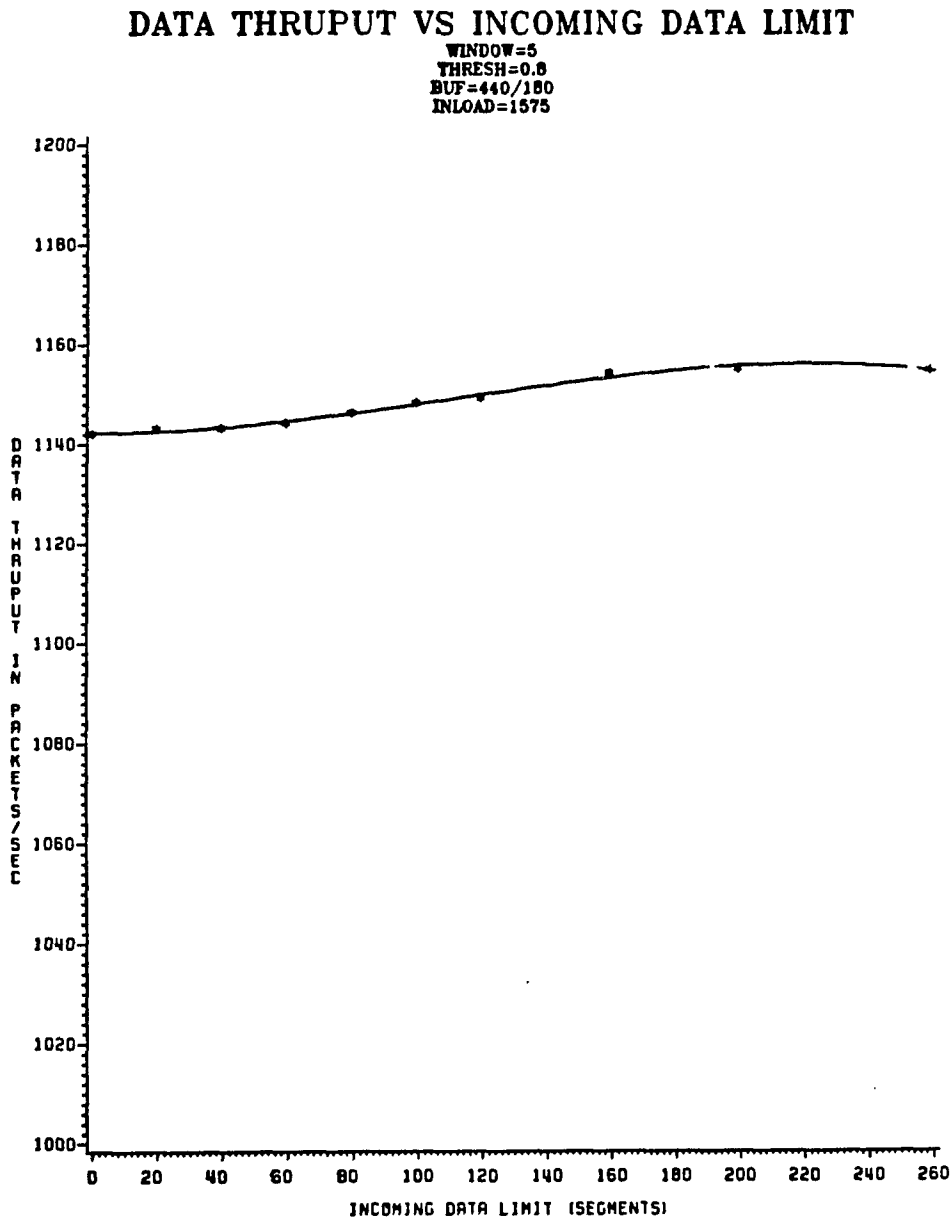


Figure 43. GATENET throughput vs incoming data limit

Figure 44 shows the behavior of data transit delay. Initially, as MAXINDATA increases, more incoming data packets are admitted to GATENET, which subsequently increases the average data transit delay. But as MAXINDATA continues to increase, even more incoming data traffic is admitted to GATENET. Consequently, the buffer segments originally available for the transit data traffic are depleted, leading to a significantly higher transit data blocking probability (see Figure 45). As a result, only the data packets with fewer hops can be delivered to their destinations, and thus the average transit delay becomes smaller.

Tuning MAXINDATA is very closely related to the setting of the threshold value. Depending on the setting of the threshold value, MAXINDATA can be tuned to maximize the system performance. However, such tuning should be carefully performed so that satisfactory performance can be constantly maintained. One possible usage of MAXINDATA is to allow a certain portion of the system resources accessible at any given time to incoming new traffic. By enforcing the incoming data limit, during the heavy traffic period the new data traffic will still have a fair chance of entering the GATENET instead of being blocked out as most flow control mechanisms usually do.

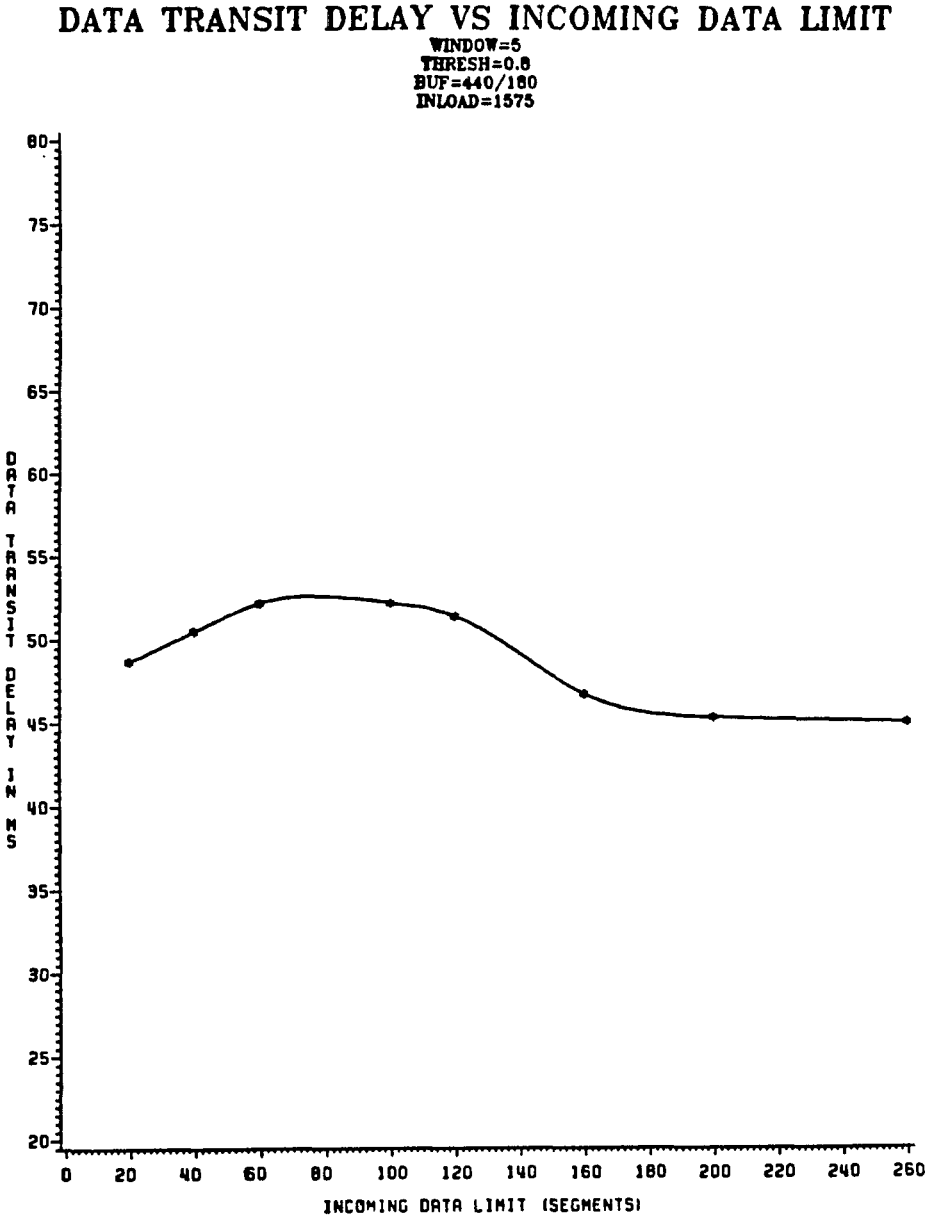


Figure 44. GATENET data transit delay vs incoming data limit

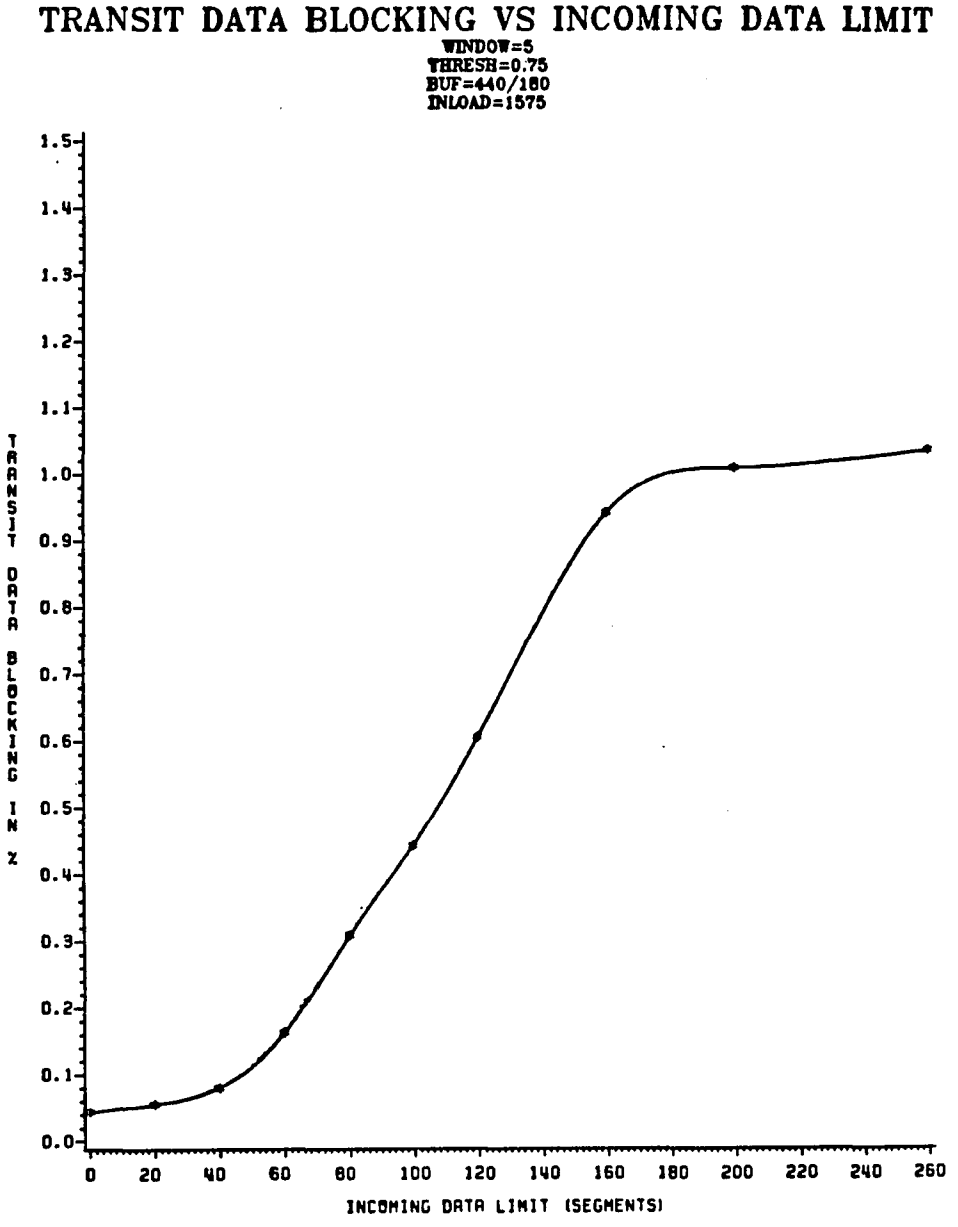


Figure 45. GATENET transit data blocking vs incoming data limit

6.6 Summary

In this chapter, we have described a GATENET simulation model. Furthermore, simulation results with respect to various operating conditions have been presented and discussed. All the results support the GATENET behavior predicted by our proposed flow and congestion control mechanisms. Thus, we conclude that the GATENET design is a feasible and cost-effective method to achieve satisfactory voice and data communication services in a campus-wide area.

CHAPTER VII
SUMMARY AND CONCLUSIONS

This chapter summarizes the main research results presented in the previous chapters. Possible directions for future research are also identified.

7.1 Summary

The objective of this research has been to design an internet transport system within a campus-wide area so that voice and data communication services can be achieved in a cost-effective and elegant way.

Driven by the recent rapid advances in computing and communication technologies and the growing demands of integrated data, voice, facsimile and video services in an office environment, network

interconnection is receiving more and more recognition as a necessary element to meet such demands in the future. Existing internets are generally constructed out of the natural growth of demands from the user community. Consequently most internets are oriented toward fulfilling the immediate application requirements. While this approach can provide a quick solution, it inevitably leaves many constraints on the ability of the internet system to adapt to evolving technologies.

In this dissertation, a new network interconnection strategy, called GATENET, is presented as an effective means to achieve satisfactory voice and data communication services. In Chapter 3, GATENET is presented from an architectural point of view. Based upon the hierarchical structure of GATENET, the addressing and routing schemes are then defined to support the internet transport functions. Advantages and disadvantages of GATENET compared with the conventional gateway internetworking approaches are also discussed.

Communication protocols are the kernel of a communication system. Efficiency of protocols can often greatly enhance or degrade the internet performance. Since the majority of data and voice traffic within a campus-wide area is for inter-office communication, special data transport and voice transport protocols have been developed in Chapter 4 to meet such communication needs. In an internet environment, with

many users engaging in various types of applications, it is likely that, from time to time, instantaneous traffic loads will greatly exceed the internet capacity. To prevent serious performance degradation due to occasional overloaded conditions, flow and congestion control is called for. In GATENET, two levels of flow control mechanisms are used to regulate the internet traffic flow so that voice and data communications can proceed smoothly under various traffic conditions. Interconnecting incompatible networks is often a complicated and tedious task. In order to minimize the software development overheads, optional transport layer protocol support is also included in the GATENET design.

An internet usually consists of a wide variety of equipment and is accessed by many users every day. Therefore, the reliability of such an internet must be properly addressed to ensure that internet service interruptions are reduced to a minimum. Although a redundancy approach can be employed to improve internet reliability, it is nevertheless a rather costly solution and is often not economically justified in a campus-wide environment. Hence, Chapter 5 presents a "buddy link" scheme as a cost-effective means to improve GATENET reliability.

Performance is the most important index in designing a communication system. Every design parameter should be chosen to

improve one or more performance aspects such as delay, throughput, cost, distance, etc. At the various stages of designing a communication system, a simulation model can often provide many helpful insights into the target system. In Chapter 6, a network communication simulator, called GATESIM, is presented as a tool to study the performance aspects of GATENET. Simulation results with respect to various operating and traffic conditions are then presented.

7.2 Areas for future research

As mentioned in Chapter 1, because of rapid advances in fiber optics, traffic bottlenecks in modern communication systems have moved from transmission elements to switching elements. In order to keep pace with the extremely high data rates of fiber optical transmission media, very fast packet switching processors must be devised. The traditional store and forward approach is inadequate to cope with such voluminous, high speed incoming traffic because the approach calls for both a tremendous amount of memory to buffer the incoming packets and the use of very fast (and thus costly) memory systems to keep up with its processing speed. To handle the high speed incoming traffic, several schemes have been proposed using non-buffering and non-blocking interconnection network (IN) techniques and have produced encouraging

results. However, the fact that these schemes are too costly and complicated to be built suggests that more research work is needed in this area.

This dissertation has concentrated on the design of an internet transport system, and in Chapter 4 protocols are defined to support the internet transport functions. However, to access GATENET, an interface protocol specific to each connecting LAN must also be properly defined to synchronize the speed mismatch between the connecting LAN and the entry gateway. Further investigations in this area will enhance the GATENET design. In addition, development of high level application protocols, such as file transfer protocols, conferencing protocols and remote access protocols, are also needed to make the GATENET design even more complete.

In Chapter 6, our simulation results have shown that the GATENET design can provide satisfactory voice and data communication services under various traffic conditions. However, the design has not been tested in real environments. It would be a worthwhile endeavour to apply the GATENET design in some prototype systems. Currently, there are several existing networks at the IRCC/CIS Computing Laboratory; experimental work built on top of these networks would provide solid groundwork for such prototyping efforts.

7.3 Conclusions

During the past decade, the price-performance revolution in computing and communication technologies has transformed the world into a new information age. As the momentum of office automation continues to grow, more and more local area networks will multiply. As a result, the capability to interconnect and communicate with other networks will become an indispensable aspect of any future network design. In the meantime, after a decade's debate and study, the economic incentive and the technical advantages of the integration of voice and data traffic over the same communication system have also gained worldwide recognition, moving from the conceptual to the realization stage.

Because of this trend, it is expected that the distinction between the functionality of a computer network and a telephone network will be blurred in the years to come. In the telephone industry, the Integrated Services Digital Networks (ISDN) project, which aims to provide cost-effective end-to-end digital connectivity supporting a wide range of voice and nonvoice services, has already gone into the planning and deployment phases. It is projected that by the turn of the century such services will be available to most commercial and residential users. On the other hand, due to the abundant bandwidth made available by fiber

optics technology, many computer vendors also have undertaken efforts to enhance their network capabilities to support integrated data and voice communication services. We believe that an efficient internet providing integrated voice and data services will soon become a reality. We also believe that the GATENET approach is suitable for meeting such a challenge.

APPENDIX A.
GATESIM: A NETWORK COMMUNICATION SIMULATOR

In this appendix, a network communication simulator called GATESIM is presented. GATESIM is written in SIMSCRIPT using discrete event-driven simulation techniques. It consists of seven gateways, 12 regular links and two buddy links, with each link in a simplex mode. Although GATESIM has been used primarily to evaluate the performance of GATENET, it is, in fact, a rather general purpose network communication simulator. In GATESIM, each major network feature is defined as a separate routine; hence, with some modifications, GATESIM can easily be reconfigured to model different network topologies, routing, or flow and congestion control mechanisms.

```
''*********************************************************
''*                                                       *
''*   THIS PREAMBLE SECTION DEFINES THE FOLLOWING ITEMS:  *
''*                                                       *
''*     1. CONSTANTS, VARIABLES AND PARAMETERS            *
''*     2. PROCESSES AND EVENTS                           *
''*     3. PERMANENT AND TEMPORARY ENTITIES               *
''*     4. STATISTICAL VARIABLES                          *
''*                                                       *
''*********************************************************

PREAMBLE LAST COLUMN IS 72 ''
   NORMALLY MODE IS INTEGER
   DEFINE .VOICE.CONNECT TO MEAN 1
   DEFINE .VOICE.CONTENTS TO MEAN 2
   DEFINE .VOICE.DISCONNECT TO MEAN 3
   DEFINE .VOICE.ACCEPT TO MEAN 4
   DEFINE .VOICE.REJECT TO MEAN 5
   DEFINE .VOICE.DISCACK TO MEAN 6
   DEFINE .DATA.CONTENTS TO MEAN 7
   DEFINE .DATA.ACK TO MEAN 8

   DEFINE .IDLE TO MEAN 0
   DEFINE .BUSY TO MEAN 1
   DEFINE .SENDER TO MEAN 0
   DEFINE .RECEIVER TO MEAN 1
   DEFINE .TALKING TO MEAN 0
   DEFINE .DISCONNECTING TO MEAN 1
   DEFINE .STARTPT TO MEAN 60
   DEFINE .SEGLENG TO MEAN 72
   DEFINE .VSEG TO MEAN 4
   DEFINE .VPACKLENG TO MEAN (.VSEG*.SEGLENG)/2
   DEFINE .BUFNO TO MEAN 440
   DEFINE .UPPERIN TO MEAN 40
   DEFINE .UPPERVO TO MEAN 180
   DEFINE .YES TO MEAN 1
   DEFINE .NO TO MEAN 0
   DEFINE .ON TO MEAN 1
   DEFINE .OFF TO MEAN 0
   DEFINE .THRESHOLD TO MEAN 0.75
   DEFINE .WINDOWSIZE TO MEAN 5
```

```
DEFINE .FAIL TO MEAN 0

DEFINE .ACKTOTIME TO MEAN 0.0001
DEFINE .HOLDTOTIME TO MEAN 0.0001
DEFINE .HOLDTIME TO MEAN 0.05
DEFINE .PROCESSORTIME TO MEAN 1/1800
DEFINE .TIMELIMIT TO MEAN 180
DEFINE .TIMEOUT TO MEAN 3.5
DEFINE .TALKTIME TO MEAN 150.0
DEFINE .INPACKRATE TO MEAN 225
DEFINE .CALLRATE TO MEAN .0014
DEFINE .NETLEVEL TO MEAN 3
DEFINE .LINESPEED TO MEAN 1500000

THE SYSTEM HAS A RANDADDR RANDOM STEP VARIABLE
THE SYSTEM HAS A RANDLENG RANDOM LINEAR VARIABLE
DEFINE RANDADDR AS AN INTEGER, STREAM 8 VARIABLE
DEFINE RANDLENG AS AN REAL, STREAM 9 VARIABLE

PROCESSES
     EVERY GENERATOR HAS A GATENO1
     EVERY VOICESESSION HAS A PACKNO2, A GATENO2, A IDENTNO2
EVENT NOTICES INCLUDE OUTPUT, BEGINSTAT
     EVERY LNKARRIVAL HAS A PACKNO3, A LNKNO3
     EVERY LNKENDSERVICE HAS A PACKNO4, A LNKNO4
     EVERY GATEARRIVAL HAS A PACKNO5, A GATENO5
     EVERY GATEENDSERVICE HAS A PACKNO6, A GATENO6
     EVERY ACKTIMEOUT HAS A PACKNO7, A GATENO7
     EVERY HOLDTIMEOUT HAS A PACKNO8, A GATENO8

TEMPORARY ENTITIES
     EVERY PACKET HAS AN ARRIVALTIME, A TIMEID, A SEQID, A SRCNO,
          A DSTNO,  A PACKCLASS, A PACKLENG, A PACKSEG,
          AN ENTRYMARK, A RETRYTIMES, A TIMESTAMP, A SUBSEQID,
          AN ACKTAG, A HOLDTAG, A DETOUR, A HOPCOUNT,
          MAY BELONG TO A LNKQUEUE,
          MAY BELONG TO A GATEQUEUE.1ST,
          MAY BELONG TO A GATEQUEUE.2ND,
          MAY BELONG TO A GATEQUEUE.3RD,
          MAY BELONG TO A BUFQUEUE, AND
          MAY BELONG TO AN ACKQUEUE
```

```
     DEFINE ARRIVALTIME,SEQID, TIMEID, TIMESTAMP AS REAL VARIABLES

     EVERY SESSION HAS A SESSTAG, A SESSSEQ, A SESSTID,
           A SESSTIMESTAMP, A SESSSRC, A SESSDST,A SESSMODE,
           A SESSSTATUS,
           MAY BELONG TO AN ACTIVESESSQUEUE
     DEFINE SESSSEQ, SESSTID, SESSTIMESTAMP AS REAL VARIABLES

PERMANENT ENTITIES
     EVERY GATEWAY HAS A GATESTATUS, AN INTRTIME, A BUF,
           A BUF.TRAN, A BUF.IN, A BUF.VOICE, AN OVERRUN,
           A PACK.TRAN, A PACK.IN, A RED, A BDLNK,
           AN UPLNK, A DNLNK1, A DNLNK2, A DNLNK3,
           A GCOUNT.VOICE.THRUPUT, A GCOUNT.DATA.THRUPUT,
           A GCOUNT.VOICE.INLOAD, A GCOUNT.DATA.INLOAD,
           A GCOUNT.VOICE.IN,
           A GCOUNT.VOICE.SUCCESS, A GCOUNT.DATA.IN,
           A GCOUNT.DATA.ADMIT, A GCOUNT.DATA.SUCCESS,
           A GCOUNT.DATA.ADMITLD,
           A GCOUNT.DATA.ACK, A GCOUNT.CONTROL.ACK,
           A GCOUNT.VOICE.SESSIN, A GCOUNT.VOICE.SESSACC,
           A GCOUNT.VOICE.SESSEND,
           A GCOUNT.RETRY.PACK, A GCOUNT.RETRY.FREQ,
           A GCOUNT.RETRY.SUCCESS, A GTRANSDELAY.VOICE,
           A GCOUNT.DATA.REJECT, A GCOUNT.VOICE.SESSREJ,
           A GTRANSDELAY.DATA,
           A GTRANSDELAY.CONTROL, A UPPERVO, A UPPERIN, A BUFNO,
           A PACKTO, A PHOLDTO, A PEXEC,
           OWNS A GATEQUEUE.1ST,
           OWNS A GATEQUEUE.2ND,
           OWNS A GATEQUEUE.3RD,
           OWNS AN ACKQUEUE,
           OWNS A BUFQUEUE,
           OWNS AN ACTIVESESSQUEUE
     DEFINE GCOUNT.DATA.ADMITLD AS A REAL VARIABLE
     DEFINE PACKTO,PHOLDTO,PEXEC,PIN AS REAL VARIABLES
     DEFINE INTRTIME, PRTIME AS REAL VARIABLES
     DEFINE GTRANSDELAY.VOICE, GTRANSDELAY.DATA,
            GTRANSDELAY.CONTROL AS REAL VARIABLES
     DEFINE GCOUNT.VOICE.THRUPUT, GCOUNT.DATA.THRUPUT,
            TCOUNT.VOICE.THRUPUT, TCOUNT.DATA.THRUPUT AS
```

```
       REAL VARIABLES
DEFINE GCOUNT.VOICE.INLOAD, GCOUNT.DATA.INLOAD,
       TCOUNT.VOICE.INLOAD, TCOUNT.DATA.INLOAD,
       TCOUNT.DATA.ADMITLD AS REAL VARIABLES

EVERY LINK HAS A LNKSTATUS, A FROMGATE, A TOGATE,
      A LNKSPEED AND OWNS A LNKQUEUE

EVERY DSTGATE HAS AN ADDR
DEFINE ADDR AS A TEXT VARIABLE

EVERY HOPCLASS HAS A HOPDELAY.D, A HOPSUC.D, A HOPREJ.D,
                   A HOPDELAY.V, A HOPSUC.V, A HOPREJ.V,
                   A HOPDELAY.C, A HOPSUC.C,
                   A HOPSESSDELAY, A HOPSESSACC, A HOPSESSREJ
DEFINE HOPDELAY.D, HOPDELAY.V, HOPDELAY.C AS REAL VARIABLES
DEFINE HOPSESSDELAY AS A REAL VARIABLE
DEFINE TCOUNT.VOICE.IN,
       TCOUNT.VOICE.SUCCESS, TCOUNT.DATA.IN,
       TCOUNT.DATA.ADMIT, TCOUNT.DATA.SUCCESS,
       TCOUNT.RETRY.SUCCESS,
       TCOUNT.DATA.ACK AS INTEGER VARIABLES
DEFINE TCOUNT.VOICE.SESSIN, TCOUNT.CONTROL.ACK,
       TCOUNT.VOICE.SESSEND,
       TCOUNT.DATA.REJECT, TCOUNT.VOICE.SESSREJ,
       TCOUNT.VOICE.SESSACC AS INTEGER VARIABLES
DEFINE TCOUNT.RETRY.PACK, TCOUNT.RETRY.FREQ AS INTEGER
       VARIABLES
DEFINE TRANSDELAY.VOICE, TRANSDELAY.DATA, TRANSDELAY.CONTROL,
       TRANSDELAY.VOICE.CONNECT AS REAL VARIABLES
DEFINE WINDOW AS A 2-DIMENSIONAL INTEGER ARRAY
DEFINE EGTEG AS AN INTEGER VARIABLE
DEFINE TVOCKT, DATASEG AS INTEGER VARIABLES

NORMALLY MODE IS REAL
TALLY AVG.HOPDELAY.D AS THE MEAN,
      MAX.HOPDELAY.D AS THE MAXIMUM,
      STDDEV.HOPDELAY.D AS THE STD.DEV OF
      HOPDELAY.D
TALLY AVG.HOPDELAY.V AS THE MEAN,
      MAX.HOPDELAY.V AS THE MAXIMUM,
```

```
      STDDEV.HOPDELAY.V AS THE STD.DEV OF
      HOPDELAY.V
TALLY AVG.HOPDELAY.C AS THE MEAN,
      MAX.HOPDELAY.C AS THE MAXIMUM,
      STDDEV.HOPDELAY.C AS THE STD.DEV OF
      HOPDELAY.C
TALLY AVG.HOPSESSDELAY AS THE MEAN,
      MAX.HOPSESSDELAY AS THE MAXIMUM,
      STDDEV.HOPSESSDELAY AS THE STD.DEV OF
      HOPSESSDELAY
TALLY AVG.GTRANSDELAY.VOICE AS THE MEAN,
      MAX.GTRANSDELAY.VOICE AS THE MAXIMUM,
      STDDEV.GTRANSDELAY.VOICE AS THE STD.DEV OF
      GTRANSDELAY.VOICE

TALLY AVG.TRANSDELAY.VOICE AS THE MEAN,
      MAX.TRANSDELAY.VOICE AS THE MAXIMUM,
      STDDEV.TRANSDELAY.VOICE AS THE STD.DEV OF
      TRANSDELAY.VOICE

TALLY AVG.GTRANSDELAY.DATA AS THE MEAN,
      MAX.GTRANSDELAY.DATA AS THE MAXIMUM,
      STDDEV.GTRANSDELAY.DATA AS THE STD.DEV OF
      GTRANSDELAY.DATA
TALLY AVG.GTRANSDELAY.CONTROL AS THE MEAN,
      MAX.GTRANSDELAY.CONTROL AS THE MAXIMUM,
      STDDEV.GTRANSDELAY.CONTROL AS THE STD.DEV OF
      GTRANSDELAY.CONTROL

TALLY AVG.TRANSDELAY.DATA AS THE MEAN,
      MAX.TRANSDELAY.DATA AS THE MAXIMUM,
      STDDEV.TRANSDELAY.DATA AS THE STD.DEV OF
      TRANSDELAY.DATA

TALLY AVG.TRANSDELAY.CONTROL AS THE MEAN,
      MAX.TRANSDELAY.CONTROL AS THE MAXIMUM,
      STDDEV.TRANSDELAY.CONTROL AS THE STD.DEV OF
      TRANSDELAY.CONTROL
TALLY AVG.TRANSDELAY.VOICE.CONNECT AS THE MEAN,
      MAX.TRANSDELAY.VOICE.CONNECT AS THE MAXIMUM,
      STDDEV.TRANSDELAY.VOICE.CONNECT AS THE STD.DEV OF
```

```
     TRANSDELAY.VOICE.CONNECT
TALLY AVG.DATASEG AS THE MEAN OF DATASEG


ACCUMULATE AVG.GATEQUEUE.1ST AS THE MEAN,
           STDDEV.GATEQUEUE.1ST AS THE STD.DEV,
           MAX.GATEQUEUE.1ST AS THE MAXIMUM OF N.GATEQUEUE.1ST

ACCUMULATE AVG.GATEQUEUE.2ND AS THE MEAN,
           STDDEV.GATEQUEUE.2ND AS THE STD.DEV,
           MAX.GATEQUEUE.2ND AS THE MAXIMUM OF N.GATEQUEUE.2ND

ACCUMULATE AVG.GATEQUEUE.3RD AS THE MEAN,
           STDDEV.GATEQUEUE.3RD AS THE STD.DEV,
           MAX.GATEQUEUE.3RD AS THE MAXIMUM OF N.GATEQUEUE.3RD

ACCUMULATE AVG.TVOCKT AS THE MEAN,
           MAX.TVOCKT AS THE MAXIMUM,
           MIN.TVOCKT AS THE MINIMUM OF TVOCKT

ACCUMULATE MAX.BUF AS THE MAXIMUM,
           AVG.BUF AS THE MEAN,
           MIN.BUF AS THE MINIMUM OF BUF
ACCUMULATE MAX.BUFTRAN AS THE MAXIMUM,
           AVG.BUFTRAN AS THE MEAN,
           MIN.BUFTRAN AS THE MINIMUM OF BUF.TRAN
ACCUMULATE MAX.BUFIN AS THE MAXIMUM,
           AVG.BUFIN AS THE MEAN,
           MIN.BUFIN AS THE MINIMUM OF BUF.IN
ACCUMULATE MAX.BUFVO AS THE MAXIMUM,
           AVG.BUFVO AS THE MEAN,
           MIN.BUFVO AS THE MINIMUM OF BUF.VOICE
ACCUMULATE MAX.PACKTRAN AS THE MAXIMUM,
           AVG.PACKTRAN AS THE MEAN,
           MIN.PACKTRAN AS THE MINIMUM OF PACK.TRAN
ACCUMULATE MAX.PACKIN AS THE MAXIMUM,
           AVG.PACKIN AS THE MEAN,
           MIN.PACKIN AS THE MINIMUM OF PACK.IN
ACCUMULATE AVG.UTILIZATION.GATE AS THE MEAN OF GATESTATUS

ACCUMULATE AVG.UTILIZATION.LINK AS THE MEAN OF LNKSTATUS
```

```
        NORMALLY MODE IS INTEGER

END
```

```
''*********************************************************************
''*                                                                   *
''*    THIS PROCESS SIMULATES THE INCOMING TRAFFIC AT EACH GATEWAY.   *
''* THE SOURCE GENERATOR GENERATES INCOMING DATA PACKETS AND VOICE    *
''* SETUP PACKETS AT THE RATE .INPACKRATE.                           *
''*                                                                   *
''*********************************************************************

PROCESS GENERATOR GIVEN GATENO

   DEFINE ID AS A REAL VARIABLE

   LET ID = 1 + 0.1 * GATENO
   UNTIL TIME.V GT .TIMELIMIT
   DO
        CREATE A PACKET CALLED INPACK
        LET ARRIVALTIME(INPACK) = TIME.V
        LET TIMEID(INPACK) = TIME.V
        LET TIMESTAMP(INPACK) = TIME.V
        LET SEQID(INPACK) = ID
        IF ID >= 99999998
           LET ID = 1 + 0.1 * GATENO
        ELSE
           LET ID = ID + 1
        ALWAYS

        LET SRCNO(INPACK) = GATENO
        LET DSTNO(INPACK) = RANDADDR
        LET HOPCOUNT(INPACK) = 1
        IF DSTNO(INPACK) = 8
           LET DSTNO(INPACK) = GATENO
        ALWAYS
        IF RANDOM.F(1) < .CALLRATE
           LET PACKCLASS(INPACK) = .VOICE.CONNECT
           LET ENTRYMARK(INPACK) = 0
           LET TCOUNT.VOICE.SESSIN = TCOUNT.VOICE.SESSIN + 1
           LET GCOUNT.VOICE.SESSIN(GATENO) =
                                GCOUNT.VOICE.SESSIN(GATENO) + 1
           LET PACKLENG(INPACK) = 32
        ELSE
           LET PACKCLASS(INPACK) = .DATA.CONTENTS
```

```
            LET ENTRYMARK(INPACK) = 1
            LET PACKLENG(INPACK) = INT.F(RANDLENG) + 26
            LET PACKSEG(INPACK)=INT.F(PACKLENG(INPACK)/.SEGLENG+0.5)
            LET DATASEG = PACKSEG(INPACK)
            LET TCOUNT.DATA.IN = TCOUNT.DATA.IN + 1
            LET GCOUNT.DATA.IN(GATENO) =
                              GCOUNT.DATA.IN(GATENO) + 1
            LET TCOUNT.DATA.INLOAD=TCOUNT.DATA.INLOAD+PACKLENG(INPACK)-26
            LET GCOUNT.DATA.INLOAD(GATENO) =
                           GCOUNT.DATA.INLOAD(GATENO)+PACKLENG(INPACK)-26
         ALWAYS
         LET RETRYTIMES(INPACK) = 0
         LET SUBSEQID(INPACK) = 0
         SCHEDULE A GATEARRIVAL GIVING INPACK AND GATENO NOW
         WAIT EXPONENTIAL.F (1/.INPACKRATE, 4) UNIT
   LOOP
END
```

```
''*********************************************************************
''*                                                                   *
''*    THIS PROCESS SIMULATES THE TRAFFIC GENERATED BY A VOICE        *
''* SPEAKER. THE VOICE GENERATOR GENERATES VOICE PACKETS ACCORDING    *
''* TO THE PREDETERMINED DATA RATES.                                  *
''*                                                                   *
''*********************************************************************

PROCESS VOICESESSION GIVEN SESSNO, GATENO

   DEFINE PERIOD, WAITTIME AS REAL VARIABLES
   LET TVOCKT = TVOCKT + 1
   IF SESSMODE(SESSNO) = .SENDER
      LET SID = 0
   ELSE
      LET SID = 1
   ALWAYS
   LET WAITTIME = 1/16
   LET PERIOD = EXPONENTIAL.F (.TALKTIME, 7)
   LET SESSSTATUS(SESSNO) = .TALKING
   WHILE PERIOD > 0 AND TIME.V <= .TIMELIMIT
   DO
        CREATE A PACKET CALLED VOICEPACK
        LET ARRIVALTIME(VOICEPACK) = TIME.V
        LET TIMESTAMP(VOICEPACK) = TIME.V
        LET SEQID(VOICEPACK) = SESSSEQ(SESSNO)
        LET TIMEID(VOICEPACK) = SESSTID(SESSNO)
        LET HOPCOUNT(VOICEPACK) = 1
        LET SRCNO(VOICEPACK) = GATENO
        IF SESSMODE(SESSNO) = .SENDER
           LET DSTNO(VOICEPACK) = SESSDST(SESSNO)
           LET ACKTAG(VOICEPACK) = .SENDER
        ELSE
           LET DSTNO(VOICEPACK) = SESSSRC(SESSNO)
           LET ACKTAG(VOICEPACK) = .RECEIVER
        ALWAYS
        LET PACKCLASS(VOICEPACK) = .VOICE.CONTENTS
        LET PACKLENG(VOICEPACK) = .VPACKLENG
        LET SUBSEQID(VOICEPACK) = SID
        IF SID >= 99999998
           LET SID = 1
```

```
      ELSE
         LET SID = SID + 1
      ALWAYS
      LET ENTRYMARK(VOICEPACK) = 0
      LET RETRYTIMES(VOICEPACK) = 0
      SCHEDULE A GATEARRIVAL GIVING VOICEPACK AND GATENO NOW
      LET TCOUNT.VOICE.IN = TCOUNT.VOICE.IN + 1
      LET GCOUNT.VOICE.IN(GATENO) =
                     GCOUNT.VOICE.IN(GATENO) + 1
      LET TCOUNT.VOICE.INLOAD=TCOUNT.VOICE.INLOAD+PACKLENG(VOICEPACK)
      LET GCOUNT.VOICE.INLOAD(GATENO) =
                GCOUNT.VOICE.INLOAD(GATENO)+PACKLENG(VOICEPACK)
      LET PERIOD = PERIOD - WAITTIME
      WAIT WAITTIME UNIT
   LOOP
      CREATE A PACKET CALLED DISCPACK
      LET ARRIVALTIME(DISCPACK) = TIME.V
      LET TIMESTAMP(DISCPACK) = TIME.V
      LET TIMEID(DISCPACK) = SESSTID(SESSNO)
      LET SEQID(DISCPACK) = SESSSEQ(SESSNO)
      LET SRCNO(DISCPACK) = GATENO
      LET HOPCOUNT(DISCPACK) = 1
      IF SESSMODE(SESSNO) = .SENDER
         LET DSTNO(DISCPACK) = SESSDST(SESSNO)
      ELSE
         LET DSTNO(DISCPACK) = SESSSRC(SESSNO)
      ALWAYS
      LET PACKCLASS(DISCPACK) = .VOICE.DISCONNECT
      LET PACKLENG(DISCPACK) = 32
      LET SUBSEQID(DISCPACK) = SESSMODE(SESSNO)
      LET ENTRYMARK(DISCPACK) = 0
      LET RETRYTIMES(DISCPACK) = 0
      SCHEDULE A GATEARRIVAL GIVING DISCPACK AND GATENO NOW
      LET SESSSTATUS(SESSNO) = .DISCONNECTING
   LET TVOCKT = TVOCKT - 1
   LET TCOUNT.VOICE.SESSEND = TCOUNT.VOICE.SESSEND + 1
   LET GCOUNT.VOICE.SESSEND(GATENO) =
                        GCOUNT.VOICE.SESSEND(GATENO) + 1

END
```

```
''*********************************************************
''*                                                       *
''*    THIS ROUTINE SIMULATES A PACKET ARRIVAL EVENT AT A  *
''* COMMUNICATION LINK.                                   *
''*                                                       *
''*********************************************************

EVENT LNKARRIVAL GIVEN PACKNO, LNKNO

   IF LNKSTATUS(LNKNO) = .IDLE
      SCHEDULE A LNKENDSERVICE GIVING PACKNO, LNKNO IN
         PACKLENG(PACKNO)/LNKSPEED(LNKNO) UNIT
      LET LNKSTATUS(LNKNO) = .BUSY
   ELSE
      FILE THIS PACKNO IN LNKQUEUE(LNKNO)
   ALWAYS

END
```

```
''*************************************************************
''*                                                           *
''*    THIS ROUTINE SIMULATES A PACKET TRANSMISSION EVENT AT A  *
''* COMMUNICATION LINK.                                       *
''*                                                           *
''*************************************************************

EVENT LINKENDSERVICE GIVEN PACKNO, LNKNO

   LET HOPCOUNT(PACKNO) = HOPCOUNT(PACKNO) + 1
   IF PACKCLASS(PACKNO) = .DATA.CONTENTS AND ENTRYMARK(PACKNO) = 0
      LET BUF.TRAN(FROMGATE(LNKNO))=BUF.TRAN(FROMGATE(LNKNO))-
                                                    PACKSEG(PACKNO)
      LET PACK.TRAN(FROMGATE(LNKNO))=PACK.TRAN(FROMGATE(LNKNO))-1
      CALL BUFRLSE GIVING FROMGATE(LNKNO),PACKSEG(PACKNO)
   ALWAYS

   IF ENTRYMARK(PACKNO) = 1
      LET ENTRYMARK(PACKNO) = 0
   ALWAYS

   SCHEDULE A GATEARRIVAL GIVING PACKNO, TOGATE(LNKNO) NOW
   IF LNKQUEUE(LNKNO) IS NOT EMPTY
      REMOVE THE FIRST PACKET FROM LNKQUEUE(LNKNO)
      SCHEDULE A LNKENDSERVICE GIVING PACKET AND LNKNO IN
         PACKLENG(PACKET)/LNKSPEED(LNKNO) UNIT
   ELSE
      LET LNKSTATUS(LNKNO) = .IDLE
   ALWAYS

END
```

```
''*********************************************************
''*                                                       *
''*    THIS ROUTINE SIMULATES A PACKET ARRIVAL EVENT AT A  *
''* GATEWAY.                                              *
''*                                                       *
''*********************************************************

EVENT GATEARRIVAL GIVEN PACKNO, GATENO

   DEFINE ADMFLAG AS AN INTEGER VARIABLE

   IF PACKCLASS(PACKNO) = .DATA.CONTENTS
      CALL FLOWCTL1 GIVING PACKNO, GATENO YIELDING ADMFLAG
   ELSE
      IF PACKCLASS(PACKNO) = .VOICE.CONTENTS
         FOR EACH PACKET IN GATEQUEUE.1ST(GATENO)
             WITH SEQID(PACKET) =  SEQID(PACKNO) AND
                  TIMEID(PACKET) = TIMEID(PACKNO) AND
                  ACKTAG(PACKET) = ACKTAG(PACKNO) AND
                  DSTNO(PACKET) = DSTNO(PACKNO)
             FIND THE FIRST CASE
             IF FOUND
                REMOVE THE PACKET FROM GATEQUEUE.1ST(GATENO)
                LET OVERRUN(GATENO) = OVERRUN(GATENO) + 1
                CALL REJRTN GIVING PACKET,2
                DESTROY THE PACKET
             ALWAYS
      ALWAYS
      GO TO ADMIT
   ALWAYS
   IF ADMFLAG = 0
      IF SRCNO(PACKNO) NE GATENO
         LET TCOUNT.DATA.REJECT = TCOUNT.DATA.REJECT + 1
         LET GCOUNT.DATA.REJECT(GATENO) =
                  GCOUNT.DATA.REJECT(GATENO) + 1
         CALL REJRTN GIVING PACKNO,1
      ALWAYS
      DESTROY THE PACKET CALLED PACKNO
      RETURN
   ALWAYS
```

```
   IF EGTEG = .YES AND
      ENTRYMARK(PACKNO) = 1 AND
      SRCNO(PACKNO) = GATENO AND
      SRCNO(PACKNO) NE DSTNO(PACKNO)
      CALL UPDWI GIVING SRCNO(PACKNO),DSTNO(PACKNO),-1
   ALWAYS
   IF ENTRYMARK(PACKNO) = 1 AND RETRYTIMES(PACKNO) = 0 AND
      SRCNO(PACKNO) = GATENO
      LET TCOUNT.DATA.ADMIT = TCOUNT.DATA.ADMIT + 1
      LET TCOUNT.DATA.ADMITLD=TCOUNT.DATA.ADMITLD+PACKLENG(PACKNO)-26
      LET GCOUNT.DATA.ADMIT(GATENO) = GCOUNT.DATA.ADMIT(GATENO) + 1
      LET GCOUNT.DATA.ADMITLD(GATENO)=GCOUNT.DATA.ADMITLD(GATENO) +
                                          PACKLENG(PACKNO)-26
   ALWAYS

'ADMIT'
   IF GATESTATUS(GATENO) = .IDLE
      IF PACKCLASS(PACKNO) = .DATA.ACK
         LET PRTIME = 0
      ELSE
         LET PRTIME = PEXEC(GATENO)
      ALWAYS
      SCHEDULE A GATEENDSERVICE GIVING PACKNO AND GATENO IN
           INTRTIME(GATENO) + PRTIME UNIT
      LET INTRTIME(GATENO) = 0
      LET GATESTATUS(GATENO) = .BUSY
   ELSE
      IF PACKCLASS(PACKNO) = .VOICE.CONTENTS
         FILE PACKNO IN GATEQUEUE.1ST(GATENO)
      ELSE
         IF PACKCLASS(PACKNO) = .DATA.CONTENTS OR
            PACKCLASS(PACKNO) = .DATA.ACK
            FILE PACKNO IN GATEQUEUE.3RD(GATENO)
         ELSE
            FILE PACKNO IN GATEQUEUE.2ND(GATENO)
         ALWAYS
      ALWAYS
   ALWAYS

END
```

```
''*****************************************************************
''*                                                               *
''*    THIS ROUTINE SIMULATES GATENET'S HOP LEVEL FLOW CONTROL    *
''* AT A GATEWAY.                                                 *
''*                                                               *
''*****************************************************************

ROUTINE FLOWCTL1 GIVEN PACKNO, GATENO YIELDING ADMFLAG

      LET ADMFLAG = 1
      IF EGTEG = .YES AND
         ENTRYMARK(PACKNO) = 1 AND
         SRCNO(PACKNO) = GATENO AND
         SRCNO(PACKNO) NE DSTNO(PACKNO)
         IF WINDOW(SRCNO(PACKNO),DSTNO(PACKNO)) <= 0
            LET ADMFLAG = 0
            RETURN
         ALWAYS
      ALWAYS
      IF BUF(GATENO) > (1-.THRESHOLD) * BUFNO(GATENO)
         LET BUF(GATENO) = BUF(GATENO) - PACKSEG(PACKNO)
         IF ENTRYMARK(PACKNO) = 1
            LET BUF.IN(GATENO)=BUF.IN(GATENO)+PACKSEG(PACKNO)
            LET PACK.IN(GATENO)=PACK.IN(GATENO)+1
         ELSE
            LET BUF.TRAN(GATENO)=BUF.TRAN(GATENO)+PACKSEG(PACKNO)
            LET PACK.TRAN(GATENO)=PACK.TRAN(GATENO)+1
         ALWAYS
      ELSE
         IF BUF(GATENO) >= PACKSEG(PACKNO)
            IF ENTRYMARK(PACKNO) = 0
               LET BUF(GATENO)=BUF(GATENO)-PACKSEG(PACKNO)
               LET BUF.TRAN(GATENO)=BUF.TRAN(GATENO)+PACKSEG(PACKNO)
               LET PACK.TRAN(GATENO)=PACK.TRAN(GATENO)+1
            ELSE
               IF ENTRYMARK(PACKNO) = 1 AND
                  BUF.IN(GATENO)+PACKSEG(PACKNO) <= UPPERIN(GATENO)
                  LET BUF(GATENO)=BUF(GATENO)-PACKSEG(PACKNO)
                  LET BUF.IN(GATENO)=BUF.IN(GATENO)+PACKSEG(PACKNO)
                  LET PACK.IN(GATENO)=PACK.IN(GATENO)+1
               ELSE
```

```
                ''CALL TRACE4("BLOCKIN",PACKNO,GATENO)
                LET ADMFLAG = 0
             ALWAYS
          ALWAYS
       ELSE
          ''CALL TRACE4("BLOCK  ",PACKNO,GATENO)
          LET ADMFLAG = 0
       ALWAYS
    ALWAYS
END
```

```
''************************************************************
''*                                                          *
''*    THIS ROUTINE SIMULATES A PACKET PROCESSING EVENT AT A  *
''* GATEWAY.                                                 *
''*                                                          *
''************************************************************

EVENT GATEENDSERVICE GIVEN PACKNO AND GATENO

   IF ENTRYMARK(PACKNO)=1 AND PACKCLASS(PACKNO) = .DATA.CONTENTS AND
      SRCNO(PACKNO) = GATENO
      CREATE A PACKET CALLED DUPPACK
      LET ARRIVALTIME(DUPPACK) = ARRIVALTIME(PACKNO)
      LET HOPCOUNT(DUPPACK) = HOPCOUNT(PACKNO)
      LET SRCNO(DUPPACK) = SRCNO(PACKNO)
      LET DSTNO(DUPPACK) = DSTNO(PACKNO)
      LET TIMESTAMP(DUPPACK) = TIMESTAMP(PACKNO)
      LET TIMEID(DUPPACK) = TIMEID(PACKNO)
      LET SEQID(DUPPACK) = SEQID(PACKNO)
      LET PACKCLASS(DUPPACK) = PACKCLASS(PACKNO)
      LET PACKLENG(DUPPACK) = PACKLENG(PACKNO)
      LET PACKSEG(DUPPACK) = PACKSEG(PACKNO)
      LET SUBSEQID(DUPPACK) = SUBSEQID(PACKNO)
      LET ENTRYMARK(DUPPACK) = ENTRYMARK(PACKNO)
      LET RETRYTIMES(DUPPACK) = RETRYTIMES(PACKNO)
      FILE DUPPACK IN ACKQUEUE(GATENO)
      IF SRCNO(PACKNO) NE DSTNO(PACKNO)
         SCHEDULE A ACKTIMEOUT CALLED ACKNO GIVING DUPPACK AND
            GATENO IN .TIMEOUT UNIT
         LET ACKTAG(DUPPACK) = ACKNO
      ALWAYS
   ALWAYS
   CALL PACKPROCESSING GIVING PACKNO AND GATENO

   IF GATEQUEUE.1ST(GATENO) IS NOT EMPTY
      REMOVE THE FIRST PACKET FROM GATEQUEUE.1ST(GATENO)
      SCHEDULE A GATEENDSERVICE GIVING PACKET AND GATENO IN
        INTRTIME(GATENO) + PEXEC(GATENO) UNIT
      LET INTRTIME(GATENO) = 0
   ELSE
      IF GATEQUEUE.2ND(GATENO) IS NOT EMPTY
```

```
      REMOVE THE FIRST PACKET FROM GATEQUEUE.2ND(GATENO)
      SCHEDULE A GATEENDSERVICE GIVING PACKET AND GATENO IN
            INTRTIME(GATENO) + PEXEC(GATENO) UNIT
      LET INTRTIME(GATENO) = 0
   ELSE
      IF GATEQUEUE.3RD(GATENO) IS NOT EMPTY
         REMOVE THE FIRST PACKET FROM GATEQUEUE.3RD(GATENO)
         IF PACKCLASS(PACKET) = .DATA.ACK
            LET PRTIME = 0
         ELSE
            LET PRTIME = PEXEC(GATENO)
         ALWAYS
         SCHEDULE A GATEENDSERVICE GIVING PACKET AND GATENO IN
               INTRTIME(GATENO) + PRTIME UNIT
         LET INTRTIME(GATENO) = 0
      ELSE
         LET GATESTATUS(GATENO) = .IDLE
      ALWAYS
   ALWAYS
ALWAYS
END
```

```
''*************************************************************
''*                                                           *
''*    THIS ROUTINE SIMULATES A DATA ACKNOWLEDGMENT TIME-OUT  *
''*  EVENT AT AN ENTRY GATEWAY.                               *
''*                                                           *
''*************************************************************

EVENT ACKTIMEOUT GIVEN PACKNO AND GATENO

  IF PACKNO IS IN ACKQUEUE
     REMOVE PACKNO FROM ACKQUEUE(GATENO)
  ELSE
     CALL TRACE1("NULAKTO",PACKNO,GATENO)
     DESTROY THE PACKET CALLED PACKNO
     RETURN
  ALWAYS
  LET INTRTIME(GATENO) = INTRTIME(GATENO) + PACKTO(GATENO)
  IF RETRYTIMES(PACKNO) < 3

     LET TCOUNT.RETRY.FREQ = TCOUNT.RETRY.FREQ + 1
     LET GCOUNT.RETRY.FREQ(GATENO) = GCOUNT.RETRY.FREQ(GATENO)+1
     IF RETRYTIMES(PACKNO) = 0
        LET TCOUNT.RETRY.PACK = TCOUNT.RETRY.PACK + 1
        LET GCOUNT.RETRY.PACK(GATENO)=GCOUNT.RETRY.PACK(GATENO)+1
     ALWAYS
     LET ENTRYMARK(PACKNO) = 1
     LET RETRYTIMES(PACKNO) = RETRYTIMES(PACKNO) + 1
     LET TIMESTAMP(PACKNO) = TIME.V
     ''CALL TRACE1("RETRY  ",PACKNO,GATENO)
     CREATE A PACKET CALLED DUPPACK
     LET ARRIVALTIME(DUPPACK) = ARRIVALTIME(PACKNO)
     LET HOPCOUNT(DUPPACK) = HOPCOUNT(PACKNO)
     LET SRCNO(DUPPACK) = SRCNO(PACKNO)
     LET DSTNO(DUPPACK) = DSTNO(PACKNO)
     LET TIMESTAMP(DUPPACK) = TIMESTAMP(PACKNO)
     LET TIMEID(DUPPACK) = TIMEID(PACKNO)
     LET SEQID(DUPPACK) = SEQID(PACKNO)
     LET PACKCLASS(DUPPACK) = PACKCLASS(PACKNO)
     LET PACKLENG(DUPPACK) = PACKLENG(PACKNO)
     LET PACKSEG(DUPPACK) = PACKSEG(PACKNO)
     LET SUBSEQID(DUPPACK) = SUBSEQID(PACKNO)
```

```
    LET ENTRYMARK(DUPPACK) = ENTRYMARK(PACKNO)
    LET RETRYTIMES(DUPPACK) = RETRYTIMES(PACKNO)
    FILE DUPPACK IN ACKQUEUE(GATENO)
    SCHEDULE A ACKTIMEOUT CALLED ACKNO GIVING DUPPACK AND GATENO IN
        .TIMEOUT UNIT
    LET ACKTAG(DUPPACK) = ACKNO
    CALL ROUTER GIVING PACKNO AND GATENO
  ELSE
    IF EGTEG = .YES AND
       SRCNO(PACKNO) NE DSTNO(PACKNO)
       CALL UPDWI GIVING SRCNO(PACKNO),DSTNO(PACKNO), 1
    ALWAYS
    ''CALL TRACE1("RDUMP  ",PACKNO,GATENO)
    LET BUF.IN(GATENO) = BUF.IN(GATENO)-PACKSEG(PACKNO)
    LET PACK.IN(GATENO) = PACK.IN(GATENO)-1
    CALL BUFRLSE GIVING GATENO,PACKSEG(PACKNO)
    DESTROY THE PACKET CALLED PACKNO
  ALWAYS
END
```

```
''*************************************************************
''*                                                           *
''*    THIS ROUTINE DEFINES THE GATENET SYSTEM CONFIGURATIONS. *
''*                                                           *
''*************************************************************

ROUTINE SYSGEN

   RESERVE WINDOW AS 7 BY 7
   LET EGTEG = .YES
   CREATE EVERY DSTGATE(7)
   LET ADDR(1) = "0.0.0.0"
   LET ADDR(2) = "2.0.0.0"
   LET ADDR(3) = "3.0.0.0"
   LET ADDR(4) = "4.0.0.0"
   LET ADDR(5) = "2.2.0.0"
   LET ADDR(6) = "4.1.0.0"
   LET ADDR(7) = "4.2.0.0"

   CREATE EVERY GATEWAY(7)
   FOR I = 1 TO N.GATEWAY
   DO
       LET GATESTATUS(I) = .IDLE
       LET BUFNO(I) = .BUFNO
       LET UPPERIN(I) = .UPPERIN
       LET UPPERVO(I) = .UPPERVO
       LET BUF(I) = .BUFNO
       LET BUF.IN(I) = 0
       LET BUF.TRAN(I) = 0
       LET PACK.TRAN(I) = 0
       LET PACK.IN(I) = 0
       LET BUF.VOICE(I) = 0
       LET PACKTO(I) = .ACKTOTIME
       LET PHOLDTO(I) = .HOLDTOTIME
       LET PEXEC(I) = .PROCESSORTIME
       LET RED(I) = .OFF
   LOOP
   LET UPPERVO(1) = 2 * .UPPERVO
   LET UPPERIN(1) = 2 * .UPPERIN
   LET BUFNO(1) = 2 * .BUFNO
   LET BUF(1) = 2 * .BUFNO
```

```
LET PACKTO(1) = .ACKTOTIME/2
LET PHOLDTO(1) = .HOLDTOTIME/2
LET PEXEC(1) = .PROCESSORTIME/2

LET UPLNK(1) = 0
LET DNLNK1(1) = 1
LET DNLNK2(1) = 3
LET DNLNK3(1) = 5
LET BDLNK(1) = 0
LET RED(1) = .FAIL

LET UPLNK(2) = 2
LET DNLNK1(2) = 7
LET DNLNK2(2) = 0
LET DNLNK3(2) = 0
LET BDLNK(2) = 0

LET UPLNK(3) = 4
LET DNLNK1(3) = 0
LET DNLNK2(3) = 0
LET DNLNK3(3) = 0
LET BDLNK(3) = 13
LET RED(3) = .ON

LET UPLNK(4) = 6
LET DNLNK1(4) = 9
LET DNLNK2(4) = 11
LET DNLNK3(4) = 0
LET BDLNK(4) = 0
LET RED(4) = .FAIL

LET UPLNK(5) = 8
LET DNLNK1(5) = 0
LET DNLNK2(5) = 0
LET DNLNK3(5) = 0
LET BDLNK(5) = 0

LET UPLNK(6) = 10
LET DNLNK1(6) = 0
LET DNLNK2(6) = 0
LET DNLNK3(6) = 0
```

```
LET BDLNK(6) = 14
LET RED(6) = .ON

LET UPLNK(7) = 12
LET DNLNK1(7) = 0
LET DNLNK2(7) = 0
LET DNLNK3(7) = 0
LET BDLNK(7) = 0

FOR I = 1 TO 7
    FOR J = 1 TO 7
        LET WINDOW(I,J) = .WINDOWSIZE

CREATE EVERY HOPCLASS(7)

CREATE EVERY LINK(14)
FOR I = 1 TO N.LINK
DO
    LET LNKSTATUS(I) = .IDLE
    LET LNKSPEED(I) = .LINESPEED
LOOP

LET TOGATE(1) = 2
LET TOGATE(2) = 1
LET TOGATE(3) = 3
LET TOGATE(4) = 1
LET TOGATE(5) = 4
LET TOGATE(6) = 1
LET TOGATE(7) = 5
LET TOGATE(8) = 2
LET TOGATE(9) = 6
LET TOGATE(10) = 4
LET TOGATE(11) = 7
LET TOGATE(12) = 4
LET TOGATE(13) = 6
LET TOGATE(14) = 3

LET FROMGATE(1) = 1
LET FROMGATE(2) = 2
LET FROMGATE(3) = 1
LET FROMGATE(4) = 3
```

```
   LET FROMGATE(5) = 1
   LET FROMGATE(6) = 4
   LET FROMGATE(7) = 2
   LET FROMGATE(8) = 5
   LET FROMGATE(9) = 4
   LET FROMGATE(10) = 6
   LET FROMGATE(11) = 4
   LET FROMGATE(12) = 7
   LET FROMGATE(13) = 3
   LET FROMGATE(14) = 6

   READ RANDADDR
   SKIP 1 INPUT LINE
   READ RANDLENG
   RETURN
END
```

```
''************************************************************
''*                                                          *
''*  THIS ROUTINE SIMULATES THE ROUTING PROCESS AT A GATEWAY.  *
''*                                                          *
''************************************************************

ROUTINE ROUTER GIVING PACKNO AND GATENO

   IF RED(GATENO) = .ON
      CALL REROUTER GIVING PACKNO, GATENO YIELDING DETOURTAG
      IF DETOURTAG = .YES
         RETURN
      ALWAYS
   ALWAYS
   LET INDEX = 0
   FOR I = 1 TO .NETLEVEL*2+1 BY 2
       WHILE INDEX = 0
   DO
     IF SUBSTR.F(ADDR(GATENO),I,1) NE
           SUBSTR.F(ADDR(DSTNO(PACKNO)),I,1)
        LET INDEX = I
     ALWAYS
   LOOP

   IF INDEX = 0
      CALL TRACE1("BUGROUT",PACKNO,GATENO)
   ALWAYS

   IF DNLNK1(GATENO) NE 0 AND
         SUBSTR.F(ADDR(DSTNO(PACKNO)), INDEX, 1) =
         SUBSTR.F(ADDR(TOGATE(DNLNK1(GATENO))), INDEX, 1)
      SCHEDULE A LNKARRIVAL GIVING PACKNO AND DNLNK1(GATENO) NOW
   ELSE
      IF DNLNK2(GATENO) NE 0 AND
            SUBSTR.F(ADDR(DSTNO(PACKNO)), INDEX, 1) =
            SUBSTR.F(ADDR(TOGATE(DNLNK2(GATENO))), INDEX, 1)
         SCHEDULE A LNKARRIVAL GIVING PACKNO AND DNLNK2(GATENO) NOW
      ELSE
         IF DNLNK3(GATENO) NE 0 AND
               SUBSTR.F(ADDR(DSTNO(PACKNO)),INDEX,1)=
               SUBSTR.F(ADDR(TOGATE(DNLNK3(GATENO))), INDEX, 1)
```

```
            SCHEDULE A LNKARRIVAL GIVING PACKNO AND DNLNK3(GATENO) NOW
         ELSE
            IF UPLNK(GATENO) NE 0
               SCHEDULE A LNKARRIVAL GIVING PACKNO AND UPLNK(GATENO) NOW
            ELSE
               CALL TRACE1("UNDELVR",PACKNO,GATENO)
               DESTROY THE PACKET CALLED PACKNO
            ALWAYS
         ALWAYS
      ALWAYS
   ALWAYS
   RETURN
END
```

```
''************************************************************
''*                                                          *
''*   THIS ROUTINE SIMULATES THE REROUTING PROCESS OF THE    *
''* BUDDY LINK SCHEME UNDER NORMAL CONDITIONS.               *
''*                                                          *
''************************************************************

ROUTINE REROUTER GIVEN PACKNO, GATENO YIELDING DETOURTAG

 LET DETOURTAG = .NO
 IF GATENO = 3 AND
       SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) = "4.1"
    LET DETOUR(PACKNO) = .NO
    LET DETOURTAG = .YES
    SCHEDULE A LNKARRIVAL GIVING PACKNO, BDLNK(GATENO) NOW
 ELSE
    IF GATENO = 6 AND
          SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "3"
       LET DETOUR(PACKNO) = .NO
       LET DETOURTAG = .YES
       SCHEDULE A LNKARRIVAL GIVING PACKNO, BDLNK(GATENO) NOW
    ALWAYS
 ALWAYS
END
```

```
''************************************************************
''*                                                          *
''*   THIS ROUTINE SIMULATES THE REROUTING PROCESS OF THE    *
''* BUDDY LINK SCHEME WHEN LINK A FAILS.                     *
''*                                                          *
''************************************************************


ROUTINE REROUTA GIVEN PACKNO, GATENO YIELDING DETOURTAG

 LET DETOURTAG = .NO
 IF GATENO = 1 AND
       SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) = "4.1"
    LET DETOUR(PACKNO) = .YES
    LET DETOURTAG = .YES
    SCHEDULE A LNKARRIVAL GIVING PACKNO, 3 NOW
 ELSE
    IF GATENO = 3 AND
          SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) = "4.1"
       LET DETOUR(PACKNO) = .NO
       LET DETOURTAG = .YES
       SCHEDULE A LNKARRIVAL GIVING PACKNO, 13 NOW
    ELSE
       IF GATENO = 4 AND
            SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) = "4.1"
          LET DETOUR(PACKNO) = .YES
          LET DETOURTAG = .YES
          SCHEDULE A LNKARRIVAL GIVING PACKNO, 6 NOW
       ELSE
          IF GATENO = 6 AND
                SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) NE "4.1"
             LET DETOUR(PACKNO) = .NO
             LET DETOURTAG = .YES
             SCHEDULE A LNKARRIVAL GIVING PACKNO, 14 NOW
          ALWAYS
       ALWAYS
    ALWAYS
 ALWAYS
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE SIMULATES THE REROUTING PROCESS OF THE  *
''* BUDDY LINK SCHEME WHEN LINK X FAILS.                  *
''*                                                       *
''*********************************************************

ROUTINE REROUTX GIVEN PACKNO, GATENO YIELDING DETOURTAG

 LET DETOURTAG = .NO
 IF GATENO = 1 AND
       SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "3"
    LET DETOUR(PACKNO) = .YES
    LET DETOURTAG = .YES
    SCHEDULE A LINKARRIVAL GIVING PACKNO, 5 NOW
 ELSE
    IF GATENO = 3 AND
          SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) NE "3"
       LET DETOUR(PACKNO) = .NO
       LET DETOURTAG = .YES
       SCHEDULE A LINKARRIVAL GIVING PACKNO, 13 NOW
    ELSE
       IF GATENO = 4 AND
            SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "3"
          LET DETOUR(PACKNO) = .YES
          LET DETOURTAG = .YES
          SCHEDULE A LINKARRIVAL GIVING PACKNO, 9 NOW
       ELSE
          IF GATENO = 6 AND
                SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "3"
             LET DETOUR(PACKNO) = .NO
             LET DETOURTAG = .YES
             SCHEDULE A LINKARRIVAL GIVING PACKNO, 14 NOW
          ALWAYS
       ALWAYS
    ALWAYS
 ALWAYS
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE SIMULATES THE REROUTING PROCESS OF THE  *
''* BUDDY LINK SCHEME WHEN LINK Y FAILS.                  *
''*                                                       *
''*********************************************************

ROUTINE REROUTY GIVEN PACKNO, GATENO YIELDING DETOURTAG

 LET DETOURTAG = .NO
 IF GATENO = 1 AND
       SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "4"
    LET DETOUR(PACKNO) = .YES
    LET DETOURTAG = .YES
    SCHEDULE A LNKARRIVAL GIVING PACKNO, 3 NOW
 ELSE
    IF GATENO = 3 AND
          SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) = "4"
       LET DETOUR(PACKNO) = .NO
       LET DETOURTAG = .YES
       SCHEDULE A LNKARRIVAL GIVING PACKNO, 13 NOW
    ELSE
       IF GATENO = 4 AND
            SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) NE "4"
          LET DETOUR(PACKNO) = .YES
          LET DETOURTAG = .YES
          SCHEDULE A LNKARRIVAL GIVING PACKNO, 9 NOW
       ELSE
          IF GATENO = 6 AND
             SUBSTR.F(ADDR(DSTNO(PACKNO)),1,3) NE "4.1" AND
             (DETOUR(PACKNO) = .YES OR
                (PACKCLASS(PACKNO) = .VOICE.CONTENTS AND
                 SUBSTR.F(ADDR(DSTNO(PACKNO)),1,1) NE "4"))
             LET DETOUR(PACKNO) = .NO
             LET DETOURTAG = .YES
             SCHEDULE A LNKARRIVAL GIVING PACKNO, 14 NOW
          ALWAYS
       ALWAYS
    ALWAYS
 ALWAYS
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF EACH TYPE   *
''* OF PACKETS AT A GATEWAY.                              *
''*                                                       *
''*********************************************************


ROUTINE PACKPROCESSING GIVEN PACKNO AND GATENO
   GO TO VOICE.CONNECT, VOICE.CONTENTS, VOICE.DISCONNECT,
      VOICE.ACCEPT,VOICE.REJECT, VOICE.DISCACK, DATA.CONTENTS,
      DATA.ACK PER PACKCLASS(PACKNO)

'VOICE.CONNECT'

  CALL VOICECONN GIVING PACKNO, GATENO
  RETURN

'VOICE.DISCONNECT'

  CALL VOICEDISC GIVING PACKNO, GATENO
  RETURN

'VOICE.REJECT'

  CALL VOICEREJ GIVING PACKNO, GATENO
  RETURN

'VOICE.ACCEPT'

  CALL VOICEACC GIVING PACKNO, GATENO
  RETURN

'VOICE.DISCACK'

  CALL VODISACK GIVING PACKNO, GATENO
  RETURN

'DATA.ACK'

  CALL DATAACK GIVING PACKNO, GATENO
```

```
  RETURN

'DATA.CONTENTS'

  CALL DATACONTS GIVING PACKNO, GATENO
  RETURN

'VOICE.CONTENTS'

  CALL VOICECONTS GIVING PACKNO, GATENO
  RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A CALL *
''* SETUP PACKET AT A GATEWAY.                        *
''*                                                   *
''*****************************************************

ROUTINE VOICECONN GIVEN PACKNO, GATENO

  IF DSTNO(PACKNO) = GATENO
     LET TRANSDELAY.CONTROL = TIME.V - TIMESTAMP(PACKNO)
     LET GTRANSDELAY.CONTROL(GATENO) = TIME.V - TIMESTAMP(PACKNO)
     CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),3
  ALWAYS

   IF BUF(GATENO) >= .VSEG AND
      (EGTEG = .NO OR BUF.VOICE(GATENO)+.VSEG <= UPPERVO(GATENO))
      LET BUF(GATENO) = BUF(GATENO) - .VSEG
      LET BUF.VOICE(GATENO) = BUF.VOICE(GATENO)+ .VSEG
      CREATE A SESSION CALLED SESSO
      LET SESSTID(SESSO) = TIMEID(PACKNO)
      LET SESSTIMESTAMP(SESSO) = TIMESTAMP(PACKNO)
      LET SESSSEQ(SESSO) = SEQID(PACKNO)
      LET SESSTAG(SESSO) = 0
      LET SESSSRC(SESSO) = SRCNO(PACKNO)
      LET SESSDST(SESSO) = DSTNO(PACKNO)
      FILE SESSO IN ACTIVESESSQUEUE(GATENO)
      IF DSTNO(PACKNO) = GATENO
         LET PACKCLASS(PACKNO) = .VOICE.ACCEPT
         LET HOPCOUNT(PACKNO) = 1
         LET PACKLENG(PACKNO) = 32
         LET ENTRYMARK(PACKNO) = 0
         IF SRCNO(PACKNO) = DSTNO(PACKNO) OR SRCNO(PACKNO) = GATENO
            LET SESSMODE(SESSO) = .SENDER
         ELSE
            IF DSTNO(PACKNO) = GATENO
               LET SESSMODE(SESSO) = .RECEIVER
            ALWAYS
         ALWAYS
         LET DSTNO(PACKNO) = SRCNO(PACKNO)
         LET SRCNO(PACKNO) = GATENO
```

```
         IF DSTNO(PACKNO) = GATENO
            FILE PACKNO IN GATEQUEUE.2ND(GATENO)
         ELSE
            CALL ROUTER GIVING PACKNO AND GATENO
         ALWAYS
      ELSE
         CALL ROUTER GIVING PACKNO AND GATENO
      ALWAYS
   ELSE
      FILE PACKNO IN BUFQUEUE(GATENO)
      SCHEDULE A HOLDTIMEOUT CALLED HOLD1 GIVING PACKNO, GATENO
               IN .HOLDTIME UNIT
      LET HOLDTAG(PACKNO) = HOLD1
   ALWAYS
   RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A CALL *
''* DISCONNECTION PACKET AT A GATEWAY.                *
''*                                                   *
''*****************************************************

ROUTINE VOICEDISC GIVEN PACKNO, GATENO

   IF DSTNO(PACKNO) =GATENO
      LET TRANSDELAY.CONTROL = TIME.V - TIMESTAMP(PACKNO)
      LET GTRANSDELAY.CONTROL(GATENO) =
                        TIME.V - TIMESTAMP(PACKNO)
      CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),3
      ''CALL TRACE1("SUCDISC",PACKNO,GATENO)
      LET DSTNO(PACKNO) = SRCNO(PACKNO)
      LET SRCNO(PACKNO) = GATENO
      LET PACKCLASS(PACKNO) = .VOICE.DISCACK
      LET HOPCOUNT(PACKNO) = 1
      LET ENTRYMARK(PACKNO) = 0
      FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
          WITH SESSSEQ(SESSION) = SEQID(PACKNO) AND
               SESSTID(SESSION) = TIMEID(PACKNO) AND
               SESSMODE(SESSION) = 1 - SUBSEQID(PACKNO)
          FIND THE FIRST CASE
          IF FOUND
             LET CURRSESS = SESSION
             IF SESSSTATUS(CURRSESS) NE .DISCONNECTING
                FOR EACH VOICESESSION IN EV.S(I.VOICESESSION)
                    WITH VOICESESSION = SESSTAG(CURRSESS)
                    FIND THE FIRST CASE
                    IF FOUND
                       CANCEL THE VOICESESSION CALLED SESSTAG(CURRSESS)
                       DESTROY THE VOICESESSION CALLED SESSTAG(CURRSESS)
                       LET TVOCKT = TVOCKT - 1
                    ALWAYS
                IF SRCNO(PACKNO) NE DSTNO(PACKNO)
                   LET BUF.VOICE(GATENO) = BUF.VOICE(GATENO) - .VSEG
                   CALL BUFRLSE GIVING GATENO,.VSEG
                ALWAYS
                REMOVE CURRSESS FROM ACTIVESESSQUEUE(GATENO)
```

```
            ''CALL TRACE2("ENDVOSD",PACKNO,GATENO,CURRSESS)
            DESTROY THE SESSION CALLED CURRSESS
         ELSE
            IF SRCNO(PACKNO) = DSTNO(PACKNO)
               REMOVE CURRSESS FROM ACTIVESESSQUEUE(GATENO)
               ''CALL TRACE2("ENDVSDD",PACKNO,GATENO,CURRSESS)
               DESTROY THE SESSION CALLED CURRSESS
               FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
                   WITH SESSSEQ(SESSION) = SEQID(PACKNO) AND
                        SESSTID(SESSION) = TIMEID(PACKNO) AND
                        SESSMODE(SESSION) = SUBSEQID(PACKNO)
                   FIND THE FIRST CASE
                   IF FOUND
                     LET CURRSESS2 = SESSION
                     REMOVE CURRSESS2 FROM ACTIVESESSQUEUE(GATENO)
                     ''CALL TRACE2("ENDVDDD",PACKNO,GATENO,CURRSESS2)
                     DESTROY THE SESSION CALLED CURRSESS2
                   ALWAYS
                   LET BUF.VOICE(GATENO)=BUF.VOICE(GATENO)-.VSEG
                   CALL BUFRLSE GIVING GATENO,.VSEG
            ALWAYS
         ALWAYS
      ELSE
         ''CALL TRACE1("DISCNUL",PACKNO,GATENO)
      ALWAYS
   ALWAYS
   IF DSTNO(PACKNO) = GATENO
      FILE PACKNO IN GATEQUEUE.2ND(GATENO)
   ELSE
      CALL ROUTER GIVING PACKNO AND GATENO
   ALWAYS
   RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A CALL *
''*  REJECTION PACKET AT A GATEWAY.                   *
''*                                                   *
''*****************************************************

ROUTINE VOICEREJ GIVEN PACKNO, GATENO

     FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
         WITH SESSSEQ(SESSION) = SEQID(PACKNO) AND
         SESSTID(SESSION) = TIMEID(PACKNO)
         FIND THE FIRST CASE
         IF FOUND
            LET CURRSESS = SESSION
            IF SESSTIMESTAMP(CURRSESS) <= TIMESTAMP(PACKNO)
               LET BUF.VOICE(GATENO) = BUF.VOICE(GATENO) - .VSEG
               CALL BUFRLSE GIVING GATENO,.VSEG
               REMOVE CURRSESS FROM ACTIVESESSQUEUE(GATENO)
               IF DSTNO(PACKNO) = GATENO
                  LET TCOUNT.VOICE.SESSREJ = TCOUNT.VOICE.SESSREJ + 1
                  LET GCOUNT.VOICE.SESSREJ(GATENO) =
                      GCOUNT.VOICE.SESSREJ(GATENO) + 1
                CALL REJRTN GIVING PACKNO,4
               ALWAYS
               DESTROY THE SESSION CALLED CURRSESS
            ALWAYS
         ELSE
            IF DSTNO(PACKNO)=GATENO AND SRCNO(PACKNO)=GATENO
               LET TCOUNT.VOICE.SESSREJ = TCOUNT.VOICE.SESSREJ + 1
               LET GCOUNT.VOICE.SESSREJ(GATENO) =
                           GCOUNT.VOICE.SESSREJ(GATENO) + 1
               CALL REJRTN GIVING PACKNO,4
            ALWAYS
         ALWAYS
     IF DSTNO(PACKNO) = GATENO
        LET TRANSDELAY.CONTROL = TIME.V - TIMESTAMP(PACKNO)
        LET GTRANSDELAY.CONTROL(GATENO) = TIME.V - TIMESTAMP(PACKNO)
        CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),3
        DESTROY THE PACKET CALLED PACKNO
     ELSE
```

```
        CALL ROUTER GIVING PACKNO AND GATENO
    ALWAYS
    RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A CALL *
''*  ACCEPTANCE PACKET AT A GATEWAY.                  *
''*                                                   *
''*****************************************************

ROUTINE VOICEACC GIVEN PACKNO, GATENO

   IF DSTNO(PACKNO) = GATENO
      LET TCOUNT.VOICE.SESSACC=TCOUNT.VOICE.SESSACC+1
      LET GCOUNT.VOICE.SESSACC(GATENO) =
                     GCOUNT.VOICE.SESSACC(GATENO)+1
      LET TRANSDELAY.VOICE.CONNECT=TIME.V-ARRIVALTIME(PACKNO)
      CALL SUCRTN GIVING PACKNO,TIME.V-ARRIVALTIME(PACKNO),4
      FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
          WITH SESSTID(SESSION) = TIMEID(PACKNO) AND
               SESSSEQ(SESSION) = SEQID(PACKNO) AND
               SESSMODE(SESSION) = .SENDER
          FIND THE FIRST CASE
          IF FOUND
             ACTIVATE A VOICESESSION CALLED NEWSESS1 GIVING
                    SESSION, GATENO NOW
             LET SESSTAG(SESSION) = NEWSESS1
          ELSE
             ''CALL TRACE2("VACCBUG",PACKNO,GATENO,SESSION)
          ALWAYS
   ALWAYS
   IF DSTNO(PACKNO) = GATENO
      LET TRANSDELAY.CONTROL = TIME.V - TIMESTAMP(PACKNO)
      LET GTRANSDELAY.CONTROL(GATENO) = TIME.V - TIMESTAMP(PACKNO)
      CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),3
      DESTROY THE PACKET CALLED PACKNO
   ELSE
      CALL ROUTER GIVING PACKNO AND GATENO
   ALWAYS
   RETURN
END
```

```
''*******************************************************
''*                                                     *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A CALL   *
''*  DISCONNECTION ACKNOWLEDGMENT PACKET AT A GATEWAY.  *
''*                                                     *
''*******************************************************

ROUTINE VODISACK GIVEN PACKNO, GATENO

     FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
         WITH SESSSEQ(SESSION) = SEQID(PACKNO) AND
              SESSTID(SESSION) = TIMEID(PACKNO)
         FIND THE FIRST CASE
         IF FOUND
            REMOVE THE SESSION FROM ACTIVESESSQUEUE(GATENO)
            IF DSTNO(PACKNO) = GATENO
               ''CALL TRACE2("ENDVOSA",PACKNO,GATENO,SESSION)
            ALWAYS

            DESTROY THE SESSION
            LET BUF.VOICE(GATENO) = BUF.VOICE(GATENO) - .VSEG
            CALL BUFRLSE GIVING GATENO,.VSEG
         ALWAYS

  IF DSTNO(PACKNO) = GATENO
     LET TRANSDELAY.CONTROL = TIME.V - TIMESTAMP(PACKNO)
     LET GTRANSDELAY.CONTROL(GATENO) = TIME.V - TIMESTAMP(PACKNO)
     CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),3
     DESTROY THE PACKET CALLED PACKNO
  ELSE
     CALL ROUTER GIVING PACKNO AND GATENO
  ALWAYS
 RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A DATA  *
''*  ACKNOWLEDGMENT PACKET AT A GATEWAY.              *
''*                                                   *
''*****************************************************

ROUTINE DATAACK GIVEN PACKNO, GATENO
     IF DSTNO(PACKNO) = GATENO
        FOR EACH PACKET IN ACKQUEUE(GATENO)
            WITH SEQID(PACKET) = SEQID(PACKNO) AND
                 TIMEID(PACKET) = TIMEID(PACKNO) AND
                 PACKCLASS(PACKET) = .DATA.CONTENTS
          FIND THE FIRST CASE
          IF FOUND
             LET CURRPACK = PACKET
             REMOVE CURRPACK FROM ACKQUEUE(GATENO)
             FOR EACH ACKTIMEOUT IN EV.S(I.ACKTIMEOUT)
                 WITH ACKTIMEOUT = ACKTAG(CURRPACK)
                 FIND THE FIRST CASE
                 IF FOUND
                    CANCEL THE ACKTIMEOUT CALLED ACKTAG(CURRPACK)
                    DESTROY THE ACKTIMEOUT CALLED ACKTAG(CURRPACK)
                 ALWAYS
             IF EGTEG = .YES AND SRCNO(CURRPACK) NE DSTNO(CURRPACK)
                CALL UPDWI GIVING SRCNO(CURRPACK),DSTNO(CURRPACK),1
             ALWAYS
             LET BUF.IN(GATENO)=BUF.IN(GATENO)-PACKSEG(CURRPACK)
             LET PACK.IN(GATENO)=PACK.IN(GATENO)-1
             CALL BUFRLSE GIVING GATENO,PACKSEG(CURRPACK)
             DESTROY THE PACKET CALLED CURRPACK
          ALWAYS
        DESTROY THE PACKET CALLED PACKNO
        LET TCOUNT.DATA.ACK = TCOUNT.DATA.ACK + 1
        LET GCOUNT.DATA.ACK(GATENO)=GCOUNT.DATA.ACK(GATENO)+1
     ELSE
        CALL ROUTER GIVING PACKNO AND GATENO
     ALWAYS
 RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A DATA  *
''*  CONTENTS PACKET AT A GATEWAY.                     *
''*                                                   *
''*****************************************************

ROUTINE DATACONTS GIVEN PACKNO, GATENO

   IF DSTNO(PACKNO) =GATENO
      LET TCOUNT.DATA.THRUPUT=TCOUNT.DATA.THRUPUT+PACKLENG(PACKNO)-26
      LET GCOUNT.DATA.THRUPUT(GATENO)=GCOUNT.DATA.THRUPUT(GATENO)+
                                        PACKLENG(PACKNO)-26
      LET TCOUNT.DATA.SUCCESS = TCOUNT.DATA.SUCCESS + 1
      LET GCOUNT.DATA.SUCCESS(GATENO)=GCOUNT.DATA.SUCCESS(GATENO)+1
      ''CALL TRACE1("SUCDATA",PACKNO,GATENO)
      IF RETRYTIMES(PACKNO) NE 0
         LET TCOUNT.RETRY.SUCCESS = TCOUNT.RETRY.SUCCESS + 1
         LET GCOUNT.RETRY.SUCCESS(GATENO) =
                              GCOUNT.RETRY.SUCCESS(GATENO) + 1
      ALWAYS
      LET TRANSDELAY.DATA = TIME.V - TIMESTAMP(PACKNO)
      LET GTRANSDELAY.DATA(GATENO) = TIME.V - TIMESTAMP(PACKNO)
      CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),1
      IF SRCNO(PACKNO) NE DSTNO(PACKNO)
         LET BUF.TRAN(GATENO)=BUF.TRAN(GATENO)-PACKSEG(PACKNO)
         LET PACK.TRAN(GATENO)=PACK.TRAN(GATENO)-1
         CALL BUFRLSE GIVING GATENO,PACKSEG(PACKNO)
      ALWAYS
      LET DSTNO(PACKNO) = SRCNO(PACKNO)
      LET SRCNO(PACKNO) = GATENO
      LET PACKCLASS(PACKNO) = .DATA.ACK
      LET HOPCOUNT(PACKNO) = 1
      LET PACKLENG(PACKNO) = 0
      LET ENTRYMARK(PACKNO) = 0
      IF DSTNO(PACKNO) = GATENO
         FILE PACKNO IN GATEQUEUE.3RD(GATENO)
      ELSE
         CALL ROUTER GIVING PACKNO AND GATENO
      ALWAYS
   ELSE
```

```
      CALL ROUTER GIVING PACKNO AND GATENO
   ALWAYS
   RETURN
END
```

```
''*****************************************************
''*                                                   *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF A VOICE  *
''*   CONTENTS PACKET AT A GATEWAY.                     *
''*                                                   *
''*****************************************************

ROUTINE VOICECONTS GIVEN PACKNO, GATENO

   IF DSTNO(PACKNO)=GATENO
      LET TRANSDELAY.VOICE = TIME.V - TIMESTAMP(PACKNO)
      LET GTRANSDELAY.VOICE(GATENO) = TIME.V - TIMESTAMP(PACKNO)
      CALL SUCRTN GIVING PACKNO,TIME.V-TIMESTAMP(PACKNO),2
      LET TCOUNT.VOICE.THRUPUT=TCOUNT.VOICE.THRUPUT+.VPACKLENG-16
      LET GCOUNT.VOICE.THRUPUT(GATENO)=GCOUNT.VOICE.THRUPUT(GATENO)+
                                       .VPACKLENG-16
      LET TCOUNT.VOICE.SUCCESS = TCOUNT.VOICE.SUCCESS + 1
      LET GCOUNT.VOICE.SUCCESS(GATENO)=GCOUNT.VOICE.SUCCESS(GATENO)+1
   ALWAYS

   IF SUBSEQID(PACKNO) = 0 AND DSTNO(PACKNO) = GATENO

      IF SRCNO(PACKNO) = DSTNO(PACKNO)
         CREATE A SESSION CALLED SESS1
         LET SESSTID(SESS1) = TIMEID(PACKNO)
         LET SESSTIMESTAMP(SESS1) = TIMESTAMP(PACKNO)
         LET SESSSEQ(SESS1) = SEQID(PACKNO)
         LET SESSSRC(SESS1) = SRCNO(PACKNO)
         LET SESSDST(SESS1) = DSTNO(PACKNO)
         FILE SESS1 IN ACTIVESESSQUEUE(GATENO)
         LET SESSMODE(SESS1) = .RECEIVER
         ACTIVATE A VOICESESSION CALLED NEWSESS2 GIVING SESS1,
                GATENO NOW
         LET SESSTAG(SESS1) = NEWSESS2
      ELSE
         FOR EACH SESSION IN ACTIVESESSQUEUE(GATENO)
              WITH SESSTID(SESSION) = TIMEID(PACKNO) AND
                   SESSSEQ(SESSION) = SEQID(PACKNO) AND
                   SESSMODE(SESSION) = .RECEIVER
              FIND THE FIRST CASE
              IF FOUND
```

```
                ACTIVATE A VOICESESSION CALLED NEWSESS4 GIVING
                      SESSION, GATENO NOW
                LET SESSTAG(SESSION) = NEWSESS4
             ELSE
                CALL TRACE1("VCTSBUG",PACKNO,GATENO)
             ALWAYS
       ALWAYS
   ALWAYS
   IF DSTNO(PACKNO) = GATENO
      DESTROY THE PACKET CALLED PACKNO
   ELSE
      CALL ROUTER GIVING PACKNO AND GATENO
   ALWAYS
   RETURN
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE SIMULATES THE PROCESSING OF RELEASING  *
''*  BUFFER SEGMENTS AT A GATEWAY.                        *
''*                                                       *
''*********************************************************

ROUTINE BUFRLSE GIVEN GATENO,SEGNO

    LET BUF(GATENO) = BUF(GATENO) + SEGNO
   IF BUFQUEUE(GATENO) IS EMPTY OR
      BUF(GATENO) < .VSEG OR
      (EGTEG = .YES AND BUF.VOICE(GATENO)+.VSEG > UPPERVO(GATENO))
      RETURN
   ALWAYS

      REMOVE THE FIRST PACKET FROM BUFQUEUE(GATENO)
      LET HLDPACK = PACKET
      LET BUF(GATENO)=BUF(GATENO)-.VSEG
      LET BUF.VOICE(GATENO)=BUF.VOICE(GATENO)+.VSEG
      FOR EACH HOLDTIMEOUT IN EV.S(I.HOLDTIMEOUT)
          WITH HOLDTIMEOUT = HOLDTAG(HLDPACK)
          FIND THE FIRST CASE
          IF FOUND
             CANCEL THE HOLDTIMEOUT CALLED HOLDTAG(HLDPACK)
             DESTROY THE HOLDTIMEOUT CALLED HOLDTAG(HLDPACK)
          ELSE
             CALL TRACE1("HLDBUG2",HLDPACK,GATENO)
          ALWAYS
      ''CALL TRACE4("HLDACC ",HLDPACK,GATENO)
      CREATE A SESSION CALLED SESS3
      LET SESSTID(SESS3) = TIMEID(HLDPACK)
      LET SESSTIMESTAMP(SESS3) = TIMESTAMP(HLDPACK)
      LET SESSSEQ(SESS3) = SEQID(HLDPACK)
      LET SESSTAG(SESS3) = 0
      LET SESSSRC(SESS3) = SRCNO(HLDPACK)
      LET SESSDST(SESS3) = DSTNO(HLDPACK)
      FILE SESS3 IN ACTIVESESSQUEUE(GATENO)

      IF DSTNO(HLDPACK) = GATENO
         LET PACKCLASS(HLDPACK) = .VOICE.ACCEPT
```

```
         LET HOPCOUNT(HLDPACK) = 1
         LET PACKLENG(HLDPACK) = 32
         LET ENTRYMARK(HLDPACK) = 0
         IF SRCNO(HLDPACK) = DSTNO(HLDPACK) OR SRCNO(HLDPACK) = GATENO
            LET SESSMODE(SESS3) = .SENDER
         ELSE
            IF DSTNO(HLDPACK) = GATENO
               LET SESSMODE(SESS3) = .RECEIVER
            ALWAYS
         ALWAYS
         LET DSTNO(HLDPACK) = SRCNO(HLDPACK)
         LET SRCNO(HLDPACK) = GATENO
      ALWAYS
      IF DSTNO(HLDPACK) = GATENO
         FILE HLDPACK IN GATEQUEUE.2ND(GATENO)
      ELSE
         CALL ROUTER GIVING HLDPACK AND GATENO
      ALWAYS
   RETURN
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE SIMULATES THE TIME-OUT EVENT OF A CALL *
''*  SETUP HOLDING AT A GATEWAY.                          *
''*                                                       *
''*********************************************************

EVENT HOLDTIMEOUT GIVEN PACKNO AND GATENO

      ''CALL TRACE4("HLDTO  ",PACKNO,GATENO)
      LET INTRTIME(GATENO) = INTRTIME(GATENO) + PHOLDTO(GATENO)
      IF PACKNO IS IN BUFQUEUE
         REMOVE PACKNO FROM BUFQUEUE(GATENO)
      ALWAYS
      LET PACKCLASS(PACKNO) = .VOICE.REJECT
      LET HOPCOUNT(PACKNO) = 1
      LET PACKLENG(PACKNO) = 32
      LET DSTNO(PACKNO) = SRCNO(PACKNO)
      LET SRCNO(PACKNO) = GATENO
      LET ENTRYMARK(PACKNO) = 0
      IF DSTNO(PACKNO) = GATENO
         FILE PACKNO IN GATEQUEUE.2ND(GATENO)
      ELSE
         CALL ROUTER GIVING PACKNO AND GATENO
      ALWAYS
   RETURN
END
```

```
''*****************************************************************
''*                                                               *
''*   MAIN ROUTINE DEFINES THE SYSTEM CONFIGURATIONS, ACTIVATES   *
''* THE SOURCE GENERATOR OF EACH GATEWAY, SETS THE BEGINNING AND *
''* ENDING POINTS FOR THE STATISTICS COLLECTION PERIOD, AND       *
''* STARTS THE SIMULATION RUN.                                    *
''*                                                               *
''*****************************************************************


MAIN

   CALL SYSGEN

   FOR I = 1 TO N.GATEWAY
   DO
       ACTIVATE A GENERATOR GIVING I NOW
   LOOP

   IF .STARTPT NE 0
      SCHEDULE AN BEGINSTAT IN .STARTPT UNIT
   ALWAYS
   SCHEDULE AN OUTPUT IN .TIMELIMIT UNIT
   START SIMULATION
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE RESETS ALL THE STATISTICAL VARIABLES AT  *
''*  THE BEGINNING OF THE STATISTICS GATHERING PERIOD.      *
''*                                                       *
''*********************************************************

EVENT BEGINSTAT

  RESET TOTALS OF TRANSDELAY.DATA, TRANSDELAY.CONTROL, DATASEG,
                  TRANSDELAY.VOICE, TRANSDELAY.VOICE.CONNECT, TVOCKT
  LET TCOUNT.VOICE.THRUPUT = 0
  LET TCOUNT.DATA.THRUPUT = 0
  LET TCOUNT.VOICE.INLOAD = 0
  LET TCOUNT.DATA.INLOAD = 0
  LET TCOUNT.VOICE.IN = 0
  LET TCOUNT.VOICE.SUCCESS = 0
  LET TCOUNT.DATA.IN = 0
  LET TCOUNT.DATA.ADMIT = 0
  LET TCOUNT.DATA.ADMITLD = 0
  LET TCOUNT.DATA.SUCCESS = 0
  LET TCOUNT.RETRY.SUCCESS = 0
  LET TCOUNT.DATA.ACK = 0
  LET TCOUNT.VOICE.SESSIN = 0
  LET TCOUNT.CONTROL.ACK = 0
  LET TCOUNT.VOICE.SESSEND = 0
  LET TCOUNT.DATA.REJECT = 0
  LET TCOUNT.VOICE.SESSREJ = 0
  LET TCOUNT.VOICE.SESSACC = 0
  LET TCOUNT.RETRY.PACK = 0
  LET TCOUNT.RETRY.FREQ = 0
  FOR EACH LINK
      RESET TOTALS OF LNKSTATUS(LINK)

  FOR EACH GATEWAY
  DO
      RESET TOTALS OF GTRANSDELAY.DATA(GATEWAY),
                      GTRANSDELAY.CONTROL(GATEWAY),
                      GTRANSDELAY.VOICE(GATEWAY),
                      N.GATEQUEUE.1ST(GATEWAY),
                      N.GATEQUEUE.2ND(GATEWAY),
```

```
                    N.GATEQUEUE.3RD(GATEWAY),
                    BUF(GATEWAY),
                    PACK.IN(GATEWAY),
                    BUF.IN(GATEWAY),
                    BUF.TRAN(GATEWAY),
                    PACK.TRAN(GATEWAY),
                    BUF.VOICE(GATEWAY),
                    GATESTATUS(GATEWAY)
    LET GCOUNT.VOICE.THRUPUT(GATEWAY) = 0
    LET GCOUNT.DATA.THRUPUT(GATEWAY) = 0
    LET GCOUNT.VOICE.INLOAD(GATEWAY) = 0
    LET GCOUNT.DATA.INLOAD(GATEWAY) = 0
    LET GCOUNT.VOICE.IN(GATEWAY) = 0
    LET GCOUNT.VOICE.SUCCESS(GATEWAY) = 0
    LET GCOUNT.DATA.IN(GATEWAY) = 0
    LET GCOUNT.DATA.ADMIT(GATEWAY) = 0
    LET GCOUNT.DATA.ADMITLD(GATEWAY) = 0
    LET GCOUNT.DATA.SUCCESS(GATEWAY) = 0
    LET GCOUNT.RETRY.SUCCESS(GATEWAY) = 0
    LET GCOUNT.DATA.ACK(GATEWAY) = 0
    LET GCOUNT.VOICE.SESSIN(GATEWAY) = 0
    LET GCOUNT.CONTROL.ACK(GATEWAY) = 0
    LET GCOUNT.VOICE.SESSEND(GATEWAY) = 0
    LET GCOUNT.DATA.REJECT(GATEWAY) = 0
    LET GCOUNT.VOICE.SESSREJ(GATEWAY) = 0
    LET GCOUNT.VOICE.SESSACC(GATEWAY) = 0
    LET GCOUNT.RETRY.PACK(GATEWAY) = 0
    LET GCOUNT.RETRY.FREQ(GATEWAY) = 0
LOOP

FOR EACH HOPCLASS
DO
    RESET TOTALS OF HOPDELAY.D(HOPCLASS),
                    HOPDELAY.C(HOPCLASS),
                    HOPDELAY.V(HOPCLASS),
                    HOPSESSDELAY(HOPCLASS)

    LET HOPSUC.D(HOPCLASS) = 0
    LET HOPSUC.V(HOPCLASS) = 0
    LET HOPSUC.C(HOPCLASS) = 0
    LET HOPREJ.D(HOPCLASS) = 0
```

```
      LET HOPREJ.V(HOPCLASS) = 0
      LET HOPSESSACC(HOPCLASS) = 0
      LET HOPSESSREJ(HOPCLASS) = 0
  LOOP
END
```

```
''*********************************************************
''*                                                       *
''*   THIS ROUTINE OUTPUTS ALL THE STATISTICS AT THE END OF *
''* THE STATISTICS GATHERING PERIOD.                      *
''*                                                       *
''*********************************************************

EVENT OUTPUT

  LET STARTTIME = .STARTPT
  LET STOPTIME = .TIMELIMIT
  START NEW PAGE
  PRINT 2 LINE WITH STARTTIME, STOPTIME THUS
### STATISTICS FOR TIME PERIOD (SEC) =  ****.* TO ****.* ###

   PRINT 47 LINES WITH AVG.TRANSDELAY.VOICE, MAX.TRANSDELAY.VOICE,
     STDDEV.TRANSDELAY.VOICE, AVG.TRANSDELAY.DATA, MAX.TRANSDELAY.DATA,
     STDDEV.TRANSDELAY.DATA, AVG.TRANSDELAY.CONTROL,
     MAX.TRANSDELAY.CONTROL, STDDEV.TRANSDELAY.CONTROL,
     AVG.TRANSDELAY.VOICE.CONNECT,
     MAX.TRANSDELAY.VOICE.CONNECT, STDDEV.TRANSDELAY.VOICE.CONNECT,
     TCOUNT.VOICE.IN/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.IN/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.IN+TCOUNT.DATA.IN)/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.ADMIT/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.IN+TCOUNT.DATA.ADMIT)/(.TIMELIMIT-.STARTPT),
     TCOUNT.VOICE.SUCCESS/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.SUCCESS/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.SUCCESS+TCOUNT.DATA.SUCCESS)/(.TIMELIMIT-.STARTPT),
     TCOUNT.VOICE.INLOAD*8/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.INLOAD*8/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.INLOAD+TCOUNT.DATA.INLOAD)*8/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.ADMITLD*8/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.INLOAD+TCOUNT.DATA.ADMITLD)*8/(.TIMELIMIT-.STARTPT),
     TCOUNT.VOICE.THRUPUT*8/(.TIMELIMIT-.STARTPT),
     TCOUNT.DATA.THRUPUT*8/(.TIMELIMIT-.STARTPT),
     (TCOUNT.VOICE.THRUPUT+TCOUNT.DATA.THRUPUT)*8/(.TIMELIMIT-.STARTPT),
     MAX.TVOCKT/2, AVG.TVOCKT/2,
     MIN.TVOCKT/2, AVG.DATASEG, TCOUNT.VOICE.IN,
     TCOUNT.VOICE.SUCCESS, TCOUNT.VOICE.SESSIN, TCOUNT.VOICE.SESSEND,
     TCOUNT.VOICE.SESSACC, TCOUNT.DATA.IN, TCOUNT.DATA.ADMIT,
```

```
TCOUNT.DATA.SUCCESS, TCOUNT.DATA.ACK,
TCOUNT.RETRY.PACK,TCOUNT.RETRY.FREQ, TCOUNT.RETRY.SUCCESS,
TCOUNT.DATA.REJECT,TCOUNT.VOICE.SESSREJ THUS

   AVG.TRANSDELAY.VOICE                = ***.*****
   MAX.TRANSDELAY.VOICE                = ***.*****
   STDDEV.TRANSDELAY.VOICE             = ***.*****
   AVG.TRANSDELAY.DATA                 = ***.*****
   MAX.TRANSDELAY.DATA                 = ***.*****
   STDDEV.TRANSDELAY.DATA              = ***.*****
   AVG.TRANSDELAY.CONTROL              = ***.*****
   MAX.TRANSDELAY.CONTROL              = ***.*****
   STDDEV.TRANSDELAY.CONTROL           = ***.*****
   AVG.TRANSDELAY.VOICE.CONNECT        = ***.*****
   MAX.TRANSDELAY.VOICE.CONNECT        = ***.*****
   STDDEV.TRANSDELAY.VOICE.CONNECT     = ***.*****
   AVG.VOICE.INLOAD(PACK/S)   = ******************
   AVG.DATA.INLOAD(PACK/S)    = ******************
   AVG.TOTAL.INLOAD(PACK/S)   = ******************
   AVG.DATA.ADMIT(PACK/S)     = ******************
   AVG.TOTAL.ADMIT(PACK/S)    = ******************
   AVG.VOICE.THRUPUT(PACK/S)  = ******************
   AVG.DATA.THRUPUT(PACK/S)   = ******************
   AVG.TOTAL.THRUPUT(PACK/S)  = ******************
   AVG.VOICE.INLOAD(BPS)      = ******************
   AVG.DATA.INLOAD(BPS)       = ******************
   AVG.TOTAL.INLOAD(BPS)      = ******************
   AVG.DATA.ADMITLD(BPS)      = ******************
   AVG.TOTAL.ADMITLD(BPS)     = ******************
   AVG.VOICE.THRUPUT(BPS)     = ******************
   AVG.DATA.THRUPUT(BPS)      = ******************
   AVG.TOTAL.THRUPUT(BPS)     = ******************
   MAX.TVOCKT  =  ***
   AVG.TVOCKT  =  ***
   MIN.TVOCKT  =  ***
   AVG.DATASEG =  **.*
   TCOUNT.VOICE.IN          = **********
   TCOUNT.VOICE.SUCCESS     = **********
   TCOUNT.VOICE.SESSIN      = **********
   TCOUNT.VOICE.SESSEND     = **********
   TCOUNT.VOICE.SESSACC     = **********
```

```
        TCOUNT.DATA.IN          = **********
        TCOUNT.DATA.ADMIT       = **********
        TCOUNT.DATA.SUCCESS     = **********
        TCOUNT.DATA.ACK         = **********
        TCOUNT.RETRY.PACK       = **********
        TCOUNT.RETRY.FREQ       = **********
        TCOUNT.RETRY.SUCCESS    = **********
        TCOUNT.DATA.REJECT      = **********
        TCOUNT.VOICE.SESSREJ    = **********
        START NEW PAGE
        PRINT 1 DOUBLE LINE THUS
GT MAXQ.1 AVGQ.1 SDEV.1 MAXQ.2 AVGQ.2 SDEV.2 MAXQ.3 AVGQ.3 SDEV.3
 UTL

        FOR EACH GATEWAY
          PRINT 1 DOUBLE LINE WITH GATEWAY, MAX.GATEQUEUE.1ST(GATEWAY),
                                AVG.GATEQUEUE.1ST(GATEWAY),
                                STDDEV.GATEQUEUE.1ST(GATEWAY),
                                MAX.GATEQUEUE.2ND(GATEWAY),
                                AVG.GATEQUEUE.2ND(GATEWAY),
                                STDDEV.GATEQUEUE.2ND(GATEWAY)-,
                                MAX.GATEQUEUE.3RD(GATEWAY),
                                AVG.GATEQUEUE.3RD(GATEWAY),
                                STDDEV.GATEQUEUE.3RD(GATEWAY),
                                AVG.UTILIZATION.GATE(GATEWAY) THUS
**  ***** ***.** ***.**  ***** ***.** ***.**  ***** ***.** ***.**
*.**

        PRINT 1 DOUBLE LINE THUS
GTE     VIN     VSU VSEIN VSEAC VSREJ  VSEED    DIN    DAD    DSU
   DREJ   DACK   RTYP   RTYF   RTYS   OVRUN

        FOR EACH GATEWAY
        PRINT 1 DOUBLE LINE WITH GATEWAY,GCOUNT.VOICE.IN(GATEWAY),
                                GCOUNT.VOICE.SUCCESS(GATEWAY),
                                GCOUNT.VOICE.SESSIN(GATEWAY),
                                GCOUNT.VOICE.SESSACC(GATEWAY),
                                GCOUNT.VOICE.SESSREJ(GATEWAY),
                                GCOUNT.VOICE.SESSEND(GATEWAY),
                                GCOUNT.DATA.IN(GATEWAY),
                                GCOUNT.DATA.ADMIT(GATEWAY),
```

```
                                GCOUNT.DATA.SUCCESS(GATEWAY),
                                GCOUNT.DATA.REJECT(GATEWAY),
                                GCOUNT.DATA.ACK(GATEWAY),
                                GCOUNT.RETRY.PACK(GATEWAY),
                                GCOUNT.RETRY.FREQ(GATEWAY),
                                GCOUNT.RETRY.SUCCESS(GATEWAY),
                                OVERRUN(GATEWAY) THUS
 **     ****     ****  ****  ****  ****    ****    ****    ****    ****
   ****    ****    ****    ****    ****     ****


      PRINT 2 DOUBLE LINES THUS


HOP DAVGDLY DMAVDLY DSTDDLY  DSUCNO  DREJNO VAVGDLY VMAXDLY VSTDDLY
  VSUCNO  VOVRUN VSAVGDLY VSMAXDLY VSSTDDLY   VSACC    VSREJ
      FOR EACH HOPCLASS
        PRINT 1 DOUBLE LINE WITH HOPCLASS,AVG.HOPDELAY.D(HOPCLASS),
                                 MAX.HOPDELAY.D(HOPCLASS),
                                 STDDEV.HOPDELAY.D(HOPCLASS),
                                 HOPSUC.D(HOPCLASS),
                                 HOPREJ.D(HOPCLASS),
                                 AVG.HOPDELAY.V(HOPCLASS),
                                 MAX.HOPDELAY.V(HOPCLASS),
                                 STDDEV.HOPDELAY.V(HOPCLASS),
                                 HOPSUC.V(HOPCLASS),
                                 HOPREJ.V(HOPCLASS),
                                 HOPSESSDELAY(HOPCLASS),
                                 MAX.HOPSESSDELAY(HOPCLASS),
                                 STDDEV.HOPSESSDELAY(HOPCLASS),
                                 HOPSESSACC(HOPCLASS),
                                 HOPSESSREJ(HOPCLASS) THUS
  * **.**** **.**** **.****  ******  ****** **.**** **.**** **.****
  ******  ****** ***.**** ***.**** ***.****  ******   ******

      PRINT 2 LINES THUS

HOPNO  CAVGDLY   CMAVDLY   CSTDDLY    CSUCNO
      FOR EACH HOPCLASS
        PRINT 1 LINE WITH HOPCLASS,AVG.HOPDELAY.C(HOPCLASS),
                                 MAX.HOPDELAY.C(HOPCLASS),
```

```
                                        STDDEV.HOPDELAY.C(HOPCLASS),
                                        HOPSUC.C(HOPCLASS) THUS
    * ****.***  ****.***  ****.***   *******

        PRINT 2 LINES THUS

        LINK       UTILIZATION

        FOR EACH LINK
            PRINT 1 LINE WITH LINK, AVG.UTILIZATION.LINK(LINK) THUS
        **         *.***
        START NEW PAGE
        PRINT 1 DOUBLE LINE THUS
GT MAXBUF AVGBUF MINBUF MAXIN AVGIN MININ MAXTR AVGTR MINTR MAXVO AVGVO
MINVO  BUF BUFIN BUFTR BUFVO

 FOR EACH GATEWAY
  PRINT 1 DOUBLE LINE WITH GATEWAY,MAX.BUF(GATEWAY),AVG.BUF(GATEWAY),
        MIN.BUF(GATEWAY),MAX.BUFIN(GATEWAY),AVG.BUFIN(GATEWAY),
        MIN.BUFIN(GATEWAY),MAX.BUFTRAN(GATEWAY),AVG.BUFTRAN(GATEWAY),
        MIN.BUFTRAN(GATEWAY),MAX.BUFVO(GATEWAY),AVG.BUFVO(GATEWAY),
        MIN.BUFVO(GATEWAY),BUF(GATEWAY),BUF.IN(GATEWAY),
        BUF.TRAN(GATEWAY),BUF.VOICE(GATEWAY) THUS
**    ***  ***.*    ***   *** ***.*   ***   *** ***.*   ***   *** ***.*
  ***  ***   ***   ***   ***
        PRINT 1 LINE THUS
GT MAXINP AVGINP MININP MAXTRP AVGTRP MINTRP    INP    TRP

 FOR EACH GATEWAY
  PRINT 1 LINE WITH GATEWAY,MAX.PACKIN(GATEWAY),AVG.PACKIN(GATEWAY),
    MIN.PACKIN(GATEWAY),MAX.PACKTRAN(GATEWAY),AVG.PACKTRAN(GATEWAY),
    MIN.PACKTRAN(GATEWAY),
     PACK.IN(GATEWAY),PACK.TRAN(GATEWAY) THUS
 * ****** ***.** ****** ****** ***.** ******   ****   ****

  PRINT 1 DOUBLE LINE THUS
GT VAVGINLOAD DAVGINLOAD TAVGINLOAD   DAVGADMIT   TAVGADMIT VAVGTHRUPUT
DAVGTHRUPUT TAVGTHRUPUT
  FOR EACH GATEWAY
   PRINT 1 DOUBLE LINE WITH GATEWAY,
    GCOUNT.VOICE.INLOAD(GATEWAY)/(.TIMELIMIT-.STARTPT),
```

```
    GCOUNT.DATA.INLOAD(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.INLOAD(GATEWAY)+GCOUNT.VOICE.INLOAD(GATEWAY))/
         (.TIMELIMIT-.STARTPT),
    GCOUNT.DATA.ADMITLD(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.ADMITLD(GATEWAY)+GCOUNT.VOICE.INLOAD(GATEWAY))/
         (.TIMELIMIT-.STARTPT),
    GCOUNT.VOICE.THRUPUT(GATEWAY)/(.TIMELIMIT-.STARTPT),
    GCOUNT.DATA.THRUPUT(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.THRUPUT(GATEWAY)+GCOUNT.VOICE.THRUPUT(GATEWAY))/
         (.TIMELIMIT-.STARTPT) THUS
** ********** ********** ********** ********** ********** **********
********** **********
  PRINT 1 DOUBLE LINE THUS
GT VAVGINLOAD DAVGINLOAD TAVGINLOAD   DAVGADMIT   TAVGADMIT VAVGTHRUPUT
DAVGTHRUPUT TAVGTHRUPUT
  FOR EACH GATEWAY
   PRINT 1 DOUBLE LINE WITH GATEWAY,
    GCOUNT.VOICE.IN(GATEWAY)/(.TIMELIMIT-.STARTPT),
    GCOUNT.DATA.IN(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.IN(GATEWAY)+GCOUNT.VOICE.IN(GATEWAY))/
         (.TIMELIMIT-.STARTPT),
    GCOUNT.DATA.ADMIT(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.ADMIT(GATEWAY)+GCOUNT.VOICE.IN(GATEWAY))/
         (.TIMELIMIT-.STARTPT),
    GCOUNT.VOICE.SUCCESS(GATEWAY)/(.TIMELIMIT-.STARTPT),
    GCOUNT.DATA.SUCCESS(GATEWAY)/(.TIMELIMIT-.STARTPT),
    (GCOUNT.DATA.SUCCESS(GATEWAY)+GCOUNT.VOICE.SUCCESS(GATEWAY))/
         (.TIMELIMIT-.STARTPT) THUS
** ********** ********** ********** ********** ********** **********
********** **********
        STOP
END
```

```
''*************************************************************
''*                                                           *
''*   THIS ROUTINE ADJUSTS THE WINDOW SIZE OF THE EGTEG FLOW  *
''*  CONTROL MECHANISM.                                       *
''*                                                           *
''*************************************************************

ROUTINE UPDWI GIVEN SRC, DST, INDEX
    LET WINDOW(SRC,DST) = WINDOW(SRC,DST) + INDEX
    IF WINDOW(SRC,DST) < 0 OR WINDOW(SRC,DST) > .WINDOWSIZE
       PRINT 1 LINE WITH SRC, DST, WINDOW(SRC,DST), TIME.V THUS
WIFLOW SRC * DST * WI ** TIM ******.**
    ALWAYS
END
```

```
''*****************************************************************
''*                                                               *
''*   THIS ROUTINE GATHERS STATISTICS REGARDING TO THE ACCEPTANCE *
''* TIMES OF VARIOUS TYPES OF PACKETS.                            *
''*                                                               *
''*****************************************************************

ROUTINE SUCRTN GIVEN PACKNO,DELAY,PACKMODE
   DEFINE DELAY AS A REAL VARIABLE
   LET INDEX = HOPCOUNT(PACKNO)
   IF PACKMODE = 1
      LET HOPDELAY.D(INDEX) = DELAY
      LET HOPSUC.D(INDEX) = HOPSUC.D(INDEX) + 1
   ELSE
      IF PACKMODE = 2
         LET HOPDELAY.V(INDEX) = DELAY
         LET HOPSUC.V(INDEX) = HOPSUC.V(INDEX) + 1
      ELSE
         IF PACKMODE = 3
            LET HOPDELAY.C(INDEX) = DELAY
            LET HOPSUC.C(INDEX) = HOPSUC.C(INDEX) + 1
         ELSE
            IF PACKMODE = 4
               LET HOPSESSDELAY(INDEX) = DELAY
               LET HOPSESSACC(INDEX) = HOPSESSACC(INDEX) + 1
            ALWAYS
         ALWAYS
      ALWAYS
   ALWAYS
   RETURN
END
```

```
''*****************************************************************
''*                                                               *
''*   THIS ROUTINE GATHERS STATISTICS REGARDING TO THE REJECTION  *
''* TIMES OF VARIOUS TYPES OF PACKETS.                            *
''*                                                               *
''*****************************************************************


ROUTINE REJRTN GIVEN PACKNO, PACKMODE
   DEFINE DELAY AS A REAL VARIABLE
   LET INDEX = HOPCOUNT(PACKNO)
   IF PACKMODE = 1
      LET HOPREJ.D(INDEX) = HOPREJ.D(INDEX) + 1
   ELSE
      IF PACKMODE = 2
         LET HOPREJ.V(INDEX) = HOPREJ.V(INDEX) + 1
      ELSE
         IF PACKMODE = 4
            LET HOPSESSREJ(INDEX) = HOPSESSREJ(INDEX) + 1
         ALWAYS
      ALWAYS
   ALWAYS
   RETURN
END
```

```
''************************************************************
''*                                                          *
''*   THE FOLLOWING ROUTINES ARE USED FOR DEBUGGING PURPOSES.  *
''*                                                          *
''************************************************************

ROUTINE TRACE1 GIVEN MSGTXT, PACKNO, GATENO

  DEFINE MSGTXT AS A TEXT VARIABLE
PRINT 1 DOUBLE LINE WITH MSGTXT,GATENO,PACKNO,SEQID(PACKNO),
    TIMEID(PACKNO),TIME.V,SRCNO(PACKNO),DSTNO(PACKNO),
    PACKCLASS(PACKNO),RETRYTIMES(PACKNO) THUS
******* GT * PACK ******* SEQ ******.* TID ******.*** TIM ******.***
S * D * C * R *
END


ROUTINE TRACE2 GIVEN MSGTXT, PACKNO, GATENO, SESSION

  DEFINE MSGTXT AS A TEXT VARIABLE
PRINT 1 DOUBLE LINE WITH MSGTXT,GATENO,PACKNO,
SESSSEQ(SESSION),SESSTID(SESSION),TIME.V,
SESSSRC(SESSION), SESSDST(SESSION), RETRYTIMES(PACKNO),
SESSTAG(SESSION),SESSMODE(SESSION),ARRIVALTIME(PACKNO) THUS
******* GT * PACK ******* SSQ ******.* SID ******.*** STM ******.***
S * D * R * STAG ******** SMOD * ARR ******.***
END


ROUTINE TRACE3(MSGTXT,PERIOD,GATENO,SESSION)

DEFINE MSGTXT AS A TEXT VARIABLE
PRINT 1 DOUBLE LINE WITH MSGTXT, GATENO,PERIOD,
SESSSEQ(SESSION),SESSTID(SESSION),TIME.V,
SESSSRC(SESSION),SESSDST(SESSION),SESSTAG(SESSION),
SESSMODE(SESSION) THUS
******* GT * PERD ******* SSQ ******.* SID ******.*** STM ******.***
S * D *    STAG ******** SMOD *
END
```

```
ROUTINE TRACE4 GIVEN MSGTXT, PACKNO, GATENO

  DEFINE MSGTXT AS A TEXT VARIABLE
PRINT 1 DOUBLE LINE WITH MSGTXT,GATENO,PACKNO,SEQID(PACKNO),
    TIMEID(PACKNO),TIME.V,SRCNO(PACKNO),DSTNO(PACKNO),
    PACKCLASS(PACKNO),RETRYTIMES(PACKNO),BUF(GATENO),
    BUF.IN(GATENO),BUF.VOICE(GATENO),BUF.TRAN(GATENO) THUS
******* GT * PACK ******* SEQ ******.* TID ******.*** TIM ******.***
S * D * C * R * BUF ** BUFIN ** BUFVO ** BUFT **
END
```

REFERENCES

[1] Burg, F., Chen, C., Folts, H.
Of Local Networks, Protocols, and The OSI Reference Model.
Data Communications, November, 1984.

[2] Callon, Ross.
Internetwork Protocol.
In *Proceedings of IEEE*, pages 1388-1393. IEEE, December, 1983.

[3] Clark, D., Pogran, K., Reed, D.
An Introduction to Local Area Networks.
Proc. of The IEEE 66(11):1497-1517, November, 1978.

[4] Cohen, D.
A Protocol for Packet-Switching Voice Communication.
Computer Networks :320-331, 1978.

[5] Cohen, D.
Packet Communication of Online Speech.
National Computer Conference :169-176, 1981.

[6] Danthine, A. S.
Network Interconnection.
Proceedings of IFIP TC6 on Local Computer Networks :289-308, April, 1982.

[7] Folts, H.
A Tutorial on The Open Systems Interconnection Reference Model.
Open Systems Data Transfer :2-21, June, 1982.

[8] Forgie, J.
Speech Transmission in Packet-Switched Store-and-Forward Networks.
National Computer Conference :137-141, 1975.

[9] Freeman, H., Thurber, K. (editor).
Local network Standards.
Local Network Equipment, IEEE Computer Society Press, 1985.
PP. 25-30.

[10] Fultz, G.
Adaptive Routing Techniques for Message Switching Computer Communication Networks.
Technical Report, UCLA School of Engineering and Applied Science, Report No. UCLA-ENG-7252, July, 1972.

[11] Gerla, M., Kleinrock, L.
Flow Control: A Comparative Survey.
IEEE Transactions on Communications :553-574, April, 1980.

[12] Hawe, B., Kirby, A., Stewart, B.
Transparent Interconnection of Local Area Networks with Bridges.
Journal of Telecommunication Networks 3(2):116-130, 1984.

[13] Hoberecht, W.
A Layered Network Protocol for Packet Voice and Data Integration.
IEEE Journal on Selected Areas in Communications
SAC-1(6):1006-1013, December, 1983.

[14] Kamoun, K.
A Drop and Throttle Flow Control Policy for Computer Networks.
IEEE Transactions on Communications :444-452, April, 1981.

[15] Kleinrock, L., Kamoun, F.
Hierarchical Routing for Large Networks.
Computer Networks :155-174, January, 1977.

[16] Lam, S., Reiser, M.
Congestion Control of Store-and-Forward Networks by Input Buffer Limits - An Analysis.
IEEE Transactions on Communications COM-27:127-134, January, 1979.

[17] Lam, S., Lien, Y.
Congestion Control of Packet Communication Networks by Input Buffer Limits - A Simulation Study.
IEEE Transactions on Computers C-30:733-742, October, 1981.

[18] Lampson, B., Paul, M., Sieg, H.
Distributed Systems: Architecture and Implementation.
Springer-Verlag, 1983.

[19] Lissack, T., Maglaris, B., Frisch, I.
Digital Switching in Local Area Networks.
IEEE Communication Magazine :26-37, May, 1983.

[20] Liu, M., Hilal, W., Groomes, B.
Performance Evaluation of Channel Access Protocols for Local Computer Networks.
IEEE Proceedings of the COMPCON Fall 82 Conference :417-426, 1982.

[21] NBS Special Publication 500-96.
The Selection of Local Computer Networks.
PP. 35-50, 1982.

[22] O'Leary, G.C., Blankenship, P.E., Tierney, J., Feldman, J.A.
A Modular Approach to Packet Voice Terminal Hardware Design.
National Computer Conference :183-189, 1981.

[23] Pitt, D.
Current and Future Medium Access Control Standards.
Proc. IEEE INFOCOM :319-322, April, 1986.

[24] Postel.
Internetwork Protocol Approaches.
IEEE Transactions on Communications :604-611, April, 1980.

[25] Saltzer, J.H., Reed, D.P. and Clark, D.D.
Source Routing For Campus-Wide Internet Transport.
Proceeding of IFIP Working Group 6.4 International Workshop on Local Networks :1-23, August, 1980.

[26] Smith, D.
Digital Transmission Systems.
Van Nostrand Reinhold Co., 1985.

[27] Stack, T., Dillencourt, K.
Protocols for Local Area Networks.
IEEE Proceedings of the Trends and Applications Conference :83-93, May, 1980.

[28] Stallings, W.
Local Network Overview.
Signal magazine :39-44, January, 1983.

[29] Stallings, W.
IEEE Project 802.
Computerworld, February, 1984.

[30] Tanenbaum, A.
Computer Networks.
Prentice Hall, 1981.
Chap. 5.

[31] Weinstein, C., Forgie, J.
Experience with speech Communication in Packet Networks.
IEEE Journal on Selected Areas in Communications
SAC-1(6):963-980, December, 1983.