NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REQUIREMENTS ANALYSIS FOR THE OPERATION
OF A REAL-TIME WARFARE SIMULATION
OVER A PACKET SWITCHED COMPUTER NETWORK

by

Jeffrey L. Paige

December 1986

Thesis Advisor: Mitchell L. Cotton

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) code 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
REQUIREMENTS ANALYSIS FOR THE OPERATION OF A REAL-TIME WARFARE SIMULATION OVER A PACKET SWITCHED COMPUTER NETWORK.

12 PERSONAL AUTHOR(S)
Paige, Jeffrey L.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 December	112

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	computer networks, packet switching, Disnet, network protocols, network analysis, wargaming.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis investigates the problems associated with operating a real-time warfare simulation system over an ARPANET-based packet switched network. The network throughput requirements of the simulation are determined from measurements taken over a local area network. The performance of the packet switched network is analyzed through the use of a switching node model, resulting in a graph of application throughput as a function of the internal network traffic. The throughput requirements are compared to this function, and maximum acceptable levels of internal traffic are determined. The effects of other aspects of packet switching are discussed including traffic dynamics, packet routing, and flow control. The results suggest that it may be possible to conduct a very (next page)

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Mitchell L. Cotton	22b TELEPHONE (Include Area Code) (408) 646-2377 22c OFFICE SYMBOL code 62Cc

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

19. (cont.) restricted warfare simulation over this network. A better networking solution may be to use dedicated networking resources instead of packet switching.

Approved for public release; distribution unlimited.

Requirements Analysis for the Operation of a Real-Time Warfare Simulation Over a Packet Switched Computer Network

by

Jeffrey Lloyd Paige
Lieutenant, United States Navy
B.S., Ohio State University, 1980

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1986

ABSTRACT

This thesis investigates the problems associated with operating a real-time warfare simulation system over an ARPANET-based packet switched network. The network throughput requirements of the simulation are determined from measurements taken over a local area network. The performance of the packet switched network is analyzed through the use of a switching node model, resulting in a graph of application throughput as a function of the internal network traffic. The throughput requirements are compared to this function, and maximum acceptable levels of internal traffic are determined. The effects of other aspects of packet switching are discussed including traffic dynamics, packet routing, and flow control. The results suggest that it may be possible to conduct a very restricted warfare simulation over this network. A better networking solution may be to use dedicated network resources instead of packet switching.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

This thesis is an investigation into a problem of applied computer network analysis. The Naval Ocean Systems Center (NOSC) has developed and maintained a real-time, interactive, warfare simulation known as the Interim Battle Group Tactical Trainer (IBGTT). NOSC has been able to operate this system in a distributed manner over a local area network, but it has never been operated over a wide area network. Current planning is for this system to be networked using the Defense Integrated Secure Network (DISNET), a wide area, packet switched network based on the ARPANET architecture. This thesis attempts to determine whether this packet switched network can be reasonably expected to provide all levels of performance required by IBGTT. This problem will be approached in three phases.

First, the network requirements of the IBGTT system will be determined. In doing this, the highest priority will be given to preserving the user characteristics of IBGTT. That is, a user should not be able to tell from the operation of the system that the simulation is being conducted over a wide area network. Any network solution which requires a loss of user function will be considered sub-optimal.

Second, the expected performance of the packet switched network will be determined. This will primarily involve performing delay and throughput analysis on the ARPANET architecture. To achieve some quantitative estimates of network performance it is necessary to model both the network and the application in a manner that makes the problem soluble. The

assumptions necessary to this model will be discussed in detail.

Finally, the networking requirements of IBGTT will be compared to the predicted performance of DISNET. This comparison is complicated, in that it requires a consideration of nearly every aspect of the operation of a packet switched network. While the complexity of these issues make it difficult to provide absolute answers, it is hoped that this discussion can at least point out what the important issues are.

II. APPROACH TO THE PROBLEM

A. PROBLEM CONSTRAINTS

1. Application

The basic characteristics of the Interim Battle Group Tactical Trainer (IBGTT) will be considered invariable for the purpose of performing analysis. These include both the user characteristics and the technical characteristics. In fact, in actual operation it may be possible to limit some of the user characteristics in order to make networking feasible, but this will not be considered a very desirable solution. The primary concern of this thesis is to determine what level of performance is required to operate the IBGTT system over the network with all of its user characteristics intact.

2. Interconnecting Network

The network performance analysis will only consider the use of the Defense Integrated Security Network (DISNET) as the interconnecting network, as this is the only system that is currently approved by the Department of Defense (DOD) for secure computer communications. Since DISNET is in the process of being implemented, this analysis will be based on information about existing networks which use the same architecture, ARPANET and MILNET. No other types of packet switched networks (such as CCITT) will be analyzed.

3. Application-Network Interfaces

The interface between the application and the existing packet switched network will not be considered to be constrained to any currently existing

configurations. For the purpose of this analysis, this interface will be assumed to be configured such that maximum data throughput is obtained. This includes both the software interface between IBGTT and the network protocols, and the physical connections between the application computers (hosts) and the network packet switches (nodes).

In the case of both software and hardware interfaces, it is reasonable to assume an optimum configuration for two reasons. First, neither of these interfaces is well defined. In the case of the software, meaningful documentation is scarce. In the case of the hardware, the actual interface specification seems to be constantly changing. Second, these interfaces are the easiest and cheapest components of the system to change. Considering the software, the data output driver can be changed without modifying the application software itself. Similarly, if the initial hardware used to connect is found to be inadequate it can be replaced with higher capacity hardware for relatively low cost.

B. ANALYSIS PROCEDURES

Two different aspects of the proposed system will be analyzed in this thesis. First, the throughput requirements of the warfare simulation will be determined. Second, the expected throughput of the packet switched network will be projected through the use of a network model. After this is done, the required throughput will be compared to the expected network performance.

1. Application Throughput Requirements

The IBGTT system is currently operating in the distributed mode over an ETHERNET local area network at

NOSC. Using DECNET, measurements of the amount of data being sent between computers during the simulation can be made. These measurements have been obtained from NOSC for different operating conditions, resulting in some reliable values for the required data throughput of the system.

However, before this data can be transmitted over a packet switched network it has to be packaged by a series of network protocols. Each of these protocols adds some information to the data in the form of packet headers and trailers. This protocol overhead will be enumerated and added to the required data throughput to obtain measures of the total throughput required by the IBGTT system for normal operation. For a network to permit unconstrained operation of the system it must be capable of handling this throughput requirement.

2. Network Performance Analysis

Expected network performance will be analyzed through the use of a switching node model. Using this model and knowing much about the behavior of the application process, it is possible to derive an expression which relates the waiting time of an application output packet to the dynamics of the network's internal traffic. If certain assumptions are made about the internal traffic, this expression can be explicitly solved. The waiting time equation is the starting point for the remaining analytical procedures.

It is possible to obtain an expression from the waiting time equation which relates traffic stability at a node to the arrival rates of the internal network traffic and the external application traffic. Using this stability equation, a critical rate can be defined in terms of these arrival rates. From the critical rate expression, an equation which defines the maximum

(critical) application arrival rate as a function of the internal traffic is obtained. From this, a curve showing application throughput as a function of internal network load will be generated.

This relationship between application throughput and network load will be used as the basis for determining under what conditions of network operation will DISNET be capable of meeting the networking needs of IBGTT.

III. INTERIM BATTLE GROUP TACTICAL TRAINER

A. USER CHARACTERISTICS

For the purpose of this thesis, it is not necessary for a complete description of the user characteristics of the application program to be included. Therefore, only those features which have a direct impact on the ability to operate over a packet-switched network will be discussed below. Additional information on the operating features of the program can be found in the system's user guides [1], [2].

1. Real-Time Simulation

Possibly the most important feature of the Interim Battle Group Tactical Trainer (IBGTT) from the perspective of the user is the fact that the game runs in real-time. This means that the user has the same amount of time to act or react to tactical situations as he would in a real conflict. For example, if approaching aircraft were expected to be within the user's weapons range in two minutes then the user would only have two minutes of actual time to take action. It is this feature of real-time operation which makes IBGTT an effective training device.

2. Different Operating Speeds

A second important user feature is the capability to run the game faster than real-time. While this may appear to contradict the real-time feature, the two actually work together to produce an effective training environment.

During a large-scale battle problem a great deal of time may be spent searching for the opponent's forces. If the problem were run at real-time the

participants might spend many hours waiting for the enemy to be detected. By allowing the game to run faster than real-time it is possible to shorten the detection phase of the problem. Once the actual conflict begins the warfare simulation can be run at real-time from that point on. During tactical training the simulation would not normally be operated at a speed slower than real-time, for the reasons already stated.

3. Opportunity for Large Scenarios

In order for the training to be realistic, the magnitude of the battle problems constructed should not be drastically scaled down from those anticipated in real life. For this reason IBGTT is capable of including a large number of combatant units in its scenarios, as many as 1500 objects.

4. Multiple Users with Different Perspectives

The IBGTT system is designed to simultaneously support each of the separate warfare commanders under the Navy's CWC concept. This means that while one user has access to all the information necessary to perform the functions of the anti-air warfare commander (AAWC), the user at the next station might need the information necessary to act as the anti-submarine warfare commander (ASWC). Thus, IBGTT must be able to present and maintain different status boards and displays to each of the different user stations.

Another important capability of the system is that of supporting one group of users as the Blue forces and another group as the Orange forces. A third person or group may need to be supported at a Control station, with access to virtually all the information so they can monitor and direct the simulation.

B. TECHNICAL CHARACTERISTICS

1. Distributed Architecture

The IBGTT system has been developed so that it can be operated in either of two ways: merged or distributed. In the merged form the system software is executed on a single, stand-alone computer. Thus, all the system functions described below are performed on that computer. This is how the system is currently operated at the Naval Postgraduate School (NPS).

In the distributed form of IBGTT the system functions are divided into two groups: user interface functions, and simulation execution. The user interface function is performed by one or more computer systems called Remote Site Modules (RSM). The simulation execution is performed by a single computer system called the Computer Support Facility (CSF). The IBGTT system is currently operated in the distributed form at NOSC using an ETHERNET to connect the CSF and RSM computer systems*.

a. Remote Site Module

System users interact with IBGTT through the RSM. A simple block diagram of an RSM is shown in Figure 1. As shown, an RSM includes a digital computer, 2 color graphics controllers, 4 user stations (2 per graphics controller), and remote communications equipment. This description of an RSM is based on the system configurations currently in use at NOSC [3]. It is quite possible that future RSMs at other sites will be physically different from this, but they should still be functionally equivalent to this description.

---

*The previous designations for these systems were User Support System (USS) for the RSM and Simulation Support System (SSS) for the CSF. Some references cited in this thesis use the previous designations.

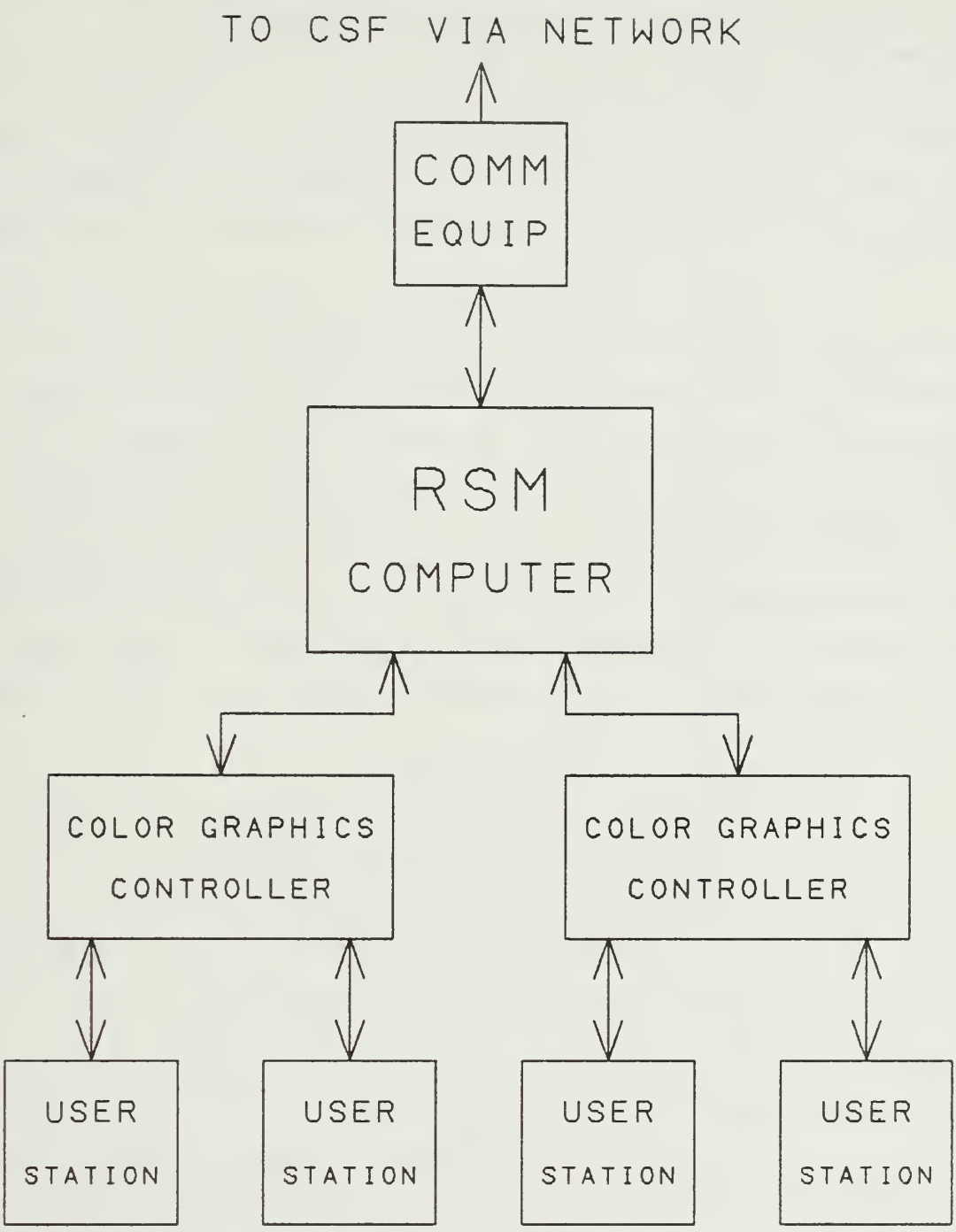

Figure 1. Remote Site Module System Diagram

The RSM functions by accepting orders from the users and transmitting them to the CSF for processing. After CSF processing, the RSM receives the updated simulation status from the CSF and uses it to update its local copy of the simulation database. The RSM uses the information in this database to generate and update the different tactical displays at the user stations.

It is possible for the IBGTT system to be configured with more than one RSM for large battle problems. For example, one RSM might be used by the Blue forces and the other by the Orange forces. In a case such as this, each RSM would only be responsible for maintaining the tactical displays appropriate to its users. However, an RSM responsible for maintaining the Control station would need to maintain nearly all the tactical information in its database. Regardless of how many RSMs are configured there would only be one CSF in the system.

b. Computer Support Facility

The CSF is that part of the IBGTT system that actually executes the warfare simulation. A simple block diagram of a CSF is shown in Figure 2. As shown, the major components of a CSF include a digital computer, disk storage, magnetic tape drive, line printers, video display terminals, and remote communications equipment. As before, this describes a CSF as currently configured at NOSC [4].

The CSF receives user commands from the RSM and processes them by calculating the updated status of the simulation. The simulation status is updated every simulation minute, called a game cycle. These game cycle updates are performed whether or not any commands are issued by users [1:I-1-2]. When the simulation is

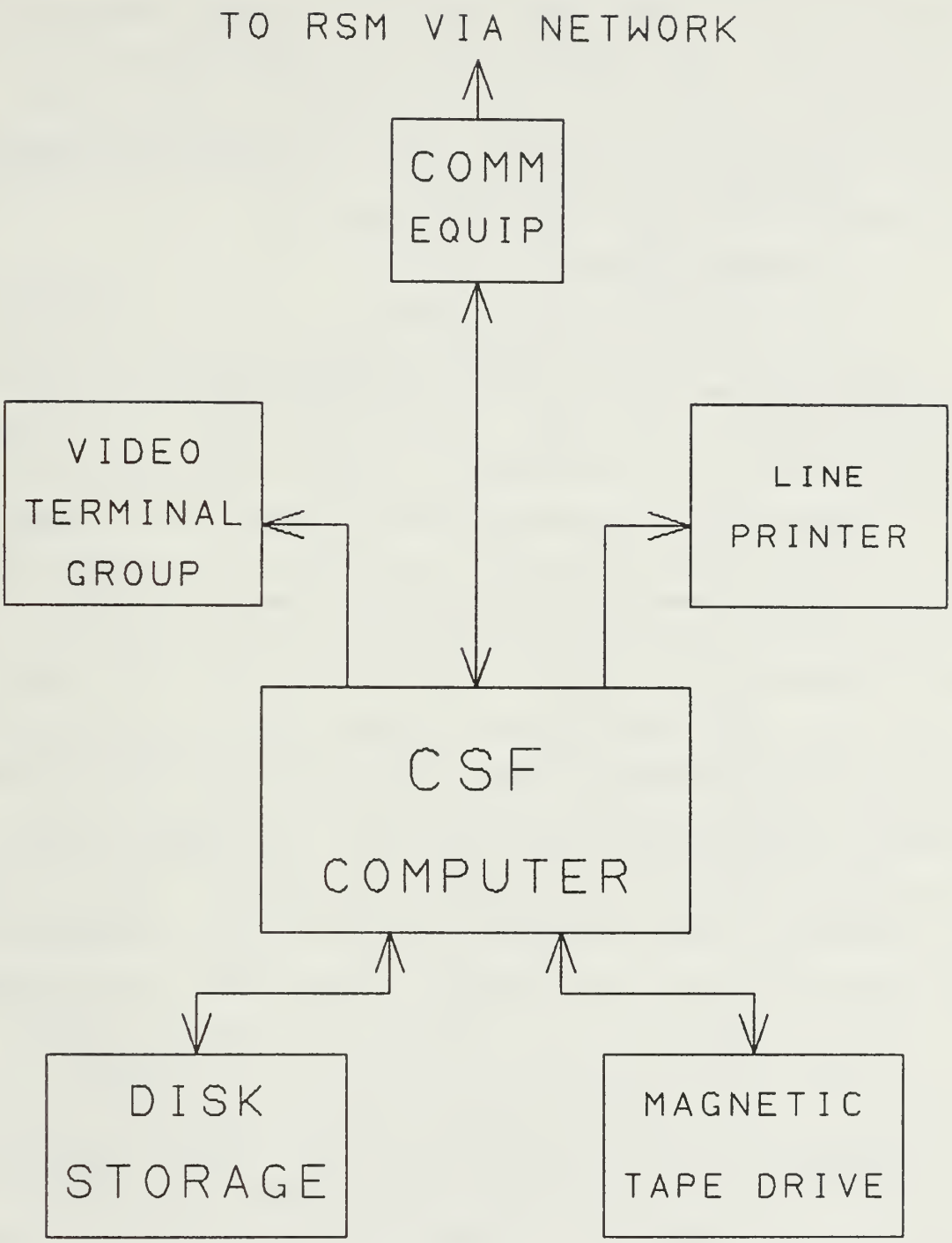

Figure 2. Computer Support Facility System Diagram

operating faster than real-time the CSF will perform two or more of these game cycles for every minute of actual time. The updated simulation status is maintained in the CSF database, which is the primary database for the simulation. The appropriate changes to the database are transmitted to the RSM for display processing and local database updating.

2. Software Design

a. Database Concurrency

As described above, the distributed IBGTT system performs the actual simulation execution on a single computer, the CSF, regardless of the size of the battle problem or the number of users participating. This means that the system should appear to the user to be a distributed processor, but that it is actually a centralized processing system with remote display processing. The distinction is very important, as explained below.

The IBGTT system creates and maintains a simulation database, called the blackboard, in the form of data tables [5:4-5]. Each object in the problem, such as an aircraft or a ship, has a separate table maintained for it in the CSF's blackboard. This table contains all the information on that object available to the simulation. An example of an object data table is shown in Figure 3. As shown, the table includes both fixed information (such as type of unit) and variable information (such as course, speed, latitude, longitude). A copy of each table can reside locally in the RSM database, but it is only a copy--the true status of the object is determined by the table in the CSF database.

When a system user enters commands at the RSM (such as changing the course of a ship) nothing is

TABLE: OWN "Own Force"

DESCRIPTION: "Contains own force reported position, status, etc.."

FORMAT

Field	Type	Word	Pos	Size	Range
UNIT	I	1	0	16	0-1600
TYPE	I	11	9	4	0-15
STATUS	I	11	13	4	0-15
MISSION	I	11	17	5	0-31
LATITUDE	F	2	0	word	-90to+90
LONGITUDE	F	3	0	word	-PIto+PI
COURSE	I	4	9	9	0-359
SPEED	I	8	10	12	0-4095
ALTITUDE	I	4	20	11	0-2047
DEPTH	I	4	20	11	0-2047

Figure 3. Example of an Object Data Table

done directly to the tables held in RSM memory. The commands are transmitted to the CSF and processed. After processing, the tables held in the CSF are changed to reflect the current status of the simulation. The table changes are then transmitted back to the RSM so that they can be displayed and the local database updated [5:18-20].

The apparent advantage of this software design is that database concurrency is maintained by only allowing table changes to be made in the CSF. The disadvantage is that a very large amount of data is frequently being transmitted from the CSF to the RSM. The CSF will update the status of the simulation every game cycle whether or not the users have entered any new commands. This results in frequent transmissions of new table changes to the RSM because the objects which are in motion (such as ships and aircraft) will have some of their table fields constantly changing.

b. Data Extraction

Anticipating the need to reduce the quantity of data transmitted from the CSF to the RSM, NOSC has developed a data extractor program which will act to filter the data before it is sent to the network [5:17]. The extractor works by testing the output buffer before it puts the most recent update onto the output queue. If the output buffer is nearly full (currently the threshold is set at 95%) then the extractor program will automatically begin to prioritize the data before it goes on the queue. High priority (data must be transmitted) will be given to any data required by an active user display. Data that is not currently being viewed by a user is given low priority and may not be sent at all.

While the extractor program may be successful in reducing the amount of data sent from the CSF to the RSM, it does so at the cost of a loss of user function. If the game were to operate under this system for many game cycles, then the problem of an out of date database at the RSM occurs. Those parts of the database needed to support the active user displays are up to date with the CSF, but those parts relating to displays which have not been viewed for many game cycles may contain information which is completely incorrect (such as listing units which have been destroyed). This problem will impact on the user when a new display is activated: the information in the new display will not be current until the next machine update is completed. Since the computer processing involved in IBGTT is supposed to be transparent to the user, this will create distracting and potentially confusing conditions.

For these reasons use of the extractor program will be considered a sub-optimal, compromise solution to the networking problem. The requirements analysis in this thesis will be performed toward the goal of maintaining a completely up to date RSM database.

C. NETWORK OPERATION OF IBGTT

1. Local Area Network Operation

The IBGTT system has been in operation over a local area network (LAN) at NOSC for quite a while. The NOSC LAN consists of an ETHERNET which is interfaced by the VAX computers through the use of DECNET. A diagram of the LAN is shown in Figure 4. While the system has been operated successfully over an ETHERNET, that does not mean that the transition to the

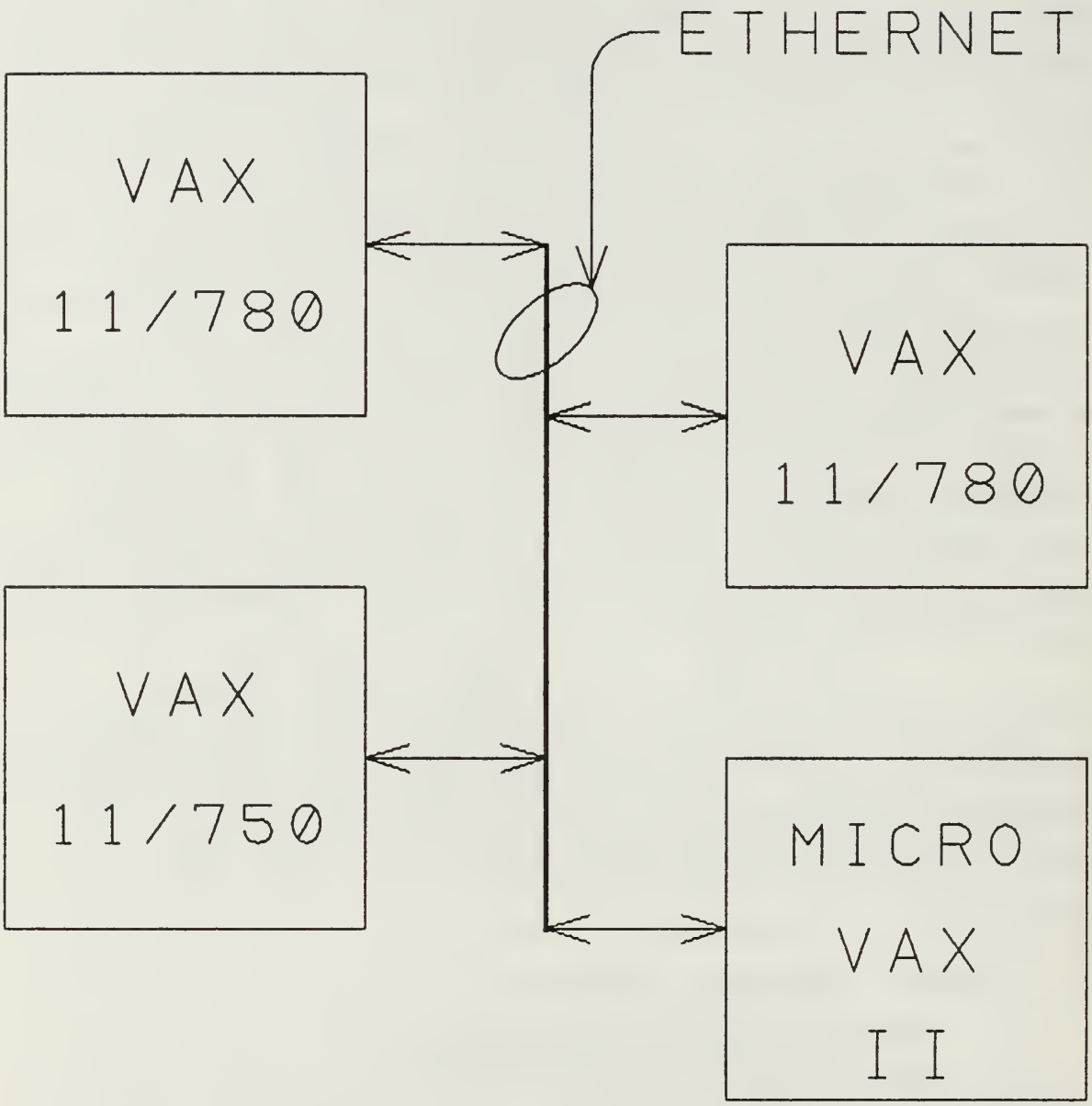

Figure 4. IBGTT Local Area Network at NOSC

proposed packet switched network will be easy. The ETHERNET LAN has a very high capacity, rated on the order of 10 Mbps. The possible capacity of an ARPANET-based packet switched network is less than this by more than two orders of magnitude. Thus, changing from an ETHERNET to packet switching involves more than a simple change in software and wiring.

2. Proposed Wide Area Network

The Joint Directors of Laboratories (JDL), a DOD agency, has directed that IBGTT be used to implement a wide area warfare simulation network, called Simulation Network or SIMNET [6:56-59]. Initially, there are to be four hosts on SIMNET, two located on the east coast and two on the west coast. These hosts are listed below in Table I.

TABLE I.
INITIAL HOSTS ON SIMNET

Host Command	Location	Computer
Naval Postgraduate School (NPS)	Monterey, CA	DEC VAX 11/780
USA Communications and Electronics Command (CECOM)	FT Monmouth, NJ	DEC MicroVAX II
USAF Rome Air Development Center (RADC)	Rome, NY	DEC MicroVAX II
Naval Ocean Systems Center (NOSC)	San Diego, CA	ETHERNET LAN connection made to a VAX 11/780

A diagram of SIMNET is shown in Figure 5. As shown, the actual topology of the interconnecting network has not been defined. The current plan for

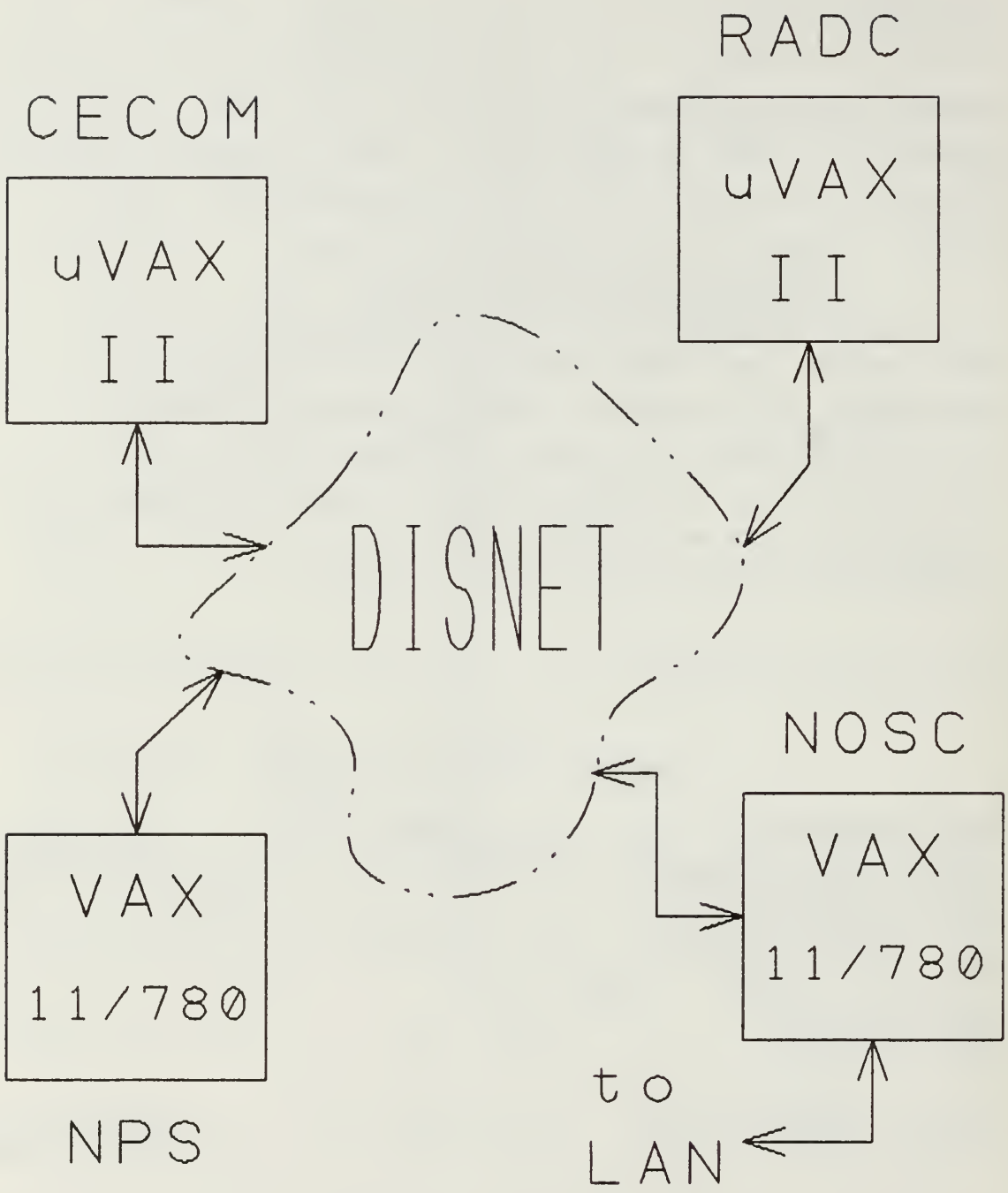

Figure 5. Block Diagram of the Proposed Simulation Network

implementing SIMNET is that it will be a logical subnet on the Defense Integrated Secure Network (DISNET). DISNET, a new packet switched network based on the ARPANET model, will be described in detail in the next section. It is important to understand that DISNET will be providing network services to numerous systems and customers other than SIMNET. Thus, a key question for this thesis is whether or not DISNET can provide the network performance required by SIMNET while providing services to the rest of its customers.

IV. INTERCONNECTING NETWORK

A. DEFENSE INTEGRATED SECURE NETWORK

1. Network Security

The Defense Integrated Secure Network (DISNET) is a wide area, packet switched network which is designed to meet the security needs of the Department of Defense. DISNET is a component of the Defense Data Network (DDN), a system which also includes the MILNET (Military Network).

Both DISNET and MILNET are based on the ARPANET (Advanced Research Projects Agency Network) architecture. However, DISNET will be physically separate from ARPANET and MILNET--no gateways will exist between DISNET and any other network for years. This physical separation, along with physical security at each computer installation, will permit DISNET customers to transmit classified information over the network. Classified data transfers are not permitted over the unclassified packet switched networks ARPANET and MILNET. The eventual goal of DDN is to provide gateways between DISNET and MILNET once special encryption hardware becomes available [7:16-19].

2. Use of the ARPANET Model

Because the DISNET design follows the ARPANET design very closely, both in software and hardware, this thesis will make extensive use of the ARPANET literature in its discussion of the DISNET architecture. In fact, the development of DISNET is so recent that there is virtually no technical literature available on it.

However, the relationship between performance measurements made on the ARPANET and the expected performance of DISNET is less clear. In 1984 the original ARPANET was divided into two networks, ARPANET and MILNET. ARPANET is now a research oriented network which is heavily used to transfer data among a number of interconnected networks. MILNET is now used as the primary computer network for unclassified operational military use. It is possible that as customers connect to DISNET over the next few years its traffic measurements will eventually resemble those currently found on MILNET. For this reason, MILNET measurements will be used as the basis for network performance analysis in this thesis. It is hoped that the results will be a good predictor of eventual conditions on DISNET.

B. NETWORK ARCHITECTURE

1. Packet Switching

The ARPANET architecture is designed to support computer communications through the use of packet switching. In a packet switching network information is transferred between computers in the form of discrete blocks of fixed maximum length, called packets. The network consists of two major components: packet switches (called nodes) and connecting trunk lines. The packet switches are minicomputers which are specially designed to perform this task. On ARPANET these packet switches are known as Interface Message Processors (IMPs). For trunk lines, ARPANET uses leased phone lines which typically have 50 Kbps capacity, though there are some lines with other capacities (9.6 Kbps and 56 Kbps). The computers which are physically connected to the network for the purpose

of using the network's services are called hosts. A diagram of the ARPANET model is shown in Figure 6.

A design requirement of the ARPANET is that there must be at least two possible paths between any two nodes. These multiple paths allow the network to send a computer's packets over the best available path, normally the path with the least delay. Often the information being sent between two hosts, such as a program or document file, consists of more than one packet. In these cases the network may send the separate packets over different paths to minimize network congestion on any one path. This is possible because each packet contains the destination address, enabling the network to treat them as independent units.

2. Interface Message Processors

Most of the critical functions of the ARPANET are performed at the IMPs, including message disassembly, reassembly, and message and packet queuing. As stated above, a packet on the ARPANET can not exceed a fixed length limit. However, it is possible for a host to transmit a message to the network which is longer than the maximum packet length. It is the responsibility of the IMP to which the host is connected to disassemble the message into several packets. The IMP which performs this function is referred to as the source node. The maximum number of packets which can be generated by a single message is eight, creating a limit on the maximum length of a host message.

The reverse of this process is message reassembly, which is performed by the IMP directly connected to the destination host. This IMP is referred to as the destination node. As stated above,

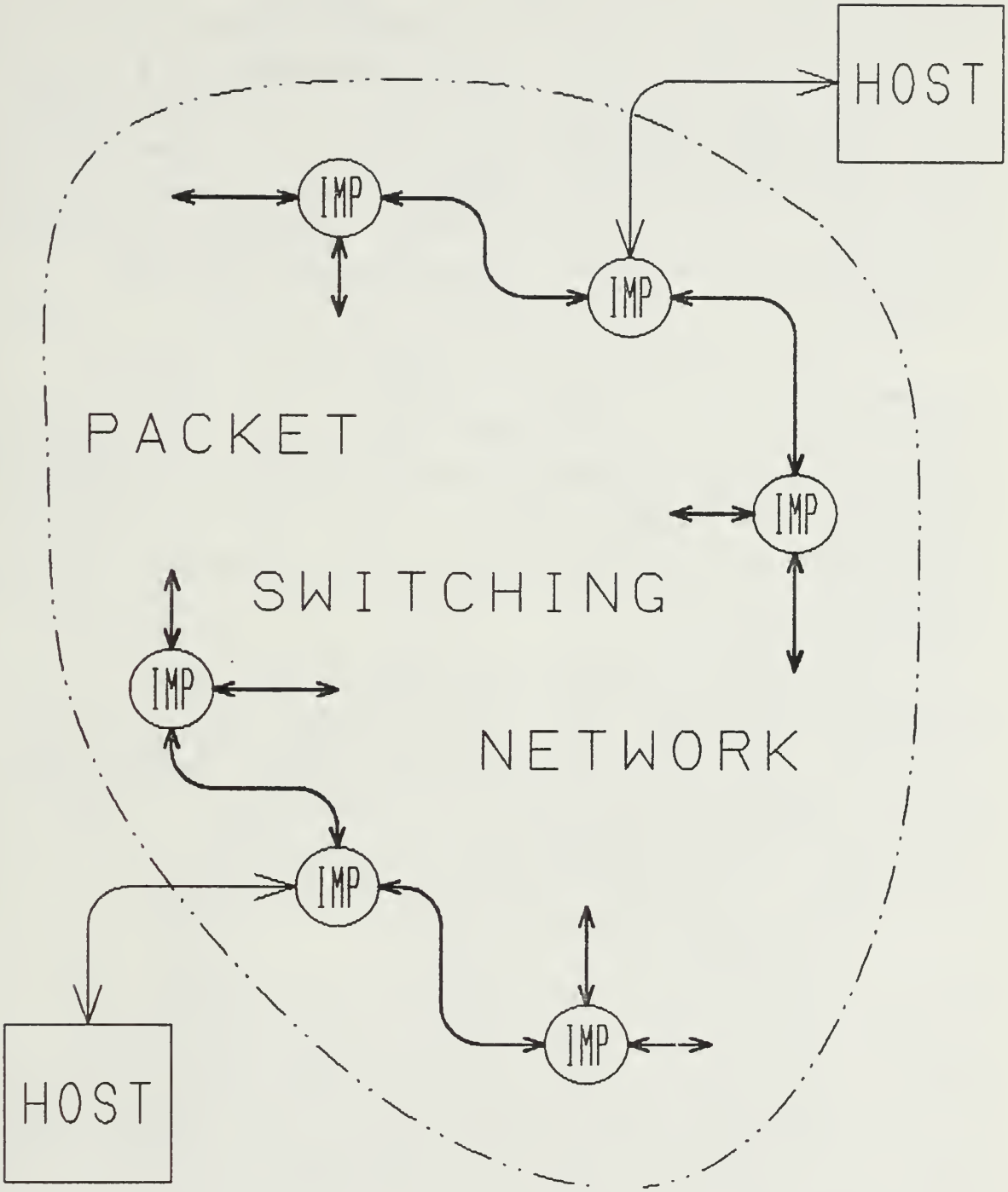

Figure 6. ARPANET Model of a Packet Switching Network

it is possible for the component packets of a single message to traverse different paths on their way to the destination node. Thus, these packets may arrive out of order. It is the responsibility of the destination node to collect the separate packets and to use them to reconstruct the original message. This message is then delivered to the destination host. Message disassembly and reassembly should be transparent to the host computers.

It is possible for a host computer to send messages to an IMP even if the IMP is not able to disassemble and transmit them immediately. The IMP will simply buffer these messages in an input queue. Similarly, the packets sent from one IMP to the next may also be buffered in input queues. Message disassembly and queuing is depicted in Figure 7, as described below.

As shown, the host has transmitted five messages to the IMP. The fifth message (M5) is in transit to the IMP. The third and fourth messages (M3 and M4) are waiting in the IMP's input queue. The first and second messages have already been disassembled into four packets each, as follows:

M1 -> P1a, P1b, P1c, P1d

M2 -> P2a, P2b, P2c, P2d

The source node has routed half of these packets through one adjacent IMP and half through the other. Some of the packets are in transit from the source node to the next IMP, some are waiting in input queues, and some are in transit from the second IMPs to the following IMPs. Though reassembly at the destination node is not shown, it is clearly possible for these packets to arrive out of sequence. Thus, some packets

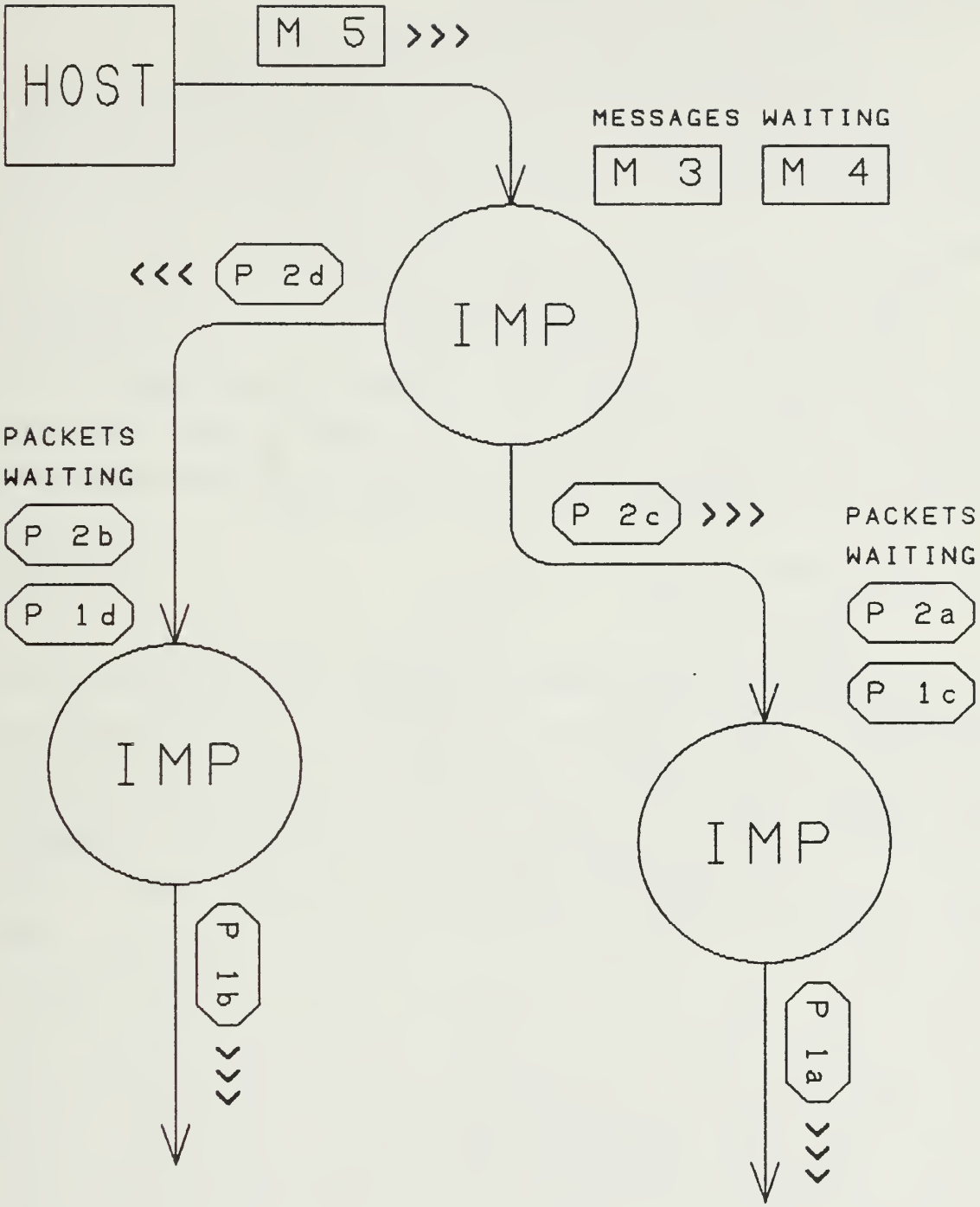

Figure 7. Packet Switching by ARPANET IMPs

may have to be stored at the destination node while awaiting the preceding packets.

3. Host Connections to the Network

On ARPANET the hosts are often located in close proximity (2000 feet or less) to the IMP which connects them to the network. This proximity allows the host to use a high capacity communications link for this connection. However, at least initially there will not be enough IMPs on the DISNET for each of its hosts to have one located in close proximity. For this reason the SIMNET hosts will not be assumed to use high capacity links for their host-IMP connections. This thesis will assume that trunk lines of 50 Kbps are used to connect the SIMNET hosts to DISNET IMPs.

C. NETWORK PROTOCOLS

1. Principles

Large, highly capable computer networks such as ARPANET are very complex. To make this complexity manageable, the network is designed as a series of layers with each layer performing specific functions. Each network layer is associated with a protocol, which is the set of algorithms designed to perform the function of the layer. By breaking the network functions into a hierarchy of protocol layers it is possible for the user to establish communication with a distant host, a high level function, without being concerned about low level functions such as the signalling method used over the trunk lines.

Layering simplifies the network problem by allowing each layer to view the set of lower layers as simply providing the network services it needs to use. How a specific protocol performs the functions of its layer is of no interest to the layers above. Thus,

only the interfaces between layers are of importance, not the internals. This is the principle of information hiding, and its use in network design permits changes to be made to individual protocols without affecting the overall functioning of the network. An excellent discussion of protocol layering is given by Zimmermann [8].

The most frequently discussed set of layered protocols is the Open Systems Interconnection (OSI) reference model, which has been proposed by the International Organization for Standardization (ISO) [8], [9:15-21]. However, the ARPANET design predates the OSI model by almost ten years and does not strictly conform to it. For this reason, the specific ARPANET protocols will be discussed below, rather than the more commonly reviewed OSI protocols.

2. Network Application

The highest layer in the hierarchy of network functions is the application program which is using the network services. In this case the application is IBGTT software, which can be either the CSF program or the RSM. This thesis will concentrate on the performance of the network from the perspective of the CSF, but the principles are the same from either perspective.

The application program has to send both data and network information to the next layer in the hierarchy, the Transmission Control Protocol (TCP). The way in which information is passed from IBGTT to TCP is determined by the TCP interface. However, the IBGTT programs do have some control over how data is transmitted, in that they can specify an end of data block (called a Push) to TCP. If no Push is sent then TCP will be able to package the data into maximum

length data blocks, which is the most efficient means of transmitting a large amount of data. Throughout this thesis IBGTT will be assumed to send data blocks of the maximum possible length to TCP until all the data for that game cycle has been sent. Only after all the game cycle data has been sent will a Push be sent, so that only one partial TCP frame may be generated per game cycle.

3. Transmission Control Protocol

TCP is primarily responsible for maintaining a reliable host-to-host connection for the purpose of transferring data. The TCP functions which will be discussed in this thesis include establishing connections, transferring data, and ensuring adequate flow control. A diagram of the TCP format is shown in Figure 8. A complete description of TCP can be found in the official protocol specification [10].

Before any data transfer can occur a connection has to be made between the two hosts. TCP does this through the use of a three-way handshake. Port numbers are assigned to each end of the connection to identify the logical channels to which the data should be sent at each host. These port numbers are included in every TCP header, as seen in Figure 8. TCP is also responsible for breaking the connection once the application is finished.

The options section of the TCP header is normally only used during connection set-up. Once data transfer begins the TCP headers usually consist of the first 20 bytes indicated in Figure 8. One option used during connection set-up is that of determining the maximum data segment length which will be used during the connection. As discussed above, this value will be

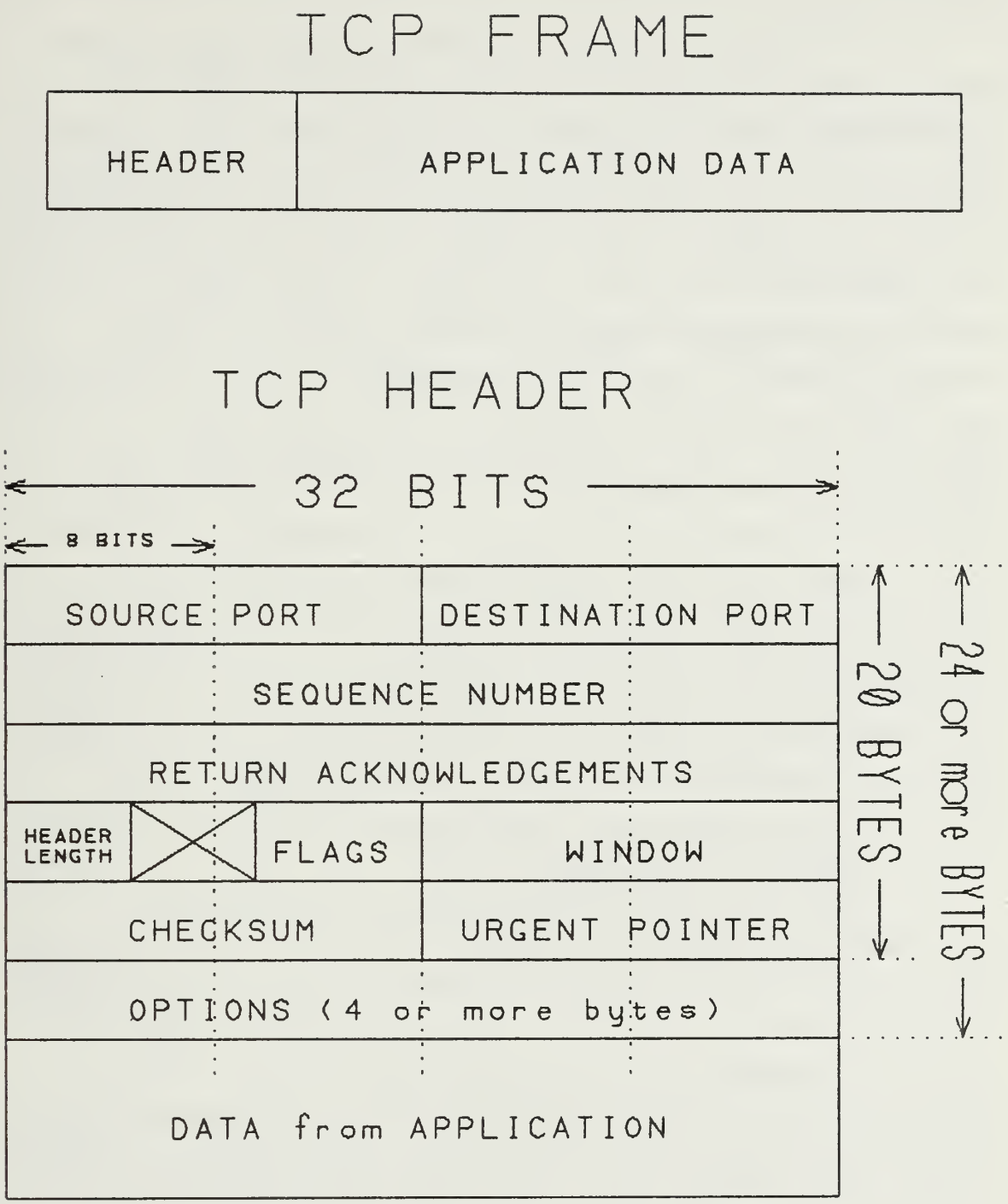

Figure 8. Transmission Control Protocol Format

assumed to be the maximum data segment available to the TCP protocol.

A large portion of the TCP header is used to perform flow control between the hosts. TCP uses windows and acknowledgements to maintain flow control, but it does this by counting bytes instead of messages. Each byte of data sent from a host is counted and given a sequence number. The sequence number field contains the number of the first data byte in the segment. Windows and acknowledgements are returned by the receiving host. The acknowledgement field contains the next expected sequence number, while the window comprises the number of additional bytes that the receiving host is prepared to accept. Flow control will be discussed again later in the thesis.

Once TCP has prepared the complete frame it is sent to the next protocol in the hierarchy, which is the Internet Protocol (IP). TCP sends some information to IP outside of this frame which IP uses to construct its own header. This information includes the source address, destination address, protocol used (i.e., TCP), and the length of the complete TCP frame.

4. Internet Protocol

The Internet Protocol (IP) is designed to provide those functions necessary to deliver a package of bits, called a datagram, from one host to another. IP has a number of features which enable the protocol to send datagrams across connected networks. These features allow the IP datagrams to be fragmented into smaller datagram fragments if the intervening networks do not permit packets as large as the intact datagram to cross. These fragments can then be reassembled at their destination using information contained in the IP header. Since it is expected that SIMNET will operate

over a single network, DISNET, these features will not be significant for this thesis.

A diagram of the IP format can be seen in Figure 9. As shown, IP treats the entire TCP frame as if it were data by simply attaching it to the end of the IP header. The only items of information from TCP that IP uses directly are those that were passed outside of the header: addresses, protocol type, and frame length. Besides fragmentation, the major function of IP is simply in providing addresses to the datagrams. Note that the IP header does not contain fields for sequencing or flow control. To this protocol each datagram is just an independent unit with a destination address.

The options field of the IP header is not used in most datagrams. Thus, the typical IP datagram has a 20 byte header. The field called time (usually "time to live") is set at some time value when the header is created. As the datagram passes through the network this field is reduced each time the IP header is processed. If the field reaches zero, then the datagram is discarded. This timeout mechanism is designed to keep undelivered datagrams from creating congestion on the network. More information on the features of IP can be found in the official protocol specification [11].

When the IP frame has been completed it is passed to a network access protocol for further packaging. The product of the network access protocol will be an ARPANET message which can be sent from the host to a network IMP.

5. Network Access Protocols

Network access protocols establish the interface between a host computer and the IMP to which

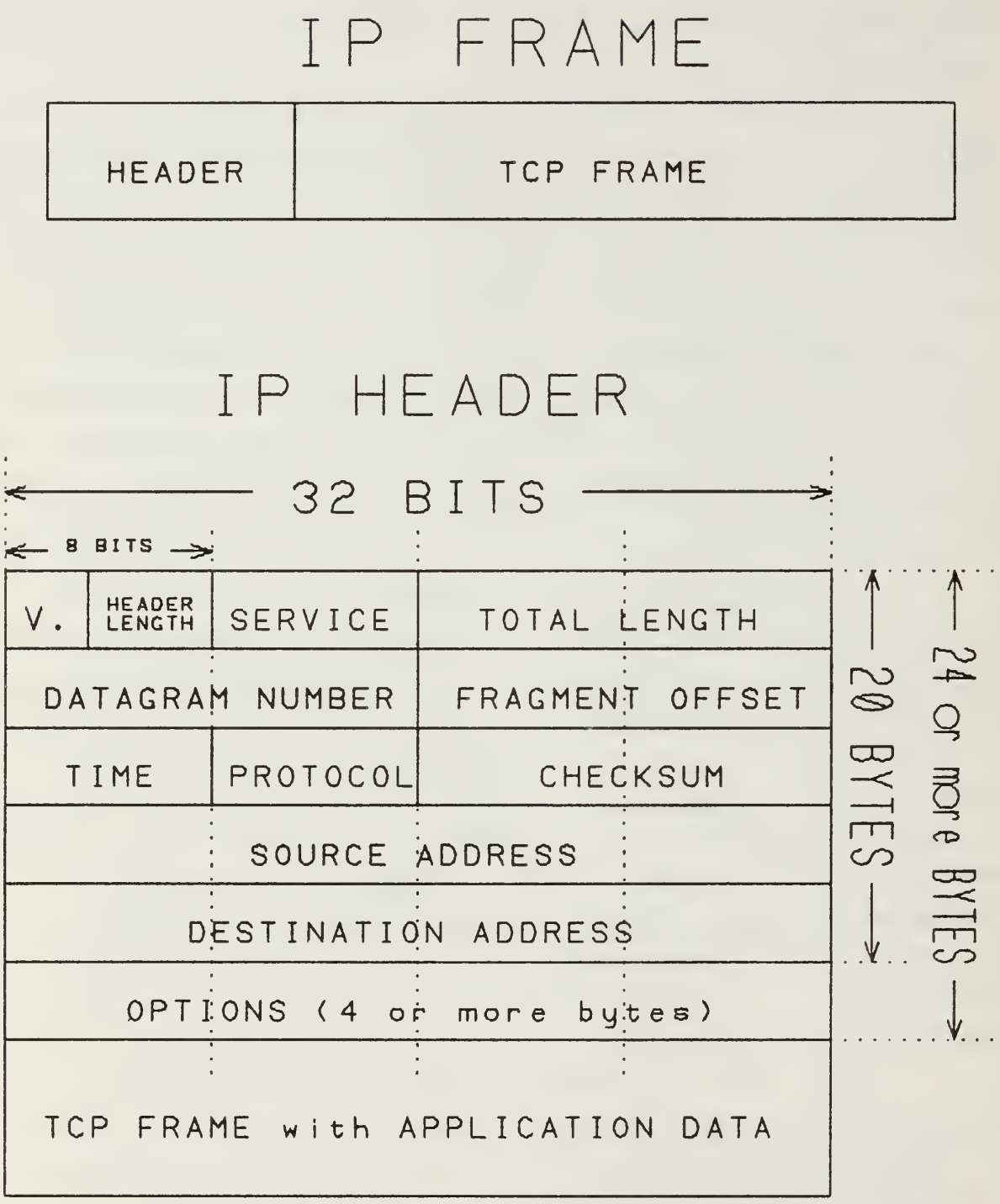

Figure 9. Internet Protocol Format

it is connected. A network access protocol is responsible for providing the reliable transfer of data, in the form of IP datagrams, between hosts and IMPs. In the ARPANET architecture there are two different network access protocols in use: the BBN 1822 protocol and the DDN X.25 protocol.

The BBN 1822 protocol is the original network access protocol on the ARPANET [12:154-157]. While 1822 is still in widespread use on the network, it is being slowly phased out in favor of the DDN X.25 protocol. It is expected that all hosts on SIMNET will interface with DISNET through the use of DDN X.25. For this reason the BBN 1822 protocol will not be discussed further in this thesis.

The DDN X.25 is a very highly specified subset of the CCITT X.25 specification [13]. The official DDN X.25 specification states,

> Thus in several areas where X.25 allows a choice, a single choice appropriate for DDN is specified; in areas which X.25 leaves unspecified, addressing in particular, conventions are specified that are consistent with the overall architecture [of] DDN . . . . [13:2]

For this reason the DDN X.25 is quite different from the more general CCITT specification. A diagram of the DDN X.25 format is shown in Figure 10.

As shown, X.25 treats the IP frame as data, inserting it between its header and trailer. The short (5 bytes) X.25 header is primarily concerned with addressing and identification. Note that there are no fields dedicated to sequencing and flow control. As with IP, these higher level functions are assumed by X.25 to be handled by TCP. The requirement that X.25 provide reliability is met through the use of a 16 bit frame check sequence as a trailer. This sequence provides an error check over the entire X.25 frame.

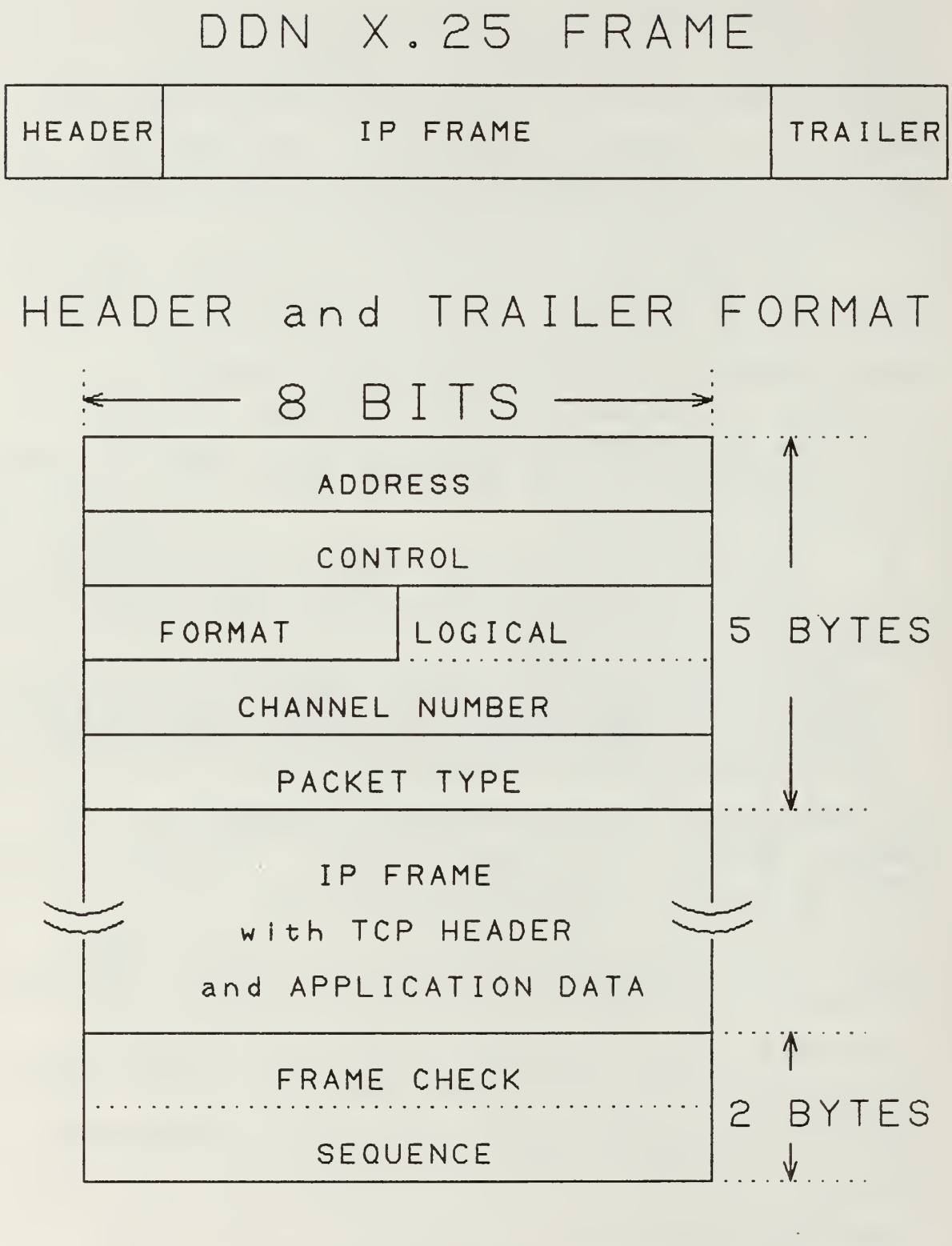

Figure 10. DDN X.25 Protocol Format

A complete DDN X.25 frame is called an ARPANET message. When a message is complete the X.25 protocol transfers it from the host to the connected IMP where it will be further processed before being sent out on the network. In a similar fashion, the X.25 protocol is also used by the IMP to transfer received messages to the destination host.

Thus, three levels of protocol packaging are performed on a segment of application data before it is sent from the host to the network. A graphical overview of this process is shown in Figure 11.

6. IMP-IMP Protocol

So far, each protocol has added information to the data package simply by attaching a header (and a trailer for X.25) to the package it receives from the higher level protocol. However, the transition from the X.25 protocol to the next protocol level is more complicated than this. This is because an X.25 ARPANET message which is longer than the maximum IMP-IMP packet length must be disassembled into smaller packets before it can be sent onto the network, as discussed earlier. Each of these smaller packets is then given its own IMP-IMP protocol header before it is transmitted.

A diagram of the message disassembly process is shown in Figure 12. As shown, only the IP frame portion of the message is disassembled and transmitted onto the network. The X.25 header and trailer are not sent beyond the source IMP. The information in the X.25 header is used to construct the IMP-IMP headers which are attached to the disassembled packets, a process called protocol conversion. When these packets reach the destination IMP the X.25 header is reconstructed from the information in the IMP-IMP headers. Similarly, the X.25 checksum trailer is only

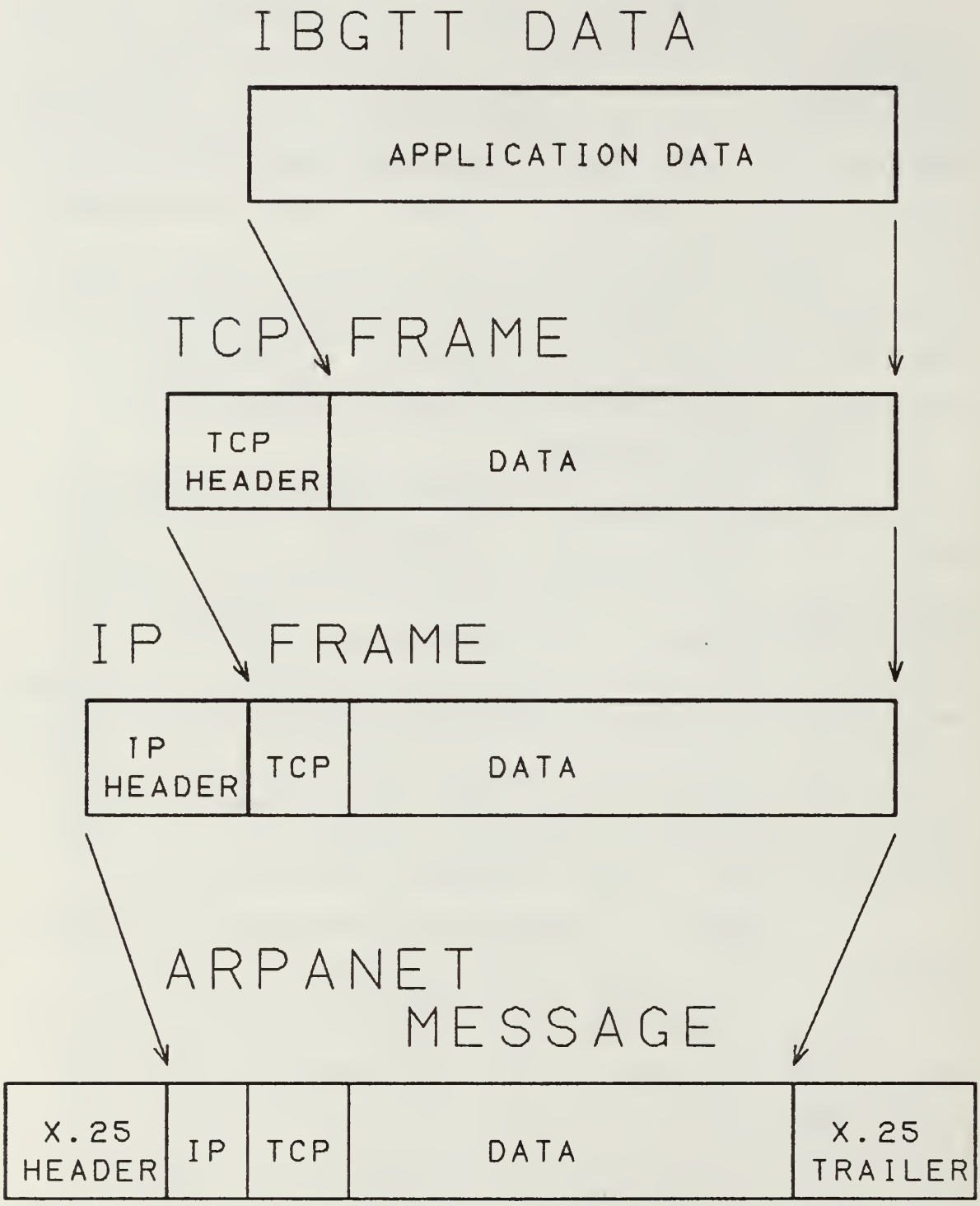

Figure 11. Data Packaging in an ARPANET Message

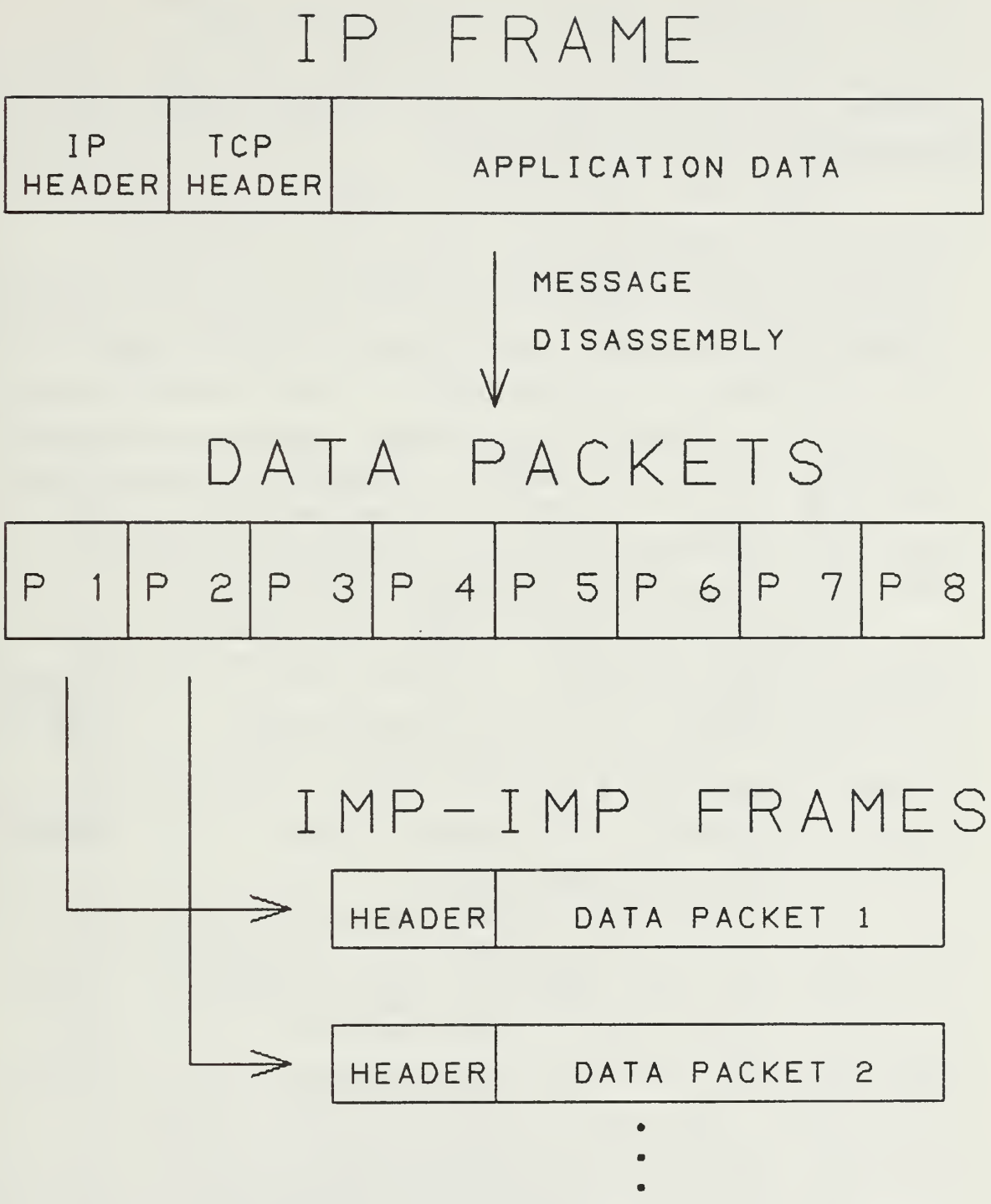

Figure 12. Disassembly of an ARPANET Message

used to determine if any errors occurred during the transmission from the host to the IMP. Once checked, the trailer is discarded by the IMP. When the X.25 message is reconstructed at the destination IMP the checksum trailer is recomputed.

Note that the entire IP frame is treated as data by the disassembly process. The information contained in the TCP and IP headers is not used by the IMP-IMP protocol. Each resulting data packet is given its own IMP-IMP header, which adds an additional 16 bytes to create an IMP-IMP frame. This header has all the address and routing information needed to transmit the packet to the destination. Separately addressing each frame allows them to be routed over independent paths, as discussed earlier. The IMP-IMP header is described in detail by Tanenbaum [9:226-231].

The IMP-IMP frames shown at the bottom of Figure 12 each represent a complete unit of information needed by the destination. However, before a frame can be reliably sent over a trunk line it needs to be further framed with a hardware generated header and trailer [9:165-167]. The format of this final IMP-IMP packet is shown in Figure 13.

As shown, the hardware generated header is only three bytes long. It consists of the three control characters SYN, DLE, and STX. These characters signal the receiving IMP that an IMP-IMP frame is arriving. These bytes are followed by the IMP-IMP header and the data packet. The start of the six byte hardware generated trailer is signalled by the control characters DLE and ETX. These are followed by a 24 bit checksum which is computed over the entire frame. The packet ends with the SYN control character.

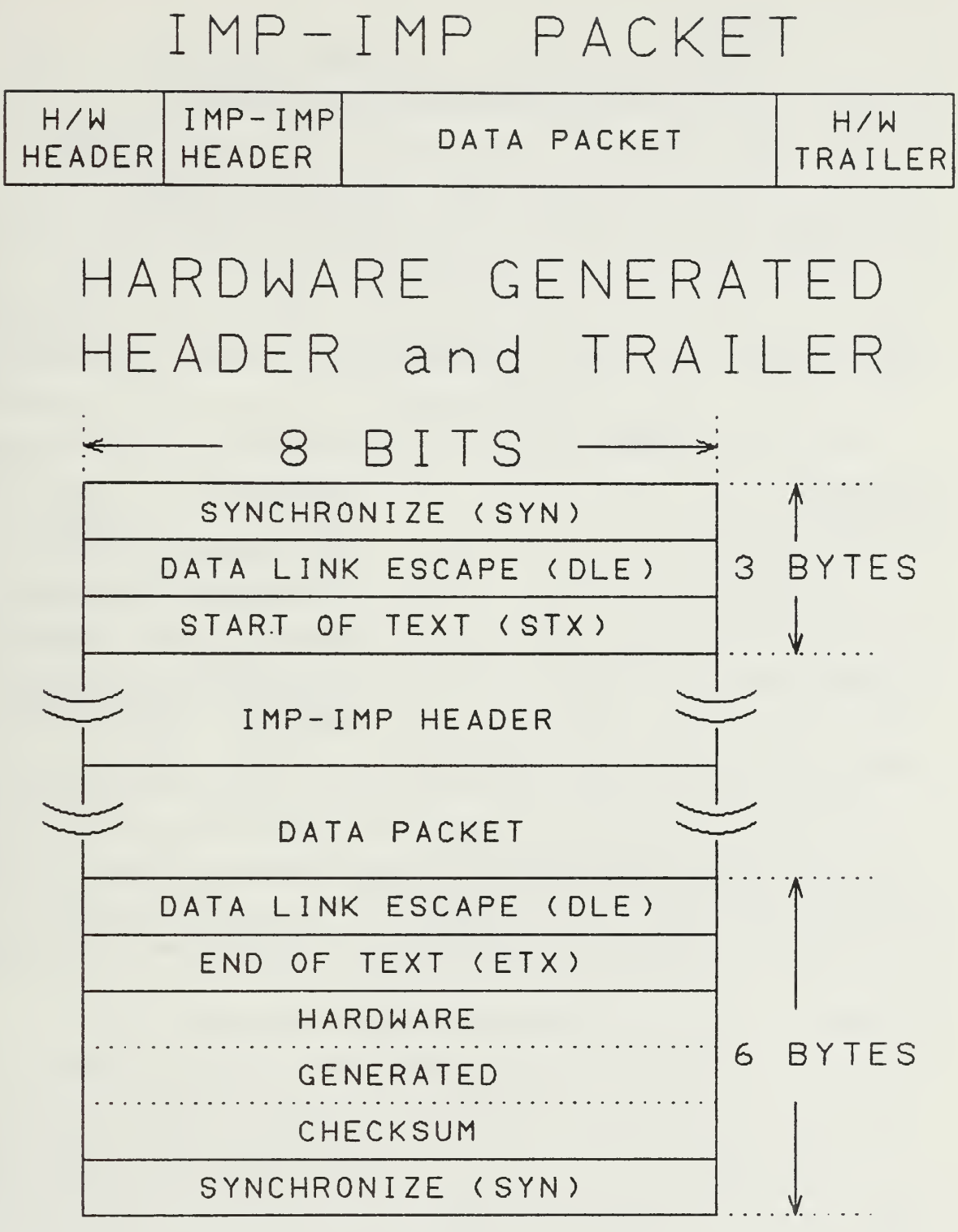

Figure 13. IMP-IMP Protocol Format

7. Summary

As described above, the IBGTT data has to pass through three levels of network protocols before it can be sent from the host to a network IMP as an ARPANET message. The message is then disassembled and repackaged using the IMP-IMP protocol. These protocols are summarized below in Table II.

TABLE II.
SUMMARY OF ARPANET PROTOCOLS

Protocol	Function
Transmission Control Protocol (TCP)	Provide a reliable host-host connection
Internet Protocol (IP)	Datagram transmission
DDN X.25 Protocol	Provide host-network interface
IMP-IMP Protocol	Reliable transfer of packets through the network

The IMP-IMP packets are delivered to the destination node, where they are used to reconstruct the original message. This message is delivered to the destination host, where it is unpackaged from the successive protocols. Finally, the original data segment is delivered to the appropriate host process.

V. REQUIRED APPLICATION THROUGHPUT

A. PROCEDURE

1. Data Collection

As described previously, IBGTT is currently operated in the distributed mode through the use of an ETHERNET LAN at NOSC. The DEC computers used for this system can monitor the ETHERNET traffic via DECNET, the standard DEC network software. Using these tools, measurements of the amount of IBGTT data transferred between the CSF and the RSMs were obtained by personnel at NOSC.

The DECNET measurements are all made with respect to the CSF (called the executor node by DECNET). Thus, the bytes sent from the CSF to an RSM (remote node) are listed under the RSM as "bytes sent". None of the data listed under the CSF reports the data sent to RSMs. Thus, to determine the total CSF to RSM traffic for a game with more than one RSM the bytes sent to each RSM must be added. The terms CSFout and RSMin will be used to describe this CSF to RSM traffic. Using this notation, CSF output can be expressed for a game with two RSMs by the following formula:

$$CSFout = RSM1in + RSM2in \qquad (5.1)$$

This thesis will be almost entirely concerned with the CSF to RSM traffic. The DECNET measurements show that the RSM to CSF traffic is negligible in comparison, normally less than one percent. Thus, CSF throughput and application throughput are considered synonymous.

2. Protocol Overhead

The DECNET measurements are for bytes of application data sent. Before these measurements can be applied to an ARPANET based network the overhead for each protocol layer must be added. The starting point for determining total overhead is the network limit on the maximum IP frame, which is 1007 bytes. As discussed previously, the length of the data segment inside this IP frame equals the frame length minus the size of the IP and TCP headers (see Figure 11 in Chapter IV). For each of these headers the most commonly used length is 20 bytes. Thus, the maximum data segment can be computed as follows [14]:

max data segment = 1007 - 20 - 20 = 967 bytes (5.2)

Since IBGTT is assumed to always use maximum length data segments, protocol overhead will be calculated using 967 bytes for the data segment length.

After the IP frame is formed it is further packaged by DDN X.25 protocol. This protocol adds a 5 byte header and a 2 byte trailer for a total of 7 bytes of overhead. The total maximum length X.25 ARPANET message is therefore 1014 bytes long. This message consists of 967 bytes of data and 47 bytes of overhead. Application data efficiency can be expressed as follows:

eff = 100% * (967)/(1014) = 95.365 % (5.3)

This ratio is used to convert the data throughputs measured by DECNET into measurements of the total throughput which the CSF must transfer across the network to the RSM. The conversion equation follows:

throughput = data / 0.95365 (5.4)

From here on the measurements discussed in this thesis will be for total throughput (data plus overhead). Listings of the original DECNET data measurements are provided in the Appendices.

3. Simulation Operating Speed

As discussed previously, an important feature of IBGTT is the ability to run the simulation at faster than real-time. Thus, the speed at which a simulation is operating is an important factor when considering the resultant throughput. For example, a game being played at 2:1 speed (two game cycles for every actual minute of time) should require twice the throughput as the same game being played at 1:1 speed. This is because the CSF has to transfer two complete simulation updates to the RSM every minute instead of one.

Any throughput measurements which are obtained at operating speeds faster than 1:1 will be normalized to the equivalent 1:1 throughput. This 1:1 throughput represents the minimum throughput required by that set of game conditions. While these measurements will be the lowest possible for that simulation, they do not necessarily represent a desirable operating condition. For reasons discussed earlier, it may not be desirable to operate at 1:1 speed during some phases of the simulation. For this reason, any measurements taken at speeds higher than 1:1 will also be evaluated at the higher speed.

B. SIMULATION MEASUREMENTS

1. Game 1

The first set of measurements to be considered were taken during a simulation conducted at NOSC on June 5, 1986. This simulation will be referred to as "game 1". Game 1 was conducted using two RSMs (RSM1 and RSM2). The simulation was operating at 2:1 speed throughout the measuring period. Data throughput measurements for each RSM were used to determine the required CSF throughput, as described earlier. These throughput requirements are listed below in Table III. Note that CSFout at speed = 2:1 reflects the actual simulation conditions, while the data for speed = 1:1 represents the normalized values. The DECNET measurements for game 1 are listed in Appendix A.

TABLE III.

GAME 1 THROUGHPUT REQUIREMENTS IN KBPS

Sampling Period	RSM1in @ 2:1	RSM2in @ 2:1	CSFout @ 2:1	CSFout @ 1:1
1	9.35	23.61	32.96	16.48
2	9.38	23.48	32.86	16.43
3	9.53	23.78	33.31	16.66
4	9.56	23.79	33.35	16.68
5	9.71	23.89	33.60	16.80
6	9.72	24.49	34.21	17.10
7	9.19	20.12	29.31	14.65
8-12	no data collected			
13	8.45	22.95	31.40	15.70
14	9.61	23.60	33.21	16.60
15	9.62	23.94	33.56	16.78
16	9.50	23.68	33.18	16.59
17	9.24	23.56	32.80	16.40
18	9.44	23.61	33.05	16.52
19	9.66	23.93	33.59	16.80
20	9.37	23.54	32.91	16.46
21	9.67	23.54	33.21	16.61
22	9.09	23.22	32.31	16.15

The throughputs listed in Table III for CSFout are also plotted in Figure 14. Plots are shown both for the actual 2:1 speed and the normalized 1:1 speed. As shown in Figure 14, the maximum throughput requirements for game 1 are 34.21 Kbps at 2:1 and 17.10 Kbps at 1:1.

These DECNET measurements were only taken during a portion of the simulation, and that portion was described by NOSC personnel as not including a "hot war" situation. During the period measured, there were about 160 objects (ships, submarines, and aircraft) present--a scenario size described as "slightly smaller than average" by NOSC personnel. During these game 1 measurements, the time interval for every sampling period was between 61 and 65 seconds. Thus, these measurements represent a very precise view of the CSF throughput requirements during the interval measured. However, the total period of measurement is also quite short, only about 23 minutes out of a simulation lasting several hours. For this reason, the data can only be considered as one possible set of throughput requirements, rather than representing some larger general case.

2. Game 2

The IBGTT measurements for "game 2" were taken during a simulation conducted at NOSC on November 7, 1986. Like game 1, game 2 was conducted using two RSMs. The entire game 2 simulation was conducted at a 2:1 speed. The calculated CSF throughput requirements for game 2 are listed in Table IV, using the same format as Table III. As before, CSFout is listed both for the actual 2:1 speed as well as the normalized 1:1 speed. The DECNET measurements for game 2 are listed in Appendix B.

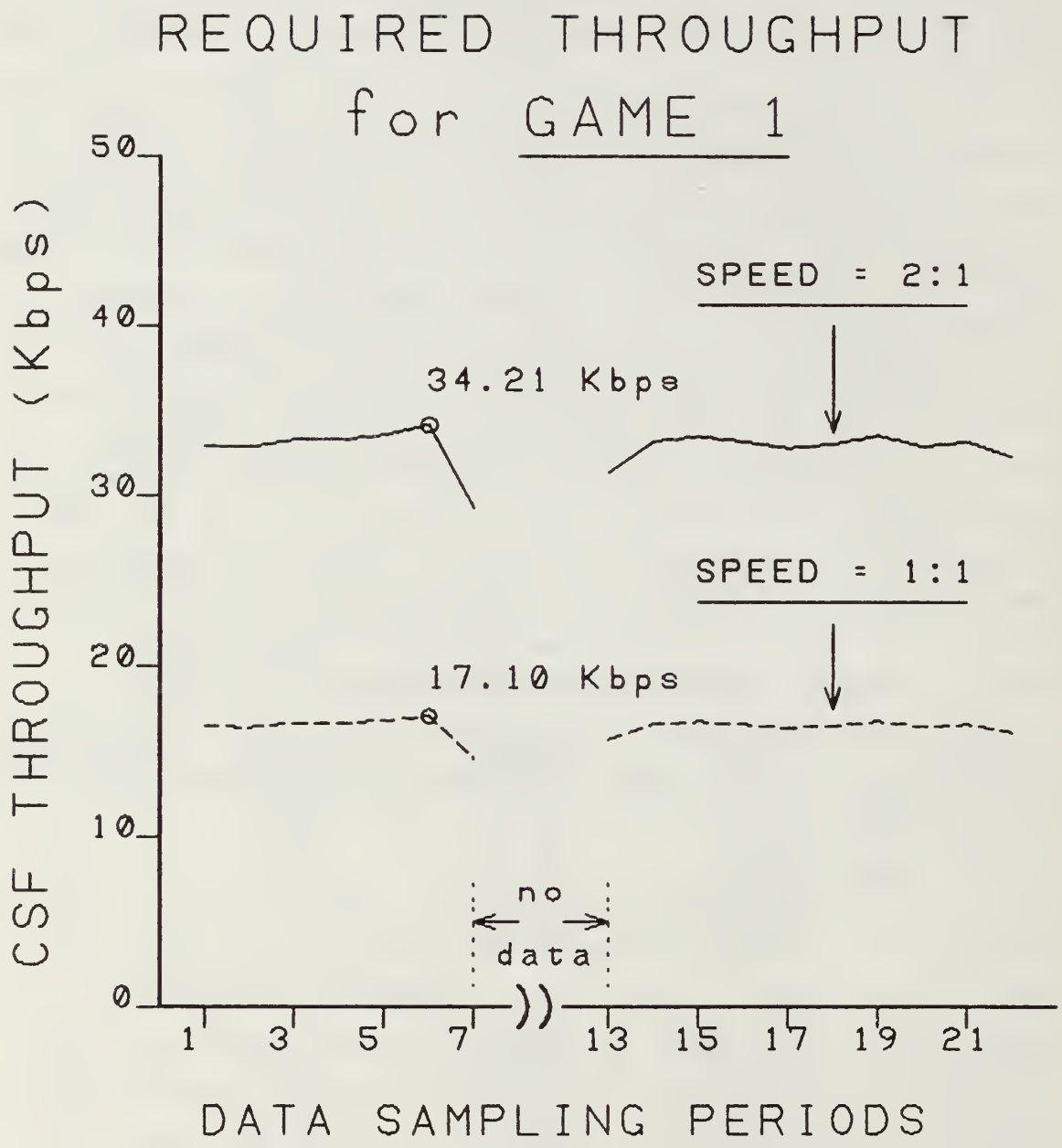

Figure 14. CSF Throughput Required to Network Game 1

TABLE IV.

GAME 2 THROUGHPUT REQUIREMENTS IN KBPS

Sampling Period	RSM1in @ 2:1	RSM2in @ 2:1	CSFout @ 2:1	CSFout @ 1:1
1	0.50	0.51	1.01	0.51
2	2.68	2.69	5.37	2.69
3	6.23	6.34	12.57	6.29
4	7.43	7.84	15.27	7.64
5	8.43	10.76	19.19	9.60
6	9.51	14.33	23.84	11.92
7	10.18	15.15	25.33	12.67
8	10.62	16.15	26.77	13.39
9	10.72	16.21	26.93	13.47
10	10.47	15.91	26.38	13.19
11	11.43	17.04	28.47	14.24
12	11.90	17.47	29.37	14.69
13 - 20	pause in the game			
21	12.11	17.75	29.86	14.93
22	12.09	17.99	30.08	15.04
23	12.74	23.08	35.82	17.91
24	14.56	28.32	42.88	21.44
25	14.46	27.79	42.25	21.13
26	15.24	29.51	44.75	22.38

The throughputs shown in Table IV for CSFout are plotted in Figure 15, both at 2:1 speed and 1:1 speed. As shown in the plots, the maximum throughput requirements for game 2 are 44.75 Kbps at 2:1 and 22.38 Kbps at 1:1.

Unlike game 1, the DECNET measurements for game 2 were taken over the entire course of the simulation. This is reflected in the plots shown in Figure 15, which reveal a steady growth in required throughput as the simulation progresses. To make coverage of the entire simulation possible, the sample periods were increased from 60 seconds to 15 minutes. The number of objects in the game varied from 60 at the beginning to approximately 300 at the peak of the conflict. This

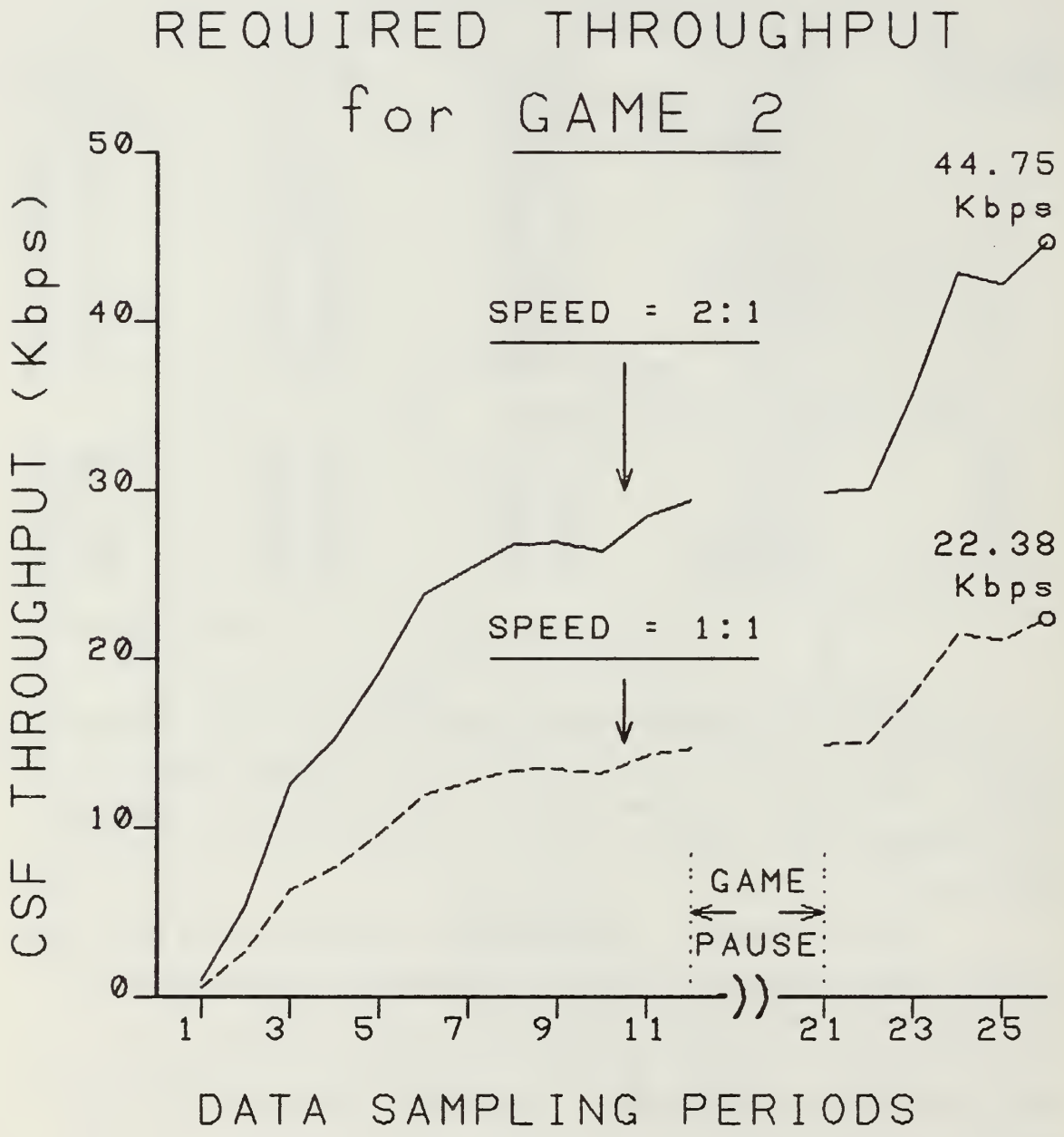

Figure 15. CSF Throughput Required to Network Game 2

scenario was described by NOSC personnel as being "slightly larger than average."

While it may appear that the curves in Figure 15 will continue to increase after the last sample period, the simulation actually ended in the period following the last one shown. The discontinuity in the data is due to the simulation being placed in a "pause" for about 1.5 hours over lunch. Note that even using 2:1 speed it took an entire working day to complete this simulation.

VI. ANALYSIS OF NETWORK PERFORMANCE

A. MODELING THE PROBLEM

Having obtained some reasonable values for the throughput requirements of the application (IBGTT), it is now necessary to determine under what consitions DISNET can be expected to support those requirements. There are many possible approaches to estimating the performance of a packet switched network, but they can generally be broken down into three areas: mathematical analysis, computer simulation, and direct measurement [15], [16]. In this case, direct measurement will not be possible until SIMNET is fully implemented.

Computer simulation and mathematical analysis are similar in that both techniques require the network to be modeled. In the case of computer simulation, the network is modeled in the form of algorithms and data structures. In the case of mathematical analysis, the network is modeled as a set of equations. Modeling typically requires that some aspects of the network be greatly simplified in order that the network properties of interest may be analyzed.

In this thesis, expected network throughput will be analyzed through the use of a simplified model of a switching node (or IMP). This model is adapted from one presented by Rubin [17], who uses it to develop general expressions for packet waiting time at a switching node. The model presented below will be used to develop an equation which relates maximum application throughput to the amount of internal network traffic present at the source node.

In fact, the result of this analysis is not application throughput through the network--it is simply application throughput through the source node. However, from the perspective of the application there are two bottlenecks on its host-host connection: the source node and the destination node. Between these two nodes there are multiple pathways available for the transfer of application data, making it reasonable to assume that the network is capable of handling packets at the rate the source node sends them. The effects of network traffic at the destination node will not be considered at this time, though clearly those conditions are important. Thus, by restricting the analysis to only the source node it is assumed that the rest of the network is capable of handling the application traffic at the rate that the source node transfers it from the source host. This assumption is discussed in detail later in the thesis.

B. SWITCHING NODE MODEL

1. Traffic Through the Node

The model presented here is a general model for any switching node in the network. However, the model will primarily be applied as a model of the source node for the transmission of IBGTT application data. Thus, the terms "source node" and "switching node" are synonymous for the purpose of this discussion.

The traffic arriving at the switching node is modeled as consisting of two classes: internal and external. The internal traffic consists of the normal flow of IMP-IMP packets which are being routed to the switching node. The external traffic consists of X.25 ARPANET messages which are being transmitted from a host to the switching node for delivery through the

network. The internal and external arrivals combine to produce a stream of output traffic from the node. A diagram of this switching node model is shown in Figure 16. In the figure, all internal traffic is shown to be on one trunk line while all output traffic is shown to be on another. In fact, there can be multiple internal arrival channels and multiple output channels without affecting any of the following discussion.

In this model, the internal traffic is characterized by a packet arrival rate of "R_i" and a packet length of "L_i". The internal traffic will be described by using an average packet arrival rate for R_i. This is because the rate of network traffic over any line is very unpredictable at any moment, but it can be time averaged to a reasonably consistent value. Similarly, an average value will be used for the internal packet length, L_i. Again, there is a wide range of possible packet lengths, but network measurements have shown that the average packet length for all network traffic is very consistent.

External arrival traffic is characterized by an arrival rate of "R_e" and a message length of "L_e". Unlike the case for internal traffic, measured averages do not have to be used for these values. Since this model is only considering IBGTT as a source of external traffic, the value of L_e is fixed to the maximum possible length of an X.25 ARPANET message. Similarly, the arrival rate, R_e, is fixed at the maximum arrival rate possible given the capacity of the host-IMP connection.

While the internal packets are considered to be unchanged by the switching process at the node, that is not true of the external messages. Each ARPANET message received at the switching node generates

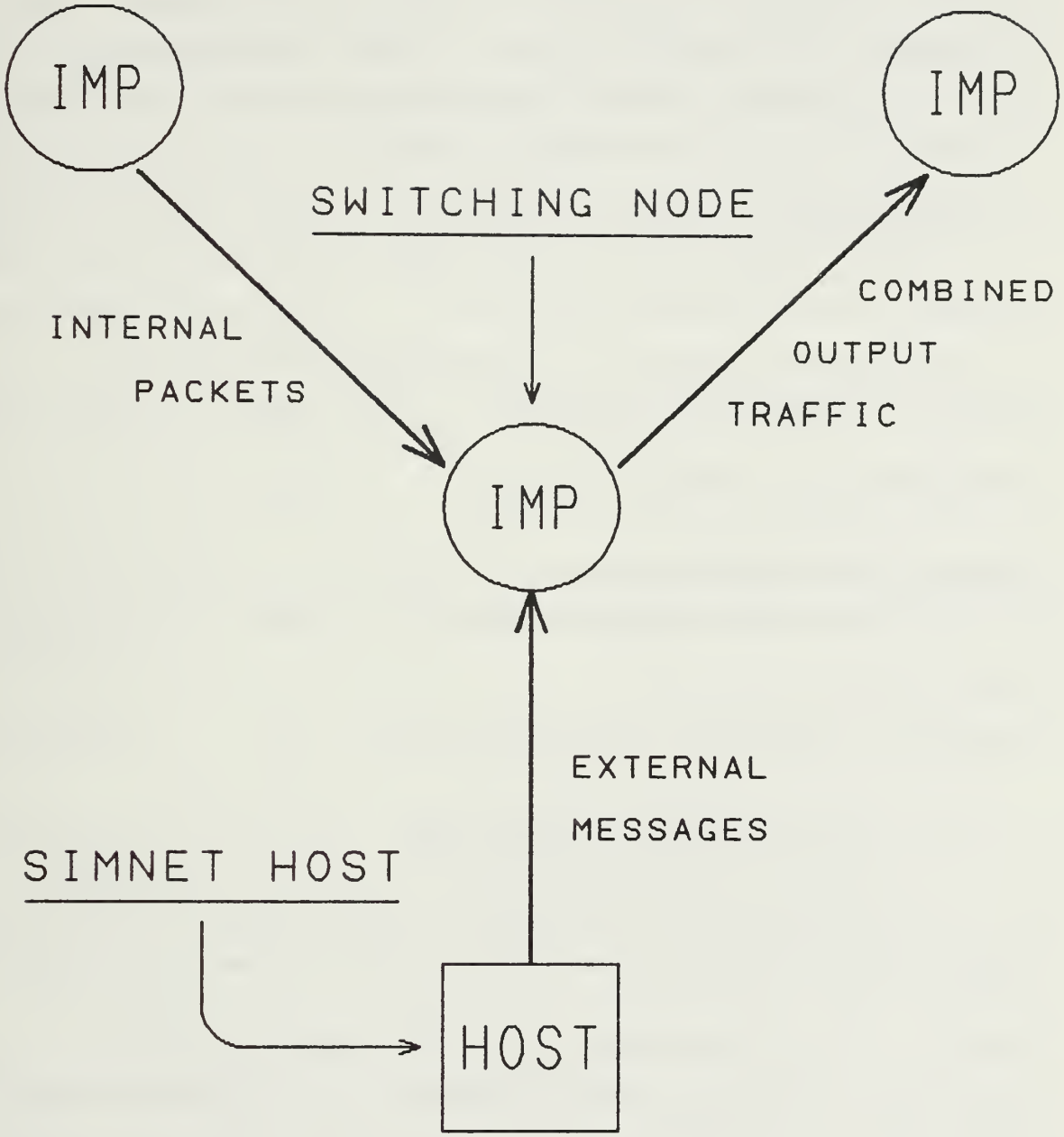

Figure 16. Model of a Switching Node

several IMP-IMP packets. These packets are characterized by a ratio of "M" packets produced per message and a packet length of "L_p". Since the arriving external messages are of maximum length, these parameters can also be fixed at their maximum value.

In summary, the following parameters have been defined for the switching node model:

internal packet arrival rate:	R_i
average internal packet length:	L_i
external message arrival rate:	R_e
fixed external message length:	L_e
number of IMP-IMP packets generated per external message:	M
fixed length of externally generated IMP-IMP packets:	L_p

2. Packet Service at the Node

Packets arriving at the switching node often have to wait for service while previously arriving packets are served by the node. Since there can be several input channels of arriving packets, each channel can be considered to have a waiting queue to hold these packets. The switching node is modeled as a single-server process. This means that only one packet can be served (i.e., transmitted out of the node) at a time. Thus, each packet in a waiting queue must wait for every previously arriving packet to be served by the switching node, one at a time.

The time required for the switching node to service an individual packet is modeled to be the time required to transmit the packet onto the output trunk line. This model assumes that processing time in the IMP is negligible. Thus, service time is strictly a function of packet length and output line capacity. In this model, all trunk lines connected to the switching node are assumed to have capacity, "C", equal to 50 Kbps, the most common trunk line capacity in MILNET. This 50 Kbps capacity will also be assigned to the host-IMP connection.

From these assumptions, a general expression for packet service time, "A", can be defined:

$$A = L / C \qquad (6.1)$$

From equation 6.1, specific expressions for the service time of the different classes of packets can be obtained:

internal: $A_i = L_i / C$ (6.2)

externally produced: $A_p = L_p / C$ (6.3)

Since each external message produces M packets of length L_p, the service time of an external message can be expressed:

external message: $A_e = M * A_p$ (6.4)

C. WAITING TIME ANALYSIS

1. Packet Interarrival Time

The parameter "t_n" is defined as the time of arrival for packet number "n" of arbitrary traffic

class. More specifically, the arrival times for each class of packets is defined:

internal packet arrival time: t_n^i

external message arrival time: t_n^e

Using this notation, a general expression for packet interarrival time can be defined as follows:

$$T_{n+1} = t_{n+1} - t_n \qquad (6.5)$$

Here, T_{n+1} is the time interval between the arrival of packet "n" and the arrival of packet "n+1". The specific equations for the different classes of traffic follow directly:

internal: $$T_{n+1}^i = t_{n+1}^i - t_n^i \qquad (6.6)$$

external: $$T_{n+1}^e = t_{n+1}^e - t_n^e \qquad (6.7)$$

2. Batch Arrival Model

The number of internal packets which arrive during the external message interarrival time, T_{n+1}^e, is defined as "$N_i(T_{n+1}^e)$". While the actual number for any given interarrival time will vary with the dynamics of network loading and routing, time averaging can be used to approximate this number, as follows:

internal: (avg) $$N_i(T_{n+1}^e) = R_i * T_{n+1}^e \qquad (6.8)$$

Similarly, the number of external messages which arrive during the internal packet interarrival time is defined as follows:

external: (avg) $N_e(T_{n+1}^i) = R_e * T_{n+1}^i$ (6.9)

For each class of traffic, this number can be considered to represent a batch arrival which must be processed before the "n+1" packet of the other traffic class. Thus, the order of service for packets and messages when viewed from the perspective of the external arrivals is:

1) external message "n"
2) $R_i * T_{n+1}^e$ internal packets
3) external message "n+1"

The order of service can be similarly expressed from the perspective of the internal packets.

Since the service time for a single packet of either class is known, the service times of these batch arrivals can be easily generated, as follows:

internal:

batch service time = $R_i * T_{n+1}^e * A_i$ (6.10)

external:

batch service time = $R_e * T_{n+1}^i * A_e$ (6.11)

Using equation 6.4, equation 6.11 can be rewritten as follows:

batch service time = $R_e * T_{n+1}^i * M * A_p$ (6.12)

3. Waiting Time Equations

The parameter "W_n" is defined as the waiting time for packet "n" of arbitrary traffic class. More specifically, the waiting time for a packet of each class is defined as follows:

internal packet waiting time: W_n^i

external message waiting time: W_n^e

Using this notation for waiting time, and considering the model as it has been developed to this point, the equations for the waiting time of a specific packet of either class of traffic can now be written:

waiting time for an internal packet:

$$W_{n+1}^i = [W_n^i + A_i - T_{n+1}^i + R_e * T_{n+1}^i * M * A_p]^+ \quad (6.13)$$

waiting time for an external message:

$$W_{n+1}^e = [W_n^e + M * A_p - T_{n+1}^e + R_i * T_{n+1}^e * A_i]^+ \quad (6.14)$$

In both equations, the notation $[x]^+$ is defined as follows:

$$[x]^+ = x, \text{ if } x \geq 0 \quad (6.15a)$$

$$[x]^+ = 0, \text{ if } x < 0 \quad (6.15b)$$

In other words, the waiting time can not be less than zero.

Each of these waiting time equations can be broken down in the following way. The waiting time of a packet equals:

- -> the waiting time of the previous packet
- -> PLUS the service time of the previous packet
- -> MINUS the interarrival time
- -> PLUS the service time for the batch arrival of packets in the other class of traffic
- -> AND IF the above sum is less than zero, then the waiting time equals zero.

Equations 6.13 and 6.14 are functionally very similar to equations 6a and 6b in Rubin [17]. However, because the development given above is quite different from Rubin's, the equations do not appear to be very similar.

Rubin uses these waiting time equations to develop a series of general packet waiting time expressions for different possible traffic configurations at the switching node. However, this thesis is more concerned with the message throughput than it is with message delay, so no further waiting time expressions will be developed here. Instead, the external message waiting time equation will be used to derive an expression which describes the maximum external message arrival rate as a function of the parameters of the internal traffic.

D. CRITICAL RATE ANALYSIS

1. Stability Equation

From this point on, network performance will be analyzed only from the perspective of the external message traffic, since this represents the data output from the SIMNET host. Therefore, the starting point for determining the maximum external message arrival rate is equation 6.14, the external message waiting time equation, which is shown below.

$$W_{n+1}^{e} = [W_n^{e} + M*A_p - T_{n+1}^{e} + R_i*T_{n+1}^{e}*A_i]^{+}$$

An important observation can be drawn from this equation. For the waiting time of a message (W_{n+1}^{e}) to be less than or equal to the waiting time of the previous message (W_n^{e}) the following condition must be met:

$$M*A_p - T_{n+1}^e + R_i*T_{n+1}^e*A_i \leq \emptyset \qquad (6.16)$$

This condition is clearly desirable. If the opposite case were to hold then the external messages would be subject to successively longer waiting times. Eventually, the message delays would be too severe to permit effective communication between hosts.

While equation 6.16 is written for the waiting time of a single message, it can be converted to a continuous-time form by multiplying through by the message arrival rate, R_e, as follows:

$$R_e*M*A_p - R_e*T_{n+1}^e + R_e*T_{n+1}^e*R_i*A_i \leq \emptyset \qquad (6.17)$$

Now consider the terms R_e and T_{n+1}^e. R_e is the arrival rate of external messages, which is fixed at the maximum possible rate. T_{n+1}^e is the time interval between these messages, so it is also fixed by the same assumption. Clearly, R_e and T_{n+1}^e are inverse quantities and the following condition holds:

$$R_e * T_{n+1}^e = 1 \qquad (6.18)$$

Using equation 6.18, equation 6.17 can be rewritten as follows:

$$R_e*M*A_p - 1 + R_i*A_i \leq \emptyset \qquad (6.19)$$

This equation can be rearranged to give the following expression:

$$R_e*M*A_p + R_i*A_i \leq 1 \qquad (6.20)$$

Equation 6.20 is the stability equation for the switching node model. The stability equation can be shown to be correct by considering its parts. The term "R_i*A_i" is the portion of time required by the switching node to service the internal traffic. Similarly, "R_e*M*A_p" is the portion of time required to service the external traffic. Clearly, if the sum of these two terms exceeds unity then the traffic at the switching node is unstable because it is arriving faster than it can be serviced. When traffic is unstable like this the waiting time for packets and messages will grow without bound.

2. Critical Rate Equation

Using the stability equation, an expression can be given for the critical rate of traffic arrival at the switching node. The critical rate is the highest rate of total traffic arrival which is not unstable. The critical rate equation is as follows:

$$R_e*M*A_p + R_i*A_i = 1 \qquad (6.21)$$

This equation is simply the stability equation set to unity -- the highest stable rate of traffic. Equation 6.21 can be easily rearranged to give the following expression:

$$R_e = (1 - R_i*A_i) / M*A_p \qquad (6.22)$$

Equation 6.22 is the critical rate equation for external message arrivals. In this equation R_e represents the maximum rate of external message arrival which can be processed by the switching node, given the values of M, A_p, A_i, and R_i.

As stated previously, the parameters M and A_p are fixed at their maximum value. The parameter A_i is a function of internal packet length, which can be reliably described by a narrow range of average values. However, the parameter R_i, the internal packet arrival rate, is a highly variable quantity. Measurements taken on the MILNET show that the range of possible values for this parameter is quite large. For this reason, the parameter A_i will be fixed and equation 6.22 will be treated as an expression of R_e as a continuous function of R_i. This function can then be computed for several different fixed values of A_i to account for possible variation in that parameter.

Thus, by computing R_e as a function of R_i and A_i, it is possible to determine the available application throughput as a function of the internal traffic conditions at the source node. The results of these computations are given in the next section.

E. RESULTS

1. Parameter Values

The external messages have been defined as ARPANET X.25 messages of maximum length. As discussed previously, this equals the maximum IP frame, 1007 bytes, plus the X.25 overhead, 7 bytes, for a total of 1014 bytes.

$$L_e = 1014 \text{ bytes} = 8112 \text{ bits per message}$$

This message produces M IMP-IMP packets of length L_p. Since L_p is the maximum possible value, it includes a data packet of 1008 bits. Added to this are the 128 bit IMP-IMP header and 72 bits of hardware

generated framing. The maximum value of M is 8 packets per message.

$M = 8$ packets per message

$L_p = 1208$ bits per packet

For all service time calculations the trunk line capacity is 50 Kbps.

$C = 50{,}000$

$A_p = (1208)/(50000) = 0.02416$

$A_e = M * A_p = (8)*(0.02416) = 0.19328$

The R_e versus R_i function will be computed for four values of L_i: 400, 500, 600, and 700 bits per packet. However, measurements for internal packet length taken on MILNET consistently produce values close to 500 bits per packet [18:p. 3]. Thus the results at $L_i = 500$ should be considered the most typical.

$A_i = (400)/(50000) = 0.008$ @ 400 bits
$A_i = 0.010$ @ 500 bits
$A_i = 0.012$ @ 600 bits
$A_i = 0.014$ @ 700 bits

R_e will be computed for values of R_i over the range 0 to 100 packets per second.

2. Maximum Message Arrival Rates

At this point, all the constants in equation 6.22 have been defined. The equation can be rewritten for L_i = 500 as follows:

$$R_e = (1 - (0.01)*R_i)/(0.19328)$$

The equations can be readily given for the other values of L_i as well.

The graph of R_e as a function of R_i is shown in Figure 17. As shown, a separate curve is drawn for each value of L_i. Note that all the curves converge to a maximum value of R_e = 5.17 messages per second. This message rate corresponds to a complete absence of internal traffic at the switching node (R_i = 0). Thus, this is the very best rate possible. Table V lists values of R_e for different values of R_i and L_i.

TABLE V.

MESSAGE ARRIVAL RATES BASED ON INTERNAL TRAFFIC

	L_i ->	400	500	600	700
R_i = 0	R_e ->	5.17	5.17	5.17	5.17
R_i = 20	R_e ->	4.35	4.14	3.93	3.73
R_i = 40	R_e ->	3.52	3.10	2.69	2.28
R_i = 60	R_e ->	2.69	2.07	1.45	0.83
R_i = 80	R_e ->	1.86	1.03	0.21	****
R_i = 100	R_e ->	1.03	0.0	****	****

**** means that the internal arrival rate is unstable.

Inspection of a MILNET cumulative statistics report (CUMSTATS) shows that the highest average packet

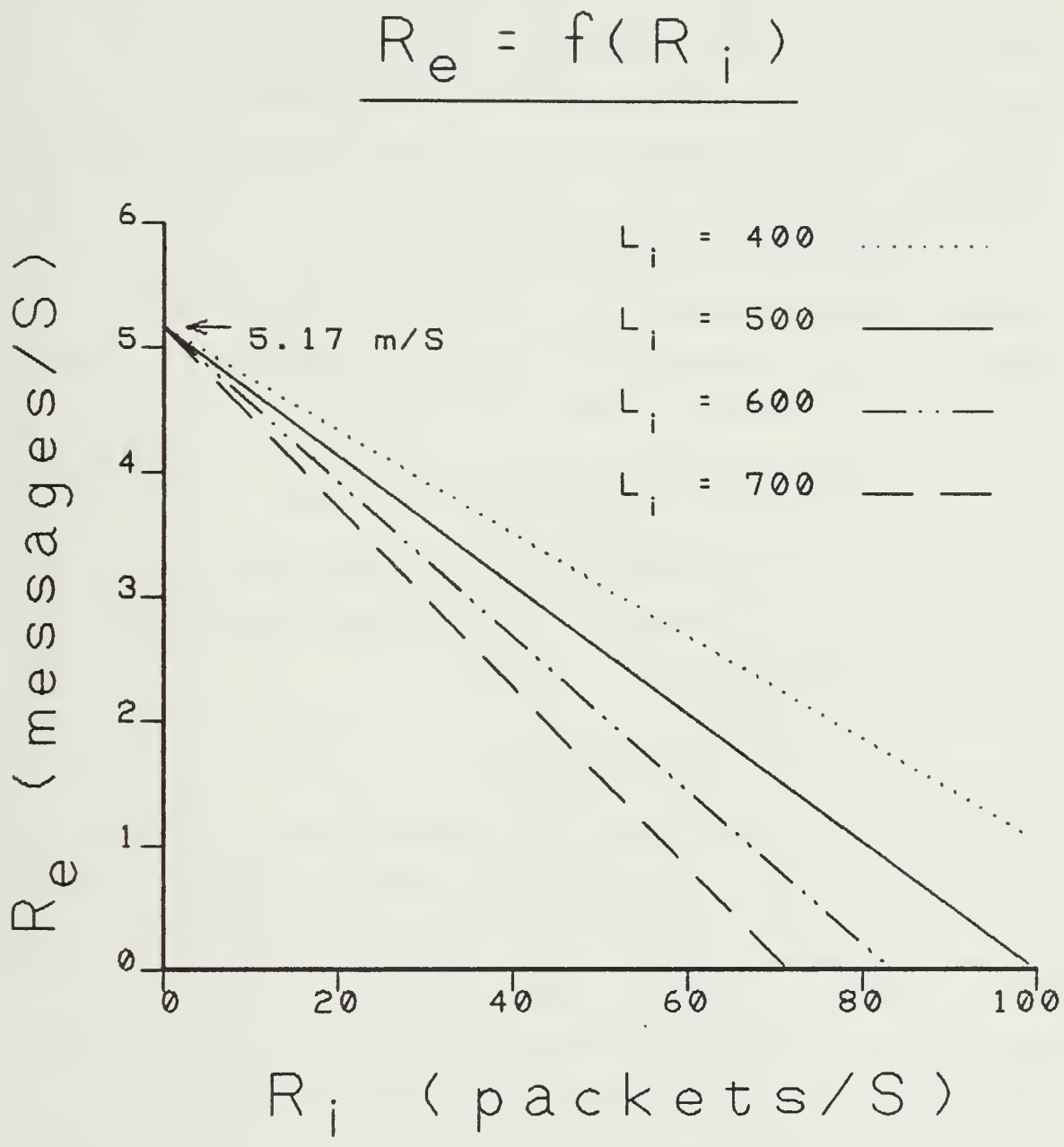

Figure 17. Graph of the Maximum Message Arrival Rate

rate for a single line over the hour of data collection was 46.4 packets per second [19]. The majority of the 50 Kbps lines on the network were operating at 10 packets per second or less. However, the value of R_i at a switching node is actually the sum of all the individual line arrival rates. Since the physical topology of the switching node is not specified, the entire range of values in Table V and Figure 17 must be considered possible.

3. Application Throughput

Since the external messages are of fixed length, the R_e values can be directly related to application throughput in bits per second (bps). The relationship is as follows:

$$\text{application throughput} = R_e * L_e \qquad (6.23)$$

The two internal traffic parameters, R_i and L_i, can similarly be combined into a single measure of traffic load in bps, as follows:

$$\text{internal load} = R_i * L_i \qquad (6.24)$$

The use of internal load as a single measure of network conditions at the switching node greatly simplifies the model because every value of L_i produces the same throughput versus load curve. In other words, it takes the same amount of time to service 5 packets of 1000 bits each as it does to service 10 packets of 500 bits each. Each of these represents a load of 5000 bits. This equivalence is a direct result of the assumption that packet processing time in the IMP was negligible.

The throughput versus load curve is shown in Figure 18. As shown, the maximum external throughput

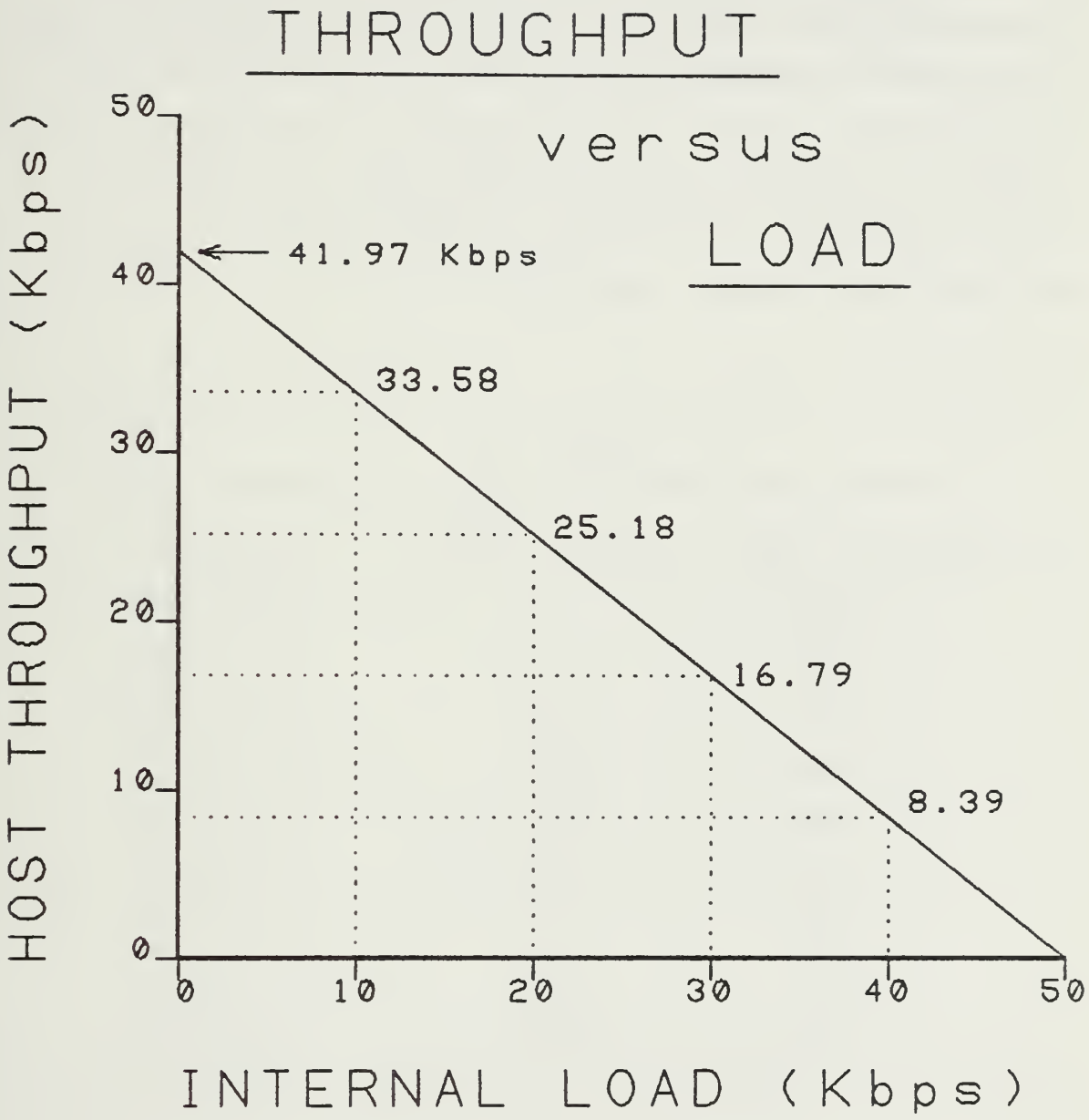

Figure 18. Graph of the Maximum Application Throughput

is 41.97 Kbps, the result of the case in which there is no internal load. This means that about 42 Kbps of application traffic to the source node produces 50 Kbps of output traffic from the node. Overhead associated with message disassembly and protocol conversion is responsible for this loss in capacity. Figure 18 also shows the throughput values for loads of 10, 20, 30, and 40 Kbps. A more complete list of throughput values is given in Table VI, which shows the internal load both in bps and in percent of output capacity. In every case, the host is only able to use about 84% of the available output capacity.

TABLE VI.
APPLICATION THROUGHPUT BASED ON INTERNAL LOAD

Internal load	Percent load	Throughput
0 Kbps	0 %	41.97 Kbps
5 Kbps	10 %	37.77 Kbps
10 Kbps	20 %	33.58 Kbps
15 Kbps	30 %	29.38 Kbps
20 Kbps	40 %	25.18 Kbps
25 Kbps	50 %	20.99 Kbps
30 Kbps	60 %	16.79 Kbps
35 Kbps	70 %	12.59 Kbps
40 Kbps	80 %	8.39 Kbps
45 Kbps	90 %	4.20 Kbps
50 Kbps	100 %	0.00 Kbps

MILNET measurements are available for traffic load in the form of line utilization [18], [19]. In these reports the most heavily used lines run at about

50% utilization, a load of about 25 Kbps. The majority of the 50 Kbps lines run at about 10% utilization or below. As before, internal load at the switching node is the sum of all the individual arriving line loads. Thus, the entire range of internal loads shown in Figure 18 and Table VI must be considered possible.

VII. DISCUSSION OF RESULTS

A. MAXIMUM LOADING AT CSF THROUGHPUTS

In Chapter V, CSF throughput requirements were obtained for two different simulations. Game 1 throughputs were very consistent at rates of about 33 Kbps for 2:1 speed and 16.5 Kbps for 1:1. However, these values were taken over a small portion of the simulation, so it is possible that a wider range of data rates may have occurred during the game. The game 2 data gives a much better view of the range of throughput requirements which occur over the course of a game. The rates are near zero at the beginning, and they consistently increase to reach their maximum values at the end. These maximums are in excess of 44 Kbps at 2:1 and 22 Kbps at 1:1.

Even though the throughput requirements of a simulation may vary greatly over time, the most important data rate to be considered is the maximum throughput. This is due to the real-time performance of these simulations, which makes it necessary for each game cycle update to be transmitted before the next cycle is processed. Thus, it is not possible to time average the high throughput periods with the low throughput periods to obtain a workable average throughput--the highest throughput requirement of the simulation must be accommodated.

Using the application throughput versus internal load results from Chapter VI, it is easy to determine what the maximum acceptable internal load is for each maximum CSF throughput value. These load values can either be estimated graphically from Figure 18, or they

can be obtained through interpolation using the data listed in Table VI. Maximum load values for games 1 and 2 are listed in Table VII. These values were obtained using the interpolation method.

TABLE VII.
MAXIMUM INTERNAL LOADS FOR GAMES 1 & 2

Simulation	Maximum Required Throughput	Maximum Acceptable Internal Load
GAME 1 CSFout @ 2:1	34.21 Kbps	9.25 Kbps
GAME 1 CSFout @ 1:1	17.10 Kbps	29.63 Kbps
GAME 2 CSFout @ 2:1	44.75 Kbps	****
GAME 2 CSFout @ 1:1	22.38 Kbps	23.34 Kbps

**** - required throughput exceeds switching node capacity

Several interesting observations can be made about the data in Table VII. For game 1, note that a 50% decrease in required throughput resulted in a more than tripling of the level of acceptable internal loading. In the case of game 2, the actual simulation (at 2:1 speed) reached data rates which exceed the capacity of the switching node even in the absence of any load. However, if this game were slowed to 1:1 speed it would be feasible to network it in the presence of loads in excess of 20 Kbps. Obviously, changing the speed at which a simulation is performed can have a dramatic impact on the amount of internal loading which the simulation can accept at the source node.

B. MEASURED INTERNAL LOADS

It is desirable for the simulation to be able to operate in the presence of significant internal loads, since the SIMNET hosts have no way to control or predict what the load at the source node will be. Unfortunately, without prior knowledge of the internal load it is impossible to predict whether or not a specific simulation will be able to operate over the network. While this uncertainty cannot be eliminated, it is possible to gain some insight into the problem through the use of MILNET internal load measurements.

No direct measurements of the total load at each node on MILNET were available, but it was possible to generate them from the line utilization data given in a CUMSTATS report [19] by summing the individual line data rates for each IMP on the network. The results of this work are shown in Table VIII.

TABLE VIII.
DISTRIBUTION OF IMP LOADS ON MILNET

Load Range in Kbps	Percent of IMPs	Cumulative Range (Kbps)	Cumulative Percentage
load < 5	40.4 %	0 - 5	40.4 %
5 - 10	15.5 %	0 - 10	55.9 %
10 - 15	13.2 %	0 - 15	69.1 %
15 - 20	6.6 %	0 - 20	75.7 %
20 - 25	7.4 %	0 - 25	83.1 %
25 - 30	4.4 %	0 - 30	87.5 %
30 - 35	2.2 %	0 - 35	89.7 %
35 - 40	5.2 %	0 - 40	94.9 %
load > 40	5.1 %		

As shown, the total load values were categorized into 5 Kbps load ranges. While the distribution of loads among the different ranges (first two columns) is interesting, the cumulative range data is more useful. By comparing this data to the maximum load values in Table VII, it is possible to gain some insight into the relative feasibility of operating these simulations over a packet switched network. For example, this comparison shows that just over 50% of the MILNET IMPs would be acceptable source nodes for game 1 at 2:1 speed, while about 87% would be acceptable at 1:1. This can be interpreted to mean that the 1:1 simulation has a much greater chance of operating successfully (87%) than the 2:1 simulation does (50% - 55%). Just as important is the observation that the 1:1 simulation has about a 13% chance of not being able to operate despite the relatively high level of loading that it can accept. Similar observations can be made about game 2, though it is clear that no MILNET IMP can accommodate the maximum throughput for the 2:1 simulation.

It is also possible to use the MILNET data to determine what the maximum application throughput should be to remain within a predetermined percentage of excessively loaded nodes. For example, if a level of 75% useful nodes is considered acceptable (25% excessively loaded) then the maximum acceptable load is about 20 Kbps. Using Table VI in Chapter VI, this corresponds to a maximum application throughput of approximately 25 Kbps. This means that a simulation with a maximum throughput of 25 Kbps will have a 75% chance of successful operation, based on the MILNET data. Similarly, a more conservative level of 90%

useful nodes (10% excessively loaded) results in a maximum throughput of about 12.5 Kbps.

Two points have to be emphasized about the use of these MILNET measurements. First, MILNET can only be considered a very general model of the possible eventual traffic on DISNET. At least initially, DISNET will have fewer IMPs, fewer lines, and fewer customers than MILNET has now. It is hard to guess whether this will result in lower or higher network loads on DISNET. Second, the MILNET data reflects measurements taken over a single hour of daytime network use. Thus, while the general form of this data is likely to be accurate much of the time, the specific percentages are not.

Having stated the limitations in the MILNET data, it is also important to point out that this is the best information that was available for the modeling of DISNET traffic. As long as these numbers are not treated as absolute parameters they can be useful in determining the nature of the application throughput problem.

C. DYNAMICS OF INTERNAL LOADING

So far, internal load at the source node has been treated as if were a constant value process. However, in Chapter VI the rate of internal traffic arrival at the switching node was described as being "very unpredictable at any moment." Thus, the values discussed for internal loads represent time averages of what may be a widely varying packet arrival process.

Computer data transmissions are often described as being "bursty" in that they typically involve short bursts of large amounts of data separated by periods of low data transmission. As the packet switches serve this burst traffic their behavior as single-server

queues will cause some smoothing in the traffic, but their output can still be characterized as being bursty. The arrival of several internal streams of this type of traffic can also be described as being bursty, since the different internal lines operate asynchronously with respect to each other. The bursty nature of the internal load at the source node is illustrated in Figure 19.

While the curves shown are specifically for a 25 Kbps load, the traffic behavior being shown occurs at all loads. Clearly, an internal load of 25 Kbps actually includes intervals greater than that value and intervals less than it. Of course, some measurement periods may have less variation than that shown in Figure 19 and some may have more. However, the most remarkable condition would be a period of significant length in which the load was constant.

The traffic behavior shown in Figure 19 has great significance for the problem of obtaining adequate throughput for SIMNET. Table VII states that game 1 at 2:1 speed can accept an internal load of 9.25 Kbps at the source node and still operate successfully. However, if that 9.25 Kbps represents the average load then the simulation will experience periods in which the available throughput is insufficient due to internal load bursts. These periods will be separated by periods of excess available throughput. It may be tempting to dismiss this problem, since it appears possible to store some application data during the burst periods and transmit it during the periods of excess capacity. However, it is not acceptable for real-time data transmissions to be stored and transmitted beyond their intended transmission time. The result for IBGTT would be simulation data

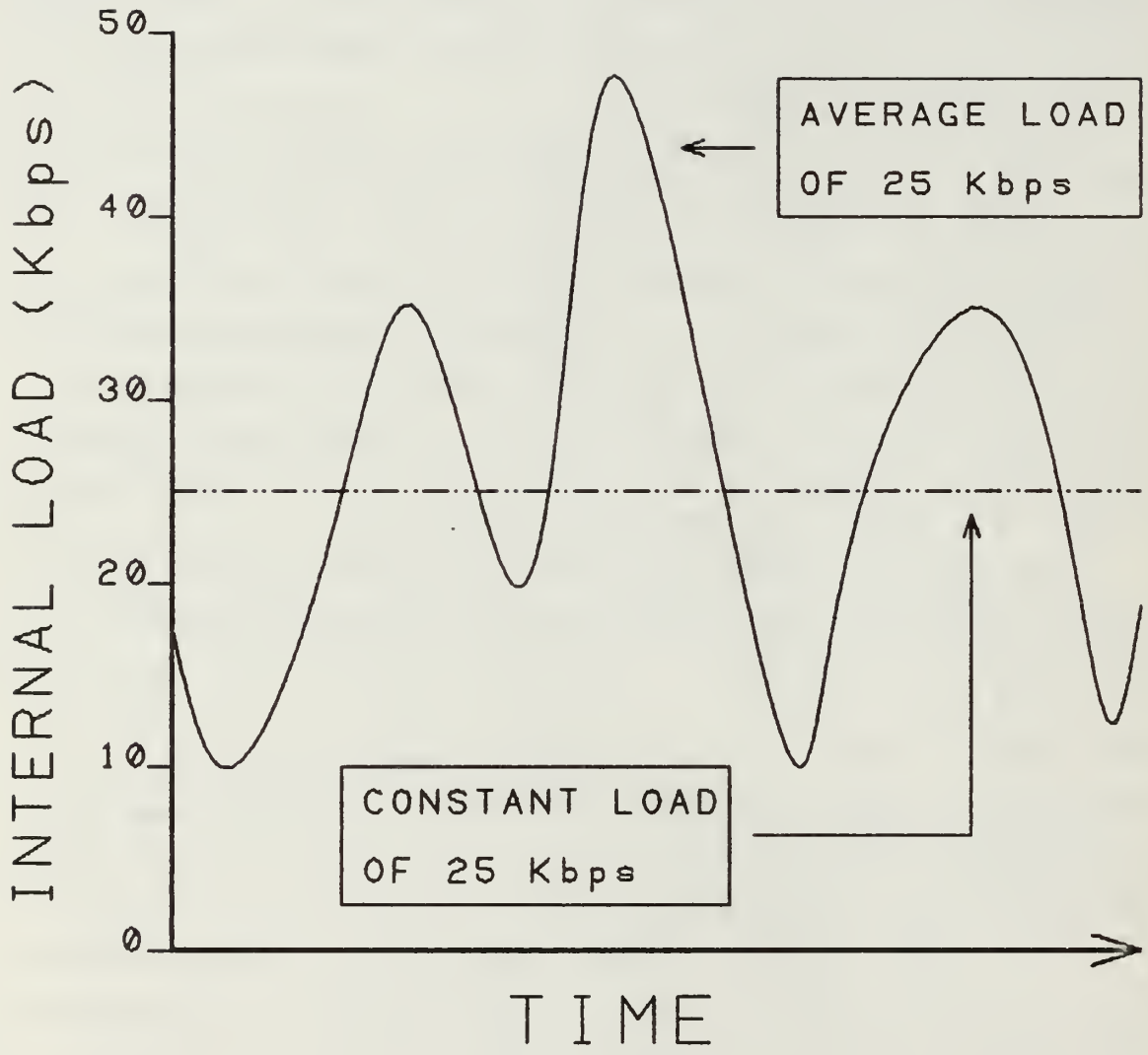

Figure 19. Bursty Nature of Internal Loading

transmitted at a time when it is already obsolete. Furthermore, the very large data transmissions involved make storage impractical over a period of several game cycles.

For these reasons, the maximum acceptable load values listed in Table VII must be considered to represent the maximum burst loads. However, due to the nature of burst traffic it is extremely difficult to predict what these values will be. Certainly the MILNET data presented in Table VIII is for average loads, and no measurements are readily available for the magnitude of the burst traffic during this measuring period.

A rather simplistic method for dealing with this problem is to apply a "burst factor" to the maximum loads in Table VII. For example, the maximum burst value could be estimated to be twice the measured average value. In this case, the burst factor, BF, would be Ø.5, and the acceptable average load could be determined from the maximum load as follows:

$$\text{avg load} = BF * \text{max load} \qquad (7.1)$$

Thus, for BF = Ø.5, game 1 at 2:1 could only accept an average load of about 4.6 Kbps. Using Table VIII, notice that this adjustment decreases the percentage of useful nodes to a value of less than 40%. A summary of the acceptable load values and useful node percentages for games 1 and 2 are given on the following page in Table IX, using BF = Ø.5 to calculate the average loads. Table IX does not include data for game 2 at 2:1 speed, since the question of internal loading is moot. Of course, the values in the last two columns can be easily found for any other burst factor.

TABLE IX

SUMMARY OF RESULTS FOR GAMES 1 & 2

Simulation	Maximum Load	Average Load	Useful Node Percentage
GAME 1 CSFout @ 2:1	9.25 Kbps	4.6 Kbps	40 %
GAME 1 CSFout @ 1:1	29.63 Kbps	14.8 Kbps	69 %
GAME 2 CSFout @ 1:1	23.34 Kbps	11.7 Kbps	60 %

Clearly, the percentage of useful MILNET nodes has been reduced significantly by this adjustment for traffic bursts. Note that a burst factor greater than 0.5 (representing smaller bursts relative to the average load) would have resulted in higher acceptable average loads and higher percentages of useful nodes. Conversely, a burst factor smaller than 0.5 (representing bursts greater than twice the average load) would have resulted in values even lower than those listed, possibly making the problem unworkable. Thus, it would be very useful to have an accurate measure for the burst factor--information not currently available.

D. USE OF DATA EXTRACTION

As discussed above, the bursty nature of internal network traffic may make it possible for a simulation to experience periods of insufficient throughput, even when the system operators had good reason to believe they were operating with excess throughput available. For this reason, it is important to consider what the effects of periodic insufficient throughput might be.

The immediate system response should be for the data extraction program to begin filtering the output data before putting it on the output queue, as described in Chapter III. As explained earlier, the use of data extraction for a prolonged period of time will result in portions of the RSM databases not being concurrent with the CSF database, an undesirable situation. However, if the data extraction program was only used for a very few game cycles--responding to a load burst of one or two minutes--then the database concurrency problem might not be too severe. This is because the RSM database could be brought up to date on the game cycle following the burst, possibly before any users had viewed the obsolete data. Obviously, the desired situation is one in which the data extraction program is not used, but for short load bursts the benefits of data extraction may outweigh the possible liabilities.

The above argument assumes that the extracted data can be transmitted during the burst period. In fact, this may not be the case. Any data being viewed in a user display must be transmitted, and for a simulation with more than one RSM this could be quite a lot of data. This problem is particularly acute when the Control station is being maintained, since this station has access to virtually all the simulation data. Thus, it is possible that data extraction will not always be sufficient, in which case the CSF will be forced to stop processing until the throughput needed to clear the output queue becomes available.

Actually, there is not a lot of information available on the use of data extraction under conditions of insufficient throughput. This is because IBGTT has only been operated in distributed mode over

an ETHERNET--an environment which should normally provide excess capacity. Therefore, the discussion given on the use of data extraction is based on an understanding of how the program was designed [5] and may not be an accurate description of how the program will actually perform. The important point is that the use of data extraction is generally undesirable because it creates remote databases which are not concurrent with the actual simulation status. Clearly, the operation of the data extraction program is an area for further investigation.

VIII. ADDITIONAL NETWORK FACTORS

A. PACKET ROUTING

As stated earlier, the IMP-IMP packets which are sent from the source node to the destination node may use different pathways to get there. The IMPS on DISNET use the ARPANET routing algorithm to determine which trunk line an individual packet should be transmitted over [20], [21]. The routing algorithm uses packet delay measurements to estimate the delay on each of its outbound lines. These delay values are broadcast throughout the network, allowing all the IMPs to use the information. Each IMP uses these delay values to generate a routing table which tells the IMP which source to destination path is the shortest (i.e., has the least delay). Under normal conditions, the shortest path is the one with the fewest hops (IMP to IMP transmissions).

The ARPANET routing algorithm is designed to be able to quickly adapt to changing network conditions. For this reason, each IMP updates its delay value estimates every 10 seconds. These frequent updates may cause the calculated shortest path between two nodes to change often. This is how packets being sent to the same destination may be routed over different paths.

The ARPANET routing algorithm impacts on the network analysis in two different ways. First, the high rate of IBGTT data is likely to quickly congest the downstream IMPs on the minimum hop path. As line delays along this path increase, the routing algorithm will choose other paths until the minimum hop path is less congested. This ability to adaptively route the

IBGTT data should enable the source node to transmit the outbound packets at a high rate.

Secondly, the high rate of IBGTT data through the source node will cause its lines to have high delay values from the perspective of the adjacent IMPs. This means that most of their traffic will be routed around the source node, possibly causing a reduction in the internal loading at the node. While this should be advantageous to a SIMNET host, it is also clearly detrimental to the traffic being routed around the node. If SIMNET causes significant congestion at some network IMPs, then other DISNET customers may experience serious performance degradation due to the use of very long packet routes.

B. FLOW CONTROL

All packet switched networks include flow control mechanisms. These mechanisms serve to prevent network congestion and resource overload by controlling access to the network. The ARPANET flow control mechanisms of interest to this thesis exist at two different levels: source node to destination node, and host to host.

1. Source Node to Destination Node

ARPANET uses a very simple algorithm for source node to destination node flow control [9:226-228], [22]. Each message transmitted from the source node to the destination node is sequentially numbered at the sending IMP. These numbers are checked at the receiving IMP and are used to sequence the messages. Both the source and the destination are restricted to a window, W, of sequence numbers. Thus, the source node can only have W unacknowledged messages in the network at any time. When the destination node receives a complete, correct message it acknowledges it to the

source node with a special packet called an RFNM (Request For Next Message).

The important question for this thesis is to determine whether or not this flow control will restrict the application throughput to values below those discussed in the last chapter. For ARPANET, the window, W, is usually set to W = 8. Since the SIMNET messages have been set to maximum length, the amount of data represented by this window can be easily calculated, as follows:

data(W) = 8 * 8112 bits/message = 64.896 Kbits (8.1)

The amount of time required to transmit this entire block of data is clearly dependent on the available application throughput. Transmit time for the best case (load = Ø) is the following:

time = (64.896) / (41.97 Kbps) = 1.55 sec. (8.2)

Thus, 1.55 seconds is the minimum amount of time required to transmit a full window of SIMNET messages onto the network from the source node.

The transmission of application data beyond this data block can not occur until at least the first message is acknowledged with an RFNM. In this case, if it takes longer than 1.55 seconds for the first RFNM to arrive at the source node, then the node will be forced to refrain from sending further application data until an RFNM does arrive.

On MILNET, measurements are commonly made of round trip delay (RTD), which is the interval between the time a message is sent and the time its RFNM is received. Average RTD for the network is normally

below 0.5 seconds [18:3]. Thus, RFNMs should arrive at the source node well before it has finished transmitting the data in its window. Of course, conditions of moderate internal loading will simply extend this data block transmission time to 2 seconds or more. Since RTDs are very seldom as long as 2 seconds, it is reasonable to conclude that this source node to destination node flow control mechanism will not normally cause a reduction in application throughput.

2. Host to Host

Flow control between the source host and the destination host is a function of the Transmission Control Protocol (TCP). As such, the details of this process were briefly described in Chapter IV. The purpose of flow control between hosts is to prevent the source host from sending more data than the destination host is capable of receiving. The destination host controls source host transmission by sending a window which declares how much buffer space is available for data. The source host can not have more data in transit than the amount stated in the most recent window [23].

For SIMNET, if small windows are used by the RSMs (representing small data buffers) then the CSF throughput could be restricted by this flow control mechanism. However, it is likely that the RSMs will be configured to provide large data buffers, resulting in large windows and unrestricted data transmissions. The evidence for this assumption is in the fact that large CSF to RSM data transmissions are commonly performed over ETHERNET with no apparent input buffering problems at the RSMs.

C. DESTINATION NODE FUNCTIONS

The analysis of network performance has been focused on the application throughput at the source node. As explained in Chapter V, this focus is based on the assumption that the critical bottleneck for the CSF to RSM data transmissions was at the source node. However, the destination node is responsible for some essential network functions which must be considered as well.

1. Message Reassembly

As described in Chapter IV, for a multi-packet message the individual packets are received by the destination node and are used to reconstruct the original ARPANET message. Since it is possible for the different packets of a single message to arrive out of order the destination node must be capable of holding these packets in buffers until they have all arrived. Every destination node has a set of reassembly buffers dedicated to performing this function [22].

However, it is possible that all this buffer space could become filled with packets which are awaiting other packets so that complete messages may be delivered. If this were the case, then any packets arriving that are part of new messages will be discarded for lack of reassembly buffers. This situation can degenerate into reassembly deadlock [24], a condition in which no message can get through the destination node.

To avoid reassembly deadlock, ARPANET requires that before a source node can send a multi-packet message it must reserve room for it in the destination node's reassembly buffers. For SIMNET, this mechanism could restrict throughput if the source node has to compete with other source nodes for buffer space at the

destination node. However, once the buffers are allocated to the SIMNET source node they can be held until the entire data transmission has been completed. Therefore, message reassembly should not impact on CSF throughput unless the source node is forced to wait for buffer space.

2. Destination Throughput

Normally the multiple paths between a source node and a destination node provide excess throughput once the data transmission has passed through the source node. In this regard, the source node has been viewed as the bottleneck on the packet switched network. However, the destination node could also be considered a bottleneck, since all the data being sent to its SIMNET host has to pass through it.

Nonetheless, the destination node will not usually restrict application throughput on SIMNET to the extent that the source node does, for two reasons. First, many of the simulations performed over SIMNET can be expected to have more than one RSM, just as games 1 and 2 did. With multiple RSMs the CSF output is divided among several remote hosts, which probably requires that it be sent to more than one destination node. This would mean that the required throughput of each destination node would only represent part of the required throughput at the source node.

Even if the simulation were configured with a single RSM, the restriction on throughput at the destination node will be less than that at the source node if the internal loading is equal. This is due to the effects of the overhead associated with message disassembly and protocol conversion, as discussed in Chapter VI. At the source node, the maximum possible throughput is about 42 Kbps because this rate of

ARPANET message traffic will produce 50 Kbps of IMP-IMP packet traffic. Thus, there is a 16% loss of throughput at the source node. However, when this IMP-IMP packet traffic reaches the destination node the reverse case is true. Now the 50 Kbps of packeet traffic produces 42 Kbps of message traffic, which is sent to the host on a 50 Kbps line. Therefore, the traffic arriving at the destination node faces 19% excess throughput. For this reason, even in a single RSM simulation, the bottleneck for CSF throughput can be assumed to be the CSF source node.

The one case where throughput may be more restricted at the destination node than at the source node would be a single RSM simulation with internal loading much greater at the destination node. While this situation is possible, it is also very hard to anticipate. If this did occur the source node would transmit at the throughput available to it, but the destination node would become overloaded by this rate of arrival. This would cause some of the IBGTT packets to be discarded, forcing the source node to retransmit them. These processes reduce the effective application throughput to a level that the destination node is capable of handling. However, they also cause congestion on the network because of the multiple packet copies sent, thereby reducing network performance for all the customers.

D. MULTIPLE HOSTS AT THE SOURCE NODE

The switching node model presented in Chapter VI described a single stream of external traffic arriving at the source node. So far, this external traffic has been modeled as consisting of only the CSF to RSM data

from a SIMNET host. However, this may not be a very good assumption.

Currently, each IMP on MILNET is connected to an average of three hosts [18:8]. Therefore, if DISNET resembles MILNET in this regard, it is likely that a SIMNET host sending CSF output may have to share its source node with one or more non-SIMNET hosts. Of course, these other hosts will be free to use the network resources whether or not a warfare simulation is in progress. Since there is a limit to the total amount of external traffic which can be handled by the source node, the output from these hosts will cause a direct reduction in the throughput available to the SIMNET host.

As discussed earlier, the transmissions from these hosts are likely to be bursty, making it very difficult to predict what the rate of non-SIMNET external traffic is going to be. For this reason, the values obtained in Chapters VI and VII for available application throughput should be treated as optimum values. That is, those throughputs can only be attained in the absence of competing host traffic at the CSF source node. Unless something is known about the behavior of these hosts, there is no way of knowing whether a simulation will be able to operate near its optimum level of throughput.

The opposite side of the above discussion is also important. From the perspective of these other hosts the large, prolonged data transmissions of a SIMNET host will greatly reduce their available throughput. This is also true at the destination node, which is equally likely to have multiple hosts attached. In this case, the other hosts have to compete with SIMNET for reassembly buffer space for their multi-packet

messages, as discussed earlier. At either of these nodes it is likely that the SIMNET traffic will cause a significant degradation in network performance for these hosts.

IX. CONCLUSIONS

One important question which this thesis has sought to answer is that of whether or not the IBGTT system can be successfully operated over a packet switched network based on the ARPANET architecture. It is clear from the results described in this thesis that because the level of available application throughput is dependent on a constantly changing traffic load, this question can not be definitively answered. However, several important conclusions can be drawn about the expected performance of the proposed SIMNET.

While it may be possible to operate IBGTT over DISNET, it will not be possible to do so without some constraints placed on the simulations. The discussion in Chapter VII indicates that neither of the two measured games is likely to work at the 2:1 speed they were originally operated at. It is probable that any simulation running over DISNET will be restricted to 1:1 speed during at least a large portion of its operation. If this does not seem like a significant penalty, consider the fact that game 2 occupied an entire working day using 2:1 speed throughout.

It is also possible that in addition to a 1:1 speed restriction it may be necessary to place restrictions on the size of the simulation. While there is no data in this thesis which directly relates required throughput to the number of objects in a game, it is clear that a game with a large number of objects will require more CSF to RSM data than a game with a small number. More extensive data collection will be

required before this relationship can be better resolved.

Even operating under speed and size constraints, it is likely that use of the data extraction program will be required at the CSF. However, if data extraction occurs on a prolonged basis then the RSM databases will not be concurrent with the primary simulation database in the CSF. This situation may create serious problems for the system's users when they change displays because they are likely to be given information from the RSM database which is incorrect. Very little information on the use of this data extraction program currently exists, primarily because its use has not yet been needed.

The presence of a data extraction program does not guarantee that a simulation will not be disrupted. The most significant characteristic of traffic on a packet switched network is that it is unpredictable. A burst of traffic at the CSF source node could cause a drastic reduction in application throughput, quite possibly forcing the CSF processor to halt execution until it has sufficient room in the output queue for its results. These traffic bursts could be due to either internal traffic from adjacent IMPs or external message traffic from a host connected to the same source node. Considering the fact that IBGTT simulations often run for many hours, the chances of one or more of these disruptions occurring during the course of a simulation may be quite good.

In view of the operating conditions described above, it is possible that SIMNET will not be able to provide the same tactical training experience that is currently provided by the IBGTT systems at NOSC and NPS. On the other hand, it may be possible to develop

simulation scenarios which provide effective training while operating within these constraints. This question can only be answered by the personnel who operate the system.

So far the question under discussion has been whether IBGTT can function effectively over DISNET. However, the opposite question is also important: Can DISNET function effectively with IBGTT as a customer? It is possible that because of its large, continuous data transfers IBGTT will create serious congestion in the vicinity of the SIMNET source and destination nodes. This congestion may significantly degrade the network performance provided to other customers, particularly for those hosts which are connected to the same nodes as SIMNET.

The premise behind packet switching is that it is more economical for many customers to share computer communication resources than it is for each customer to obtain their own. This system works because most computer systems only need to use the network periodically (i.e., in bursts). The sacrifice these customers make in return for economy is that they may experience some delay in their communications due to the dynamics of resource sharing. For applications such as electronic mail and file transfer these delays are not considered significant.

However, IBGTT does not conform to any aspect of this description of a typical packet switching network customer. IBGTT will not use the network in bursts. Instead, it will subject the network to continuous streams of application data. Furthermore, IBGTT can not afford to have its communications appreciably delayed while other customers use the network resources. Instead, it requires that its computers

transfer large amounts of data within small time frames. In short, IBGTT needs to perform continuous data transfer in real-time between its distributed computers. This is an application which current packet switching networks were not designed for. It seems reasonable to conclude from this that IBGTT is not an appropriate application for a shared resource, packet switching network based on the ARPANET architecture. While it is possible that future network architectures may be able to meet the throughput requirements of an apllication like IBGTT, these networks are not available today. For these reeasons, it appears that SIMNET could be better implemented using its own dedicated network resources.

LIST OF REFERENCES

1. Interim Battle Group Tactical Trainer Instructor User's Guide, V. 1, Overview, Naval Ocean Systems Center, 17 January 1983.

2. Interim Battle Group Tactical Trainer Introductory Student User's Guide, Naval Ocean Systems Center, 30 November 1983.

3. Barksdale, J. R., and Casey, P. A., Prime Item Specification of the User Support System of the Battle Group Training Computer Support Facility, Naval Ocean Systems Center, 20 November 1982.

4. Barksdale, J. R., and Casey, P. A., Prime Item Specification of the Simulation Support System of the Battle Group Training Computer Support Facility, Naval Ocean Systems Center, 20 November 1982.

5. NWISS Version 1.0 Program Design Report for Multiple CPU Architecture, Naval Ocean Systems Center, 19 November 1982.

6. Joint Directors of Laboratories, C3 Research and Technology Panel, Program Plan for Fiscal Year 1986, 9 August 1985.

7. The Defense Data Network - High Capacity for DOD Data Transmission, Bolt, Beranek and Newman Inc., April 1986.

8. Zimmermann, H., "OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection," IEEE Trans. on Comm., V. 28, N. 4, April 1980, pp. 425-431.

9. Tanenbaum, A. S., Computer Networks, Prentice-Hall Inc., 1981.

10. Postel, J. (Ed.), Transmission Control Protocol - DARPA Internet Program Protocol Specification, RFC 793, USC/Information Sciences Institute, September 1981.

11. Postel, J. (Ed.), Internet Protocol - DARPA Internet Program Protocol Specification, RFC 791, USC/Information Sciences Institute, September 1981.

12. Defense Data Network Program Plan, Defense Communications Agency, DDN PMO, May 1982.

13. Defense Data Network X.25 Specification, Defense Communications Agency, DDN PMO, January 1985.

14. Postel, J., "The TCP Maximum Segment Size and Related Topics," RFC 879, USC/Information Sciences Institute, November 1983.

15. Inose, H., and Saito, T., "Theoretical Aspects in the Analysis and Synthesis of Packet Communication Networks," IEEE Proceedings, V. 66, N. 11, November 1978, pp. 1409-1422.

16. Tobagi, F. A., et. al., "Modeling and Measurement Techniques in Packet Communication Networks," IEEE Proceedings, V. 66, N. 11, November 1978, pp. 1423-1446.

17. Rubin, I., "An Approximate Time-Delay Analysis for Packet-Switching Communication Networks," IEEE Trans. on Comm., V. 24, N. 2, February 1976, pp. 210-222.

18. MILNET Traffic Measurement Report No. 4 - Performance Analysis, Defense Data Network, CDRL Item S003, 11 June 1986.

19. Cumulative Statistics Report, Bolt, Beranek and Newman Inc., 21 July 1986, 08:45 to 09:45.

20. Schwartz, M., and Stern, T. E., "Routing Techniques Used in Computer Communication Networks," IEEE Trans. on Comm., V. 28, N. 4, April 1980, pp. 539-552.

21. McQuillan, J. M., Richer, I., and Rosen, E. C., "The New Routing Algorithm for the ARPANET," IEEE Trans. on Comm., V. 28, N. 5, May 1980, pp. 711-719.

22. Gerla, M., and Kleinrock, L., "Flow Control: A Comparative Survey," IEEE Trans. on Comm., V. 28, N. 4, April 1980, pp. 553-574.

23. Clark, D. D., "Window and Acknowledgement Strategy in TCP," RFC 813, USC/Information Sciences Institute, July 1982.

24. Lai, W. S., "Protocol Traps in Computer Networks - A Catalog," IEEE Trans. on Comm., V. 30, N. 6, June 1982, pp. 1434-1449.

APPENDIX A. DECNET DATA FOR GAME 1

Period Number	Period Length	RSM1 Bytes	RSM1 B/sec	RSM2 Bytes	RSM2 B/sec
1	62 sec.	69160	1115	174496	2814
2	63 sec.	70456	1118	176348	2799
3	62 sec.	70448	1136	175804	2835
4	62 sec.	70680	1140	175832	2836
5	62 sec.	71843	1158	176620	2848
6	62 sec.	71888	1159	178060	2919
7	65 sec.	71220	1095	153508	2398
8 - 12	no data				
13	61 sec.	64355	1055	166948	2736
14	63 sec.	72176	1145	177232	2813
15	63 sec.	72320	1147	179836	2854
16	62 sec.	70272	1133	175036	2823
17	63 sec.	69368	1101	174132	2808
18	62 sec.	69784	1125	174472	2814
19	62 sec.	71372	1151	176928	2853
20	62 sec.	69284	1117	174016	2806
21	63 sec.	72699	1153	176816	2806
22	62 sec.	67184	1083	171620	2768

APPENDIX B. DECNET DATA FOR GAME 2

Period Number	Period Length	RSM1 Bytes	RSM1 B/sec	RSM2 Bytes	RSM2 B/sec
1	901 sec.	54856	60	55792	61
2	901 sec.	287696	319	289224	321
3	901 sec.	668888	742	681248	756
4	902 sec.	800044	886	843756	935
5	903 sec.	907764	1005	1158960	1283
6	901 sec.	1021288	1133	1539248	1708
7	902 sec.	1095695	1214	1629496	1806
8	902 sec.	1142085	1266	1736576	1925
9	902 sec.	1153641	1278	1743520	1932
10	902 sec.	1125919	1248	1708432	1896
11	902 sec.	1229491	1363	1832284	2031
12	903 sec.	1280491	1418	1879384	2083
13 - 20	pause in the game				
21	903 sec.	1304136	1444	1911060	2116
22	903 sec.	1301249	1441	1937504	2145
23	902 sec.	1370736	1519	2482116	2751
24	902 sec.	1565004	1735	3045280	3376
25	902 sec.	1555668	1724	2988936	3313
26	903 sec.	1640800	1817	3173888	3518

BIBLIOGRAPHY

Cerf, V. G., and Lyons, R. E., "Military Requirements for Packet-Switched Networks and Their Implications for Protocol Standardization," Computer Networks, V. 7, N. 5, October 1983, pp. 293-306.

Davies, D. W., et. al., Computer Networks and Their Protocols, John Wiley & Sons, 1979.

Kleinrock, L., "Principles and Lessons in Packet Communications," IEEE Proceedings, V. 66, N. 11, November 1978, pp. 1320-1329.

Kleinrock, L., Naylor, W. E., and Operdeck, H., "A Study of Line Overhead in the ARPANET," Communications of the ACM, V. 19, N. 1, January 1976, pp. 3-13.

Leiner, B. M., et. al., "The Darpa Internet Protocol Suite," IEEE Communications, V. 23, N. 3, March 1985, pp. 29-34.

Pouzin, L., and Zimmermann, H., "A Tutorial on Protocols," IEEE Proceedings, V. 66, N. 11, November 1978, pp. 1346-1369.

Schwartz, M., Computer-Communication Network Design and Analysis, Prentice-Hall Inc., 1977.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code 62Rr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Prof. M. L. Cotton, Code 62Cc Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	3
5.	CDR J. S. Stewart, Code 55St Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	2
6.	LT J. L. Paige Naval Ocean Systems Center Attn: Code 30 271 Catalina Blvd. San Diego, California 92152-5000	2
7.	Prof. M. G. Sovereign, Code 74 Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1
8.	MAJ T. J. Brown, Code 62Bb Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
9.	Bill Dejka Naval Ocean Systems Center Attn: Code 411 271 Catalina Blvd. San Diego, California 92152-5000	1
10.	Jim Goertz Naval Ocean Systems Center Attn: Code 411 271 Catalina Blvd. San Diego, California 92152-5000	1

11. Kyle S. Adler 1
BBN Communications Inc.
50 Moulton St.
Cambridge, Massachusetts 02238

12. Stephen N. Cohn 1
BBN Communications Inc.
50 Moulton St.
Cambridge, Massachusetts 02238

13. H. Miller, Code 33T 1
Naval War College
War-Gaming Dept.
Sims Hall C109
Newport, Rhode Island 02841-5010

14. G. Baker 1
Space and Naval Warfare Systems Command
PMW 161-22
Washington, D. C. 20363-1500

15. CAPT Mike StJohns 1
Defense Communications Agency
Code B612
Washington, D. C. 20305-2000

16. Director [Attn: R. Lyons] 1
Defense Communications Agency
Washington, D. C. 20305

17. CDR G. Porter 1
Tactical Training Group, Atlantic
FCTCLANT, Gallery Hall
Dam Neck
Virginia Beach, Virginia 23461

18. Dr. A. E. Brandenstein 1
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, Virginia 22209

19. USA CECOM 1
AMSEL-SEI-F [Attn: Dr. I. Myke]
Ft. Monmouth, New Jersey 07703

20. Dennis C. McCall 1
Naval Ocean Systems Center
Attn: Code 443
271 Catalina Blvd.
San Diego, California 92152-5000

221150

Thesis
P1185
c.1

Paige
Requirements analysis for the operation of a real-time warfare simulation over a packet switched computer network.