DEVELOPMENT

OF

AN IN-HOUSE COMPUTER NETWORK KUIPNET

Shigeyoshi KITAZAWA

Department of Information Science

Kyoto University

December 1976

DEVELOPMENT
OF
AN IN-HOUSE COMPUTER NETWORK KUIPNET

Shigeyoshi KITAZAWA

December 1976

Department of Information Science
Kyoto University
Kyoto, Japan

Submitted to
the Faculty of Engineering of Kyoto University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

DEVELOPMENT
OF
AN IN-HOUSE COMPUTER NETWORK KUIPNET

Shigeyoshi KITAZAWA

ABSTRACT

The Kyoto University Information Processing NETwork (KUIPNET) is a network connecting several computers located in one house. Its purpose is to realize resource sharing in information processing. The six local heterogeneous computers are connected with a switching computer through high-speed lines (1.6 Mbps), and a remote computer is connected with the switching computer through a 4,800 bps line. General and efficient facilities for interprocess communication have been implemented in the medium sized computers, and the in-house computer network has been successfully connected, as mentioned above, to a remote computer. The priority of the channel of the switching computer is controlled to guarantee a stream of high-speed transmission (800 kbps) concurrently with other streams of low-speed transmission. Speech and picture data, files, and character strings can be transferred among the computers in this network, which allows for the mutual resource sharing. The performance so far of the network, measured by specially prepared tools, has turned out to fulfill the required transmission rate of the real-time speech wave form—200 kbps.

ACKNOWLEDGMENTS

I would like to express my hearty gratitude to Professor Toshiyuki Sakai for his guidance, constant support and encouragement during the course of development of the KUIPNET, and also to Associate Professor Koh-ichi Tabata for his continuing support. Dozens of students and members of the staff have contributed to the implementation and testing of the system described in this thesis, but special appreciation is extended to Associate Professors Shigeharu Sugita (National Museum of Ethnology) and Takeo Kanade, Dr. Kenji Ohtani, Mr. Tsunetoshi Hayashi, Mr. Hirokazu Ohnishi, Mr. Kimio Maekawa, Mr. Hideji Saka, Mr. Michiaki Matsu-ura, Mr. Kiyoshi Ito, Mr. Hiro-o Ueda, Mr. Masakazu Tanaka, Mr. Kenichiro Yanagi, Mr. Yasuo Ariki, Mr. Naoto Fujiwara, and Mr. Hirofumi Fujiwara.

In completing this thesis, I wish to thank to Professor Shuji Doshita and Mr. Susumu Yamasaki for their support and lastly I would like to thank to Miss Gretchen K. Mittwer for her effort in correcting the manuscript.

TABLE OF CONTENTS

CHAPTER I: INTRODUCTION

In this thesis a computer network means "a resource sharing computer network." In most computer networks, the computer systems are geographically distributed. A resource sharing computer network located in a single vicinity, or located in-house, however, can be built efficiently and with less difficulty. High-speed transmission capability is the main asset of the incorporation of computers on the premises. This thesis describes the in-house computer network KUIPNET (Kyoto University Information Processing NETwork), in which heterogeneous computer systems are connected together with a high-speed switching computer to support various researches on information processing.

We describe in this first chapter our motive behind the construction of this network. Then its development is chronologically explained.

1.1 Motive for Development of KUIPNET

KUIPNET has been developed funded by our intensive motivation to carry out research on information processing [Sakai, 1973]. The word "information" is used here to mean one of the following: (1) data, programmed in coded form, (2) waveforms such as speech wave, (3) line drawings, figures, and grey-scale or color photographs. Our main goal has been to give computers a capability to understand the intentions of we human beings who use them as a powerful partner in our problem-solving process. Computer systems also greatly facilitate our studies in information processing research. Through computer processing utilizing powerful question-answering mechanisms, researchers both individually and collectively have spent years of great effort to understand natural speech, handwritten and printed characters, or natural scenes. Such studies would be impossible without the aid of computer systems. For the input/output of speech data in speech recognition or synthesis, the on-line real-time mode of computer system has been used.

From the beginning of our research through 1970, we depended on only a single computer system utilizing the on-line real-time and batch processing methods. In 1971, a new minicomputer-controlled terminal system which handled the input and output of pictorial data was installed. This minicomputer and its inclosing terminals and peripheries plus our conventional computer system made up a computer complex on which an interactive picture-processing system was constructed [Kanade, 1973]. Another minicomputer-controlled terminal system which handled the input and output of speech data was then installed. Although this computer system made it capable for us to carry out fundamental experiments on speech recognition and the synthesis of single words, it was still necessary to connect it with a larger computer system which would provide a higher level understanding mechanism for long and continuous speech sentences and a large-volume high-speed file storage.

From our experience, what would be most effective for our purpose was an on-line real-time information-processing system with powerful question-answering capability. This requirement might have been fulfilled by composing a computer complex according to individual application as mentioned above. However, there is a great difference in organization philosophy between a computer complex and a computer network.

A computer complex may be considered a special form of network in which the various components are integrally linked within a single facility. The relationship between two computers in a computer complex is as that between a parent company and its subsidiary. From when the system is in the stage of being designed, the functional relation between the two computers is usually specified in detail and fixed. On the other hand, a computer network is a system in which computers are autonomously combined without altering their domestic control. Computer systems communicate with each other in order to share certain resources such as programs or data—and/or for load sharing and reliability reasons. "To share certain resources" means that a certain resource is made useful for a user who could not use it before in a productive manner. If a certain resource is usable for one user in the network, that resource can be utilized by any other user in the network by following the way the first user utilized it. The resource becomes, as it were, the common property of the network. May I remark that it is the uniformity of the network structure which, notwithstanding the independency of the Hosts' dissimilarities, allows the establishment of common procedure, or "protocol". The generality and non *ad hoc* approach of the computer network satisfies our purpose.

Eventually we decided to build a communication network among the already distributed processors. In 1972, the computers were connected effectively by a communication network, each component of which had been developed for different purposes of information processing and possessed its own characteristic terminals and peripheries that the other computers could not provide.

There are many computer networks intended to share certain resources [Farber, 1972]. We will outline below several typical networks, the motive for the construction of which seems partially common to ours.

DCS— The Distributed Computer System (DCS) [Farber, 1972] is an experimental computer network under development at the University of California at Irvine. It is intended to service mini and midi scale computers. Its communication facility is a ring network, whose data rate is 2.3 Mbps. Under its particular hardware and communications facilities, a network-wide standard process naming and addressing protocol was incorporated.

DCN— The Distributed Computer Network (DCN) [Mills, 1976] is a resource sharing computer network under development at the University of Maryland. The DCN is implemented on several PDP11/45s; therefore the composition is homogeneous. An important goal of the DCN is to provide the user with a uniform interprocess communication mechanism, independence of process location, and a process structure that is identical throughout the network.

ARPANET— Perhaps the most widely publicized computer network of the type we focus on is ARPANET [Roberts, 1970], funded by the Advanced Research Projects Agency, which connects heterogeneous computer sites *via* a packet-switching communication network. The communication network is composed of IMP

(Interface Message Processor) node computers which perform store-and-forward message switching [Heart, 1970]. These IMP node computers are distributed nationwide. A Host computer is a subscriber connected to the network *via* a local IMP. In the ARPANET most Host computers offering service are time-sharing systems (TSS). A remote time-sharing system may be used *via* a local terminal through the network as if it were a local terminal of the system. Most data transferred through the network are characters from/to various key-board terminals.

Very high-speed transmission of raw data is the key for an in-house computer network. One advantage of our KUIPNET network is that it is confined to a building or a local network wherein communication costs are not critical. Most data communication facilities are composed of lines leased from the common carrier and coupled with various concentrating units. In a single building or campus, communication facilities may be composed of private-exchange lines or high-speed lines, available with only a small initial investment. A computer network need not cover the entire nation. Indeed, we believe a computer network limited, for example, to a building or campus also has merits in its own right.

The need for such local networks is becoming increasingly common. The greater the number of computers housed in a common center, the more intense the need for inter-communication. Non *ad hoc* solutions to this particular problem is very important, and one such solution which we are working on is general wideband interfacing. Recently a large special communication system was reported in the ARPANET [Mann, 1976] wherein a front-end computer provides connections among several on-site Host computers, a number of shared peripheral control computers, several network ports, and several hundred terminals of various types.

The benefits expected from our computer network are as follows:

(1) Facilitation of communication—If a certain job requires the interactive usage of more than one computer, this necessitates automatic switching from one computer to another at electronic speed. In other words, the computers are to be switched over automatically as the program runs. This characteristic is also essential for dynamic and efficient process-process communication and is similar to the role of the electronic exchanger device in the tele-communication system. It may be possible for a minicomputer to be used as an exchange device. Note that the data which passes through the exchanger is not only command characters but also high-speed large-volume raw data (*e.g.*, speech data or pictorial data).

(2) Hardware sharing—The total cost of peripheries is very large if each computer is fully and individually equipped, but the store-and-forward method for communication between computers and terminals makes it possible for a group of computers to share the file storage or peripheries as common storage or terminals.

(3) Distributed computation—In order to increase both efficiency and cost performance, it is sometimes necessary to decentralize the processing power and distribute it among many processors which are more suitable in their processing capabilities or hardware. No matter how high the degree of integration may be, the variety of functions

and amount of data that can be accumulated in one place are limited. A larger system may be constructed using computers which individually do not have enough capacity to house the complete system. This type of layout is anticipated to greatly improve the efficiency of the whole system and to open up a new field of computer application. Subsequently, however, a problem of specialization arises in such a community of computers. In each computer it would be desirable to have characteristics that the other computers cannot provide. It would be very advantageous to construct a network by connecting computers that have distinguishing features.

(4) Protocols—One of the most important benefits of our KUIPNET system is the homogeniety of the network structure, accomplished through the design of the communication network and inter-computer communication control mechanism, and finally the protocols on the user level. If a user himself had to prepare an enormous amount of complicated procedures at all points, a computer network would be impracticable. This has greatly affected the development of the resource-sharing network. In our system, as resource production activities are very high, a resource is available also for other users if it follows an authorized method and is usable anywhere in the network, this is made possible through the construction of protocols.

Although the new concept of the ARPANET resource-sharing computer network has influenced us very much, the concept and motivation behind the in-house computer network which we have developed is different. The ARPANET offered us a possible solution to the make-up of a resource-sharing computer network.

The network combines many TSS computers and realizes resource sharing—the interactive use of remote computers. The kind of input/output information to and from a TSS computer that is joined to the computer network is usually in the form of a command consisting of several characters. Communication of short messages *via* a key-board may be sufficient for a TSS computer network; however, the prospective computer network which we were trying to develop was not to be used only for key-board communication but also for the transmission and processing of data from speech waves, drawings, and photographs. In these cases, it would be necessary to send a great deal of what we call "raw data". Such raw data is very difficult to transmit on a voice-grade communication line (from 50 to 1,200 Baud); it needs an extremely high-speed communication line (about 10^6 bps). To permit the transmission of raw data, the specifications of the network design would be somewhat different from those of ARPANET.

1.2. Development of In-house Computer Network KUIPNET

Let us here look back to when our computer network project was started and describe in brief the history of its development.

Fundamental Studies of Computer Networks:

When we began the project in 1972, an initial investigation on computer networks was first made especially on the ARPANET. Through the investigation we came to think that the most important concept of such a computer network was "resource sharing"; and this was well realized in the ARPANET. So, the basic design of our network was directed toward constructing a heterogeneous resource-sharing computer network for information-processing research.

The first version of our design specifications were based on a message-switching computer (IMP) in August, 1972, and the high-speed Host-IMP connection hardware was standardized for in-house use in September of the same year. Experiments using the above-stated standardized interface hardware on communication between NEAC 2200/200 and TOSBAC 40 in the distance of 400 m were successful, achieving a 1 Mbps transfer rate. After this we began to design layered protocols for inter-process communication; IMP-Host protocol and Host-Host protocol. In February, 1973, the first version of an NCP (Network Control Program) was implemented on NEAC 2200/200 to test these IMP-Host and Host-Host protocols.

The following month, an experimental resource sharing was demonstrated using two computers, NEAC 2200 and TOSBAC 40, connected by the above-stated hardware. Wave form data sent from NEAC 2200 to TOSBAC 40 was processed *via* a Fast Fourier Transform program, and the resultant spectrum data was sent back to NEAC 2200 and displayed on a line printer. NEAC 2200, a medium-sized general-purpose computer, performed as an input file machine and an output file machine in this session. TOSBAC 40 provided its high-speed processing capability but relied on the other computer, which had various peripherals, for data input and output. This was an example of peripheral sharing and processor sharing.

Foundation of KUIPNET:

April, 1973 saw the commencement of our in-house network project, which was named KUIPNET (Kyoto University Information Processing NETwork). Many staff members of our laboratory were put to work on the implementation of its hardware and software. The interface hardware for each Host and the NCPs, and also for the IMP and its message-switching program had to be materialized.

In designing both the hardware and software of the IMP, the main problem was how high-speed raw data typical to speech could be made to go through the IMP. This point was our challenge, since it was not clear whether computers of the time (stored-program digital computers) could possibly perform such high-speed switching. A hardware circuit-switching system would have high-speed switching capability, but would be costly and not flexible. Programmability was indispensable for our system, because it might be extended to a wider network or altered to test other control algorithms. After considering several possibilities, store-and-forward message switching with a minicomputer was decided upon. We employed a new idea to allow co-existing real-time data transfer and character-oriented transfer; a programmable priority control circuit which assigns channel capacity to any channel designated by a message.

The basic design of the IMP interface circuit was modified to adjust to the interface condition of each Host; for MACC 7/F and MELCOM 70. By March the design of the interface circuit for the IMP side of the connection had been drawn up and was ordered to be etched on the card. The necessary number of identical printed-circuit cards was produced.

By June, 1973, the design and implementation of the Host monitor, NCP, and File Control Program for NEAC 2200 were almost completed, and the operating monitors and NCPs of minicomputer Hosts MACC 7/F and MELCOM 70 had been embodied. Around July, 1973, implementation of the NEAC monitor, NCP, and File Control Program was completed. Debugging of the MACC's NCP and monitor, and the MELCOM's NCP and monitor was finished in the middle of August.

The wiring and assembly of parts onto the IMP interface board then began, and it too was debugged. In September, IMP hardware tests were carried out and several troubles were solved. An IMP-Host connection test was carried out including a Host interface hardware test. Cables were routed between the IMP and Hosts.

The first version of our IMP program was implemented and tested in October. This version contained only a core of IMP functions sized about 1K words. Through November, Host-IMP-Host communication tests, NCP-NCP communication tests and process-process communication tests were carried out, and the Host NCPs, monitors and IMP were debugged.

The first demonstrative specimen job, which included speech data, pictorial data and character-transfer over the network, was successful. And by this time dubugging of the total system had been completed. The HITAC NCP was implemented in February, 1974.

Improvement and Enrichment of the IMP, Hosts and User Processes:

Through 1974, improvement and enrichment of the IMP, Hosts and user processes was being advanced. A revised version of the IMP program was installed, which increased its reliability, function and efficiency. Measurements were taken on the performance of the IMP and the network throughput, and it was decided to implement the IMP with a Terminal IMP (TIP).

The original NEAC 2200/200 Host computer was replaced by the more powerful NEAC 2200/250 which works on a disk base operating system and possesses a larger main memory size, a scientific calculation unit, and a memory protection mechanism. On this machine, more authorized methods of network access have now been implemented by Hayashi[1977]—the NOS(Network-Oriented Operating System). The MELCOM 70 Host computer was enhanced by a powerful operating monitor which allows multi-job operation, and equipped with an additional 8K-word memory.

As sophisticated user-level application programs appeared, many difficulties on inter-process communication and on synchronization of independent processes in the separate computers sprang up. HITAC 8350 was used through the network from MELCOM 70 for Fast Fourier Transformation of on-line transferred speech data. TELNET was implemented in MELCOM 70 and in the IMP which controlled file access in NEAC 2200. The network-shared memory was installed in the KUIPNET under U-200's control.

Extension of KUIPNET:

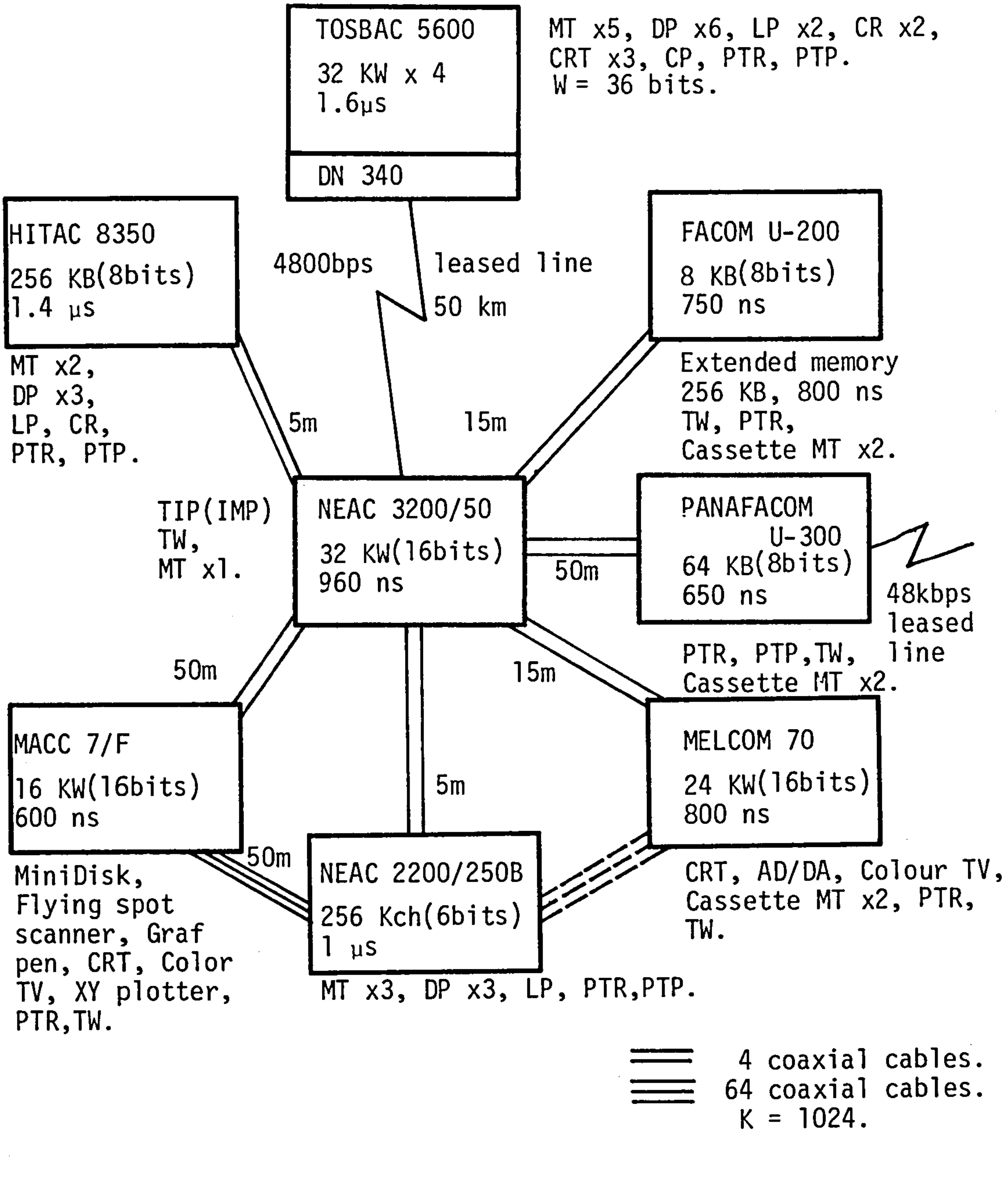
In 1975, a new version of the Terminal IMP(TIP) and also of the Server TELNET were implemented around the IMP. Magnetic tape file transfer became possible between the IMP and NEAC 2200. A Very Distant Host, TOSBAC 5600 in Osaka, was connected to the IMP *via* a 4,800 bps line to provide a GCOS time-sharing system accessible through the TIP Server TELNET from the Host TELNET system in KUIPNET. Process-to-process communication in transparent mode was also made possible over this connection. The Picture-Processing Program [Kanade, 1976] also became accessible through the IMP from TOSBAC 5600.

KUIPNET was hooked up with the university network which connects Kyoto and Tokyo by a 48 kbps line. In order to do this, protocol conversion was necessary, and for this purpose a Host U-300 was connected to the IMP. The establishment of a connection and flow of data between the two different networks were controlled *via* this gateway.

The network-shared memory, composed of the U-200 and 256K-byte IC-memory, was authorized to have access in the network through the "procedure call" protocol supervised by the POEM (Paging Oriented Executive Monitor) [Tanaka, 1976] in U-200.

In order to gather data about the traffic between the Hosts, the IMP load and generally about network usage, a network measurement system was implemented in the IMP.

Manuals describing use of the KUIPNET were published in March, 1976 [Kitazawa, 1976; Hayashi, 1976; Kanade, 1976].



APRIL 1976.

Figure 1.1 Configuration of KUIPNET.

1.3 Configuration of KUIPNET

The configuration of KUIPNET is shown in Figure 1.1. Fundamentally, the Hosts are connected in star fashion to the message-switching computer IMP. The types and characteristics of the Hosts in the network are as follows:

NEAC 3200/50— a message-switching computer IMP, it possesses a 32K-word (16 bits/word) core memory filled to capacity, 0.96 μsec memory cycle, and 16 slots of Direct Multiplex Channel (DMC) with priority control. It takes four memory cycles to transfer a word *via* DMC channel. The order of priority of the channel is programmable *via* the control logic that is specially designed for this purpose. This computer is equivalent to Honeywell's DDP-516. Other programmable features are; interruption, memory protection, and 250 μsec interval timer. This computer is equipped with a teletypewriter, a paper-tape reader, a magnetic-tape handler, a 4,800 bps communication controller and modem. Currently the IMP is connected with 5 local Hosts, a gateway, and a Very Distant Host in Osaka *via* a 4,800 bps communication line. This computer serves not only as a message-switching device but also handles the communication between the Host in Osaka, the User and Server TELNET, and takes care of statistics.

NEAC 2200/250— a medium-sized machine equivalent to Honeywell's 1250, this Host is equipped with a disk, magnetic tape, line printer and 256-kilocharacter main memory. Around this computer a new operating system has been developed which provides multiprogramming and multi-network-entry facilities [Hayashi, 1977].

HITAC 8350— a micro-programming computer with multi-programming operating system, its on-line user is restricted to one.

MACC 7/F— a minicomputer for the pattern recognition and picture-processing group, this mechanism consists of a 16K-word (16 bits/ word) core memory, 0.6 μsec memory cycle, the on-line control of a flying spot scanner, a Graf-pen, CRT, and Color TV display. Recently, however, it has almost retired from the KUIPNET computer network and matured as a front-end processor of a computer complex with NEAC 2200/250 for picture processing.
MELCOM 70— a minicomputer for the automatic speech-processing group, this is equipped with a 16K-word (16 bits/word) core memory, 0.8 μsec memory cycle, A/D converter, D/A converter, CRT, and cassette MTs. Since a large volume of high-speed speech data is transferred in real-time through the network to NEAC 2200/250 to be stored into the disk file, MELCOM 70 is the keenest computer of the network.
FACOM U-200— a minicomputer with common-bus architecture, the 256K-byte IC-memory is attached to the common-bus through a paging mechanism. An operating system POEM (Paging Oriented Executive Monitor) [Tanaka, 1976] controls memory space allocation to multi-users of remote Hosts, and this system provides server-oriented facilities—namely, PCP (Procedure Call Protocol)—for the remote use of its sub-systems.
TOSBAC 5600— a large general-purpose system located in the Kansai Institute of Information Systems in Osaka, this computer has participated in KUIPNET since 1975 *via* a 4,800 bps communication line. TOSBAC 5600/140 (equivalent to Honeywell's 6040) is seen as a "Very Distant Host". Its connection to KUIPNET allows bit-transparent transmission, and the GCOS operating system provides TSS service which is accessible from KUIPNET *via* TELNET connection. GCOS also provides a powerful file system, LISP 1.6, and program packages for general picture-processing. An interactive picture-processing system has been developed over this connection.

PANAFACOM U-300— a minicomputer acting as a gateway to another computer network, this is connected with a front-end processor in the data-processing center of Kyoto University. Through this, the network will be extended to Tokyo *via* a 48 kbps line.

Problems discussed in the following chapters are:

- the design and implementation of a communication network for in-house use allowing high-speed real-time raw data transmission. This includes the design of a message-switching computer for an in-house computer network, its hardware and software.
- the connection of rather small non-timesharing computer systems as Hosts to the computer network. This includes the design of the monitor and network access mechanism— a Network Control Program.
- the implementation of user-level protocols, such as the TELNET system.
- the connection with a remote, large computer system *via* a public communication line.
- high-speed data transmission between in-house computer systems, digitized real-time speech signals, and pictorial data transmission.
- the measurement system for an in-house computer network.

18 項欠

CHAPTER II: DESIGN OF KUIPNET'S IN-HOUSE COMMUNICATION NETWORK

A communication network is a facility which provides connection between computers, and through which data are transferred. In this chapter we shall show how the in-house computers are connected together and how the high-speed communication is guaranteed.

2.1 System Design of Communication Network

The computer communication network described below is inherently designed to provide service to local subscribers of heterogeneous types of computer systems, ranging from middle-scale general-purpose machines to simple minicomputers handling a variety of input/output devices for information processing, and of which are in a single building. It provides a consistently short response time for interactive traffic and real-time traffic—*e.g.*, transmission of speech—while maintaining a fairly large bandwidth to handle high-volume (bulk data) users.

The Communication Network

There are two major approaches to designing a network on the premises. One uses hardwired componets, such as the ring system based on the Bell System T1 technology, and the other is the store-and-forward system employing message-switching computers. The former incurs modest start up and low expansion costs but is not programmable. The latter is programmable and the messages can be processed according to their contents. Because of these reasons, the store-and-forward system therefore had greater advantages for us, providing rich flexibility and extensibility which were our key requirements, and so we chose it for our communication network.

Within the class of network under consideration, there already existed several operational networks and many designs for such a network, mentioned in Chapter I.

The following are some brief definitions to isolate the kind of computer network we were concerned with:

- Node (IMP)— The node computer of the network is a real-time computer which performs the basic message-switching functions.
- Hosts— The Hosts of the network are computers, connected to the node (IMP), which are the providers and users of the network's services.
- Message— The unit of data exchanged between source Host and destination Host.
- Store-and-Forward Subnetwork— The node computer (IMP) both stores a message that it receives and forwards it to the destination Host. The essential task of the "subnet", which functions as a communication network, is to transfer bits reliably from a source Host to a specified destination. Its operation is completely autonomous; the IMP continues to operate whether the Host is functioning properly or not.

We applied the Host-subnetwork configuration because isolation between independent Hosts, which are of different architecture made by different manufacturers, was necessary.

Though the subscribing computers of our network were not distributed widely but rather located together in the same building, a communication processor was found to be necessary. The key functions or properties required of this communication processor were:

a. Buffering— Buffering is required because it is necessary to receive multiple data units before sending a previously received one. Because of the difference in computer speed, it would even be desirable to have the buffering data in flight between source and destination in order to increase throughput. That is, a system without adequate buffering would have low throughput due to the time wasted in waiting for both agreement and for transmission. The bandwidth of the circuits might be effectively used by buffering. If communication is half-duplex, a large amount of buffering space is necessary to allow a smooth flow of traffic in both directions.

b. Error Control— The communication processor can detect transmission errors and inform them to the sender. Incorrect data is also detected and isolated. Furthermore, acknowledgement of message delivery or non-delivery might be useful.

c. Flow Control— Different source and destination data rates may necessitate implicit or explicit flow control rules to prevent the network from becoming congested when the destination is slower than the source.

d. Front-End-Processor— The communication processor is used to interface communication terminals or computers to a remote Host's data-processing system.

e. Gateway— The communication processor may be used as an interface between networks.

Network Topology

We placed a node computer (IMP)—a store-and-forward message switching computer having communication processor capability—at the center of the network, the network topology, in other words, being of the star type. This physical configuration of our network is reasonable in the following points: It is more efficient and convenient to connect computers *via* a communication network than to connect individual pairs combinationally. Because a node switching computer is very costly even in a localized network (*e.g.*, located within a single building as ours is), it requires large buffering space and versatile control functions to ensure high throughput, reliability and availability. Rather than provide facilities on a user-pair basis, we think it better to provide a large number of users with a single high-speed facility which can be shared in some fashion; this then allows us to take advantage of the powerful "large number law" which states that the demand at any instant, with very high probability, will be approximately equal to the average sum of demands of that population. In this way the facility capacity required to support the user traffic will be considerably less than in the case of unshared, dedicated facilities. A node computer, with a high fan-in of nearby Hosts and terminals, will present an increasing opportunity for shared use of centralized node's resources among many different devices. The most centralized form of network is the star net, such as KUIPNET. From this choice arose the problem of how many Hosts can be connected to a node —the number of fan-ins. This has affected our decision regarding Host-IMP connections, which is discussed later in this part.

Traffic Characteristics

The anticipated traffic to be handled by KUIPNET was of three kinds:

(1) The rate of messages which would pass through a modem would be rather low, being limited by the bandwidth of the communication line. Although traffic density might be low, however, the processor should be able to guarantee perfect action at any time for this traffic, because the communication processor would require real-time responses.

(2) Messages which would be transferred between local Hosts in real time would have to be guaranteed an effective, continuous transfer rate of more than 200 kbps (kilo-bits per second) on a communication path of the highest capacity. However, such traffic would only create a momentary flash in the network, and would be restricted to an exceptional environment.

(3) Messages consisting of character strings from keyboards would not need to be treated in real-time in the network. These messages would be given lower priority than those of (2), so they would not be allowed to utilize the full capacity of the communication facility during other real-time data transfer; but we must remark that the transfer of lower priority traffic would not be entirely prohibited even under this condition.

The traffic of these three types of messages should be controlled appropriately by the node processor's given capacity.

In order to transmit a speech signal through a store-and-forward network, it is necessary to convert it to an appropriate digital form. If high-fidelity reproduction of the speech waveform were required, and conventional analog-to-digital conversion techniques were used, the resultant PCM (pulse code modulation) representation of the signal would require the communication system to handle a data rate of about 250,000 bps [Forgie, 1975]. Speech transmission requires a constant data rate equal to the peak rate. The store-and-forward type network poses a problem for speech communication in that a delay is introduced into the speech data stream. While it is true that under overload conditions the average throughput of a network may fall below the required data rate for speech transmission, a straightforward way to minimize such delay is to give speech messages priority on the communication links. In order to handle this priority of data traffic, it is necessary to reserve the appropriate processing capability, buffer space, and channel capacity in the node.

The bandwidth of the network of KUIPNET depends mainly on three factors; the node switching computer (IMP), Host-to-IMP connection, and message size.

The Node Switching Computer

The architecture of the node switching computer is related to several network design parameters, as is explained here in brief. This problem was discussed in the report by Crowther *et al.* [1975], some of which is common to ours.

The speed of the processor is important in determining the throughput rates possible in the network. The store-and-forward processing bandwidth of the processor can be computed by counting the instructions in the inner loop. (See section 5.2)

The speed of the memory is a major determinant of the processor's speed, thus affecting the node bandwidth. The input/output transfers, acting in a "cycle-stealing" fashion, will slow down the processor. Our solution to this problem is that the maximum frequency of cycle-stealing should be restricted in order to keep the maximum processing time degraded by cycle-stealing within the allowable rate. (See part 2.2.2) This restriction is also necessary to prevent any continuous "memory busy" period which would arise when all input/output devices try to request a memory transfer at the same instance. There is another approach to this problem — memory will be used concurrently if it is effectively multi-ported. Some such high bandwidth communication processors have been reported, such as Pluribus machines [Heart, 1973; Mann, 1976].

The size of memory is another key parameter. The storage space in the node is mainly used for the program and associated data structures, the remainder being devoted to the buffering of messages. The need for a larger memory space arises from various factors. In the interests of improved fault diagnosis and increased maintainability, we would move more of the functions presently outside the control of the network operators to within their control. Larger data buffering and message queuing space becomes necessary when the system requires high throughput and short response time. (See section 5.2)

Real-time demands on the processor can be very heavy (parallel processing of asynchronous data streams). Fast response to interruptions is very important. In order to achieve this, the control structure of the switching program is arranged in the order of priority so as the rigid real-time events can be preemptively served. (See section 2.3.4)

The Host-to-IMP Connection

The design of the Host-to-IMP connections includes several problems specific to our in-house computer network, such as high bandwidth and maintenability in hardware devices.

Bandwidth:

The bandwidth of the Host connections are most likely the most important characteristics of our network. They define the traffic-carrying capacity of the network between any given source and destination, and are also an important factor in delay. To reduce delay in sending or receiving long messages, and also to allow high peak throughput rates, the Host connection bandwidth should be as high as possible (within the limits of cost-effectiveness)—even higher than the average Host throughput rate would indicate.

Full-duplex or half-duplex interface:

A node requires high performance from the input/output system, both in the number of connections and data rate.

The interface should be designed to allow messages to flow in both directions at once. The reason why the interface should be full-duplex is as follows: "...There are several situations in which an IMP may temporarily block the transmission of a message from the source Host to the source IMP. In general, any such blockage will last for only a few milliseconds, but in some cases the blockage may be indefinite. In at least one such case the IMP will be unable to accept the remainder of a message from its Host until it frees buffer space by delivering some messages to the Host (it is for this reason that half-duplex Host-IMP interfaces are prohibited)..." (BBN Report No.1822 p3-2) [BBN, 1974].

However, we were compelled to employ a half-duplex communication procedure between IMP and Host (see section 2.3.5) wherein messages could flow in either direction, because of two kinds of obvious hardware constraints: Firstly, we desired our IMP to connect as many sites as possible, but the input/output system allowed only 16 ports. For a full-duplex interface two ports are necessary, therefore the maximum number of fan-ins is 8. This was unsatisfactory for us. Secondly, the input/output system of the general-purpose computer system is too complicated to be tailored to our requirements. In our case, therefore, the interface hardware was prepared through some modifactions of an inherent peripheral controller for a half-duplex device.

Although we did not adopt full-duplex interfaces because of the above mentioned reasons, we did require a more efficient communication control procedure than that of conventional half-duplex interfaces. Usually for control of half-duplex communication, the "primary-secondary" procedure is adopted to determine the direction of data flow; that is, the primary station can start transmission any time but the secondary station is not allowed to transfer data unless the primary station accepts. The direction of data flow is alternated by exchanging control messages. A procedure of this type can be seen in Chapter IV. The efficiency of this procedure, however, is low because the communication bandwidth is greatly degraded by the extra control messages. In our network, efficient usage of communication line capacity is required for the real-time transfer of messages. Therefore, we adopted not the primary-secondary procedure but the "primary-primary" procedure; that is, wherein either side can start transmission whenever a message is ready.

We must be prepared to resolve conflicts which arise when more than one demand is simultaneously placed upon both ends of a half-duplex connection. There are two obvious but not final solutions to this problem; the first is to "throw out" or "lose" both demands; and the second is to accept the demand from the side having priority, losing the other.

In the first solution, the problem is solved only temporarily, because both ends will try to issue their demands again and may cause another conflict, though the possibility of such conflicts depends on the statistical characteristics of the traffic. A similar approach was taken in the ALOHA system, which uses packet switching with a broadcast radio channel where only one packet for transfer is accepted at a time [Abramson, 1973].

According to the second solution, either of the two ends takes the priority. Here, two alternatives are possible. We must consider another factor in making the final selection. In order to avoid any lockup in the IMP, which would be caused by the filling up of all buffers and which would block further input, Hosts should accept the demands from the input and receive an output from the IMP. (See section 2.3.5) Every Host should prepare at least two buffers — one for data in transmission, and one as a receive buffer.

Synchronous or asynchronous transmission:

When we use synchronous devices (*e.g.*, modems), a different aspect of the node computer is in question — its responsiveness. Synchronous devices require precisely timed service. If the node does not prepare for a new input within a given time, the next input arriving on that circuit will be lost. Likewise on output, the node must be responsive in order to keep the circuits fully loaded. This requirement suggests that some form of interruption system is necessary to keep the response latency low, *i.e.*, that the overhead of the operating system, task scheduler, and dispatcher may be prohibitive.

When we use asynchronous devices (*e.g.*, the handshake procedure in transmission), the responsiveness of the node computer need not be so strained. However, for high throughput the node must be responsive in order to keep the circuit fully loaded.

Asynchronous operation is preferable to permit the bit rate to adjust to the rate of the slower member of the pair, and it is free of critical timing situations, especially the problems associated with contention over memory among the various synchronous input/output devices, consequently limiting the transfer bit to a low rate. On the other hand, for asynchronous devices the maximum transfer bit rate can be set much higher, and therefore a higher throughput can be achieved. (See section 2.2.3)

Message Size

The basic elements of communication exchange to be taken into consideration in data communication network concepts are the data segment and the data packet. The segment, composed of a leader followed by text, is the element entered by a subscriber into the network for subsequent delivery to some other network subscriber. The packet, composed of a header followed by text, is the fundamental element handled and processed by the switching nodes of the network. Both of these elements are mutually conformed in configuration in the local network; *i.e.*, a leader is the same as a header and both texts are the same.

Message size needs to be large because the overhead on messages is significant if the node has to address many messages, and it may be inefficient for the Hosts to have many message interruptions. The upper limit on message size is what can conveniently be stored and assembled within given node storage and delays. We chose the maximum message size of 8095 bits.

A speedier Host connection allows the use of a longer message with less overhead and Host processing per bit, and therefore greater efficiency. The faster the Host connection, the longer the message should be, since long messages have less overhead and permit higher throughput, while the added delay due to the length is less important at high circuit rates.

2.2 Hardware Design of Communication Network

The hardware of the communication network consists of a node switching computer, lines, and the high-speed computer connection control equipment.

2.2.1 Hardware Configuration of Switching Computer

We adopted NEAC 3200/50 as a node switching computer (IMP). This computer is the same as Honeywell's DDP-516, which is employed as an IMP in the ARPANET. The high-speed data channel of this computer, called DMC (Direct Multiplex Control) channel, has a maximum transfer capability of 260K words/sec (1 word = 16 bits), summing up all the partitioned 16 channels. Each channel is given ordered priority, so each Host receives priority determined by the IMP channel.

The configuration of the IMP hardware is illustrated in Figure 2.1. The IMP accepts 8 Hosts, but at present is equipped with Host-interface cards for 5 Hosts. The full-duplex commu-

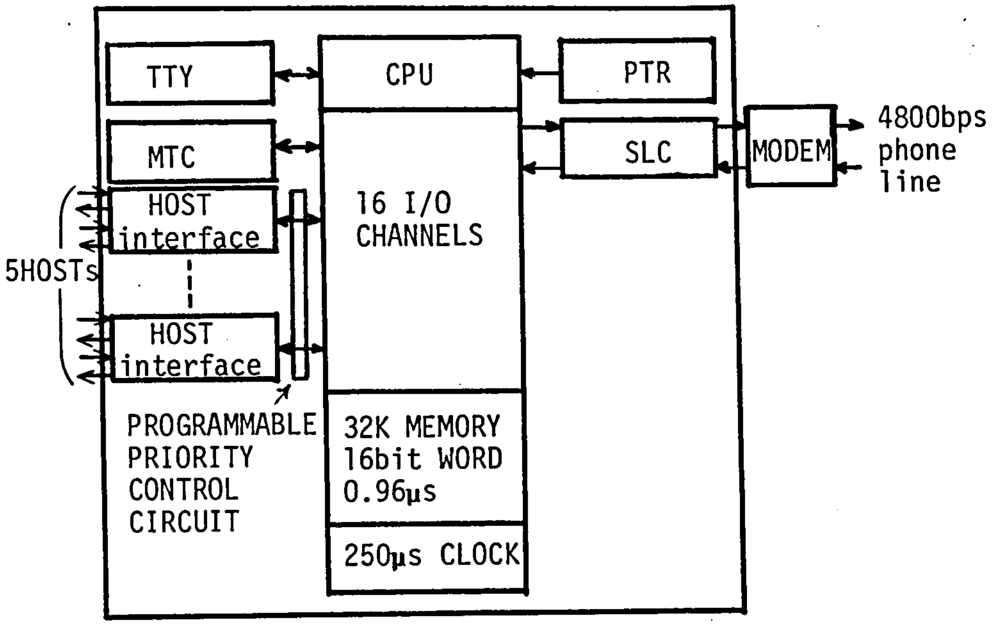


Figure 2.1 IMP hardware configuration.

nication lines between Host and IMP are composed of four coaxial cables, two cables for each direction. Both ends are connected directly with the cables, without any insertion of such equipment as modems or the like.*

The computer-connection-control equipment (CCCE) attached to the IMP is a standard interface for IMP-Host connection (see section 2.2.3), and is constructed from about 80 TTL 74-series IC's on a board of printed circuit. Each Host is different in its architecture, and its interfaces are designed to accomodate its electrical interface condition. But since the serial-parallel conversion part and the drivers and receivers are fixed, they are produced of printed circuit.

* An experimental device for DC isolation was tested and proved to have the same functions as designated.

The transfer method is asynchronous, bit-serial, DC mode. Every bit is converted into two kinds of pulse widths — 200 nano-seconds for "0" and 400 nano-seconds for "1", respectively. A stream of data bits is sent through a cable enclosed by two control signals; start of transmission (a pulse of 1 μsec width), end of transmission (a pulse of 2 μsec width). A "ready for next byte" signal (a pulse of 0.5 μsec width) is sent back to the sender through another cable for control when the receiver is ready for the next 9 bits (8 data bits and 1 parity bit). (See section 2.2.3)

The procedure stated above is a "hand-shake procedure". Data bits in a Host are sent from the most significant bit of a word in order of decreasing power and increasing address of memory. The bit transfer rate is 1.6 Mbps within a distance of 50m, 1.1 Mbps at a distance of 200 m. (See section 2.2.4) Cables are extensible to 400 m.

The IMP includes a 250 μsec clock and a 4,800 bps communication controller called a Single Line Controller (SLC). A teletype and a paper tape reader which are attached to the IMP are used for maintenance, debugging, and operation. A magnetic tape controller which drives a 1,200 foot, 800 bpi, 9-track tape is used for logging of measurement data. (See section 5.5)

2.2.2 Control Hardware for High-Speed Transmission

The control hardware of the IMP is developed so as to harmonize with the three kinds of traffic stated in section 2.1. In this section the control hardware particular to KUIPNET is described.

Control of channel rate: To achieve real-time data transfer (1.6 Mbps) between two Hosts connected to the same IMP, the high rate of access to the core storage should be authorized to the CCCE through the DMC channel. The rate is 100K words/sec when the data transfer rate is 1.6 Mbps, though the maximum rate of the DMC channel allows a rate of 260K words/sec. Since access to the memory is determined by the controller in the IMP in a way that a request from a channel has priority over a request from a CPU, the number of instructions executed per second may decrease extremely due to memory-cycle stealing by channel when there is a concurrence of data transfer between many Hosts. In order to allow the CPU a reasonable amount of time to carry out its processing function, access of the data-transfer channel to the memory is restricted to within 50% of the total available memory access slots by the hardware attached in front of the DMC channel. In this way, the ratio of channel utilization is limited to less than 50%. As shown in Figure 2.2, the length of time of one cycle of the core memory itself

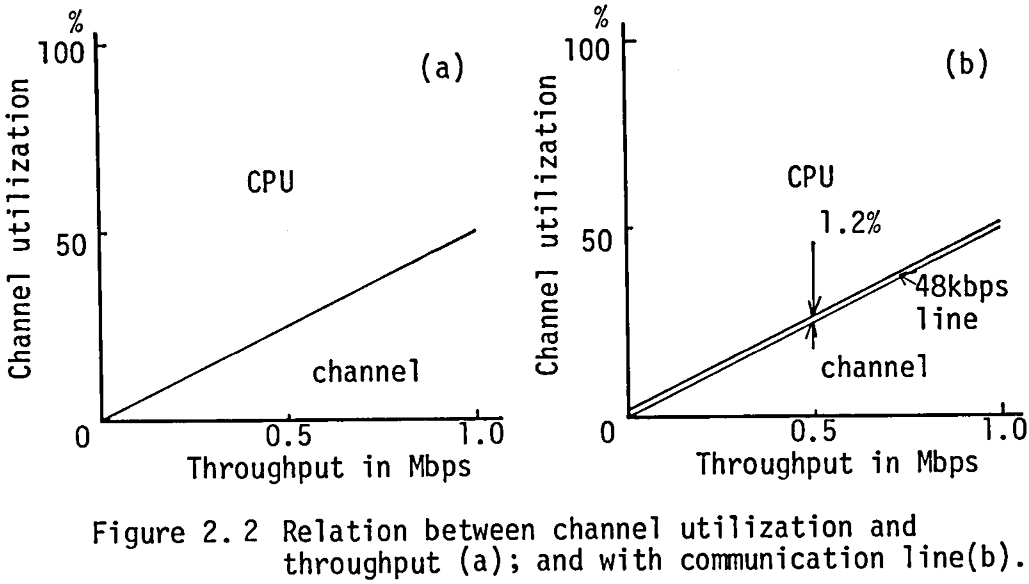


Figure 2. 2 Relation between channel utilization and throughput (a); and with communication line(b).

is 0.96 μsec, and the access interval of the CPU is not degraded by more than two of these cycles.

A communication line of 48 kbps uses only about 1.2% of the available memory time slots. Communication lines were employed in this in-house computer network to connect it with a nationally distributed computer network and a remote computer system. (See Chapter IV)

Programmable priority control: A special logic was designed and inserted between the IMP-Host communication controller and the DMC channel so as to be able to assign transfer capacity to a pair of Hosts which needs real-time high-speed large-volume data transfer. Each channel, by DMC mechanism, has its individual priority which is adopted in the transfer of every word. The order of priority goes from #1 to #16 downward in opposite order with the fixed order of the channel number. The programmable priority control circuit is a logic circuit which may change the order of priority of a channel any time by means of a message sent from a Host.

The fundamentals of the logic are shown in the following boolean formula. First, we shall describe here the fixed ordered priority method.

Consider the case of n pieces of service request lines. At a time point t, $u_1(t)$, $u_2(t)$, ..., $u_n(t)$ are binary variables that indicate a request on a channel by 1 and no request on a channel by 0. The value changes from 1 to 0 when a request on the channel is accepted. Values $v_1(t)$, $v_2(t)$, ..., $v_n(t)$ are output values of channels. The fixed ordered priority is given in the following expression:

$$\left.\begin{array}{l} v_1(t) = u_1(t) \\ v_i(t) = (\overline{\sum_{j=1}^{i-1} u_j(t)})u_i(t), \quad i = 2,\ldots, n. \end{array}\right\} \qquad (2.1)$$

Next, we shall describe the programmable priority control logic.

There are n pairs of service requests and accept lines. A binary value $x_i(t)$ at a time point t, $(i = 1,\ldots, n)$ is assigned to each request line. $y_i(t)$ is a value of output from the programmable priority control circuit. The complexity of the control circuit increases depending on depth of specification of priority. Shown here is an example of an implementation which has been adopted in our subnet that allows the first order and the second order to be designated to two channels arbitrarily. The order of designation is extensible in the general way up to the k-th order $(k \leq n)$. A set of outputs $y_i(t)$'s are fed to the fixed ordered priority logic ($i.e.$, $y_i(t) = u_i(t)$). In the following formula, $p_i^{(1)}(t)$ and $p_i^{(2)}(t)$ are outputs of the first and second priority designation flip-flops, respectively, and each is set by external command:

$$\left.\begin{array}{l} y_i(t) = \{p_i^{(1)}(t) + \overline{\sum_{j=i+1}^{n} x_j(t)\cdot(p_j^{(1)}(t) + p_j^{(2)}(t))}\}x_i(t) \\ \qquad\qquad\qquad\qquad\qquad\qquad (i = 1,\ldots, n-1) \\ \\ y_n(t) = x_n(t) \end{array}\right\} (2.2)$$

If it is desired that the first priority be designated, one of the $p_i^{(1)}(t)$'s ($i = 1,\ldots, n$) is set to 1 and all the others are set to 0. The second priority is exactly analogous. If all the $p_i^{(1)}(t)$'s and $p_j^{(2)}(t)$'s are set to 0, the programmable priority control circuit has no effect and so the order of the priority is determined by that of the fixed order circuit in the computer. If all $p_i^{(1)}(t) = 0$, and $p_i^{(2)}(t) = 1$, $(i = 1, \ldots, n)$, then the order of priority becomes the reverse of the fixed order of priority in the computer. Figure 2.3 is an example of this logic equipped on the IMP.

The measurement results shown in Chapter V, show that with the aid of the above-mentioned two control mechanisms, the message throughput of the highest priority path which is changeable by program is about 800 kbps even under a heavy load of traffic. If these control mechanisms were not employed:

- By memory-cycle stealing from channel, the processor would be completely stopped during heavy traffic causing a "memory busy" period.
- Due to the fixed ordered priority assignment, only one path (fixed and not program changeable) could achieve high throughput, while the transmission of other paths placed on lower priority levels would be interfered with.

2.2.3 Computer Connection Control Equipment

Here the hardware configuration and control mechanism regarding the computer connection control equipment which connects the IMP and Host is described. The description includes the standards of the Host interface design, the IMP interface design, and the IMP-Host data transmission method.

The computer connection control equipment is designed for on-line data transfer between computers generally of different

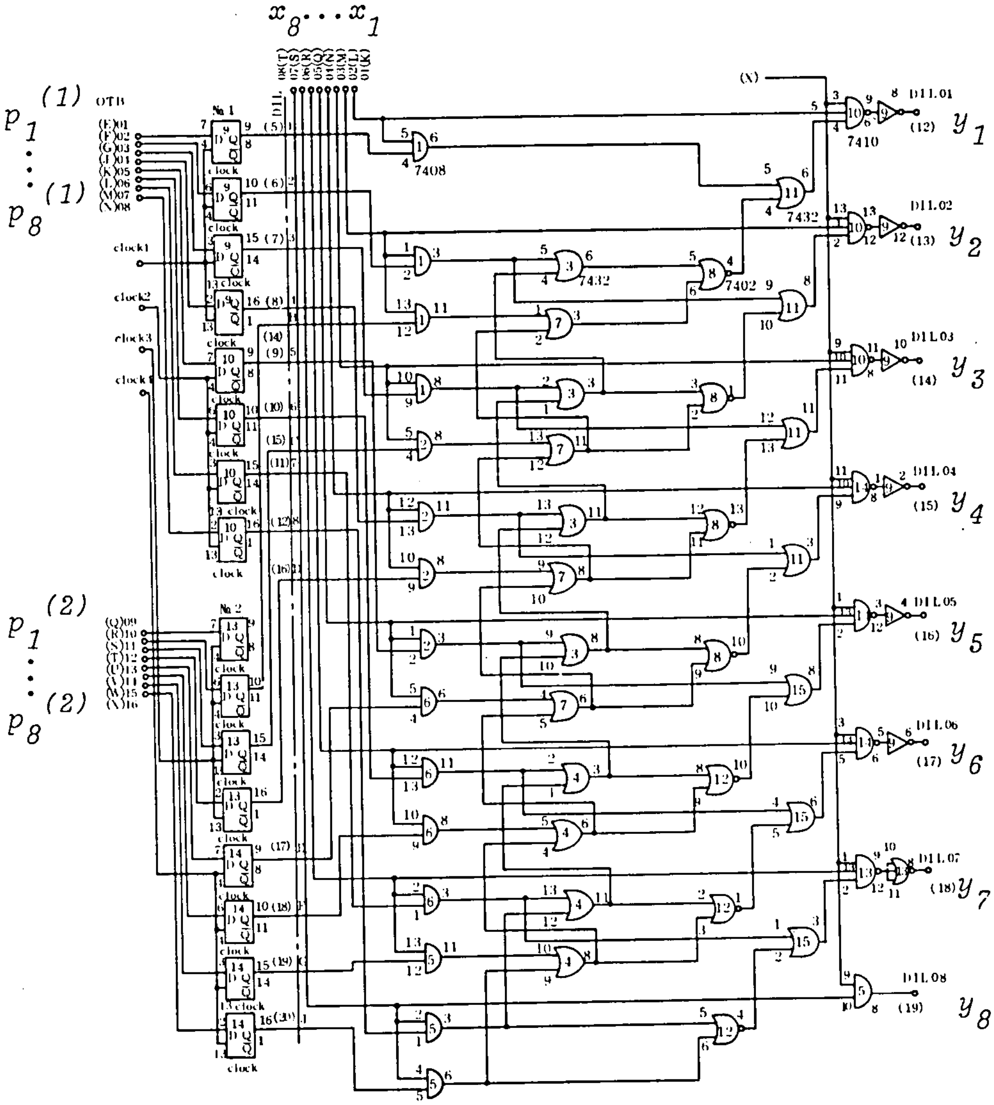


Figure 2.3 Programmable priority control circuit.

manufacturers. An outline of the computer connection control equipment (CCCE) is as follows:

(1) Bit-serial transmission— To adjust to the word length of individual computers, the data format dispatched by a sender on a line is in the form of a bit stream which is accepted by the receiver, splitting them into a sequence of words of the same length as itself.

(2) Hand-shaking— To adjust the data transfer rate of both parties (each of which has different channel rates), the transmission method is fundamentally asynchronous. Every eight bits are sent only after both the sender and receiver are ready for transmission.

(3) Half-duplex communication control procedure— IMP-Host communication control procedure is half-duplex, due to the restriction on the number of channel slots available.

(4) Interruption— To accomplish interleaving with the channel flow, the controller communicates with a computer *via* interruption line.

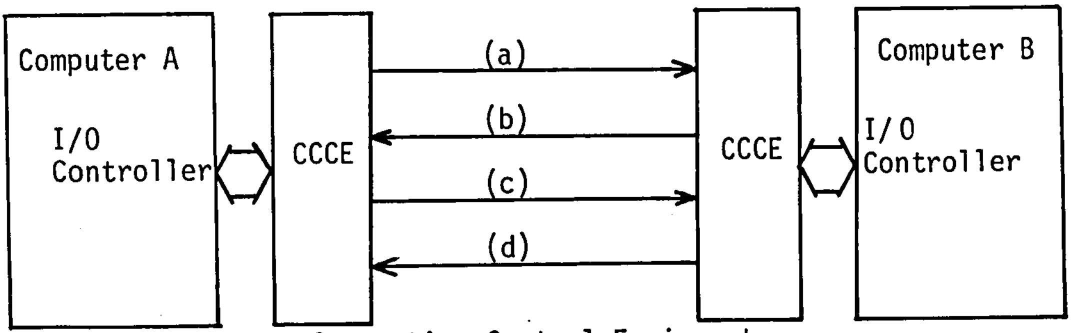
(5) Separation of transmitter part and receiver part— To make possible a self-standing local loop test, the interface hardware itself is devised to have the capability of a full-duplex operation. Adaptation of the transmitter and/or receiver to any Host is possible through an input/output controller, the special part of which is designed to be as small as possible compared to the common part and to adopt to the architecture of the computer.

(6) Pulse width coding— Data bits and control signals are coded in a pulse width of geometrical progression, of which one unit is 200 nano-seconds, which is transmitted along the cables connecting two computers. A unit of four coaxial cables makes up the communication medium between computers.

(7) Error detection— The error detection method consists of the addition of an odd parity bit to every eight bits of data sent by a sender, and inspection of this parity bit by the receiver.*

(8) Padding function— The padding, by hardware, of zeros onto the afterend of a data stream is carried out in order to make a multiple of a unit when the total amount of bits dispatched is not a multiple of 8 or when the total amount of bits received is not a multiple of the word length of the receiving computer.

The configuration of a computer connection is shown in Figure 2.4. Two computers are connected to each other *via* the CCCEs, each of which is attached to an input/output control unit.



CCCE: Computer Connection Control Equipment
(a) data and control signal sent from computer A to B
(b) Receive Ready signal sent from computer B
(c) Receive Ready signal sent from computer A
(d) data and control signal sent from computer B to A

Figure 2.4 Configuration of computer connection.

* From our experience, the in-house direct connections are highly reliable if they are well designed and adjusted.

Two CCCEs are connected *via* communication cables and a party communicates data by exchanging control signals. The CCCEs communicate data with a main memory through their input/output control units. The data in the memories of two computers which are connected by CCCEs are communicable.

The organization of a CCCE is shown in Figure 2.5. The functions of the blocks in the figure are as follows:

Output buffer— In order to speed up the transmission rate, preemption of the next data while the current data is still in the process of being transmitted is accomplished by storing the output data dispatched from a computer in this buffer.

Parallel - serial conversion block— Parallel data fed into the output buffer from a computer is passed through the shift register in this block, converted here into a sequence of bits, and dispatched.

Coding block— In this block, the sequence of data bits converted in the parallel - serial conversion block and control signals for data transfer are converted into a sequence of pulse widths, and then sent to a cable. (encoder in Figure 2.5)

Decorder block— Data and control signals that have been coded according to pulse width by the sender are decoded in this block by testing the pulse width.

Serial - parallel conversion block— Sequences of bits received are put into a shift register until the register is filled to the length of a word. Then the parallel bits are moved to the input buffer.

Input buffer— Any data which is to be fed into a computer is stored once in this buffer. In this way, the receiver can receive the next byte before the previous byte has been fed into the computer.

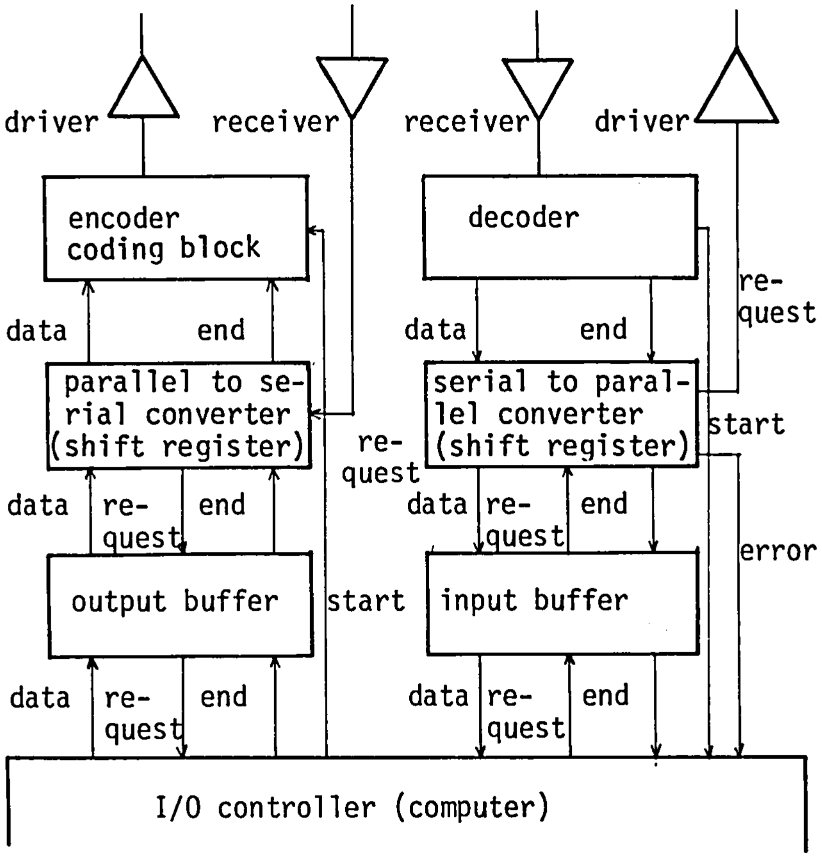


Figure 2.5 Organization of computer connection control equipment.

Below is a brief description of the behavior of a CCCE.

Transmitter mechanism— The output data stored in the output buffer waits until all current bits are sent out from the shift register to the coding block and the shift register becomes empty, when the parallel÷serial conversion block requests the output buffer to move the next data into the shift register. At this point, a request for further data is then made by the output buffer block to the computer. During this time, the number of bits moved through the shift register is counted by a shifting pulse. When all bits in the shift register are moved out and the shift register becomes empty, as before, a request for the next data is again made to the output buffer.

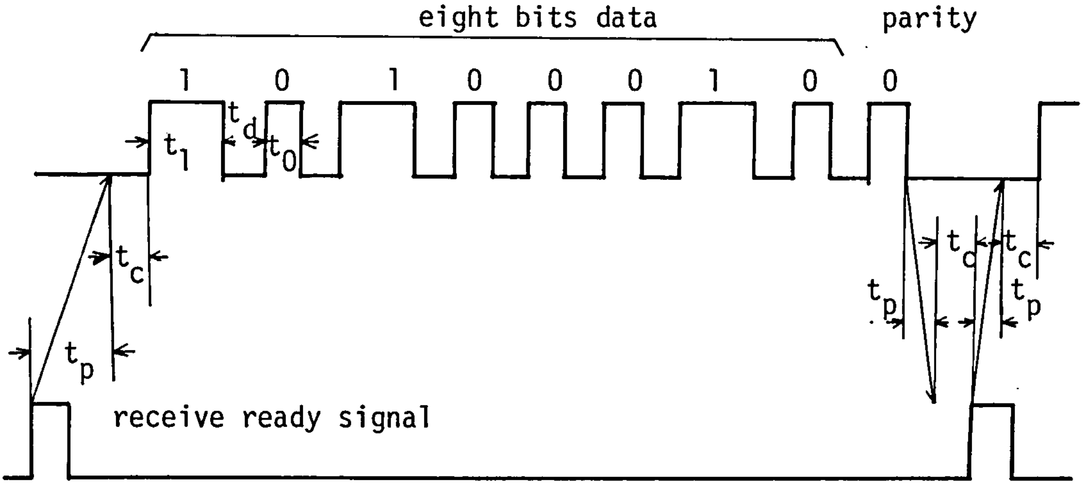
From the bits conveyed from the shift register, a sequence of pulses is generated of which width corresponds to the value of a *0/1* bit, and is transferred from the driver circuit. Zero is a pulse of 200 nano-seconds and one is 400 nano-seconds. A sequence of pulses contains 200-nano-second zero-level intervals between every two bits. One unit of transmission consists of eight bits plus an odd parity bit, spaced by the above-mentioned zero-level intervals. After sending one such unit, or in other words 9 bits, the sender waits for a response from the receiver which signals that the receiver is ready for the next unit, when the next 8 bits plus parity bit are sent. A sender can issue a "send" request signal to a receiver. This signal—a pulse 1-μsec wide—is touched off through an instruction by the computer.

The end of transmission is indicated from a computer. After the output buffer and the shift register have become empty and a "ready" signal from the receiver is received, and "end of transmission" signal of 2-μsec pulse is sent from the sender. A padding sequence of *100...0* is added at the end of the last output data, calculated so that the final unit is 8 bits in length—a complete unit.

Receiver mechanism— An input is composed of a sequence of pulses of various widths which determine their digital values —*0* or *1*. The decorder circuit descriminates these values, and also determines the inherent timing sequence. A sequence of such timing signal bits accompanied with digital value bits is fed into the shift register of the serial-parallel conversion circuit, where they are accumulated by a bit counter so that they match the length of a unit of the computer —— (*e.g.*, 16 bits). When the counter reaches the length of a word, this accumulated data is moved to the input buffer in which block a request is made to the computer to read the content. The serial-parallel conversion block has another counter which matches the input unit length (in our case, one byte = 9 bits). It checks the parity bit at the end of the byte and detects transmission error. Thus a unit of transmission is completed. The serial-parallel conversion block issues a request, when the shift register moves its content to the input buffer and the receiver is able to accept the next byte (in our case, 1 byte = 8 bits + 1 parity bit). The receiver side issues a "ready for next byte" signal and clears the bit counter which counts 9. Receipt of the "end of transmission" pulse causes an addition of a padding sequence to complete the length of the final data unit —*i.e.*, the length of a word—in order to be fed to the computer as the last data.

2.2.4 Estimation of Transmission Rate of Computer Connection Control Equipment

Described here is an estimation of the transfer capability of the CCCE. Figure 2.6 shows a time chart of digital data on a line where t_1 is the width of a pulse of *"1"* (typically 400 nano-seconds), t_0 is the width of a pulse of *"0"* (200 nano-seconds), t_d is the width of the interval between two successive pulses (200 nano-seconds), t_p is the delay in propagation through a cable (800 nano-seconds/200 m), t_c is the delay in propagation through a circuit, n_0 is the number of *"0"*s in a transfer block (9 bits), and n_1 is the number of *"1"*s in a transfer block. The transfer bit rate P (bps) is given by the following expression:



t_1: pulse width of logic 1
t_0: pulse width of logic 0
t_d: zero level interval
t_p: propagation delay
t_c: circuit delay

Figure 2.6 Time chart of line signal.

$$P = 8/(n_0\ t_0 + n_1\ t_1 + 8t_d + 2t_p + 2t_c\ ),\ n_0 + n_1 = 9 \qquad (2.3)$$

Delay in cables is doubled because a sender must wait for a response from the receiver, making the propagating path a two way. A delay in a circuit is made by both the sender and receiver.

The average bit transmission rate is P = 1.31 Mbps under the condition $n_0 = n_1 = 4.5$, t_p = 800 nano-seconds, t_c = 100 nano-seconds. The transmission rate may change depending on the ratio of *0* and *1* in the data, since the coding principle is based on pulse width.The fastestset of bit patterns is the case where n_0 = 8 and n_1 = 1; then the maximum transmission rate is P = 1.60 Mbps. The worst case is n_0 = 0, n_1 = 9; then the maximum transmission rate is P = 1.14 Mbps.

From the above discussion follows the conclusion that a maximum transmission rate of at least 1.0 Mbps is assured even under the worst conditions between computers at a distance of 200 m. The transfer rate of computer channels is high enough that it does not place any limitation on the CCCE's transfer rate. For example, the selector channel of NEAC 2200 is 1 Mbps.

2.3 Software Design of Communication Network

There are two usual techniques: message-switching and packet-switching. In KUIPNET, the message-switching system was employed instead of the packet-switching system in which a message is split into many packets. This is because the Hosts in KUIPNET are adjacent to the switching computer so that a message need not pass multi-nodes, and the given path allows high-speed transmission.

2.3.1 Message Switching Program

The main role of the IMP operational program is message processing which includes processing a leader, dispatching a message to the destination, and generating a RFNM (Ready For Next Message). (See section 3.2.3) As we have stated before, none of the following IMP functions are provided in KUIPNET: packeting, which splits a message from a Host into packets attached with a packet header; exchanging routing information; re-assembling a message from received packets before dispatching it to a Host; or send back "acknowledge" packets to confirm receipt of the correctly received packets. However, this IMP program does gather statistical data on traffic and investigates the IMP's status such as any IMP-Host interface trouble or buffer lockup. (See section 5.5)

The message-switching program consists of several tasks which are activated by external interruptions (Figure 2.7). When the transmission of a message has finished, the HOST-IN routine is touched off by an interruption from a CCCE on the receiver side, and the message received is placed on the tail end of the HOST-TASK queue. The TASK routine takes a message on the HOST-TASK queue, in turn from the top, and places it on the HOST-OUT queue according to the destination Host's name denoted in the initial leader. In turn, the HOST-OUT routine dispatches the messages on the HOST-OUT queue to the destination Host. The buffers from which the messages were sent are then returned to

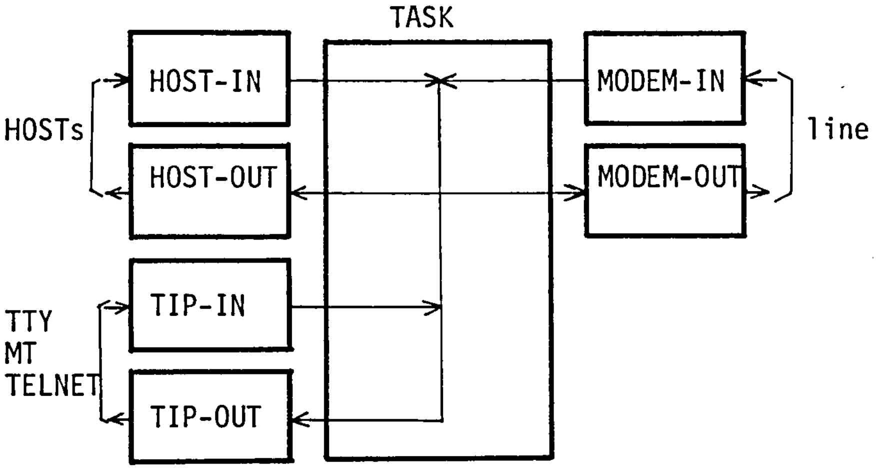


Figure 2.7 Message flow in IMP.

the "free buffer" list for further use as new Host input buffers. Since CCCEs are similar to each other, the program that controls communication with Hosts can be used as a common routine by any Host.

Besides the IMP functions, this program provides extensible capabilities for the optional use of TIP (Terminal IMP) (See section 3.4); that is, it is capable of such special communication control as terminal accommodation in the network and access to and from the VDH (Very Distant Host) (See Chapter IV).

The whole program consists of 21 functionally divided routines, each of which occupies one or two sectors of the main storage (1 sector is 512 words, and an IMP's word is 16 bits). These routines communicate with each other *via* the common data resident in the storage. The storage for the switching program is about 4K words, and 6K words for buffers. (See the next section)

2.3.2 Core Map of Switching Computer

NEAC 3200/50 is equipped with full 32K words storage, half of which is used to store the codes related to the message-switching IMP functions and the store-and-forward buffers. The rest of the storage is used for TIP and VDH functions. Each individual routine, which contains machine codes and address pointers to machine codes in other sectors, is clustered with care in a single sector. The core map of IMP NEAC 3200/50 in Figure 2.8 shows its current status of utilization.

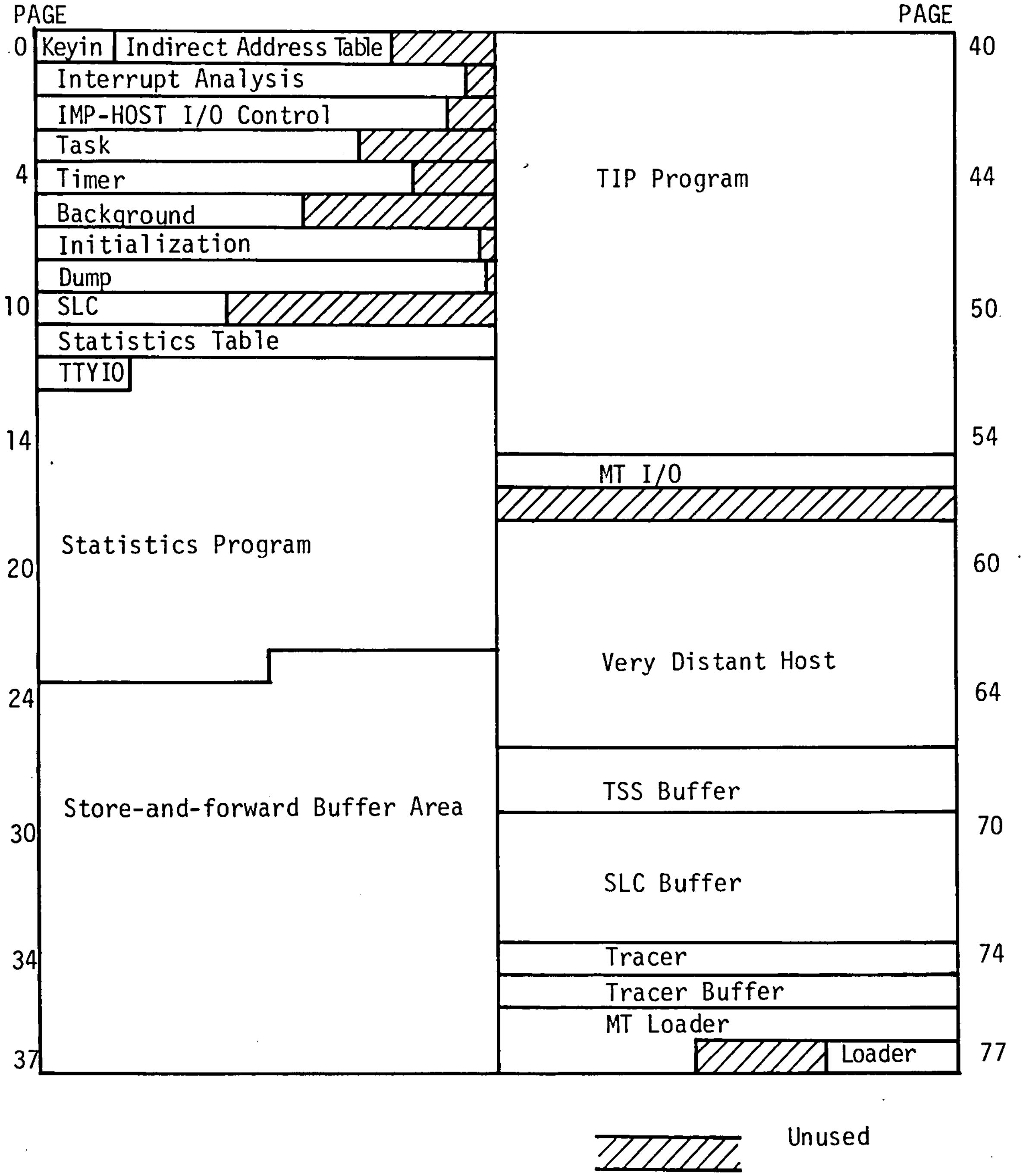


Figure 2.8 IMP core map.

2.3.3 Routines of Message Switching Program

The IMP operational program is composed of several functionally partitioned modular routines.

The Interruption Routine analyzes the cause of interruptions by external events in turn; memory parity error, interval timer, communication controller, Host interfaces, task interruption, teletypewriter, magnetic tape drive unit, and console.* After the analysis of the source of the interruption according to the above-given sequence of priority, the previous status of the processor is saved; the contents of the A, B, P, K, and X registers, four software registers, and the program mode indicator.** Then this routine passes the control onto the particular routine. At the completion of the process which was triggered by the analyzed interruption, the exit routine is called back to its pre-interrupted state.**

There is another kind of interruption which arises from the violation of "protective" restrictions toward the execution of special instructions—instructions to write into a protected sector, input/output instructions, or priviledged instructions which may change the CPU mode.

The IMP-Host Input/Output Control Routine handles IMP-Host half-duplex interfacing. (See section 2.3.5) This routine is subdivided into five sub-routines which each supports a different stage of the communication control procedure. (1) Host-Out Start sends a "start" signal to a Host to request the initiation of the Host's receiving channel, and waits for a response from

* This process requires an average of 13 main memory cycles.

** This process requires 57 main memory cycles for the enter and 63 for the exit operation, which amounts to about 30 to 70% overheads.

same Host. The (2) Mode Decision Routine receives an interruption from a Host when the Host accepts the request sent from the IMP or the Host requests the IMP to receive a message. The (3) Out-End Interruption Routine is touched off by the interruption from a channel when it has finished transmission of a message block up to the final address. The IMP dispatches an "end" pulse which indicates the end of data, then goes to the next message output if any. The (4) Start Interruption Routine is activated through an interruption by a Host's "start" signal that begins the sending of a message. Upon receipt of this signal, the IMP is expected to prepare a "receive" buffer and to return a "ready" pulse to let the Host know that the IMP has accepted the previous request and is ready to receive a message. If the IMP can not obtain a receive buffer from the free list, the response will be postponed for 30 seconds to wait for a free buffer. If no free buffer is available after 30 seconds, then the request is discarded and a control message is sent to the Host to the effect that the request has been discarded because of lack of a free buffer. The (5) Input-End Interruption Routine is activated by the "end" pulse which is sent from a Host at the end of transmission of a message. It senses the status of the interface, reads the address counter to determine the message length, places the message on the HOST-TASK queue, sets the programmable task interruption, and gets a receiving buffer from the free list to prepare for any further request from the Host.

The Task Routine, activated by the programmable task interruption, handles messages on the HOST-TASK queue according to their destination on the HOST-OUT queue and triggers the Host-Out Start routine.

The Timer Routine is composed of a fast timeout routine, a middle timeout routine, and a slow timeout routine. The fast timeout routine is started every 25 ms. Its purpose is to find out if a Host receive buffer has been allocated, and if not, to allocate one from the free buffer list. It also watches the IMP utilization. (See section 5.5) The middle timeout routine is started every 100 ms and cares for transmission control. The slow timeout routine is started every 0.5 second and investigates Host-channel lockup. If locked for more than 30 seconds, it initiates the channel. The slow timeout routine also executes a pseudo-timeout routine (see below the Pseudo-Timeout Routine) which is used by various routines which need timeout functions, such as SLC and TIP. Furthermore, this routine executes statistics (see below the Statistics Routine).

The Background Routine is an idle loop which senses several statuses and is interrupted by higher level routines. It monitors the pseudo-timeout routine, the statistics routine, the Server TELNET, and the TIP routine.

The Statistics Routine is triggered by the timer routine at every statistics interval and observes several parameters related to the network measurement system. (See section 5.5)

The Initializing Routine brings all the flags, tables and queues back to their initial state, and is the first routine in the IMP program.

The Console Routine handles console interruption and prints out the statuses of the IMP program.

The Pseudo-Timeout Routine handles interval timeout requests from the TELNET, TIP, statistics and SLC routines.

Although the principle of the organization of KUIPNET's IMP program is quite different from the usual organization of general-purpose computers, similar system calls are established in both that are procedures commonly and exclusively used by several routines related to the manipulation of message queues and message buffers, and to the manipulation of program priority level. These procedures are also used to implement TIP and the

VDH around the IMP program. (See section 3.4 and Chapter IV)

Queues and lists are used for allocating buffer area, for keeping the order of incoming or outgoing messages, and for scheduling interval timer requests.

2.3.4 Transition Control between Routines

The IMP program is divided into many sub-routines (see previous section) which are entered by sub-routine calls or hardware interruptions. The method of division is based on priority levels, introduced to improve the responsiveness and efficiency of the processor resources. Each routine is given a different priority level and is executed as a task. Task switching takes place only from a lower level task to one of a higher level. A lower priority task can always be interrupted by higher priority tasks. The control is returned to the lower priority task only after completion of the higher level task. There is no supervising monitor. Individual tasks inherently know when they can go.

All tasks enter and leave in the same manner as the interruption routines do. They may be triggered not only by true interruptions but also by artificial interruptions, wherein a task triggers another higher task by artificial interruption and a lower task by programmable interruption which causes hardware interruption. These functions provide lower level processes, such as TIP (see section 3.4), a means to communicate with the IMP. Table 2.1 shows a list of tasks and the ways to activate them.

It is easy to realize mutual exclusion, since there is only one task per level. Critical sections or indivisible operations are entered after an INH (inhibit interruption) instruction and

Table 2.1 Task activation methods in IMP program.

Task Name	Symbol	Activation Method	Task Interrupt
Memory Parity Error	MPE	II	
SLC Control	SLC	EI, AI	*
Host Interface Control	HOST	EI, AI	*
Timer 1	TM1	EI	*
Timer 2	TM2	AI	
Timer 3	TM3	AI	*
Task	TSK	PI, AI	
Restricted Mode Monitor	RMV	II	
TTY	TTY	EI	
Traffic Statistics	TRF	SC	
TIP	TIP	SC	
Back Ground	BG	-	
Memory Dump	MDP	CI	

II: Internal Interrupt
EI: External Interrupt
AI: Artificial Interrupt
PI: Programmable Interrupt
SC: Subroutine Call
CI: Console Interrupt
-: always active

leave through an ENB (enable interruption) instruction.

Table 2.2 shows possible transitions of tasks and their causes.

2.3.5 Communication Control Procedure

As mentioned in section 2.1, "The Host-to-IMP Connection",

Table 2.2 Task transition and its causes.

to	MPE	SLC	HOST	TM1	TM2	TM3	TSK	RMV	TTY	TRF	TIP	BG	MDP
from													
MPE													SC
SLC	II						PI						CI
HOST	II	EI					PI						CI
TM1	II	EI	EI,AI		AI		PI						CI
TM2	II	EI	EI	EI		AI	PI						CI
TM3	II	EI,AI	EI,AI	EI			PI						CI
TSK	II	EI,AI	EI,AI	EI									CI
RMV	II	EI	EI	EI			EI		AI				CI
TTY	II	EI	EI	EI			EI						CI
TRF	II	EI	EI	EI			EI	II					CI
TIP	II	EI	EI	EI			EI,AI						CI
BG	II	EI	EI	EI			EI		EI	SC	SC		CI
MDP													

symbols are same as in Table 2.1

the communication control procedure is based on a half-duplex mode between IMP and Host. Although the CCCE is capable of full-duplex operation, the communication path to a Host is limited to one channel. In a half-duplex operation, the direction of data transfer is decided by mutual interruption from both ends. The IMP-Host communication control is represented by the state-transition matrix in Table 2.3.

A sender CCCE issues a "start" pulse which requests the start of transfer of a block. The receipt of such a start pulse causes a "start interruption" to the CPU of the receiver side. If a receive buffer is available, the receiver party issues a positive acknowledgement pulse to the CCCE of the sender side

Table 2.3 IMP-Host communication control state transition matrix.

	State Event	0	1	2	3	4	5	10	12	14	23
		Neutral	Read End Wait	Ready Wait	Buffer Request	Write End Wait	Overrun	Receive Buffer Empty	Ready Wait	Write End Wait	Receive Buffer Wait
a	Receive Start	1	3	1	\	3	3	23	23	23	\
b	Receive Ready			4					14		
c	Receive Read End	\	3	\	\	\	3	\	\	\	\
d	Receive Write End		5			0	5			10	
e	Send Start	2						12			
f	Timeout		0	0		0	0		10	10	
g	Get Receive Buffer				0			0			1
h	Receive Buffer Empty				10						

blank: program error
\ : default
explanation of states and events is shown in the next page.

through a peripheral control instruction. On the other hand, if a receive buffer is not available in an IMP, the IMP-Host control message informs the sender party's Host of that fact, *i.e.*, that a receiver buffer is not available.

A few additional comments would be appropriate here. In a particular case where both the IMP and a Host have simultaneously issued a pulse to request transfer, in order to avoid heavy congestion in the IMP, it receives priority over the Host for the transfer. (See section 2.1, part "Full-duplex or half-duplex interface")

states and events in Table 2.3 have following meaning:

Read End Wait: waiting for the "end" pulse which is received at the end of input.

Ready Wait: waiting for the "ready" pulse acknowledging the start of message transmission.

Buffer Request: searching in the free list to get a free buffer.

Write End Wait: waiting for the end of output.

Overrun: receiving a message longer than the buffer length.

Receive Buffer Empty: receive buffer is not available.

Receive Buffer Wait: waiting for allocation of a receive buffer.

Receive Start: receipt of a "start" pulse.

Receive Ready: receipt of a "ready" pulse.

Receive Read End: receipt of an "end" pulse.

Receive Write End: receipt of an interruption from a channel at the end of transmission.

Send Start: sending of a "start" pulse.

Time out: timeout occurrence.

Get Receive Buffer: a receive buffer is obtained.

Receive Buffer Empty: no receive buffer is available.

CHAPTER III: DESIGN OF COMMUNICATION FACILITIES FOR PROCESSES

The communication facilities used for supervising the processes distributed among the individual Hosts are constructed upon the inter-computer communication network. In this chapter the concept of inter-process communication, the facilities for inter-process communication implemented on the Hosts, and some of their applications are explained.

3.1. Concepts of Inter-Process Communication

The computer system may be seen as having two facets: The monitor and the processes. The monitor performs such functions as switching control from one process to another, fielding interruptions, creating processes, caring for sleeping processes, and providing the processes with a set of machine-extending operations (supervisor calls). The process facet performs the normal user functions (user processes).

A co-operative job may be performed by more than two processes. In such a case, mutual communication is necessary to control the progression of the two individual processes. For example, a co-operation of two processes may proceed as shown in Figure 3.1. Suppose that process A concerns computing functions and process

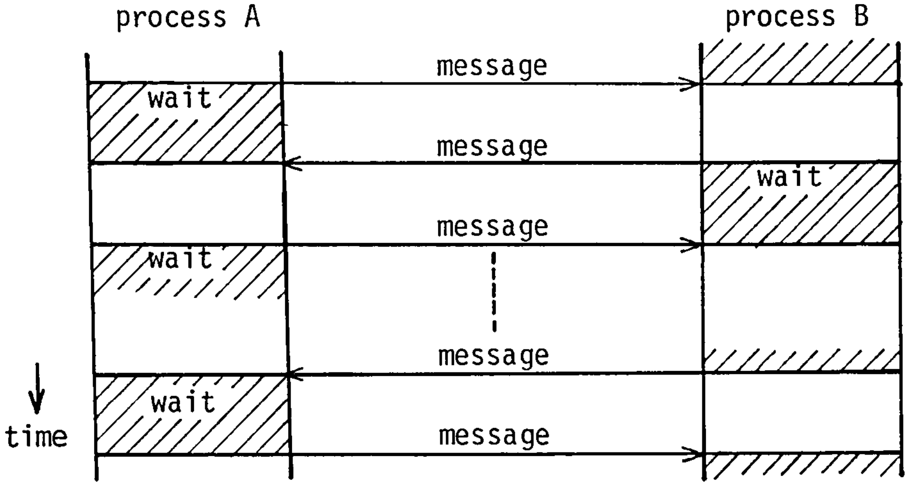


Figure 3.1 Inter-process communication.

B is a printer process. Process A sends a message to process B which is waiting for a message. Then process A waits for a message from process B which is now controlling a printer. Upon completion of movement, process B will then send a message to process A to inform A of the completion. Process A and process B are in different and independent computers, so some parts of their computation may proceed parallel.

Communication between distributed processes must be controlled by individual monitors, each in a different computer. Control is distributed in this way. The monitors then simplify the interface condition between the processes.

Usually, inter-process communication functions are provided by the monitor:

— RECEIVE— This operation allows a specified process to send a message to another process which is executing RECEIVE. The receiving process is informed when the transmission has completed.

— SEND— This operation sends a message from the process executing the SEND to a specified receiving process. The sending process is informed when the transmission has completed.

The fundamental functions necessary in process communication are: (1) identification of individual messages, (2) synchronization, and (3) format [Metcalfe, 1973]. For a sender, the first function is to recognize an individual message dispatched from one process and to identify its destination (where to go). For a receiver it is to identify the expected message (which one is desired to be received). In other words, this function is generalized to match the sender request and the receiver request somewhere in the network at the rendezvous site [Walden, 1972]. The second function is to know when a sender can send a message or when the most preferrable time is; that is, exactly when the receiver would be ready to accept the message correctly. For a receiver it is necessary to know when a message will arrive. Each participating process must know the status of the other. The third function is a set of agreements among sender and receiver processes on the interpretation of data in a message (*e.g.*, a unit of data or a set of characters).

One approach to accomplishing the identification function between processes in different computers is "conceptual line-switching"—"thin-wire connection" as called by Metcalfe[1973]. As shown in Figure 3.2 the two co-operating processes communicate with each other through two paths *via* a sending port and a receiving port for each path. The data-receiving port to a process is identified by the name "receiver socket", and the data-sending port from a process is identified as "sender socket". A combination of a sender socket and a receiver socket is uniquely identified in the network, this relation being called a

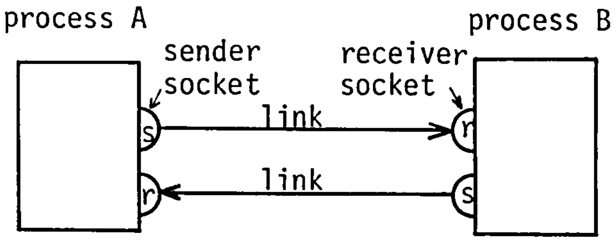


Figure 3.2 Concept of process connection.

"connection". When either one of the two processes wishes to communicate with the other, it identifies the partner process on the other monitor and they both set up the connection. This is very similar to the line-switching of the telephone system. After the establishment, the conncetion is identified as a "link". Generally, a pair of connections (or links) is necessary for bi-directional inter-process communication, since data flow on a single link is uni-directional. In cases where data flow is voluminous, a connection is a very effective way of utilizing resources, because the setup costs are amortized over the large number of streamlined transactions. However, if the traffic among processes is predominantly light and burstly, the relatively high setup cost of such connections will dominate, and efficiency will be low.

On the user process level the synchronization mechanism is implicit, because a process must wait for another process's reaction which may happen immediately, or may not happen at all. A SEND operation would not be allowed until the receiver process executes a RECEIVE operation. A RECEIVE operation will not be complete until a message arrives from the SEND operation designating the process. A synchronization mechanism of this type will work well if both of the communicating processes are sound and the communication path is reliable, but may cause some trouble if the processes are ill-matched or a message is lost on the communication path. The latter case is more common. Consequently another type of synchronization mechanism such as the following, which is more explicit, is necessary. If a process wishes to communicate directly with the partner process whatever the status of the partner process, the partner process is forced to go into the predefined state. This type of operation is similar to the "INTERRUPT" or "BREAK" in conventional systems.

On the monitor level, however, inter-monitor communication is asynchronous. The integral part of the monitor is the Network Control Program (NCP) which provides inter-monitor communication functions. The NCP may receive and send messages any time. Two NCPs, of which processes are in communication, exchange control commands and notify their statuses to each other. For instance, a connection request may be sent any time from a primary NCP to a secondary NCP when a primary process issues a network command to establish a connection with a secondary process. The request will be fulfilled if the secondary NCP has a corresponding outstanding connection request; if not it will be refused. In this way, an NCP can know the status of the other NCP any time, asynchronously. Any way, the primary process is informed about the result of the previous command.

The communication system that provides the inter-process communication functions is organized in a layered fashion, as shown in Figure 3.3.

The first level is the packet-switching communication network—similar to the subnet of ARPANET.

The second level is the interface between the autonomous Host computers and switching node computer IMP (IMP-Host

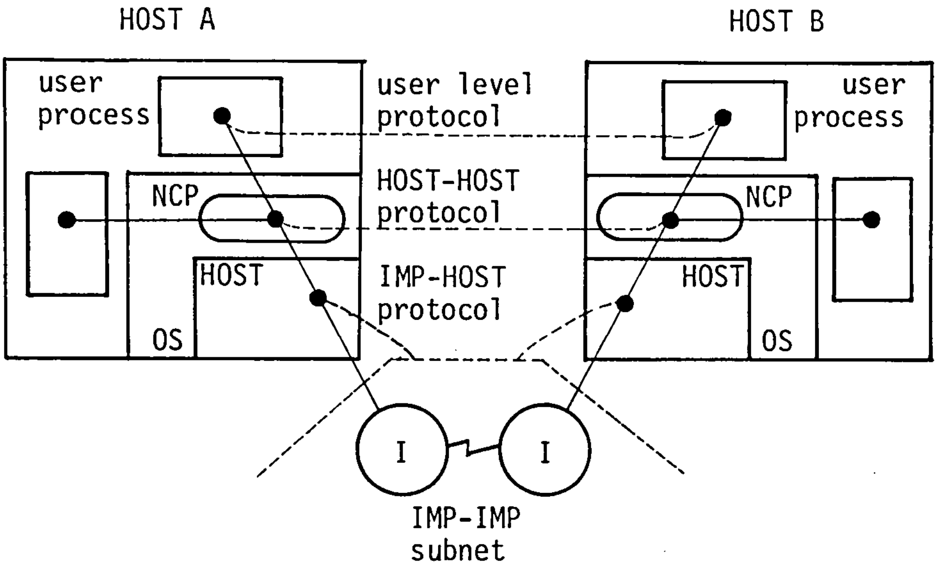


Figure 3.3 Hierarchical protocols for inter-process communication.

protocol). Isolation between Hosts is accomplished on this level because a disconnection of a Host does not affect the functions of the IMP. Even heterogeneous computer systems use the standardized interface and communication procedure on this level, so that from the IMP's view all Hosts are identical.

The third level is the inter-monitor communication wherein the NCP's control-exchange commands control inter-process communication (Host-Host protocol) [Carr, 1970].

The fourth level is the inter-process communication. Processes communicate with each other utilizing the communication facilities provided by the NCP (user level protocol).

3.2 Network Control Program (NCP)

3.2.1 Inter-NCP Communication Functions

An NCP is a unique process, often in the form of a system process, existing singularly in a Host and representing the Host's operating system. Each Host's NCP can communicate with any other Host's at any time, since an NCP is an autonomous process. There are two types of transactions between NCPs; control commands which inform the other of the status of the NCP and relate requests to it, and regular messages which are data sent or received by user processes. The functions of the NCP are to establish connections, terminate connections, control flow, transmit interrupts, and respond to test inquiries. Table 3.1 offers a summary of the control commands—the definitions of which follow that of the ARPANET (NIC #8246) [McKenzie, 1972]. However, these definitions of NCP functions are only conceptual; from our experiences, we offer the following practical interpretations:

"Host-Host protocol" is a formal description of the interface condition between Hosts on the software level. It is performed by the NCP usually as a part of the operating system. Although the functions defined in the Host-Host protocol should all be implemented, only the NCP functions which are absolutely necessary for process communication or implementation of user-level protocols are actually implemented. The selection of the NCP functions depends on the degree of implementation of the protocol.

Control commands RFC (Request For Connection)—RTS and STR—

Table 3.1 NCP control commands.

command	operation	function
NOP	no operation	no operation
RTS	receiver to sender	connection establishment
STR	sender to receiver	
CLS	close	connection termination
ALL	allocate	flow control
GVB	give back	
RET	return	
INR	interrupt by receiver	interruption
INS	interrupt by sender	
ECO	echo	test inquiry
ERP	echo reply	
RST	reset	initialization
RRP	reset reply	

and CLS (close) are indispensable for the establishment of a connection between two processes and must be implemented in all NCPs.

NOP (no-operation) is necessary for the NCP function test and for IMP-Host connection establishment.

The message flow control mechanism is defined in the Host-Host protocol of the ARPANET (NIC #8246). The control command used for this is ALL (allocate). Real-time digital speech transmission protocol implicitly employs this allocation mechanism to improve the efficiency of the communication path (See 3.2.3). In more complicated timing situations such as the TELNET or the File Transfer where the control of inter-process communication moves frequently between the two processes concerned, the ALL mechanism is necessary to certify that the receiver process has issued a Receive command from the arrival of an ALL before dispatching a message. Many NCPs which handle a number of co-existing connections do not allocate a receive buffer to an individual connection beforehand in order to save the would-be-used amount of storage. If a receive buffer is not available for an arriving message, the message will normally be discarded. These NCPs, therefore, must inform the sender process of the allocation of a receive buffer. The allocation mechanism implemented in our NCPs works well in these cases.

GVB (give back) and RET (return) are also parts of the allocation mechanism, but are not employed in KUIPNET because they are too complicated to be implemented in such a simple in-house network as the KUIPNET, and furthermore, it was difficult to understand the authentic meaning of the mechanism. Another reason was that we could not believe that there exists any practical application for user-level protocol, in which these mechanisms are useful in controlling behavior. Hence, at present these are not implemented in any NCP in the KUIPNET.

The NCP-level interruption mechanism (Host-Host and process-process communication level) is necessary for the unconditional transition of the current state of a process. This is necessary to implement the TELNET protocol synchronization sequence or reinitialization procedure (NIC #18639) and the abort procedure which appears in the File Transfer protocol (NIC #17759). This mechanism is also necessary in other user-level protocols to implement sophisticated operations such as interruption or suspension, purge, state-recovery, synchronization, and reinitialization. All of these functions are implemented through usage of the interrupt function on the NCP level.

Echo (ECO) and echo reply (ERP), which are used to test the

Table 3.2 Implementation of selected NCP functions in KUIPNET.

		NOS	MACC	MELCOM	HITAC	TIP	U 200	TOSBAC 5600
Control command	NOP	○	○	○	○	○	○	○
	RTS	○	○	○	○	○	○	○
	STR	○	○	○	○	○	○	○
	CLS	○	○	○	○	○	○	○
	ALL	○				○	○	○
	GVB							
	RET							
	INR						○	○
	INS						○	○
	ECO							
	ERP							
	ERR					○		
	RST							
	RRP							
Network command	NO — operation					○		
	Init	○	○	○	○	○	○	○
	Listen	○	○	○	○	○	○	○
	Close	○	○	○	○	○	○	○
	Send	○	○	○	○	○	○	○
	Receive	○	○	○	○	○	○	○
	Interrupt						○	○
	Purge					○		
	LCBRD							○
	ACCEPT							○
	INHIB							○

well-being of the NCP, are not needed so much.

Reset (RST) and reset reply (RRP) are reinitialization commands from the NCP itself. If reinitialization, synchronization, interruption, or pause functions are not available in user-level protocols, these NCP commands may be used as an alternative or next-best substitution. They are also very useful for debugging user-level protocols. Implementation of these functions, therefore, is of great advantage to the operationability of user-level protocols. Although the initialization of the NCP is done mainly through the connection tables, for all practical purposes it should include the initialization of the logger process and the initialization of processes related to the NCP.

The error-reporting functions of the NCP are implemented together with the functions of error detection; the detection of erroneous sequences or command parameters on the NCP level. These functions do not affect the operationability of the user-level protocol, but may provide a powerful means for debugging higher level protocols.

From the above discussions, it can be seen that some of the NCP functions are indispensable, some of them useful but not indispensable, and some of them not particularly important. The choice of NCP functions can be made arbitrarily by the designer of an NCP, taking into account the necessity of the NCP functions to implement the user-level protocols.

The present state of NCP implementation in KUIPNET is shown in Table 3.2.

3.2.2 Inter-Process Communication Functions

A user process is given a set of commands to communicate with a remote process through the local NCP. The interface between NCP and user process is designed by the implementer of the NCP. NCPs of usual organization communicate with a user process through a set of network commands which are almost of the same form as system calls. Generally the NCP functions become transparent to a user process *via* network commands. User processes are provided with any of the NCP functions.

From typical examples of NCPs which have been implemented so far (for example, see Refereces [TENEX, 1973 and Winett, 1973]), we deduced the set of network commands presented in Table 3.3.

A connection is established by the exchange of a pair of RFC commands on the NCP level. (See part 3.2.4). Since both the processes which are associated with the connection are in independent processors, both RFC commands are not necessarily synchronized together. Synchronization of commands means that

Table 3.3 Network commands.

Command	Description
Init	Request NCP to issue an RFC.
Listen	Request NCP to place in the listen state.
Send	Request NCP to transmit a buffer.
Receive	Accept a buffer from the NCP.
Close	Request NCP to close a connection.
Interrupt	Request NCP to issue an INS or INR.
Status	Request status from the NCP.
Purge	Request NCP to purge an entry.

when a process issues a connect request (Init) and the corresponding RFC command is transmitted through the network to another Host and then to the NCP, the partner process has already issued an Init, the arriving RFC is matched with this Init in the NCP, and a matching RFC is sent back. Another case is when the partner process has issued a Listen and is waiting for the arrival of an RFC. Command-matching fails when neither the matching Init nor Listen has been issued from the partner process corresponding to the arriving RFC. One solution to this problem is to place the arriving RFC on a queue and have it wait for the local process to issue an Init or a Listen. The problem here, however, is that nobody can decide beforehand the length of queue to place the arriving RFC on. Another solution is to regard the arriving RFC as a failure of connection establishment when the matching process is not found, and to send back a CLS instead of queuing the RFC. Many NCPs employ the latter mechanism, but in these NCPs the initiating process does not abort the connection after the first refusal by the remote NCP. It must try to establish the connection several times, sending RFC commands by issuing Inits, hence RFCs, even if the remote NCP answers by CLS. Meanwhile, the target process will issue "Init" or "Listen", the matching RFC will be returned, and the connection will be established.

"Listen" is a network command to request the establishment of a connection by the waiting for arrival of an RFC instead of issuing an RFC. Listen does not designate either the name of the foreign Host or foreign socket. The foreign Host's and socket's names are assigned by the NCP from the RFC command received on the connection establishment. This is advantageous to server processes which accept connection requests from versatile user processes. If the coming RFC is an RFC which admits an Init, a response RFC is not necessary; but if the coming RFC is one which corresponds to a Listen, an acknowledging RFC is necessary.

There are several typical combinations of network commands to establish a connection:

- (Init-Init)— Both commands must be synchronized to establish a connection.
- (Init-Listen)— If Init is issued before Listen is, the connection request fails. But a connection will be established after a succession of Init retries, at the moment a Listen from the other side is issued. If Listen is issued beforehand, a connection is established immediately upon the sending of Init. This case is most preferable.

A connection is terminated by a network command "Close". Close may be sent from either side of the process at the end of the connection. Ordered by a Close command, the NCP dispatches a CLS to the destination Host and waits for a response CLS. When the NCP receives a CLS on the open connection, then it returns a CLS and records that the connection is closed. After this the process may issue Close; the NCP itself does not send a CLS but responds to the process that the connection has already been closed. The process may issue Send or Receive instead of Close, but such cases are handled in almost the same way; the NCP simply responds to the process that the connection has already been closed.

Incoming CLSs should be interpreted differently by the NCP depending on the current state of the process: (1) A CLS corresponding to an RFC, (2) a CLS corresponding to a CLS, (3) a CLS to an open connection, or (4) a CLS to a non-existent connection. In cases (1) and (3) a CLS is returned. In case (2) no response is returned since the closing procedure has been completed. In case (4) an ERR (error) is returned. The NCP should not interpret a CLS to a socket in the Listen state as a failure of connection establishment, because the only way to terminate a Listen is to establish a connection with the other end.

There are two alternatives in the implementation of network commands: One is wherein the process which has issued an Init or Listen command may be blocked until the completion of the connection establishment by an RFC or the refusal of connection by a CLS. The other is wherein the process which has been issued the command immediately gets back the control before reception of an RFC or CLS. The former is simple to implement, but the latter is more flexible for controlling the user process. For example, the process may listen to many sockets concurrently and may be opened through any one of them from the remote process. In the latter case, the completion of the connection establishment is checked with the Status command of the NCP before the execution of data transfer.

A user process need not wait for the complete closure of the connection after a Close command is issued. If the socket is to be reopened immediately after one disconnection, the next open may fail unless it is executed after the acknowledging CLS has been received from the remote Host [TENEX, 1973]. The acknowledging CLS does not always arrive, because of some failure of the remote Host. To caution against connection failures due to this, the user process should terminate the command after a predetermined timeout period.

3.2.3 Flow Control Mechanism for High-Speed Transmission

Although user processes are not concerned with the timing when the message is sent or received, nor with the control that the procedures the message is transmitted by come from, the communication system involved must pay a great deal of attention toward the prevention of congestion and lockout and the efficiency of the communication path. Concerning these problems, a number of measures are taken [Kahn, 1972]. These are of special concern to us, since our network deals with voluminous high-speed data transmission such as the digital speech data. By filling the communication path with messages, forming a "pipe line" from source to destination, the flow control mechanism for real-time transmission in KUIPNET has achieved high throughput and low delay.

Before describing the high-speed data transmission control procedure, conventional flow control techniques will be described.

The first is the control by the RFNM (Ready For Next Message) [Heart, 1970]. When a message is sent to the destination Host, the destination IMP sends back an RFNM to the source Host assuring it that the previous message has arrived. In ARPANET this mechanism is also used to prevent reassembly lockup in the destination IMP [Kahn, 1972; McQuillan, 1972]. An RFNM is sent back to the source IMP to inform it that a reassembly buffer has been reserved in the destination IMP. However, in our network reassembly is not performed, since packeting is not employed; a message (max. 8096 bits) corresponds to a unit of transmission in the network. RFNMing, therefore, has been made optional in our network—an NCP may use it as an acknowledgement of transmission of a message, may not use it, or can even suppress its return by the IMP. The schematic expression of

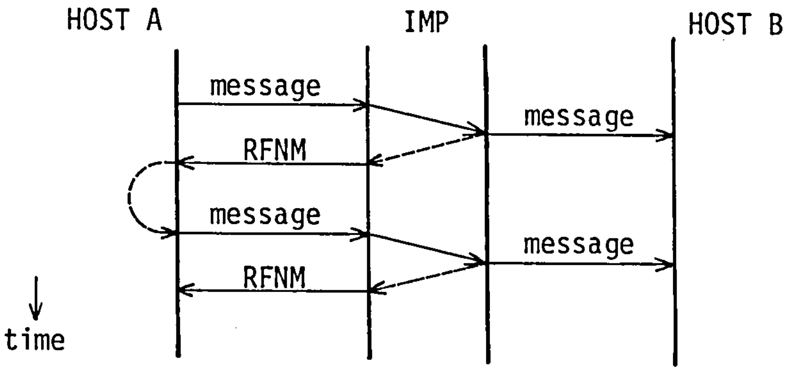


Figure 3.4 Message transfer with RFNM control.

the sequence related to message transmission using this RFNM control technique is shown in Figure 3.4. Employment of RFNM control will increase transfer delay to about twice that of non-RFNM usage for short messages, because the IMP-Host connection is half-duplex in KUIPNET. This, in turn, will also increase the processing time in both the NCP and the IMP.

The second is the control by the ALL (allocate). (See reference [McKenzie, 1972]) When the receiving Host has reserved a receive buffer space, the NCP sends an ALL, the parameter of which is the connection link number and the size of receiving space. The sending Host is prohibited from sending a message over the link until the receiving space has been allocated in the destination Host. Our interpretation of the ALL mechanism is that a Receive operation of a user process makes the NCP send an ALL, because the NCP contains no room for storing received messages. Figure 3.5 shows a schematic representation of the transfer controlled by ALL. A Send operation may fail before the receiving buffer is allocated in the destination process. The NCP will discard a message if a destination buffer for it is not found in the user process. Therefore, synchronization between the Receive and the Send is necessary by the ALL

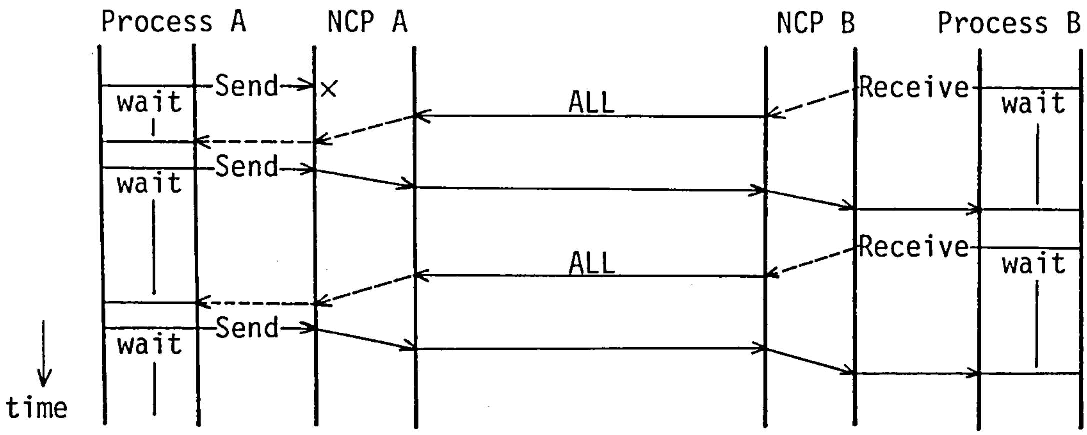


Figure 3.5 Message transfer with ALL (allocate) control.

mechanism. (See part 3.1) However, this mechanism will degrade the message throughput between processes to about half that of the non-ALL control mode, since for every message utilizing this mechanism an ALL must be received from the destination NCP. Furthermore, this mechanism will thus increase processing delay in the NCP of both ends.

Below is a description of the high-speed transmission mechanism. This mechanism is applied to real-time speech signal transmission in KUIPNET to maximize the message throughput by simply avoiding unnecessary control messages.

The two above-stated flow control mechanisms would work implicitly, we thought, if the following assumptions were satisfied:

(1) The RFNM has no meaning of flow control in KUIPNET. It was used in the early ARPANET to control traffic in the network by restricting a number of links—hence a number of messages dispatched from Hosts—because only one message can be transmitted over one link at a time [Heart, 1970]. But

later it came to be used to prevent reassembly lockup in the destination IMP, because it was proved that the most significant congestion in the network occurred because the destination Host could not obtain reassembly space for multi-packet messages [Kahn, 1972]. However, KUIPNET is a single-node network (star type) and messages in the network are not split into multi-packets. Therefore, flow control by RFNM is not necessary.

(2) The destination Host receives messages faster than the source Host sends them, and therefore, messages do not accumulate in the IMP. Even in the case of real-time transfer, when both the source and the destination Hosts are executed in the real-time mode, the destination process receives messages at an extremely fast rate; but still the destination Host is able to satisfy most real-time transfer.

(3) However, if the source process sends messages faster than the destination process receives...

The destination NCP should not discard messages received before the destination process issues a Receive. If both the processes are sound and the receiver receives the same amount of messages as the sender sends, the destination NCP may hold the incoming messages for a short while until the process receives them. In this case the IMP is forced to suspend sending successive messages which arrive from the source Host, since then the IMP will be filled with these messages. Are the other processes prohibited from sending messages? No, they aren't. Processes in other Hosts may feel a small delay in sending their messages to the IMP, where all buffers are in use. However, a new buffer will be available in the IMP because the receiving Host which is concerned with real-time transmission consumes messages from the IMP at a very fast rate.

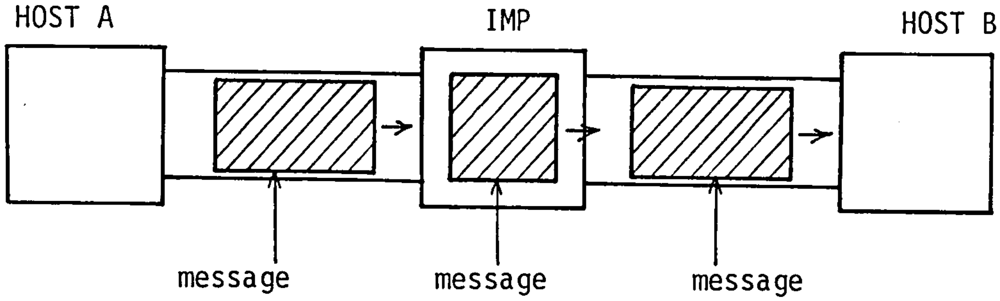


Figure 3.6 Message transfer with pipe-lining control.

(4) In the exceptional case that the destination Host does not receive messages because of some failure of the receiving process, the IMP will be filled with the messages missing their acceptor. The IMP will discard these messages after a delay of 30 seconds.

The schematic configuration of the message path in this transmission mode is shown in Figure 3.6. Messages will fill up the whole path from source to destination; one in access to the IMP, some in the IMP buffers, and one in delivery to the destination Host, maximizing the message throughput. This effect is called "pipe lining" by Rosener[1975]. This technique keeps the access pipe line 80% full (800 kbps). (See part 5.2)

To a large extent, network congestion is reduced through the combination of several features. The reduced holding time of buffers in the IMP, due to the fact that messages are transferred over high-speed line, leads to a higher probability that buffer space will be available for arriving messages. For character-oriented messages from/to low-speed terminals, the conventional flow control also helps to prevent the congestion which could result from a high-speed source (such as a computer) transmitting into the network at a much higher rate than the destination (such as a teleprinter terminal) can receive, thus holding a full buffer for an inordinately long time.

3.2.4 Example of Implementation of Network Control Program

The establishment of logical connections for inter-process communication and control of data flow over the connection are supported by the Network Control Program (NCP) resident usually as a part of the Host operating system. An example of the data structure and control mechanism of an NCP implemented in a minicomputer is briefly described.

The connection status is registered in the "Link Control Block" (LCB) resident in the NCP or in the user process, the LCBs scattered in the main-storage being chained together by their pointer field. Figure 3.7 shows the configuration of an LCB corresponding to a connection.

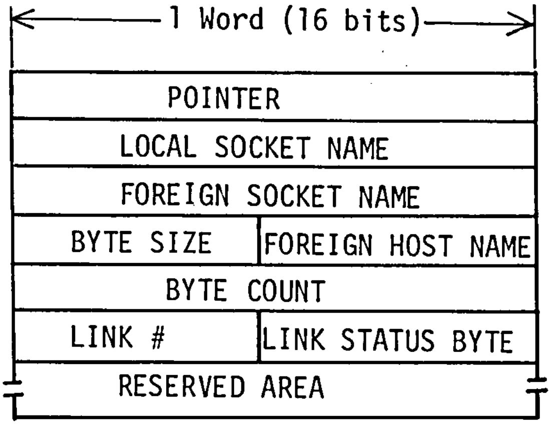


Figure 3.7 Configuration of LCB.

An LCB contains information for connection identification: The local socket name is the identifier of the local process; the foreign socket name is the identifier of the destination process; the foreign Host name is the identifier of the destination computer; the byte size is the unit of data sent over the connection, each message containing an integral number of bytes of this size; and the byte count shows the length of data area—send data or receive buffer. A link status byte shows the status of the connection—whether or not it is open or closed, waiting for a response or an acknowledgement from the other end to an issued RFC or CLS, holding a message to be received, or holding a message to be sent. The link number itself is the identifier of the link, and all messages are sent over the connection identified by this number. In order to assign a unique number to a receiving socket of a process, or to identify an incoming message with its destination socket, the NCP refers to the link table. The RFC stack is used by the NCP to recognize the process which is engaged in communication with a process in the network.

A message consists of a header and a text. The header contains a leader, byte size, and byte count. A control command is conveyed to the destination NCP in the text portion of a Control Message, over the predefined control link.

Figure 3.8 illustrates the control of the NCP resident in the main-storage of a Host. Suppose that a user process in Host A establishes a connection with another user process in Host B, transfers data, and closes the connection. The user process in Host A issues a connection request (Init) to Host B. In Host A the local and foreign socket names and the foreign Host's name are recorded in the LCB from the parameters of the Init. The Init issued is kept in the RFC stack in the NCP. From the information in the LCB, the NCP compiles a control command RFC

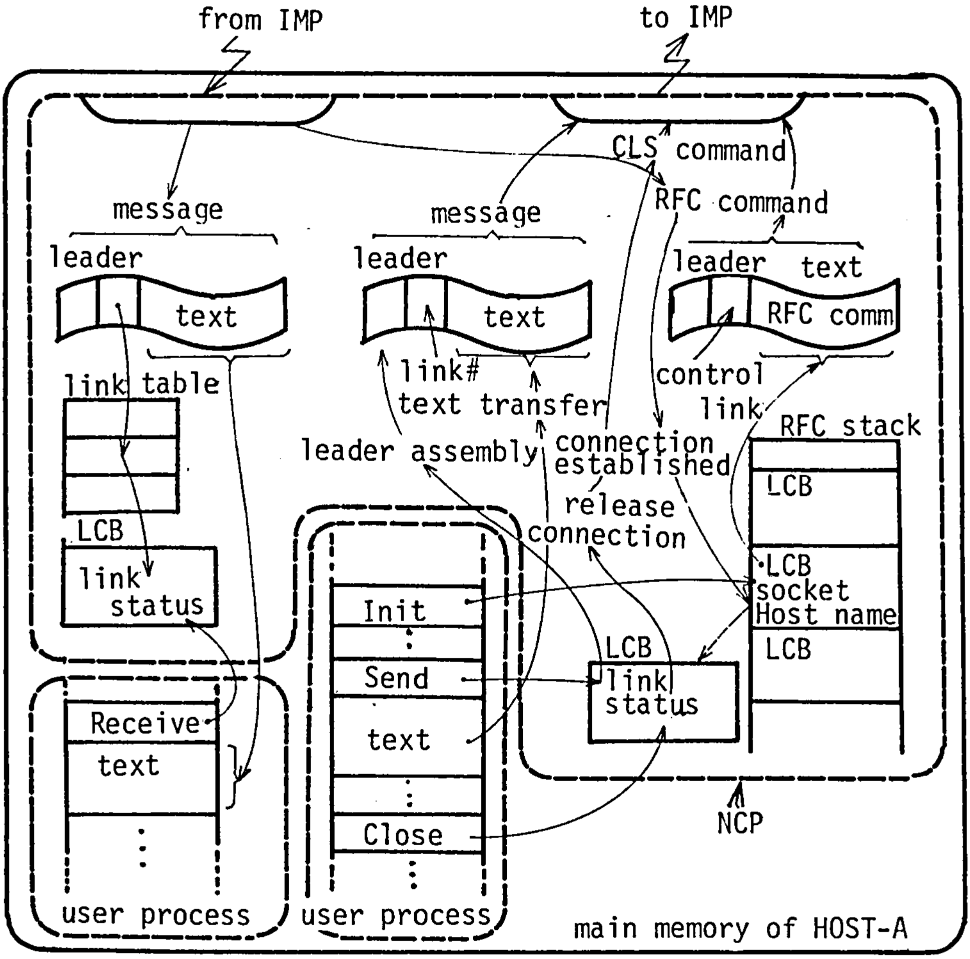


Figure 3.8 Data and control flow in NCP.

which is sent in the text portion of the message. Ahead of the text field a leader is added which indicates the destination NCP. The control command is dispatched to the destination *via* the well-known control link. The process in Host B has already issued a connection request "Listen" which is kept in the RFC stack, but does not produce an RFC command. When an RFC arrives in Host B, the NCP examines the correspondence between the RFC and the connection requests in the RFC stack, and if a corresponding LCB is found, then a link number is allocated to identify this connection and an RFC command is sent back to the NCP in Host A to acknowledge the establishment of the connection. Thus, a connection is established.

Over this connection the sender process issues a network command "Send" and transmits a message containing the data denoted in the Send command. When this message has arrived in the NCP, the receiving process is identified referring to the link number contained in the leader and in the link table. The text portion of the message is delivered to the user process which is issuing "Receive".

When either process issues "Close", the NCP sends a control command CLS to the other NCP. Then the connection is disconnected and the processes at both ends are notified that the connection has terminated.

The size of NCP is 1K words (16 bits/word) for MACC and MELCOM, and 4K characters (6 bits/character) for NEAC 2200/200 in the early version. Each includes buffer space of about 8000 bits.

3.2.5 Example of Inter-Process Communication

A special user-level protocol is implemented so that minicomputers which do not have auxiliary file devices to store their user programs can perform as co-workers in the network-wide jobs—a kind of resource sharing. This protocol allows a minicomputer to load its user program from the file system of a large computer in the network, and also includes timing control or synchronization mechanisms between user processes scattered among several computers in the network.

In this protocol, synchronization between two processes is achieved by a pair of system calls—"Init" and "Listen". An Init issued by a primary process will be successful if the secondary process has already issued a Listen; if not, it will be a failure. The primary process tries to initiate the connection again and again until the secondary process issues a Listen, by this way achieving a kind of process synchronization.

An example of the procedure of this protocol, in this case where cooperation among processes scattered between the three Hosts—MACC, MELCOM and NEAC is possible, is shown below:

First, every Host is loaded with its own NCP and monitor and is an integral part of the network-wide monitor which performs the protocol as a whole.

Either MACC or MELCOM can initiate a network-wide job. An operator may type a command *via* either MACC or MELCOM's console typewriter and initiate a network-wide job. Then all Hosts are loaded with their user processes, which are transferred from the NEAC's file system.

Suppose that MACC initiates the job. The procedures stated below are shown in Figure 3.9. Arrow 1 in the figure is the procedure where the process in MACC informs the job name to the process in NEAC by executing a series of system calls (shown in Table 3.4 as stage 1); Init, Send (job name), and Close. The process in NEAC executes a series of system calls; Listen, Receive (data), and optional Close. The process in NEAC, which issued a Listen and was blocked, is unblocked and executes the program transfer (arrow 2 in Figure 3.9, and stage 2 in Table 3.4).

By stage 3, the same task name is sent from MACC to the process in MELCOM which issued a Listen, and the process repeats

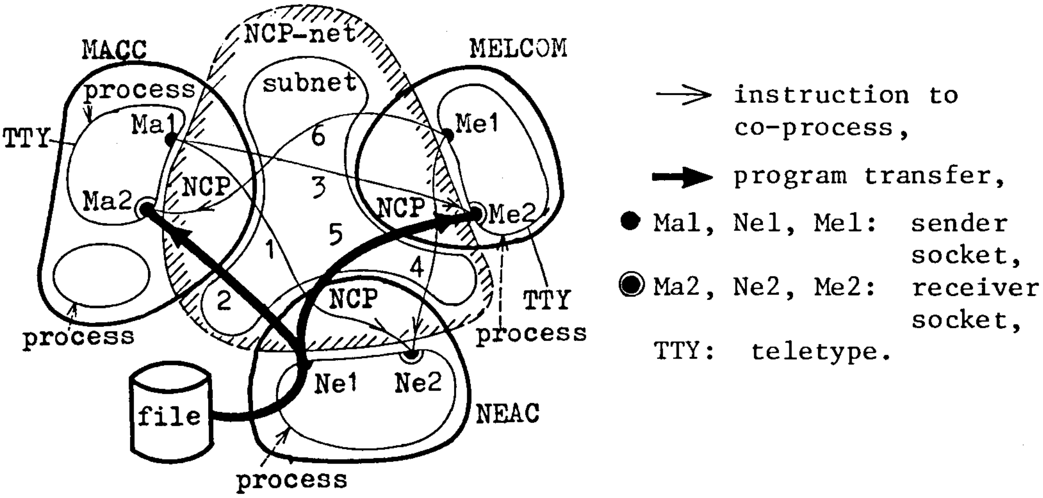


Figure 3.9 Example of cooperating processes over three Hosts.

Table 3.4 Sequence of system calls corresponding to Figure 3.9.

D: cooperating process name, d: program to be transferred.

stage	MACC	NEAC	MELCOM
		Listen(Ne2)	Listen(Me2)
1	Init(Ma1,Ne2)		
	Send(Ma1,D)	Receive(Ne2,D')	
	Close(Ma1)		
2	Listen(Ma2)	Init(Ne1,Ma2)	
	Receive(Ma2,d')	Send(Ne1,d)	
		Close(Ne1)	
3	Init(Ma1,Me2)	Listen(Ne2)	
	Send(Ma1,D)		Receive(Me2,D')
	Close(Ma1)		
4	Listen(Ma2)		Init(Me1,Ne2)
		Receive(Ne2,D')	Send(Me1,D)
			Close(Me1)
5		Init(Ne1,Me2)	Listen(Me2)
		Send(Ne1,d)	Receive(Me2,d')
		Close(Ne1)	
6			Init(Me1,Ma2)
	Receive(Ma2,D')		Send(Me1,D)
			Close(Me1)

itself in stages 4 and 5 just the same way as in stages 1 and 2. NEAC loads its program by itself and with this the three programs corresponding to the job name are distributed to every Host.

Stage 6 certifies that all Hosts are loaded with the processes which are going to cooperate in this particular job. Then the network-wide monitor passes the control to the individual process in each Host, and resource-sharing is achieved. After the completion of the job, the network-wide monitor resumes its control from each process.

Under this protocol several resource-sharing jobs are implemented. (See examples shown in part 5.4.3)

3.3 Operating Monitor for Minicomputers

Minicomputer Hosts connected to the KUIPNET provide peripheral-oriented processes: input/output processing for raw speech data; output processing of synthesized speech data; processing of photographs, including color-modified photographs; and line drawings. The operating monitor for these minicomputer Hosts was designed to provide access to the network and terminals.

The features of the monitor are as follows [Sakai, 1974]:

(1) Multi-tasking and scheduling— In order to enable a minicomputer Host to concurrently manage many peripheral devices including the network communication unit, the monitor provides the conventional multi-tasking mechanism. Although regular tasks are scheduled according to first-come first-serve basis, real-time tasks are exceptionally scheduled in

response to external events. Input/output of speech and pictorial data are handled as real-time tasks to realize high-speed processing and control. On the other hand, complicated processing of extracted parameters and reorganization of data are performed by the lower priority tasks or by the large computer system in the network.

(2) Input/output control— In cooperation with the above multitasking mechanism, input/output control is performed by the monitor in response to interruptions from peripheries. So as to reduce input/output overhead of the monitor and to achieve high-speed real-time processing, input/output instructions that are not accompanied by interruptions can be directly executed by user processes. An extended monitor function is provided to user processes—a user process is allowed to directly handle special interruptions that have been registered beforehand. This function reduces the monitor overhead's processing time and memory space accompanied with input/output control, and provides a tool for extending monitor functions to handle various peripheral devices which may be attached afterward.

(3) Users' interface— The interactive interface with the monitor is provided to operators *via* a teletype or a character-display terminal in order to control user-developed programs containing conversational processing methods. The functions available from a terminal are initiation/termination of jobs, loading of programs from auxiliary storage, and making inquiries about allocation of memory space and peripheral devices.

(4) Interfacing to the network— For input/output with the network interface device (CCCE), four kinds of interruptions are processed (see part 2.3.3) to achieve inter-computer transmission. Accompanied with the above function, the

Network Control Program (NCP) provides functions for interprocess communication control (see part 3.2.4). User processes are provided with several network commands to communicate with other processes in the network (see part 3.2.2).

The number of jobs which may be carried out concurrently is limited by the available memory space and the allowable execution time (lower priority jobs have a low probability of obtaining the processor, and hence take a long time for processing). The number of jobs regarded as practical is two at most—a real-time job and a background job.

The monitor occupies about 2K words (1 word = 16 bits), including the NCP (which contains buffer space of 500 words) for MACC and MELCOM.

3.4 Terminal Supporting Processes

In conventional systems, the "batch" mode of processing has been dominant. KUIPNET, too, is designed to allow non-time-sharing, or in other words, non-character-oriented systems such as real-time raw data processing. In a computer network—which ideally is a total complex of facilities constituting a pool of resources—however, interactive processing between the various facilities is also important, not only for time-sharing systems (in KUIPNET, NOS (see section 1.3) and GCOS (see Chapter IV) are actual examples of time-sharing systems) but also for other applications. Interactive processing makes it possible to take fullest advantage of the "pool of resources" available in the network.

To make this interactive processing possible, a simple mini-Host combined with a Network Control Program (NCP) called Terminal IMP (TIP) was implemented in the IMP. How to utilize the TIP efficiently, however, was a new problem. It was different from other Hosts because there were no pre-existing constraints to accomodate with. But for this very reason, the TIP system has achieved high performance with a small, relatively limited processor.

For the success of interactive processing the programs of the various Hosts must be able to communicate smoothly with each other. This necessitates a terminal-oriented standard interface, which in turn involves the establishment of procedure and format agreements throughout the network. Importance has been attached to the establishment of such homogeneous structures for communication by many computer network designers, and many efforts have been made about this problem [Crocker, 1972; Ornstein, 1972; Mimno, 1973; Sanders, 1976].

Among these efforts, the TELNET protocol, by its concept of a "Network Virtual Terminal" which is an imaginary but logically defined terminal, is one of the most well developed systems. It provides a fairly general bi-directional, eight-bit oriented communication facility—a character device with printer and keyboard (NIC #18639), and can be used not only for terminal-to-process communication but also for many kinds of process-to-process communications. In the case of KUIPNET, the TELNET is a part of the TIP in the IMP.

The TELNET possesses two major facets. One is the User TELNET, a human-operator oriented system which controls the terminal devices, and the other is the Server TELNET, a process oriented system which typically serves User TELNET processes. Both possess identical interfaces.

User TELNET: The User TELNET interprets the commands from the TIP and controls the input/output of terminals. At present, the teletype is the only terminal being served by this User TELNET in the KUIPNET. The input/output control block, when there is no character to print, watches the status flag which is set when a character is typed on the keyboard. From the characters typed, command characters discriminated by control codes are stored in the command buffer, while the remaining characters are stored in the input buffer. The complete command, when it has been gathered, is then interpreted and executed.

From messages passed from the IMP, control commands are processed in the NCP. (See section 3.2.4.) The remaining messages are passed from the NCP to the text receiver of the User TELNET. After some code conversion in order to interpret the meaning of the text, the received text is handed to the input/output control block. Since the teletype is operated in half-duplex mode, it is set in the input-from-keyboard mode when there is no output. At the moment an output character is placed on the print-out queue, the teletype is set in the output mode and starts to print out. Input from the keyboard is blocked until all output characters have gone out. Output characters are generated independently both by the command interpreter and by the text receiver. These texts are put on a print-out queue and wait for processing by the input/output control block executed as an autonomous task.

Server TELNET: The Server TELNET, a specific process prepared to give access to the remote Host computer, is described in detail in section 4.6. It should be explained here, however, that under the Server TELNET there is a special Logger process which supervises the server processes and is the mechanism that interprets the commands sent through the TELNET. A main frame of the Logger is the Initial Connection Protocol (ICP) procedure [Postel, 1971]. Upon arrival of a connection request from a User TELNET process, the Logger creates a server process and reconnects the user process to it—this action being the ICP.

Besides being used in the IMP, the TELNET systems are also implemented in several Hosts. For example, the two minicomputers MACC and MELCOM each have User TELNET processes. NOS (NEAC 2200/250), developed by Hayashi[1977], has both User and Server TELNETs and is operated completely in the time-sharing mode. For TOSBAC 5600 too, both the User TELNET and Server TELNET are implemented (see Chapter IV).

The TIP, in addition to having a User and Server TELNET, also consists of magnetic tape option (MT option) and the NCP sub-tasks.

MT option: The functions of the MT option will be described in the following section. The relation between the MT option and the User TELNET, however, is as follows: The command executive of the MT option is started by "@ M" type commands typed locally *via* the User TELNET. Commands and response messages returned from the destination Host are trapped in the User TELNET text receiver block before they are handed to the print-out queue, and are used to control the MT option.

NCP: The NCP of the TIP simulates IMP-Host communication through the IMP system calls (see section 2.3.4) whose functions are (a) removal of a message from the HOST-OUT queue (see section 2.3.1) and (b) addition of a message to the HOST-TASK queue (see section 2.3.1). The input process from the IMP is performed by the status check of the HOST-OUT queue. If there is any message on the queue, it is transferred to a receive buffer in the TIP, and then the IMP buffer is returned to the free-buffer list. The output process to the IMP is performed by transfer of the output buffer in the TIP to the IMP buffer which is obtained from the free-buffer list. Then the IMP buffer is placed on the HOST-TASK queue. The NCP has twelve ports, half of them allocated to sender sockets and the other half to receiver sockets. A pair of sender and receiver ports is allocated to the User TELNET, the ICP logger, and the MT option, and three pairs of ports are allocated to the Server TELNET. The NCP is used concurrently by six processes each of which possesses a pair of these ports.

Monitor routine: Execution of all the above-mentioned TIP sub-tasks is scheduled by a monitor routine which is directly coupled with the background routine (see section 2.3.3) of the IMP. The monitor routine passes the control to each sub-task by the following principles: (1) The input/output control block is started when the teletype is found to be ready for the next output or a new input character; (2) The NCP is started when a message has arrived from the network; (3) Other sub-tasks are started by the occurrence of the events—input/output completion, connection establishment, allocation of buffer space, disconnection, or timeout. The started task completes its execution by naming the expecting events after processing of data and notifying the other tasks of occurence of new events.

3.5 Magnetic Tape File Transfer

The file transfer mechanism is an important data-sharing facility in a computer network. In ARPANET, it is named File Transfer Protocol (NIC #17759). In

KUIPNET, file transfer is important for medium conversion; for instance, conversion between two different types of magnetic tape files—one written on the 9-track 800 bpi tape drive unit, the other on the 7-track 556 bpi tape drive unit. As was described in section 3.4, an optional TIP function to transfer these magnetic tape files was implemented in KUIPNET. To begin with, a pair of TELNET connections is established with the destination Host *via* the TELNET's ICP mechanism provided by the TIP. Magnetic tape commands are entered to the magnetic tape option routine which controls tape drive unit and manipulates files. The magnetic tape files are transferred in a way that allows any set of codes and any file format. Here the file transfer mechanism is briefly described, including the correspondence of data and file formats.

Data conversion format: A unit of data conversion is a frame on a tape. Eight bits, or one byte, is a unit on 9-track tape, and six bits, or one character, is a unit on 7-track tape. Let us consider the case of a 9-track tape being copied onto a 7-track tape. The 8 bits of the n-th byte on the 9-track tape corresponds to the under 2 bits of the (2n-1)th character and 6 bits of the 2n-th character on the 7-track tape. All bit positions in a frame are filled in lower-order bits, and any bit positions unused are filled up with logic zeroes. A record on the 9-track tape is correspondingly converted into a record on the 7-track tape. The maximum record length of 9-track tapes, being restricted by read buffer size, is 2048 bytes.

File conversion format: A file on a 9-track tape is defined as a sequence of records enclosed by two tapemarks. The first record of a tape begins with a BOT (Beginning Of Tape) and ends at with a tapemark. The end of the records is defined by a double tapemark. A file on 7-track tape is defined as a sequence of records which begins by a HDR (Header) record and ends by an EOF (End Of File) record, the next file beginning by another HDR record and ended by an EOF record. The end of all records is defined by an ERI (End of Real Information) record.

Files in the above-mentioned two different mediums correspond with each other. For example, suppose a 9-track tape is transferred to a 7-track tape, a destination file where the records are stored is opened, wherein a header label is written down on the tape. A record is read from the 9-track tape and transmitted to the destination Host. The process that performs transfer in the Host converts a frame (eight bits) into two characters (twelve bits) as stated above, and reproduces a record to be written into a tape; that is, an output record on 7-track tape is converted so as to have twice as much length of frames as that of the record originally written on the 9-track tape. This process continues until a tapemark is found, when a trailer label is written on the output tape. Three EOF records are written and two records are back-spaced. If new records follow, they are treated as a new file, and a header label is produced to precede them. When a tapemark is followed by a tapemark from the input tape, this means the end of the recorded data and that no more records remain. Then the connection used for magnetic tape transfer is closed. The process in the Host finds that the sender process has closed the connection and closes the file by writing two trailer labels followed by an ERI and an EOR (End Of Reel), and rewinds the tape.

File transfer protocol: The file transfer mode of KUIPNET is almost analogous to the eight-bit byte stream mode of the FTP of ARPANET (NIC #17759). A unit of a byte is composed of eight bits and data is trasmitted in the form of a stream of bytes.

EOR (End Of Record) and EOF (End Of File) are represented by two bytes of control codes which describe the records and files. Control code 1 represents an EOR, 2 represents an EOF, and 3 represents an End Of Record & End Of File. Escape characters—all logical—which appear in the data stream are doubled by the sender to distinguish them from control codes. The data stream is conveyed by a sequence of messages whose length is fixed to 72 bytes. A record may end in the midst of a message, the characters following after an End Of Record code included in the next record. Characters following after an End Of File code are padding bytes and are all zeroes. This protocol allows the transfer of any pattern of binary data.

As we can imagine from the above description, a fairly good file conversion mechanism is provided between two different types of machines—a byte machine and a character machine. However, this conversion mechanism is very complicated and requires lengthy processing time. This is because the conversion process contains a number of bit-shifting operations and each byte must be separated from a data stream to test the control codes. Fortunately, we can appreciate the high-speed transmission capability of KUIPNET. Usually, file transfer speed is rather limited not by the line speed but by two factors—the processing time in computers and the mechanical speed of tape drive units. From our experience, it takes about 20 minutes or so to convert a reel of 1200-foot tape.

CHAPTER IV: EXTENSION OF IN-HOUSE COMPUTER NETWORK KUIPNET TO REMOTE COMPUTER SYSTEM (TOSBAC 5600/140)

KUIPNET is not necessarily confined within a limited location—in-house, but can be widely extended. Two major efforts have been made toward the extension of our computer network: (1) connecting it to a remote large computer in Osaka *via* 4,800 bps line, and (2) making possible an inter-university computer network by connecting two computer centers, one in Tokyo and one in Kyoto *via* 48 kbps line. In this chapter the former effort, in which the author participated, is described. (See Reference [Yanagi, 1976])

4.1 Outline

TOSBAC 5600/140 (which is equivalent to Honeywell's 6040) is a distant Host at the Kansai Institute of Information Systems (KIIS) in Osaka connected to the IMP through 4,800 bps communication line. The key functions accomplished by this connection are: (1) communication control with a Very Distant Host; and (2) establishment of an intellectual Server TELNET which provides an authorized time-sharing system (TSS). The connection between TOSBAC 5600 in Osaka and the IMP on the Kyoto University campus, located 50 km apart, is an extension of the local IMP-Host connection. For precision on a communication line, a mechanism for reliable transmission was introduced (See 4.3). TOSBAC 5600/140 is called a Very Distant Host (VDH) since it has an NCP and provides Host-Host transmission procedures.

The powerful operating system GCOS provides versatile subsystems including the TSS. Access to this is very desirable for the KUIPNET. TELNET allows a way for remote terminals in the network to gain access to the terminal-oriented processes; typically, the TSS. The usefulness of the TELNET is augmented by the vigorous TSS. A new approach to open an entry to the TSS from the remote TELNET was attempted on the IMP. Without altering the GCOS-TSS system, a Server TELNET was implemented in the IMP to serve as the external computer of TOSBAC 5600.

4.2 Configuration of TOSBAC 5600/140 - KUIPNET Connection

The configuration of the VDH (TOSBAC 5600/140)-KUIPNET connection is shown in Figure 4.1. First, several layers of interfacing blocks for the VDH are described below. The communication line which connects the two computers 50 km apart is at the transmission rate of 4,800 bps and is leased from N.T.T.. The communication controllers at both ends are operated in synchronous full-duplex (though half-duplex in TOSBAC 5600) basic mode by the IMP (NEAC 3200/50) and DN340 (a front-end communication processor of TOSBAC 5600). The communication control between the IMP and DN340 is an extended type of the basic mode procedure, called DDS120B interface in TOSBAC 5600. The communication between DN340 and TOSBAC 5600 is performed by two packages, GRTS-340 in DN340 and GRTS-5600 in TOSBAC 5600. In the IMP the communication with TOSBAC 5600 is controlled by two packages, DDS-120B and GRTS, both of which have the corresponding functions of those inherent in DN340. The VDH

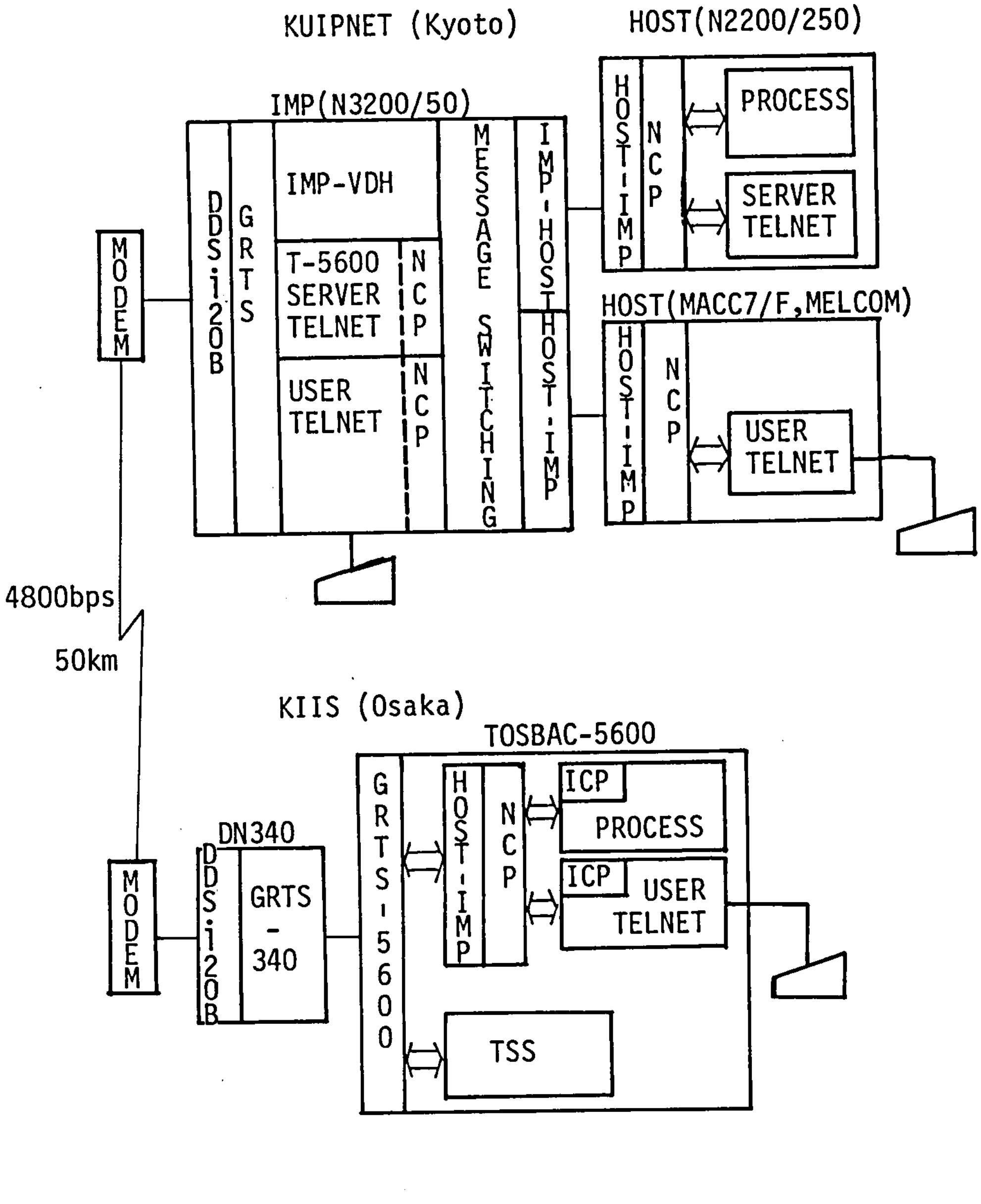


Figure 4.1 Configuration of VDH(TOSBAC 5600/140)-KUIPNET connection.

special communication control between IMP and Host is performed by the two packages IMP-VDH in the IMP, and HOST-IMP in TOSBAC 5600. Messages destined to TOSBAC 5600 are handed to the IMP-VDH package and messages sent from TOSBAC 5600 are handed to the message-switching program *via* the IMP-VDH package. NCP-level communication is performed by two NCPs, one in TOSBAC 5600 and the other in KUIPNET.

Besides being a VDH, TOSBAC 5600 has another aspect—a TSS system. A user process in the KUIPNET can communicate with the TSS in TOSBAC 5600 through the IMP. The IMP, by the complex function of two packages—DDS-120B and GRTS, performs as if it were an inherent remote terminal concentrator of TOSBAC 5600. From outside, the GRTS interface is seen virtually as a set of terminals, each of which is mapped onto a Network Virtual Terminal (see 3.4) having a pair of TELNET sockets. The "T-5600 SERVER TELNET" package (a Server TELNET) and the NCP provide a Host function which allocates a process corresponding to a virtual terminal in response to a request for a TELNET connection from a user process. At the beginning of communication with the TSS, a user process connects with the Server TELNET in the IMP (TIP). After establishment of a connection, the Server TELNET is invisible to a user process, being apparently connected with the TSS beyond the Server TELNET. Thus any user process which has a TELNET interface, including TIP itself, may become a virtual terminal of the TOSBAC 5600 TSS.

4.3 Communication Control Procedure

Although no special control mechanism is required within a distance of several hundred meters of IMP-Host communication, a modem and communication controller with error control mechanism are employed for the connection with a Host located at a distance of 50 km.

Basic Communication Control:

The communication control procedure between IMP and VDH is a variation on the basic control mode of 7-bit character JIS-code with an eighth parity bit. It is called the DDS120B-type communication control procedure, and was originally used between the front-end-processor DN340 and its terminals. In order to realize bit-transparency concerning communication between NCPs (network mode), small modifications are introduced into the DDS120B interface.

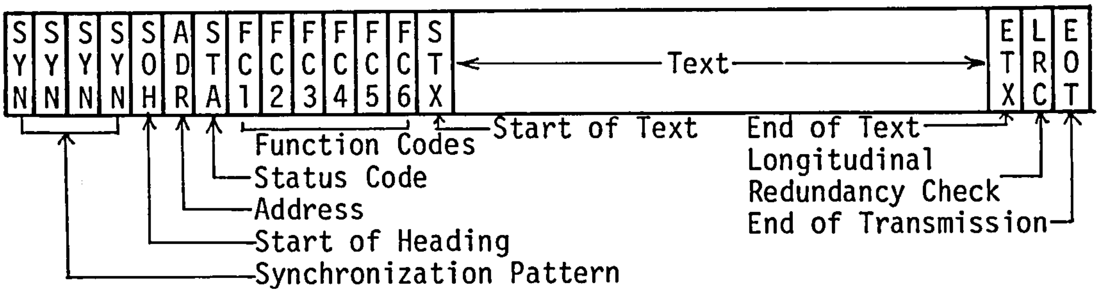
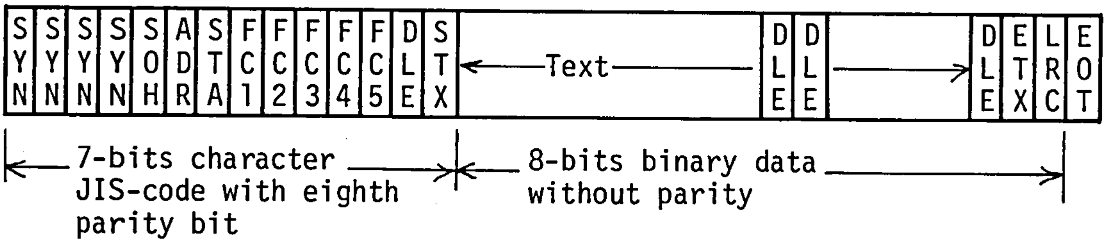


Figure 4.2 Message frame format on DDS120B interface.

(1) Message format between DN340 and IMP

Figure 4.2 shows the message format of a frame transmitted between DN340 and the IMP. ADR(address), represented by a set of eight bits—p1laaaaa (p is a parity bit)—is an identifier of terminals. The network mode is particularly represented by aaaaa = 00000. STA is a status code representing either NUL(null), ACK(acknowledge), NAK(nack), BSY(busy), INT(interrupt), BRK (break), or DIS(disconnect). FC1 - FC6 are function codes — all zero for regular messages; FC6 = DLE(data link escape) when in transparent mode. The text part is composed of a sequence of 7-bit JIS-code characters with parity. LRC is a longitudinal redundancy check code computed by the exclusive-or operations from ADR through ETX(end of text).

For bit-transparent transmission, an 8-bit code is used instead of the 7-bit JIS-code. Figure 4.3 shows the message format of a bit-transparent mode frame. The pair of control characters DLE/STX(start of text) shows the beginning of a binary mode text, and the pair of control characters DLE/ETX(end of text) shows the end. DLE/DLE may appear in a binary mode text, meaning a data byte DLE. In the transparent mode, a data byte DLE is dispatched with an extra DLE from the sender, one of them being removed upon receipt.


SYN
SYN
SYN
SYN
SOH
ADR
STA
FC1
FC2
FC3
FC4
FC5
DLE
STX
Text
DLE
DLE
DLE
ETX
LRC
EOT
7-bits character JIS-code with eighth parity bit
8-bits binary data without parity

DLE: Data Link Escape

Figure 4.3 Message frame format in transparent mode.

(2) Connection establishment between DN340 and IMP

IMP issues an empty frame (EF) after a predetermined interval of no traffic. If DN340 has no outstanding message frame (MF), an EF is sent back to the IMP. In response to an EF, IMP may send an MF to DN340.

(3) Message and response

The configuration of message frames and their possible response messages are shown in Figure 4.4.

Connection Establishment:

GRTS is a set of communication functions of GCOS resident in DN340. Remote terminals (*e.g.*, IMP) must connect with GRTS-340, which is a subsystem of GRTS, before connecting with GCOS (TOSBAC 5600/140). There are two possible connection modes, DAC(direct access) and TSS. Since the NCP in TOSBAC 5600 is implemented as a user process (see 4.4), it communicates with another NCP *via* DAC mode, and since the Server TELNET is implemented in the IMP, it communicates with the TSS in TOSBAC 5600 *via* TSS mode. Other available control commands are Disconnect and Break.

Higher Level Communication Control:

Communication control between Host and IMP is usually

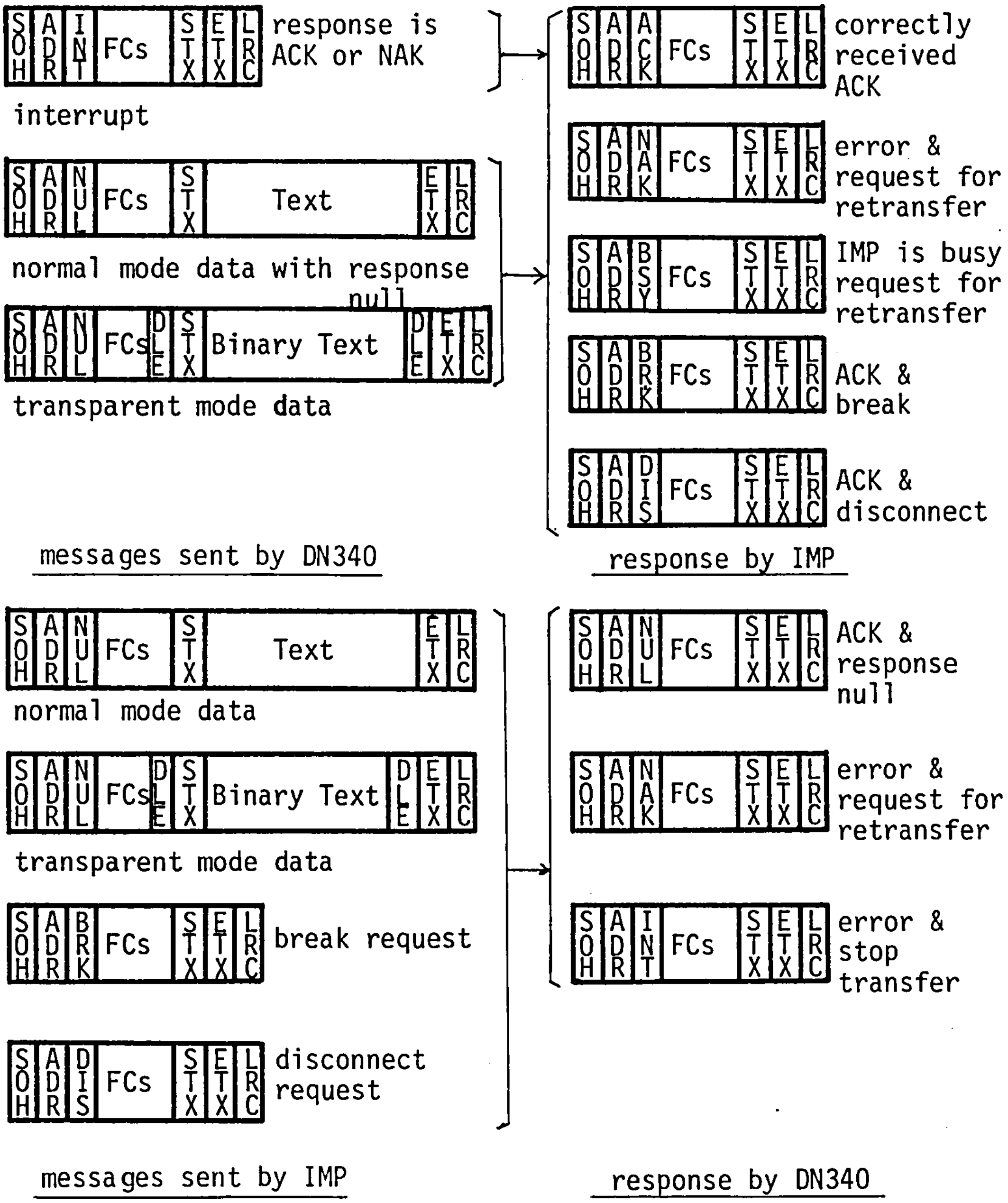
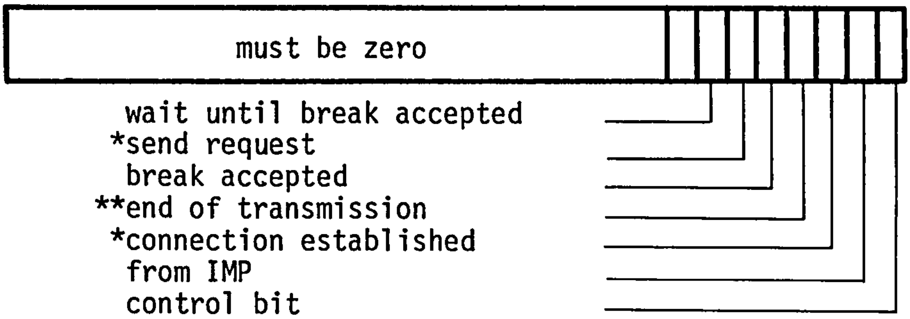

SOH ADR INT FCs STX ETX LRC
response is ACK or NAK
interrupt
SOH ADR NUL FCs STX Text ETX LRC
normal mode data with response null
SOH ADR NUL FCs DLE STX Binary Text DLE ETX LRC
transparent mode data
messages sent by DN340
SOH ADR ACK FCs STX ETX LRC
correctly received ACK
SOH ADR NAK FCs STX ETX LRC
error & request for retransfer
SOH ADR BSY FCs STX ETX LRC
IMP is busy request for retransfer
SOH ADR BRK FCs STX ETX LRC
ACK & break
SOH ADR DIS FCs STX ETX LRC
ACK & disconnect
response by IMP
SOH ADR NUL FCs STX Text ETX LRC
normal mode data
SOH ADR NUL FCs DLE STX Binary Text DLE ETX LRC
transparent mode data
SOH ADR BRK FCs STX ETX LRC
break request
SOH ADR DIS FCs STX ETX LRC
disconnect request
messages sent by IMP
SOH ADR NUL FCs STX ETX LRC
ACK & response null
SOH ADR NAK FCs STX ETX LRC
error & request for retransfer
SOH ADR INT FCs STX ETX LRC
error & stop transfer
response by DN340

Figure 4.4 Configuration of message frames
and their possible responses.

specified by the IMP-Host protocol, but communication control between VDH and IMP requires an additional control mechanism. Most of the necessary controls related to the communication line (*e.g.*, sequencing, error control) are performed by GRTS and and DDS120B, but some kinds of control are performed using information from an IMP-VDH control-word(32 bits) as shown in Figure 4.5. The first 32 bits of binary mode text is used as the control word, and the rest of the text which contains a leader and a header is the message defined by the IMP-Host protocol. The control word contains status flags used especially for communication control between the IMP and the NCP in the VDH. The basic functions of the IMP-VDH protocol are founded on the GRTS functions. Connection establishment and data transfer control are described as follows:



* is effective in NCP → IMP
** is effective in IMP → NCP

Figure 4.5 IMP-VDH control-word.

(1) Connection establishment between the IMP and the VDH

After the establishment of connection on the DDS120B interface level, the IMP opens a communication link with the NCP, GRTS-340 informs the NCP of the opening of this link, and also informs the IMP of the Connection Established state by turning on the flag bit in a control word.

(2) Data transfer from VDH to IMP

After the establishment of connection, VDH can send data to the IMP if there is any data to send; even a sole control word. The length of data must be less than 8096 bits.

(3) Data transfer from IMP to VDH

The IMP interrupts the VDH by a Break command when IMP has data to send to the VDH. On detection of a Break command, if the VDH has any space to accept data, it notifies the IMP by a control word, turning on the flag of Break Accepted. If there is not any space to accept data in the VDH, the VDH responds to the IMP by turning on the flag of the Break Accepted But (the transmission must) Wait in a control word, and continues its sending process to the IMP. If after transfer of a Break command there is no Break Accepted nor Break Accepted But Wait in two successive non-empty messages returned from the VDH, the IMP regards the Break command as not having been received by the VDH. In continuous transmission of data from IMP to VDH, the IMP must halt transmission between every message and wait for a Break Accepted. On reception of a Break Accepted, the IMP sends a control word of Transmission Finished if there is no further data to send.

(4) Alternation of direction of transmission between IMP and VDH

After continuous reception of several messages from the IMP, if the VDH has data to send to the IMP, the VDH sends a control word of Stop Transmission. The IMP responds by a Break Accepted But Wait if the IMP has any more data to send, or if the IMP has no more data to send, by simply a Break Accepted. The VDH begins to send data in the case of Break Accepted, but returns a Break Accepted to the IMP in the case of Break Accepted But Wait. After a succession of transmission (typically 10 messages) to the IMP, the VDH sends a Break Accepted and tries to accept a message from the IMP. The IMP may either send data, if any, or an empty frame if there is no data to send out.

4.4 Network Control Program for TOSBAC 5600/140

The NCP implemented in TOSBAC 5600 is exactly harmonized with the IMP-Host and Host-Host protocols in the KUIPNET. This NCP was implemented as a user-level job program in order to avoid introducing any modification to GCOS, since it is difficult to change the monitor in the main Host and new monitors pose continuing compatibility problems. While this approach is good in principle, and in fact was the only possible approach, it often leads to problems. The I/O system between the main Host and the peripheral processor is not designed for network connection and presents serious timing and bandwidth constraints that greatly degrade performance (see 4.5). Furthermore, the logical protocols that have preexisted preclude the main Host from acting as a general-purpose Host on the network. For instance, only batch jobs are accessible *via* the NCP, and TSS mode jobs are not.

4.5 Measurement and Estimation of Throughput

The message throughput was estimated between IMP and NCP in TOSBAC 5600. Message transfer is controlled by the IMP-VDH protocol (see 4.3), the procedure in which a message is transmitted being as follows:

Parentheses ( ) and < > represent a DDS120B-level message sent from DN340 to IMP and a message sent from IMP to DN340, respectively. A message is sent from IMP to NCP in the sequence; (Break Accepted), <ACK>, (EF), <message>, (Response Null), <ACK>, (EF), <EF>. A message is sent from NCP to IMP in the sequence; (message), <ACK>, (EF), <Break>, (Break Accepted), <ACK>, (EF), <RFNM>, (Response Null), <ACK>, (EF), <EF>. The processing delay both in the IMP and TOSBAC 5600 was quantified by the roundtrip delay of an empty frame. Transmission delays of individual messages are proportional to the messages' lengths and can be estimated based on the bit rate of the line (2.08 ms/bit). Figure 4.6 shows the computed message throughput between the IMP and the NCP in TOSBAC 5600. The result is shown as the ratio of throughput to the line speed of 4,800 bps.

In comparison with this, another measurement report is presented. V.G. Cerf reported in his Very Distant Host measurements that the actual line utilization was about 20% in each direction, assuming a normal 50 kbps available in full-duplex capacity between sites, where the average text length was 560 bits [Cerf, 1976]. The actual bandwidth of delivered data on a full-duplex line was 3.8 kbps, and average message rate was 6.6 messages per second. He used a very complicated transmission procedure (large overhead) and high-speed full-duplex line (50 kbps), while we use a rather simple procedure (less overhead) and low-speed half-duplex line (4,800 bps).

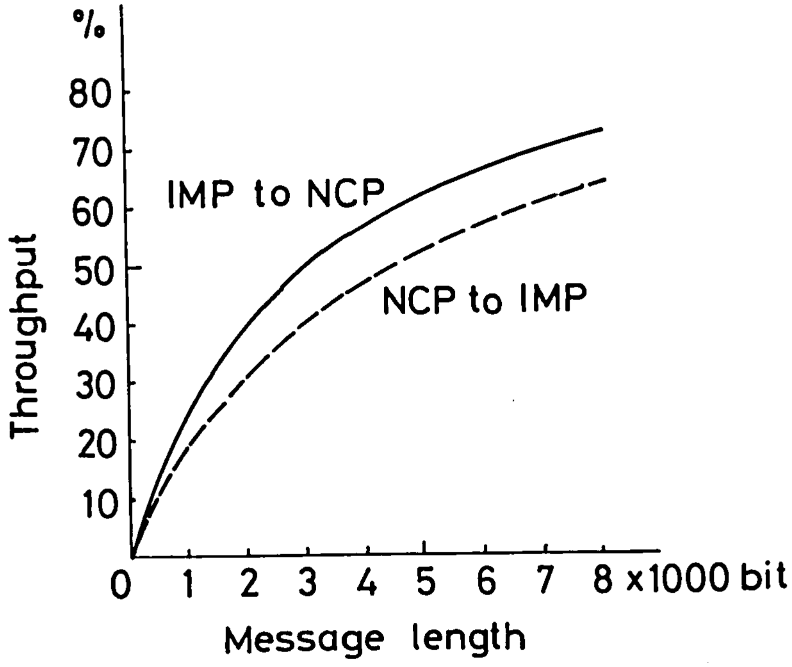


Figure 4.6 Computed message throughput between IMP and NCP (TOSBAC 5600).

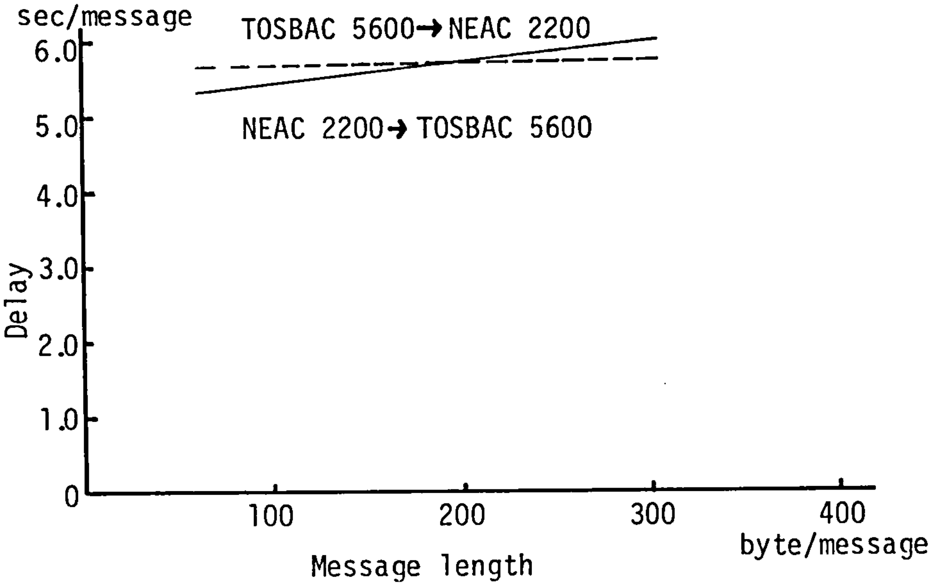


Figure 4.7 Message delay between TOSBAC 5600 and NEAC 2200.

The throughput between a pair of user processes is degraded because of response delay induced by the communication between DN340 and the main TOSBAC 5600, by the flow control between NCPs which is performed by exchanging allocation commands, and by the operating system overhead. The message delay observed between user processes in TOSBAC 5600 and in NEAC 2200/250 is shown in Figure 4.7. In a practical application, a frame of a picture (128 x 128 elements, total 101,376 bits) was sent from Kyoto (NEAC 2200/250) to Osaka (TOSBAC 5600). It took 221 seconds, where the text length was 3168 bits. The actual delivered data bandwidth was 0.46 kbps.

4.6 Higher-Level Communication Function —— Server TELNET for TOSBAC 5600 TSS Subsystem

Use of the IMP to connect the TSS in TOSBAC 5600 to the KUIPNET with no Host computer hardware or software modifications has been accomplished. For this, the NCP was removed from the Host computer and placed in the IMP system to become a front-end processor of TOSBAC 5600.

One way of connecting a Host is through the use of a number of slow speed terminal ports back-to-back with the terminal ports of the IMP. This particular approach is straitforward, with no modification to the Host computer; however it is wasteful in the large number of asynchronous interfaces it requires. In our case, the connection is made through the interface of a miniprocesser-based remote concentrator designed to concentrate data from as many as 32 terminals. The basic software to serve a defined number of lines can be configured by the manufacturer. On the IMP side, special applications are required so that it can be incorporated with the standard routines in TOSBAC 5600.

The Server TELNET (terminal oriented interfaces, see 3.4) in the IMP, combined with TIP is a cooperating process with GCOS in TOSBAC 5600 and appears as a provider of the time-sharing system to user processes in the KUIPNET. The major functions of this process are as follows: (1) connection establishment with processes supported by the User TELNET through the ICP (Initial Connection Protocol) [Postel, 1971]; (2) protocol conversion between the DDS120B-type interface on the TOSBAC 5600 side and the TELNET protocol on the KUIPNET side; and (3) management of buffer space.

(1) ICP and logger:
In the ICP process, the logger part of the Server TELNET assigns a pair of server sockets and the NCP assigns a link number to individual connection requests. The logger searches for a pair of free sockets and allocates them to the request from a user process. The sockets are returned to the free list when the connection has been closed.
(2) Data format conversion between DDS120B interface and TELNET:
Data within the TELNET connection is transmitted a line at a time ending by a CR/LF, and has less than 72 bytes. The Server TELNET deletes the CR/LF at the end of each line and hands the line to the DDS120B interface. The Server TELNET also splits a message from the DDS120B interface, where the length of a message is at maximum 1280 bytes with a parity bit, into several segments of less than 72 bytes with parity bits off, scanning

the text of the message from head to end for CRs to split the text into multi-lines, and sends them over the TELNET connection to the User TELNET.

(3) Allocation algorithm of buffer space:

Input buffer space in the Server TELNET is divided into 72 bytes of fixed length for each of three user ports. 1280 bytes of output buffer space is reserved in order to accomodate the maximum transmission block size from TOSBAC 5600. However, the probability that all users will use the maximum block size of 1280 bytes at the same instance is very small. Therefore, the buffer field of 1280 bytes is dynamically divided into three parts (zero byte is possible) and co-used by three individual users, hence making it possible for every user to be supplied the maximum length of 1280 bytes, though not coincidently. The first user is allocated a buffer area from the top of the field, the second from the tail, and the third from the middle of the unused field equally extended in both directions.

The Server TELNET accepts only User TELNET processes constructed through the User TELNET protocol. Those which satisfy the above conditions are implemented in two Hosts—TIP and NOS (NEAC 2200/250). The prime points to consider for new implementations on other Hosts are: (a) the ICP and the allocation mechanism; and (b) the size of a unit of transfer—under 72 bytes—to adjust to the buffer length of the Server TELNET. Up to three user processes are connected to the Server TELNET, hence to the TSS of TOSBAC 5600, and served concurrently. The kinds of terminals which are provided services are DDS120B display, teletype, typewriter (132 characters/line), and 4010-type display.

This connection has been successful, since users in KUIPNET can appreciate the full set of TSS functions. However, these approaches to Host interfacing that require minimal effort limit the Host computer's network interaction to that supported by existing Host communication hardware and software. For instance, user jobs in TOSBAC 5600 can be activated remotely, but TOSBAC 5600 itself can not initiate any job in the remote Host computers. Any connection with TOSBAC 5600 must be initiated from the remote terminals.

110 項欠

CHAPTER V: PERFORMANCE MEASUREMENTS OF SWITCHING COMPUTER AND NETWORK

5.1 Introduction

When constructing a computer network, the validity of the design can be tested through quantification of its actual performance.

Other papers on the performance of computer networks have usually discussed the problem *via* the combined use of analysis, simulation, and measurement. Analytical considerations, applying queueing models, have been given by Kleinrock[1970] and some others. The major problem dealt with in these reports was the queueing delay suffered by messages or packets in the network nodes. The queueing model is effective in a large system where the contribution of individual factors is small; however, not effective in those systems where network traffic is heavy or interference between factors is high. The analytical results are examined with simulations of the network behavior. In order to simulate a complicated system, a detailed model is necessary, though the level of detail should be held down to avoid long simulation run times. The model, therefore, is simplified causing important details to be neglected. The results obtained from measurements are the most realistic as compared with the two other approaches. However, some analytical models are also necessary to understand the measurement results. Cole[1972] measured the performance of the early ARPANET at the Network

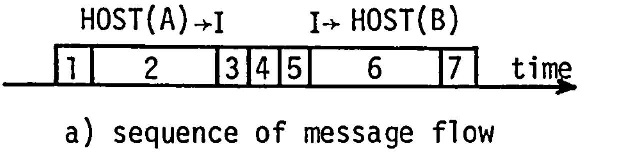
Measurement Center, and its traffic characteristics have been observed by Kleinrock[1974]. Kleinrock[1976] studied the line overhead through analysis and measurements.

Usually the performance of many computer networks has been discussed under the assumption that network traffic is very light. On the other hand, the behavior of the network under heavily loaded conditions is important in our case, bacause the high-speed transmission required in our network induces an extremely heavy load which would scarcely appear in other large networks.

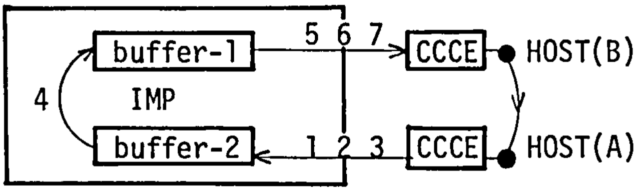
In the following sections, the performance of the switching computer is discussed through measurement and computational estimations. Under extremely loaded conditions, the characteristics of the programmable priority control circuit which is described in the section "Control Hardware" (see 2.2.2) is tested. The throughputs between the computers in the in-house network have been measured. A measurement tool has been developed to investigate the traffic in the network and observe the characteristics of the traffic caused by digitized speech wave forms, pictorial data and so on. Furthermore, a network measurement system has been developed which gathers data on the usage of the network—resource utilization of the switching computer, traffic matrixes, and control of the network.

5.2 Performance Measurement of Switching Computer

As shown in Figure 5.1, two lines of the switching computer IMP bound for two Hosts (four coaxial cables) are short linked to form a closed loop. Messages are circulated on this loop and



a) sequence of message flow

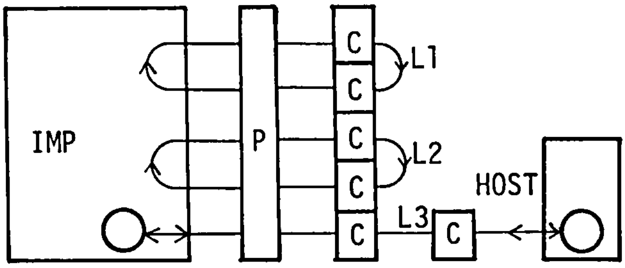


b) circulation of message

Figure 5.1 Factors of message delay in IMP.

their throughput is measured.

The performance of the IMP was investigated by the following method: Two similar closed loops *L1* and *L2* were made, and also an extra logical loop *L3* on which messages would go to a Host and then come back (Figure 5.2).



P: programmable priority control circuit
C: computer connection control equipment

Figure 5.2 Configuration for measurement of IMP throughput.

The time spent for a message to pass the IMP depends on the following time-consuming factors (Figure 5.1-(a), (b)):

(1) Establishment of transmission path from Host (A) → IMP.
(2) Data transmission from Host (A) → IMP.
(3) Release of transmission path from Host (A) → IMP.
(4) Message-switching in the IMP.
(5) Establishment of transmission path from IMP → Host (B).
(6) Data transmission from IMP → Host (B).
(7) Release of transmission path from IMP → Host (B).

Terms (1), (3), (4), (5), and (7) are delays due to processing by the CPU, and are denoted by d_p (μsec). Terms (2) and (6) are delays due to the transmission rate of the IMP channel, and are denoted by d_c (μsec). l (Mbits) is the message length, s (Mbps) is the data transmission rate of the IMP channel, and t (Mbps) is the throughput. The following relationship exists between these elements:

$$d_c = \frac{l}{s} , \quad t = \frac{l}{d_p + d_c} \tag{5.1}$$

Below are some results of our investigations:

Case 1: Varing the message length (l), the maximum throughputs (t) on a single loop were observed (results, Figure 5.3). Applying the above-mentioned formula (5.1), the graph was constructed according to the method of "least squares" wherein the two coefficients d_p and s are adjusted so that the sum of the difference between the computed throughputs (points on the curve) and the observed throughputs (scattered dots nearby the curve) is as minimal as possible, with consideration to all observations as the message length varies, the processing of the

CPU (d_p) and the effective channel speed (s) were deduced to be 1.13 ms and 1.02 Mbps, respectively.

The message delay ($d_p + d_c$) is proportional to the message length (l) (see Figure 5.4).

By summing up the processing routines of messages (Table 5.1), the average d_p was deduced to be 1.126 ms, and if the IMP channel is assumed to behave as stated in section 2.2.2, another value of s — 1.04 Mbps —is obtained. These values coincide well with the ones deduced by the least square method.

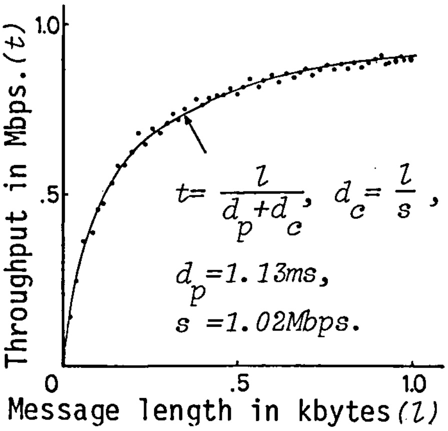


Figure 5.3 Throughput of a single loop.

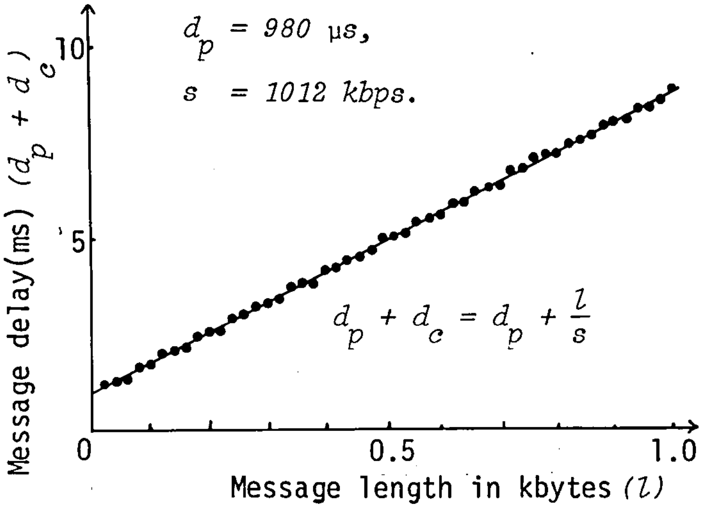


Figure 5.4 Message delay of a single loop.

Table 5.1 Processing delay calculated from amount of cycles contained in each stage.

STAGE	cycle/message
(1) Establishment of transmission path Host(A) → IMP	259
(3) Release of transmission path Host(A) → IMP	365
(5) Establishment of transmission path IMP → Host(B)	268
(7) Release of transmission path IMP → Host(B)	281
Total	1173
1 cycle = 0.96 μsec	1.126 ms/message

Case 2: Two loops, the configuration is *L1* and *L2*, excluding *L3*, shown in Figure 5.2, are formed. When *L1* and *L2* work concurrently, the superiority or inferiority of each of these loops is determined by the given priority of each channel where the loop is inserted. The highest throughput path or loop, in other words, is established through the designation of the order of priority of the pairs of channels (sender and receiver), *L2* > *L1* means the priority of *L2* over *L1*. If no priority is designated, the order of priority is *L1* > *L2* owing to the inherent priority order of the IMP channel.

Figure 5.5 shows a comparison of the maximum throughput of *L2* and *L1* when the message length of *L2*, which has priority, is fixed and the message length of *L1*, which does not have priority, is varied. Two situations, where we changed the fixed length of *L2*, were examined and the resulting pairs of curves are shown in the figure. The longer the length of *L2*, the higher its throughput. The throughput of the priority loop—in this case *L2*—is dependent only on its own message length, almost independent of the message length of *L1*—the secondary loop. Therefore, the path given priority in the IMP maintains high

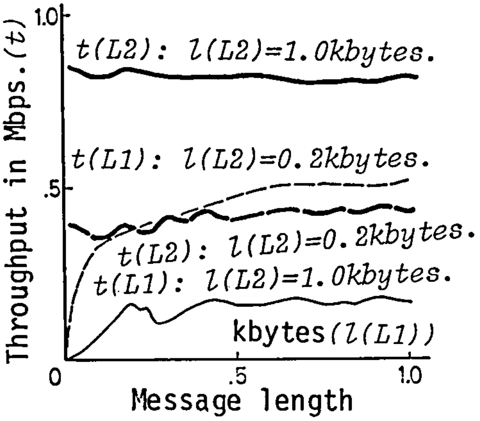


Figure 5.5 Throughputs of two loops; (*L2* > *L1*).

throughput capacity independent of the traffic of the lower priority channels. Our investigations show, furthermore, that the throughput and message length of the priority loop conform to the formula (5.1); in this case d_p amounting to twice that obtained in *Case 1*, because the CPU speed is degraded by the memory cycle-stealing from the channel, which has priority over that from the CPU and which can be carried out concurrently with the CPU processing. Memory cycle-stealing from the channel is regulated by the circuit stated in section 2.2.2 so that it does not consume more than half of the available memory access slots.

Case 3: The configuration is the same as in *Case 2*, where *L1* > *L2* and the length of *L2* is fixed. The throughputs of *L1* and *L2* are plotted against the length of *L1* in Figure 5.6. This figure shows the decrease in throughput of the lower priority loop (*L2*) as the messages in the higher priority loop (*L1*) become longer. The throughput of *L2* is at its worst because *L1* achieves the maximum throughput and consumes the maximum amount of memory cycles, which is not a practical situation. Nevertheless, even in the worst condition, the lower priority loop can achieve more than 140 kbps of throughput.

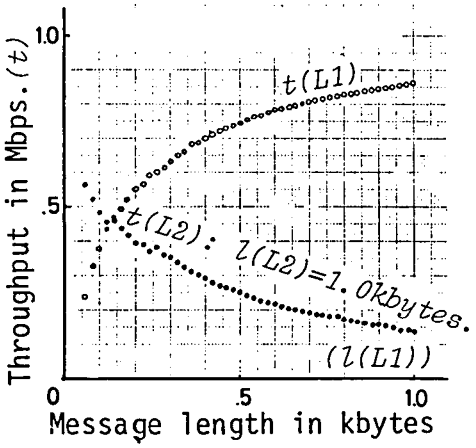


Figure 5.6 Throughputs of two loops; (*L1* > *L2*).

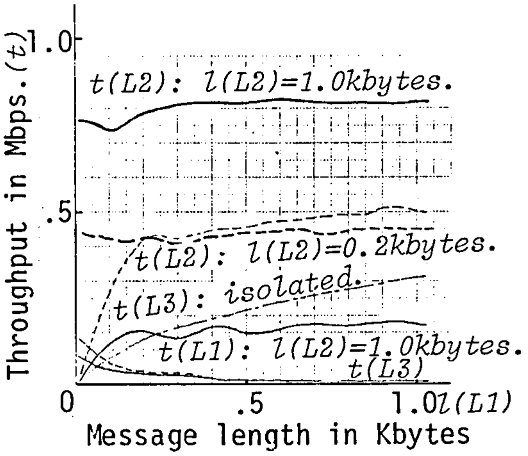


Figure 5.7 Throughputs of two loops (*L1* and *L2*) and a Host connection (*L3*); (*L2* > *L1* > *L3*).

Case 4: Figure 5.7 shows the relation between the throughputs of *L1*, *L2* and *L3* (*L2* > *L1* > *L3*); that is, the configuration in Figure 5.2. *L1* stands exactly in the same relation to *L2* as in *Case 2*, and their throughputs are also almost the same as in *Case 2*. The only difference is that loop *L3* generates extra messages to and from a Host. Moreover, the extra *L3* traffic can be handled by the IMP concurrently, though its volume is very small (about 10 kbps).

Case 5: Figure 5.8 shows the relation between the throughputs of *L1* and *L2* as plotted against their lengths in the case where they both are of equal length. The difference between the two curves shows the effect of channel priority. Besides this, the sum of these throughputs is also shown. This sum is limited to 1 Mbps due to the hardware constraint introduced in section 2.2.2 where the total capacity of the IMP channel is limited to 2 Mbps, which is divided into two equal parts for sending and

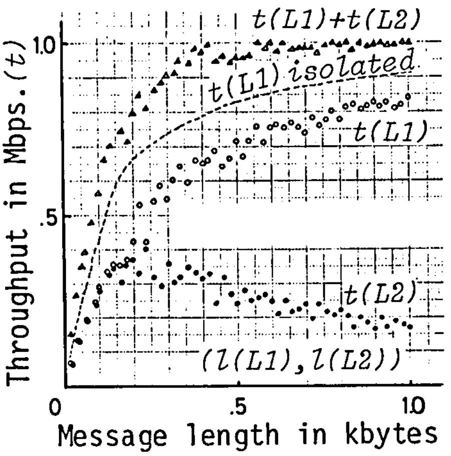


Figure 5.8 Throughputs of two loops; (*l(L1)* = *l(L2)*, *L1* > *L2*).

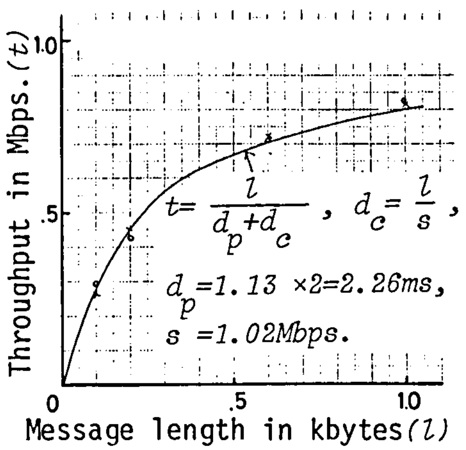


Figure 5.9 Throughput of the highest priority loop (*L2*).
O *L2* > *L1*.
× *L2* > *L1* > *L3*.

receiving channels—hence corresponding to 1 Mbps of throughput.

As can be seen in *Cases 2* and *4*, the path given priority in the IMP maintains a high throughput capacity independent of the traffic of lower priority channels. Figure 5.9 shows the throughput of a priority loop and its length of message. From the results in *Cases 2* and *4*, the throughputs of the priority loops and their lengths of messages are obtained. The curve in the figure is a computed result according to the formula (5.1) where the processing delay d_p is doubled because the execution speed of the CPU is degraded by 50% due to channel interference, doubling the estimated d_p as compared to that obtained in *Case 1*. This estimation proves accurate.

Case 6: Figure 5.10 depicts the situation wherein *L1* and *L2* are alternatively given priority; that is, both paths take equal probability in priority of access to the main storage on

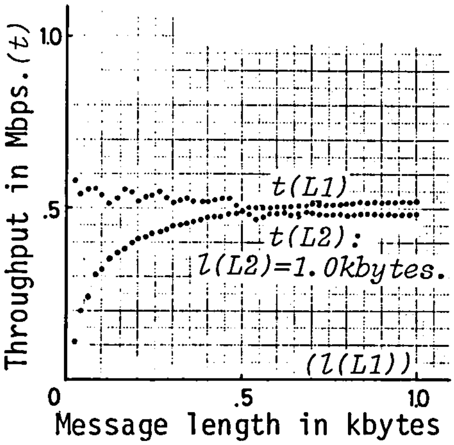


Figure 5.10 Throughputs of two loops of equal priority.

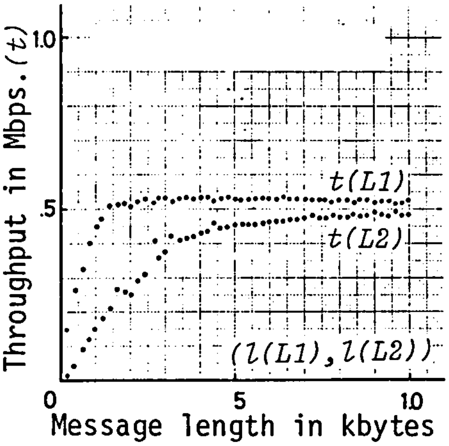
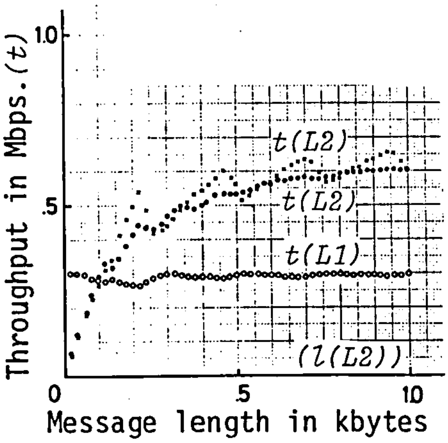


Figure 5.11 Throughputs of two loops of equal priority; (*l(L1)* = *l(L2)*).

the hardware level. Alternation of priority is performed by a task which is independent from the message-switching task. The programmable priority control circuit can change the priority order between channels at any time. Since the frequency of the alternation is very high—almost 10^5 times per second—every channel seems to have exactly the same priority. Figure 5.10 corresponds to Figure 5.5. Both *L1* and *L2* have limited maximum throughputs of 0.5 Mbps; that is, the capacity of the IMP channel is divided into equal parts. However, there do remain effects due to the order of interruption priority. Figure 5.11, in correspondence with Figure 5.8, reveals these effects.

Case 7: Measurements have been performed with fixed message length. However, the way in which the message lengths are distributed may change the characteristics of the throughputs. Following the usual assumption, exponential distribution



× *t(L2)*; with fixed 50 bytes of message length of *L1*.
● *t(L2)*; with random message length of *L1*.
○ *t(L1)*; with exponentially distributed random message length (mean = 50 bytes).

Figure 5.12 Throughputs of two loops; with exponentially distributed random message length (*L1*).

was examined. The test data was generated by the sequence of random numbers stored in the IMP memory of 8192 words. Only *L1* was adopted in this random sequence. The result was almost the same as that of the fixed message length which is equal to the average of the exponential distributed random number sequence (Figure 5.12).

In a practical situation, a Host dispatches messages discontinuously or burstly, so it is natural to expect that real-prevalant traffic due to the Host's messages does not cause such a heavy load on the IMP as that of the above experiments. Usually, messages are dispatched and received *via* NCPs in the Hosts, this introducing the processing delay in NCPs. If processing includes data transfer between external storage, such as between a disk and a computer, the service time of the disk accounts for a great deal of processing delay. Owing to these factors, from our experience the total maximum traffic dispatched or consumed by a Host is restricted to about 640 kbps—the maximum throughput from MELCOM to NEAC. (See section 5.3)

Suppose that interactive processing is predominant in the network, and the expected average message length dispatched from a Host is 40 bytes. In this case, the throughput of the IMP obtained from the formula (5.1) is 220 kbps. The traffic should be within 50% of full capacity, allowing a safety factor for inaccuracies in the message length data. The total traffic dispatched from all Hosts is restricted to less than 110 kbps. Dividing this capacity by 10 kbps for each Host, the maximum number of Hosts connectable to a single IMP would be eleven.

5.3 Measurement of Network Throughput

Host computers communicate with each other through the IMP, and the concurrent throughput is defined here as the maximum amount of bits from one Host to another Host which passes the IMP in a second.

Maximum throughput was measured between arbitrarily chosen Host pairs where the message length was variable. The flow control mechanism for high-speed transfer was adopted. The transmission path was filled up with messages—pipe lining (see section 3.2.3). One Host sent messages as fast as possible, the partner Host receiving them as fast as possible. Messages were generated by a user process, sent through an NCP to the network, and received *via* another NCP and another user process. Hence, transmission delay included the processing delay in the NCPs at both ends.

Table 5.2 Measured message delay in KUIPNET.

Path	Processing delay	Channel capacity
MACC → MELCOM	K = 1491 μsec	C = 799 kbps
NEAC → MELCOM	2977	432
MACC → NEAC	2623	831
MELCOM → NEAC	2372	821

From the observed throughput we were able to obtain figures on the average interval from the arrival of one message to the arrival of the next. Message delay is proportional to the length of the messages and is represented by the formula:

$$D = K + \frac{1}{\mu C} \qquad (5.2)$$

D (sec) is the message delay, K (sec) is the processing delay, $\frac{1}{\mu}$ (bit) is the message length, and C (bps) is the channel capacity.

The message delay for each tested combination of Hosts is shown in Table 5.2.

The maximum throughputs between Hosts were: MACC → MELCOM 682 kbps (8000 bits/message); NEAC → MELCOM 374 kbps (7760 bits/message); MACC → NEAC 632 kbps (8064 bits/message); and MELCOM → NEAC 646 kbps (8064 bits/message) respectively.

5.4 Measurement of Network Traffic

The IMP of KUIPNET is designed to handle high-speed in-house communication. Throughput was measured on the IMP itself, and in this part the measurement results of the traffic, in the operating state, which passes through the IMP shall be discussed.

5.4.1 Classification of Traffic

In KUIPNET, traffic is sorted according to their characteristics, as shown in Table 5.3.

The first type of traffic is real-time transmission of data that appears in speech data processing by computers, and which is to be stored into computer's files for further processing. The network allows a message throughput of 200 kbps (this rate corresponds to the speech data sampled every 20 kHz, 10 bits for each sampled value), though measurements show the maximum message throughput to be 400 kbps from MELCOM to NEAC. The key objective of this real-time speech transmission is not to keep the average

Table 5.3 Classification of network traffic.

Type	Content	Message length	Number of messages
Real-time data transfer	Speech	485 word (16bits)	110
Large volume data transfer	Picture	134	6
File transfer	Program	95	33
Character strings	Command	9	1

message throughput within the range of the data rate, but to maintain a continuous flow with constant delay without gaps or discontinuities.

There are two other types of real-time speech data: The first is a time-sequence of sound spectra obtained through a 20-channel filter bank, the element of which is an amplitude of 10 bits at 10 ms sampling intervals (*i.e.* 20 kbps). The second is a sequence of zero-crossing wave elements, which is represented by characters of variable lengths. Its average data rate was proved by Tomita[1973] to be 9 kbps. These two representations of real-time speech data, however, are compressed forms of speech data and demand less rigid real-time transmission.

The second type of traffic is that of a large volume of digital pictures or photographs which is preferably transferred over the high-way in the network, though it may be treated as a kind of background traffic. For example, the total amount of bits of a photo is 256 x 256 x 5 in monochrome, three times larger in color. Requirements of a much higher resolution show a quadratic increasing rate of the amount of bits transferred. Such a case would arise in interactive processing or designing with computer aids.

As was stated in section 5.2, the longer the length of a message becomes, the higher the throughput which is achieved. However, user process implementers are apt to utilize simpler methods of manipulating data, since blocking and deblocking processes, introduced for the sake of improvement of transmission efficiency, are rather time and space consuming. Still more, some user programmers often feel a dislike for elaboration on programming. For instance, the message length of picture data is 134 words (Table 5.3) which corresponds to a unit of picture data, *e.g.*, a horizontal line in TV scanning mode. Sometimes the size of a unit of a logical element of pictorial data is determined from the record size or buffer size of the auxiliary storage (*e.g.*, a disk or a tape) in order to avoid packing/unpacking here again.

The third type of traffic is transfer of files between computers. The type of files transferred are programs of minicomputers and magnetic tape files. Two distinguishing points of file transfer are: a) the efficiency of communication path or throughput—that is, the amount of bits sent per second should be large, and b) the reliability of the transferred data—that is, all bits in the file are transferred exactly, without error, from one computer to another remote computer. In many cases the latter is a more important feature than the former. File transfer is required for many purposes, storage sharing and others. Minicomputers which are not equipped with auxiliary files share random access files equipped around a large Host computer as if those files were their local file. (See section 3.2.5) NEAC 2200 performs as the file computer in KUIPNET.

Another case where file transfer is utilized is to facilitate medium conversion. For example, a 9-track magnetic tape is transferred through the network to a Host which is equipped with a 7-track megnetic tape in order to convert the storage medium. (See section 3.5) From our own experience we know that information transfer through the network is more convenient than that by transportation of recording mediums, such as a magnetic tape or a paper tape, because the former is faster and more reliable than the latter. Medium transportation is time-consuming; and still more, the magnetic tape drive units of different manufacturers are sometimes incompatible even though they are of the same specifications—that is, the same number of tracks and the same recording bit-density. A magnetic tape written by one computer often causes difficulty for another computer to read.

The fourth type of traffic is transfer of characters which are produced by TELNET commands or of characters of natural sentences. These data are intermittent and of short length, and are typed *via* keyboard. For such traffic the key feature is transmission delay in the network. One of the main specifications of ARPANET was to satisfy the condition that the maximum message delay in the network would be less than 0.5 second. It is remarkable that ARPANET achieved the above condition perfectly for messages which pass over several node computers (IMPs) connected *via* communication lines of 50 kbps. But this specification is easy to accomplish for us. Because KUIPNET is an in-house computer network, a) a message passes only one node computer which is the center of a star type network, and b) communication lines which connect Host computers are high-speed (1.6 Mbps—about 30 times faster than that of ARPANET).

5.4.2 Traffic Measurement Tool

Traffic is measured during daily usage of the network by a built-in program of the IMP. The statistics program in the IMP calculates the traffic matrix of every 5 minutes and prints it on a teletypewriter. The status of the traffic in the network is monitored immediately and directly by the IMP.

The quantity and lengths of the messages which pass the IMP are measured, and these messages are accumulated in columns corresponding to their source and destination Hosts. This measurement is performed in a sub-routine which is built in the IMP's message-switching program. After integration during a fixed period, the results are processed by the statistics program, which is executed as a background program. There the average message length, total message count, and average traffic density are calculated, these measurements printed out on the teletype-

Table 5.4 Programs used for traffic measurement.

Routine	Size	Processing time	
Data gathering	63 words	90 μsec	per message
Statistics	5.5K words	2 sec	5 minute intervals
Output		3 min	100 baud
Message switching	4.5K words	980 μsec	per message

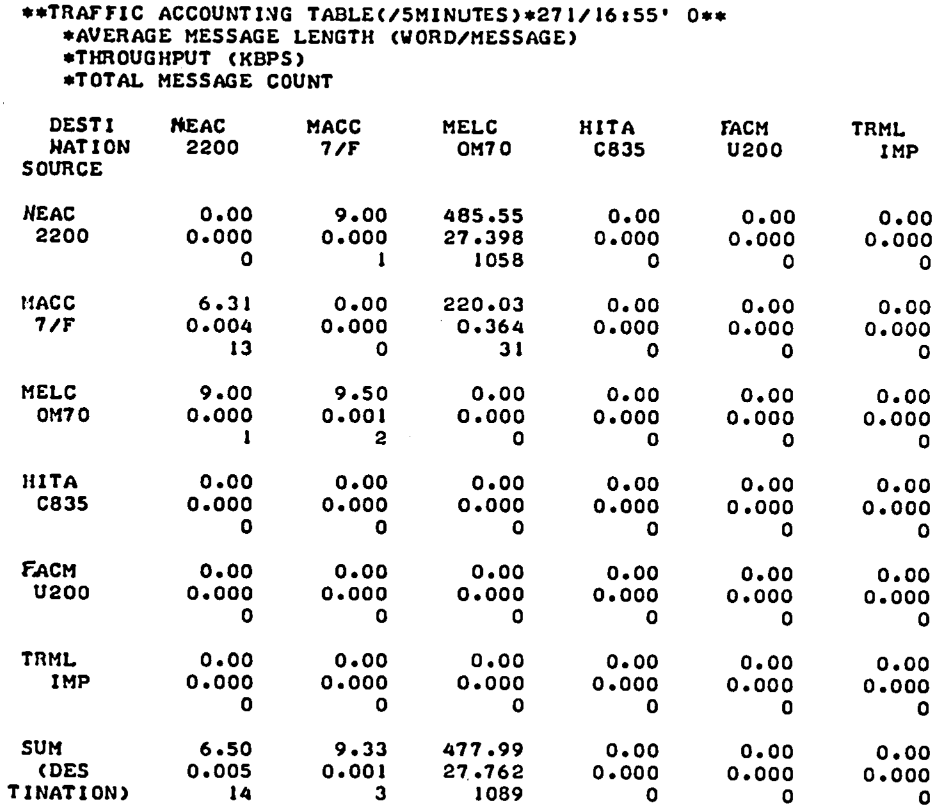
writer of the IMP. This statistics calculation is set to 5 minutes in order to meet the printing speed which takes about 3 minutes.

Table 5.4 shows a list of programs which are related to traffic measurement. The data-gathering routine that watches individual messages coming into the IMP and accumulates message dimensions, coded in assembler language, occupies 63 words. The processing time of this routine is 90 μsec per message.

The statistics routine, coded in FORTRAN, occupies 5.5K words. Its processing time is 2 seconds/traffic-matrix, and it takes about 3 minutes to print a traffic matrix with a teletypewriter of 100 baud.

The data-gathering routine may cause a small disturbance in the IMP's switching characteristics. It takes about 980 μsec for a message to pass the IMP. The data-gathering routine adds about 10% extra delay to the switching delay of the CPU, excluding the delay by channel of 1 μsec/bit. Hence, degradation on message throughput is small.

Figure 5.13 shows an example of output from the statistics



```
**TRAFFIC ACCOUNTING TABLE(/5MINUTES)*271/16:55' 0**
   *AVERAGE MESSAGE LENGTH (WORD/MESSAGE)
   *THROUGHPUT (KBPS)
   *TOTAL MESSAGE COUNT

  DESTI    NEAC      MACC      MELC      HITA      FACM      TRML
  NATION   2200      7/F       OM70      C835      U200       IMP
SOURCE

NEAC         0.00      9.00    485.55      0.00      0.00      0.00
 2200       0.000     0.000    27.398     0.000     0.000     0.000
                0         1      1058         0         0         0

MACC         6.31      0.00    220.03      0.00      0.00      0.00
 7/F        0.004     0.000     0.364     0.000     0.000     0.000
               13         0        31         0         0         0

MELC         9.00      9.50      0.00      0.00      0.00      0.00
 OM70       0.000     0.001     0.000     0.000     0.000     0.000
                1         2         0         0         0         0

HITA         0.00      0.00      0.00      0.00      0.00      0.00
 C835       0.000     0.000     0.000     0.000     0.000     0.000
                0         0         0         0         0         0

FACM         0.00      0.00      0.00      0.00      0.00      0.00
 U200       0.000     0.000     0.000     0.000     0.000     0.000
                0         0         0         0         0         0

TRML         0.00      0.00      0.00      0.00      0.00      0.00
  IMP       0.000     0.000     0.000     0.000     0.000     0.000
                0         0         0         0         0         0

SUM          6.50      9.33    477.99      0.00      0.00      0.00
 (DES       0.005     0.001    27.762     0.000     0.000     0.000
TINATION)      14         3      1089         0         0         0
```

Figure 5.13 Example of traffic matrix.

program: the average message length in words (16 bit/word), throughput (kbps), and total message count that were accumulated within 5 minutes. A cell in the *i*-th row, *j*-th column represents the traffic sent from the *i*-th Host to the *j*-th Host.

5.4.3 Example of Observed Result

The traffic and time of execution of three kinds of network jobs are observed: (a) color-enhanced display of on-line signature, (b) color display of speech spectrum, and (c) speech synthesis by zero-crossing wave elements. In job (a), speech response is used. Speech data converted into digital form is sent through the network in real-time and continuously at a uniform rate of 120 kbps. In job (b), the speech sound spectrum is analyzed by a 20 channel filter bank, sampled every 10 ms, blocked into a message of 100 ms samples, and then sent to the main Host to be stored into a file. This takes place in real-time continuously for 10 seconds, after which designated sections are interactively retrieved to be sent to the other Host to be displayed in a color-enhanced spectrum on the color TV. In job (c), a "kana" letter sentence of natural Japanese, fed from a keyboard, is synthesized from zero-crossing wave elements. Again, speech is reproduced in real-time.

Besides these messages a small number of control commands are exchanged between Hosts to establish connections.

Figure 5.14 shows a sample of one hour's traffic density. Traffic in the network is greatly affected by the types of jobs using the network—the message count and amount of bits transferred. Figure 5.15 shows the distribution of message length. Speech and pictorial data are almost a hundred times larger in numbers, and also in length, than character data. These are distinctive features of the raw data traffic which appears in the information processing of KUIPNET; quite distinct from other computer networks.

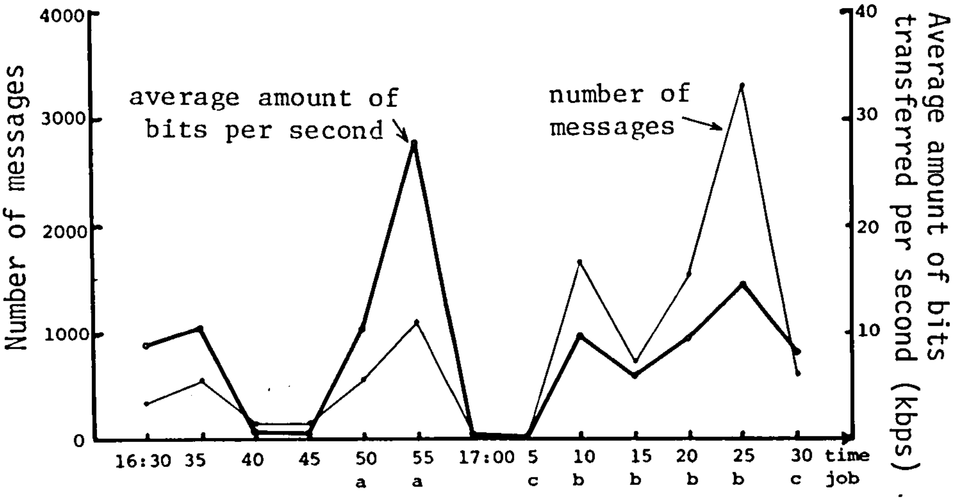


Figure 5.14 Number of messages and amount of bits transferred in each 5 minutes.

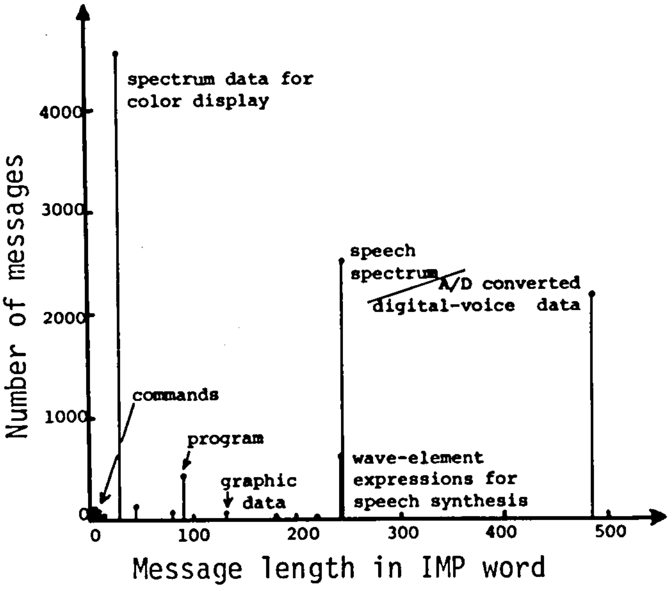


Figure 5.15 Distribution of message length.

5.5 Network Measurement System

The measurement system developed for KUIPNET gathers several parameters in the daily usage of the network. The following three points are the main purposes of this measurement system: To know (1) the status of utilization of the resources in the actually operating switching computer; (2) the characteristics of the traffic in the network; and (3) the status of utilization of the network.

5.5.1 Parameters Dealt with in the Measurement System

Considering the purposes stated above, we employed the following parameters which are observable in the IMP:

(1) Status of the IMP — The following parameters in the IMP are collected every minute:

- Store-and-forward buffer utilization — There are four counters; one which counts the buffers from the free list, one for the buffers returned to the free list (from these two counters the number of messages in the IMP can be estimated), a counter to identify the number of times a free buffer is unavailable, and a counter of the pendency of transfer requests from a Host due to lack of a receive buffer.
- IMP-Host channel conflicts — There are two counters; one to measure the number of conflicts on the IMP-Host half-duplex line, and a counter of the extra output operations executed due to these conflicts.

- Idle time of the CPU — The background loop counts the number of times it is executed, sampled every 25 ms, and makes a histogram of CPU utilization during the last one minute. This histogram supposedly reveals momentary high overload, although the values represented in the histogram are actually the averages of the loads during time spaces of 25 ms.

(2) *Traffic measurement* — With regard to every source and destination Host pair, the length and count of every message passing through the IMP are measured. The result is a traffic matrix which shows the total amount of messages and the total amount of bits which passed the IMP from one Host to another Host. The columns of the traffic matrix, each composed of two 32-bit counters (*i.e.,* four 16-bit words), accumulate and thus count the number of words in a message and the number of messages. This data is calculated by a sub-routine of double precision addition. A traffic matrix is sampled every minute, there being two traffic tables used in turn for a minute each. The size of a table is 4(words) x 8 x 8 x 2 = 256(words) x 2 = 512(words). After a minute, a background task is triggered to write the table just used into a magnetic tape. The magnetic tape is operated by the background program.

(3) *Status of network usage* — For what jobs the network is used is an interesting problem. This may be known from the users' accounting information, usually performed by individual Hosts. Another way is to ask a user to report his usage of the network. Here we adopted a new method which monitors the Host-Host protocol performed by every process in a Host. This observation shows the way a user uses the Host-Host protocol; the user messages sent over the connection during a session, from the establishment of a connection through the closing of it. Information concerning the Host-Host protocol is obtained from messages which pass

the IMP over the control link (line# 0). Those show the control between NCPs; the connection status between sockets (processes), buffer allocation, interrupt, error messages, reset and so on. Except for "allocation" commands, usually such data are not so large as compared to regular messages—for example, the establishment and closing of a connection is simply represented by the sequence RTS, STR, CLS, CLS. Control messages which have entered the IMP are stored and copied into a magnetic tape for further retrieval. The status of network usage may be investigated later by retrieving this tape in the off-line environment. ALLs (allocation) are not stored, since a great many—almost the same quantity as regular messages on the user level—are generated.

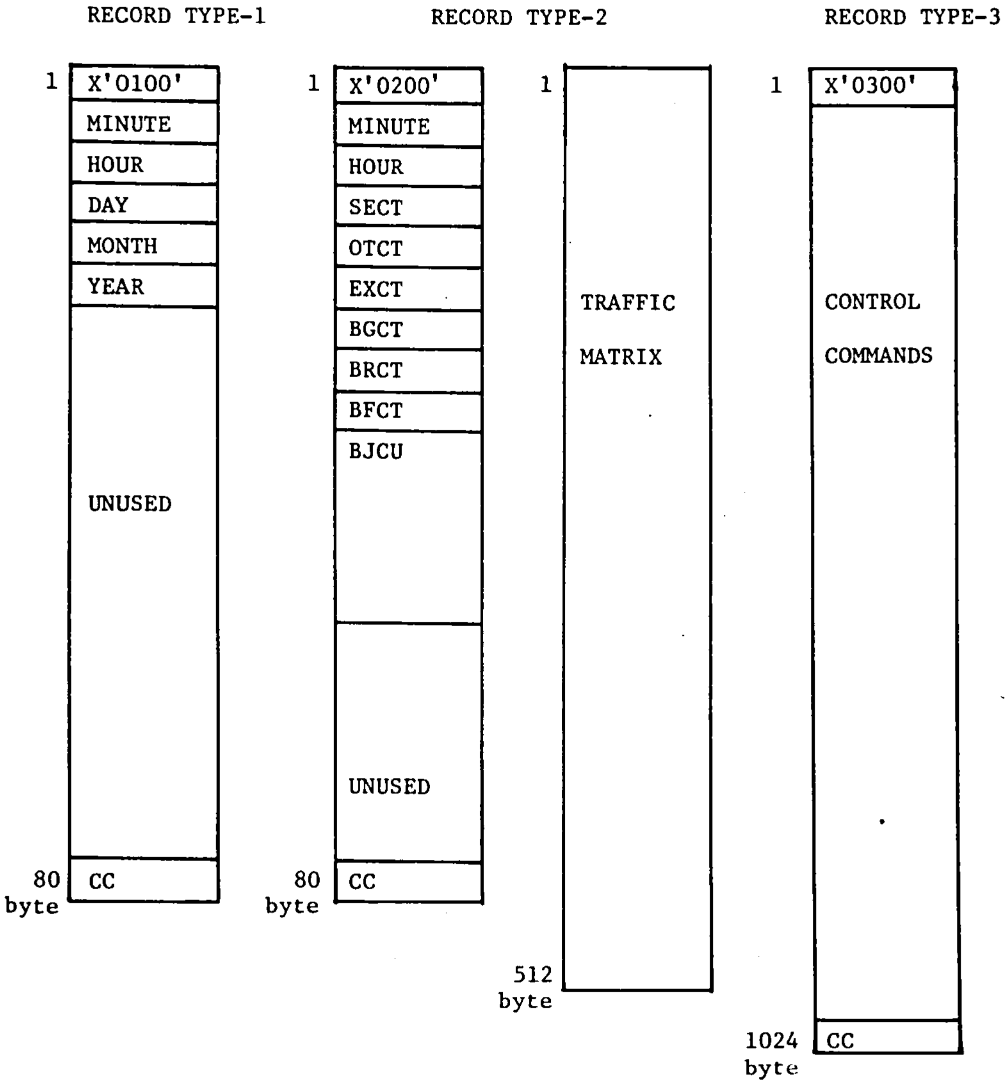
5.5.2 Log-File Record Format

All data are stored in a magnetic tape attached to the IMP. It is a 1/2-inch, 800 bpi, 1200-foot, 9-track tape, the transfer rate being 30 kilocharacter/sec. The magnetic tape contains two files. The first file stores a copy of the entire IMP program codes on the IMP main memory that amounts to 32,768 words (a word = 16 bits). As was shown in Figure 2.8, the last 1,024 words of the IMP main memory contains the loader which loads the IMP program from the magnetic tape into the IMP memory. A copy of the IMP program, amounting to 31,680 words (the loader and other miscellaneous codes account for the difference of 1,088 words) is recorded in the file with 125 records of 512 byte length. In order to assure the program loading between the tape and the main memory, a check word is attached at the end of all words of the copy. The check word is calculated by iterative exclusive-or operations of all words contained in the copy. It is verified at the time of loading. A tape mark ends this file. The second file is a collection of measurement data. The records in this file are composed of three intermixed types of records described below, the lengths of which are undefined as long as they are less than 1,024 bytes. A tapemark also ends this file. For maintenance of the recorded tape, a double tapemark may be added at the end of all files.

In the initialization process the IMP program sets the tape head to the point at the end of the last data. New data gathered by the IMP is then recorded after this point.

Measurement data composes one file, which we call a "log-file". A log-file is composed of several "run-files". All records produced from the start through the stop of the IMP are sequentially related in one run-file, but the records in the different run-files are disconnected. The first input in a run-file is the date and time when the IMP was started (type-1 record), which is given by an operator from a keyboard, and this record is what separates each run-file. Following the type-1 record, records regarding the CPU status and traffic—written every minute—(type-2), and control commands related to the Host-Host protocol (type-3) are entered. Two buffers, each 1,024 bytes, are prepared for the type-3 record; one to store arriving commands and one as an output buffer, used in turn. Type-3 records are written into a tape when the buffer becomes full or 30 seconds has past from the time of the last-stored command. The type-2 and type-3 records are intermixed and written sequentially in the order they are produced. Figure 5.16 shows the formats of these records.

When an operator stops the IMP, no closing operation is necessary. The log-file remains in its present position, since the initiation of the IMP program can find and set the log-file



SECT: number of "receive" requests sent to bufferless channel.
OTCT: number of conflicts on IMP-Host half-duplex lines.
EXCT: number of extra output operations executed due to conflicts.
BGCT: number of buffers obtained from free list.
BRCT: number of buffers returned to free list.
BFCT: number of times free buffer list is empty.
BJCU: histogram of IMP utilization.
CC: continuation code.

Figure 5.16 Record formats in log-file.

at the end of the last record by computing the continuation codes in the successive records. The initiation process of a log-file takes only a few minutes and is accurate (see next section). But it would be unreliable to request an operator to close a log-file.

5.5.3 Initiation of Log-File

A log-file is written in sequential mode so that old data is protected and the newest record is the last one in the file. In the initiation process of the IMP program, the log-file is forwarded to the end of the last record and positioned for a new record. This process was perfected with the following principles in mind: (1) old records should be protected, (2) searching should be completed quickly, (3) the file should be kept consistent. All records in a log-file are accompanied with a 16-bit continuation code. The continuation code is generated by a feedback shift-register simulated by software. For the first record of a log-file, a continuation code is calculated using the initial value of zero. The continuation code of the previous record is used as the initial value of the next computation of continuation code. Consequently, all records in a log-file are related with each other through this sequence of continuation codes. The log-file initiation process is performed through computation of the sequence of continuation codes, and it is completed by the first incorrect continuation code or read error.

5.5.4 Space-Saving in Log-File

In order to use file space efficiently, logging is suppressed when there is no traffic. If a traffic matrix is all zero, then the type-2 record is skipped. This rule greatly reduces the total amount of records written at intermittent times.

5.5.5 Results and Considerations

(1) CPU utilization in idle time:

Idle time is defined as the time when there is no traffic in the network. During such times, the IMP program executes only timer and background routines which contain the following functions: (a) Timer-task triggered by an interval-timer interruption. This task updates current time and supervises IMP channel lock-up, buffer allocation, and timeout functions including statistics. (b) Communication control of 4,800 bps line. When there is no traffic, empty frames are exchanged to test line breakage (see section 4.3). (c) TIP supervisory routine. This routine, carried out once every 25 ms, supervises the status of input/output completion from teletype, inputs from the network, and occurences of events (see section 3.4). (d) Pseudo-timeout. Pseudo-timeout is supervised in the background routine. When timeout happens, a modification is made in the event table to cause timeout interruption (see section 2.3.3). (e) Magnetic tape output. All records, which are formated in the measurement routine, are written by this routine. This routine is executed by a sub-routine call from the background loop.

The results of measurements of CPU utilization during idle time is shown in Table 5.5. CPU utilization, as the table

Table 5.5 CPU utilization in idle time.

Timer routine	4.56%
Communication control	0.04%
Terminal status sensed in TIP	15.38%
Pseudo-timer routine	0.82%
Total	20.80%

denotes, is 20.8% when there is no traffic. This is subdivided into four parts: (1) timer routine—4.56%, (2) communication controller—0.04%, (3) TIP—15.38%, and (4) pseudo-timeout—0.82%. Since the execution priority of (1) and (2) are higher than that of the message-switching task, CPU resources of this part are used independent of traffic density. Tasks (3) and (4) are in background level, so CPU resources in these are overridden by the message-switching task and become smaller with the traffic.

(2) IMP utilization and traffic:

IMP utilization was sampled at 25 ms intervals during one minute, and a histogram was made. The results give us the frequency distribution of IMP utilization in relation to traffic of 280 kbps which passes the IMP during one minute, as compared with the cummulative frequency distribution of IMP utilization when there is no traffic (Figure 5.17). The former case (solid line in Figure 5.17) shows about 20% of IMP utilization.

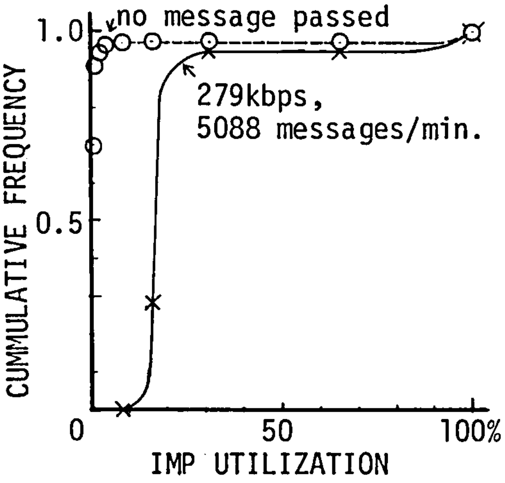


Figure 5.17 Cummulative distribution of IMP utilization.

(3) Traffic matrix:

Table 5.6 is a traffic matrix of all messages which passed the IMP since April 19, 1976 through July 16, 1976. In each column, the upper is the total bits contained in the messages and the lower is the total number of messages. As is seen from the table, the most active pair of Hosts has been NEAC 2200 and MELCOM 70. They transfer digital speech wave forms in real-time. Note that they used long message size, *i.e.*, 1642-bits from NEAC to MELCOM, and 700-bits from MELCOM to NEAC. Self-looping was performed by NEAC 2200 and TIP. Somewhat heavy traffic was observed between NEAC 2200 and PANAFACOM U-300. This was artificial traffic generated for the measurement of throughput. TIP has frequently been connected with TELNET in NEAC 2200. However, the character-oriented traffic over this connection has been small. TOSBAC 5600 in Osaka was used only slightly. HITAC 8350 and FACOM U-200 were inactive during this period.

Table 5.6 Network traffic summary.

(number of bits which passed the IMP)
(number of messages which passed the IMP)

from APR. 19, 1976 through JUL. 16, 1976.

DESTINATION SOURCE HOST	NEAC 2200	PF-U300	MELCOM-70	HITAC8350	FACOM-U200	TIP	TOSBAC5600
NEAC 2200	91143280 82501	10037920 1736	285216960 173737	0 0	0 0	84672 677	1280 12
PF-U300	6337456 1728	0 0	0 0	0 0	0 0	0 0	0 0
MELCOM-70	291952528 416772	0 0	0 0	0 0	0 0	0 0	0 0
HITAC8350	0 0	0 0	0 0	0 0	0 0	0 0	0 0
FACOM-U200	0 0	0 0	0 0	0 0	0 0	0 0	0 0
TIP	60384 498	160 1	0 0	0 0	0 0	53232 536	160 1
TOSBAC5600	768 8	0 0	0 0	0 0	0 0	0 0	0 0

MACC 7/F was disconnected from KUIPNET and replaced by PF-U300, which was under experiment for a gateway to the inter-university network.

(4) Status of network usage:

Control commands related to the Host-Host protocol can be observed from the type-3 records in the files. Here are some descriptive features of this type record: (a) The majority of observed commands are related to connection establishment between processes that are implemented on the Hosts in the network. (b) These data are very useful to trace NCP-level communication (Host-Host protocol) and to detect misinterpretation of the protocol. Several extraordinal oscillations of connection requests owing to mismatching of two related processes were recorded. (c) These records contain wealthy and interesting information about the behavior of the IMP, NCPs, and user processes; *e.g.,* the time when messages entered the subnet, the responsiveness of NCPs, identification (by socket number) of establishment, duration and breakage of connections.

Statistical considerations on this data should give us a deeper insight into this network. Unfortunately, because the measurement system began to accumulate data just recently and because network usage is not yet very high, the amount of collected data is too small to be applied to some of the statistical processes.

Sometimes these log-files are removed from the IMP and merged in another tape in order to protect them from accidental destruction. A program is being developed to retrieve these merged log-files.

CHAPTER VI: CONCLUSIONS

In this thesis I have described the development of a high-speed communication network, called KUIPNET, utilized for information processing by the cooperation of on-site computers.

In Chapter II the design of the message switching computer, having the following high-speed transmission properties, is explained:

(1) Maximum throughput set to 1.0 Mbps.

(2) Switching computer guaranteeing one stream of high-speed transmission concurrently with several other streams of low-speed transmission.

(3) Standardized computer connection—single and asynchronous transmission, and half-duplex control. The maximum transmission rate achieved is 1.6 Mbps.

The message switching computer has achieved high throughput by regulating the memory access conflicts between input/output channels and CPU in the following ways:

(1) The priority of the channels has been made programmable, high priority being assigned when and where high throughput is necessary.

(2) The rate of memory-cycle stealing has been restricted in order to guarantee the real-time communication control maintained by the CPU.

Performance of the switching computer is measured in Chapter

V, the summarized results being:

(1) Maximum message throughput = about 900 kbps.
(2) Message throughput of highest priority path = about 800 kbps under heavy traffic load.
(3) Message throughput of second priority path = about 160 kbps under heavy traffic load.

These throughput characteristics of KUIPNET are important for achieving real-time transfer, for such transfer brings a heavy load into the switching computer. The maximum throughput of 900 kbps satisfies the bandwidth required for real-time transmission of high fidelity speech wave forms, which require more than 200 kbps continuous transmission rate.

In Chapter III, I have described the interprocess communication mechanism. The functions for interprocess communication control are composed of (1) communication control between a computer system and communication network, (2) communication control between computer systems—an integral part of the monitor, and (3) communication control between processes.

The efficiency of implementation, both in the size and in processing overheads, are important factors. The selective communication functions are implemented in small systems—like minicomputers. The monitors for minicomputers are designed efficiently to achieve the desired properties described below:

(1) preemptive processing of high priority tasks—real-time tasks, controlled with multi-programming mechanism.
(2) extensible input/output functions and user programmable input/output functions.
(3) interactive console functions with operators.
(4) interfacing to the network.

I have shown several cases of interprocess communication: a case of process cooperation supervised by the network-wide monitor; a case of implementation of a terminal supporting function for direct entry into the network; and a file transfer procedure designed for conversion between two different types of magnetic tapes.

In Chapter IV the in-house computer network is extended to a distant large computer system. A reliable communication control mechanism over a distance of 50 km, and 4,800 bps line was adopted. The procedure of the basic communication control was modified from that of character-oriented computer-terminal communication (*i.e.*, the computer being master and the terminals slaves,) into that of character code transparent computer-computer communication (*i.e.*, either side of both ends can be master). The time-sharing system in the distant system becomes accessible from the User TELNET in the KUIPNET through the Server TELNET in the switching computer. In the explanation, the effective message throughput is estimated and measured over this connection.

In Chapter V the performance of the switching computer is measured. From the results of the measurements it may be concluded that the message switching computer guarantees one stream of high-speed transmission concurrently with several other streams of low-speed transmission.

The throughputs between Host computers, which are limited by the rate of the transmission path and delay in controlling the input/output in the Hosts, are also measured. The maximum throughput accomplished is 700 kbps.

We have developed a network measurement system which measures and records the following parameters:

(1) Utilization of processor of the switching computer—processor utilization under idle state is 20.8%, while under heaviest traffic ever observed (280 kbps) it is about 30%.

(2) The matrix of traffic from source to destination. Descriptions show the actual utilization of Host computers under network modes.

(3) Interprocess connections and disconnections.

In summary then,

1. A new application of computer network technology to the information processing research system has been accomplished in this network. A high-speed communication facility has been constructed connecting in-house computer systems, which allows transmission of raw data such as speech and picture.
2. On not-so-large systems such as minicomputers, economical interprocess communication facilities have been implemented. Among several computer systems, process cooperating functions and general terminal supporting functions have been implemented.
3. The in-house computer network is not necessarily confined within a small area but can be extended to remote computer systems or large networks. In this thesis I have shown the extension of KUIPNET, which necessitates the control of a long communication line, and a large computer system which greatly enhanced the value of the total resources available in this network was appreciated.

REFERENCES

Abramson, N.[1973]: "Packet Switching with Satellites", *AFIPS NCC*, Vol.42, pp.695-702.

BBN[1974]: "Specification for the Interconnection of a Host and an IMP", *Bolt Beranek and Newman Inc. Report No.1822.*

Carr, C.S., Crocker, S.D., and Cerf, V.G.[1970]: "HOST-HOST Communication Protocol in the ARPA Network", *AFIPS SJCC*, Vol.36, pp.589-597.

Cerf, V.G.[1976]: "ARPA Internetwork Protocols Project Status Report", for the period November 15, 1975 - February 15, 1976.

Cole, G.D.[1972]: "Performance Measurements on the ARPA Computer Network", *IEEE Trans.*, COM-20, No.3, pp.630-636.

Crocker, S.D., Heafner, J.F., Metcalfe, R.M., and Postel, J.B. [1972]: "Function-Oriented Protocols for the ARPA Computer Network", *AFIPS SJCC*, Vol.40, pp.271-279.

Crowther, W.R., Heart, F.E., McKenzie, A.A., McQuillan, J.M., and Walden, D.C.[1975]: "Issues in Packet Switching Network Design", *AFIPS NCC*, Vol.44, pp.161-175.

Farber, D.J.[1972]: "Networks: An Introduction", *DATAMATION*, Vol. 18, No.4, pp.36-39.

Forgie, J.W.[1975]: "Speech Transmission in Packet-Switched Store-and-Forward Networks", *AFIPS NCC*, Vol.44, pp.137-142.

Hayashi, T.[1976]: "NEAC 2200/250 NOS Operating System—Design Concepts, Functions, and Operations", *KUIPNET Manual-2*, (in Japanese).

Hayashi, T.[1977]: "NOS Operating System: An Operating System for a Host Computer in a Computer Network", (Doctral Thesis, in preparation), Kyoto University.

Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R., and Walden, D.C.[1970]: "The Interface Message Processor for the ARPA Computer Network", *AFIPS SJCC*, Vol.36, pp.551-567.

Heart, F.E., Ornstein, S.M., Crowther, W.R., and Barker, W.B. [1973]: "A New Minicomputer/Multiprocessor for the ARPA Network", *AFIPS NCC*, Vol.42, pp.529-537.

Kahn, R.E., and Crowther, W.R.[1972]: "Flow Control in a Resource-Sharing Computer Network", *IEEE Trans.*, COM-20, No.3, pp.539-546.

Kanade, T.[1973]: "Picture Processing System by Computer Complex and Recognition of Human Faces", *Doctral Thesis*, Kyoto University.

Kanade, T., *et al.*[1976]: "Manual for the Conversational Picture Processing System", *KUIPNET Manual-3*, (in Japanese).

Kitazawa, S.[1976]: "Guide to KUIPNET—Outlines, Operations, and Specifications", *KUIPNET Manual-1*, (in Japanese).

Kleinrock, L.[1970]: "Analytic and Simulation Methods in Computer Network Design", *AFIPS SJCC*, Vol.36, pp.569-579.

Kleinrock, L., and Naylor, W.E.[1974]: "On Measured Behavior of the ARPA Network", *AFIPS NCC*, Vol.43, pp.767-780.

Kleinrock, L., Naylor, W.E., and Opderbeck, H.[1976]: "A Study of Line Overhead in the Arpanet", *C.ACM*, Vol.19, No.1, pp.3-13.

Mann, W.F., Ornstein, S.M., and Kraley, M.F.[1976]: "A Network-Oriented Multiprocessor Front-End Handling Many Hosts and Hundreds of Terminals", *AFIPS NCC*, Vol.45, pp.533-540.

McKenzie, A.A.[1972]: "Host/Host Protocol for the ARPA Network", NIC #8246.

McQuillan, J.M., Crowther, W.R., Cosell, B.P., Walden, D.C., and Heart, F.E.[1972]: "Improvements in the Design and Performance of the ARPA Network", *AFIPS FJCC*, Vol.41, pp.741-754.

Metcalfe, R.M.[1973]: "Packet Communication", MAC TR-114.

Mills, D.L.[1976]: "An Overview of the Distributed Computer Network", *AFIPS NCC*, Vol.45, pp.523-531.

Mimno, N.W., Cosell, B.P., Walden, D.C., Butterfield, S.C., and Levin, J.B.[1973]: "Terminal Access to the ARPA Network: Experience and Improvements", *Compcon 73*, pp.39-43.

NIC #8246: See McKenzie[1972].

NIC #17759, Neigus, N.: "File Transfer Protocol", July, 1973.

NIC #18639, "TELNET Protocol Specification", August, 1973.

Ornstein, S.M., Heart, F.E., Crowther, W.R., Rising, H.K., Russell, S.B., and Michel, A.[1972]: "The Terminal IMP for the ARPA Computer Network", *AFIPS SJCC*, Vol.40, pp.243-254.

Postel, J.B.[1971]: "Official Initial Connection Protocol", NIC #7101.

Roberts, L.G., and Wessler, B.D.[1970]: "Computer Network Development to Achieve Resource Sharing", *AFIPS SJCC*, Vol. 36, pp.543-549.

Rosner, R.D., Bittel, R.H., and Brown, D.E.[1975]: "A High Throughput Packet-Switched Network Technique Without Message Reassembly", *IEEE Trans.*, COM-23, No.8, pp.819-828.

Sakai, T., Kanade, T., Nagao, M., Tabata, K., Kitazawa, S., Ohnishi, H., and Ohta, Y.[1973]: "Inhouse Computer Network for Information Processing and Some Applications", presented at the Seminar on Computer Assisted Chemical Research Design, July 2-6 1973, Hawaii U.S.A.: published in Fujiwara, S., and Mark, H.B.[1975]: *"Information Chemistry: Computer Assisted Chemical Research Design"*, University of Tokyo Press, pp.309-336, Sakai, T., Tabata, K., Kanade, T., Kitazawa, S., Ohnishi, H., and Ohta, Y.: "KUIPNET: In-House Computer Network for Information Processing and Some Applications".

Sakai, T., Tabata, K., Ohnishi, H., and Kitazawa, S.[1974]: "Inhouse Computer Network and HOST Computer", *Jour. Information Processing Soc. of Japan*, Vol.15, No.12, pp.948-954, (in Japanese); also appeared in *Information Processing in Japan*, Vol.15, pp.75-79.

Sanders, R.W., and Cerf, V.G.[1976]: "Compatibility or Chaos in Communications", *DATAMATION*, Vol.22, No.3, pp.50-55.

Tanaka, M.[1976]: "Computer Network Oriented Shared Memory", *Master's Thesis*, Dept. of Information Science, Kyoto University, (in Japanese).

TENEX[1973]: *"TENEX JSYS MANUAL — A Manual of TENEX Monitor Calls"*, Bolt Beranek and Newman Inc., Section 2, pp.81-90.

Tomita, S.[1973]: "On-Line, Real-Time, Multiple Speech Output System and Its System Evaluation", *Doctral Thesis*, Kyoto University.

Walden, D.C.[1972]: "A System for Interprocess Communication in a Resource Sharing Computer Network", *C.ACM*, Vol.15, No.4, pp.221-230.

Winett, J.M., and Sammes, A.J.[1973]: "An Interface to the ARPA Network for the CP/CMS Time-Sharing System—Volume 1", *MIT Technical Note 1973-50.*

Yanagi, K.[1976]: "Connection of Very Distant Host (TOSBAC-5600) to the KUIPNET", *Master's Thesis*, Dept. of Information Science, Kyoto University, (in Japanese).

LIST OF PUBLICATIONS AND TECHNICAL REPORTS

1. "Inhouse Computer Network for Information Processing and Some Applications"; T.Sakai, T.Kanade, M.Nagao, K.Tabata, S.Kitazawa, H.Ohnishi, and Y.Ohta, presented at the Seminar on Computer Assisted Chemical Research Design, July 2-6 1973, Hawaii USA; published in Fujiwara, S., and Mark, H.B.: *"Information Chemistry: Computer Assisted Chemical Research Design"*, University of Tokyo Press, pp.309-336, T.Sakai, K.Tabata, T.Kanade, S.Kitazawa, H.Ohnishi, and Y.Ohta: "KUIPNET: In-House Computer Network for Information Processing and Some Applications", 1975.

2. "Computer Network"; T.Sakai, and K.Tabata, *Jour. Soc. of Instrument and Control Engrs. Japan,* pp.863-873, Vol.12, No.11, Nov. 1973, (in Japanese).

3. "In-house Computer Network KUIPNET"; T.Sakai, K.Tabata, S.Kitazawa, and H.Ohnishi, *Proc. of the 14th Annual Convention of Information Processing Soc. Japan,* Dec. 1973 (in Japanese).

4. "In-house Computer Network KUIPNET and Its Subnet"; *ibid.*

5. "In-house Computer Network KUIPNET and HOST Software"; T.Sakai, K.Tabata, H.Ohnishi, and S.Kitazawa, *ibid.*

6. "Inhouse Computer Network KUIPNET"; T.Sakai, K.Tabata, S.Kitazawa, and H.Ohnishi, *Technical Report for the Professional Group on EC of IECE Japan,* EC73-56, Dec. 1973, (in Japanese).

7. "Inhouse Computer Network and HOST Computer"; T.Sakai, K.Tabata, H.Ohnishi, and S.Kitazawa, *Jour. Information Processing Soc. Japan,* Vol.15, No.12, pp.948-954, Dec. 1974, (in Japanese); also appeared in *Information Processing in Japan,* Vol.15, pp.75-79, 1975.

8. "Throughput Measurement of Switching Subnet of the KUIPNET"; T.Sakai, and S.Kitazawa, *Proc. of the 15th Annual Conv. of Information Processing Soc. Japan,* Dec. 1974, (in Japanese).

9. "Inhouse Computer Network KUIPNET for Information Processing"; T.Sakai, T.Hayashi, S.Kitazawa, and K.Tabata, *Proc. of PACNET Symposium on Computer Networks,* pp.39-46, Aug. 1975.

10. "Measurement of IMP and Subnet of the KUIPNET"; T.Sakai, and S.Kitazawa, *Proc. of the 16th Annual Convention of Information Processing Soc. Japan,* Nov. 1975, (in Japanese).

11. "Message-Switching System for the KUIPNET"; T.Sakai, K.Tabata, S.Kitazawa, and H.Ohnishi, *Trans. Inst. Electronics Comm. Engrs. Japan Abstracts,* Vol.58, No.11, pp.57-58, Nov. 1975 (Originally appeared in Japanese; *Trans. IECEJ,* 58-D, 11, pp.697-704, Nov. 1975).

12. "KUIPNET and Very Distant HOST (TOSBAC 5600)"; T.Sakai, K.Tabata, S.Kitazawa, K.Yanagi, and S.Iki, *1975 Nat. Conv. Rec. IECE Japan,* 1272, Mar. 1975, (in Japanese).

13. "Guide to KUIPNET—Outlines, Operations, and Specifications"; S.Kitazawa, *KUIPNET Manual-1,* Feb. 1976, (in Japanese).

14. "NEAC 2200/250 NOS Operating System—Design Concepts, Functions, and Operations"; T.Hayashi, *KUIPNET Manual-2,* Feb. 1976, (in Japanese).

15. "Manual for the Conversational Picture Processing System"; T.Kanade, *et al., KUIPNET Manual-3,* Feb. 1976, (in Japanese).

4-6