A local area network for intercomputer, interprocess data communications
by James Leroy Schlegel

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
With the ever increasing number of computers being used at Montana State University, a mechanism for exchanging data between these computers was needed. A local area network to interconnect mainframe, mini, and micro computers for the purpose of allowing programs running on different computers to intercommunicate was to be designed and implemented.

The techniques of structured analysis and design were to be employed to provide a top-down hierarchical design for this network. This entailed reviewing the present systems used for interprocess communications, modeling a new system for intercomputer, interprocess communications, designing this system and implementing it on a number of computers.

A local area network called STARNET, acronym for STAR topology NETwork, was designed to meet this need. STARNET consists of a set of computer programs which connect their computers together in a star shaped topology. One program resides on the star's central computer and performs the packet-switching functions to route packets between their origin and destination computers. The other programs reside on the star's point computers and provide network access to user programs running on these computers. An asynchronous RS-232 line is used to connect each point computer to the central computer, providing a cost effective network.

STARNET has proved to be an effective communications mechanism for intercomputer, interprocess communications. STARNET itself does not perform any application functions but is a mechanism upon which applications can be written. The star topology configuration has both good and bad points. The cost of connecting computers to the network in terms of hardware is minimal and the software routing algorithm was easily implemented. However, the restriction of always sending data via the central node is not always the preferred method. Whenever the central node is down, the entire network is down which is not the case with other topologies. Also, whenever the central node is very busy, the network throughput degrades.

A LOCAL AREA NETWORK FOR INTERCOMPUTER, INTERPROCESS DATA COMMUNICATIONS

by

James Leroy Schlegel

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 1984

APPROVAL

of a thesis submitted by

James Leroy Schlegel

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Nov. 26, 1984
Date

Martin Laulkner
Chairperson, Graduate Committee

Approved for the Major Department

Nov 26th 1984
Date

J. Dunbigh Stanley
Head, Major Department

Approved for the College of Graduate Studies

11-28-84
Date

WB Malone
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master of Science degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted by the copyright holder.

Signature James L. Schlignl

Date 11/26/84

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With the ever increasing number of computers being used at Montana State University, a mechanism for exchanging data between these computers was needed. A local area network to interconnect mainframe, mini, and micro computers for the purpose of allowing programs running on different computers to intercommunicate was to be designed and implemented.

The techniques of structured analysis and design were to be employed to provide a top-down hierarchical design for this network. This entailed reviewing the present systems used for interprocess communications, modeling a new system for intercomputer, interprocess communications, designing this system and implementing it on a number of computers.

A local area network called STARNET, acronym for STAR topology NETwork, was designed to meet this need. STARNET consists of a set of computer programs which connect their computers together in a star shaped topology. One program resides on the star's central computer and performs the packet-switching functions to route packets between their origin and destination computers. The other programs reside on the star's point computers and provide network access to user programs running on these computers. An asynchronous RS-232 line is used to connect each point computer to the central computer, providing a cost effective network.

STARNET has proved to be an effective communications mechanism for intercomputer, interprocess communications. STARNET itself does not perform any application functions but is a mechanism upon which applications can be written. The star topology configuration has both good and bad points. The cost of connecting computers to the network in terms of hardware is minimal and the software routing algorithm was easily implemented. However, the restriction of always sending data via the central node is not always the preferred method. Whenever the central node is down, the entire network is down which is not the case with other topologies. Also, whenever the central node is very busy, the network throughput degrades.

CHAPTER 1

INTRODUCTION

Computer networking has been around since the late 1960s when Arpanet was developed by the Defense Advanced Research Projects Agency. Arpanet linked computers in universities and Federal research laboratories around the U.S. The success of Arpanet and the increasing need for computer networks prompted standards organizations to work on setting up guidelines for networking. Manufacturers of computers were developing their own networking systems which were all incompatible with each other. The hope of the standards organizations was to create a universal networking standard which would result in compatibility between all computer networks. In 1974, IBM announced System Network Architecture (SNA); Digital Equipment Corporation followed with Digital Network Architecture (DNA) and Univac with Distributed Communications Architecture (DCA). These networking systems were similar in concept but incompatible in reality. Since that time, the International Standards Organization (ISO) developed a 7-layer reference model for networking called the Open System Interconnection (OSI) model which was first published in 1978 (2,9,16,29,32). The

OSI model has led IBM, DEC, Univac and others to move toward more compatible networks, but most are still not totally compatible.

This incompatibility is a problem at Montana State University. Currently, we have seven time-sharing computers running four different operating systems. This variety makes it very difficult to share data between computers. Users of MSU's computers need to be able to send data files from one computer to another to take advantage of software packages which are unique to each computer. Also, sharing resources such as high-speed printers or graphic plotters is economically important. Buying one high quality resource which can be shared by all is an admirable goal.

This sharing of data and resources requires a common communication mechanism so all computers will 'speak' the same language. The research and development of a mechanism to allow all of MSU's computers to communicate with each other is the object of this thesis. The result is not expected to be a solution to all existing needs but should be a step in the right direction.

CHAPTER 2

STATEMENT OF THE PROBLEM

> Design a communication mechanism which will allow the computers on Montana State University's campus to share data in a convenient and reliable manner.

This communication mechanism is to be a system service which application programs can use to transmit data between themselves whether or not the programs are executing on the same computer. It should transfer data reliably and should be easily called by application programs.

The solution to this problem was divided into three phases. Phase one, the research phase, dealt with reviewing the available literature concerning networking and data communications. Besides articles and books, this literature included the operating systems manuals for MSU's computer systems. The knowledge gained in this phase was used in phase two to analyze and design the communication network. The second phase resulted in a specification from which the implementation in phase three was written.

CHAPTER 3

LITERATURE REVIEW

The literature on networking is abundant with articles appearing regularly in most computer related magazines and publications. A list of literature read is given in the bibliography at the end of this thesis. The topics covered the spectrum of design, use and management of computer networks. Of most interest were the articles on network design. However, the articles on the use and management of networks gave insight into what was needed by the end users of networks to accomplish their tasks.

Most articles dealing with network design referenced the Open System Interconnection (OSI) model developed by the International Standards Organization (ISO). This model is used as the basis for most modern network design. A review of this model is included as it is also used as a basis for the design of this network. Other articles reviewed existing networks such as SNA and Ethernet while others reviewed network topologies and the routing algorithms used by each topology to transfer data between the nodes on the network.

Open System Interconnection Model

The International Standards Organization (ISO) began to develop the Open System Interconnection (OSI) model in 1975. This model, published in 1978, is the basic reference model for designing networks (2,9,16,29,32). It defines seven functional levels and details the function of each level. The levels are numbered from 1 to 7 with 1 being the lowest level and 7 the highest. Each level uses and adds to the functions of the levels lower than it. Besides being numbered, each level is named according to its function. The OSI model is depicted in Figure 1.

Figure 1. Open System Interconnection Model. Developed by the International Standards Organization.

Computer A				Computer B
PRESENTATION	7	<---------->	7	PRESENTATION
APPLICATION	6	<---------->	6	APPLICATION
SESSION	5	<---------->	5	SESSION
		Peer-to-peer protocol		
TRANSPORT	4	<---------->	4	TRANSPORT
NETWORK	3	<---------->	3	NETWORK
DATA LINK	2	<---------->	2	DATA LINK
PHYSICAL	1	<---------->	1	PHYSICAL

Each level in the ISO model communicates with its peer level on another computer using the services provided by the levels lower than it. These services are then built upon to provide a higher-level communication protocol. Gaining an understanding of each level enables one to perceive what a networking system is all about. Today, these levels are implemented in a combination of both hardware and software with hardware gradually working its way up the levels. As each level is defined in a standard, that level is subject to implementation in an integrated circuit. This enables more efficient and standardized networks to be developed. Each level is summarized in the following paragraphs as this information forms the basis on which STARNET was designed.

Level 1, the physical level, defines the electrical and mechanical interfaces between the network and the user. An RS-232 interface is one example, broadband and baseband coaxial systems are others. This level allows any two devices to be physically connected with a common cable. It is up to higher levels to decipher the data which traverses this cable. One important fact to keep in mind is that level 1 interfaces are NOT error-free. They are prone to noise and other electrical and mechanical interference which may cause errors to occur in the data transmissions.

This leads to the purpose of level 2, the data link level. The data link level corrects all detected errors in the transmissions between two devices. This is accomplished

by adding a checksum to the data being transmitted and verifying this checksum when the data is received. If the checksum is verified, an acknowledgment signal (ACK) is sent back to the sending unit which then sends the next piece of data otherwise a negative acknowledgment (NAK) is sent which results in the data being retransmitted by the sending unit.

Since a network can be composed of many nodes which are not fully connected, data may have to travel between several nodes before it reaches its final destination. A fully connected network contains a link between each and every node so any data being transmitted can be sent directly to the node for which it is destined. This is not true of any other incomplete network. The routing of data from its origin node to its destination node is the responsibility of the network level, level 3. This is accomplished by adding addresses to the data which specify both the origin and destination of the data. Each node uses the destination address to determine where to next send the data. Many algorithms, which vary with the type of topology, are used to route data in a network (18).

Level 4, the transport level, provides verified end-to-end transmissions. While the data link level provides verified transmissions between adjacent nodes, the transport level verifies these transmissions between the origin and destination nodes.

These first four levels comprise the transfer services of a network. Together, they provide verified end-to-end data transmission mechanism which can be used to implement applications. The origin and destination nodes use all four levels to transfer data while intermediate nodes use only levels 1 to 3. These nodes do not get involved in end-to-end verification, only in point-to-point verification. The next three levels comprise the user levels. An application program must be aware of them and interface to the network via their protocols.

Level 5, the session level, gives the user the ability to establish a connection to another user on the network. This session is then used as the data communication channel between the two users.

Next, level 6, the presentation level, defines how a user communicates with the network and thus with another user. How the data must be presented to the network affects the implementation of the application programs.

This leads to level 7, the application level. This level is the application program which does the actual work the user desires. It uses the lower 6 levels to provide the data communication needed for the application.

Most networks fully support the lower four or five levels leaving the upper levels for the user to support in the application programs.

Packets and Protocols

Some levels in the OSI model require the addition of information to the data to provide verified transmission and to allow the network to correctly route the data between its origin and destination. This combination of the data and the additional network required information is referred to as a 'packet'. It is also referred to as a 'frame' in some literature. A packet consists of several fields, each of which has its own purpose. The data link level requires a checksum field and an identification field to verify point-to-point communications. The network level adds addressing fields to route the packet. Besides these fields, the user data field is referred to as the message or information field. Some packets also contain control characters which are used for synchronization of the packets and for identification of the start and end of fields within the packet. An example packet might look like this:

< id >< addresses >< message >< checksum >

Each level in the OSI model communicates with its peer levels on other computers via a common protocol. A protocol is a formal set of rules which define exactly how data is exchanged and what that data means. Without a protocol, the levels would be 'speaking' a different language and would be

unable to communicate. This is the major problem between networks from different manufacturers. Each manufacturer has defined its own unique protocol in one or more of the levels. This makes their networks incompatible at those levels and thus totally incompatible. Total compatibility requires that all seven levels use the same protocols. As the ISO organization has been standardizing the lower levels, some networks are now compatible in these levels. As work continues for standardizing the higher levels, more and more networks will become compatible and in the future it is hoped that all networks will be compatible.

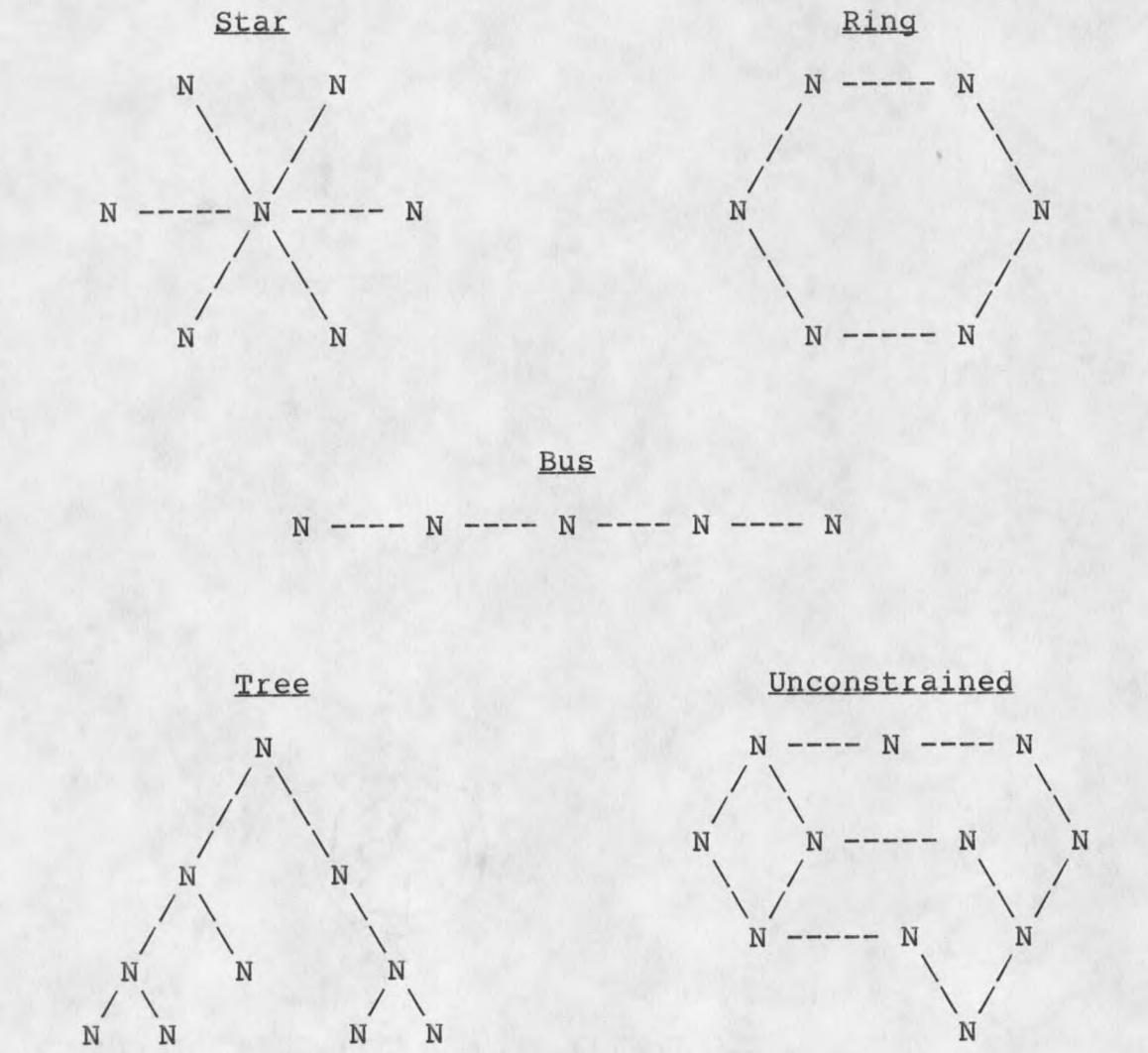
Network Topologies

The computers in a network are physically connected in a manner called a topology. Several types of topologies are identified as being useful when building a network. Which type is used depends on the desired result. This selection is based on cabling costs, relative location of the devices, complexity of the routing algorithm, desired reliability, maximum path length and other factors which contribute to the cost of designing and building a computer network.

Five topologies are discussed within computer networks. They are the star, the ring, the bus, the tree, and the unconstrained topology as shown in Figure 2. They can be compared and evaluated to determine their good and bad

attributes. These attributes include reliability, interface complexity, modularity, flexibility and cost. The result of comparing various network topologies will vary depending on what the network being developed is supposed to do.

Figure 2. Network Topologies. N = node.



Current Systems

Each computer system for which STARNET was implemented has a mechanism for interprocess data communications. Each mechanism was studied in detail for two reasons. First, to find out how other systems implemented data communications and second, because these mechanisms would be needed to write the programs for these computer systems.

CP-6 comgroups

A facility called a comgroup, acronym for communication group, is available on CP-6 for transmitting data between devices (computers, printers, terminals, etc.) and processes (CP-6 programs). A comgroup enables a process to read and write messages to and from other processes or devices which are connected to the comgroup. Information which describes the attributes of the message are associated with a comgroup message. The attributes of interest are the origin and the destination names. The origin is the name of the sender of the message while the destination is the name of who is to receive the message. To ensure proper delivery, each device or process has a unique name in the comgroup. Thus, process ABC sends a message to device XYZ by specifying XYZ as the destination when the message is written to the comgroup. The comgroup then takes care of transmitting the message to XYZ. A process can connect to a comgroup by opening a DCB

(data control block) to that comgroup's file. A device connects by logging onto the CP-6 system in the same manner a timeshare user would log on. The logon's account and password, however, direct CP-6 to connect the device to CP-6 as a comgroup user rather than as a timeshare user. The topology of this mechanism is a star, with the comgroup at the center and the processes and devices at the points. The CP-6 Monitor Services Reference Manual (17) contains a detailed discussion of comgroups and their capabilities.

VMS mailboxes

The VMS operating system uses a different mechanism, called mailboxes, to allow processes to intercommunicate. A mailbox can be created and used, much like a post office mailbox, as a communication link between processes. Any number of processes can read and write messages to and from the same mailbox, but there are no attributes associated by the mailbox to the messages. When a process reads a message from a mailbox, it gets the next message in the mailbox regardless of whether the message was destined for that process. This limits the usefulness of a mailbox to one-way transfers between two processes. For two-way communication, two mailboxes are needed, one for messages sent from process A to process B and the other for messages sent from B to A. For communications between many processes, each process needs its own mailbox to receive data. Since the origin of

the message is not available from the mailbox as an attribute associated with the message, it must be added to the message itself if it is needed. Usually, the origin of the message is implicitly defined by what mailbox is being read. The VMS System Services Reference Manual (13) and I/O User's Guide (14) should be read for more details.

RSTS/E Receivers

A RSTS/E system allows local communications between processes via its send/receive facilities. This mechanism is similar to the mailbox mechanism used in the VMS system described above except that only one-way communications can be created. A process must first declare a 'receiver' which other processes can then use to send data to that process. This facility provides only one-way data transfers as only the process which declared the receiver can read data from it. Two or more way communications requires that each process declare its own receiver which it then uses to accept messages from the other processes. In this manner, these receivers function much like the VMS mailboxes except that the RSTS/E receivers allow the senders to add parameter information to the message as a second data block which is transferred with the message. The message's origin and destination information could be included in this second block. The RSTS/E System Directives (12) and Programming (11) manuals have more information on this facility.

CHAPTER 4

ANALYSIS AND DESIGN OF STARNET

The analysis and design of a system requires using various tools which aid and control the development of the system. These tools are analysis and design methodologies, documentation standards, and coding practices as well as any computer programs, books, etc. which help the development.

For the purposes of this thesis, system analysis is used to model an existing system (if one exists), then to modify this model to meet the new needs of its users. If no system currently exists, a model for the new system is created. System design is used to change this model into a specification which can be automated using a computer. The analysis phase is separated from the design phase as not all systems are implemented entirely as computer programs. Some systems require hardware to perform some of the tasks while other tasks may even have to be done manually. The model represents the tasks and the relationships between the tasks but not how these tasks are going to be done. When part or all of the model is to be automated on a computer, system design techniques are used to create the specifications for the computer programs.

Structured Analysis and Design

The technique used to create STARNET is referred to as Structured Analysis and Design (10,33,34). It is called structured as it is a top-down, hierarchical analysis and design methodology which results in a document called a Structured Specification.

The system being analyzed and designed is broken down and documented using a top-down approach. Top-down refers to looking at the whole system first, then breaking the whole into several smaller parts, and then breaking these parts into even smaller parts. This continues until the parts cannot be logically subdivided any further. The resulting hierarchy lends itself nicely to being implemented using a structured computer language.

The result of the analysis and design is a document called a Structured Specification. A fundamental principal that this document is built upon is that any level within the hierarchy should be able to be expressed on a single page as an entity unto itself.

Extensive details on Structured Analysis and Structured Design can be found in the books by De Marco (10) and Yourdon (33,34).

Structured Specification

A Structured Specification is a document that is the result of analyzing and designing a system which is to be automated on a computer. This specification is used when writing the computer programs which perform the various tasks that are required by the system. It consists of four sections each of which has a different purpose although all four sections must be used together to fully define and document the system. These sections contain: 1) Data Flow Diagrams, 2) Process Specifications, 3) Structure Charts and 4) the Data Dictionary.

Data Flow Diagrams. Data flow diagrams graphically depict each task in the system's hierarchy and what data flows between these tasks. The term 'flow' is used to indicate data passing from one task to another task. The elements which make up a data flow diagram are the process, the dataflow, the data store and the terminator. A process is drawn as a circle with its name and number written inside. A dataflow is a directed line with the data's name written beside the line. The direction the data is flowing is shown by arrows at the ends of the line. A data store is a repository of data such as a disk file or memory buffer. It is drawn as two parallel lines with the data store's name written inside. A terminator is either an external source of data or an external sink of data and is drawn as a labeled box. These sources and sinks represent the outside

world as viewed by the system being described. Terminators usually only appear on a special data flow diagram called the Context Diagram. A context diagram depicts the system as a whole and its relationship to the outside world. Two data flow diagrams are depicted in Figures 3 and 4.

Figure 3. First Level Data Flow Diagram.

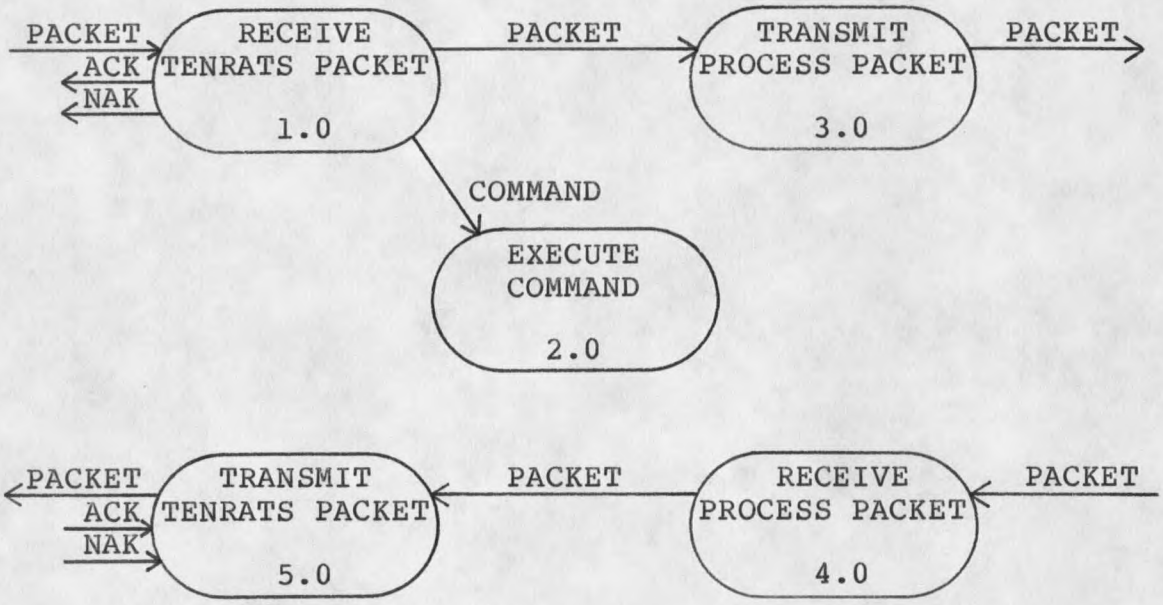


Diagram 0.0 STARNET (Point Node)

The data flow diagram depicted in Figure 3 represents the first, or top, level breakdown of the program STARNET. This breakdown resulted in five subprocesses labeled 1.0, 2.0, 3.0, 4.0 and 5.0. The data which flows between these

processes and to and from the outside world are drawn as directed lines into and out of the processes.

Some of the processes represented in Figure 3 can be broken down further into more data flow diagrams. The next level breakdown of process 5.0, TRANSMIT TENRATS PACKET, is shown in Figure 4. Note that all the data flows into and out of process 5.0, as shown in Figure 3, are shown as inputs and outputs to the processes in Figure 4.

Figure 4. Second Level Data Flow Diagram

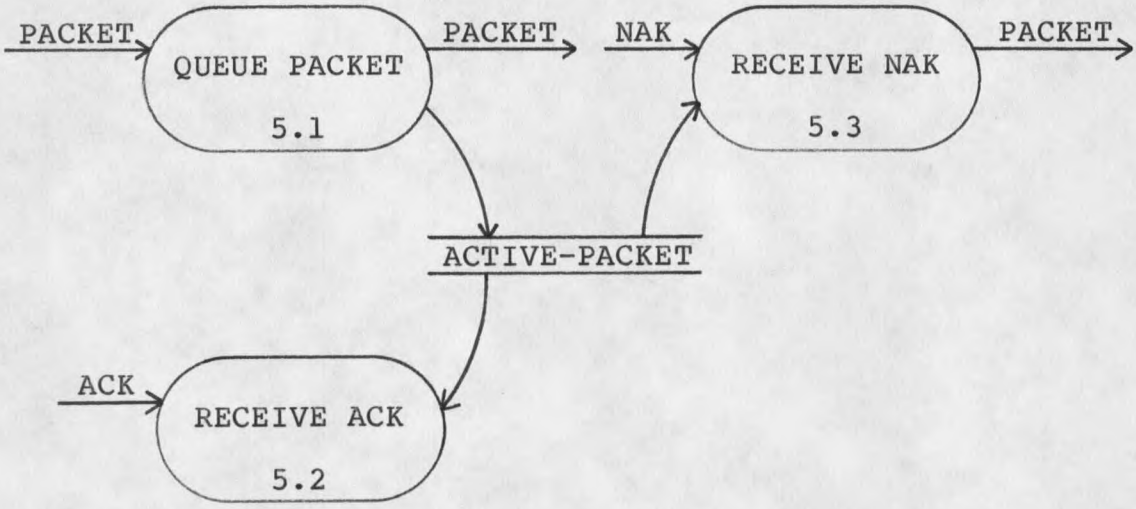


Diagram 5.0 TRANSMIT TENRATS PACKET

<u>Process Specifications.</u> When a process can no longer be logically broken down into another data flow diagram, a process specification is used to write its algorithm. Process 4.0, as shown in Figure 3, is such a process and the resulting algorithm is shown below.

Process 4.0 - RECEIVE PROCESS PACKET

1) Receive PACKET from a process.
2) Load ID with NEXT-ID.
3) Increment NEXT-ID modulo 1000.
4) Load ORIGIN-NODE with NODE-NAME.
5) Compute and load CHECKSUM.
6) Call Process 5.1 - QUEUE PACKET.

Structure Charts. An important aspect of data flow diagrams is that they only show the tasks and the data which flows between these tasks. They do not indicate in what order the tasks need to be executed. This is the job of the Structure Chart. After all the tasks have been defined either by a data flow diagram or by a process specification, they are grouped hierarchically by control flow. The top task on a structure chart can be viewed as the main program with the other tasks becoming subroutines. An example structure chart can be found in either Appendix A or B.

Data Dictionary. The data elements referred to in the data flow diagrams and the process specifications are defined in the data dictionary. This definition needs to be very specific as it is used to define these elements within a computer program.

Appendices A and B are the Structured Specifications for the programs written for STARNET. These can be studied for a better understanding of how all these elements fit together to form a complete description of a system.

Functional Requirements

The function of a system dictates the design and implementation requirements. The first step in specifying these requirements is to list for what and how the system will be used. This communication mechanism must support the following requirements:

1) RS-232 asynchronous interfaces.
2) Realtime intercomputer, interprocess communications.
3) One-to-one interprocess communications.
4) Many-to-one interprocess communications.
5) Many-to-many interprocess communications.

The above list came from hardware requirements and two applications which intended to use the network. Requirement number one dictates that inexpensive RS-232 terminal lines should be used to connect the computers. This type of interface is readily available but limits the transmission speed between computers to 9600 baud, 19.2K baud at best, which should be ample bandwidth for the applications under consideration.

The first application which uses this network is a centralized computer accounting system which maintains a single data base for all computer systems. This data base resides on the Honeywell CP-6 system and must be accessed

whenever a user logs on or off any computer system. This requires realtime program to program data communications yielding requirement number two. When a user logs on or off a computer, that computer executes a program which must access the data base on CP-6. This requires immediate feedback from the program executing on CP-6 which services the data base. Since users log on and off computers frequently, many queries to the data base may be pending simultaneously. This yields requirement number four, that many programs be able to communicate with a single program on another system.

The second application is a file transfer system which allows a user to transfer a file between any two computers on the network. This requires that a program on one computer be able to connect to a program on a second computer and be protected from other users who may be transferring files at the same time so file integrity will be maintained. This results in the third requirement of one-to-one communications.

Extending the one-to-one and many-to-one requirements presents an obvious addition of being completely flexible and allowing many-to-many communications.

Since each computer system already has a mechanism for interprocess communications within its own system, it is logical that the intercomputer mechanism be an extension of these existing mechanisms. A programmer can then use the

same system service routines regardless of whether the data communications are within a single computer or between several computers. This lessens the effort needed to learn the network. Secondly, trying to implement a duplicate mechanism within a computer system, even though the total network may benefit, is probably impossible.

The above requirements and the decision to use the existing interprocess data communication mechanisms affect the network's design. The design of this network follows levels one through three of the ISO's OSI model.

RS-232 Asynchronous Interface

Level one of the OSI model deals with the physical electrical and mechanical interfaces between computers. The decision to use the RS-232 asynchronous interface was made as one of the requirements this network must meet. This was done for purely economic reasons as RS-232 interfaces are readily available and relatively inexpensive when compared with other options.

Packet Switching

The OSI model's second level, the data link level, defines how any two nodes within the network are going to exchange packets reliably. A checksum, which is added to each packet, is verified by a node when it receives the packet. If the checksum received matches the checksum

calculated the packet has been received correctly and a positive acknowledgment is returned to the sending node, allowing it to send the next packet. If this comparison fails, a negative acknowledgment is returned which requests that the packet be retransmitted. This function is needed as the RS-232 interfaces and lines are subject to mechanical and electrical interference which may cause errors in the data transfers.

The use of packets and packet-switching adds tremendous functionality and usability to a network over the method of dedicating the physical connection to a single user. This latter method has been used in the past to connect two programs on different computers with the link between the two programs being created at the physical level. To use this link, a user must request that the physical link between the two computers be allocated to his and only his use. That user then ties up the link until he decides to release it. Any other users must wait in line to access the link. These users cannot even contend for the link on a first-come, first-served basis as the allocation of the link is purely random.

With packet-switching, the link is created at the network level as defined by the OSI model, not at the physical level. This link, called a 'virtual' link, does not exist physically, only logically. Moving from physical links to virtual links between programs enables more than

one user to concurrently share the hardware connection between two systems. This sharing increases the usability and flexibility of the network. Now all the users have immediate access to the link without having to wait. A further benefit arises in that with the dedicated link method, the bandwidth of the link is seldom fully utilized. This wasted bandwidth is available for use in a packet-switching network. One drawback, that an individual user may see, is slower response. Having to share a single link may slow the communications of users individually but will increase the overall use of the link. This is the same reasoning used to justify time-sharing operating systems over single user operating systems.

Star Topology Network

In the Network Topologies section in Chapter 3, five possible topologies are discussed along with the criteria for choosing the most effective topology for an application. These criteria are reliability, interface complexity, modularity, flexibility, and cost. Ed Pevovar and Brian McGann (28) used these criteria to compare the five topologies as shown in Table 1. They use the terms 'reliability' to mean the inherent reliability of the topology, 'modularity' to mean ease of adding new nodes to the network, 'flexibility' to mean ease of adding new lines between nodes, 'interface complexity' to mean both hardware

and software interface complexities, and 'cost' to mean overall implementation costs.

Table 1. Network Topology Comparison for LANs.

Topology	Reliability	Interface Complexity	Modularity	Flexibility	Cost
Star	poor	simple	moderate	poor	high
Ring	moderate	simple	good	moderate	mod.
Bus	good	moderate	good	good	low
Tree	good	moderate	good	good	low
Unconst.	very good	very high	moderate	poor	high

This comparison was made regarding the use of these topologies in a local area network for interconnecting computers, peripherals, and terminals. The results of this comparison are not directly applicable to other networking applications. They do, however, represent valid arguments which may be applied to more than one situation and most importantly, define a set of criteria which can be used to compare various topologies.

These criteria were applied to the functional and system requirements for the MSU intercomputer, interprocess communication mechanism being considered in this thesis. The results are shown in Table 2 which can be compared to the results shown in Table 1.

Table 2. Network Topology Comparison for MSU.

Topology	Reliability	Interface Complexity	Modularity	Flexibility	Cost
Star	good	simple	very good	poor	low
Ring	very good	simple	good	moderate	mod.
Bus	poor	moderate	good	good	high
Tree	poor	moderate	good	good	high
Unconst.	high	very high	moderate	poor	high

The changes in the ratings can be attributed to a new application and to the 'real world' effects on the ratings. Pevovar and McGann's ratings are based on more theoretical considerations rather than practical considerations. This does not mean that what they said is invalid, only that the results of applying these criteria change in the real world.

A discussion of each criterion and why the particular ratings were assigned will explain why the star topology was chosen as the best approach when it looks very weak by the standards set by Pevovar and McGann.

The reliability of a network hinges on what happens when a node fails. What affects this failure has on the rest of the network varies considerably from topology to topology. In a star topology network, the central node is critical! Its failure causes the entire network to fail, i.e. without the central node there is no network. The failure of a point node only affects that node which is the least harmful ramification. The inherent reliability in a star network is poor as a result of the total dependence on

the central node. If, however, this node is itself very reliable, then the entire network is very reliable. The reliability of a star network is directly related to the reliability of the central node.

Losing a node in the ring topology still leaves an alternate path which all the remaining nodes may use. This redundancy is very good as there would be no problems until two or more nodes failed. This situation is even postponed using the unconstrained topology with many redundant paths between nodes. Both the bus and tree topologies lack this redundancy which will cause the network to separate if any internal node fails. The result would be two smaller disjoint networks. From the point of view of reliability, the unconstrained network wins with the ring and star topologies following in that order if the star topology has a reliable central node. The bus and tree topologies are less desirable as a result of the problem of dividing the network if an internal node fails.

The interface complexity is related to the software and hardware costs necessary to implement the topology. The more complex the topology, the more complex the interface. In this sense, no arguments could be found with Pevovar and McGann's results. This criterion is probably the most important for this problem as the time needed to implement the software and the cost of the hardware were two critical factors in the network's implementation. The CP-6 comgroup

facility is suited perfectly to implementing any of the topologies while VMS and RSTS/E would require a large effort to realize some of the topologies. VMS and RSTS/E do not have the facilities to easily manage the dynamic number of connections which are required in the tree and unconstrained topologies. So, the simpler the topology, the simpler the implementation.

The modularity ratings were changed only for the star topology assuming that CP-6 was the central node. The comgroup facility imbedded within CP-6 makes the connection of additional nodes easy. These connections can be made dynamically without affecting the rest of the network. This ability does not exist in VMS or RSTS/E.

The flexibility criterion like the interface complexity criterion is a function of the topology. The same ratings are used in both comparisons.

The cost of a topology is the bottom line summary of the other four criteria. The star topology, with CP-6 as the central node, was rated as the lowest cost for several reasons. While the unconstrained network is theoretically the best network, its cost to implement is beyond the scope of this network. The unreliability and moderate interface complexity of the bus and tree topologies make them unattractive also. This leaves the star and ring topologies, and possibly a hybrid between the two, open for consideration. It was decided that the best approach to

solving this problem would be to implement the star topology first, as it would be easiest. After this topology was working, upgrades could be considered using the knowledge gained in implementing and using this network.

The Honeywell computer running CP-6 was picked to be the central node because it is very reliable as a computer. The policy of maintaining 24 hour operation of the Honeywell computer makes it an attractive choice for this node. Other systems on campus are either not as reliable as it or do not have the 24 hour operational commitment. The comgroup facility of CP-6 also makes implementation of the central node much simpler than on any other system on campus. This positions all the other systems on campus as point nodes on the star topology which makes their implementation straight forward.

CHAPTER 5

IMPLEMENTATION OF STARNET

Having chosen the star as the topology for the network, the name STARNET, acronym for STAR topology NETwork, was given to this project. With this topology, two programs were needed to implement the network. The first runs on a point node and gives network access to the users of that node while the second runs on the central node and performs the packet-switching. These two programs use the identical handshaking protocol to transfer information, so in that sense, they appear as mirror images of each other. One result of this is that the central node is necessary only if more than two computers are to be networked. Two computers can be networked using only the point node programs as a point node cannot tell whether it is communicating with another point node directly or via the central node.

```
                    STARNET
                       |
          STARNET      |      STARNET
                \      |      /
                 \     |     /
STARNET -------- TENRATS -------- STARNET
                 /     |     \
                /      |      \
          STARNET      |      STARNET
                       |
                    STARNET
```

STARNET | TENRATS - A Mirror Image

These two programs were given names to reflect their function. The point node program was named STARNET so the users of each computer would easily recognize which program gives them access to the network. The central node program was given the name TENRATS, which is the mirror image of STARNET, because each STARNET program does not see TENRATS when handshaking data to another node, it only sees a mirror image of itself.

While a STARNET program is needed for each computer on the network, only one TENRATS program is needed to perform the packet-switching.

Packet Structure

The general structure of a packet was discussed in Chapter 3 of this thesis and was shown as:

< id >< addresses >< message >< checksum >

The exact structure of the packet must now be selected to fit the requirements of the network. This structure is directly affected by the systems upon which the network is implemented. The 'id' field is a three byte numeric field used to identify each packet as it is generated by a point node on the star network. This field is initialized to a

decimal '000' and is incremented by one, modulo 1000, each time a new packet is generated. This field is used in conjunction with the checksum field to verify packet transmissions between nodes.

The 'addresses' field contains both the packet's origin and destination addresses. Both the origin and destination addresses contain two subfields. One is the node address while the other is the process address within that node. An address thus looks like this:

< node address >< process address >

The node address is an eight character alphanumeric name which follows the rules dictated by CP-6's comgroup facility. This follows since the central packet-switching node is CP-6. Each point node connects to the network by connecting to TENRATS' comgroup file as a comgroup station. A comgroup station name is defined as an eight character alphanumeric name. A process address is a ten character name which must follow the rules of the system for which it is intended to be used. On CP-6, it must be a valid comgroup station name. On VMS, it must be a valid mailbox name, and on RSTS/E, it must be a valid receiver name.

Together, the id and origin node address, uniquely identify a packet. This unique identification is crucial to guaranteeing verified communications.

The message field is the actual data which is being transferred between programs. It is a variable length field from zero to 512 8-bit ASCII characters which make sense only to the programs sending and receiving the data.

The checksum field, a single character, is calculated using a rotate-and-add-with-carry algorithm. This algorithm is a modification of the standard linear checksum algorithm to provide better reliability. The higher reliability is achieved because bytes of X'00' affect the checksum in the rotate-and-add-with-carry algorithm where they have no effect in the linear checksum algorithm. Although a cyclic redundancy checksum would provide even better reliability, it was ruled out because of the difficulty of calculation. The rotate-and-add-with-carry algorithm used is:

1) Initialize checksum to X'AA'.

2) For each byte in the packet do:

   a) Rotate checksum left 1 bit.
   b) Add byte to checksum.
   c) Add carry from step b to checksum.

3) If checksum < X'21'
   Then add X'21' to checksum.

Table 3. STARNET Packet Structure.

Field	Length in bytes	Type	Description
1	3	Numeric	Id
2	8	Alphanumeric	Origin node address
3	10	Alphanumeric	Origin process address
4	8	Alphanumeric	Destination node address
5	10	Alphanumeric	Destination process address
6	0-512	8-bit ASCII	Message
7	1	Integer	Checksum
Total	40-552		

Because the message field is of variable length, a terminator was added to the end of the packet to tell each node when it had completed reading a packet. The terminator used is a carriage return (X'0D'). The carriage return was chosen because it is used as a standard input terminator by all the computers being considered. Using more than one terminator, as some computers do, was not done as this only adds to the complexity of transferring the packets. Adding a terminator adds the requirement of masking any terminator which may occur as a character in the message field. This embedded terminator must be masked or it will cause a node to identify the end of the packet somewhere in the middle of the message and not at the end of the packet. A terminator is masked by changing it into a two character sequence which can be detected by the receiving node and unmasked. This two character sequence is composed of the data link escape character followed by the terminator with its high-order bit

set to 1. This also requires that the data link escape character be masked. When a packet is received, the receiving node scans the message for a data link escape character. If one is found, it is discarded and the next character in the message is modified back to its original form by setting its high-order bit to 0. Since the carriage return is the only terminator used, it and the data link escape character are the only characters which need to be masked in the message.

Handshaking Protocol

A major function of a network is to see that the data communications will proceed with minimal errors. Since errors can occur at the physical (hardware) level, the higher data link (software) level must be able to correct these errors. This is done by handshaking the data packets between nodes. During handshaking, a node will transmit a packet to a second node and then wait for an acknowledgment before proceeding any further. If a positive acknowledgment (ACK) is returned, then the node can transmit the next packet. If, however, a negative acknowledgment (NAK) is returned, then the node must retransmit the last packet as it was not received correctly by the receiving node. A sample handshaking sequence is shown in Figure 5.

Figure 5. Sample Handshaking Sequence.

```
Node A                          Node B

              Packet 000
     ------------------------------>
                 ACK
          <------------------
              Packet 001
     ------------------------------>
                 NAK
          <------------------
              Packet 001
     ------------------------------>
                 ACK
          <------------------

              Packet 002
     ------------------------------>
                 ACK
          <------------------
```

The transfer of packets from Node A to Node B is shown in Figure 5. This is only half of the communication between the nodes. The transfer of packets from Node B to Node A, though not shown, is occurring concurrently as well. This full-duplex communication yields better utilization of the link, as opposed to a half-duplex protocol like HASP.

A key feature of verified communications is the retransmission of packets which were not received correctly. This retransmission occurs whenever a node receives a NAK (negative acknowledgment) for a packet and also whenever a

timeout of an acknowledgment occurs. This timeout will occur when a transmission error causes the acknowledgment to be garbled beyond recognition or when the transmitting node is too busy to return a timely acknowledgment. Since a node is waiting for an acknowledgment before it transmits the next packet, if that acknowledgment was garbled, it would terminate transmission without the timeout feature. This timeout feature does cause some problems however. If the acknowledgment which was garbled was an ACK (positive acknowledgment), a correctly received packet would still be retransmitted. This might also occur when the other node is too busy to return a timely acknowledgment. Since each packet is uniquely identified, this duplication can be checked for and corrected by the receiving node.

Network Access

The users on any point node will use that computer's standard interprocessor communication mechanism to gain access to the STARNET program running on that node. Once access is gained to STARNET, the user then has access to the network via STARNET.

On CP-6, users connect to and use STARNET's comgroup file, as described in the CP-6 Monitor Services Reference Manual, and can then transfer data between themselves and any other program on the network to which they are linked.

Similarly, on VMS, users use mailboxes to communicate with STARNET. Mailboxes and their uses are described in the VMS System Services Reference Manual and the VMS I/O Users Guide.

Likewise, RSTS/E users use receivers to communicate with STARNET. Receivers are described in the RSTS/E System Directives Manual and the RSTS/E System Programming Manual.

Process Control

Being able to establish communication links between any number of processes on the network does not completely satisfy the needs of the users. How these processes are started has not yet been addressed. Obviously, a user can start a process executing on the computer he is currently logged onto. The user cannot, however, directly start a process on another computer without first logging onto that computer also. This latter method has been used at MSU to transfer files between computers.

A brief scenario of a user transferring a file using this method will demonstrate why a better method is needed. This user must first log onto one of computers that is to be involved in the file transfer. The next step is to allocate the link to the second computer. The disadvantages of dedicating this physical link were discussed previously in the section on packet-switching. Next, the user logs onto the second computer using the above allocated link. Now, a

program to transfer a file must be started on one computer and a program to receive this file must be started on the second computer. This is a manual process which the user must synchronize to ensure proper communications. The file is then transferred, the link deallocated, and the user is done.

Since a packet-switching network has overcome the necessity of dedicating a link between two computers to a single user, it would be advantageous if the network would also allow users to start programs on remote computers without having to first log onto that computer. This capability exists on all three operating systems being considered by this thesis. The CP-6 operating system allows a process to start the execution of ghost jobs and both the VMS and RSTS/E operating systems allow processes to start detached jobs.

Using this capability, the file transfer scenario becomes a simple matter of a user logging onto one of the computers in question and executing a file transfer program. This program, via the network, would start the execution of the remote program and then transfer the file. The user does nothing more than specify what file to transfer and where to transfer it to.

TENRATS on CP-6

The CP-6 operating system was chosen for the central packet-switching node, TENRATS, because of its comgroup facility. A computer which connects to the comgroup file managed by TENRATS is considered a point node on the star topology network and must perform packet handshaking with TENRATS. Appendix A is TENRATS' Structured Specification.

STARNET on CP-6, RSTS/E, and VMS

The point node program, STARNET, was written for three different operating systems. On CP-6, STARNET, like TENRATS, uses the comgroup facility but its comgroup file is used to provide communications between it and the processes on CP-6 which need to use the network. On VMS, a mailbox named STARMBX is created by STARNET to receive packets from other VMS processes. The RSTS/E version uses a receiver named STARNT to receive packets. Appendix B is STARNET's Structured Specification.

CHAPTER 6

APPLICATIONS WHICH USE STARNET

Two application programs are using the intercomputer, interprocess communications provided by STARNET. The first is the centralized computer accounting system which is an example of the many-to-one communication link. The second is a file transfer system which relies on one-to-one links.

Centralized Computer Accounting System

The first application to use STARNET was a centralized computer accounting system. MSU's Computing Services manages four computer systems for which a single accounting system was desired. The CP-6 system had an accounting system but neither the VMS or RSTS/E systems had one. The VMS and RSTS/E systems only had a capability to authorize users. They did not have the ability to limit the amount of usage that any particular user received. This capability was desired to facilitate management and allocation of these computers.

It was decided to extend the CP-6 accounting system to cover the VMS and RSTS/E computers as well. This is where

STARNET comes in. To extend this facility between many computers, a communications mechanism between the computers is needed. Now, when a user logs any of these four systems, the accounting data base residing on CP-6 is checked to see if that user has depleted his allocation for that computer. This is a realtime query which occurs every time a user logs on. Likewise, every time a user logs off a computer, the accounting data base is updated to reflect the new charges.

File Transfer System

The second application, which is still in progress, is a universal file transfer system. Currently, users can transfer files back and forth between the CP-6 and VMS systems but not between other systems except by magnetic tape. STARNET is being used as a basis for implementing a file transfer system where a user on any computer can transfer a file to any other computer.

The user runs the program XFER on any computer and tells it what file to transfer and what computer to transfer it to. XFER then starts, via STARNET, its counterpart, REFX, on the second computer. Together, these two programs transfer the file. When the transfer is complete, REFX stops and XFER prompts the user for more file to transfer. This is a very user friendly system.

CHAPTER 7

SUMMARY AND CONCLUSIONS

Summary

An intercomputer, interprocess data communication mechanism was needed at Montana State University to allow the diverse computer systems on campus to share data and peripherals. The existing literature on networking and data communications was studied and a network named STARNET was designed. A star topology was used as the basis for this network with one computer serving as the central packet-switching node allowing any number of computers to be attached as point nodes. STARNET was implemented using CP-6 as its central node and CP-6, RSTS/E and VMS as its point nodes. Two applications were written which needed to use STARNET as a communication mechanism. These applications were observed to see if and how well STARNET functioned. STARNET has been operational for two years and has evolved slowly over those two years as problems were fixed and enhancements were added. The current version runs very reliably and has not been changed for several months.

Discussion of Observed Performance

The performance of the network can be observed by the users of the applications which use the network. The time these applications need to execute is affected by the responsiveness of the network. It has been observed that the network response is related to the individual computer system activity. When the CP-6 system is very busy at the end of a quarter, the network throughput lowers. At the beginning of this project, it was thought that the limiting factor would be the asynchronous RS-232 connections between the computers which were limited to 9600 baud. The RS-232 lines have proved quite capable of handling the current load. Perhaps in the future, if the load increases, they may become the limiting factor but for now the computers themselves determine how well the network performs.

Recommended Enhancements

The weak point of the star network is the total dependence of the central node. This is the obvious place to start when considering future enhancements. Whether a totally new topology is used or a mutation of the star into something more inherently reliable will be left for that time.

Another enhancement that may be considered is blocking messages together before transferring them between nodes. The file transfer system saw a decrease of about an order of magnitude in the time needed to transfer a file by blocking many records into one message over the method of sending a single record per message. The latter method required many more packets to be sent between computers which increased the time necessary to transfer the file. This blocking of data may, however, be better served as a function of the network rather than as a function of the applications. As a network function, it may increase the performance of the whole network not just the applications which perform the blocking themselves.

Last, it would be interesting to implement STARNET on some microcomputer systems. What use they would be put to is unknown but a couple of ideas come to mind. First, the file transfer capability could be written. Micros can now transfer files to mainframes via the FTP (File Transfer Program) system provided by Computing Services. This system still depends on dedicated links so a network approach would be a more elegant solution. Second, some type of realtime data acquisition which needs the services of a mainframe would make a nice demonstration of the power of this network. The microcomputer could perform the realtime data acquisition and control and use STARNET to transfer this information to a mainframe for realtime analysis. A single

mainframe could monitor and control several microcomputer systems at the same time.

Conclusions

This project has been an interesting as well as challenging experience. The field of networking has been around for 20 years but has recently been thrust into the limelight mainly as a result of the proliferation of the microcomputer. The need to share data between computers is real and must be met by providing flexible and reliable network communications. STARNET by no means is an answer to everyone's needs but has been used to satisfy some of these needs, so in that light, its design and implementation has been a success.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Adiletta, W.F. "Packet network proven best for optimizing a library operation." Data Communications, February 1984:125-134.

2. Berry, D. "Protocol standardization works it way up the ladder of the OSI model." Electronics, June 14, 1984:148-151.

3. Beser, E.L. "Implementing X.25 Communications Protocol." Microsystems, June 1984:46-54.

4. Bjurlof, T. and W. Clausen. "Before you build a private packet switching network..." Data Communications, November 1983:267-270.

5. Bracker, W.E. Jr. "Surveying the protocol conversion vendors' offerings." Data Communications, August 1983:89-101.

6. Bullard, D. "Local Area Networks." Cause/Effect, November 1983:12-15.

7. Byers, T.J. "Micro-to-micro communications." Popular Computing, February 1984:113-119.

8. Dahlberg, B. "Chips support two local area networks." Computer Design, May 1984:107-114.

9. Data Decisions. "The ISO Reference Model-A Blueprint for Network Communication." Communications Systems, August 1982:1-11.

10. De Marco, T. Structured Analysis and System Specification. New York: YOURDON inc. 1978. 352p.

11. Digital Equipment Corporation. RSTS/E Programming Manual. 1983.

12. Digital Equipment Corporation. RSTS/E System Directives Manual. 1983.

13. Digital Equipment Corporation. VAX/VMS System Services Reference Manual. 1982.

14. Digital Equipment Corporation. VAX/VMS I/O User's Guide. 2 vols. 1983.

15. Feltman, C. "The micro-to-mainframe connection." Popular Computing, February 1984:105-108.

16. Goldberger, A. and S.Y. Lau. "Understanding datacomm protocols be examining their structures." EDN, March 3, 1983:109-118.

17. Honeywell Information Systems Inc. Control Program-Six (CP-6) Monitor Services Reference Manual. 1983.

18. Hsieh, W. and G. Gitman. "Routing Strategies in Computer Networks." Computer, June 1984:46-56.

19. Hurst, M. "Sharing data between microcomputers on a local net." Data Communications, September 1984:139-148.

20. Irvine, C.A. and M. Overgaard. "P-System network lets dissimilar computers share resources." Electronics, December 1, 1983:133-138.

21. Jepson, T. "Documentation - That Most Onerous of Chores." Computer Design, July 1983:113-120.

22. Joshi, S. and V. Iyer. "Protocols and network-control chips: a symbiotic relationship." Electronics, January 12, 1984:157-163.

23. Mier, E.E. "Data communications for personal computers." Popular Computing, February 1984:99-100.

24. Mullen, J. "In pursuit of industry-specific network standards." Data Communications, Nov. 1983:187-201.

25. Myers, G.J. Composite/Structured Design. New York: Van Nostrand Reinhold Company. 1978. 174p.

26. "Overview of Ethernet - A Local Area Network." TI Professional Computing, July 1984:10-14.

27. Palmer, L. and R. Palmer. "TTNET Asynchronous File Transfer for VMS." PAGESWAPPER, July 1984:34-44.

28. Pevovar, E. and B McGann. "Sorting Through The LAN Morass." Digital Design, November 1982:54-62.

29. "Protocols: The Etiquette of Making Connections." TI Professional Computing, July 1984:6-9.

30. Smith, K. "Local network links machines that are incompatible." Electronics, October 20, 1983:85-86.

31. Thurk, M. and L. Twaits. "Inside DEC's newest networking phase." Data Communications, September 1983:215-223.

32. Witt, M. "An Introduction to Layered Protocols." BYTE, September 1983:385-398.

33. Yourdon, E. Techniques of Program Structure and Design. New Jersey: Prentice-Hall, Inc. 1975. 364p.

34. Yourdon, E. and L.L. Constantine. Stuctured Design. New York: YOURDON inc. 1978. 446p.

APPENDICES

APPENDIX A

TENRATS' Structured Specification

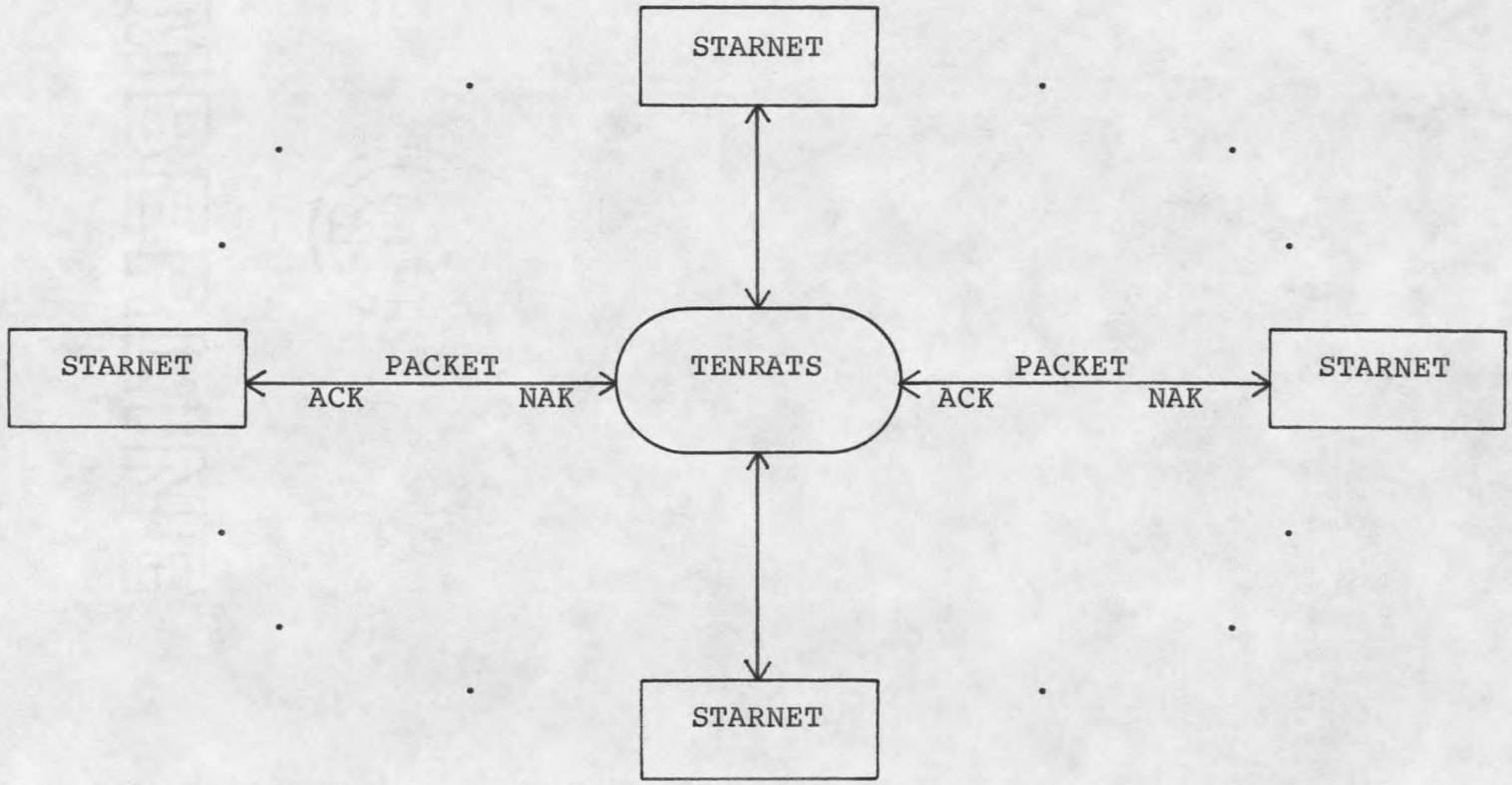


Figure 6.
Context Diagram TENRATS (Central Node) Version A00

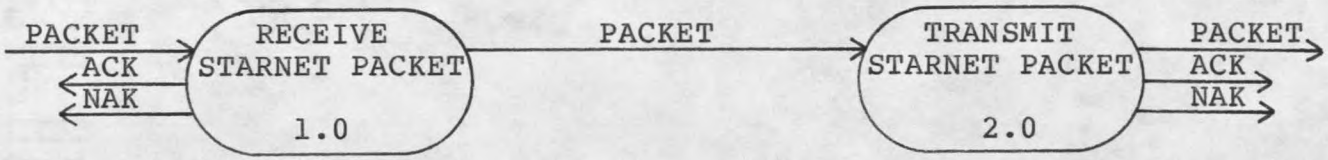


Figure 7.
Diagram 0.0 TENRATS (Central Node) Version A00

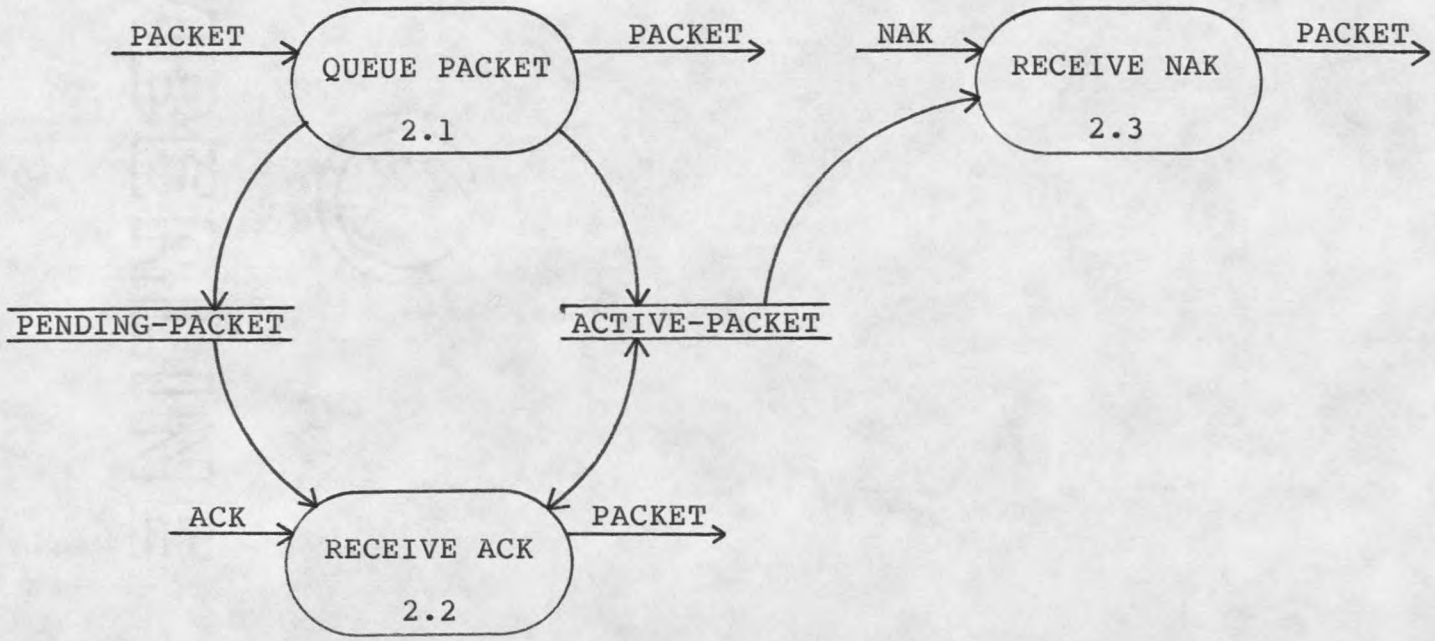


Figure 8.
Diagram 2.0 TRANSMIT STARNET PACKET Version A00

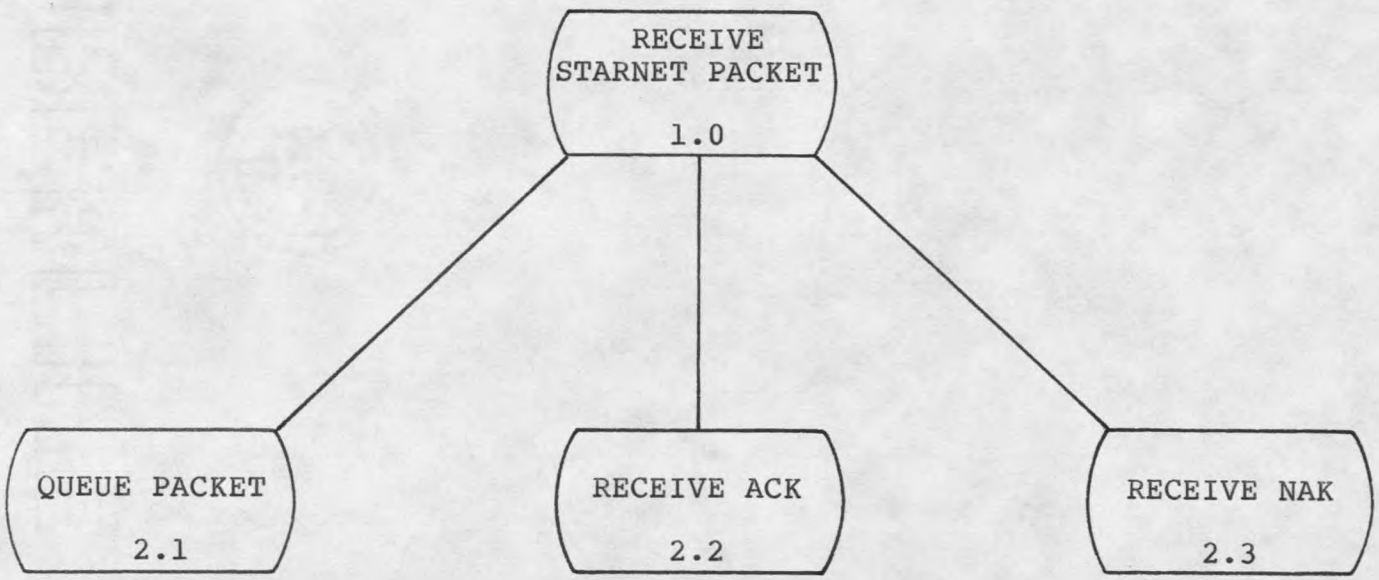


Figure 9.
Chart 0.0 TENRATS (Central Node) Version A00

Process 1.0 - RECEIVE STARNET PACKET

```
1)  Receive PACKET/ACK/NAK from STARNET node.

2)  If ACK,
    Then a) call Process 2.2 - RECEIVE ACK,
         b) goto step 1.

3)  If NAK,
    Then a) call Process 2.3 - RECEIVE NAK,
         b) goto step 1.

4)  If CHECKSUM is correct,
    Then a) transmit ACK to ORIGIN-NODE,
         b) If ID ~= LAST-ID,
            Then a) save ID as LAST-ID,
                 b) call Process 2.1 - QUEUE PACKET,
    Else a) transmit NAK to ORIGIN-NODE.

5)  Goto step 1.
```

Process 2.1 - QUEUE PACKET

```
1)  If DESTINATION-NODE is BUSY,
    Then a) store PACKET in PENDING-PACKET queue,
    Else a) store PACKET in ACTIVE-PACKET queue,
         b) transmit PACKET to DESTINATION-NODE,
         c) save ID as CURRENT-ID,
         d) save ORIGIN-NODE as CURRENT-NODE,
         e) set BUSY flag.

2)  Return.
```

Table 4. TENRATS' Process Specifications

Process 2.2 - RECEIVE ACK

```
1)  If ACK-ID    = CURRENT-ID and
       ACK-NODE = CURRENT-NODE,
    Then a) reset BUSY flag,
         b) If PACKET in ACTIVE-PACKET queue,
            Then a) delete PACKET in ACTIVE-PACKET queue,
         c) If PACKET in PENDING-PACKET queue,
            Then a) read PACKET in PENDING-PACKET queue,
                 b) delete PACKET in PENDING-PACKET queue,
                 c) store PACKET in ACTIVE-PACKET queue,
                 d) transmit PACKET to DESTINATION-NODE,
                 e) save ID as CURRENT-ID,
                 f) save ORIGIN-NODE as CURRENT-NODE,
                 g) set BUSY flag.

2)  Return.
```

Process 2.3 - RECEIVE NAK

```
1)  Reset BUSY flag.

2)  If PACKET in ACTIVE-PACKET queue,
    Then a) read PACKET in ACTIVE-PACKET queue,
         b) transmit PACKET to DESTINATION-NODE,
         c) set BUSY flag.
    Else a) If PACKET in PENDING-PACKET queue,
            Then a) read PACKET in PENDING-PACKET queue,
                 b) delete PACKET in PENDING-PACKET queue,
                 c) store PACKET in ACTIVE-PACKET queue,
                 d) transmit PACKET to DESTINATION-NODE,
                 e) save ID as CURRENT-ID,
                 f) save ORIGIN-NODE as CURRENT-NODE,
                 g) set BUSY flag.

3)  Return.
```

Table 4. TENRATS' Process Specifications - Continued

A

ACK	'ACK' + ACK-ID + ACK-NODE * positive ACKnowledgement *
ACK-ID	ID * ID of PACKET being ACKed *
ACK-NODE	ORIGIN-NODE * ORIGIN-NODE of PACKET being ACKed *
ACTIVE-PACKET	0 { PACKET } 1

B

BUSY	0 { TRUE | FALSE } N * Array of boolean flags, one/node *

C

CHECKSUM	1 character * PACKET's verification character *
CURRENT-ID	0 { ID } N * Array of IDs transmitting *
CURRENT-NODE	0 { ORIGIN-NODE } N * Array of ORIGIN-NODEs transmitting *

D

DESTINATION-NODE	8 characters * PACKET's destination node name *

I

ID	3 characters (000-999) * PACKET's identification number *

Table 5. TENRATS' Data Dictionary

L

LAST-ID	0 { ID } N * Array of IDs transmitting *

N

NAK	'NAK' * Negative AcKnowledgement *

O

ORIGIN-NODE	8 characters * PACKET's origin node name *

P

PACKET	ID + ORIGIN-NODE + ORIGIN-PROCESS + DESTINATION-NODE + DESTINATION-PROCESS + (COMMAND | MESSAGE) + CHECKSUM
PENDING-PACKET	0 { PACKET } M

Table 5. TENRATS' Data Dictionary - Continued

APPENDIX B

STARNET'S STRUCTURED SPECIFICATION

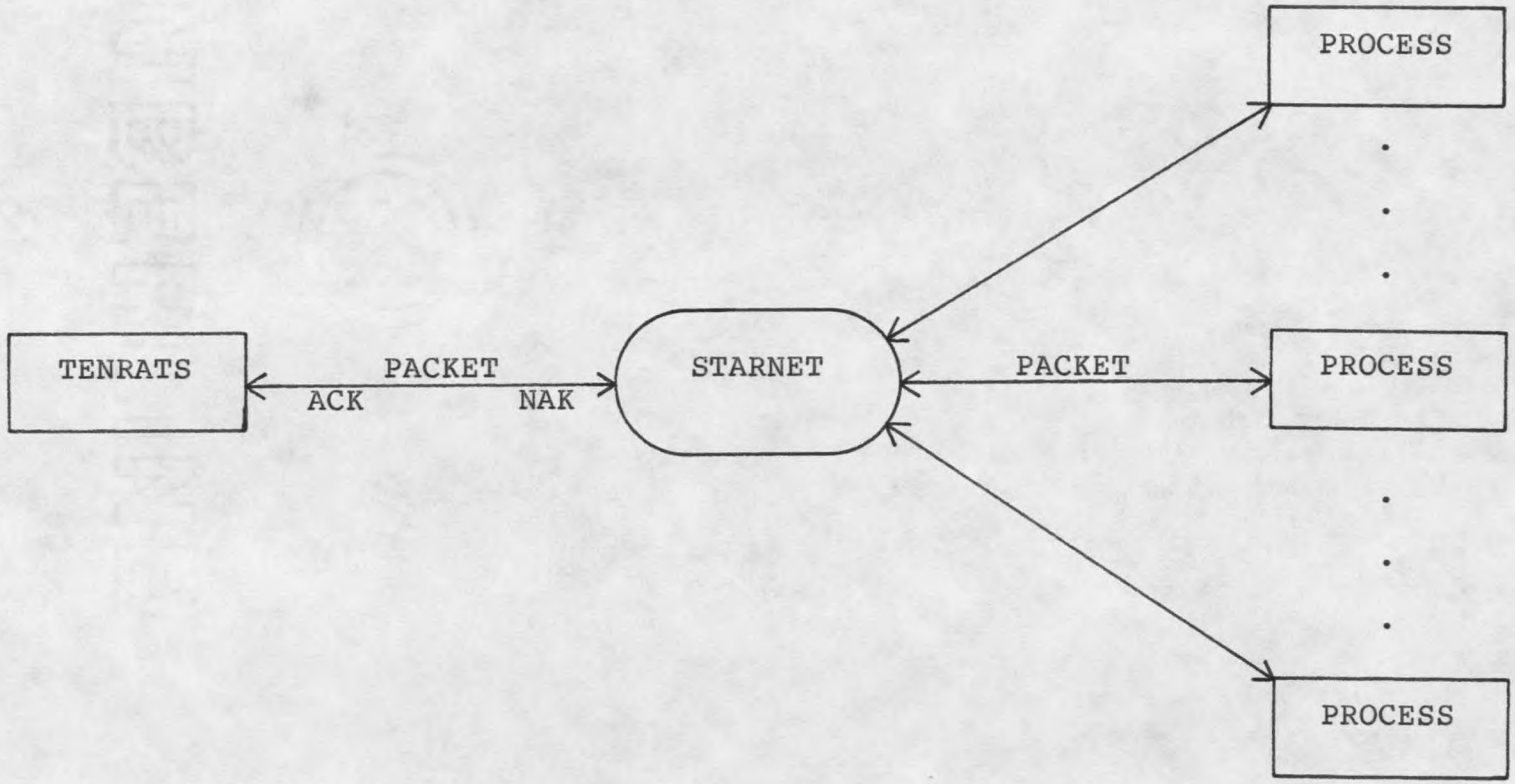


Figure 10.
Context Diagram STARNET (Point Node) Version A00

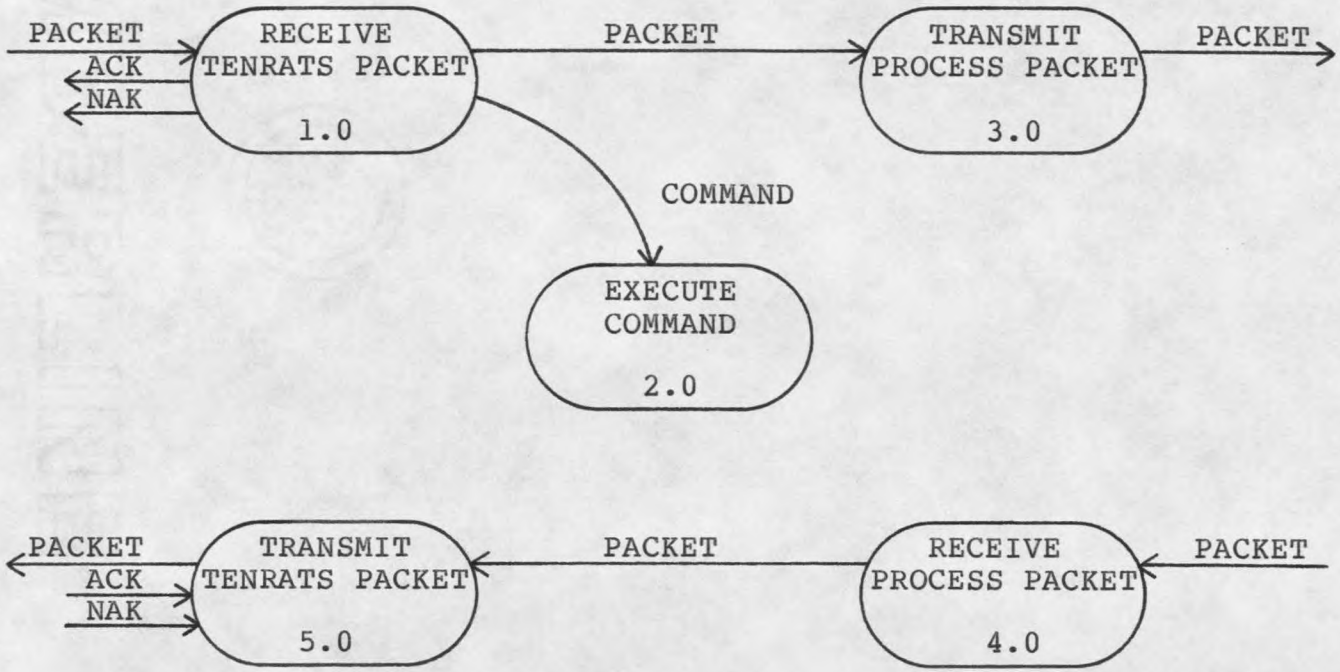


Figure 11.
Diagram 0.0 STARNET (Point Node) Version A00

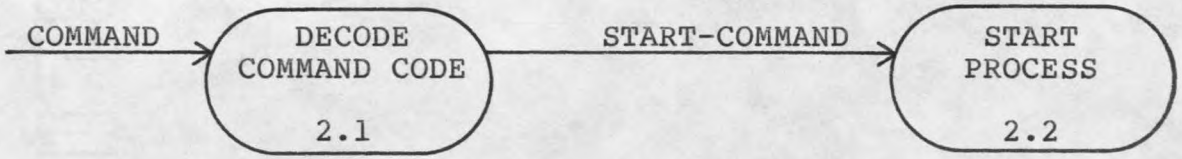


Figure 12.
Diagram 2.0 EXECUTE COMMAND Version A00

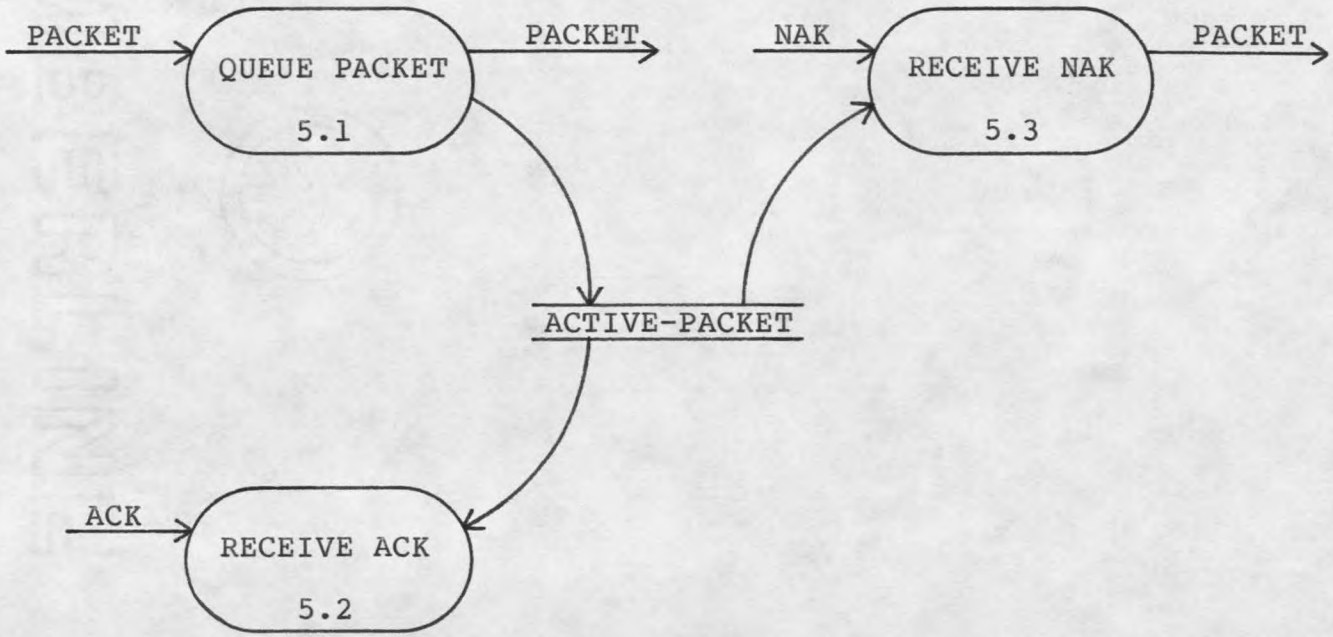


Figure 13.
Diagram 5.0 TRANSMIT TENRATS PACKET Version A00

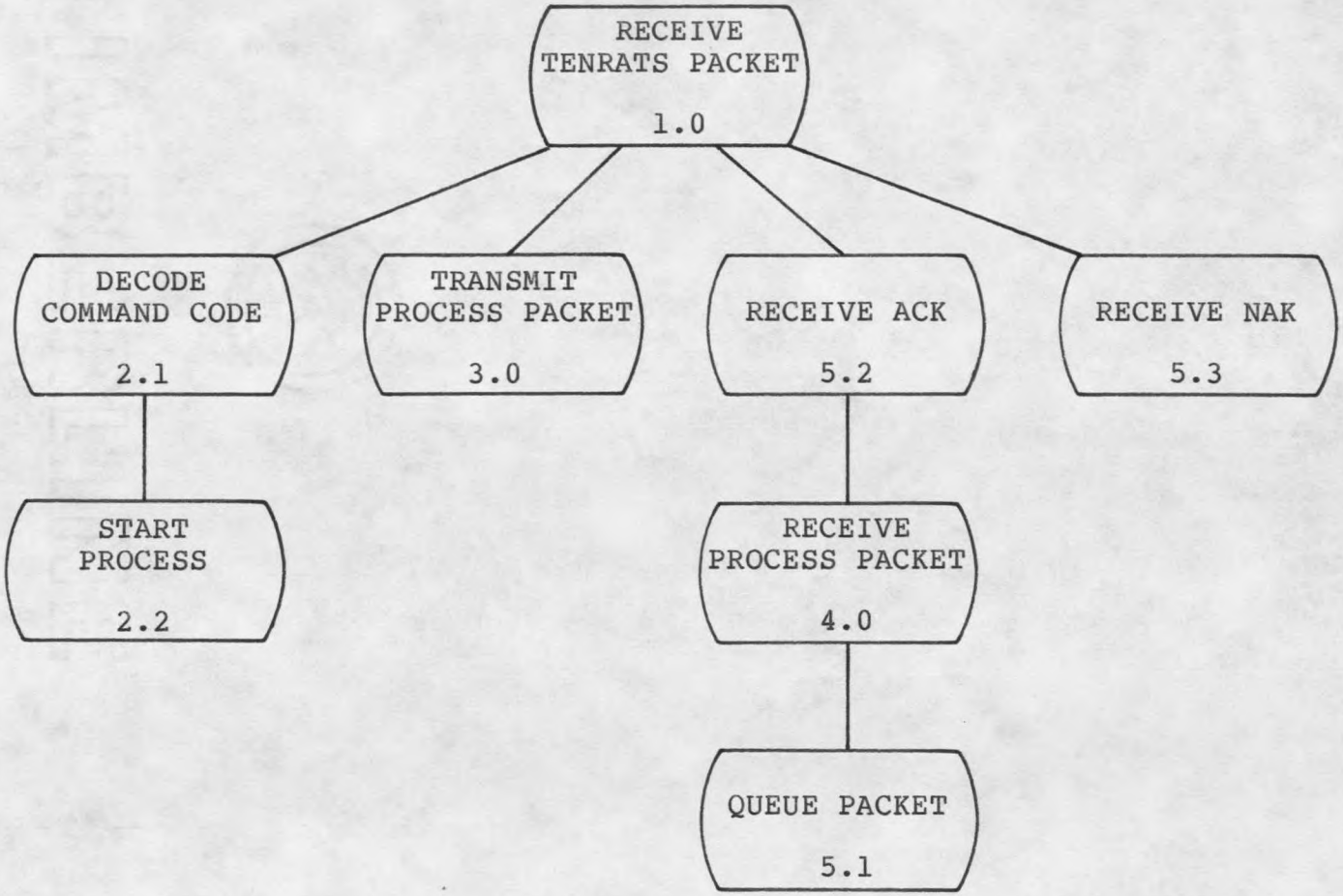


Figure 14.
Chart 0.0 STARNET (Point Node) Version A00

Process 1.0 - RECEIVE TENRATS PACKET

```
1)  Receive PACKET/ACK/NAK from TENRATS.

2)  If ACK,
    Then a) call Process 5.2 - RECEIVE ACK,
         b) goto step 1.

3)  If NAK,
    Then a) call Process 5.3 - RECEIVE NAK,
         b) goto step 1.

4)  If CHECKSUM is correct,
    Then a) If ID = LAST-ID-IN and
               ORIGIN-NODE = LAST-NODE-IN
            Then a) transmit ACK to TENRATS,
                 b) goto step 1.
            Else a) save ID as LAST-ID-IN,
                 b) save ORIGIN-NODE as LAST-NODE-IN,
                 c) transmit ACK to TENRATS,
                 d) goto step 5.
    Else a) transmit NAK to TENRATS,
         b) goto step 1.

5)  Unmask characters:
    <DLE> <char> --> <x'7F'>AND<char>.

6)  If DESTINATION-PROCESS = 'STARNET ',
    Then a) call Process 2.1 - DECODE COMMAND CODE,
    Else a) call Process 3.0 - TRANSMIT PROCESS PACKET.

7)  Goto step 1.
```

Process 2.1 - DECODE COMMAND CODE

```
1)  Case COMMAND-CODE,

    'START   ' a) call Process 2.2 - START PROCESS.
```

Table 6. STARNET's Process Specifications

Process 2.2 - START PROCESS

1) Start process specified by START-ARGUMENT.

2) Make ORIGIN-NODE and ORIGIN-PROCESS available to started process.

Process 3.0 - TRANSMIT PROCESS PACKET

1) Transmit PACKET to DESTINATION-PROCESS.

Process 4.0 - RECEIVE PROCESS PACKET

1) Receive PACKET from a process.

2) Load ID with NEXT-ID and save as LAST-ID-OUT.

3) Increment NEXT-ID modulo 1000.

4) Load ORIGIN-NODE with NODE-NAME.

5) Mask <CR> and <DLE>:
   <CR> --> <DLE> <x'80'>OR<CR>,
   <DLE> --> <DLE> <x'80'>OR<DLE>.

6) Compute and load CHECKSUM.

7) Call Process 5.1 - QUEUE PACKET.

Table 6. STARNET's Process Specifications - Continued

Process 5.1 - QUEUE PACKET

1) Store PACKET in ACTIVE-PACKET queue.

2) Transmit PACKET to TENRATS.

Process 5.2 - RECEIVE ACK

1) If ACK-ID = LAST-ID-OUT,
   Then a) call Process 3.0 - RECEIVE PROCESS PACKET.

Process 5.3 - RECEIVE NAK

1) Read PACKET from ACTIVE-PACKET queue.

2) Transmit PACKET to TENRATS.

Table 6. STARNET's Process Specifications - Continued

A

ACK	'ACK' + ACK-ID + ACK-NODE * positive ACKnowledgement *
ACK-ID	ID * ID of PACKET being ACKed *
ACK-NODE	ORIGIN-NODE * ORIGIN-NODE of PACKET being ACKed *
ACTIVE-PACKET	0 { PACKET } 1

C

CHECKSUM	1 character * PACKET's verification character *
COMMAND	COMMAND-CODE + COMMAND-ARGUMENT
COMMAND-ARGUMENT	0-462 characters * command dependent argument(s) *
COMMAND-CODE	10 characters * name of command to execute *

D

DESTINATION-NODE	8 characters * PACKET's destination node name *
DESTINATION-PROCESS	10 characters * PACKET's destination process name *

I

ID	3 characters (000-999) * PACKET's identification number *

Table 7. STARNET's Data Dictionary

L

LAST-ID-IN	ID * last ID received *
LAST-ID-OUT	ID * last ID transmitted *
LAST-NODE-IN	ORIGIN-NODE * last ORIGIN-NODE received *

M

MESSAGE	0-472 characters * variable length message *

N

NAK	'NAK' * Negative AcKnowledgement *
NEXT-ID	ID * next ID to transmit *
NODE-NAME	8 characters * STARNET's node name *

O

ORIGIN-NODE	8 characters * PACKET's origin node name *
ORIGIN-PROCESS	10 characters * PACKET'S origin process name *

Table 7. STARNET's Data Dictionary - Continued

P

PACKET	ID + ORIGIN-NODE + ORIGIN-PROCESS + DESTINATION-NODE + DESTINATION-PROCESS + (COMMAND | MESSAGE) + CHECKSUM

S

START-COMMAND	START-CODE + START-ARGUMENT
START-ARGUMENT	* computer dependent argument *
START-CODE	'START '

Table 7. STARNET's Data Dictionary - Continued

DATE	ISSUED TO
[redacted]	
	R