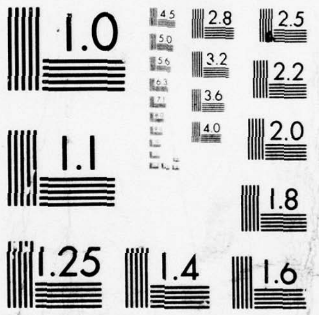
AD-A063 092 CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAI--ETC F/G 17/2
A SURVEY OF THE PROPERTIES OF COMPUTER COMMUNICATION PROTOCOLS.--ETC(U)
SEP 78 A E ITZKOWITZ
UNCLASSIFIED CERL-TR-O-1 NL

1 OF 1
AD
A063092

END
DATE
FILMED
3--79
DDC


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

construction
engineering
research
laboratory

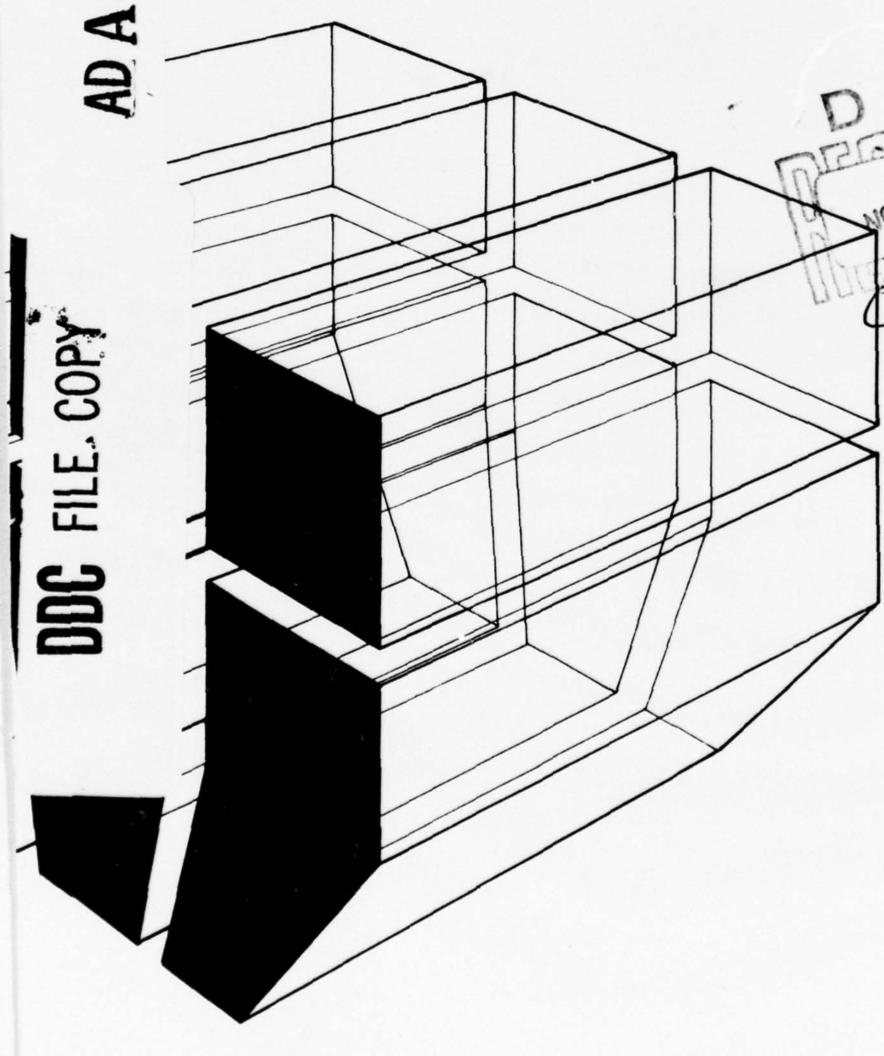
LEVEL III

12

TECHNICAL REPORT O-1
September 1978

A061647
VOL II

A SURVEY OF THE PROPERTIES OF COMPUTER COMMUNICATION PROTOCOLS VOLUME I: THE FUNCTION, PROPERTIES, SPECIFICATION, AND ANALYSIS METHODS OF COMPUTER COMMUNICATION PROTOCOLS



by
A. E. Itzkowitz

CERL

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT NUMBER~~ (14) CERL-TR-0-1 ✓ Volume	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER (9)
4. ~~TITLE (and Subtitle)~~ (6) A SURVEY OF THE PROPERTIES OF COMPUTER COMMUNICATION PROTOCOLS, ~~VOL~~ I. THE FUNCTION, PROPERTIES, SPECIFICATION, AND ANALYSIS METHODS OF COMPUTER COMMUNICATION PROTOCOLS.		5. ~~TYPE OF REPORT & PER~~IOD COVERED FINAL rept.
		6. PERFORMING ORG. REPORT NUMBER
7. ~~AUTHOR(s)~~ (10) Avrum E./Itzkowitz		8. CONTRACT OR GRANT NUMBER(s) (17) 02
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. ARMY ENGINEER CONSTRUCTION ENGINEERING RESEARCH LABORATORY P.O. Box 4005, Champaign, IL 61820		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 4A762725AT11-2-209
11. CONTROLLING OFFICE NAME AND ADDRESS		12. ~~REPORT DATE~~ (11) Sep~~tember 19~~78
		13. NUMBER OF PAGES 41 (12) 44p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Copies are obtainable from National Technical Information Service
Springfield, VA 22151

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer communication protocols
graphical techniques
remote access

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is a two-part study on the properties of computer communication protocols. This volue focuses on the function and properties of current computer communication protocols. Many different approaches to the specification and analysis problem, ranging from textual description to formal graphical techniques, are presented and discussed.

405 279

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 78 UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This investigation was performed for the Engineer Information and Data Systems Office (EIDSO), Office of the Chief of Engineers (OCE), under Project 4A762725AT11, "Engineering Software Development Methods"; Task 2, "Data and Language Structures"; Work Unit 209, "Design of a Machine Protocol Language." The OCE Technical Monitor was Mr. R. McMurrer.

The investigation was performed by the Computer Services Branch (SOC), Support Office (SO), U.S. Army Construction Engineering Research Laboratory (CERL). Personnel directly involved in the study were Mr. A. Itzkowitz and Ms. L. Lawrie of SOC.

Mr. W. Schmidt is Chief of SOC, and Mr. W. Assell is Chief of SO. COL J. E. Hays is Commander and Director of CERL, and Dr. L. R. Shaffer is Technical Director.

CONTENTS

A SURVEY OF THE PROPERTIES OF COMPUTER COMMUNICATION PROTOCOLS VOLUME I: THE FUNCTION, PROPERTIES, SPECIFICATION, AND ANALYSIS METHODS OF COMPUTER COMMUNICATION PROTOCOLS

1 INTRODUCTION

Background

The ability to access computer facilities from a remote location has become increasingly important. This remote access capability makes some previously expensive uses of computers economically effective, and makes other uses, which until recently were nearly impossible, both feasible and practical. Remote access makes computer facilities available where they were previously unavailable, reduces the cost of data processing, and enables the timely use of information by organizations and persons separated by large distances.

A computer facility which supports any type of remote access capability contains a subsystem called the communications network. This network contains the communication links necessary to connect machines and terminals at different locations. Each point of the network to which either a terminal or computer may be connected is called a node. Whether the network has been designed primarily for machine to machine communication, machine to terminal communication, or terminal to terminal communication, there must be some agreement which defines how the nodes may communicate over the network. This agreement is called a communication protocol. The protocol may be simple or complex, depending on the sophistication and function of the devices which will use it.

Certain necessary functions and capabilities of communication networks are well defined--for example, the remote batch job entry facility. Though the functions are defined in principle, their actual implementations are not. It is rare that two computers produced by different manufacturers have any communication facility implemented in the same way. Even systems of a single manufacturer may differ. It is unlikely that these different implementations can be standardized in the near future. Therefore, an organization which must use the services of more than one remote facility must be prepared to support many communication protocols. One major obstacle to this type of support is the lack of any standardized method of describing protocols. Each designer develops his/her own notation, and specifications are often ambiguous, incomplete, and/or cryptic.

Objective

The objective of this report is to survey the function, properties, specification, and analysis methods of computer communication protocols; to discuss the criteria which have been used for judging specifications; and to recommend a general approach to the problem of specifying communication protocols.

Outline of Report

Chapter 2 discusses aspects of data communications and properties of protocols. Chapter 3 surveys some currently used methods of protocol specification, modeling, and verification. Chapter 4 discusses the protocol specification criteria.

2 PROTOCOL FUNCTION AND DESIGN

The issues of data communication protocols can be divided into two levels: functional issues and design issues. Functional issues deal with the execution time problems of data communications, such as error recovery and flow control. Design issues deal with the reliability and efficiency of the overall protocol design. These issues must be completely resolved before the protocol is specified.

Functional Issues

There are three phases in a data communication procedure: connection, message transfer, and termination. The phases may be distinct, or they may appear to merge together, depending on the function and topology of the communication network. There are many functional issues with which a protocol must deal. Although these issues are interrelated, they have been divided for this discussion as follows:

1. Control of data transfers
2. Information coding
3. Flow control
4. Message framing
5. Error recovery
6. Protocol layering.

Control of Data Transfers

The protocol must specify how each party should control the transfer of data. Three elements should be considered: message format, control information, and acknowledgements. The message format specifies the form of the information contained in the message. The format will typically consist of some leading control information (also called a header), the data, and some trailing control information. Embedded in the header and trailer is control information, which is typically used to specify the source and destination of the message, and other information needed by the receiver (and/or the communication network) to process the message. Redundant information is also generally provided for use in error recovery (see *Error Recovery* section). The acknowledgement, or hand shaking, specifies the interaction of messages, how the receipt of a message is acknowledged, when a particular kind of message may be sent, etc.

Information Coding

Information coding deals with the meaning of the message, i.e., the convention used to interpret the bits transmitted. The protocol must either explicitly or implicitly specify the transformations needed to be performed on the message in order to convert it to and from its transmitted format to the internal machine representations. There are some well-defined code sets, such as ASCII,[1] BCD, or EBCDIC. Other conventions exist, and others may also be useful. The message may be bit-oriented, so that the bit combinations do not conform to some character-oriented code set, for example, transmission of binary data, or use of the SDLC[2] protocol. Conventions must be observed that specify which bit combinations, if any, have special meanings as control characters. In addition, if a transparency mode is necessary (a mode in which all bit combinations may be transmitted as valid data), conventions for the entry and exit from this mode must be observed.

Flow Control

The flow control properties of a communication protocol depend on several factors, the most important of which are topology and reliability of the communication network, and the functional level of the protocol. Protocols may exist at different functional levels. For example, in a message switching network such as ARPANET,[3] the machines at the nodes specify the destination of the message, but the internal network processors must route the messages from node to node until the intended receiver accepts the message. The user does not see the internal network protocol, which specifies how messages are routed.

The topology of the communication network partially dictates the expected efficiency of the communication links. A multipoint or multidrop data link will require more control overhead to route messages than a point to point link. However, if the frequency of communication between individual nodes is low, the extra control overhead can increase the utilization of the data link. If there is frequent communication between nodes, the extra overhead may cause contention problems, decreasing the utilization of the link.

---

1 *American National Standard Procedures for the Use of the Communication Control Characters of American National Standard Code for Information Interchange in Specified Data Communication Links*, ANSI X3.28 (ANSI, 1976).

2 R. A. Donnan and J. R. Kersey, "Synchronous Data Link Control: A Perspective," *IBM Systems Journal*, Vol 13, No. 2 (1974), pp 140-162.

3 Stephen D. Crocker, John F. Heafner, Robert Metcalfe, and Jonathan Postel, "Function-Oriented Protocols for the ARPA Computer Network," *Proceedings of the Spring Joint Computer Conference*, Vol 40, (American Federation of Information Processing Societies [AFIPS], 1972), pp 271-279.

Direction control is another factor. There are three types of direction control: simplex, half duplex, and full duplex. A simplex link allows transmission in one direction only. A half duplex link allows transmission in both directions, but only in one direction at a time. A full duplex link supports simultaneous bidirectional communication, and is often modeled as two simplex links. A full duplex link requires a protocol capable of handling simultaneous two-way transmission in order to use the link efficiently.

The reliability of the data links will affect the flow of data for different protocols. If the network is highly reliable, and only a small number of errors is expected, then the protocol will probably specify error detection and retransmission schemes. If the reliability is low, the protocol will probably specify a more complex error recovery scheme, which can allow errors to be corrected in each message without retransmission. If the link is reliable, the number of expected retransmissions will be low, so that the former method is more efficient. If the link is unreliable, a large number of retransmissions could be expected, and the latter scheme will be more efficient.

Flow control must also deal with the problems of source and destination synchronization. There must be a mechanism which prevents a source from producing messages faster than the destination can process them. Some protocols treat this as an error condition, but this could cause extra transmissions and overheads. Another method is the preallocation of buffer space, which increases control overhead.

Message Framing

It must be possible to determine the start and end of a message. Generally, some pattern of bits, sometimes referred to as the synchronization pattern, precedes the message. This pattern also forces the framing of the individual characters of the message. If the message falls one bit out of sync, it can become completely garbled; however, this issue is generally only important in low-level protocols.

Error Recovery

If all communication links were error free, and all nodes of a communication network were always available, the design of protocols would be straightforward and fairly simple. However, a major problem of protocol design and specification is error detection and recovery. The treatment of error conditions directly affects the number of message retransmissions, buffer requirements at the nodes, line throughput and utilization, and message delays. The different errors which must be detected and recovered include:

1. Transmission errors
2. Sequence errors
3. Synchronization violations
4. Long-term failure.

Transmission Errors. Transmission errors change individual bits in a message. There are several ways to detect and recover from transmission errors. Some systems require a destination to repeat the message to the sender. If the message is returned as it was sent, then it is assumed to be correct. However, most networks cannot afford this amount of overhead, so error-checking information is generally added to each message. Vertical Redundancy Checking (VRC), also referred to as parity, is one such method. VRC adds one bit to each character (byte) of the message, and requires that the total number of bits set in each character be either always odd or always even. For example, using odd parity, if the number of bits set in the character is odd, then the VRC bit should be set to zero; otherwise, it should be set to one. This will detect all one-bit errors in a character. This form of error detection can be extended by adding more VRC bits per character. For example, there is a four-out-of-eight code, where only four bits in an eight-bit character may set to one. The only errors not detected by this method are those which cause an equal number of zero and one bits to change value. There are also codes which can correct errors, in effect giving a multiple-bit pattern to a one-character mapping.

The entire message can be checked similarly. Logitudinal Redundancy Checking (LRC), also called checksum or block checking, sums the character codes for the entire message, maps the sum into a single character, and includes this block-checking character (BCC) in the message. The destination node performs the same operation on the incoming message and compares the checksums. Cyclic Redundancy Checking (CRC) is another method of message checking which uses polynomial division of the message to compute the checksum. These methods usually detect false message framing, since it is unlikely that a falsely framed message will have correct checksums. VRC and LRC are commonly used together in data communication systems and are effective when the expected error rate is fairly low.

Sequence Errors. Sequence errors occur when messages arrive at the destination in a different order than they were transmitted from the source. An example of this is a packet switching network, where the messages are divided into smaller packets, and each packet is routed separately; however, networks where the messages are not divided into packets can also exhibit this kind of error. For example, in protocols which do not require the acknowledgement of the previous message before sending another message, the second message could arrive first.

Another type of sequence error occurs when an expected message is missing or duplicated. A message can be duplicated if the protocol specifies retransmission or no acknowledgement, and the destination is either slow to respond or the acknowledgement is delayed.

Synchronization Violations. Synchronization violations occur when the communicating nodes do not agree on the state of the communication; i.e., both may be trying to transmit or receive. They may be synchronized as source and destination, but the message being transmitted may be different from the message that the destination is prepared to receive. This type of error is generally short-term, and can be caused by a short-term failure of the communication link or a node.

Another type of error occurs when a node is slow to respond or when a message is delayed. If one node is waiting for a message and it is not delivered within some specified time, the destination must be able to perform some recovery action. This is commonly known as a time-out. When this kind of error occurs, synchronization must be re-established whenever the transmitting node becomes available. If the node does not become available reasonably quickly, the condition is then classified as a long-term error.

Long-Term Failure. The previous section dealt with short-term synchronization problems. If communication between two nodes is lost for a long period of time, then some specific recovery action may be specified. If the failure is in one link of the network, then messages may be rerouted around that link. If the node itself has failed, then the other communication nodes should be notified so that they may perform some recovery action. For example, if the protocol is for remote batch job entry, jobs may be able to run on another facility in the network.

Protocol Layering

In a network like ARPANET,[4] host to host protocols are layered on host to IMP protocols, which are layered on IMP to IMP protocols. Layering may exist even in a simple point to point protocol. For example, the Houston Instrument plotters[5] are interfaced to the CDC UT200 remote batch entry terminal by layering them on top of the existing line printer protocol. Any line which begins with the characters '+:' is diverted to the plotter instead of the printer. The lower-level protocol does not need to be aware of the higher-level protocol above it.

---

[4] Stephen D. Crocker, John F. Heafner, Robert Metcalfe, and Jonathan Postel, "Function-Oriented Protocols for the ARPA Computer Network," *Proceedings of the Spring Joint Computer Conference*, Vol 40 (1972), pp 270-271.

[5] *Batch Terminal Controller-Complot BTC-7-200 Instruction Manual* (Houston Instrument, Inc., revised May 1972).

One problem with layered protocols is error recovery. The fact that a message is correct at one level does not mean that it will be correct at a higher level. Each level must be prepared to handle error conditions, and this may lead to additional overheads. Recovery at the low levels can reduce the number of errors detected at higher levels, but will not eliminate them. Pouzin[6] concludes that there is no simple criterion to determine at which levels error recovery should appear.

Design Issues

Depending on the individual communication network and the intended use of its protocols, the particular functional issues which must be considered and their solutions can differ from protocol to protocol. The design issues (listed below), however, are consistent and must be considered for every protocol.

1. Verification
2. Identification of protocol inadequacies
3. Detection of deadlocks
4. Detection of critical races.

It must be possible to verify that a protocol performs as required. It is necessary to identify the situations in which the protocol is inadequate or incurs unacceptable overheads, and be able to redesign it when necessary. Possible deadlock situations must be detected: for example, whether two nodes can go into a receive state and never return to the send state. Similarly, critical races must be detected (those conditions which may cause different actions to take place, depending on how quickly a message is delivered or how a node responds). The error recovery procedure chosen must also yield a stable system; for example, is the protocol self-synchronizing (that is, does it tend to synchronize itself even if it was initially not synchronized).

6 Louis Pouzin, "An Integrated Approach to NETWORK Protocols," *Proceedings of the National Computer Conference* (1975).

3 CURRENT PROTOCOL SPECIFICATION METHODS

The design, verification, and implementation of communication protocols are interrelated. Because of this interrelationship, the specifications and models which have been used in any one of these activities may also be considered useful for the others. The following general schemes have been used during at least one of the activities of protocol design and analysis.

1. Text
2. Communication control graphs
3. General machine descriptions
4. Synchronization models
5. Linguistic models
6. Specialized machine descriptions.

Text

Natural language text is usually the first method employed to specify almost anything. Its advantages are generality and expressibility. However, the completeness, conciseness, and clarity of a textual protocol specification depend totally on the author. Often, many types of graphical aids accompany a textual description for the sake of clarity.

The difficulty with textual specifications stems from the generality of natural languages. Two persons reading the same text can interpret the protocol differently. The definition may be ambiguous, its completeness is usually questionable, and it is often quite lengthy. One textual description of the BSC protocol,[7] which is a reasonably established and simple line control discipline, is 36 pages long, but may not be complete. It is unsuitable as an implementation guide because of its length.

Communication Control Graphs

Graphical methods have long been used in conjunction with text to clarify the exchange of messages and the different possible interactions. There are several different graphical representations which are fundamentally the same.

[7] J. L. Eisenbies, "Conventions for Digital Data Communication Link Design," *IBM Systems Journal*, Vol 6, No. 4 (1967), pp 267-392.

Time Line Diagrams

The time line diagram is laid out in columns, with one column for each communicating part or station. The interactions of messages and time flow from the top to the bottom of the diagram. The messages which may be sent or received at each interaction are shown as branches of the diagram within the column of the active party. When a message is complete, the diagram crosses from the column of the message sender to the column of the message receiver. To illustrate the differences between the graphical methods, consider a two-station half-duplex communication which uses the control characters SOH, STX, ETX, ACK, and NAK. Figure 1 is a time line diagram of a small fragment of the protocol.[8] The branch around the SOH character indicates that it is optional.

ANSI Representation

The ANSI representation collapses the multiple column notation of the time line diagram into a single flow by marking the line turnarounds (those positions in the graph where the flow of messages changes directions). Figure 2 is the ANSI representation of the example protocol fragment.[9] The graph is read from left to right.

State Transition Graph

The state transition graph is an extension of the ANSI representation. The system states of the communicating stations are added as circles on the graph. Figure 3 is the state transition graph of the example protocol fragment.[10] The state transition graph notation may be converted into tables that define a finite state machine which can recognize the valid messages of the protocol.

Summary of Communication Control Graphs

The graphical methods clearly show the flow of information over the communication link, and may show the state of the sender or receiver. These methods can become complicated as the complexity of the protocol and the number of stations increase. They are incomplete as a sole source of protocol specification, ignoring the transformations performed on the messages, such as code translation and parity. They may be used to show which messages are possible, but not how, why, or when.

---

[8] Byron W. Stutzman, "Data Communication Control Procedures," *Computer Surveys*, Vol 4, No. 4 (December 1972).
[9] Byron W. Stutzman.
[10] Byron W. Stutzman.

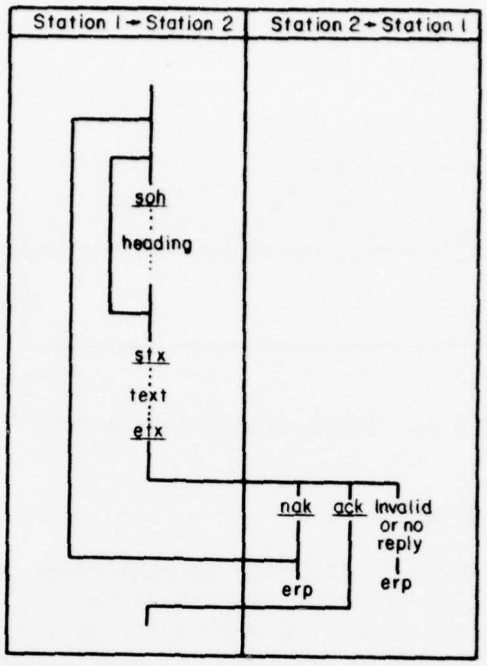


Figure 1. Time-line diagram.

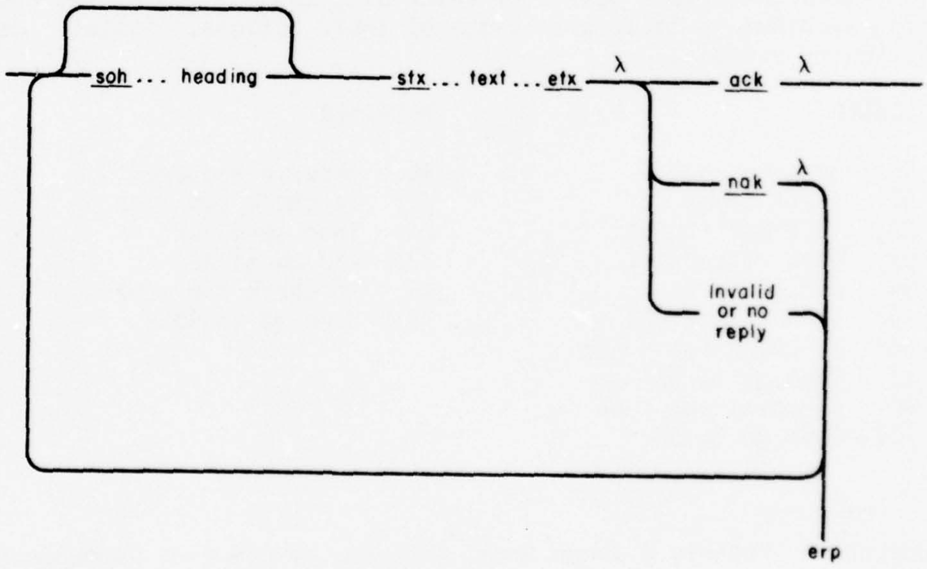


Figure 2. ANSI representation.

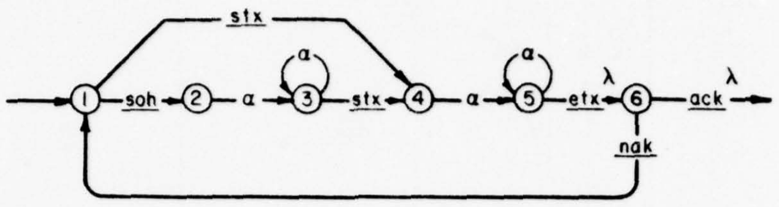


Figure 3. State transition graph.

General Machine Descriptions

Since machines are used in the data communication process, one way to describe a communication protocol is to describe the actions of a machine which can communicate using the protocol. In this context, almost any method that has been used to describe computer hardware or software can be used to describe a protocol. Generally, to describe a protocol in this way, two machines must be described: the sender and the receiver. Often, both the sender and receiver are the same machine.

Sender-Receiver Interactions

One straightforward method of describing the actions of two communicating machines is to compare lists of their actions. Postel[11] uses the following example.

SENDER

S0: Send message
S1: Start timer
S2: If timer runout
S3: Then retransmit
S4: And go to S1
S5: If ACK received
S6: If check sum = bad
S7: Then go to S2
S8: If check sum = ok
S9: Then go to S0

RECEIVER

R0: Receive message
R1: If check sum = ok
R2: Then send ACK
R3: And go to R0
R4: If check sum = bad
R5: Then go to R0

[11] Jonathan B. Postel, *A Graph Model Analysis of Computer Communication Protocols*, Ph.D. Dissertation, NTIS No. AD 777 506/H (UCLA, January 1974).

This example is shown to be a simple positive acknowledgement protocol, but little other information is revealed. The description itself is a type of structured text, and suffers from the problems of textual descriptions. It is adequate as a general description meant to give the flavor of a communication protocol, but not as a complete description for use as an implementation guide.

Flow Charts

Flow charts, used to describe both computer hardware and software, clearly show the flow of control of an algorithm. For example, the protocol used by the CDC 200 user terminal is described in a detailed four-page foldout flow chart in the *200 User Terminal Hardware Reference Manual*.[12]

The problems with flow charts are similar to the problems with structured text. They may not be sufficiently detailed (for example, leaving critical implementation details unspecified), or they may become too complex to understand.

Finite State Machines

As previously stated, a state transition diagram can be turned into tables which define a finite state machine that can recognize the protocol. Finite state techniques have been used for both protocol descriptions and implementations.[13-17] The basic procedure is to make a table consisting of one column for each valid character in the message and one

---

12 *200 User Terminal Hardware Reference Manual*, Publication No. 82128000 (Control Data Corporation, June 1969).

13 Dennis M. Birke, *State Transition Programming Techniques and Their Use in Producing Teleprocessing Device Control Programs*, M. S. Thesis (University of Pittsburgh, 1971).

14 Dennis M. Birke, "State-Transition Programming Techniques and Their Use in Producing Teleprocessing Device Control Programs," *Proceedings of the 2nd Symposium on Problems in the Optimization of Data Communications Systems* (Association for Computer Machinery [ACM], October 20-22, 1971), pp 21-31.

15 Dines Bjorner, "Finite State Automation--Definition of Data Communications Line Control Procedures," *Proceedings of the Fall Joint Computer Conference* (International Federation for Information Processing [IFIP], 1970), pp 477-491.

16 Gregor V. Bochman, "Communication Protocols and Error Recovery Procedure," *Proceedings of the ACM SIGCOMM SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 45-56.

17 Byron W. Stutzman, "Data Communication Control Procedures," *Computer Surveys*, Vol 4, No. 4 (December 1972).

row for each state. The valid transitions from each state to its successor state are marked in the states row in the column of the character that will make it change states. The action that the machine must perform as it makes the transition, such as check message parity, is also marked in the table. All unmarked transitions are invalid. Figure 4 is an example of the state transition matrix. Bochman[18] extends the notation to specifically show whether the action performed is a local action, a send action, or a receive action.

EVENT

CURRENT STATE	soh	stx	α	etx	ack	nak	Invalid or no reply
1	A,2	B,4					
2			C,3				
3		E,4	D,3				
4			F,5				
5			G,5	H,6			
6					I,1	J,1	

A-J ROUTINES TO BE PERFORMED UPON TRANSITION TO THE NEXT STATE

α – TEXT CHARACTERS

Figure 4. State transition matrix.

The problem with the finite state description is that the actions the protocol machine is to perform are not formally specified. Also, there are no guarantees that finite state machines are sufficient to describe all protocols.

Computer Programs

The most complete description of a protocol machine which exists as a computer program is the computer program itself. Most protocol descriptions are transmitted as programs. Unfortunately, however, most protocol machines are programmed in assembly language, which requires

[18] Gregor V. Bochman, pp 45-56.

the user to have specific knowledge of the implementation hardware in order to transfer it to another machine. Also, since machine architectures vary widely, the description may need drastic revision before the protocol will function on another machine. Protocol machines generally require the use of interrupt processing, and so far there is no commonly used high-level language which handles asynchronous processing clearly and efficiently. However, this is currently an active research area. Brinch Hansen[19] has developed a language based on Pascal[20] called Concurrent Pascal. This language is designed to handle systems of communicating asynchronous processes and has a computer operating system written into it. Hansen and Lindahl[21] have specified a similar language called Real-Time Pascal. Wirth has designed a language called Modula,[22] also based on Pascal, which is oriented toward the DEC PDP-11 computer. The analysis of protocol software is subject to all of the problems of general software analysis and verification.

Summary of General Machine Descriptions

Protocols that are based on general machine descriptions are promising. They can be complete; they are executable, thereby affording some verification of their operation; and they seem to be the shortest path to implementation of a protocol. The problem appears to be that general methods do not yield a clear enough description, independent of some hardware, to be easily usable. Two special forms of machine representations will be discussed in the Specialized Machine Descriptions Section.

Synchronization Models

Recently, there has been much interest in verifying and analyzing the actions of protocols. Synchronization properties have received much attention. Within the context of this work, several different formalisms have evolved which have been used for different analyses. While none of these models seems to be general enough to be used alone as a

---

19 Per Brinch Hansen, *Concurrent Pascal Introduction and Concurrent Pascal Report* (California Institute of Technology, July 1975).

20 Kathleen Jensen and Niklaus Wirth, *Pascal User Manual and Report* (Springer-Verlag, 1975).

21 Gilbert J. Hansen and Charles F. Lindahl, *Preliminary Specification of Real Time Pascal*, Technical Report NAVTRAEQUIPCEN 76-C-0017-1, NTIS No. AD A03 1451 (Computer Laboratory, Naval Training Equipment Center, July 1976).

22 Niklaus Wirth, "Modula: A Language for Modular Multiprogramming," "The Use of Modula," and "Design and Implementation Modula," *Software Practice and Experience*, Vol 7, No. 1 (January, February 1977).

protocol description, some have properties which may prove to be useful as part of a total description. The following are some of the more prominent models mentioned in recent literature.

Petri Nets

Petri nets[23-25] ar· designed to model the synchronization problems of a system composed of concurrent asynchronous processes. This is a directed graph model which has two types of nodes: places and transitions. A place which is connected to a transition via an arc is called an input place of the transition node if it precedes the transition, and is called an output place if it follows the transition. Places may be marked by zero or more tokens. On the Petri net diagram, a place appears as a circle, a token as a heavy dot, and a transition as a line perpendicular to one of its arcs. The state of the system is represented by the pattern of tokens on the places, which is called the marking. The system changes states by moving tokens. The mechanism which moves the tokens is the firing of a transition. A transition fires where each of its input places has at least one token. The action of the transition is to remove one token from each of its input places, and to add one token to each of its output places. Figure 5 is an example of a Petri net.

Petri nets can clearly show the overall state of a system of communicating processes. The possible sequence of states is identifiable. Certain properties of the Petri net have a direct bearing on the system stability. The practical problem with Petri nets is that they get so complex that it becomes nearly impossible to analyze a real system.

The UCLA Graph Model

The UCLA graph model[26] has been used by Postel as an alternative to Petri nets for analyzing protocols. The model consists of arcs and nodes, where the arcs correspond to the flow of control and the nodes

---

23 Robert C. Chen, "Representation of Process Synchronization," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 37-42.

24 Kurt Lautenbach and Hans A. Schmid, "Use of Petri Nets for Proving Correctness of Concurrent Process Systems," *Proceedings of IFIP 74* (IFIP, 1974), pp 187-191.

25 Pamela B. Thomas, *The Petri Net as a Modeling Tool*, NTIS No. K/CSD/INF-76/8, CONF 760430-1, presented at the 14th Annual Southeast Regional ACM Conference, Birmingham, Alabama (April 22-24, 1976).

26 Jonathan B. Postel, *A Graph Model Analysis of Computer Communication Protocols*, Ph.D. Dissertation, NTIS No. AD777506/H, (UCLA, January 1974).

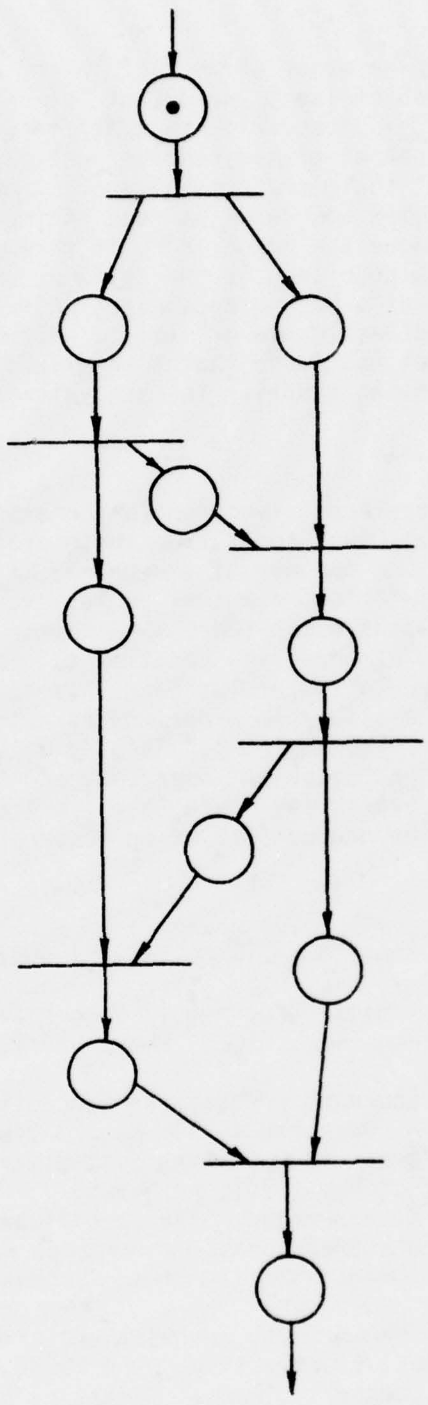


Figure 5. Petri net.

represent computations. The major advance of the UCLA graph model over Petri nets is the addition of input and output logic for a node. Petri nets use only an "and" logic, requiring that all input places have at least one token and all output places receive one token. The UCLA graph model allows either "and" logic or "exclusive or" (EOR) logic for both node input and output. When EOR logic is used on input, the node may choose which input to remove the token from, if more than one is marked. Similarly, on output, the node chooses the position of the token. Figure 6[27] is an example of a UCLA graph model; "*" represents "and" logic, and "+" represents "exclusive or" logic. Although the graphs may become complicated, Postel has found that some graphs may be reduced into graph modules, which can simplify the analysis of the graphs.

Other Synchronization Models

The subjects of interprocess synchronization and communication are active areas of research. Other tools from these areas which may apply to the protocol area include the use of semaphores,[28] critical regions,[29] and monitors[30] to insure mutual exclusion. These techniques are oriented toward processes which share some common memory. A different approach is to use an algebra-like notation to specify the allowable synchronization as opposed to the exclusion. This "path expression notation[31-33] specifies the allowable orderings of activities. There are various notations used. For example, ";" is used as a sequencing operator, "+" is used as an "exclusive or" operator, and "*" is used to mean zero or more times. The expression "path (a ; (b + c)*) end" means that an allowable execution can consist of an occurrence of "a" followed

---

27 Jonathan B. Postel, *A Graph Model Analysis of Computer Communication Protocols*, Ph.D. Dissertation, NTIS No. AD 777 506/H (UCLA, January 1974).

28 F. W. Dijkstra, "Hierarchical Ordering of Sequential Processes," *Operating Systems Techniques*, C. A. R. Hoare and P. H. Perrot, eds. (Academic Press, 1972).

29 Per Brinch Hansen, "Concurrent Programming Concepts," *Computing Surveys*, Vol 5, No. 4 (ACM, December 1973), pp 223-245.

30 C. A. R. Hoare, "Monitors: An Operating System Structuring Concept," *CACM*, Vol 17, No. 10 (October 1974), pp 549-557.

31 R. H. Campbell and A. N. Habermann, "The Specification of Process Synchronization by Path Expressions," *Proceedings of an International Symposium Held at Racquencourt on Operating Systems* (Springer Verlag, April 23-24, 1974).

32 A. N. Habermann, *Path Expressions*, Technical Report (Computer Science Department, Carnegie-Mellon University, June 1975).

33 P. E. Laver and R. H. Campbell, "Formal Semantics of a Class of High Level Primatives for Coordinating Concurrent Processes," *Acta Informatica*, No. 5 (1975), pp 297-332.

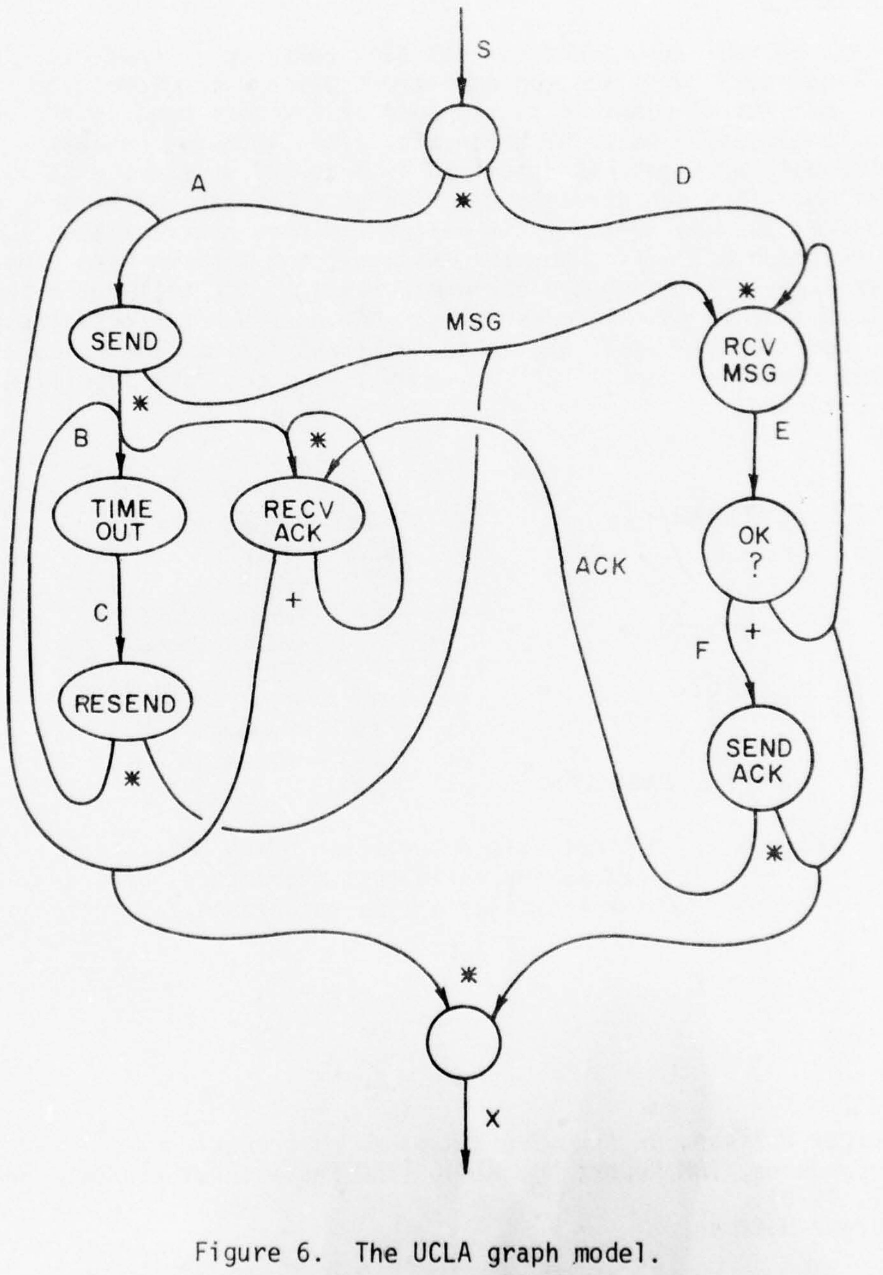


Figure 6. The UCLA graph model.

by zero or more occurrences of either "b" or "c." Work is continuing in this area, and its ultimate application to protocols is not yet determined.

Linguistic Models

The data communications process has been compared to communications by formal language,[34] with the communications message considered to be a sentence. Thus, it is possible to use much of the work done in the area of parsing languages to describe protocols. The language, consisting of all the allowable messages, is specified by a formal grammar, usually in a BNF notation. This can directly describe an automaton which can receive (parse) the messages. It is neater and more powerful than the finite stage graph method for complex messages, but suffers from many of the same problems. Although the allowable messages are well described, several items are not specified formally: how and when the messages are generated, how they are used, and those functions termed "semantics" in the compiler areas. Figure 7[35] is an example of a protocol specified using BNF notation.

```
<FRAME>            ::= "SOH"  <HEADING>
                   ::= "STX"  <TEXT>
                   ::= "EOT"  <TAIL>
                   ::= "DLE"  <DISCONNECT>
<HEADING>          ::= "XX"   <HEADING>
                   ::= "STX"  <TEXT>
<TEXT>             ::= "XX"   <TEXT>
                   ::= "ETX"  <FRAME>
<TAIL>             ::= "|=|"  <FRAME>
 <DISCONNECT>      ::= "EOT"
```

Figure 7. A linguistic description using BNF.
(xx is any valid text character;
1=1 indicates a line turnaround.)

34 Hans-Jurger Hoffman, *On Linguistic Aspects of Communication Line Control Procedures*, IBM Report No. RZ345 (IBM Research Laboratory, February 2, 1970).

35 Hans-Jurger Hoffman.

Specialized Machine Descriptions

Recent work in defining abstract machines for describing communication protocols could allow programming technology to be applied to protocols, without dependence on a particular implementation. The following approaches have appeared recently.

Interlocutors

Danthine and Bremer[36,37] have modeled a protocol as a colloquy between two special machines called interlocutors. Each interlocutor has three inputs and outputs and a set of internal states. The inputs are for text from the other interlocutor (the communications links), text from the user for transmission to the other user, and commands from the user. The outputs are text to the other interlocutor, text to the user, and commands for the user (see Figure 8).[38] The internal structure of the interlocutor consists of a processing unit, an input unit, an output unit, and several buffers. The processing unit is divided into two parts: one deals with fixed operations (Danthine lists eight), and one deals with the variable part of message processing, called context processing. The operation of the fixed portion is governed by a set of matrices defined for the protocol and application. Figure 9[39] shows the internal structure of an interlocutor. The problem with the interlocutor approach appears to be that the context processing part has not yet been sufficiently structured. The parts of a protocol that require the most analysis would likely fall into this category, thus defeating many of the advantages of the structure.

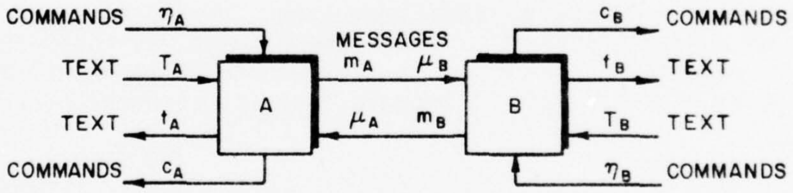


Figure 8. Communication structure of two interlocutors of a colloquy.

---

36 Andre A. S. Danthine and Joseph Bremer, "An Axiomatic Description of the Transport Protocol of Cyclades," *Professional Conference on Computer Networks and Teleprocessing* (March 31 - April 2, 1976).

37 Andre A. S. Danthine and Joseph Bremer, "Communication Protocols in a Network Context," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 87-92.

38 Andre A. S. Danthine and Joseph Bremer, "An Axiomatic Description of Cyclades," *Professional Conference on Computer Networks and Teleprocessing* (March 31 - April 2, 1976).

39 Danthine and Bremer.

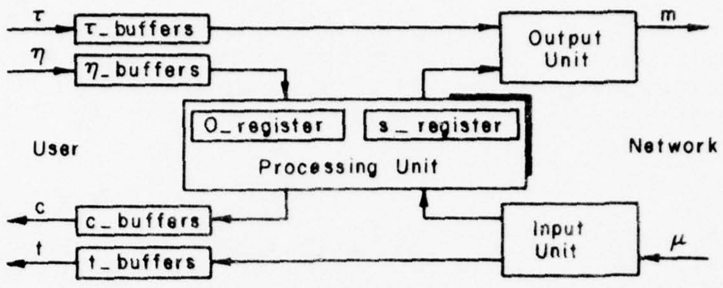


Figure 9. Internal structure of an interlocutor.

Protocol Machines

Gouda and Manning[40-42] have defined a "Protocol Machine" as an entity capable of communicating with other compatible protocol machines via sent and received messages. Each protocol machine consists of two structures: a data structure and a control structure. The control structure is a directed graph, called a sequence graph, which has four possible types of nodes: (1) the send node, (2) the receive node, (3) the update node, and (4) the decision node. A subclass of protocol machines (sr machines) which have only send and receive nodes has also been identified. The data structure which the protocol machine may manipulate consists of variables that map to and from fields in the messages that may be sent or received. Figure 10[43] shows a sample data structure and sequence graph for a protocol machine. Gouda and Manning have shown that send and receive nodes have direct representations as Petri nets. The approach appears to have some promise, although the model does not presently appear to be general enough to represent arbitrary protocols. In particular, it appears that asynchronous processing is excluded.

---

40 Mohamed G. Gouda and Eric G. Manning, "On the Modelling, Analysis, and Design of Protocols--A Special Class of Software Structures," *Proceedings of the 2nd International Conference of Software Engineering,* IEEE No. 76CH1125-4C (Institute of Electrical and Electronic Engineers [IEEE], 1976) pp 256-262.

41 Mohamed G. Gouda and Eric G. Manning, "Protocol Machines: A Concise Formal Model and Its Automatic Implementation," *Proceedings of the 3rd International Conference on Computer Communication* (International Council for Computer Communication, August 3-6, 1976), pp 346-350.

42 Mohamed G. Gouda and Eric G. Manning, "Toward Modular Hierarchical Structure for Protocols in Computer Networks," *Proceedings of Compcon 76* (IEEE, September 1976).

43 "On the Modelling, Design, and Analysis of Protocols."

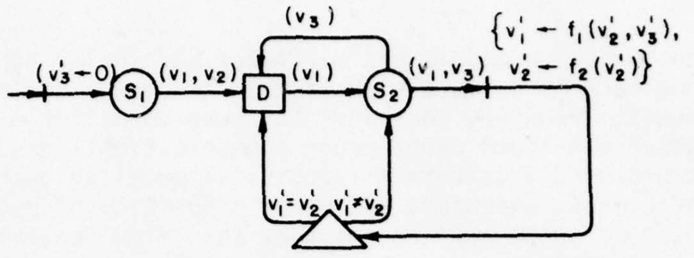


Figure 10a. Sequence graph.

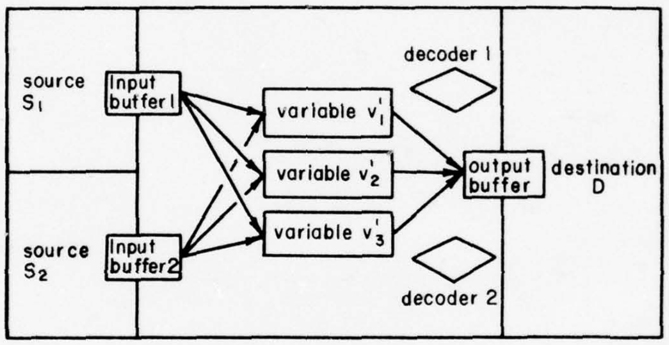


Figure 10b. The data structure of a protocol machine.

Program Process Modeling Language

Riddle[44] uses a programming language form of notation (the Program Process Modeling Language [PPML]) to model the message transfer activities of a system of asynchronous processes. The language does not provide details about the internal operation of each process, but the sending and receiving of messages and the internal flow of control are modeled. Riddle also develops an algebra-like notation to describe message flow within the system, called Message Transfer Expressions (MTE). The basic modeling scheme chosen does have some limitations: in particular, the direct modeling of synchronous communication is not possible. However, the overall approach appears to be very promising.

44 William Ewing Riddle, *The Modeling and Analysis of Supervisory Systems*, Ph.D. Dissertation (Stanford University, 1972).

Summary

Each of the preceding models has merit for particular applications and analyses, but each is primarily applicable to communications over error-free channels. However, some work has been done (for example, by Sunshine[45] which considers error-prone communication links. Schneider[46] has proposed a uniform all-inclusive modeling technique to model error, efficiency, overhead, and other properties of protocols. There does not appear to be any consensus at this time concerning which approach is most promising.

---

[44] William Ewing Riddle, *The Modeling and Analysis of Supervisory Systems*, Ph.D. Dissertation (Stanford University, 1972).

[45] Carl A. Sunshine, *Interprocess Communication Protocols for Computer Networks*, NTIS No. ADA 025 508, Ph.D. Dissertation (Stanford University, December 1975).

[46] Michael G. Schneider, *A Structural Approach to Computer Network Simulation*, Technical Report 75-20 (Computer Information and Control Sciences Department, Institute of Technology, University of Minnesota, December 1974).

4 PROTOCOL SPECIFICATION CRITERIA

This report has surveyed the functional requirements of communication protocols and the ways in which protocols have been specified, analyzed, and implemented. Throughout the survey, the interrelationships of the different phases of the protocol life cycle requirements, design, analysis, implementation, and use have become apparent. Furthermore, any protocol specification or description should be useful during all phases. Bochman[47] has defined three general properties that a protocol specification should have.

1. The protocol specification should be in a comprehensive form.

2. The protocol specification should allow the proving of certain protocol properties, particularly that the error recovery is effective, and that all possible situations have been considered.

3. The protocol specification should lead easily and naturally toward its implementation.

A protocol specification must not only be complete, but it should be provably complete. It should be useful at each phase of the protocol life cycle and should not require reinterpretation to be useful at another phase. It must be comprehensive and unambiguous. It should be possible to partition a protocol specification into different levels of abstraction. Protocol layering should also be apparent in the specification; when a specification depends on a lower-level protocol, this should be explicitly noted. The protocol specification should be independent of any particular hardware or any particular implementation. It should clearly indicate how the functional issues of the protocol are resolved.

[47] Gregor V. Bochman, "Communication Protocols and Error Recovery Procedures," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 45-50.

5 SUMMARY AND CONCLUSIONS

This report has surveyed the current status of communication protocols. The issues of protocol design are considered in two categories: functional issues and design issues. The functional issues vary from protocol to protocol, depending on its intended function and environment. The design issues deal with the reliability and efficiency of the protocol, and should be considered for any protocol.

The methods employed to describe and analyze protocols range from informal textual descriptions to very formal specific machine representations. Bochman's goals--comprehensive form, proving of properties, and natural implementation--are used as a general basis for evaluating protocol specification methods. No currently used representation was found to be sufficient as the sole method of specifying protocols.

No general approach to the specification of communication protocols can be ruled out at this time; however, it appears that methods of the general machine description class are more likely to lead toward a useful specification methodology, since they directly describe the actions of the communicating parties, which is what must be implemented. The finite state subclass is useful when the state space of the protocol is reasonably small, since it is concise, directly implementable, and analyzable. However, when the state space is large, a more general approach must be taken.

The trends observed in some of the newer computer languages can lead to a language which is machine independent, efficiently implementable, and suitable as a vehicle for protocol specification and implementation. Current work in languages should be augmented to include the formal description of messages and the primitive operations which manipulate the messages. Techniques to analyze the behavior of systems of communicating parties specified in this way must also be developed in order to detect abnormal conditions such as critical races and deadlocks. The combination of these approaches into a unified methodology can lead to the better understanding and implementation of communication protocols.

ANNOTATED REFERENCES

1. Abrams, Marshall, Robert Blanc, and Ira Cotton, editors, *Computer Networks: A Tutorial,* Cat. No. JH3100-5C (IEEE Computer Society, October 1975).

   A compendium of papers, mostly previously published, dealing with many aspects of computer networks. It covers the networking vocabulary, configurations, components, software, utilization, measurement, evaluation, costs, and management.

2. Abramson, Norman and Franklin F. Kuo, editors, *Computer Communications Networks* (Prentice-Hall, Inc., 1973).

   A collection of articles by several authors covering different aspects of designing an entire system. It surveys some existing networks, design philosophies, communication links, data transmission, interfacing, concentrators, and economic and regulatory considerations.

3. Akkoyunlu, Erlap, Arther Bernstein, and Richard Schantz, "Interprocess Communication Facilities for Network Operating Systems," *Computer,* Vol 7, No. 6 (June 1974), pp 46-55.

   The reasons and goals of computer networking are surveyed. The requirement for interprocess communication is introduced, and several different approaches are compared.

4. *American National Standard Procedures for the Use of the Communication Control Characters of American National Standard Code for Information Interchange in Specified Data Communication Links,* ANSI X3.28 (ANSI, 1976).

5. *Batch Terminal Controller-Complot BTC-7/200 Instruction Manual* (Houston Instrument, Inc., revised May 1972).

6. Belford, Geneva, Steve Bunch, John Day, et al., *A State of the Art Report on Network Data Management and Related Technology,* CAC Document No. 150 (Center for Advanced Computation, University of Illinois, April 1, 1975).

   A wide-ranging report considering hardware, software, communication media and techniques, and other topics of interest in the context of networking. Some emphasis is given to facilities found in ARPANET.

7. Birke, Dennis M., *State-Transition Programming Techniques and Their Use in Producing Teleprocessing Devices Control Programs*, M.S. Thesis (University of Pittsburgh, 1971).

The use of state transition methods for programming communication link procedures is discussed, and a simple example is shown.

8. Birke, Dennis M., "State-Transition Programming Techniques and Their Use in Producing Teleprocessing Device Control Programs," *Proceedings of the 2nd Symposium on Problems in the Optimization of Data Communications Systems* (Association for Computer Machinery [ACM], October 20-22, 1971), pp 21-31.

A shortened conference paper covering the same material as the thesis with the same title.

9. Bjorner, Dines, "Finite State Automation--Definition of Data Communication Line Control Procedures," *Proceedings of the Fall Joint Computer Conference* (AFIPS, 1970), pp 477-491.

This paper describes the use of finite state descriptions for the specification, analysis, and implementation of communication protocols.

10. Bochman, Gregor V., "Communication Protocols and Error Recovery Procedures," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 45-50.

This publication lists the types of situations that a communication protocol must deal with. An extended finite state notation is used as an analysis tool. A method for determining synchronization properties of protocols is introduced.

11. Bochman, Gregor V., "Logical Verification and Implementation of Protocols," *Proceedings of the 4th Data Communications Symposium* (October 7-9, 1975), pp 7-15 to 7-20.

The problems of specifying, analyzing, and implementing a data communication protocol are discussed. An example is analyzed, and some existing tools from other disciplines are considered.

12. Brinch Hansen, Per, *Concurrent Pascal Introduction and Concurrent Pascal Report* (California Institute of Technology, July 1975).

An introduction and description of the Concurrent Pascal language, designed for programming systems of concurrent processes.

13. Brinch Hansen, Per, "Concurrent Programming Concepts," *Computing Surveys*, Vol 5, No. 4 (ACM, December 1973), pp 223-245.

This paper describes the evolution of language features for multiprogramming, including semaphores, critical regions, and monitors. Problems of multiprogramming are illustrated by examples.

14. Campbell, R. H., and A. N. Habermann, "The Specification of Process Synchronization by Path Expressions," *Proceedings of an International Symposium Held at Racquencourt on Operating Systems* (Springer Verlag, April 23-25, 1974).

An introduction to the use of path expressions for describing the synchronization of a system of asynchronous processes.

15. Chen, Robert C., "Representation of Process Synchronization," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 37-42.

This paper shows the application of Petri nets to the representation of process synchronization.

16. Chou, W., H. Frank, and R. Van Slyke, "Simulation of Centralized Computer Communications System," *Data Networks: Analysis and Design, 3rd Data Communications Symposium*, IEEE No. 73CH0828-4C (IEEE, 1973), pp 121-130.

This paper describes a simulation approach for a general centralized computer communication system with emphasis on efficiency and versatility. A hybrid approach is used to ease program development and to shorten computer running time. The techniques are illustrated by application to the NASDAQ system.

17. Clark, David W., Charles Hoffman, John Stannard, and Walter Levy, *Communication Computer Language Comtran*, Report RADC-TR-69-190 (Rome Air Development Center, July 1969).

This report describes a communications-oriented compiler system which accepts high-level statements that describe a set of communication line units combined to constitute a real-time on-line communications test. Limitations are noted and discussed.

18. Cohen, P. M., *Programming Languages for Communication Processors*, Technical Note No. 24-73, NTIS No. ADA-014 544 (Defense Communications Agency System Engineering Facility, May 1973).

This note describes some work in the area of implementation languages for communication.

19. Cohen, P. M., *The Use of a Communications-Oriented Language Within a Software Engineering System*, Technical Note No. 17-75, NTIS No. ADA-014511 (Defense Communications Engineering Center, April 1975).

Some requirements of communications-oriented language are outlined, and the feasibility of using such a system is discussed.

20. Crocker, Stephen D., John F. Heafner, Robert Metcalfe, and Jonathan Postel, "Function-Oriented Protocols for the ARPA Computer Network," *Proceedings of the Spring Joint Computer Conference*, Vol 40 (AFIPS, 1972), pp 271-279.

This paper provides a general description of the different protocols available in the ARPANET.

21. Danthine, Andre A. S., and Joseph Bremer, "An Axiomatic Description of the Transport Protocol of Cyclades," *Professional Conference on Computer Networks and Teleprocessing* (March 31 - April 2, 1976).

This paper introduces the notion of the interlocutor as a model of a data communication protocol machine. It describes the Cyclades transport protocol using the interlocutor.

22. Danthine, Andre A. S., and Joseph Bremer, "Communication Protocols in a Network Context," *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop* (ACM, March 24-25, 1975), pp 87-92.

This paper provides a further discussion of the interlocutor as a model for protocol analysis.

23. *Datapac Standard Network Access Protocol* (Bell Canada, November 30, 1974).

A description of the protocols available via the Bell Canada Datapac service.

24. Dennis, Jack B., *First Version of a Data Flow Procedure Language*, MAC Technical Memorandum No. 61 (M.I.T., May 1975).

A language for representing computational procedures based on the concept of data flow is presented in terms of a model which allows concurrent execution of noninterfering program parts.

25. *Digital Network Architecture Design Specification for: Data Access Protocol D.A.P.* (Digital Equipment Corp., July 10, 1975).

This document describes the DECNET-DAP communication protocol. The basic goal of DECNET is to create a set of facilities allowing for device sharing, program sharing, and interprogram communication.

26. *Digital Network Architecture Design Specification for: Network Service Protocol (NSP)* (Digital Equipment Corp., July 10, 1975).

This document describes the DECNET-NSP protocol.

27. Dijkstra, E. W., "Hierarchical Ordering of Sequential Processes," *Operating Systems Techniques*, C. A. R. Hoare and R. H. Perrot, eds. (Academic Press, 1972).

This paper describes some of the problems of synchronizing systems of concurrent processes, using semaphores as the synchronizing primitive, and illustrating them with examples.

28. Donnan, R. A., and J. R. Kersey, "Synchronous Data Link Control: A Perspective," *IBM Systems Journal*, Vol 13, No. 2 (1974), pp 140-162.

This is a description of the IBM SDLC communication protocol.

29. Eisenbies, J. L., "Conventions for Digital Data Communication Link Design," *IBM Systems Journal*, Vol 6, No. 4 (1967), pp 267-392.

This is a description of the IBM binary synchronous communication protocol (BSC).

30. *General Information--Binary Synchronous Communications*, Order No. GA27-3004-3, 3rd ed. (IBM, October 1972).

31. Gouda, Mohamed G., and Eric G. Manning, "On the Modelling, Analysis, and Design of Protocols--A Special Class of Software Structures," *Proceedings of the 2nd International Conference on Software Engineering*, IEEE No. 76CH1125-4C (IEEE, 1976), pp 256-262.

This paper discusses the problems of protocol specification, analysis, and design. A model called protocol machines is introduced, and examples of its use are shown.

32. Gouda, Mohamed G., and Eric G. Manning, "Protocol Machines: A Concise Formal Model and Its Automatic Implementation," *Proceedings of the 3rd International Conference on Computer Communication* (International Council for Computer Communication, August 3-6, 1976), pp 346-350.

This is an introduction and overview for using Protocol Machines to model data communications. Another model, called synchronous digital machines, is introduced and the relation between the two models is discussed.

33. Gouda, Mohamed G., and Eric G. Manning, "Toward Modular Hierarchical Structures for Protocols in Computer Networks," *Proceedings of Compcon 76* (IEEE, September 1976).

A structure using independent processes is advocated for modeling and designing communication protocols.

34. Gray, James P., "Line Control Procedures," *Proceedings of the IEEE,* Vol 60 (IEEE, November 1972), pp 1301-1312.

This paper discusses the basic principles of line control procedures. Two specific line controls are used as examples.

35. Habermann, A. N., *Path Expressions,* Technical Report (Computer Science Department, Carnegie-Mellon University, June 1975).

The use of Path Expressions to describe the synchronization of a system of concurrent processes is introduced.

36. Hansen, Gilbert J., and Charles E. Lindahl, *Preliminary Specification of Real Time Pascal,* Technical Report NAVTPAEQUIPCEN 76-C-0017-1, NTIS No. ADA031451 (Computer Laboratory, Naval Training Equipment Center, July 1976).

This report describes a programming language based on PASCAL for use in real time control applications.

37. Heart, Frank E., *Interface Message Processors for the ARPA Computer Network,* Quarterly Technical Report No. 3, NTIS No. ADA016614 (1 July 1975 to 30 September 1975).

This is a status report on IMPs on the ARPANET. It includes a discussion of contention errors between independent concurrent processes, and of some work done to identify where these types of errors can occur.

38. Heart, F. E., R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The Interface Message Processor for the ARPA Computer Network," *Proceedings of the Spring Joint Computer Conference,* Vol 40 (AFIPS, 1972), pp 551-567.

This is a description of the IMP hardware and software. It discusses network design, messages, links, reliability, and recovery.

39. Hoare, C. A. R., "Monitors: An Operating System Structuring Concept," *CACM*, Vol 17, No. 10 (October 1974), pp 549-557.

This paper develops the concept of the monitor as a structuring tool for synchronizing the actions of asynchronous concurrent processes.

40. Hoffman, Hans-Jurger, *On Linguistic Aspects of Communication Line Control Procedures*, IBM Report No. RZ345 (IBM Research Laboratory, February 2, 1970).

The strings of a digital communication are considered as instances of sentences in a language. Linguistic treatment defines the rules to be obeyed for message transmission. Finite state implementation is shown to be directly obtainable from the linguistic description.

41. *Information Processing--Basic Mode Control Procedures for Data Communication Systems*, Reference No. ISO 1745-1975(E) (International Standards Organization, January 2, 1975).

42. *The Interface Message Processor Program* (Bolt, Beranek, and Newman, Inc., September 1975).

A description of the IMP software, covering the HOST-IMP and IMP-IMP protocols, interfacing, routing, and data structures.

43. *Introduction to Minicomputer Networks* (Digital Equipment Corp., 1974).

This introduces computer to computer communications, covering network structures, functions, and hardware and software components. Line control procedures and applications of networks of PDP-11 computers are also described.

44. Jensen, Kathleen, and Niklaus Wirth, *Pascal User Manual and Report* (Springer-Verlag, 1975).

This manual provides the definitive description and reference for the programming language Pascal.

45. Lauer, P. E., and R. H. Campbell, "Formal Semantics of a Class of High-Level Primatives for Coordinating Concurrent Processes," *Acta Informatica*, No. 5 (1975), pp 297-332.

This is a more formally and theoretically oriented treatment of path expressions. Alternate solutions to a well-known problem of synchronization are analyzed.

46. Lautenbach, Kurt, and Hans A. Schmid, "Use of Petri Nets for Proving Correctness of Concurrent Process Systems," *Proceedings of IFIP 74* (IFIP, 1974), pp 187-191.

The theory of Petri nets is introduced and is shown to be useful for deriving theorems on the properties of concurrent systems.

47. Martin, James, *Introduction to Teleprocessing* (Prentice-Hall, Inc., 1972).

This is a basic introduction to computer communications. This book surveys most communications topics very briefly.

48. Martin, James, *Telecommunications and the Computer* (Prentice-Hall, Inc., 1969).

This is a survey of the technology of data communications. Heavy emphasis is placed on hardware, telecommunications networks, signaling, and message routing.

49. Martin, James, *Teleprocessing Network Organization* (Prentice-Hall, Inc., 1970).

This provides a complete overview of data communications. Different transmission media and modes of operation are discussed. Some examples of line control are given.

50. Metcalfe, Robert M., *Strategies for Interprocess Communication in a Distributed Computing System,* presented at the Symposium on Computer Communications Networks and Teletraffic (Polytechnic Institute of Brooklyn, April 4-6, 1972).

This paper distinguishes between distributed interprocess communication and centralized interprocess communication.

51. Mills, David L., "Communication Software," *Proceedings of the IEEE,* Vol 60 (IEEE, November 1972), pp 1333-1341.

This is a tutorial introduction to the function and construction of communications software, including network control, message preprocessing, and error recovery.

52. Parent, Michel, "Presentation of the Control Graph Models," *Proceedings of an International Symposium Held at Racquencourt on Operating Systems* (Springer Verlag, April 23-25, 1974).

A graph model that can be used to analyze problems in systems of concurrent processes is introduced and explained.

53. Postel, Jonathan B., *A Graph Model Analysis of Computer Communication Protocols*, NTIS No. AD777506/H, Ph.D. Dissertation, (UCLA, January 1974).

This dissertation focuses on the analysis of computer to computer communication protocols, using graph modeling techniques. The UCLA graph model is explained and used to model and analyze a sample protocol. The concept of a graph module is introduced.

54. Pouzin, Louis, "An Integrated Approach to Network Protocols," *Proceedings of the National Computer Conference* (1975).

This provides a description of the requirements for communication protocols in a heterogeneous computer network. The relationship between different protocol levels is discussed and the concept of protocol nesting is introduced.

55. Pouzin, Louis, "Presentation and Major Design Aspects of the Cyclades Computer Network, *Data Networks: Analysis and Design 3rd Data Communication Symposium*, IEEE No. 73CH0828-4C (IEEE, 1973), pp 80-87.

This describes the design and functions of the Cyclades computer network.

56. Riddle, William Ewing, *The Modeling and Analysis of Supervisory Systems*, Ph.D. Dissertation (Stanford University, 1972).

A programming language-like modeling language (PPML) is introduced and used to model systems of asynchronous concurrent processes. An algebraic notation (message transfer expressions [MTE]) is used to analyze the message interactions of the system.

57. Schneider, G. Michael, *A Structural Approach to Computer Network Simulation*, Technical Report 75-20 (Computer Information and Control Sciences Department, Institute of Technology, University of Minnesota, December 1975).

A modeling system intended to model all phases of computer to computer communication is described.

58. Schneider, G. Michael, *DSCL--A Data Specification and Conversion Language for Networks*, Technical Report 74-27 (Computer Information and Control Sciences Department, Institute of Technology, University of Minnesota, December 1974).

This paper describes the initial work on a processor for supporting the real time translation and transmission of data streams between nodes of a computer network. A primary component is the data specification and conversion language specifying the physical and logical structure of data streams and their translations.

59. Schneider, G. Michael, *Resource Sharing Computer Networks,* Technical Report 75-3 (Computer Information and Control Science Department, Institute of Technology, University of Minnesota, February 1975).

This paper defines a resource-sharing computer network, and looks at user services that can be provided. Some current design problems are outlined.

60. Schneider, G. Michael, *The Implementation of Centralized Services in Resource Sharing Computer Networks,* Technical Report 75-16 (Computer Information and Control Sciences Department, Institute of Technology, University of Minnesota, September 1975).

This report describes a model for the implementation of centralized services within a distributed resource-sharing computer network. It describes existing models for low-level interprocess communication and uses them to construct a model for a centralized network service protocol.

61. Stanford Research Institute, *ARPANET Protocol Handbook,* compiled by F. Feinler and J. Postel, NTIS No. ADA027964 (April 1976).

This collection of documents describes ARPANET protocols as of April 1976.

62. Stutzman, Byron W., "Data Communication Control Procedures," *Computer Surveys,* Vol 4, No. 4 (December 1972).

This is a tutorial on the methods used to control the transmission of digital information over data communication links. Models and terminology of communication systems and their functions are introduced.

63. Sunshine, Carl A., *Interprocess Communication Protocols for Computer Networks,* NTIS No. ADA025508, Ph.D. Dissertation (Stanford University, December 1975).

This paper focuses on the design and analysis of interprocess communication protocols for networks of computers. It examines requirements and performance of host to host protocols.

64. Thomas, Pamela B., *The Petri Net as a Modeling Tool*, NTIS No. K/CSD/INF-76/3, CONF 760430-1, presented at the 14th Annual Southeast Regional ACM Conference, Birmingham, Alabama (April 23-24, 1976).

The Petri net model is defined and presented as a modeling tool for coordination of asynchronous processes.

65. *200 User Terminal Hardware Reference Manual*, Publication No. 82128000 (Control Data Corporation, June 1969).

An overview of the hardware, operation, and programming considerations of the 200 user terminal, a terminal for remote batch job entry.

66. *Fundamentals of Computer Network Communications*, No. SP0076 (Sperry Rand, 1970).

This introduction to computer communications covers communication links, transmission methods, modem operation, data terminals, codes, message exchanges, control operations, and future developments.

67. Walden, David C., "A System for Interprocess Communication in a Resource Sharing Computer Network," *CACM*, Vol 15, No. 4 (April 1972), pp 221-230.

A system of communication between processes in a time-sharing system is described, and the communication system is extended so that it may be used between processes distributed throughout a computer network.

68. Wirth, Niklaus, "Modula: A Language for Modular Multiprogramming," "The Use of Modula," and "Design and Implementation of Modula," *Software Practice and Experience*, Vol 7, No. 1 (January, February 1977).

This provides a discussion and examples of the use and implementation of Modula.

CERL DISTRIBUTION

Chief of Engineers
ATTN: DAEN-DSE
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-RDL
Dept of the Army
WASH DC 20314

Defense Documentation Center (12)

EIDSO

Each District and Division
ATTN: ADP Coordinator

University of Illinois
Computer Science Dept.
ATTN: Dr. J. W. S. Liu
ATTN: Dr. M. D. Mickunas
Urbana, IL 61801

Computer Systems Command
Institute for Research in Mgt. Info.
and Computer Sciences (CSCS-AT)
Fort Belvoir, VA 22060

U.S. Army Computer Systems Command
ISRAD Project Office
Fort Belvoir, VA 22060

A survey of the properties of computer communication protocols. - Champaign, IL : Construction Engineering Research Laboratory ; Springfield, VA : NTIS, 1978.
2 v. ; 27 cm. (Technical report ; O-1)
Contents : v. 1. Itzkowitz, A. E. The function, properties, specification and analysis methods of computer communication protocols.--v. 2. Liu, J. W. S., Mickunas, M. D. Future developments of computer network protocols.

1. Computer networks. I. Itzkowitz, Avrum E. II. Liu, J. W. S. III. Mickunas, M. D. IV. Title : Function, properties, specification and analysis methods of computer communication protocols.

V. Title : Future developments of computer network protocols. VI. Series: U.S. Construction Engineering Research Laboratory. Technical report ; O-1.