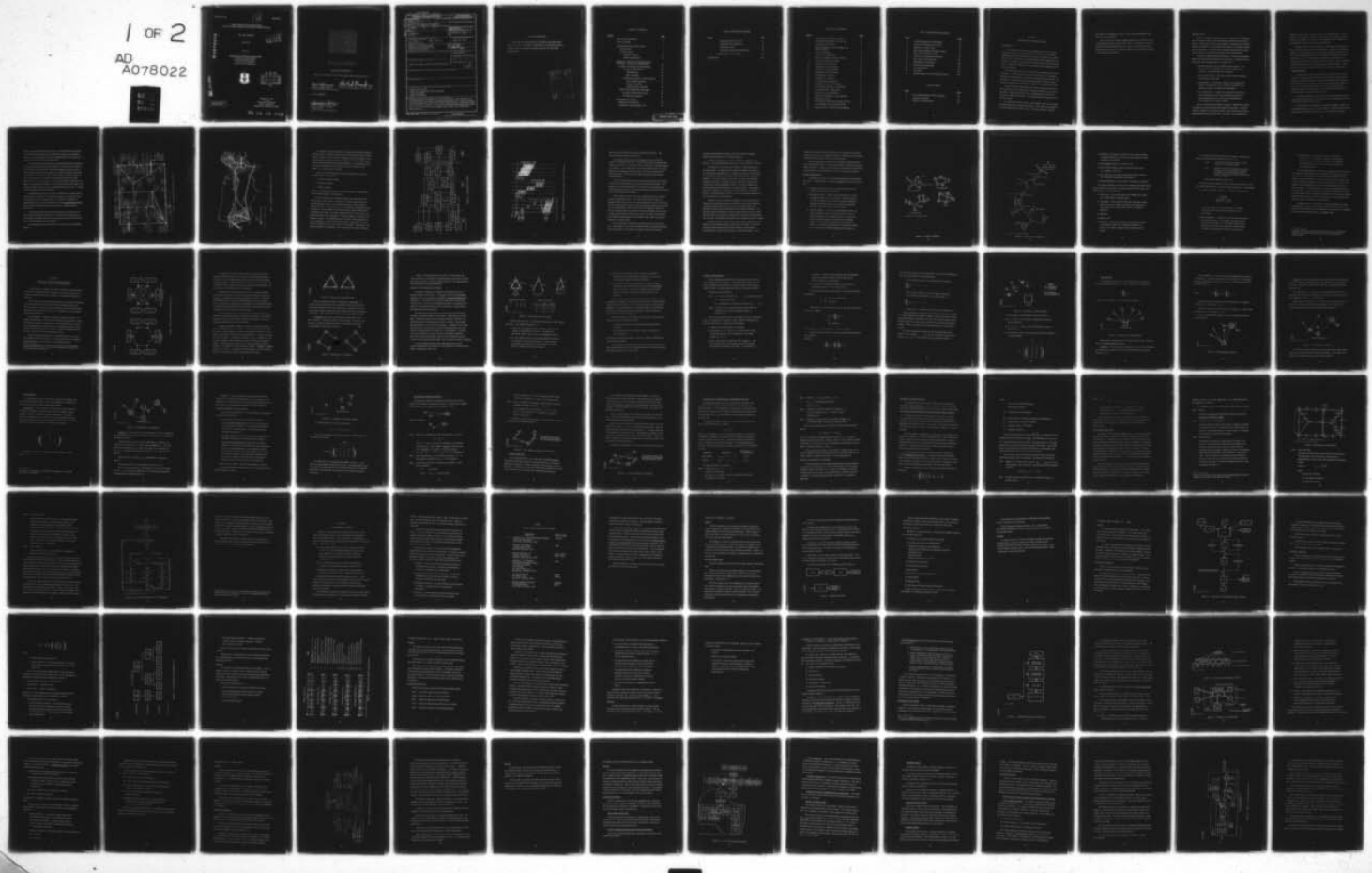
AD-A078 022 MITRE CORP BEDFORD MA F/G 17/2
COMPUTERIZED MODELS FOR DESIGN AND ANALYSIS OF COMPUTER-COMMUNI--ETC(U)
MAY 79 C TROPPER F19628-78-C-0001
UNCLASSIFIED MTR-3664 ESD-TR-79-128 NL

1 OF 2
AD A078022



ESD-TR-79-128

MTR-3664

COMPUTERIZED MODELS FOR DESIGN AND ANALYSIS OF COMPUTER-COMMUNICATIONS SYSTEMS

BY CARL TROPPER

MAY 1979

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

Project No. 5720
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-78-C-0001

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

Daniel R. Baker

DANIEL R. BAKER, Capt, USAF
Computer Systems Engineer
Technology Applications Division

Charles J. Greene, Jr.

CHARLES J. GREWE, JR., Lt Colonel, USAF
Chief, Technology Applications Division

FOR THE COMMANDER

Normand Michaud

NORMAND MICHAUD, Colonel, USAF
Director, Computer Systems
Engineering
Deputy for Technical Operations

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER: ESD-TR-79-128	2. GOVT ACCESSION NO.
	3. RECIPIENT'S CATALOG NUMBER
(6) 4. TITLE (and Subtitle): COMPUTERIZED MODELS FOR DESIGN AND ANALYSIS OF COMPUTER-COMMUNICATIONS SYSTEMS	5. TYPE OF REPORT & PERIOD COVERED
	(14) 6. PERFORMING ORG. REPORT NUMBER: MTR-3664
(10) 7. AUTHOR(s): Carl Tropper	(15) 8. CONTRACT OR GRANT NUMBER(s): F19628-78-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, P. O. Box 208, Bedford, MA 01730 (12) 137	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 5720
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Technical Operations, Electronic Systems Division, AFSC, Hanscom AFB, MA 01731 (11)	12. REPORT DATE: MAY 1979
	13. NUMBER OF PAGES: 130
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 235 050	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited. (9) Technical Rept.,

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

18. SUPPLEMENTARY NOTES

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

COMPUTER ANALYSIS
COMPUTER COMMUNICATIONS NETWORKS
COMPUTER DESIGN
COMPUTER MODELS

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

This paper presents a comparison of several models that may be employed for the design and analysis of computer-communications networks. The comparison focuses on the algorithms used for the various facets of network design — routing algorithms, network reliability models, etc. Two models of particular interest have been singled out — one is in the area of protocol design, and the other is an overall network design program.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

235 050

ACKNOWLEDGMENTS

This report has been prepared by The MITRE Corporation under Project No. 5720. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

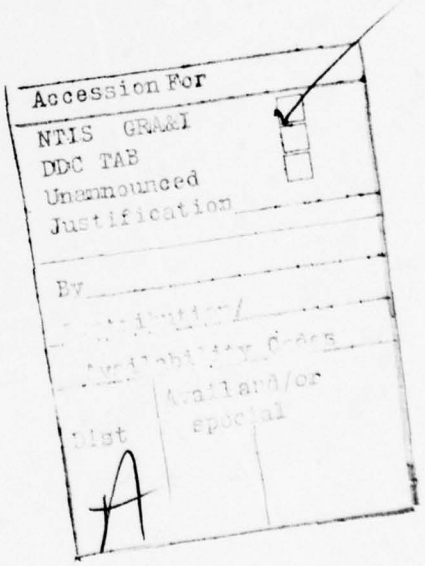


TABLE OF CONTENTS

TABLE OF CONTENTS (concluded)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (concluded)

LIST OF TABLES

SECTION 1

BACKGROUND AND INTRODUCTION

BACKGROUND

The performance modeling and simulation project (5720) currently underway at MITRE has as its objective the creation of a C^3 simulation capability during the Conceptual Phase of system acquisitions on the part of the ESD SPOs. The project is thus attempting to identify appropriate tools (models or modeling systems) that may be employed early in the acquisition cycle. Another goal of this project is the specification of guidelines for C^3 systems simulation.

With these goals in mind, two surveys were undertaken at the outset of this project. The first was a survey of the current uses of C^3 simulation during the Conceptual Phase, along with an analysis of selected models. The second survey, which forms the content of this paper, focused on models that may be employed for the design of computer-communications networks.

Several models were identified which might be employed for this purpose and analyses of their algorithmic content are presented as the heart of the report. Summary sections are also included presenting the author's views on the utility of each model. An overall summary is presented at the end of the report.

It is important to point out that to our knowledge, none of these models are being employed on a current ESD problem. (This remark also applies to the models identified in the first survey.) However, plans are currently

being made for the application of one or more model(s) identified in these surveys to such a problem.

One of the more promising candidate problems is the design of a tactical distributed data-base system (Project 522G). This problem involves the design of both local and global networks. Hence, it is felt that an attempt should be made to discover any special models that are particularly relevant in the design of local networks.

INTRODUCTION

An area of fundamental importance in the construction of C^3 systems is most certainly the ability to effectively design a computer-communications network. Indeed these networks might properly be recognized as forming for all practical purposes, the central nervous system of our military command and control system. The world-wide AUTODIN II network and the SACDIN network testify to the reliance of the military upon these networks.

In spite of a rapidly progressing technology, C^3 system requirements place a heavy load upon the designer of such a system. Among the requirements which must be placed upon such a system are:

- Excellent Response Time/Throughput Characteristics — In a time of crisis, no possibility of a system overload can be tolerated. A fast response time is mandatory.
- System Survivability — The system must be made as resistant to interference as possible.
- Expandability — As technology changes, and as political and military considerations vary, the ability to reconstruct and update the system is an important consideration.
- Cost-effectiveness — In spite of the importance of the above-listed requirements, the ravages of inflation simply do not permit unlimited funds for the inclusion of all desirable characteristics in any military system.

Thus, an excellent system design is required, coupled with an ability to maintain the system efficiently and expand it as necessary. Given the technical complexity of the design questions, one is led to search for automated design tools that can take some, if not most, of the burden off the

designers of the system. A number of such tools do indeed exist. It is the purpose of this paper to describe and evaluate the utility of these tools in the context of the design of a C^3 system. The emphasis is upon a comparison of the algorithms employed in the various design phases.

With this purpose in mind, the paper is organized along the following lines. Section 1 provides a general background on computer-communications networks — their classification, the technologies employed in their construction, and the fundamental design issues and difficulties involved in addressing these issues. Section 2 then presents a collection of algorithms that are commonly employed in the construction of these networks. Section 3 provides detailed descriptions of the software packages presently available. Finally, Section 4 is a summary, in general terms, of the results of the analyses.

Computer Networks

A wide variety of computer networks are in existence today, ranging from intra-company networks that might be used for stock-control and distribution, to packet-switched networks such as ARPANET (ARPA Network), whose purpose is to interconnect a collection of heterogeneous computers in the U.S. so that users and programs at one center will have access to facilities available at other centers. The CYLADES network in France is another example of a packet-switched network.

On a functional basis, it is common to distinguish between three types of networks: Remote-Access Networks (RAN), Value-Added Networks (VAN), and Mission-Oriented Networks (MON).

The Remote-Access Networks are essentially designed to support communications between a user and a host computer. Commercial time-sharing systems such as TYMNET (Tymshare, Inc.) and INFONET (Computer

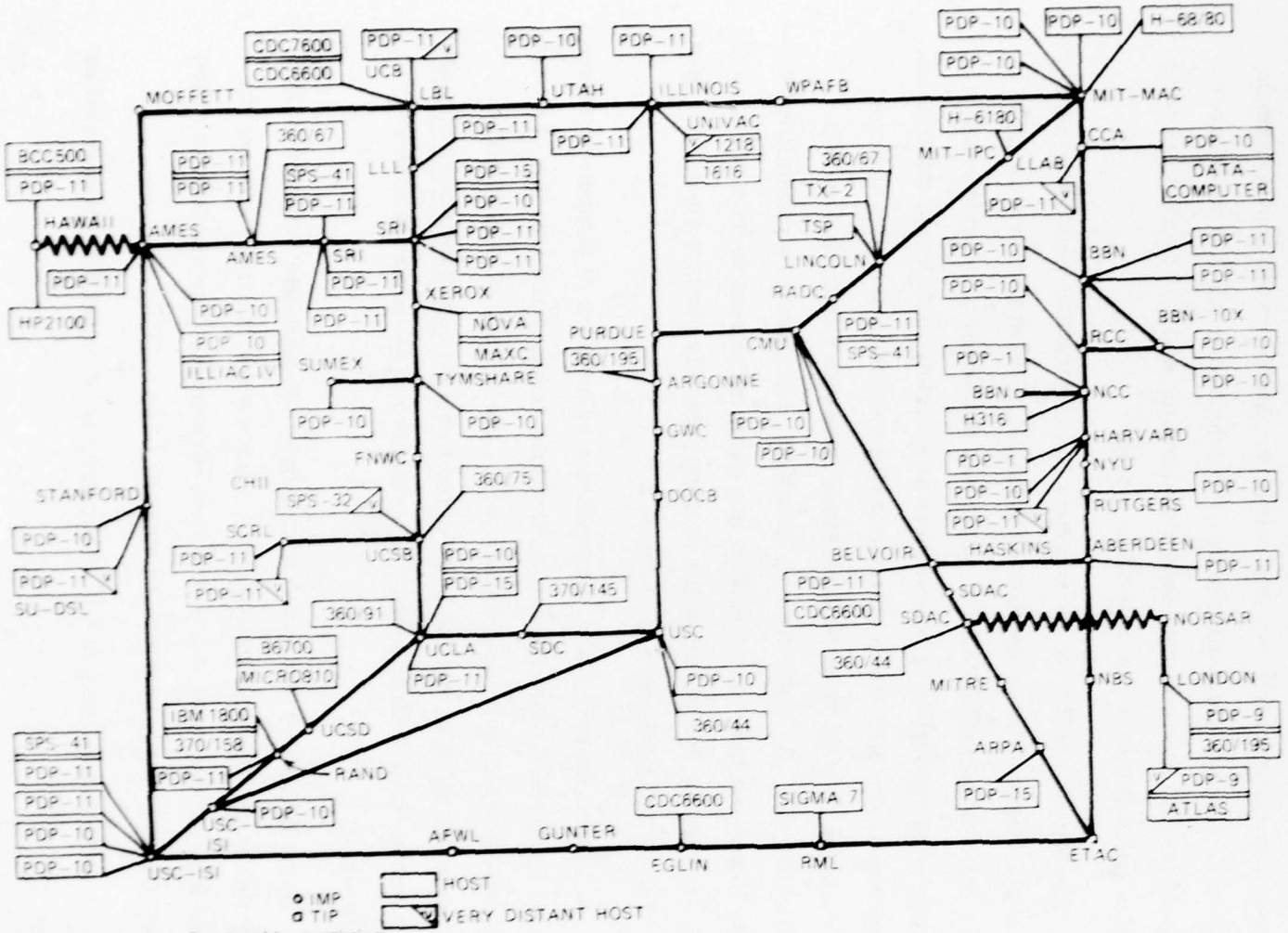
Sciences Corporation) fall into this category. Another well-known example is the SITA (Société Internationale de Télécommunications Aeronautiques) network, used to interconnect computers handling airline reservations for a group of some 160 carriers around the world.

The Value-Added Network supports communications between computers rather than between users and host computers. The well-known example of such a network is the ARPANET, which was developed to provide enhanced computational resources to researchers at a variety of different institutions. For example, the MIT Multics time-sharing system is available through the net, as is the ILLIAC IV at NASA's Ames Research Center and an IBM 360/91 at U. C. L. A. The network, as a VAN, has among its capabilities the ability to support long file transfers, the querying of remote data-bases, and multi-processing, which can occur at geographically distinct sites. A logical map of the ARPANET appears in figure 1.

In implementing a VAN, one creates a subnetwork of computers whose function is strictly within the communications realm — the transfer of messages between the "host" processors (e. g. , the ILLIAC of ARPANET). Appropriately enough, this net is referred to as the communications sub-net and the computers utilized therein are referred to as communications processors.

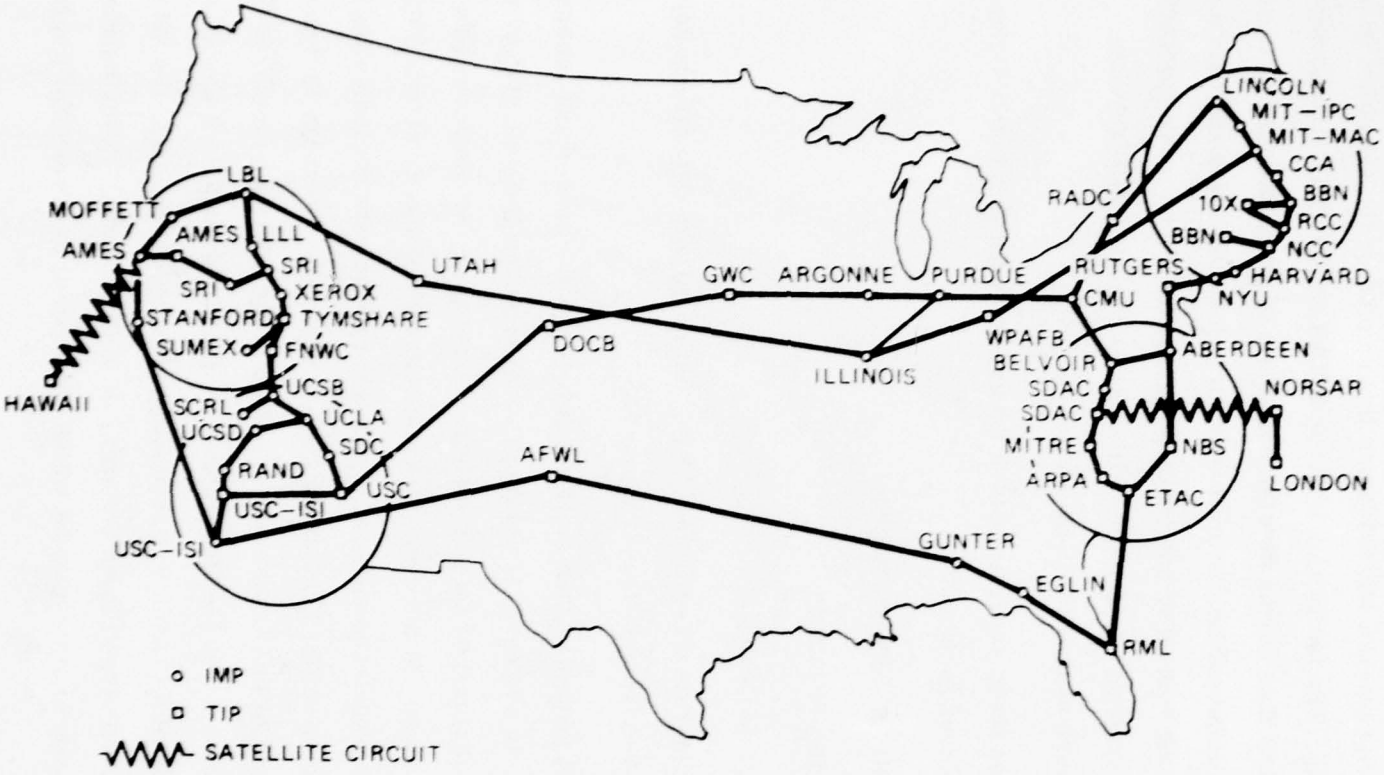
In the ARPANET the communications processors are referred to as IMPs (Interface Message Processors). In the event that the IMPs are connected only to terminals, and not to host computers, they are then referred to as TIPs (Terminal Interface Processors).

A geographical map of the ARPANET, as of June 1975, is included as figure 2.



From KLEI76, p. 304. Reprinted by permission.

Figure 1. Logical Map of ARPANET, June 1975



From KLEI76, p. 309. Reprinted by permission.

Figure 2. Geographic Map of ARPANET, June 1975

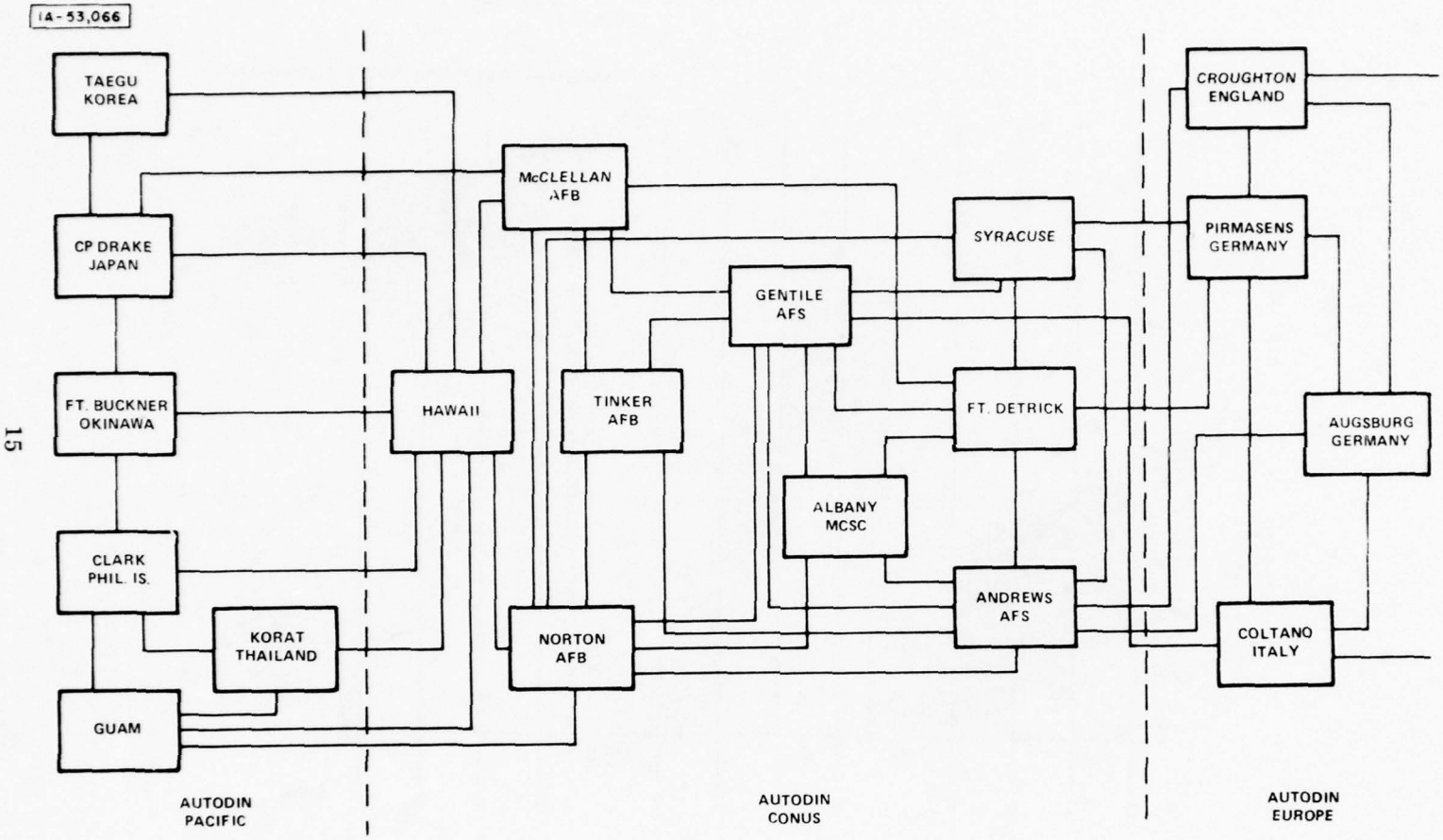
The highest level of our taxonomy is occupied by the Mission-Oriented Network (MON). The distinction between a VAN and MON is strictly organizational. Rather than have a diverse collection of users attempting to access the communications subnet for their own purposes, the hosts and the subnet operate under the control of one single organization. The proposed SACDIN network is an example of such a network, as is the AUTODIN network. Figure 3 is a logical map of AUTODIN.

There are three separate point-to-point transmission technologies presently employed in networks:

1. Circuit (or Line) Switching
2. Message Switching
3. Packet Switching

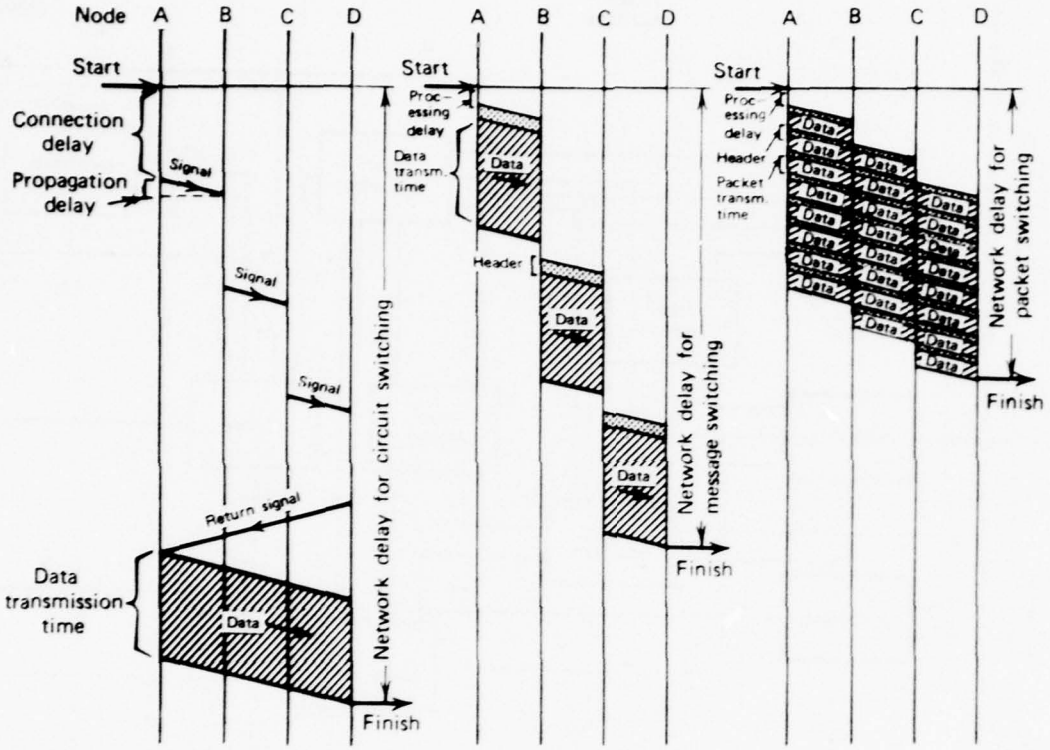
Figure 4 is a diagram taken from KLEI76 that compares the network delay time for the three types of transmissions.

In circuit switching, the entire path between the sender and receiver (the two hosts, in our case) is set up prior to message transmission by a special signal that makes its way through the network, seizing available channels as it goes. In the event that a successful path is established through the network, a return signal is sent back through the network indicating that transmission may begin. The entire path between the two users is then completely dedicated to this transmission. This is the approach employed by the telephone network. The DATRAN network [FLAT74] (before it passed on), was an example of a circuit switched data network. In general, the set-up time for a dial-up circuit is on the order of 5-20 seconds. Nevertheless, the increasing speeds of transmission which came about as a result of improved



Based on KIMB75, p. 136.

Figure 3. Logical Map of AUTODIN



From KLE176, p. 295. Reprinted by permission.

Figure 4. Message Delays for Circuit, Message, and Packet Switching

switches made the DATRAN network an operational possibility. The interested reader should consult FLAT74.

In message switching, the entire message is first sent from the "sender" to a predetermined node. The message is then sent along the next channel of its journey to a third node, and so forth, until it eventually reaches its destination. If a channel is busy, the message simply waits in an output buffer until the appropriate channel is idle, or it may be routed along an alternate channel. The SITA and AUTODIN networks are both message-switched networks.

Packet switching differs from message switching in that each message is broken into small segments (or packets) of predetermined length. Then, the packets are sent independently of one another in "store-and-forward" fashion through the network. (Both message and packet switching are referred to as "store and forward" switching.)

Figure 4 illustrates one of the major advantages of packet switching — namely the "pipelining" effect. The essence of this effect is that if the message length is not too long, the message delay time for packet switching will be considerably less than for circuit switching, and will also be less than for message switching. Indeed, the delay time for message switching is found to be proportional to the product of the message length and the number of hops required, while the delay time for packet switching is proportional to the product of packet length and the number of hops, plus a term proportional to the message length.

If the transmission requirements consist basically of long messages (e.g., file transfers), then circuit switching would be the preferred approach. On the other hand, in a somewhat more mixed ("bursty") environment,

consisting of both long file-transfers and brief interactive messages, packet switching provides a very effective approach.

A number of discussions of packet switching have appeared in the literature. The interested reader might wish to consult CROW75 or KLEI76.

The ARPANET is the best-known example of a (distributed) packet-switched network. As indicated by the ARPA geographical map (figure 2), the network links approximately 100 computers throughout the United States to one another as well as to some sites in Hawaii and Europe (achieved by satellite). Each host computer is connected to an IMP via a 100 KB/sec asynchronous channel. The IMPs are then connected to one another via synchronous 50 KB/sec full duplex channels. Each host computer has a network control program resident in its operating system. This program allows host computers to communicate with one another via a host-host protocol. Another program, TELNET, serves as an interface between the user and the network control program.

In order for a user, located at a terminal, to access a remote host computer, the user's byte stream is first broken into messages (maximum size approximately 8000 bits) by the user's host computer and is then delivered to its associated IMP (along with the message destination). The IMP then breaks the message into packets (maximum size 1008 bits), determines the route for each packet, provides error control and message buffering "services," and handles acknowledgments of messages from host to sender.

In the last few years, interest has focused on the possibilities of combining circuit switching and packet switching in one network. The circuit switching would be employed for long file transfers, while the packet switching would be utilized for the shorter, interactive messages. Some

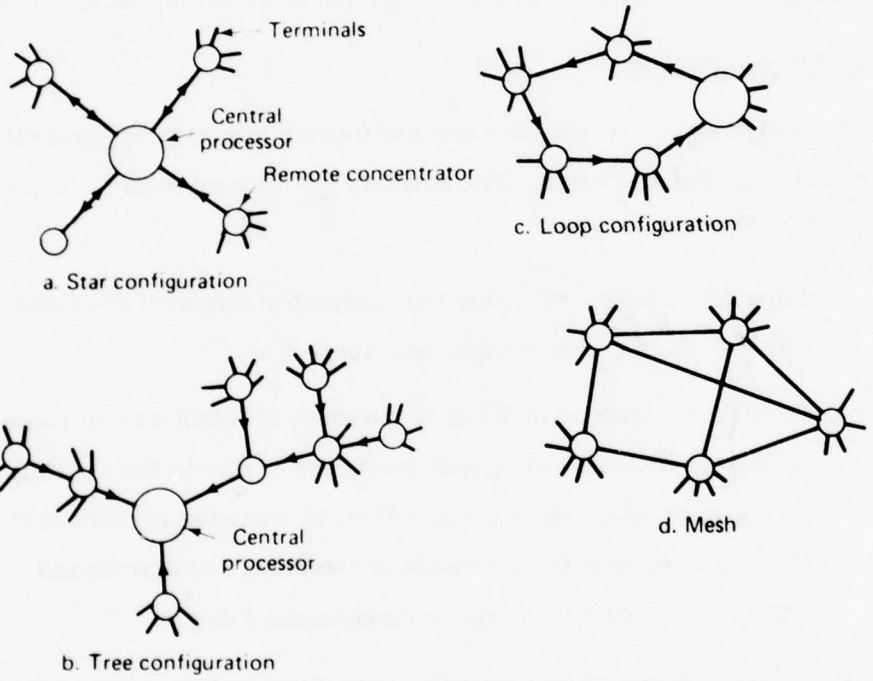
studies have attempted to determine which sort of messages would be best suited for each of these technologies [CLOS72], while others have attempted to analyze the performance of these networks [FISC76].

Further complicating the issue is the possibility of integrating both voice and digital networks, as well as the possibility of digitizing and packetizing voice traffic. Finally, using satellites as transmission facilities lends yet another level of complication to the choice of switching technologies.

Design Considerations

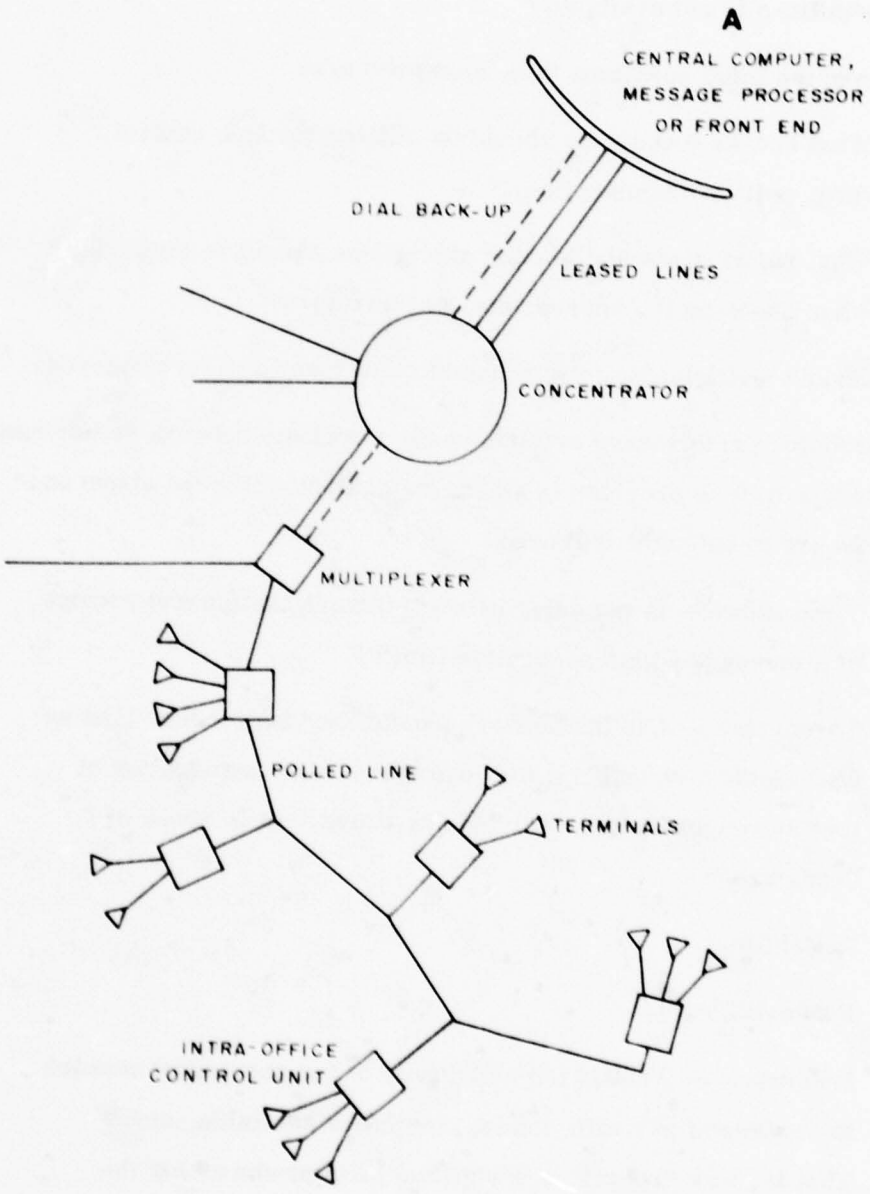
In designing a network, one can distinguish between two general problem areas — global and local. The primary global problems to be addressed are:

- Capacity Allocation — How much capacity (bit/sec) does one allocate to each communications line?
- Routing Algorithms — What is the most efficient way to route messages between users and resources? Should the routing be static or adaptive (that is, adjust to changing conditions)? Routing is especially important in the design of distributed networks, a point that will be emphasized later.
- Network Topology — Given the user (terminal) locations, the resource locations, and the traffic statistics, how can one locate concentrators and multiplexors, and then connect the resources with the users in a cost-effective and reliable manner? Some well-known topologies are represented in figure 5. A "typical" configuration which characterizes either a local access or centralized topology is shown in figure 6.



From SCHW77, p. 4. Reprinted by permission.

Figure 5. Network Topologies



From PIC K76, p. 59. Reprinted by permission.

Figure 6. Local Access Configuration

- Reliability — How does one ensure that there will be alternate communication paths in the event that some collection of nodes and links is not available?

Among the local questions to be answered are:

- What access techniques should be utilized for line control (e.g. polling or contention)?
- What buffer sizes and loading algorithms should be employed when choosing the appropriate concentrators?
- Should multiplexing or concentration of messages be employed?

The major performance criteria used to evaluate network performance (which must therefore provide the standards against which the above considerations are to be evaluated) are:

- Time Delay — Is the delay between the origination and receipt of a message within acceptable limits?
- Throughput — Can the network handle the "busy-hour" load as well as the "normal" traffic load without the introduction of lost messages and excessive delay times, not to speak of deadlocks?
- Reliability
- Network Cost
- Sensitivity — What is the sensitivity of the particular network to variations in traffic loads, equipment available, etc.? That is, how "robust" a design has been produced for the network?

A formulation of the design problem for computer-communication networks might be phrased as follows [PICK76, p. 54]:

Given: Location of data sources and sinks, and the required data flow between them.

Find: Minimum cost network to meet these requirements by choosing the topology, location of multiplexing, concentrating and switching devices, capacities of the links and the routing strategies; subject to

Constraints: Reliability, Delay, Flow

In order to avoid any optimism about the nature of this problem, some of the difficulties inherent in its solution should be pointed out. These include:

- Number of Possible Networks — It can be shown that there are

$$\frac{\frac{N(N-1)}{2}}{\left[\frac{N(N-1)}{2} - M\right]!\ M!}$$

  ways of arranging M links among N nodes. Obviously, a brute force "look at everything" approach is out of the question.

- Discrete Elements — Capacities for lines (and other elements) come in discrete sizes. For example, line speeds can be 2400, 3600, 50,000 bits/sec. Since integer programming techniques are not sufficiently advanced to handle problems of ordinary size, approximate methods are forced on us.

- Nonlinearities — Time-delay functions, component-cost structures, etc. are nonlinear. Hence, we are faced with the difficult area of nonlinear optimization. The nonlinear functions are neither the concave, nor convex functions for which algorithms exist.

The above difficulties are in reality but a sample of the true difficulties associated with the global design of a computer-communications network. The heart of the dilemma lies in the fact that all the problems are interrelated. The routing affects the choice of capacities, and vice versa. The choice of topology is dependent on both capacity and routing. Another area, reliability, is ill-understood and at present only approachable via brute-force techniques. The design problem is greatly compounded in the case of large networks.* Other complicating factors include a tariff-structure that constantly changes, the incorporation of satellites in communications networks, proposed mixtures of voice as well as data networks, and the integration of packet and circuit switching in the same network.

The approach, then, is to attempt to (rationally) separate the design considerations, and then to systematically iterate towards a solution. All hope of an optimal solution must be abandoned, and replaced with a willingness to accept an heuristically produced, yet <u>feasible</u> design.

---

*The paper HSIE76 provides an illuminating discussion of the difficulties involved in the design of a large network as well as algorithms for switch location.

SECTION 2

DESIGN OF COMPUTER-COMMUNICATIONS NETWORKS — ISSUES AND APPROACHES

In this section, we outline some of the considerations associated with the design of computer networks, and present, in summary form, some of the major algorithms involved in the design of centralized and distributed networks.

A discussion of several simulation approaches to reliability is presented in section 3 as part of the discussion of Network Analysis Corporation's (NAC's) models. We chose to present it in section 3 because NAC's models represent the state of the art in this area. A detailed survey of the subject may be found in FRAN70a.

One distinguishes, in general, between two types of networks — centralized and distributed. The fundamental difference between these two types of networks is reflected by their names. In a centralized network, there is one computational facility and all the users are attempting to gain access to it, while a distributed network has a number of (linked) computational facilities that a number of users are attempting to access.

Among the potential topologies for a distributed network,* two stand out at opposite ends of the continuum — the ring network and the completely connected network. These topologies are illustrated in figure 7.

*We take this opportunity to remind the reader that we are really talking about the organization of the communications subnetwork, in spite of our casual use of the word "network."

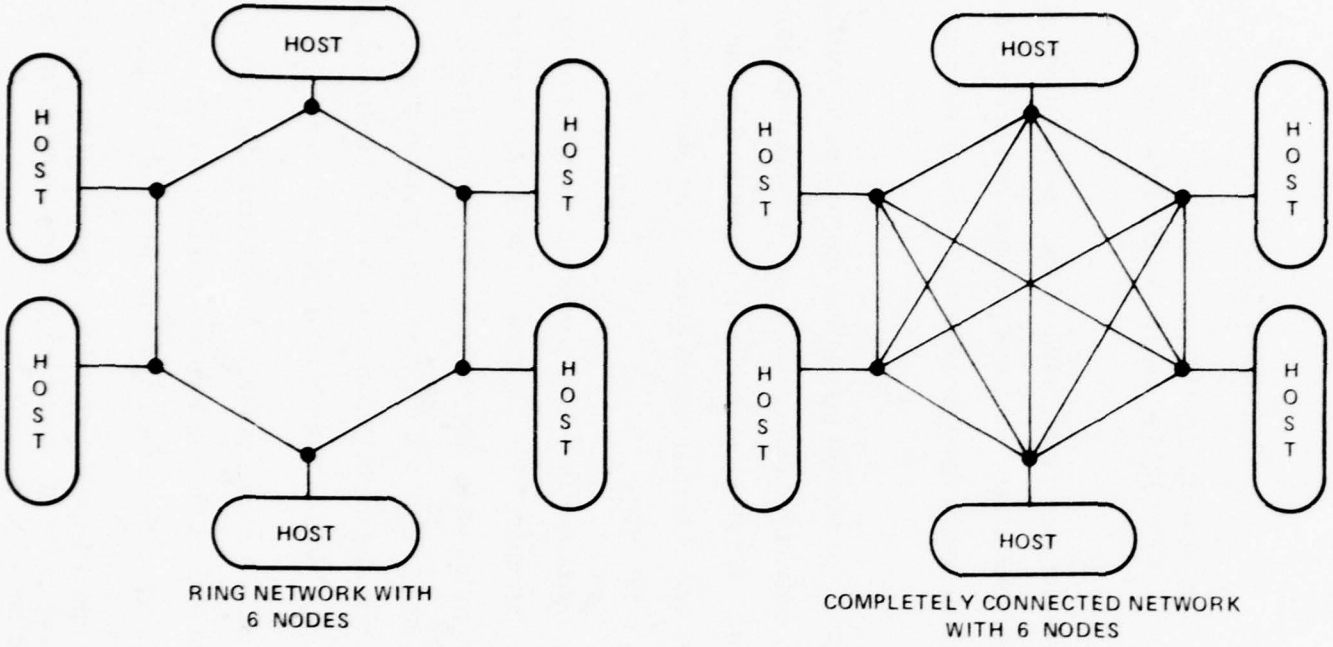


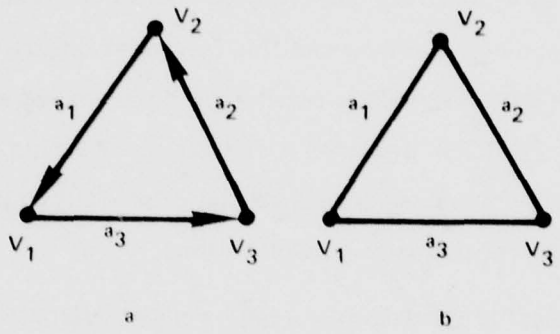
Figure 7. Ring and Completely Connected Networks

The ring network minimizes line-related costs but presents some potential reliability problems, since one line (or node) failure threatens to disconnect the entire network (unless traffic can be reversed along duplex transmission lines). FARB72 provides a discussion of a ring network. The completely connected network possesses no such line-failure problems, but does present some interesting cost considerations.

With the advent of large networks (100 or more computer facilities, and associated terminals), the notion of hierarchies within the network has sprung up. The hierarchy consists of a high-level backbone of "important" nodes (implemented as a packet-switching network), and a second tier of local-access networks (which might be implemented as centralized networks). The interested reader should consult GERL75 for a discussion of the issues and problem areas of large-scale network design.

We proceed then to discuss centralized and distributed networks, and present some salient algorithms associated with their design. Since design algorithms are often stated in graph-theoretical terminology, a few (very) elementary definitions precede our discussion.

A graph is essentially a collection of nodes (or vertices) and links (or arcs). Symbolically, $G = (V, \Gamma)$ where V is the set of vertices and Γ is the set of arcs. In our situation, the nodes represent the concentrators or computers, while the arcs represent the transmission lines. A path is a sequence of arcs on the graph. (An example of a path is a_1, a_3, a_2 in figure 8a.) The ring and completely connected networks of figure 7 provide two examples of graphs. The ring network is referred to as a cycle in graph-theoretic terminology. The completely connected network is called a completely connected graph on six vertices. A completely connected graph on n vertices has each of its nodes connected to all the remaining nodes in the graph.

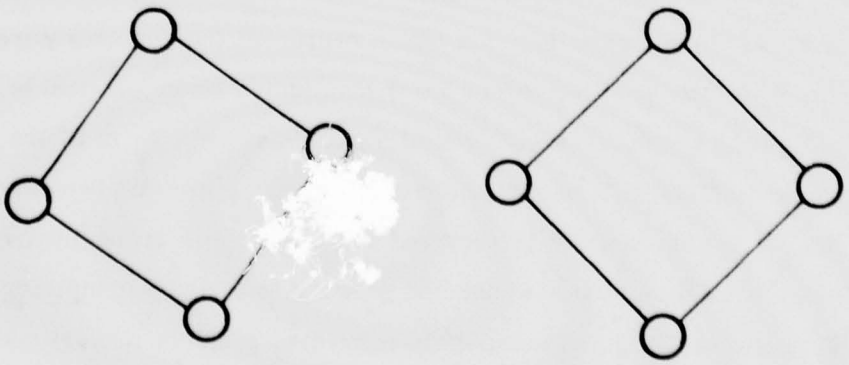


IA-53,068

Figure 8. Directed and Undirected Graphs

Graphs are often distinguished as directed and undirected. For example, the graph in figure 8a is directed (each of its arcs has an arrow on it), while the graph of figure 8b is undirected. Physically, this might correspond to simplex transmission lines oriented in the direction indicated by the arrows. The reader will note that both of these graphs are cycles.

A connected graph is one in which there is a path between every pair of vertices. The graph of figure 9 is disconnected. The notion of connectivity in a graph has important bearing on the design of survivable (nonvulnerable) communications networks.



IA-53,082

Figure 9. Graph with Two Components

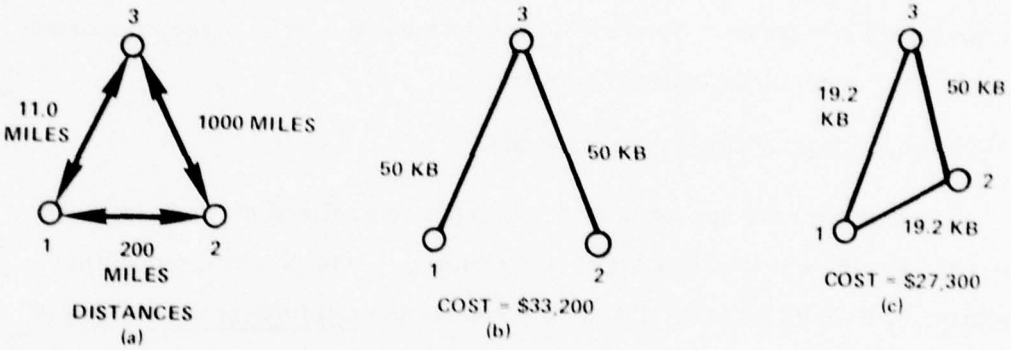
A tree is a connected graph with no cycles: it can be shown that this is equivalent to requiring the (connected) graph to have exactly one path between any two nodes. This last property turns out to be a very important one in the design of centralized networks.

DESIGN OF CENTRALIZED NETWORKS

In general, the approach to designing a centralized network is to attempt to define a tree topology. Conceptually, this is a very appealing approach, as it is an easy matter to generate the minimal spanning tree of a graph (i.e., a tree that will connect all the nodes and have the shortest total arc length). Such a tree corresponds physically to connecting all the nodes with the shortest-distance communications links. We will discuss this algorithm later in this section.

It should be made clear from the outset that this is not necessarily the ideal or "optimal" topology, by any means. Frank and Chou, in their paper "Topological Optimization of Computer Networks" [FRAN72] give several examples of tree topologies that are not minimal. The example presented in figure 10 is taken from this paper. As implied by the monthly line costs chart, we are given the option of connecting nodes 1 and 2 (the users) to the central node (3) via either 19.2 or 50 KB lines. It turns out to be more economical to connect the nodes in the fashion indicated in part (c) than via the tree topology of part (b): this of necessity forces us to route some of the flow between nodes 1 and 3 via node 2 (at least 5.8 KB/sec).

Another example mentioned in the same paper points out that a minimal spanning tree can be improved via the addition of so-called "Steiner Points" [FRAN72, pp. 1388-1389].



TRAFFIC MATRIX (KBPS)

NODES	1	2	3
1	0	0	25
2	0	0	40
3	25	40	0

Based on FRAN72, p. 1388.

MONTHLY LINE COSTS

CAPACITY	($) DATA SET	COST PER MILE
19.2	850.0	$7.50
50.0	850.0	$15.00

Figure 10. Centralized Network Example

In spite of these caveats, a tree topology does possess certain definite advantages. Most notable among them are:

- If link costs are convex functions of capacity, then some tree solution is indeed optimal [YAGE71]. As pointed out previously, however, link costs are often discrete functions.
- In a tree topology, there is a unique path between the user and the computer. This obviates the need to consider routing algorithms in determining the network topology, and makes life much simpler for the network designer.

- In the event that the link charges are high, it is sensible to minimize the number and length of the edges. In such a case, a tree topology should be considered [ROTH71].

- It can be shown [ROTH71] that given any "optimal" topology, there exists a tree within the network such that all unsaturated links are in the tree. Hence, the "optimal" topology deviates from a tree topology to the extent that saturated links are added to it.

A further note of optimism inserts itself when we note that even in the event that one is willing to restrict one's options to trees, a globally optimal solution still cannot be found. It is, however, possible to assign capacities optimally to a given tree topology. A design procedure for centralized networks has been devised based on this algorithm. FRAN71b contains an interesting discussion of this approach.

We restrict ourselves in this discussion to a network design technique which consists of two fundamental stages:

1. An algorithm that locates the position of concentrators within a network.

2. Several algorithms that are used to connect the terminals to the concentrators.

As promised, the design procedure, even in the "simple" centralized case, is hierarchical in nature.

The algorithms presented will be referenced in the next section, when they are described as part of the comparative study of the software packages examined during the course of the survey.

Location of Concentrators

The two algorithms described here are referred to as the "add" and "drop" algorithms in the literature. The add algorithm appears to be the most commonly used algorithm for locating the position of concentrators within a network. Our description will follow that of Schwartz [SCHW77].

The statement of the problem is:

Given: (1) A set of n terminals $\{T_1, \ldots, T_n\}$, and their locations.

(2) A central facility S_0.

(3) A potential set of concentrator locations $\{S_1, \ldots, S_m\}$.

Choose: A subset of the concentrator locations, and connect the terminals to S_0 via the concentrators chosen or directly to S_0.

Note that the concentrators are to be connected to S_0 via a high-speed line, to accommodate the increase in traffic, while the terminals, when directly connected to S_0, will utilize low-speed lines.

The following restrictions apply to the problem:

- Concentrator S_j has a fixed cost f_j, which consists of both the hardware and the cost of connecting it to the central facility S_0. ($f_0 = 0$)
- Each concentrator has a maximum "port" capacity e. The port capacity refers to the maximum number of terminals that the given concentrator can accept. e_0 is assumed $\geq n$.

In practice, of course, the port capacity can vary depending upon the concentrator chosen, but this results only in further extensions of the algorithm.

- c_j is the cost of connecting terminal i to concentrator j. This results in an m x n cost matrix (array).

To formulate the problem mathematically, we introduce the following notation:

$$x_{ij} = \begin{cases} 1 & \text{if } T_i \text{ is connected to } S_j \\ 0 & \text{otherwise} \end{cases}.$$

Concentrator S_j is said to be open if it is in use, and closed if it is not in use. Defining

$$y_j = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

we note that $y_j = 1$ if S_j is open and $y_j = 0$ if S_j is closed.

Letting Z be the cost function, we find the following representation for the problem:

Minimize:

$$Z = \sum_{j=0}^{m} y_j f_j + \sum_{i=1}^{n} \sum_{j=0}^{m} x_{ij} c_{ij}$$

The first term is the cost of the concentrator, while the second represents the cost of the terminal-to-concentrator links.

Z is to be minimized subject to the following two constraints:

1. $\sum_{j=0}^{m} x_{ij} = 1, \; i = 1, \ldots, n$.

   Each terminal i must be connected to some concentrator. (This includes the possibility of the central facility S.)

2. $\sum_{i=1}^{n} x_{ij} \leq e, \; j = 1, \ldots, m$.

   No more than e terminals may be connected to concentrator j.

When the problem is posed in the above manner, we have on our hands a 0-1 integer programming problem. While it is possible to employ branch-and-bound techniques in solving it, these are very time-consuming, and the solution represents, in fact, only a portion of the overall network-planning problem. Hence, the algorithm to be described turns out to be heuristic in nature.

We chose to describe the algorithm using an example taken from SCHW77. Six terminals, $T_1 \ldots, T_6$ are to be connected to three concentrators, S_1, S_2, S_3, or to the central node, S_0, as shown in figure 11.

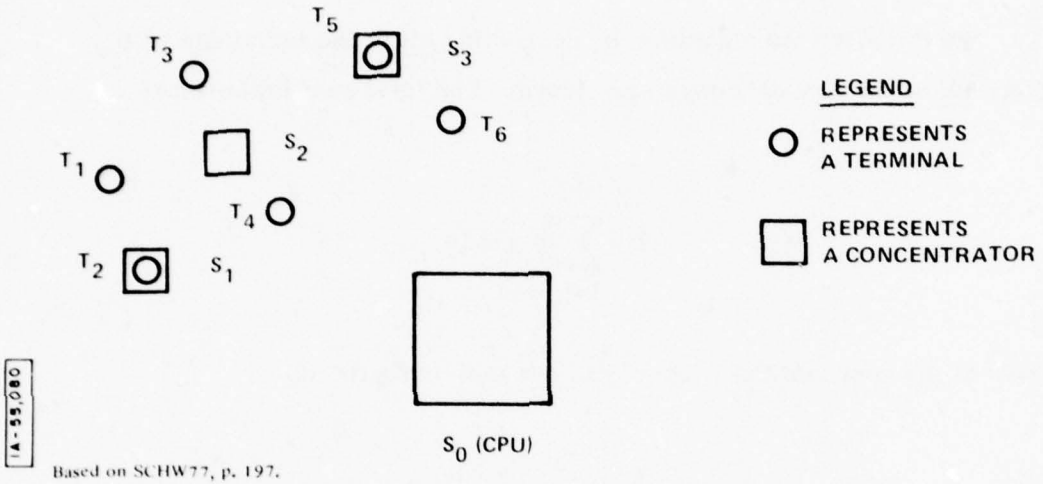


Figure 11. Concentrator Location Problem

For the purposes of this example, we also assume that:

- e = 3. That is, no more than 3 terminals may access the same concentrator.
- $f_1 = f_2 = f_3 = 2$. That is, all of the concentrator costs are the same (= 2).
- The matrix of costs (c_{ij}) of terminal-concentrator connections is given as follows:

$$
\begin{array}{cc} & S_j \\ & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ T_i \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \left(\begin{array}{cccc} 2 & 1 & 2 & 4 \\ 1 & 0 & 1 & 2 \\ 4 & 1 & 2 & 2 \\ 1 & 2 & 1 & 2 \\ 2 & 3 & 2 & 0 \\ 4 & 4 & 3 & 2 \end{array} \right) \end{array}
$$

ADD Algorithm

We initialize the algorithm by connecting all of the terminals to S_0. Hence all of the concentrators are closed. The total cost is therefore

$$Z = \sum_{i=1}^{6} c_{i0} = 14$$

(refer to the cost matrix). This is illustrated by figure 12.

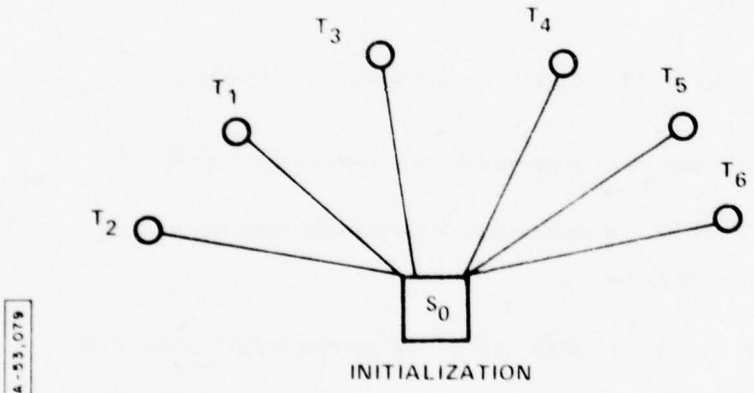


Figure 12. Initialization of Add Algorithm

We proceed by opening up the concentrators one at a time, and searching for the maximum decrease in cost.

Iteration 1. We open up each concentrator, one at a time, to determine that location (for the first concentrator) which will produce the greatest decrease in cost.

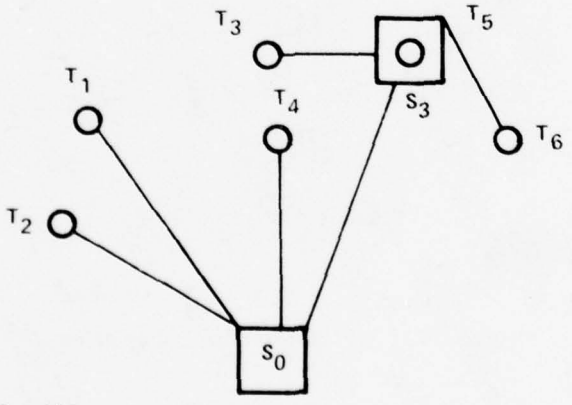
Thus, opening S_1, we connect the terminals that produce the greatest cost savings $(c_i - c_{i0})$ up to a maximum of $e = 3$ terminals. The terminals chosen for connection to S_1 are T_1, T_2, and T_3. The remaining terminals are connected to S_0.

Hence,
$$Z_1 = \sum_{i=4}^{6} c_{i0} + f_1 + \sum_{i=1}^{3} c_{i1} = 11$$

Next, we open up S_2, and discover that terminals 3 and 6 should be connected to S_2. In this case $Z_2 = 13$.

Finally, we open up S_3, and discover that terminals 3, 5, and 6 should be connected to it. In this case $Z_3 = 10$.

Since Z_3 represents the smallest cost, we connect the terminals as shown in figure 13.



IA-53,078

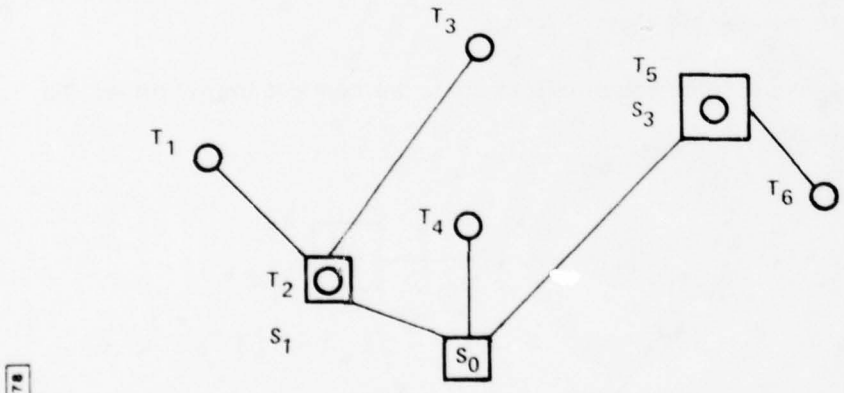
Based on SCHW77, p. 199.

Figure 13. Add Algorithm, Iteration 1

Iteration 2. We now leave S_3 open, and check the other two concentrator locations, S_1 and S_2, successively to see if any improvement in cost results. Note that we are going to check all of the terminals, including those already connected to S_3.

By connecting T_1, T_2, and T_3 to S_1, we find a decrease of three cost units and an increase of two, resulting in a net decrease of one cost unit from this arrangement. Hence $Z = 9$ at this stage.

Repeating the procedure for S_2, we produce no improvement in our cost structure. Therefore we terminate the algorithm with $Z = 9$ and with the arrangement of terminals and concentrators shown in figure 14.



Based on SCHW77, p. 199.

Figure 14. Add Algorithm, Iteration 2

It is the author's observation that the "add" algorithm is really nothing more than the Vogel solution to the allocation problem. Another algorithm in use for this problem is the so-called "drop" algorithm.

Drop Algorithm

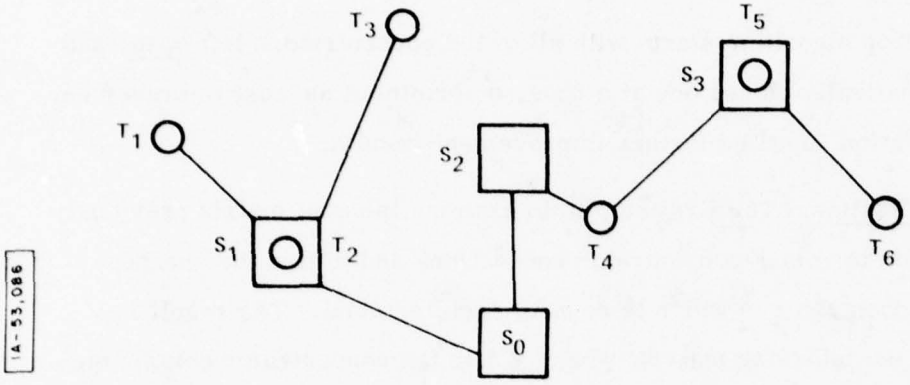
The drop algorithm starts with all of the concentrators left open, and then systematically closes one at a time, determining the cost improvement at each iteration, until no further improvement results.

Initialization. The first step is to examine the cost matrix previously developed for terminal-concentrator connections and to find the cheapest possible concentrator to which to connect each terminal. The result is depicted in the following matrix, where a 1 in the concentrator column signifies the decision to connect a given terminal to that particular concentrator. This process takes into account the concentrator's port capacity.

$$
\begin{array}{c} \\ T_i \end{array}
\begin{array}{c}
\begin{array}{cccc} & S_j & & \\ 0 & 1 & 2 & 3 \end{array} \\
\begin{pmatrix}
1 & & 1 & & \\
2 & & 1 & & \\
3 & & 1 & & \\
4 & & & 1^{*} & \\
5 & & & & 1 \\
6 & & & & 1
\end{pmatrix}
\end{array}
$$

This choice results in the configuration shown in figure 15, with a cost of $Z = 11$.

*The choice of concentrator 2 for terminal 4 is arbitrary, as we could have chosen concentrator 0.



INITIALIZATION, Z = 11

Based on SCHW77, p. 200.

Figure 15. Drop Algorithm, Initialization

Iteration 1. We now close each concentrator one at a time, transferring the terminals attached to these concentrators to other locations, and examine the impact on the cost.

Starting with S_1, we discover that transferring T_1 to either S_0 or S_2 results in a cost increase of 1. The same remark applies to T_2. By transferring T_3 to either S_2 or S_3 we obtain, once again, a cost increase of 1. Since $f_2 = 2$, we obtain a net increase in cost of 1 $(3 - f_2)$. Hence we do not close S_1.

Next we close S_2, transfer T_4 to S_0, and produce a net improvement of 2.

Closing S_3 results in an (unacceptable) increase in cost of 1.

Hence, we choose to close S_2, and transfer T_4 to S_0, with a resultant cost of $Z = 9$. The resultant configuration is the same as that produced by the add algorithm's final pass (figure 15).

Iteration 2. Using the configuration obtained from the second iteration, we now determine if closing either S_3 or S_1 results in a cost improvement. No improvement turns out to be possible; therefore we end with the same network configuration produced by the add algorithm.

Connecting Terminals to the Concentrator

In describing these algorithms, we once again follow the exposition given by Schwartz [SCHW77].

Three algorithms are used for the "line-layout" problem.

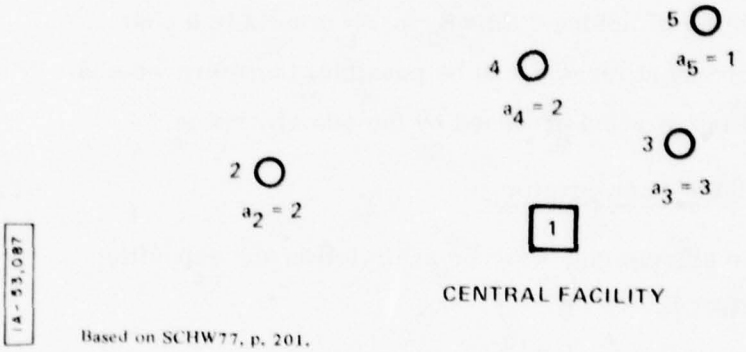
1. The Esau-Williams Algorithm forms the core of the approach most commonly in use today. It is the algorithm employed by the IBM network design program, CNDP (Communications Network Design Program).

2. Kruskal's Algorithm is based upon the minimum spanning tree algorithm, which produces the cheapest network design.

3. The Unified Design Algorithm embodies the Esau-Williams algorithm as a special case, as well as other design algorithms that are currently implemented. This algorithm generates better solutions than the Esau-Williams algorithm as a result of the fact that it includes other algorithms as special cases of it. Hence a number of solutions can be simultaneously generated, and a selection of the best one can be made.

Figure 16, from SCHW77, illustrates the algorithm's operation.

The four terminals illustrated in figure 16 are to be connected to the indicated computer at node 1.



14-53,087

Based on SCHW77, p. 201.

Figure 16. Example for Line-Layout Problem

The traffic generated at each link is given by:

$$a_2 = 2,\ a_3 = 3,\ a_4 = 2,\ a_5 = 1$$

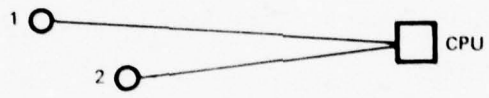
The cost of establishing a link between any pair of nodes is given by the following cost matrix:

		NODE				
		1	2	3	4	5
NODE	1	–	3	3	5	10
	2	3	–	6	4	8
	3	3	6	–	3	5
	4	5	4	3	–	7
	5	10	8	5	7	–

We denote the cost of connecting the two links i and j by c_{ij}, and allow the maximum flow on any link to be 5 units. This maximum flow could correspond to the actual capacity of the lines themselves, or to a maximum allowed flow on the lines that is determined from a maximum time-delay requirement.

Esau-Williams Algorithm [ESAU66]

This algorithm operates by searching out the nodes most costly to connect to the computer and replacing each one by a connection to a neighboring node (which is then connected to the computer). Thus,



might be replaced by

Step 0 Initialize by calculating all trade-off parameters, t_{ij}, where

$$t_{ij} = c_{ij} - c_{i1}$$

for all i, j. Thus t_{ij} measures the difference in cost between connecting node i to the computer <u>via node</u> j, and connecting node i <u>directly</u> to the computer. From our example, $t_{24} = c_{24} - c_{21} = 1$. Clearly, we are in search of <u>negative</u> t_{ij}'s.

Step 1 Select the minimum t_{ij}, and consider connecting i to j. In our example, the expression $t_{53} = -5$ is the minimum.

Step 2 Check to see that the flow constraints are satisfied. (Is the line overloaded?)

IF go to step 3.

ELSE set $t_{ij} = \infty$, and go to step 1.

For our example, $a_5 = 1$ if it is connected, which is clearly less than the maximum of 5. Hence, we continue to step 3.

Step 3 Link i-j is added to our network. Reevaluate the constraints and update the trade-off functions. Go to step 1.

In our example, flow a_3 becomes $a_3 = a_5 + a_3 = 4$, since node 5 is connected to node 3. Functions t_{53} and t_{35} are deleted from our list.

It would be worthwhile for the reader to continue with our example. The final result is depicted in figure 17.

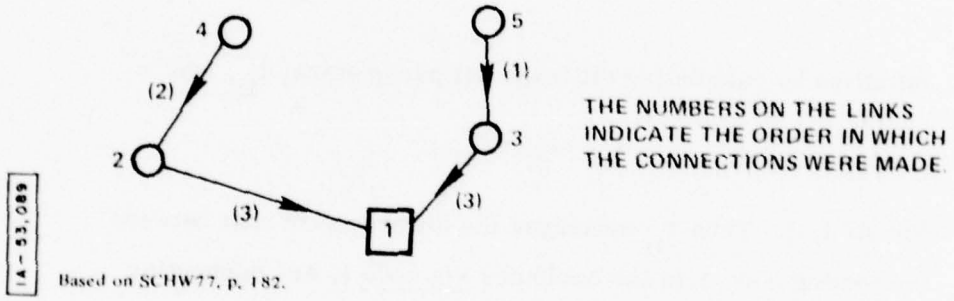


Figure 17. Esau-Williams Algorithm, Final Result

Kruskal's Algorithm

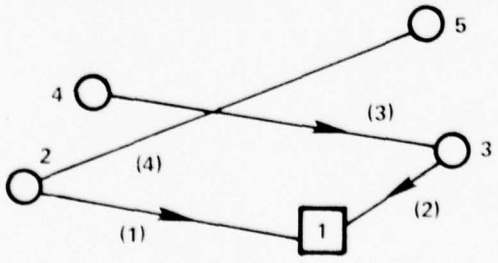
As was mentioned earlier, one of the most conceptually appealing algorithms involved in the design of centralized networks is the minimum spanning tree algorithm. Kruskal's algorithm, which is based on the m.s.t. algorithm, is included here for the sake of illustration, despite the fact that it is not used in any of the commercially available design packages.

The statement of this algorithm is quite simple. Carry out the following step until no longer possible: Among the links not yet selected, choose the least-cost link, checking to make sure that it does not form a circuit with the links already selected, and that flow constraints are not violated. If one does not check for the flow constraints, one produces the minimum spanning tree.

The m.s.t. algorithm provides a lower bound on the cost of the network. All of the algorithms discussed in this section will reduce to this design if all constraints are removed.

Returning to our example, we note that the minimum entry in the cost matrix is 3. Links 2-1, 3-1, and 4-3 all have this same value. Hence we choose randomly among these links, picking link 2-1. The flow constraint is satisfied and the connection is made. Repeating this procedure for links 3-1 and 4-3, we find that these connections can also be made. However, no further connections may be made through node 3, as the flow in link 3-1 has now reached its maximum value of 5.

Continuing in this fashion, we arrive at the network shown in figure 18.



THE NUMBERS IN PARENTHESES INDICATE THE ORDER IN WHICH THE CONNECTIONS ARE MADE.

IA-53,090

Based on SCHW77, p. 183.

Figure 18. Kruskal Algorithm, Final Result

Unified Heuristic Algorithm (due to Kershenbaum and Chou)

Kershenbaum and Chou show that a large number, if not all, of the algorithms used for the line-layout problem may be unified under their banner [KERS74]. As mentioned before, this produces a significant advantage for their algorithm, as no single line-layout algorithm is really perfect for all possible cases. Hence the ability to generate parallel designs is an important one.

The algorithm assigns a weight, w_i, to each terminal, and then computes a trade-off function t_{ij} as follows:

$$t_{ij} = c_{ij} - w_i$$

where c_{ij} is, as before, the cost of connecting terminal i to concentrator j. By defining w_i differently, one obtains each of the algorithms already discussed as well as a host of others. The reader is referred to KERS74 for a more complete discussion of this property. The weights appropriate to each of the algorithms discussed in our paper are listed below (from KERS74, p. 1766):

Algorithm	Initialization	Update When (i - j) Is Brought In
Esau-Williams	$w_i = c_{i1}$, i = 1, . . . , N	$w_i \leftarrow w_j$
Kruskal	$w_i = 0$, i = 1, . . . , N	None

The algorithm proper is as follows:

Step 0 Initialize the w_i for i = 1, . . . , N.

Initialize the constraints.

Set $t_{ij} \leftarrow c_{ij} - w_i$ for all i, j when no constraints are violated by making this connection.

Step 1 Compute $t_{i,j} = \min t_{ij}$ for all i, j, $i \neq j$.

IF $t_{i,j} = \infty$, terminate the algorithm.

ELSE go to step 2.

Step 2 Evaluate the constraints under the connection i' - j'.

IF any are violated, set $t_{i,j} = \infty$, and go to step 1.

ELSE go to step 3.

Step 3 Add link i'j'. Reevaluate the constraints and update w_i as in the KERS74 table, reevaluate t_{ij}, and go to step 1.

One rule for parametrizing the weights w_i which has proved useful is given by

$$w_i = a\left[bc_{i1} + (1 - b)\, c_{i2}\right] \quad .$$

If a = 0, one obtains the Kruskal algorithm, while if a = b = 1, the Esau-Williams algorithm is obtained. Differing values of a and b produce a family of solutions. It is useful to note that the expression in parentheses is essentially a straight line joining c_{i1} and c_{i2}.

The interested reader should consult SCHW77, chapters 9 and 10, for a more complete coverage of design algorithms for centralized networks, as well as MART72, chapters 40 through 43, for the "IBM approach."

DESIGN OF DISTRIBUTED NETWORKS

As indicated earlier, the principal design difficulty arising in distributed networks is the interplay of routing considerations with the remaining problems (capacity allocation, etc.) In light of this difficulty, we present a brief discussion of routing in a distributed network, and follow this with a discussion of a global design algorithm, known as the "cut-saturation" algorithm.

Routing in a Distributed Network

It is important to be aware of the fact that we are concerned with specifying a fixed routing policy, that is, one that does not vary with time according to changing traffic conditions. The objective is to provide a minimum average time-delay design, assuming stationary traffic conditions.

This approach is clearly consistent with our situation, as we are concerned with the design of the network; and, because of the interplay of routing with other design considerations, we must have some idea as to the appropriate distribution of flows in the network. For informative discussions on routing, the reader may wish to consult SCHW77, chapter 11 or KLEI76, chapter 5.

We wish to present an algorithm for routing which was employed in the design of the ARPANET, the Flow Deviation algorithm. First, however, we need an expression for the average time-delay encountered by a packet in a distributed network. Such an expression has been developed by Kleinrock [KLEI76]. In developing the time-delay formula, he assumes Poisson arrivals and exponential message lengths.

In assuming exponential message lengths, Kleinrock invokes the so-called independence assumption — essentially ignoring the interdependence between arrival rates and service times at a given node in the network. This assumption is tenable due to the external arrival of messages at a given node, and has been verified experimentally. See KLEI64 and KLEI76.

Assuming NA channels (arcs) and N nodes in our network, Kleinrock's formula is

$$T = K + \sum_{i=1}^{NA} \frac{\lambda_i}{\gamma} \left(\frac{\lambda_i / \mu' C_i}{\mu' C_i - \lambda_i} + \frac{1}{\mu C_i} + P_i + K \right)$$

where:

λ_i = flow (bits/sec) in the i^{th} channel

γ = total traffic in network

$\frac{1}{\mu}$ = average length of a data packet

$\frac{1}{\mu'}$ = average length of all packets (including acknowledgments)

C_i = capacity (bits/sec) of the i^{th} channel

P_i = propagation delay on i^{th} channel

K = nodal processing time

The first expression in the brackets $\frac{\lambda_i / \mu' C_i}{\mu' C_i - \lambda_i}$ is an expression for the waiting time at the i^{th} channel, while $\frac{1}{\mu C_i}$ is an expression for the service (i.e. transmission time) on this channel. One also includes P_i, the propagation time to transmit one bit down the length of the channel. This result, based on Jackson's theorem [JACK57 and JACK63], is essentially a summation of delay times due to a series of M/M/1 queues.

We now are in a position to present a summary version of the Flow Deviation algorithm, which was employed in the design of the ARPANET.

Step 1 Suppose NA = number of arcs, and $f^n = (f_1, \ldots, f_{NA})$, the vector representing the set of link flows at the n^{th} iteration of the algorithm. Compute:

$$l_i = \partial T / \partial f_i, \; i = 1, \ldots, NA \quad .$$

Step 2 Find the shortest-route flow vector V, utilizing the above l_i as the link "lengths."

Step 3 Let

$$f^{n+1} = (1 - \lambda) f^{n} + \lambda \underline{v} , \quad 0 \leq \lambda \leq 1 ,$$

and employ a search method (e.g. Fibonacci) to locate a λ which minimizes T. A more detailed discussion of this algorithm may be found in KLEI76, chapter 5.

Another algorithm for routing, the Extremal Flows algorithm due to Cantor and Gerla [CANT74], is presented in section 3 of this paper in the discussion of Network Analysis Corporation's model for distributed network design.

Cut-Saturation Algorithm

We present in this section an outline of the "cut-saturation algorithm," which is one of three "known" design techniques employed in distributed-network design. The algorithm was developed by Network Analysis Corporation, and will be discussed in somewhat greater detail later in this paper. The other two algorithms are the Branch-Xchange algorithm, and the Concave Branch Elimination Algorithm. Kleinrock and Gerla [GERL77] attempt a comparison of the performance of these algorithms, and reach some interesting conclusions. One of the most notable is the underlining of the difficulty in even making these comparisons. The authors point out the need for work in this area.

The cut-saturation (c-s) algorithm aims at producing the least-cost distributed network for a given throughput, subject to time delay and reliability constraints. The algorithm as presented below assumes that link capacities are given, and that they are all equal. However, it is claimed that a straightforward extension of the algorithm that includes the

multiple-capacity case is easily implemented. For a fuller discussion of the algorithm, see GERL74.

The algorithm consists of the following five steps (for each iteration):

Step 1 Routing

Starting with a given network design, link flows are found that minimize the overall time delay.

Step 2 Saturated Cut-Set Determination*

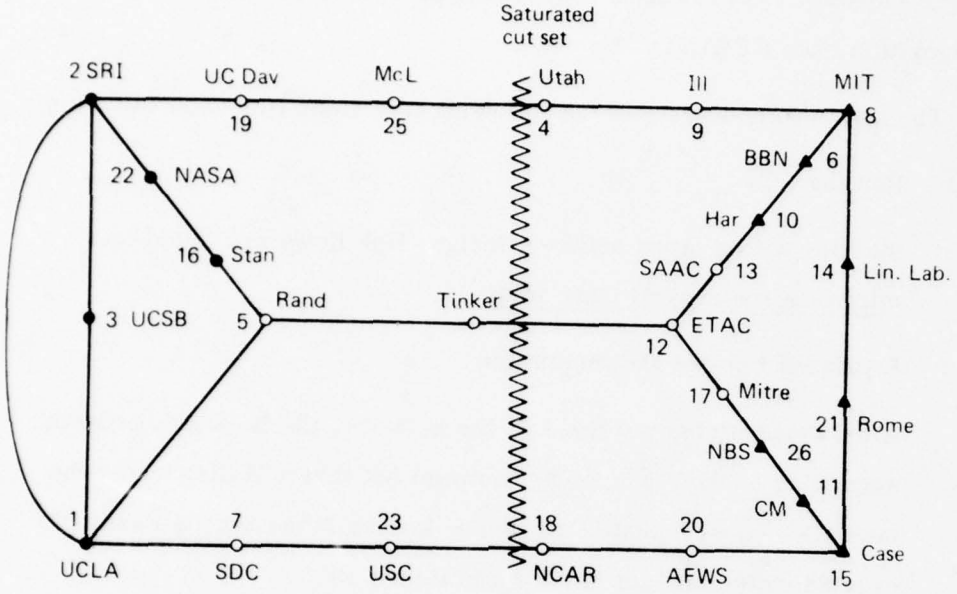
After determining the flows in the network, the links are ordered according to their use. The minimal set that will disconnect the network — the saturated cut-set — is then removed. An example of a saturated cut-set is shown in figure 19.

Step 3 Add-Only Step

The purpose of this step is to add the least-cost nodes to the network — which will divert traffic from the saturated cut-set. Since one would like to divert traffic as far away from the saturated cut-set as possible, and since the cost of communication links increases with distance, one must effect a compromise of sorts. Such a compromise is achieved by using a "distance 2" criteria — nodes that are a minimum of two links removed from the cut-set are chosen as potential candidates for removal. This is illustrated in figure 19.

---

*Gerla, et al. [GERL74] claim that this cut-set approximates the famous minimum cut-set of the Ford-Fulkerson theorem.



From NAC73, p. 4-7. Reprinted by permission.

Figure 19. Saturated Cut in the ARPANET

Step 4 Delete-Only Step

The purpose of this step is to remove links that have the least "marginal utility" from the network, one per iteration. Mathematically, the links are chosen according to the following criteria.

Maximize $$E_i = D_i \cdot \frac{c_i - f_i}{c_i}$$

where

D_i is the cost of the link

c_i is the capacity of the link

f_i is the flow in the link

Step 5 Perturbation Step

This step reduces the network cost once the desired throughput range has been achieved. Using add-only and delete-only operations, the network links are "reorganized," while maintaining the throughput within ± 5 percent of the desired goal.

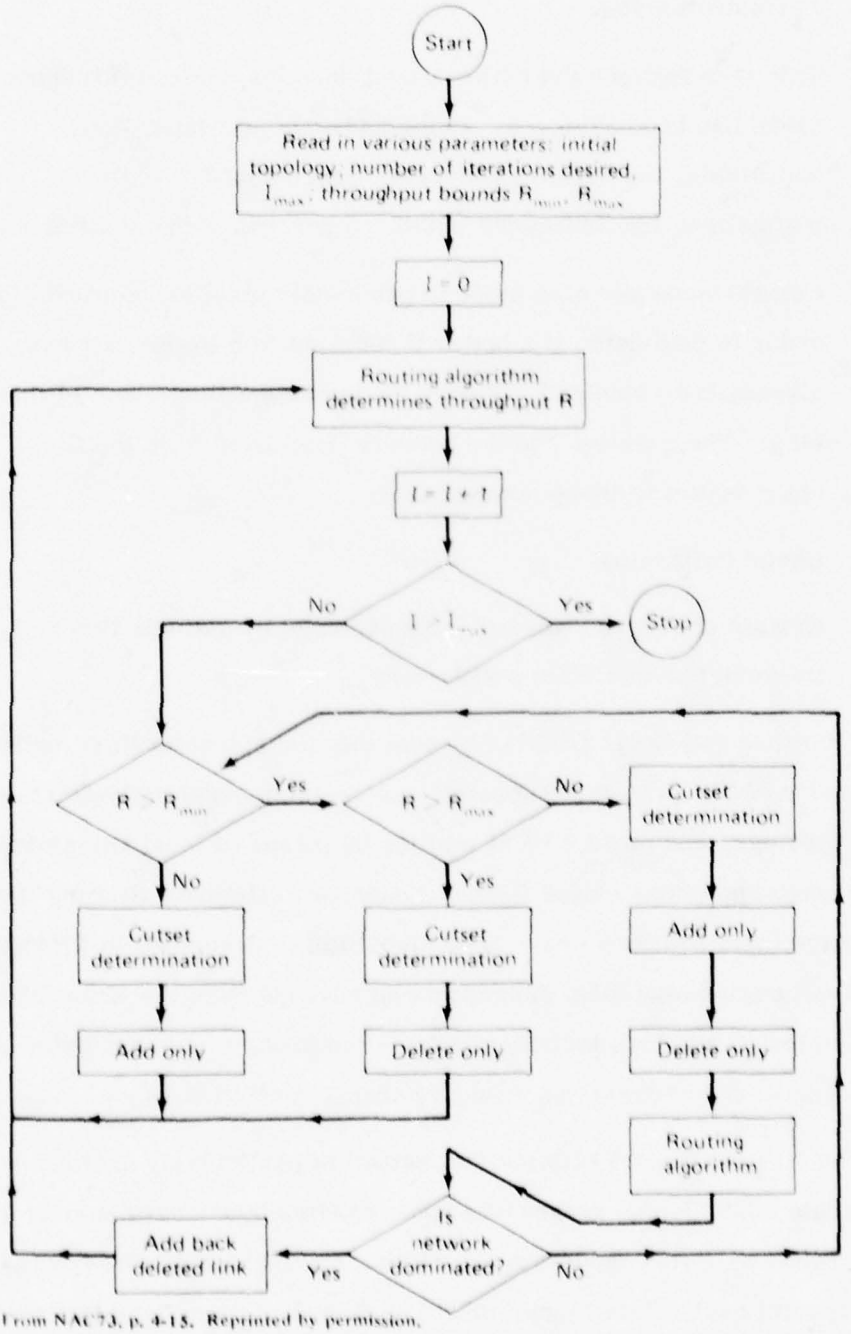
Comparisons are also made to previously obtained solutions in order to determine if a better throughput at a lower cost has already been obtained. Figure 20 contains a flow-chart of this step. The question "Is the network dominated?" in the flow-chart refers to these comparisons.

Step 6 Chain-Collapsing

Certain chains (in "series") are replaced by a single link to improve the algorithm's efficiency.

Kleinrock and Gerla [GERL77] note that the cut-saturation method is an extension of the Branch Xchange method — which essentially starts with a given topology, and obtains local minima by means of local transformations. These transformations, called Branch Xchanges, eliminate existing links and replace them with new ones. The algorithm is described in FRAN70b. The cut-saturation algorithm represents an advance over the Branch Xchange method in that it employs information from the problem domain while modifying the network, whereas the Branch Xchange method does not.

The concave branch elimination method is particularly useful whenever the discrete costs can be reasonably approximated by concave costs. The method relies on a flow deviation algorithm, making use of concave costs, and a capacity assignment algorithm. The flow deviation algorithm tends to eliminate uneconomical links from the topology. Kleinrock and Gerla point



From NAC 73, p. 4-15. Reprinted by permission.

Figure 20. Flow Chart of Saturated Cut-Set Algorithm

out that the choice of the particular algorithm depends upon the cost and capacity structure, as well as upon the degree of precision required. They also present a comparison of these methods for a particular design problem, and reach the conclusion mentioned before — that it is a difficult task to even attempt a reasonable comparison among the existing design algorithms for distributed networks. They suggest that research in the analysis of algorithm complexity should be pursued to these ends.

This is probably a fitting note upon which to end our discussion of design algorithms (and comparisons between them), and launch into a discussion of commercially available design packages.*

*In the event that the reader is not sufficiently intimidated by Kleinrock and Gerla's enjoiner, we remind him of some of the unapproached problems discussed earlier in this section.

SECTION 3

COMPARISON OF MODELS

A number of organizations were contacted in the course of this survey concerning the programs they have written for designing and analyzing computer networks. The organizations were identified as a result of a literature search in which three sources proved to be the most fruitful. These sources were:

- A memorandum written by Mr. Bruce Feldmeyer of MITRE, D71-M-1319, which identified industrial organizations that have produced software packages intended to aid in the design of communications processors (front-ends, concentrators, etc.).
- "Third Annual Survey of Performance Related Software Packages," EDP Performance Review, December 1975.
- Articles in the technical literature describing algorithms employed in the various facets of computer network design.

The organizations were contacted by telephone or letter. In all cases literature was requested concerning their software, with a particular emphasis on technical papers. The respondents in general were cooperative, although one company preferred to discuss its philosophy instead of its product.

Two meetings were held with companies in the course of this project. One meeting was with Scientific Time Sharing (STS) at MITRE's Bedford

offices. STS demonstrated their product, ODIN, and discussed its capabilities. Another meeting was held at the Randolph, Mass., offices of Scientific Applications Inc. (SAI), at which the company's philosophy was discussed.

A list of the organizations discussed in this report appears in table 1. As is to be expected, not all organizations contacted in the course of this project appear in the table — for a variety of reasons. A prime example is IBM, which possesses a tool for network design, but is interested in using it for the design of networks built by IBM, and not by the competition.

The focus of the report is on a comparison of the algorithms employed in the various facets of network design. No hands-on evaluation of the various design tools has been attempted — only an analysis of the algorithms employed. In keeping with this stated approach, the descriptions of the packages are organized along the following lines:

- General — The types of networks designed (centralized or distributed) are mentioned. Specifics of implementation (computers, languages) are also described.
- Functional Organization of the Programs — The functions performed by each of the program modules, as well as their interrelationships, are discussed.
- Algorithmic Structure — We often make use of the discussion in section 2, and simply refer to the appropriate description.
- Summary — Brief concluding remarks that attempt to portray the software constitute the focus of this section.

Table 1

List of Organizations and their Programs

	Organization	Program Name
1.	Kranzley & Co., Telecommunications Division 1010 South Kings Highway Cherry Hill, New Jersey 08034	PLANET
2.	Scientific Time Sharing 7316 Wisconsin Avenue Bethesda, Maryland 20014	ODIN
3.	Systems Architects, Inc. Thomas Patton Drive Randolph, Massachusetts 02368	SADC, SADM, SAND, TALK
4.	Professor G. M. Schneider Department of Computer Science University of Minnesota 114 Lund Hall 207 Church Street, S.E. Minneapolis, Minnesota 55455	VANS
5.	The DMW Group, Inc. 2975 Hickory Lane Ann Arbor, Michigan 48104	NDMS ANDMS
6.	Network Analysis Corporation 130 Steamboat Road Great Neck, New York 11024	GRINDER MIND

One exception has been made to this format: that is the VANS model created at the University of Minnesota. This exception was made as a result of the special nature of the program.

On the whole, two distinct approaches appeared in these design tools. One approach was modeled after IBM's Communications Network Design Program (CNDP). In spite of the secrecy that shrouds the official documentation of this program, James Martin has written extensively about the algorithms employed therein in his well-known book, Systems Analysis for Data Transmission [MART72]. It became quite clear in the course of this project that all of the organizations contacted, with two exceptions, follow the IBM lead in structuring their design tools. The two exceptions were the Network Analysis Corporation (NAC) models, and the VANS model. It should be noted that CNDP is oriented towards the design of centralized networks, and accordingly, the only company that designs truly distributed networks is NAC.

Having concluded our preamble with these words, we now commence our discussion of the individual models.

KRANZLEY & COMPANY — PLANET

General

PLANET (Programs for Line Analysis and Network Engineering Tasks) is essentially a collection of programs, in the CNDP mold, which address different parts of the network design issue. These packages, to be discussed later, may be put together in a variety of ways, depending on the user's specific application(s).

The package, as a whole, is oriented towards circuit-switched and message-switched networks. For networks of the latter category, it aids in the design of centralized networks. It is written in FORTRAN and is available in local batch as well as time-sharing environments. Support, in the form of "post-installation" consulting, is included as part of contractual arrangements.

Functional Organization

The basic modules which comprise the PLANET system are described below.

MODEL is a discrete event-stepped simulation of a communications line. This program produces performance statistics (response time, throughput) for different line configurations and transaction profiles (message lengths, arrival rates, etc.).

NETSYN provides a least-cost configuration of a (private) network, such that this configuration will meet the response-time and throughput characteristics required by the user. Hence, for inputs of terminal, CPU, and modem/concentrator locations, NETSYN produces a low-cost multipoint line layout that will meet the user's response time and throughput requirements.

LOCON is a program that positions multiplexors and concentrators in the network.

WATSYN (WATS synthesizer) is a set of programs for use in the selection of the appropriate WATS service. It develops the sets of bands to be equipped with lines as well as the full/measured split for each band so equipped for inputs on load distribution, performance criteria (e.g., desired grade of service) and costs.

TOLLPRO is a series of programs, designed to gather statistics about long distance calls, that can be used for cost allocation purposes as well as design purposes. The break-up of calls per WATS band can be used as input to WATSYN.

SYSMODEL is used to model computer system performance. For a given configuration it develops response time and throughput characteristics, and also pinpoints bottlenecks in the system.

The first three modules can be arranged as shown in figure 21.

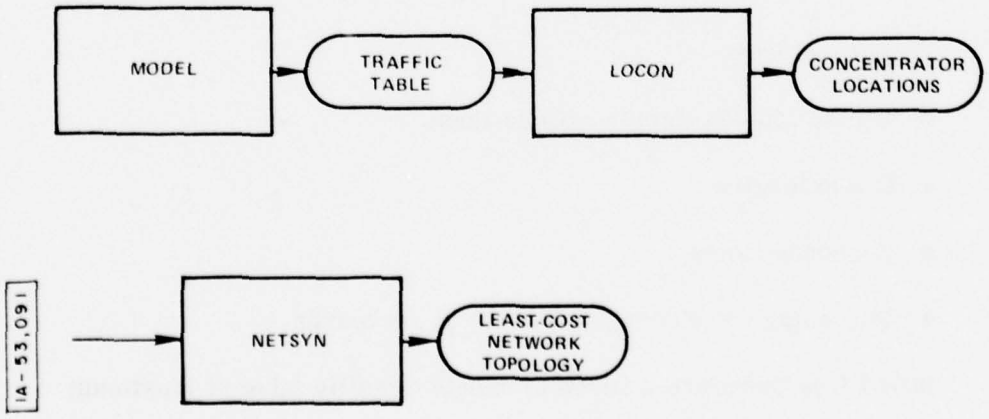


Figure 21. Modules in PLANET

The last module discussed, SYSMODEL, can be used in conjunction with MODEL to produce a more accurate traffic table, as it will include the computer system's performance in creating the traffic table.

Algorithmic Structure

MODEL is, as mentioned before, a discrete event-stepped simulation of a communications line.

The inputs to the simulation include the following:

- Terminal and control unit configurations
- Transaction profiles for each application
  Different types of inquiries and responses (formatted, unformatted, etc.)
  Message lengths
  Mean arrival rates per transaction
- Equipment characteristics
- Response-time objectives

Output includes:

- Utilization of communications lines
- Queue lengths
- Response times
- Percentage of messages delayed at keyboards

MODEL is therefore utilized to create a traffic table of maximum line loadings, to be employed as input to LOCON.

LOCON employs an add algorithm, as discussed in the preceding section, to position the concentrators.

NETSYN essentially employs Kranzley & Co.'s version of the Esau-Williams algorithm for least-cost line-layout as also discussed in the preceding section.

Summary

As indicated earlier, PLANET is essentially written along the lines of IBM's CNDP. Its major drawbacks are a lack of design tools for distributed networks and a lack of reliability models (in the military sense). PLANET appears (on paper) to be a solid, well-maintained product, useful in the design of centralized networks.

SCIENTIFIC TIME SHARING, INC. — ODIN

General

STS markets a network design model called ODIN. (The company seems to have an interest in mythology.) The model is interactive, is written in APL, and is supported by a local (Boston) area office. It should be noted that STS specializes in time-sharing services directed towards APL.

The model is oriented towards the design of a centralized circuit-switched network, and can be utilized in the design of message-switched networks. Essentially, ODIN consists of a series of modules, which can be used in a variety of ways for the design of computer networks. The company provides a well-written user's guide to the system, as well as support services as part of the contractual arrangements.

Functional Organization

Figure 22, which appears in the user's guide to ODIN [ELLI78], presents the modules and their interrelationships.

The ODIN module serves as a system executive: it initializes parameters and determines the options to be utilized in each run. It accepts input in the form of performance goals (response time, throughput), data requirements (tariff environments, message lengths, traffic loadings, etc.) and facilities (terminals, types of lines, modems, etc.) and creates a data set with them. It also constructs a distance matrix of the cities in the network.

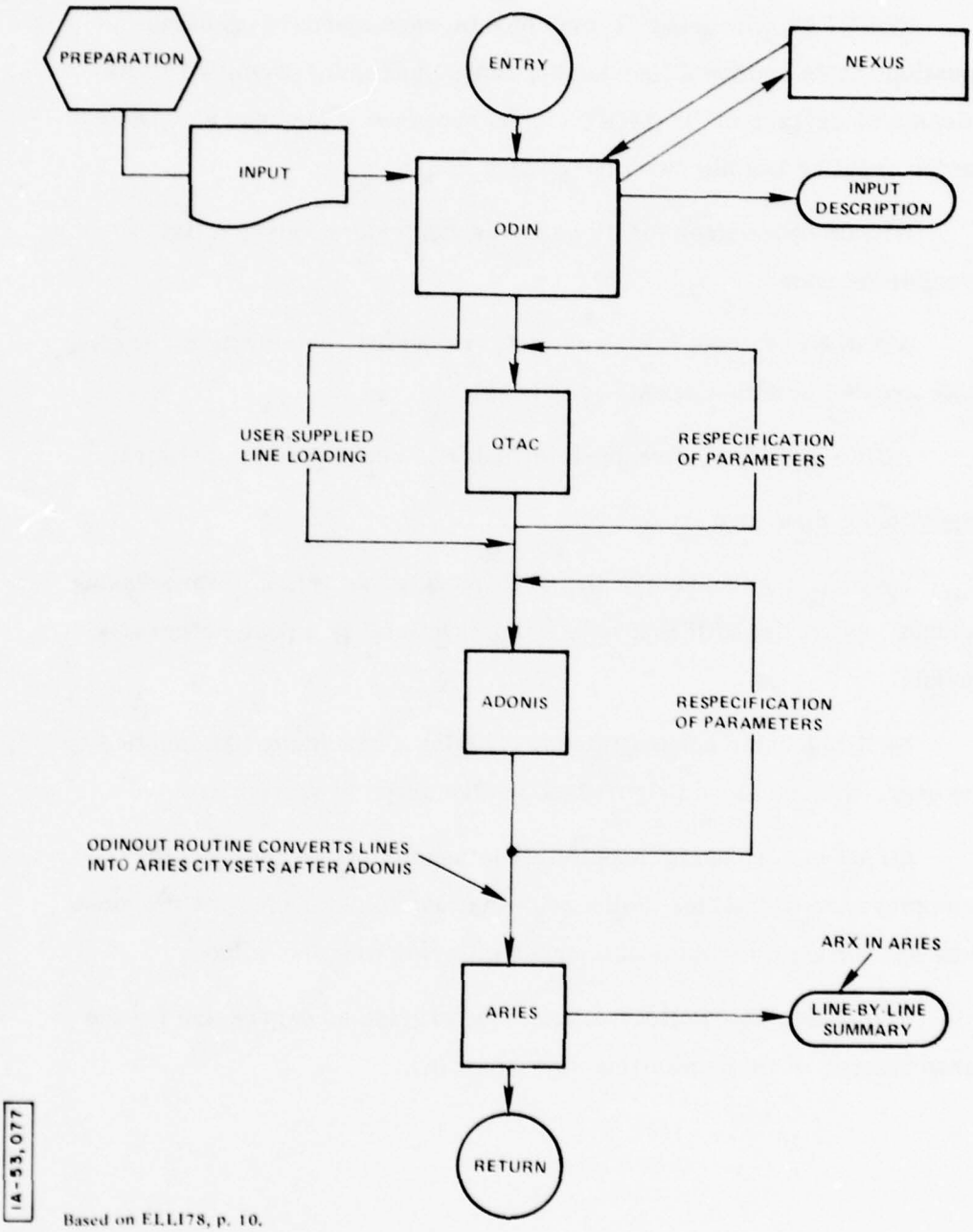


Figure 22. Flow Chart for ODIN Network Design Program

The QTAC (Queueing Theory) module uses standard queueing equations to determine a line-loading table (maximum permitted transmission rates/terminal). QTAC can be bypassed if the user so wishes (in the event he has his own line-loading table).

NEXUS determines locations for concentrators, given a list of possible locations.

ADONIS creates a least-cost network design, employing the loading table created by either QTAC or the user.

ARIES provides a line-by-line "expense report" of the network.

Algorithmic Structure

We omit a discussion of the ODIN module, as it is a preprocessing module, and of the ARIES module which functions as a post-processing module.

NEXUS locates concentrators, utilizing a candidate list supplied by the user. It uses an add algorithm, as discussed in section 2.

QTAC is a queueing module that is used to obtain line-loadings. It employs the Khintchine-Polloczek equations to determine the response time and throughputs obtainable under differing line disciplines.

The Khintchine-Polloczek equations provide an expression for the mean number of items awaiting service, E(w).

$$E(w) = \frac{\rho^2}{2\,(1-\rho)} \left(1 + \left[\frac{\sigma_{ts}}{E_{(t_s)}} \right]^2 \right)$$

where

w = number of items awaiting service.

ρ = facility utilization. In a steady-state model $\rho = E(n)\, E(t_s)$ where $E(n)$ is the mean arrival rate and $E(t_s)$ is the mean service rate.

QTAC uses this equation in building models of line disciplines, and then solves it to build a line-loading table. The line disciplines are essentially combinations of the following six parameters:

Line Operation: Full-duplex or Half-duplex

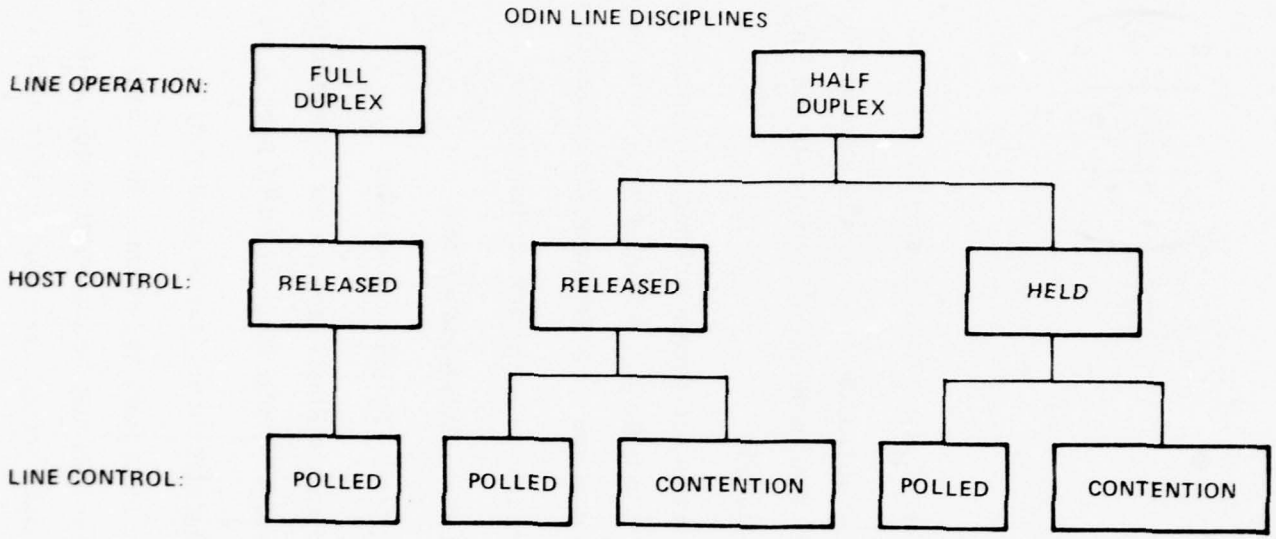
Host Control: Released or Held

Line Control: Polled or Contention

Figure 23 is a diagram illustrating the six models included in QTAC. This figure is taken directly from the ODIN user's manual.

Two polling algorithms may be employed:

1. FDX (Full-duplex) Algorithm — Here outbound messages from the host may be transmitted only while an inbound message is being transmitted or after all of the terminals have been polled once. Polls have priority over messages on the outbound line.



Based on ELL178, p. 83.

Figure 23. Line Disciplines Accommodated by ODIN

2. HDX (half-duplex) Algorithm — Outbound messages have a higher priority than inbound messages. They also have a priority over polls.

Figure 24 (from the user's manual) summarizes the response-time modules.

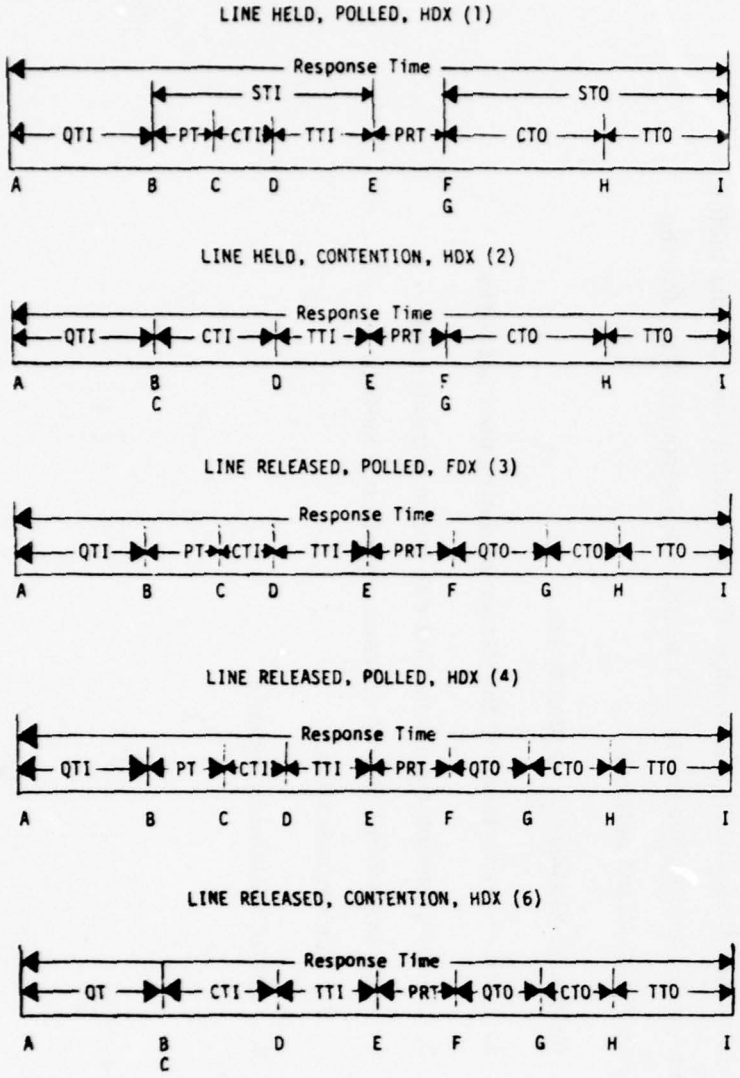
ADONIS is STC's implementation of the Esau-Williams algorithm for configuring least-cost centralized networks. This algorithm was discussed in section 2.

Summary

STS's model is written along the lines of CNDP (again). Its novel features are that it is written in APL, and that it employs queueing models rather than simulation models for developing line-loading tables. On the whole, ODIN appears to be a solid, well-supported tool for the design of centralized networks.

Its major drawbacks appear to be:

1. Lack of provisions for the design of distributed networks.
2. Use of queueing equations that can become inaccurate when evaluating polling (versus contention) models for line organization.
3. Lack of reliability models.



LEGEND

A	Message appears at the station; joins the queue at the terminal
B	Preceding message has reached the HOST
C	Station is polled and gets ready to send the message
D	Station is ready to send the message; transmission starts
E	Message reaches the HOST; processing starts
F	Message has been processed; message joins the Terminal Acceptance queue at the HOST
G	Station gets ready to receive the message
H	Station is ready to receive the message; transmission starts
I	Message has reached the station
CTI	Connection Time for Terminal Entry
CTO	Connection for Terminal Acceptance
PRT	Processing Time
PT	Polling Time
QTI	Queue Time for Terminal Entry
QTO	Queue Time for Terminal Acceptance
STI	Service Time for Terminal Entry
STO	Service Time for Terminal Acceptance
TTI	Transmission Time for Terminal Entry
TTO	Transmission Time for Terminal Acceptance

From F.I.I.178, p. 86. Reprinted by permission.

Figure 24. Response Time for Different Line Disciplines

SYSTEMS ARCHITECTS, INC. — SADC, SADM, SAND, TALK, SAPA

General

SAI markets a series of models that may be used for planning and reviewing a data communications network. The major design programs are interactive in nature, while other supporting programs are accessed in batch mode.

The programs are written in FORTRAN, and are available through a time-sharing service, or may be installed at the user sites.

Unfortunately, SAI declined to give any information as to the nature of the algorithms employed in their packages. A substantive discussion of SAI's products is impossible under the circumstances, and we can only discuss them from a functional point of view, omitting the algorithm description. Some summary remarks are still in order, and these will be presented.

Functional Organization

SAI has five programs applicable to the network design problem:

SADC Systematic Approach to Data Collection

SADM Systematic Approach to Data Management

SAND Systematic Approach to Network Design

TALK A network design package (full name not available)

SAPA Systematic Approach to Performance Analysis

SADC is a data collection module (batch mode), whose function is to create a data-base from user-supplied information as to traffic flow, planned user facilities, equipment, etc. This information may then be used as input to other modules.

SADM is, in the words of SAI, "a general purpose data management program, designed to handle small to medium data base applications" [PANG78]. Its use in the network design context is in the development of input data for the two major design modules, TALK and SANDS, as well as SAPA. Inputs to this module may come from SADC, or be user-supplied.

SANDS is an interactive program whose major purpose is to study various design reconfigurations. The user may start with either the existing network or with an initial design, and then examine the costs and benefits (response time, throughput) for various design changes. Analytical (ostensibly queueing) models are used here.

SAPA is a set of programs that performs the response time/throughput analysis. Analytical models are used for single-host networks, while "simplified simulation models" (in the words of SAI) are used for networks provided by the various vendors. The output of this module also includes maximum permissible line-loadings as well as queue sizes.

TALK is used either to evaluate an existing system or to plan a new one. The review function (performed via simulation) examines the effects of changing grades of service on the performance level. An auditing function keeps track of existing communications facility inventory, and verifies the common carrier billings.

TALK provides a network designer with the following design capabilities:

1. The User Profile Development Model characterizes the individual users, workload, and traffic requirements.
2. The Remote Locations Configurator determines equipment configurations and whether a location should be "on" or "off" the backbone network.
3. The Regional Network Configurator partitions the network into areas and designs a local network within each area using concentrators and/or multiplexers.
4. The Integrated Network Configurator designs the backbone network covering all on-net locations and WATS, TELEX, or a combination of other services to cover all off-backbone stations.
5. The Front-End Configurator configures the front-end system.

In addition to these five capabilities, SAI maintains a collection of data bases which contain information on telecommunications products (modems, front-ends, etc.), common carrier tariffs (e.g., AT&T, Western Union), and value-added tariffs (e.g., TELENET, TYMSHARE).

Summary

As indicated before, the company declined to provide specific information on the algorithms employed in their packages. The only material provided was promotional in nature. Consequently, no in-depth

analysis of SAI's products can be attempted. However, some comments may still be made:

1. The ability to design distributed networks appears to be lacking.

2. There are no reliability models.

3. Partitioning a network into separate areas within which centralized networks could be designed is a difficult problem, especially in the case of large networks. One would especially wish to examine the algorithms employed here.

UNIVERSITY OF MINNESOTA — VANS (Value-Added Network Simulator)
Professor M. Schneider, Computer Science Department

Professor Schneider is presently completing the implementation of the VANS model. His major purpose in building it is to produce a model capable of investigating the utilization of different network protocols.

Protocols, in the context of a computer network, are simply rules which govern the exchange of information between processes — establishing and maintaining connections, transferring messages after making connections, etc. They must cover the following areas:

- Opening, maintenance and closing of connections
- Flow control
- Process initiation
- Process interruption
- Development and maintenance
- Recovery from failure

A discussion of network protocols (for the interested reader) may be found in DAVI73, chapter 13.

Schneider, in "A New Methodology for Computer Network Simulation" [SCHN78], makes the point that present simulations of computer networks are what he terms parametric simulations. By this, he means that the user is required to input a list of parameters — line loadings, error rate, etc. — but a detailed study of different protocols and their effects on the network is not possible. In contrast to this "genre" of simulation, he defines

a structural simulation as being characterized by the three following points [SCHN78]:

(a) All subnetwork activity would be factored into a disjoint, nonoverlapping set of responsibilities called protocol areas.

(b) A user would initially "build" the structure of a subnetwork by providing the name of the process which will support each of the above protocol areas. These processes, which can come either from the user or a standard library, will be collected, compiled and linked together to form a simulation model of a specific network.

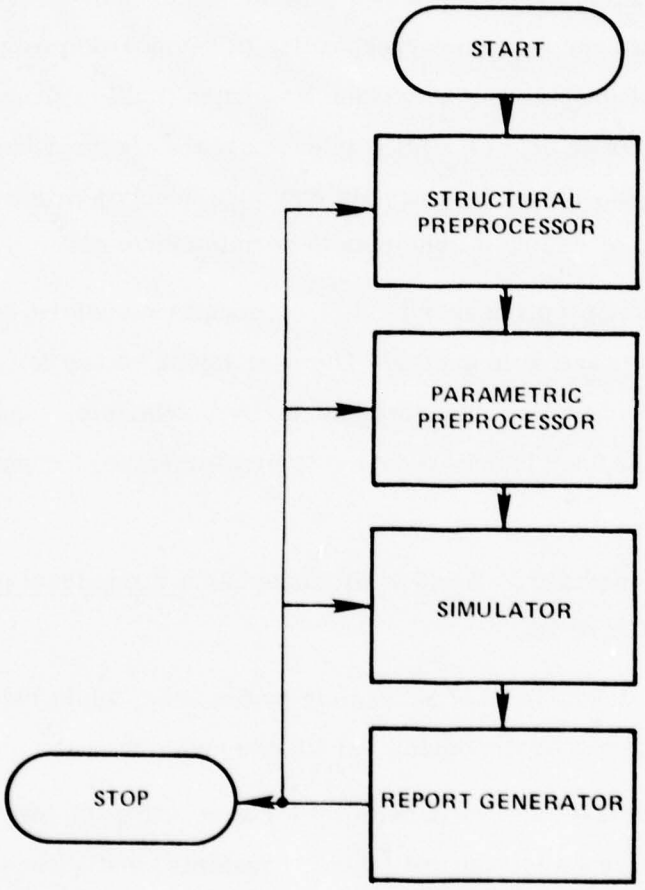
(c) A user would parametrize the network just constructed by providing those data values consistent with and needed by the model.

Such a simulation provides the user with the ability to "parametrize," as before, and also to modify the protocol structure of the network. As a result, the user could, for example, investigate protocols in the areas of dynamic routing, error detection and correction, flow control techniques, line allocation policies, etc. Schneider* has built such a model, and a brief description of its organization, together with an interesting list of experiments one may perform with it, is included below. Our discussion of his model follows his own description in SCHN78, and the figures which accompany this discussion appear in that paper.

Description of VANS Model

VANS is an interactive model, consisting of four phases, as depicted in figure 25. As is indicated in the diagram, the user has the option of reentering any of the preceding stages upon completing one run of the model.

*For a more complete discussion of Professor Schneider's model, the reader is urged to consult SCHN78 and SCHN76.



Based on SCHN78.

IA-53,085

Figure 25. Overall Organization of VANS System

The preprocessing phase of the model is broken into two parts, structural and parametric (which by now should be suggestive names). Both of these phases are written in PASCAL and are interactive.

The structural preprocessor essentially allows the user to build a library of protocols to support some 32 protocol areas. Each of these 32 areas has at least one module corresponding to "standard" protocols in that area. A default module has also been designated in all of these areas, obviating the need for an explicit choice in each area — the user has only to pick those that are of interest to him. He may also supply a protocol module written by himself, providing it conforms to certain standards.

The parametric preprocessor is also, as mentioned before, written in PASCAL, and is interactive in nature. The user inputs values for quantities such as the number of nodes, line speed, priority levels, etc. Again default values exist, and the user is only required to provide values for parameters which he selects.

The simulation phase of VANS is organized as a three-level process tree, as shown in figure 26.

The first two levels are not accessible by the user, while the third level consists of the protocol modules, which are under the user's control.

The executive level has responsibility for generating the level 2 node processes, as well as initializing all tables, constants, and queues. The six level 2 processes (five are shown in the diagram) are implemented by SIMULA processes. Following Schneider's description [SCHN78] these processes are:

- HOST — The HOST processes model the message generation/consumption operations of each HOST at the node. There is a

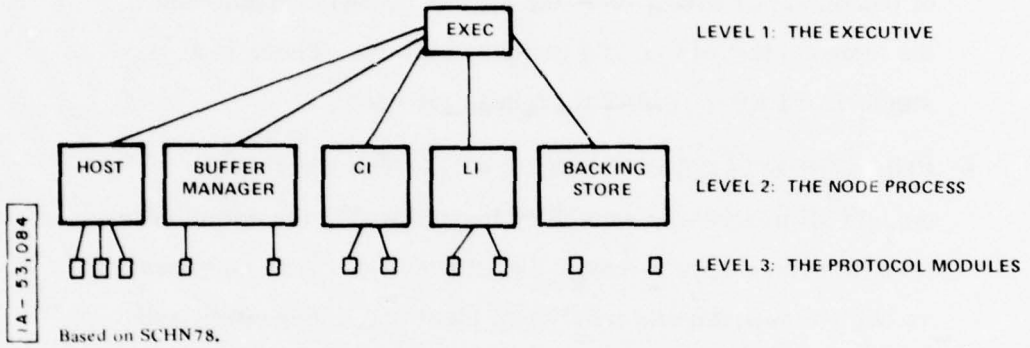


Figure 26. Process Tree Organization of VANS

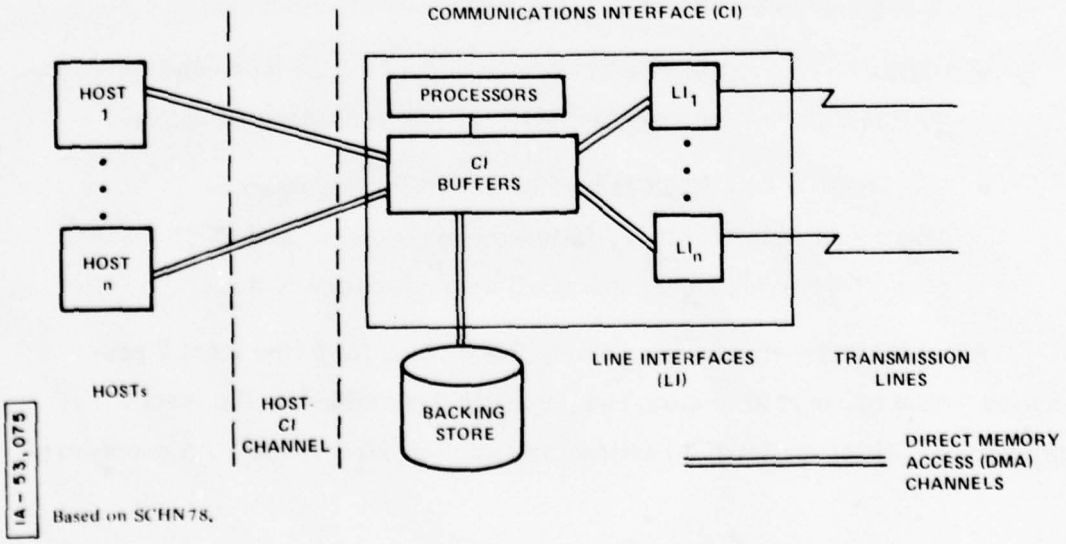


Figure 27. Diagram of a Network Node

separate process for each HOST type. As shown in the example in figure 27, there would be n separate HOST processes for that node.

- BUFFER-MANAGER — This process models the operation of the HOST-CI Direct Memory Access (DMA) channels and the management of the CI's primary bufters. There is a single BUFFER-MANAGER process for each CI.
- COMMUNICATIONS INTERFACE (CI) — The CI process models all processing capability located within the communications interface. Even though the CI processor may be viewed as being composed of a number of identical, independent subprocessors, there is only a single CI process per network node.
- LINE INTERFACE (LI) — Each LI process models the operation of a communication line interface with DMA capabilities. Referring to the example in figure 26, there would be m distinct LI processes for this node.
- STORE — The STORE process models the backing store and the DMA channel between the primary and secondary CI buffers.
- GREMLIN — GREMLIN is a randomly occurring process causing unplanned errors, failures, and outages, and in general wreaking havoc among all network components.

All of these processes are accomplished via calls to the level 3 processes whose order of activation and choice is determined by the level 2 processes. All of the level 3 routines are implemented as SIMULA procedures.

In an attempt to assure that a change in a level 3 procedure doesn't cause an unexpected change in another procedure, each protocol module must conform to a certain set of rules or capability descriptors. These rules address the following areas:

- Requirement of a standardized calling sequence, providing for a standard interface between all levels.
- Specification of the data structures that the protocol module is allowed to modify. Omission of a list implies that no data structures may be modified by a module supporting this protocol area.
- A list of the other level 3 modules it can activate.
- Actions a module can perform.

An example of a capability descriptor is presented in detail in Schneider's paper.

The last portion of VANS is a report generator. This is a standard post-processor, which prints out a list of statistics in the following five categories:

1. Message Statistics — total traffic by links, HOST-HOST, communications processor to communications processor.
2. Time/Efficiency Statistics — average message delays by priority classes, HOST-HOST, communications processor to communications processor.
3. Utilization Statistics of Various Resources — lines, processors, etc.
4. Queue Statistics
5. Error Statistics

Following this phase, the user has the right to reenter any preceding stage, giving him the opportunity to experiment further with both the physical network and the protocol structures.

Clearly, a number of fascinating experiments may be conducted with this model. Schneider's menu includes:

- A study of dynamic routing algorithms.
- A study of flow control and deadlock prevention algorithms.
- A study of forward error correction versus retransmission schemes under different conditions.
- A study of message versus packet switching under different conditions.
- Finally (and most ambitiously), a possible utilization of VANS as part of a topological design project for the construction of computer networks.

VANS is, in the opinion of the writer, an exciting model that shows the potential for breaking new ground in network design. As such, its further progress should certainly be watched.

DMW ASSOCIATES — NDMS, ANDMS

General

DMW Associates markets a series of models directed towards the design and management of centralized networks. The models are interactive in nature, and are available through a national time-sharing service (TYMSHARE). The company will also install them on the user's computer facilities.

Dr. Dixon Doll, the president of the company, declined (with one exception) to give any information on the algorithmic structure of his company's models. He did, however, explain that they were "battle-tested." Descriptive material of a promotional nature was also supplied.

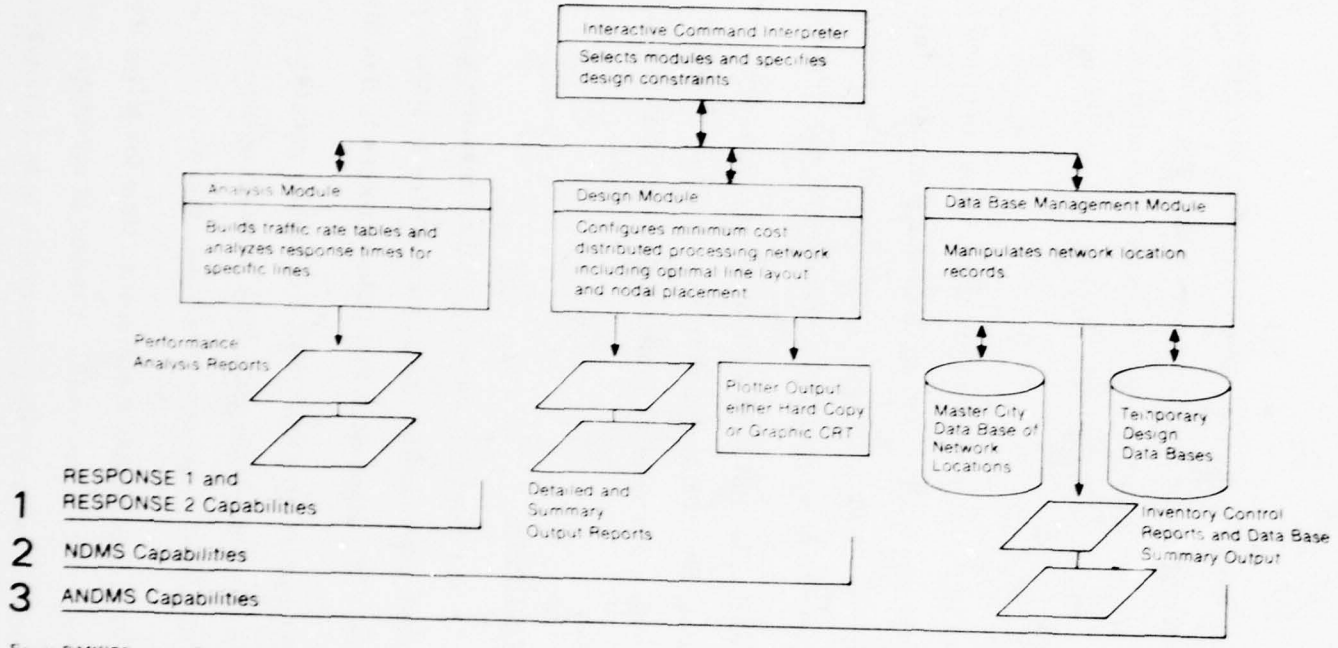
In view of this lack of information, no analysis, however tentative, is possible. Hence, we merely present a functional description of the models, and conclude with a few general comments on the product and its applicability to C^3 problems.

Functional Description

Figure 28 is a diagram, taken from the literature supplied by DMW [DMW78], which summarizes the major modules in DMW's package as well as their functions. The discussion that follows refers to this diagram.

The module termed RESPONSE 1 basically generates a line-loading table, given a response-time goal, thus permitting the user to compare different line disciplines, message priorities, etc.

RESPONSE 2 computes the response time for a line that has already been configured; that is, its traffic, number of terminals, etc. has already been determined. As such, its primary use is to estimate response time once the network has been designed by the DESIGN module of NDMS.



From DMW78, p. 1. Reprinted by permission.

Figure 28. DMW Network Design and Analysis Models

NDMS (Network Design and Management System) solves the concentrator-assignment and line-layout problem. Dr. Doll mentioned that the Esau-Williams algorithm is at the heart of this module. A number of options for the use of this module are available. For example, the user can perform his own concentrator placement and rely on a subset of NDMS to perform the line-layout activity. He may also choose to develop his own line-loading tables instead of relying on RESPONSE 1. ANDMS (Advanced Network Design and Management System) is, as is evident from figure 28, an extension of NDMS. In addition to NDMS, ANDMS contains a data-base management subsystem, DBMS, that can be used to keep track of and control user locations and facilities (modems, terminals, etc.). This information may then be passed on to the DESIGN module as well as to the ANALYSIS module. Temporary data bases may also be created, merged, and stored.

The analysis module (aptly named ANALYSIS) is used for determining response time once the DESIGN module has completed its activities. It is not clear how this submodule differs from RESPONSE 2.

In addition to the above network design products, DMW provides other management tools for communications-related expenses.

TELECOST provides reports on telephone utilization and costs. For example, one may obtain detailed summaries of long-distance phone calls, trunk group usage (WATS, FX, Direct Dial, etc.), lists of frequently called numbers, etc.

WATSTOLL provides information on the costs and utilization of WATS lines, and also determines an optimal mix of tariffs and facilities.

Online Tariff-Guide is an interactive program for determining the price schedule for both point-to-point and multipoint circuits. It contains an extensive directory of common-carrier tariffs.

Summary

As no algorithmic information was made available (with the exception of the DESIGN module), not much can be said about DMW's package. It evidently suffers, however, from the lack of a distributed network design capability and reliability capabilities.

Given Dr. Doll's association with IBM's Systems Research Institute, and his use of the Esau-Williams algorithm (which constitutes an important building block of CNDP), one may reasonably speculate that the remainder of his package is also tailored along the lines of CNDP.

NETWORK ANALYSIS CORPORATION (NAC) — GRINDER, MIND

General

NAC has developed tools for the design of centralized systems (MIND), as well as distributed systems (GRINDER) — it is the only vendor that supplies a package capable of designing distributed networks. Reliability models are included with both GRINDER and MIND. Again, NAC is alone among the vendors in offering reliability models, a feature clearly of great importance in military circumstances. Both of these tools consist of a series of modules, which are written in FORTRAN, and are interactive in nature. The modules can also be utilized in a batch mode. They are available through a national time-sharing service or can be installed on the user's computer.

Functional Description

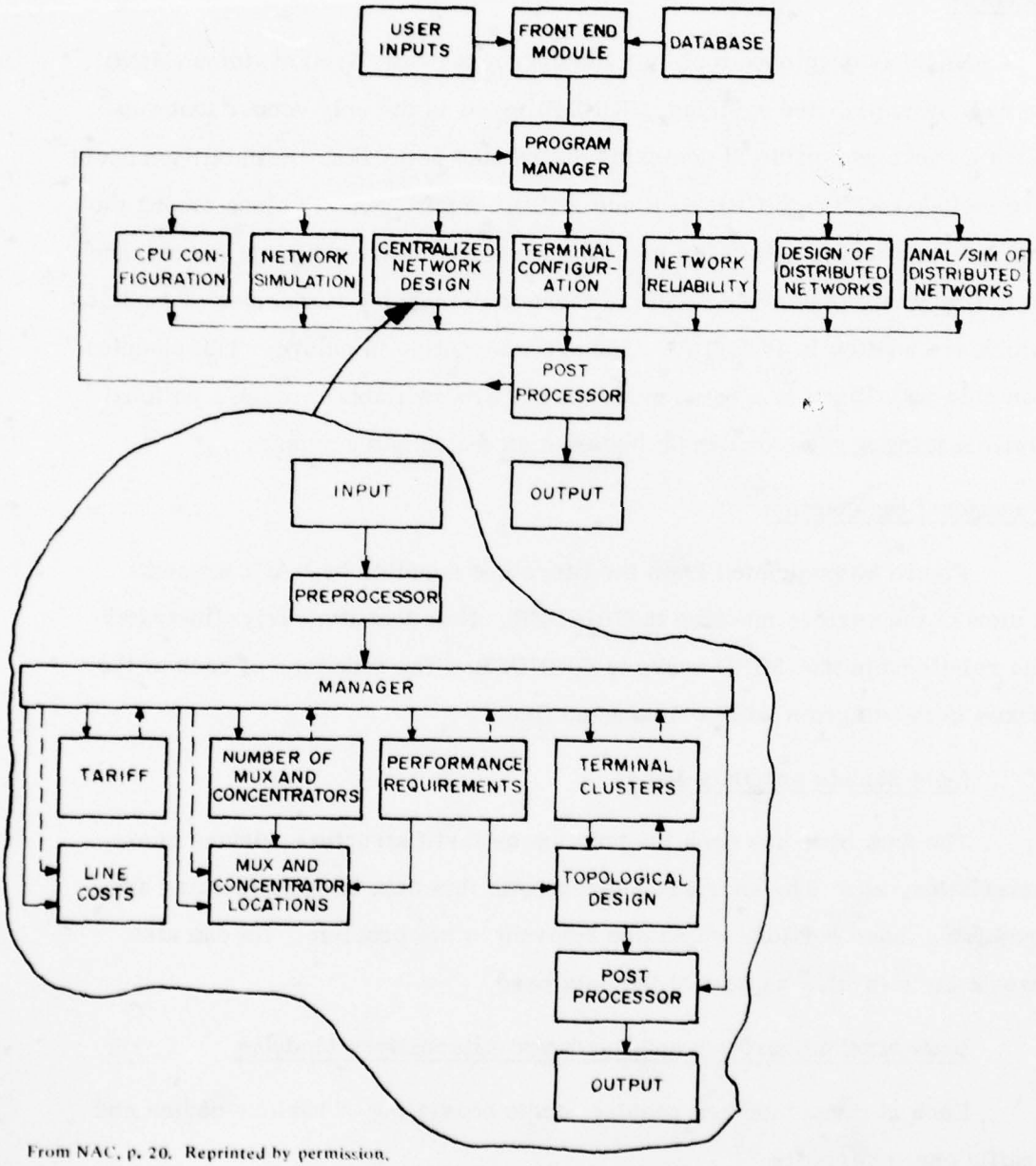
Figure 29 (reprinted from the literature supplied by NAC) presents a view of the various modules in GRINDER. Note that it clearly illustrates the relationship that MIND bears to GRINDER. The functions of each of the boxes in the diagram will be discussed first.

Input Module and Data Base

The data base has such information as tariff structure, device characteristics, etc. The user can interact with this data base by fetching and modifying those portions which are relevant to his problem. He can also create his own files as part of the data base.

Network Configuration and Hardware Alternatives Modules

Each of these modules require inputs consisting of various device and traffic characteristics.



From NAC, p. 20. Reprinted by permission.

Figure 29. NAC's Network Design Program

CPU Configuration. This module evaluates the cost-effectiveness of a particular central site configuration (i.e., a combination of central processor(s) and peripheral devices). Its outputs include system bottlenecks, throughput and cost for the various device characteristics, as well as for different distributions of CPU processing time, and different message arrival rates.

Terminal Configuration. This module evaluates the cost-effectiveness and throughput capabilities of different terminal configurations. One topic of interest here is the use of intelligent terminals versus slave terminals employed with line controllers. The number of terminals for a given throughput and the cost of that throughput are the basic outputs of this module.

Number and Location of Multiplexors and Concentrators. This module portrays the effect of each of these devices on the network's response time as well as its reliability.

Network Simulation Module

This module, along with its "distributed" companion, performs the response time throughput analysis for the network. It is a hybrid model in that it is a simulation, with analytic models employed at the device level.

The parametric input to this module includes message length distributions, interval time distributions, number of terminals per line, polling and addressing sequence lengths, device turnaround times, etc. The output of the module can be presented in the form of an overall response time, or waiting times in various devices. Response time as a function of throughput is also an option. This module is used to produce line-loading tables as an input for the topological design modules. Various other options for input and output are also available.

Reliability Module

The function of this module, as clearly implied by the name, is to calculate various measures of network reliability.

The module requires as inputs a given topological design as well as device failure rates (in the form of mean time to failure, mean time to recovery, probability of failure, etc.).

The output can be presented in the form of the fraction of node pairs that are not able to communicate.

Programs for calculating the reliability of "standard" network configurations, such as trees, loops, series parallel networks, etc., have been written. Since networks are often a combination of the above topologies (as well as a backbone), the above programs may often be combined.

Topological Design Modules

There are two design modules in the package. One corresponds to a centralized network, while the other corresponds to a distributed network. The centralized design module addresses the problems of clustering the terminals, concentrator/multiplexor placement, line-layout and capacity allocation. The distributed design module, in addition to solving the same problems as the centralized design module, addresses the routing issue as well as the backbone design problem for large networks.

Program Manager

This module has two functions. It allows information to be passed back and forth in between modules. For example, the results of the response time module can be passed on to the design module, and vice versa, thus allowing an interaction to occur that can successively produce more effective

designs. The Program Manager module also produces a variety of designs under different assumptions. Thus for a variety of different traffic levels, one can obtain corresponding designs and the costs for different response-time/throughput requirements.

Algorithmic Structure

We divide our description of NAC's model into three parts. The first part deals with three modules common to the design of both centralized and distributed networks — the reliability and CPU configuration modules. The second part deals with the design of centralized networks; and the last section presents the design of distributed networks, including a discussion of the backbone design problem for large distributed networks.

Models Common to Both Centralized and Distributed Network Design

CPU Configuration Module. This module evaluates the utility of various hardware configurations. It is applicable to both central site configurations and concentrators. The module essentially determines the following characteristics for different "conditions" — capacity, response time, throughput, and the location of processing bottlenecks. The conditions are the following design variables:

- Hardware configurations
- Software design (i.e., processing cycles of transactions)
- Traffic characteristics, both demand and service

The model employed is based upon a paper by Chou and McGregor [CHOU75], entitled "A Unified Simulation Model for Communication Processors." In this paper, a methodology for constructing such a simulation is developed by employing a queueing model of a communications

processor, and then basing a discrete event-stepped simulation on this model. This hybrid approach combines the advantages (and, one hopes, eliminates the disadvantages) of the pure analytic and the pure simulation approach to the problem. The advantage of simulating a queueing model as opposed to the actual system is the gain in speed of the simulation without an attendant loss of accuracy. This is unfortunately too often the case when one relies strictly upon queueing models.

The model itself, shown in figure 30, is a collection of interconnected queueing structures. There are four servers (each possessing its own queue) — the CPU, the peripheral devices, the I/O channels associated with the peripheral devices, and the communications lines.

The CPU is modeled as a single-server queue. Inputs may come from the communication lines, the peripheral devices, or the CPU proper. Outputs are directed to these same devices. The service time for the CPU is considered to be the processing time.

The peripheral devices are modeled as a collection of single-server queues. In case of a disk, transactions first queue-up for access to the desired disk, and are then serviced provided that no other transaction is making use of the disk. The service time is the disk-seeking time. After completion of this activity, a queue is formed "in front of" the I/O channel of the disc controller.

The I/O channel of a device controller is portrayed as being shared by a number of devices, while the controller governs the activity of one device at a time. The service time is the read/write time.

The communication lines are modeled as a collection of single-server queues.

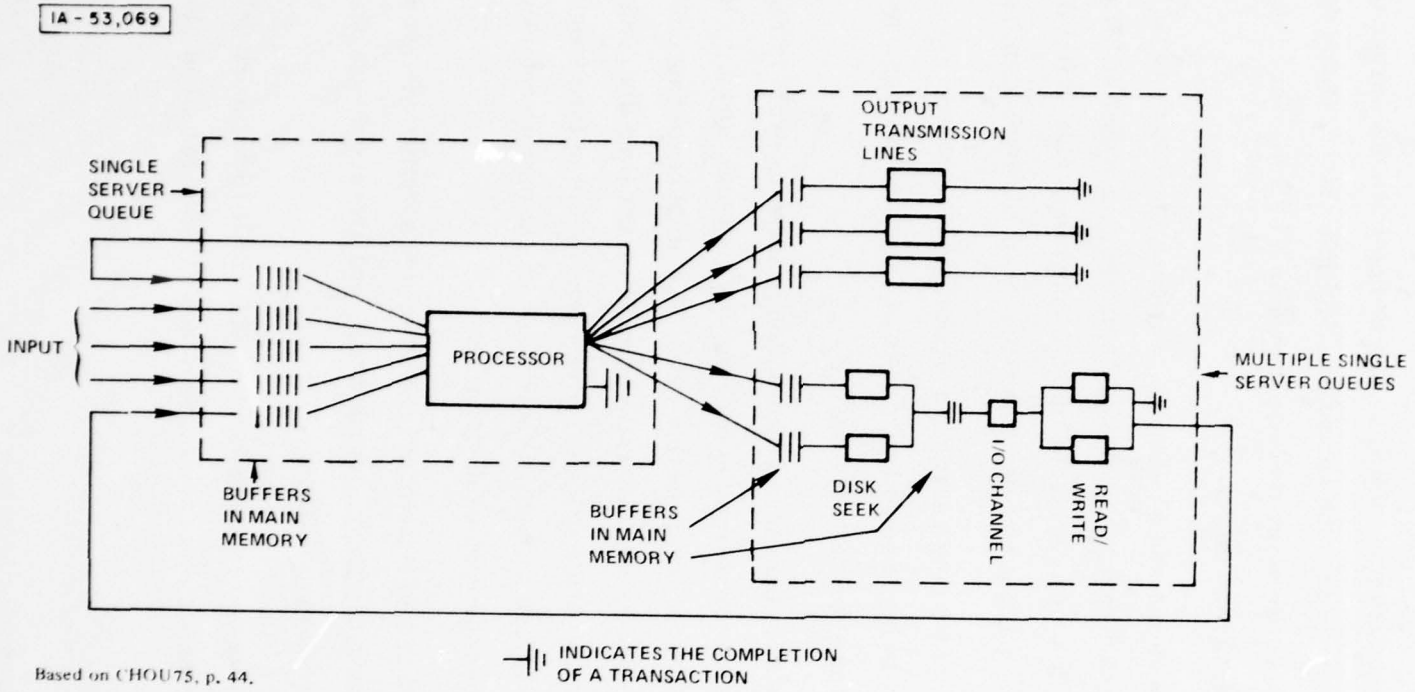


Figure 30. Model of a Communications Processor

Each transaction is viewed as proceeding from one server to another as it progresses through the computer. Multiple entries to any device, as well as simultaneous service of one transaction by a number of devices, are included as part of the model.

The simulation was written, as previously indicated, to be sensitive to changes in hardware, the transaction processing cycle, and traffic characteristics. Hence, a hierarchy of levels corresponding to the above three system characteristics was devised as the basis for implementing the program.

The hardware configuration level occupies the highest rank of the hierarchy. It is implemented as a table of hardware devices, and associated transaction times for each device.

The software design, or transaction processing cycle is represented by a state vector corresponding to each facility (server), and a cycle vector corresponding to each transaction type. The state vector associated with each device characterizes its state by an interrupt time distribution, a service time distribution, and a priority. The cycle vector in turn specifies the sequence of servers as well as their states. These two vectors provide the linkage between the various devices.

The demand/service characteristics are specified via average message lengths and message length distributions, and average message arrival rates and inter-arrival distributions.

The interested reader should consult CHOU75 for more details; another paper of interest on this specific approach [MCGR75a] might also be consulted.

AD-A078 022 MITRE CORP BEDFORD MA F/G 17/2
COMPUTERIZED MODELS FOR DESIGN AND ANALYSIS OF COMPUTER-COMMUNI--ETC(U)
MAY 79 C TROPPER F19628-78-C-0001
UNCLASSIFIED MTR-3664 ESD-TR-79-128 NL

2 OF 2
AD A078022

END
DATE FILMED
1-80
DDC

Reliability Module. In this discussion, we focus on an analysis model for network reliability.* Our discussion is based upon papers by Van Slyke and Frank [VANS72] and Frank and Chou [FRAN72].

The measure of reliability that is employed in this model is the average fraction of node pairs that cannot communicate because of node and link failures. With this definition in mind, we formulate the analysis problem in terms of the probability of the network being disconnected. Specifically, given a network with N nodes and A arcs, as well as a probability p of failure for each of these elements, estimate the probability h(p) of the network being disconnected as a result of component failures for P values of the probability p. One may also wish to estimate the expected number of node pairs, n(p), not able to communicate for P values of p.

Two approaches are presented, the functional-simulation approach, and the Moore-Shannon approach. Both approaches benefit from an algorithm that is used to calculate the number of components in a graph with N nodes and A arcs. This algorithm operates by successively adding one branch at a time to the network, and then calculating the number of components in the resultant graph.

Employing the notation A_k for the graph formed upon the addition of link a_k to the graph a_{k-1}, we now present the algorithm:

Step 0 Initialization

Let $A_0 = 0$, and assign a separate label to each node.

Set k = 0 (k is used to keep track of the number of nodes added).

---

*Network reliability (survivability) is a major subject in its own right, with many results scattered throughout the literature. A comprehensive survey of the results is contained in a paper by Frank and Frisch [FRAN70a].

Step 1 Add link a_k to A_k. If $a_k = (n_i, n_j)$, then examine the labels on n_i and n_j. If they are different, go to step 2. Otherwise, set $k = k + 1$, and repeat step 1.

Step 2 Change all of the node labels that are the same as the label of n_i to the label of n_j. Set $k = k + 1$, and check to see if $A_{k+1} = A$ (i.e., if all the arcs have been added). If so, STOP. Otherwise, go to step 1.

As one proceeds through the algorithm, one may keep statistics as to the number of communicating node pairs, the number of nodes in each component, etc.

We now proceed to a discussion of the two simulation approaches employed in NAC's reliability models.

1. The Functional Simulation Method. This suggestive name is employed because in this method, the expected number of node pairs communicating with one another are calculated as a function of the probability of link failure. Based upon work by Hammersly and Handscomb [HAMM64], the method is as follows.

For each link, a random number between 0 and 1 is generated, and they are then listed in decreasing order. As is usual in Monte Carlo simulations, if p is larger than a random number, then the link associated with that random number is taken out of the network, while if p is smaller than the number, the link is left in, and the resulting network is analyzed for its connectivity. Hence, for the given sequence of random numbers $r_1 > r_2 > \ldots > r_{na}$, if $p > r_1$ all the links are removed. If $r_1 > p > r_2$, then only the first link remains, and in general, if $r_i > p > r_{i+1}$, then the network having links $a_1, \ldots, a_i$ is examined for connectivity.

By using the connectivity algorithm just described, however, the network's connectivity can be evaluated for all P values of p in one pass, thus saving a considerable amount of computation time. This is seen as follows: First, order the collection of probabilities and the random numbers generated in one sequence, e.g., $r_1 < p_1 < r_2 < \ldots < p_a < r_a$. Then, adding one link at a time as permitted by the above algorithm, analyze the resulting networks for connectivity, keeping track of the appropriate statistics (e.g., node pairs which can communicate, etc.), if so desired.

2. Moore-Shannon Approach. Based on a paper by Moore and Shannon [MOOR56], this method estimates the probability of the network being disconnected, h(p), via the following formula:

$$h(p) = \sum_{k=0}^{N+A} c(k)\, p^{N+A-k}\, q^{k}$$

where $q = 1 - p$ and c(k) is the number of networks having k elements (i.e., nodes and arcs) which are disconnected.

A similar formula is employed to estimate the number of node-pairs that are not able to communicate:

$$n(p) = \sum_{k=0}^{N+A} D(k)\, p^{N+A-k}\, q^{k}$$

where D(k) is the number of node pairs unable to communicate, given that there are k arcs in the network.

In evaluating the formula for h(p), a simple combinatorial result is brought to bear. If c is the minimum number of arcs required to disconnect the network, then there are only A - c + 2 non-trivial terms in the expansion. This is so because:

1. If the network has n nodes, it takes at least n - 1 arcs to connect them. Hence $c(k) = \binom{NA}{k}$ when k = 0, 1, ... , n - 2.
2. If c is the minimum number of links that must be deleted to disconnect the network (i.e., the minimum cut-set), then c(NA - k) = 0 for k = 0, 1, ... , c - 1.

In addition to this, if $\binom{NA}{k}$ is sufficiently small, then c(k) can be calculated by enumeration — if not, they may be estimated by simulation. In the event that they are estimated via simulation, the theory of stratified random sampling is applied. A discussion of this technique may be found in FISZ63.

The algorithm for computing the number of components of a graph can also be applied here.

Centralized Network Design

Two of the fundamental questions in the design of centralized networks have been discussed in section 2 — the location of concentrators (multiplexors) and the line-layout problem. NAC's approach to the line-layout problem — the Unified Heuristic Algorithm, due to Kershenbaun and Chou — has been discussed in that section. Hence, the main topic of this section falls under the following heading.

Network Partitioning/Concentrator Location Module

This algorithm is referred to as the COM (center-of-mass) algorithm. It functions as a local-access designer by first partitioning the users in the network, then assigning them to local access facilities (i.e., concentrators or multiplexors), and finally tying the users via their local access facility to a central computer.

Employing NAC's own terminology, a resource connection point will be called a RESCOP, while a local access facility will be referred to as a TACOM (the TIP and ANTS of ARPANET fame, concentrator or multiplexor). The choice of this terminology reflects the generic nature of the access location problem.

The access location problem may be formulated as follows.

Given:

- A set of nodes
- A particular node that serves as a RESCOP
- Constraint on the number of nodes that may share a line
- Constraint on the number of nodes that a TACOM may serve
- Cost of connecting nodes
- Cost of TACOM

Find: A low-cost feasible design employing TACOMS

Our description of the algorithm follows the one given by NAC. We first present a brief summary of the algorithm, and then expand the discussion to include some of the details. The interested reader is referred to MCGR75b.

The algorithm may be outlined in four steps, as follows.

1. Reduce the problem to a point-to-point problem by replacing clusters of nodes by single center-of-mass (COM) nodes. (Hence the title of the algorithm.)

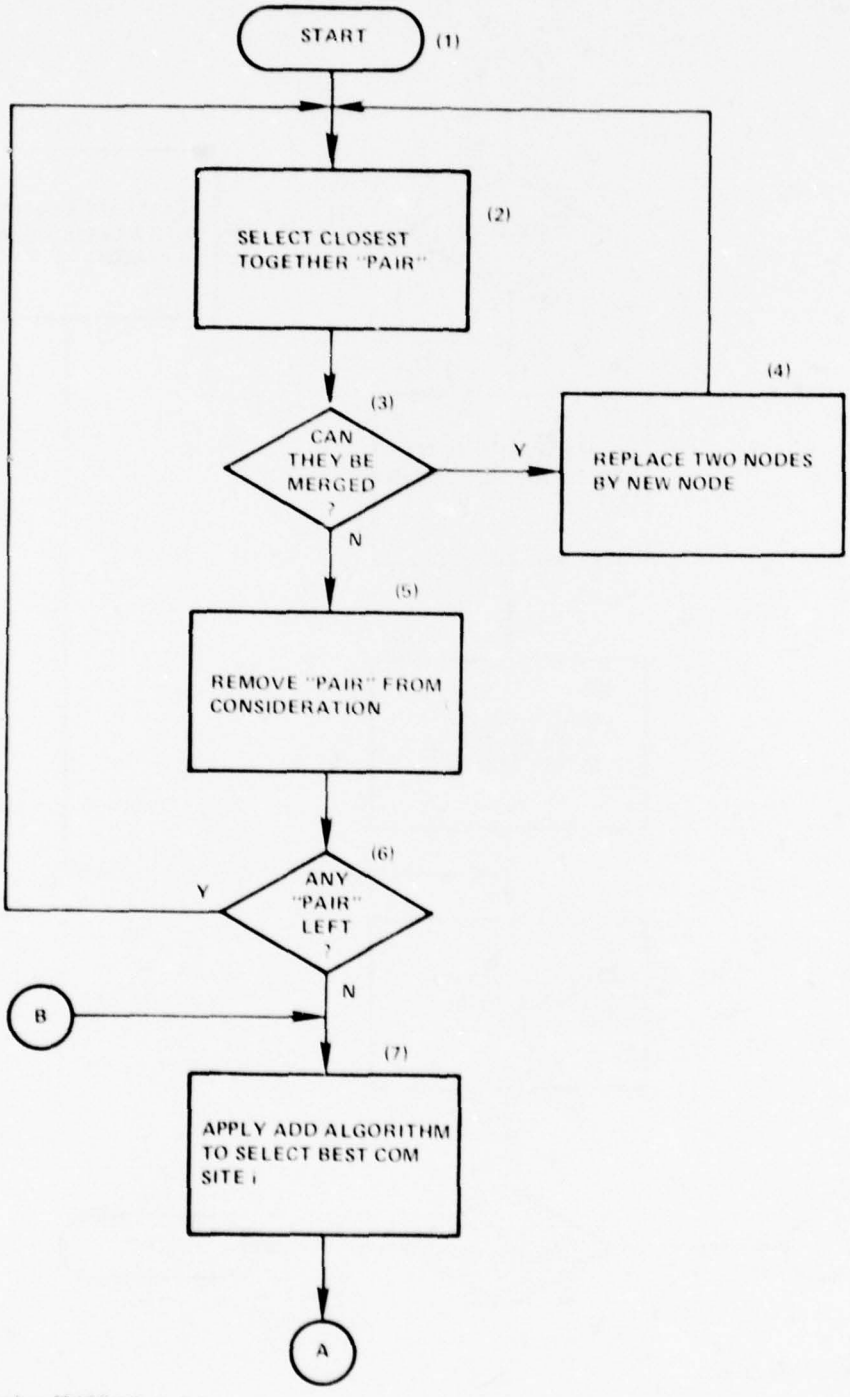
2. Apply an add algorithm to the resulting collection of COM sites, resulting in a placement of TACOMS at several COM sites.

3. Locate the chosen TACOMS at real nodes by choosing the original nodes closest to the designated COM nodes.

4. Apply a line-layout algorithm to each partition, utilizing the TACOM as the central node.

A simplified flow chart taken from MCGR75b is presented in figure 31, and a series of diagrams depicting the outcomes of the four steps is presented in figures 32 through 35. In the following expanded discussions of each of the four steps, numerals in parentheses refer to the flow chart (figure 31).

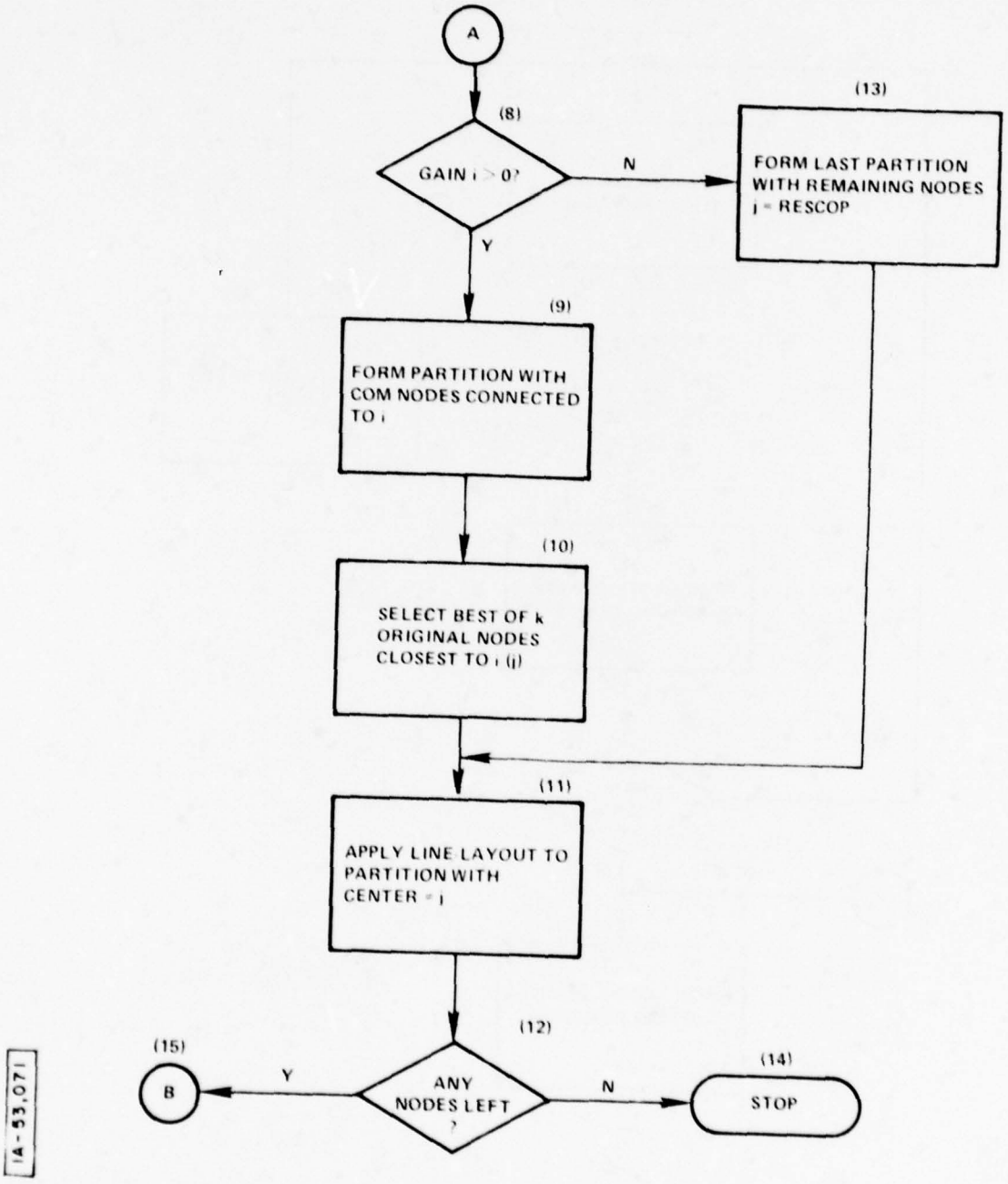
Step 1: Clustering. Clusters of nodes are formed by "snowballing" nodes together. The two nodes closest together are selected (2), and line-loading constraints are checked (3). If they were not violated, the two nodes are replaced by a new node at the center-of-mass of the original two nodes (4). The nodes start with a weight of one. The weight then becomes the number of nodes in the cluster. Nodes that cannot be merged are removed from consideration. The "snowballing" continues until all nodes belong to their own cluster, as portrayed in figure 32.



IA-53,070

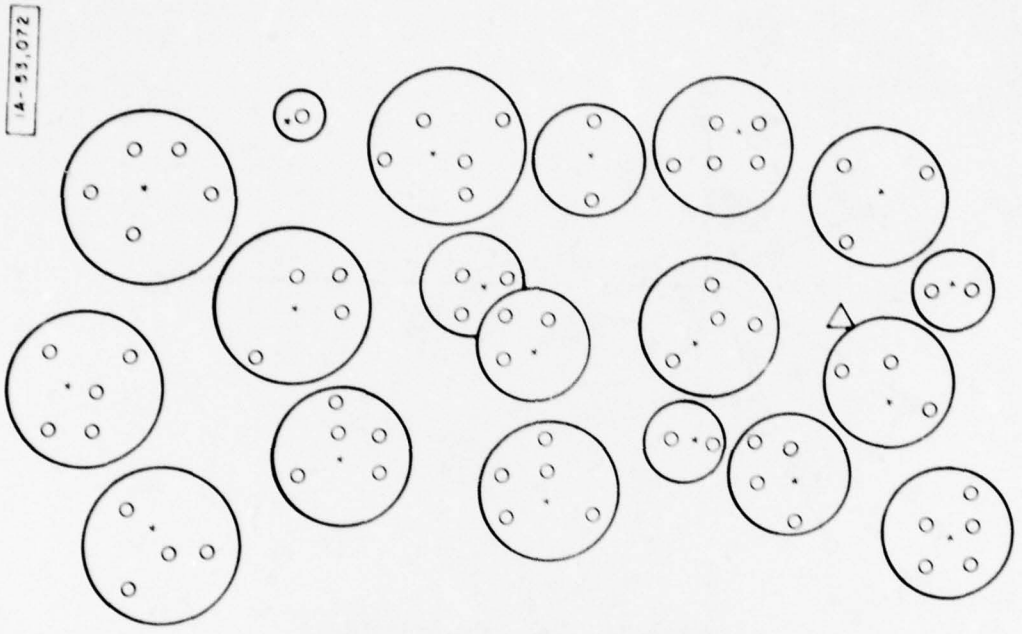
Based on MCGR 75b, p. 4-3

Figure 31. Flow Chart of COM Algorithm



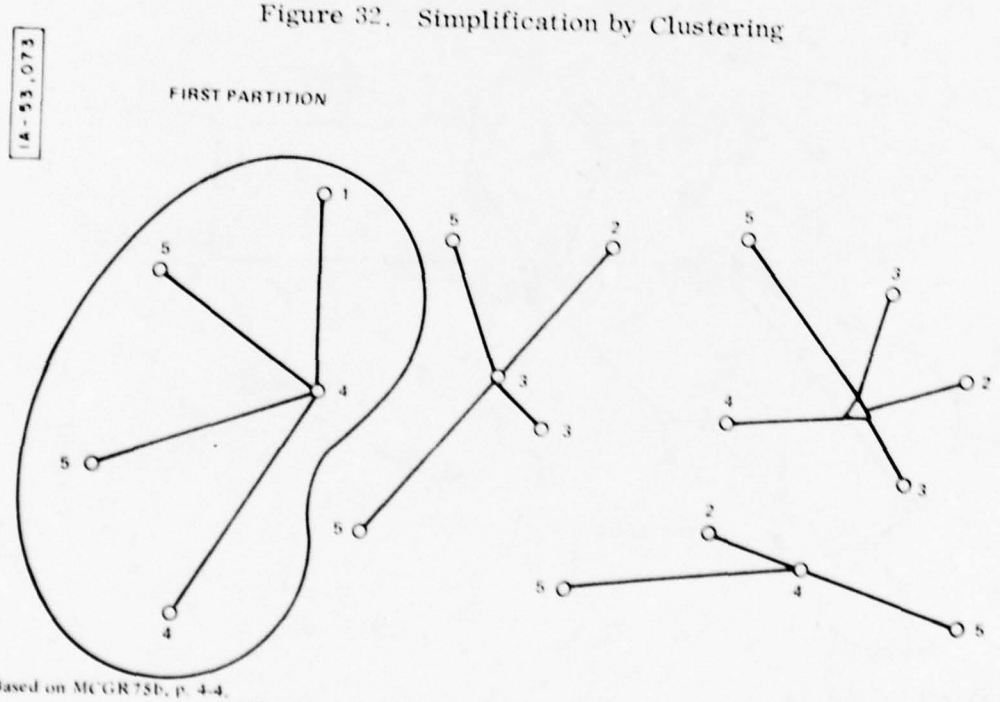
Based on McGR75b, p. 4-3.

Figure 31 (concluded). Flow Chart of COM Algorithm



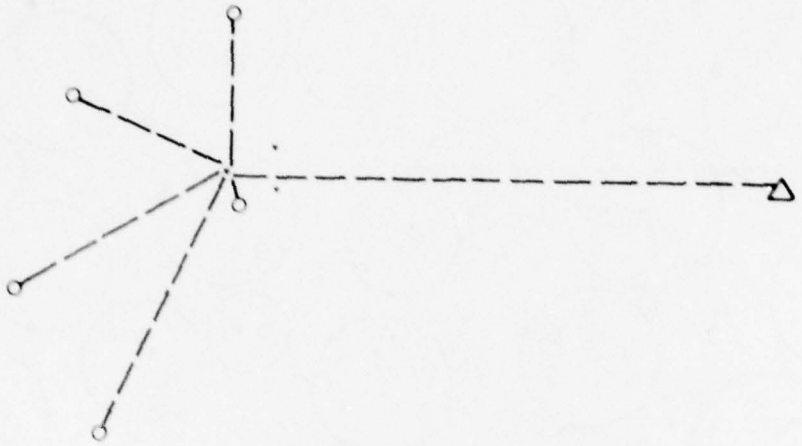
Based on MCGR75b, p. 4-4.

Figure 32. Simplification by Clustering



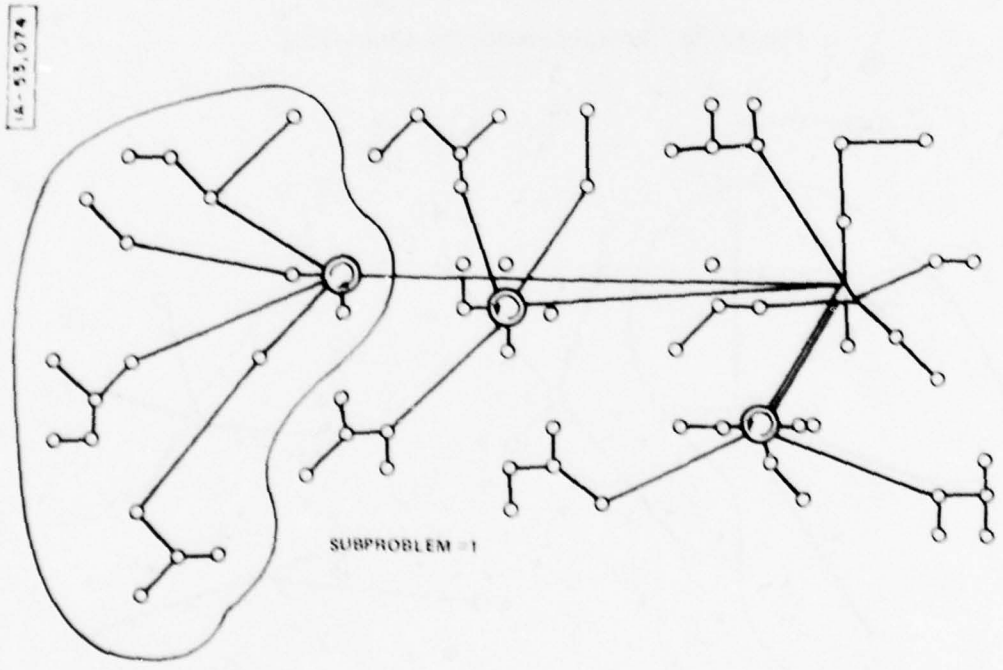
Based on MCGR75b, p. 4-4.

Figure 33. Partition by Add Algorithm



Based on NAC75.

Figure 34. Local Optimization



Based on MCGR75b, p. 4-5.

Figure 35. Line Layout

Step 2: Partitioning. An add algorithm is applied to the COM sites in order to locate TACOMs (7). After choosing each successive TACOM site, an heuristic algorithm incorporating the weight of the nodes is applied, resulting in a possibly new choice for the TACOM. If the choice is cost-effective (8), the TACOM is kept, and the process continues (9). If not, all the remaining COM nodes are directly connected to the central site (13). The output of this step is shown in figure 33.

Step 3: Local Optimization. Step 2 results in the selection of a particular COM site as a TACOM location. Because this site can easily be an idealized site — i.e., not an actual node — it is necessary to locate the TACOM at an actual node. To accomplish this, several of the nodes closest to the ideal TACOM site are evaluated (10), and the most cost-effective is chosen. This step is depicted in figure 34.

Step 4: Line Layout. At this stage a line-layout algorithm (e.g., unified algorithm) is applied to the given partition (11). A check is made to see if any nodes remain (12). If any remain, the add algorithm of step 2 is applied again. If not, the design is finished. The result of the fourth, and final step is depicted in figure 35.

Distributed Network Design

As mentioned in section 2 of this paper, NAC's model for the design of distributed networks is the cut-saturation algorithm. An outline of the algorithm was presented in that section, leaving us with only two remaining tasks. These are a discussion of the flow assignment (or routing) algorithm which is embedded in the cut-saturation algorithm, and a discussion of the backbone design algorithm at the heart of large network designs.

An efficient implementation of the cut-saturation algorithm is dependent upon a fast-routing algorithm. The algorithm developed is essentially an outgrowth of the Flow-Deviation algorithm which was discussed in section 2. As was noted, the developers of the Extremal Flows algorithm have essentially employed a gradient-projection method to speed up its rate of convergence, and thus make it more amenable for its inclusion in the cut-saturation algorithm.

Flow Assignment. The algorithm for flow assignment is based on a paper by Cantor and Gerla [CANT74], and is called, appropriately enough, the Extremal Flows (EF) algorithm.

We should remind the reader at this juncture of some remarks made in section 2 concerning the nature of *routing algorithms employed during the* design phase of a network. Of necessity, these algorithms provide a fixed routing policy (i.e., one that does not vary with time). Because of the interdependence of routing with other design considerations, we are in need of a routing policy, and so we settle for one that will provide us with a minimum average time delay design.

Cantor and Gerla approach the problem via mathematical programming. The solution is exact in that a global optimum may be achieved.

The EF algorithm is essentially an improved version of the Flow-Deviation algorithm discussed in section 2. The improvement is to be found in the use of a gradient-projection method to speed up the rate of convergence of the algorithm. This is significant, because an efficient implementation of the cut-saturation method is dependent upon a fast routing algorithm.

The problem, as formulated by Cantor and Gerla, is as follows:

Given: A directed network of NN nodes and NA arcs, and nn x nn requirements matrix $R = [r_{ij}]$ whose entries consist of flows between nodes i and j,

Minimize* over f (network flow vector)

$$T = \frac{1}{\gamma} \sum_{i=1}^{NA} f_i \Big/ \left(C_i - f_i \right)$$

subject to the following constraints:

- f is a multicommodity flow satisfying the requirements matrix R
- $f \leq C$, where the equation $C = (C_1, \ldots, C_{NA})$ represents the capacity constraints on the flows.

Cantor and Gerla first note that the expression for T can be reformulated in terms of shortest route flows. They do this by noting that the extreme points of the set of feasible flows can be shown to be shortest route flows. This conclusion is derived from the following two facts: (1) f is a multicommodity flow satisfying a requirements matrix, and (2) the collection of

---

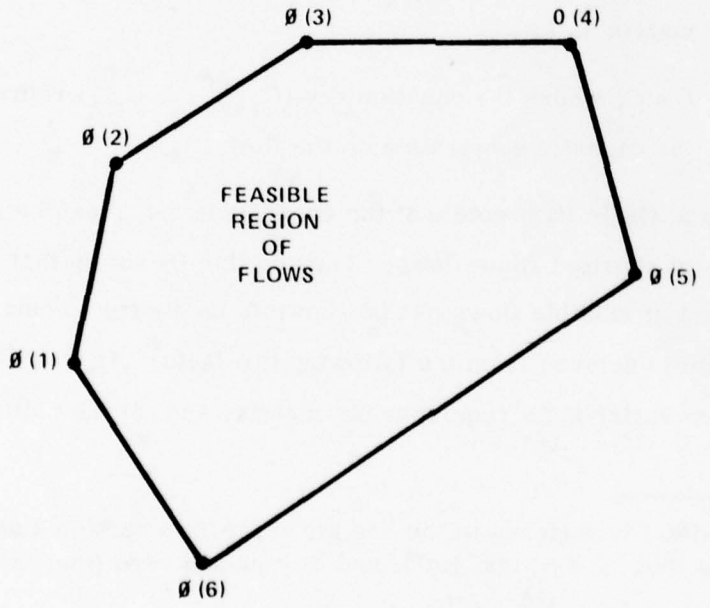
*This expression corresponds to the one given for T in section 2 as follows. First assume that $\mu' = \mu$, and set k and P equal to zero (that is, ignore them). We obtain $T = \sum_{i=1}^{NA} \frac{\lambda_i}{\gamma} \left(\frac{1}{\mu C_i - \lambda_i} \right)$, then divide top and bottom by μ, obtaining the above expression.

multicommodity flows satisfying the requirements matrix R is a closed and bounded convex polyhedron.

Cantor and Gerla refer to shortest route flows as extremal flows and represent f as a convex combination of a collection of extremal flows, ϕ^i, i = 1, ... , r,

$$f = \sum_{i=1}^{r} \lambda_i \phi^{(i)}, \quad \text{where} \quad \sum_{i=1}^{r} \lambda_i = 1, \ \lambda_i \geq 0, \ i = 1, \ldots, r \ .$$

Figure 36 is a geometrical representation of this situation.



Based on CANT74, p. 1064.

Figure 36. Feasible Region of Flows and Extremal Points

The authors then point out that at most NA + 1 extremal flows are necessary (i.e., $r \leq NA + 1$, where NA is the number of arcs). This turns out to have significant implications for the algorithm proper, because a pivoting operation may be employed to reduce the extraneous flows.

In order to determine conditions for an optimal feasible flow, Cantor and Gerla invoke the Kuhn-Tucker conditions for optimality. Before stating them, we offer some preliminary remarks.

Let $$l_i = \frac{\partial T}{\partial f_i} = \frac{c_i}{\left(c_i - f_i\right)^2}$$

$\frac{\partial T}{\partial f_i}$ can be thought of as the increase in time-delay, given an infinitesimal increase in the flow f_i on channel i. Having thus defined l_i, we are in a position to define the quantities μ_k which are instrumental in stating the Kuhn-Tucker theorem.

$$\mu_k = \frac{\partial T}{\partial \lambda_k} = \sum_{c=1}^{NA} l_i \phi_i^{(k)}$$

One may interpret μ_k as the increment in T, given an infinitesimal increment in λ_k, or alternatively, as the cost of the extremal flow ϕ^k, given a cost of l_i per infinitesimal increase in the flow on arc i.

With these definitions out of the way, the Kuhn-Tucker conditions may be written as:

$$\mu_k = \begin{cases} \mu_0 & \text{if } \lambda_k > 0 \\ > \mu_0 & \lambda_k = 0 \end{cases} \qquad \text{for each } k \ .$$

The interpretation of these conditions is that essentially all non-zero extremal flows must have the same cost μ_0 (i.e., the same incremental delay), and that any extremal flow with a cost greater than μ_0 must be zero. After some further manipulation, the authors point out that the conditions are equivalent to the statement that f is optimal if and only if there exists no flow with a smaller delay increment. Since one can verify this with a shortest-route calculation, a valuable bit of information has been obtained.

We now present the Extremal Flows algorithm, which uses a shortest-distance algorithm to generate extremal flows, and then employs a gradient-projection algorithm to minimize the time delay T over the set of extremal flows.

Step 0 Initialization

Let $b = NA + 1$, the number of current extremal flows

$\Phi = \left(\phi^{1,0}, \ldots, \phi^{b,0}\right)$, the initial set of extremal flows

$\lambda^0 = \left(\lambda_1^0, \ldots, \lambda_b^0\right)$, the initial basic solution

$f^0 = \sum_{k=1}^{b} \phi^{(k),0} \cdot \lambda^0{}_k$, the initial feasible flow

$n = 1$

Step 1 Gradient Projection

The gradient,* defined by $\nabla_\lambda T = (\mu_1, \ldots, \mu_b)$, is projected on the hyper-plane $\sum \lambda_i - 1 = 0$. By setting the components of $\nabla_\lambda T$ which point in unfeasible directions equal to zero, a feasible downhill direction $(-\nabla_\lambda T')$ is obtained. The following minimization is performed at each iteration:

$$\min_{\alpha > 0} T \left\{ \sum_{i=1}^{b} \left[\lambda_i + \alpha \left(-\nabla_\lambda T' \right)_i \phi^{(i)} \right] \right\}$$

where

$$\alpha \leq \tau_i \Big/ \left(\nabla_\lambda T' \right)_i \quad \text{if} \quad \left(\nabla_\lambda T' \right)_i > 0, \; i = 1, \ldots, b$$

The two steps are repeated until $||\nabla_\lambda T'|| = 0$, at which point the Kuhn-Tucker optimality conditions are satisfied.

Step 2 Pivot Step

The purpose of this step is to eliminate extraneous basic solutions in preparation for the next step.

If $b = NA + 1$, let $\underline{\Phi}^n \leftarrow \underline{\Phi}^{n-1}$,

and go to step 3. If $b = NA + 2$, eliminate one of the extremal flows with a pivot operation.

---

*The reader who is not conversant with mathematical programming might well wish to consult HADL64 for a general discussion of the gradient-projection method.

Step 3 Shortest-Route Computation

Employing any efficient shortest-route algorithm (e.g., FLOY62), compute the shortest route flow under the metric l^n.

$$l^n = \left(l_1^n, l_2^n, \ldots, l_{NA}^n \right) ,$$

where

$$l_j = \left(\partial T / \partial f_i \right)_f \; n ,$$

as defined previously.

Step 4 Stopping Rule

Let

$$\theta^n = \sum_{i=1}^{NA} l_i^n \left(f_i^n - \phi_i^n \right) .$$

IF $\theta^n < \epsilon$, for some previously defined ϵ, stop.

ELSE, go to step 5.

Step 5 Let $b = NA + 2$

$$\phi^{(NA+2),\,n} \leftarrow \phi^n$$

$$\lambda_{NA+2}{}^n = 0$$

$$n = n + 1$$

Go to step 1.

A starting feasible flow is obtained by relaxing capacity constraints and introducing penalty functions outside the feasible region. For an exact formulation, the reader should consult CANT74.

Backbone Design Model. The models developed by NAC are based on a paper [HSIE76] entitled "Locating Backbone Switches in a Large Packet Network." Our discussion of NAC's model(s) is based upon this paper.

At the outset, it is important to note that for a large network,* the following problems pose special and important challenges.

- Selection of the number and location of the backbone nodes.
- Assignment of terminals and hosts to backbone switches (Partitioning Problem).

The design of local access networks may be approached in the same way as the design of centralized networks, for which there exist good algorithms. A number of these algorithms have already been discussed in this paper; the reader is referred to the appropriate discussions for further references.

Similar remarks may be made concerning the design of the communications subnet.

A statement of the problem (from HSIE76) is as follows.

Given:

- User sites.
- Point-to-point traffic requirements.

---

*For discussions on the unique design problems involved with large networks, the reader would do well to consult GERL75 and FRAN73.

- Backbone switch cost and characteristics.
- Set of switch candidate sites.
- Line tariff alternatives.

Minimize:

Total communications cost D; where D = trunk cost + switch cost + local access cost.

Over the Variables:

- Switch number and location.
- Backbone topology.

Such that:

- Backbone traffic requirements are met.
- Backbone delay constraints are met.
- Backbone reliability constraints are met.
- Switch capacity constraints are met.

The following assumptions and approximations are used to simplify the design procedure:

- Switch cost is assumed linear with capacity.
- Backbone trunk cost is calculated using a direct distance model.

Those models will be presented which are at the core of the backbone design approach.

1. Backbone Network Cost Estimation — This model, which provides an estimate of the cost of a particular design, is employed in conjunction with the following two algorithms in order to evaluate the cost of different configurations without suffering an enormous "run-time" penalty.

2. An Add Algorithm — Patterned after the add algorithm discussed in section 2, this algorithm provides a solution to the backbone location problem.

3. A Cluster Algorithm — Patterned after the COM algorithm discussed in this section, this algorithm also provides a solution to the location problem.

It is important to note that the fundamental issue in selecting a location for a backbone switch is the trade-off between the local access costs of homing a particular user node to a switch which is already in place, and the costs of adding another switch to the backbone.

1. Backbone Network Cost Estimation. The first construct to be identified is the routing penalty, P. This is defined as the ratio of the shortest path distance in the network and the direct (actual) distance for the average source destination (i,j) pair. P is experimentally determined.

The mileage on the route between i and j, $|R|$, may be approximated by $|R| = P \times d_{ij}$, where d_{ij} is the actual mileage between i and j.

Given these two expressions, the model for the backbone line cost becomes:

$$D = \sum_{i} \sum_{j} \frac{cr_{ij}\, d_{ij}\, (1 + b)\, \rho}{\rho} + NN \cdot F$$

where:

ρ = average link utilization ($\rho \leq 1$)

F = fixed cost/node

c = cost/mile x unit bandwidth x month

r_{ij} = traffic requirement from i to j

NN = number of switches

b = line overhead (protocols)

d_{ij} = actual mileage between i and j

The first expression after the double summation is an estimate of the line costs between i and j, while the second expression represents the switch costs.

The coefficients P, ρ, and F are experimentally determined for different values of NN.

2. The Add Algorithm. We assume that the backbone locations $e_1, \ldots, e_k$, have already been chosen, and that not all user locations have been assigned to a switch. Let d represent the set of unassigned users. We then examine the trade-off of assigning each one of the users to either:

- A switch located at the traffic-weighted center of mass of mass C_0, or
- A switch that has not as yet been selected from the list of candidate backbone switches.

If T is an arbitrary user from d, the saving $S(Q)$ in homing T to Q is given by:

$$S(Q) = LAS_T(Q) + BLS_T(Q)$$

where the first expression refers to the local access saving, while the second refers to the backbone line saving.

The local-access saving is given by

$$LAS_T(Q) = k_T \cdot \left[d\left(T, C_0\right) - d\left(T, Q\right) \right]$$

where k_T = cost/mile for the local access line.

The backbone line saving is approximated by

$$BLS_T(Q) = t_T \cdot \frac{c_B}{t_{TOTAL}} \cdot \sum_{i=1}^{k} t_i \cdot \left[d\left(C_0, C_i\right) - d\left(Q, C_i\right) \right] \quad ,$$

where:

t_T = total traffic at site T

t_i = total traffic of users assigned to switch C_i

t_{TOTAL} = total network traffic

$c_B = c \cdot \frac{P}{\rho}$, where these symbols have been defined above

c_B is essentially the "cost/average route."

In true add algorithm fashion, we assign the user sites to Q in the order of their greatest saving until Q's capacity is used up.

The total savings from adding Q is then given by

$S(Q) = \sum_{T \text{ assigned to } Q} S_T(Q) - C_F$, where C_F is the switch cost.

The candidate with the largest saving S(Q) is then selected as the next switch site. In the event that no saving is possible among any of the candidates, the switch closest to the center of mass is chosen as the next site.

3. The Cluster Algorithm. This algorithm operates in much the same way as the already-discussed COM algorithm for TACOM location. The assumptions upon which its use is based are:

a) Backbone switch locations which minimize local access cost tend to minimize the total network cost.

   Hsieh, et al. feel, as a result of their practical experience, that the dominant factor in total network cost is the local access cost.

b) Natural geographic groupings of nodes are a good basis for partitioning the network.

Hsieh, et al. have also performed a comparison of the two algorithms, and note that the add algorithm performs better than the cluster algorithm, although it is more costly to use. They therefore recommend its use in the event that there are relatively few switch sites in comparison to the number of user sites. Given the opposite circumstances, use of the cluster algorithm is recommended.

Summary

NAC is, in the opinion of the author, the preeminent organization in the area of network analysis and design in the country. This is a natural

consequence of the company's history. Two of the principals, Howard Frank and Ivan Frisch, were largely responsible for the topological design of the ARPANET. While they were professors at Berkeley (prior to founding the company), they were active researchers in network analysis. This work culminated in their writing the standard text in the field, Communication, Transportation, and Transportation Networks [FRAN71a]. Dr. Frisch is presently editor-in-chief of the journal, Networks.

Two other well-known figures on their staff are Drs. Wushow Chou and Richard Van Slyke.

The company has made fundamental contributions to the state of the art in:

- Distributed Network Design — They developed the branch Xchange and cut-saturation algorithms.
- Network Partitioning and Concentrator Placement Algorithms — They developed the COM algorithm.
- Line-Layout Problem — The unified heuristic algorithm is theirs.
- Network Reliability — The functional simulation and Moore-Shannon simulations are theirs.
- Design of Large Distributed Networks — The backbone design approach presented in this paper appears to be without rival.
- Work on packet-radio and satellite communications has also been conducted.

In addition, NAC also maintains an active, on-going research program under the aegis of an ARPA contract.

SECTION 4

SUMMARY AND CONCLUSIONS

As indicated in the introductory section, the objective of this report has been to assess the efficacy of software packages available for the design of computer-communications networks. Our approach to the task has been to perform a comparative analysis of the algorithms employed by the packages in the various network design areas. As no opportunity presented itself for the actual application of any of these packages to a suitable Air Force problem, we have had to be content with this approach.

Several major observations may be made as a result of our comparisons. (The reader may wish to consult the summary charts appearing in tables 2 and 3.)

With two notable exceptions — the VANS model and the models developed by Network Analysis Corporation — the design tools are directed toward the design of centralized computer networks. This is a perfectly understandable tendency, since for most of the commercial market-place, a centralized design is what is really required. The military situation is quite different and will be further addressed below.

Another observation that may readily be made is that those organizations that only design centralized networks have patterned their products after IBM's Communications Network Design Program (CNDP). CNDP

Table 2

Programs' Functions

Name of Organization	Name of Program	Functions
Network Analysis Corporation	GRINDER MIND	Design of Distributed Networks Design/Analysis of Centralized Networks Design/Analysis of Large Networks Network Reliability
University of Minnesota Computer Science Dept.	VANS	Design of Network Protocols
Scientific Time Sharing Corp.	ODIN	Design/Analysis of Centralized Networks
Kranzley & Co. Telecommunications Division	PLANET	Design/Analysis of Centralized Networks
The DMW Group	NDMS ANDMS	Design/Analysis of Centralized Networks
Systems Architects, Inc.	SADC, SADM, SAND, TALK	Design/Analysis of Centralized Networks

Table 3

Specifics of Implementation

Name of Organization	Name of Program	Language/ Machine(s)	Availability	Access Method
Network Analysis Corporation	GRINDER MIND	FORTRAN IV Can run on most computers as doesn't require access to peripherals during execution	Time-Sharing or Installed on User's Computer	CRT
University of Minnesota Comp. Science Dept.	VANS	SIMULA and PASCAL CDC Cyber-74	User's Computer	CRT
Scientific Time Sharing Corporation	ODIN	APL	Time-Sharing Only	CRT
Kranzley & Co. Telecommunications Division	PLANET	FORTRAN IV	Time-Sharing or Installation on User's Computer	Batch
DMW Group	NDMS ANDMS	FORTRAN IV Can be installed on most systems	Time-Sharing or Installation on User's Computer	CRT and BATCH
Systems Architects, Inc.	SADC, SADM, SAND, TALK	FORTRAN IV TALK UNIVAC 1108 PPD-10 SANDS CDC 6600 UNIVAC 1108 IBM 370	Time-Sharing or Installation on User's Computer	CRT and BATCH

is available only in the form of an IBM marketing tool, and as such is not available for study. On the other hand, its algorithms have been extensively described by James Martin [MART72] of the IBM Systems Research Institute. Consequently, the line-layout problem is approached via the Esau-Williams algorithm (developed for CNDP) and the concentrator placement problem is solved employing an add algorithm, as is also done in CNDP.

A serious shortcoming in all of these packages, from the military point of view, is the lack of network reliability (survivability) programs.

With the exception of Scientific Time Sharing Corporation's model, all of the "central network" companies have written their programs in FORTRAN. Most of them are available in a time-sharing mode. The STS model is written in APL. Table 3 is a summary of the specific implementation details of each of the packages.

In noting some of the special requirements that must be made of a computer-communications network in the introduction to this paper, we listed four characteristics: good response time/throughput characteristics, survivability, expandability, and cost-effectiveness. Although the list could easily be extended, characteristics such as these indicate the necessity for a sophisticated technology — the ability to design survivable, distributed, packet-switching networks, for example. A corollary to this is the absolute necessity of possessing the best available design tools, which can make effective use of all the options a rapidly changing technology makes available. A second corollary is the support of a first-rate consulting staff to go along with these tools, as all problems are really unique and pose their own special challenges.

It is the opinion of the author that Network Analysis Corporation is the preeminent organization in the area of designing computer-communications networks. Indeed, the company may be said to have created the state of the art in many of the import design areas. NAC is alone in offering a tool for the design of a truly distributed, packet-switched network, alone in offering reliability models, and alone in offering tools for the backbone design of a large network. In addition, their staff is composed of acknowledged experts in the area.

The VANS model, created by Professor M. Schneider of the University of Minnesota, appears to have great potential for answering questions in the area of protocol design.

Both of these models would make excellent choices for application in any program being conducted by ESD in the area of network design.

REFERENCES

CANT74 D. G. Cantor and M. Gerla, "Optimal Routing in a Packet-Switched Computer Network," IEEE Trans. on Computers, C-23, 10 (1974), 1062-69.

CHOU75 W. S. Chou and P. McGregor, "A Unified Simulation Model for Communication Processors," in Trends and Applications 1975: Computer Networks, National Bureau of Standards Symposium, January 1975.

CLOS72 F. Clos, "Time-Delay and Trunk Capacity Requirements in Line-Switched Networks," ISS Record, 1972, pp. 428-33.

CROW75 W. R. Crowther et al., "Issues in Packet-Switching Network Design," in Proc., AFIPS 1975 National Computer Conference, 44 (1975), 161-76.

DAVI73 D. W. Davies and D. Barber, Communication Networks for Computers, New York: John Wiley & Sons, 1973.

DMW78 DMW Group, "Network Design and Management Tools," 1978.

ELLI78 Scientific Time Sharing, Inc., "ODIN Network Optimization System User's Guide," Bethesda, Md., January 1978.

ESAU66 L. R. Esau and K. C. Williams, "A Method for Approximating the Optimal Network," IBM Systems J., 5, 3 (1966), 142-47.

FARB72 D. J. Farber and K. Larson, "The System Architecture of a Distributed Computer System — The Communications Systems," in Proceedings, Symposium on Computer-Communications Network and Teletraffic, New York: Polytechnic Press, April 1972, pp. 21-27.

FISC76 M. J. Fischer and T. C. Harris, "A Model for Evaluating the Performance of an Integrated Circuit- and Packet-Switched Multiplex Structure," IEEE Transactions on Communications, COM-24, 2 (1976), 195-202.

FISZ63 M. Fisz, Probability Theory and Mathematical Statistics, New York: John Wiley and Sons, 1963.

FLAT74 L. Flato, "Know Your Common Carriers," Computer Decisions, December 1974, pp. 17-30.

FLOY62 R. Floyd, "Algorithm 97, Shortest Path," Communications of the ACM, 5 (1962), 345.

FRAN70a H. Frank and I.T. Frisch, "Analysis and Design of Survivable Networks," IEEE Transaction on Communication Technology, COM-18, 5 (1970), 501-19.

FRAN70b H. Frank et al., "Topological Considerations in the Design of the ARPA Network," in AFIPS Conference Proceedings, Spring Joint Computer Conference, Montvale, New Jersey, 36 (1970), 581-87.

FRAN71a H. Frank and I.T. Frisch, Communication, Transmission and Transportation Networks, Reading, Mass.: Addison-Wesley, 1971.

FRAN71b H. Frank et al., "Optimal Design of Centralized Computer Networks," Networks, 1 (1971), 43-57.

FRAN72 H. Frank and W. Chou, "Topological Optimization of Computer Networks," Proc. IEEE, 61, 11 (1972), 1385-97.

FRAN73 H. Frank, "The Practical Impact of Recent Computer Advances on the Analysis and Design of Large-Scale Networks," distributed by NTIS, United States Department of Commerce, Springfield, Va., 1973.

GERL74 M. Gerla et al., "A Cut-Saturation Algorithm for Topological Design of Packet-Switched Communication Networks," in Proc. IEEE National Telecommunications Conference, San Diego, Cal. (December 1974), 1074-85.

GERL75 M. Gerla et al., "Design Alternatives for Large Distributed Networks," in National Telecommunications Conference 1975, Conference Record, New Orleans, La., 2 (1975), 27-15 through 27-21.

GERL77 M. Gerla and L. Kleinrock, "On the Topological Design of Distributed Computer Networks," IEEE Transactions on Communications, COM-25, 1 (1977), 49-60.

HADL64 G. Hadly, Nonlinear and Dynamic Programming, Reading, Ma.: Addison-Wesley, 1964.

HAMM64 J.M. Hammersley and D.C. Handscomb, Monte Carlo Methods, Methuen Monographs on Applied Probability and Statistics, Methuen, Mass., 1964.

HSIE76 W. Hsieh et al., "Locating Backbone Switches in a Large Packet Network," in Proceedings of The Third International Conference on Computer Communications, Toronto, Canada, 3-6 August 1976.

JACK57 J.R. Jackson, "Networks of Waiting Lines," Operations Research, 5 (1957), 518-21.

JACK63 J.R. Jackson, "Jobshop-Like Queueing Systems," Management Science, 10, 1 (1963), 131-42.

KERS74 A. Kershenbaum and W.S. Chou, "A Unified Algorithm for Designing Multidrop Teleprocessing Networks," IEEE Transactions on Communications, COM-22, 11 (1974), 1762-72.

KIMB75 S.R. Kimbleton and G.M. Schneider, "Computer Communications Networks: Approaches, Objectives, and Performance Considerations," A.C.M. Computing Surveys, 7, 3 (1975), 129-73.

KLEI64 L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, New York: McGraw-Hill, 1964.

KLEI76 L. Kleinrock, Queueing Systems: Vol. II, Computer Applications, New York: Wiley-Interscience, 1976.

MART72 J. Martin, Systems Analysis for Data Transmission, Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

MCGR75a P. McGregor et al., "Communications Processor Simulation: A Practical Approach," in National Telecommunications Conference, New Orleans, La., vol. I, December 1975.

MCGR75b P. McGregor and D. Shen, "Locating Concentrators in Data Communications Networks," in Proceedings of the Fourth ACM Data Communications Symposium, Quebec, Canada, 7-9 October 1975.

MOOR56 E. G. Moore and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays," J. of the Franklin Institute, 262, part I (1956), 191-208.

NAC Network Analysis Corp., "Software Tools for Network Design and Analysis," Great Neck, N. Y.

NAC73 Network Analysis Corporation, "Second Semi-Annual Report, 1973," Great Neck, N. Y.

NAC75 Network Analysis Corp., "COM Algorithm for Access Facility Location," Great Neck, N. Y. (1975).

PANG78 G. Pan, "Organizational Profile of Systems Architects, Inc.," Randolph, Mass., 1978.

PICK76 R. L. Pickholtz and M. Schwartz, eds., "Symposium on Modeling and Analysis of Data Networks," Washington, D. C., 18-19 March, 1976.

ROTH71 B. Rothforb and M. C. Goldstein, "The One-Terminal TELPAK Problem," J. of the ORSA, 19 (1971), 157-69.

SCHN76 G. M. Schneider, "A Modular Approach to Computer Network Simulation," Computer Networks, 1 (1976), 95-98.

SCHN78 G. M. Schneider, "A New Methodology for Computer Network Simulation," in Simulation, 1978.

SCHW77 M. Schwartz, Computer-Communication Network Design and Analysis, Englewood Cliffs, New Jersey: Prentice-Hall, 1977.

VANS72 R. Van Slyke and H. Frank, "Network Reliability Analysis: Part I," Networks, 1, 3 (1972), 279-90.

YAGE71 B. Yaged, "Minimum Cost Routing for Static Network Problems," Networks, I (1971), 139-72.