On the Topological Design of Distributed Computer Networks

MARIO GERLA, MEMBER, IEEE, AND LEONARD KLEINROCK, FELLOW, IEEE

Abstract—**The problem of data transmission in a network environment involves the design of a communication subnetwork. Recently, significant progress has been made in this technology, and in this article we survey the modeling, analysis, and design of such computer-communication networks. Most of the design methodology presented has been developed with the packet-switched Advanced Research Projects Agency Network (ARPANET) in mind, although the principles extend to more general networks.**

We state the general design problem, decompose it into simpler subproblems, discuss the solutions to these subproblems, and then suggest a heuristic topological design procedure as a solution to the original problem.

I. INTRODUCTION

MANY stand-alone computer systems were configured and put into operation long before anyone seriously analyzed their performance (a procedure which sometimes led to embarrassing failures). In contrast, the field of computer-communication networks is at once both unusual and fortunate in that great care has gone into analysis and design techniques prior to system implementation. In this paper we wish to survey some of the recent mathematical techniques which have been found useful in the topological design and performance evaluation of computer-communications networks. Most of the procedures we describe below were first developed in the process of designing the Advanced Research Projects Agency Network (ARPANET) [3], [6], [7], [13], [15], [21], [24], [25], [31], [32], [43], [44], [46], [47], but were later applied to the design of a large variety of Government and commercial distributed data networks.

Many of the early computer networks were constructed mainly to provide access to a *centralized* computer service from a large number of remote users. Such centralized networks have a tree structure, with the computer located at the root of the tree and the terminals located at the nodes. The communication lines are shared among several terminal users by means of multidrop, multiplexing, and concentration techniques. Considerable research effort has been spent on the minimum cost design of these centralized computer networks, and a vast literature is now available [2], [4], [12].

In the pursuit of more efficient computer configurations, it was recognized in the late 1960's that the utilization of existing computer systems could be improved by connecting them together as a resource-sharing network [32]. Among the shared resources we include: computer power (for load sharing); specialized hardware; specialized software; and data files. This type of network differs from the former in that the computer resources are distributed among the nodes, rather than accumulated in a central node; this configuration is here referred to as a *distributed* computer network. One of the first examples of a distributed network is represented by the ARPANET, a recent configuration of which is shown in Fig. 1.

In distributed networks traffic demands can arise between any two nodes of the network, and not only between terminals and the "central" node. Consequently, better cost-effectiveness and performance are achieved with topological configurations which present a higher degree of connectivity than the centralized tree structures [20].

Computer network users share not only processing facilities, but also communication facilities [30]. The cost-effective configuration and use of communication channels is the main concern of this paper. Conventional line-switching techniques, as used by the common carrier switching network, in which a dedicated path is established for each conversation, are inefficient for computer communications in a bursty mode (e.g., terminal-to-computer conversations). In fact, with the present technology, the time required for establishing and clearing (disconnecting) the path is much longer (on the order of seconds) than the average intercomputer conversation. A more efficient solution for line sharing and speed conversion in a bursty data communication environment is provided by the packet-switching (P/S) technique [27]. Each message is segmented into packets at the source; these packets, instead of traveling along paths reserved in advance, adaptively find their way through the network independently, in a store-and-forward fashion. More than one route between source and destination may be used; such routes can be thought of as pipelines, along which several packets may travel simultaneously, interleaved with other packets corresponding to different source-destination pairs. It is through this pipelining and interleaving that large improvements in network thoughput and message delay are achieved.

The design of distributed P/S networks is substantially different and more complex than the design of centralized networks. The presence of more than one route between each origin and destination in the distributed case requires the solution of complex routing and capacity allocation problems; furthermore, the use of P/S techniques requires the analysis of the relationship between packet delay and line and buffer utilization.

Several algorithms have been proposed for the design of distributed networks. Some of the algorithms are of a heuristic nature; some others are based on more rigorous mathematical

Manuscript received April 9, 1976; revised July 23, 1976. This work was supported in part by the Advanced Research Projects Agency of the Department of Defense, under Contracts DAHC 15-75-C0135 and DAHC 15-73-C-0368.

M. Gerla was with Network Analysis Corporation, Glen Cove, NY 11542. He is now with the Department of Computer Science, University of California, Los Angeles, CA 90024 and the Computer Transmission Corporation, Los Angles, CA.

L. Kleinrock is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

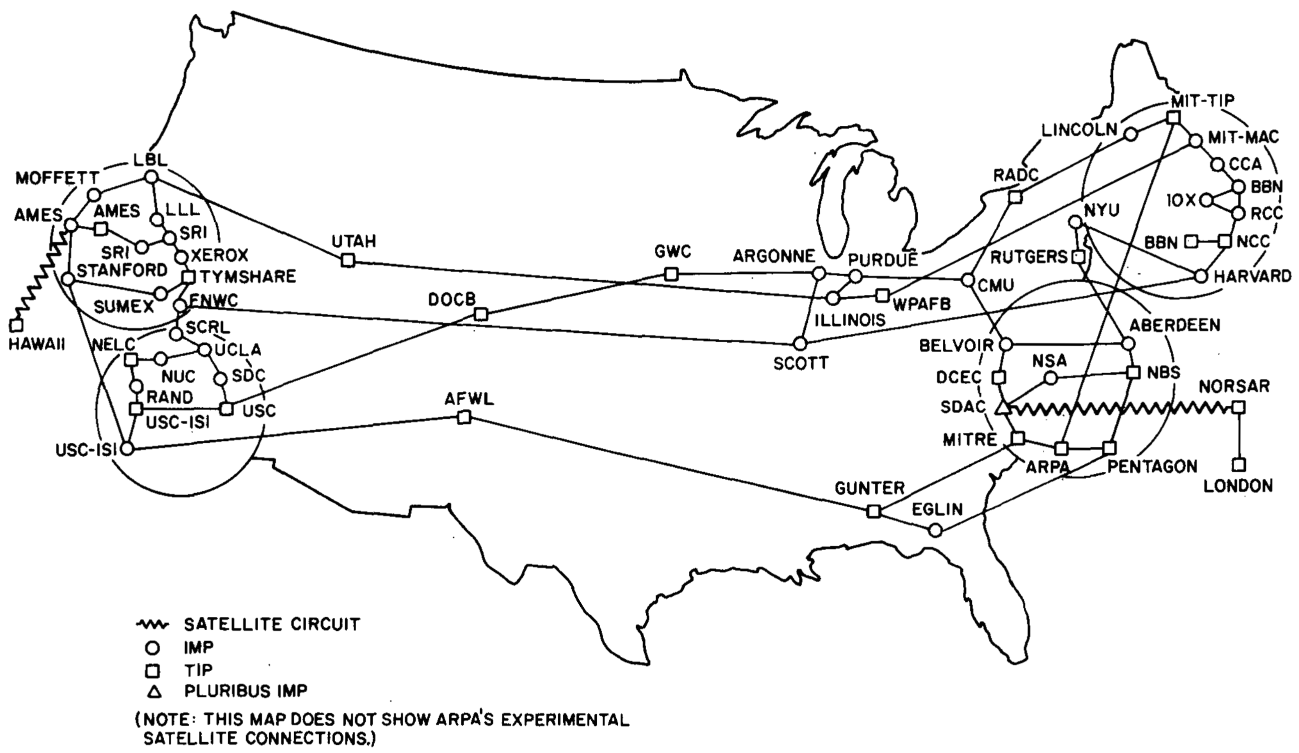


Fig. 1. ARPA geographical map, June 1976.

programming, queueing, and network flow concepts. In this paper we present a survey of the mathematical programming and network flow approaches that are available for the design of packet networks. We also describe the most common heuristics and compare them to the mathematical programming approach based on the criteria of computation time and solution accuracy.

II. THE DESIGN PROBLEM

Several different formulations of the design problem can be found in the literature [27]; generally, they correspond to different choices of performance measures, of design variable, and of constraints [20], [29]. Here, we select the following very general formulation:

Given	Node locations Peak-hour traffic requirements between node pairs
Minimize	Total line cost
Over the design variables	Topology Channel capacities Routing policy
Subject to	Link capacity constraints Average packet delay constraints Reliability constraint

Other common formulations of the design problem are the following: 1) minimize average packet delay given the network cost; 2) maximize network throughput given cost and admissible average delay. It is shown in Gerla [20] that all these formulations are closely related and that the solution techniques that apply to our general formulation also apply to the other problems.

In the following sections, we first introduce a network model and discuss the relations between performance measures, input parameters, design variables, and constraints that appear in the general design problem. Then we define and solve (in various degrees of completeness) three design subproblems (capacity assignment, flow assignment, flow and capacity assignment) which are derived from the general problem formulation by fixing some of the design variables. Finally, we study the solution of the general problem (i.e., including the topological design) in which the three above-mentioned problems appear as essential subproblems.

III. MODELING AND ANALYSIS

A. The Model

In a P/S network, packets are transmitted through the network using a store-and-forward technique [26], [27]. That is, a packet traveling from source node s to destination node d is received and "stored" in queue at any intermediate node k, while awaiting transmission, and is sent "forward" to node p, the next node on the route from s to d, when channel (k,p) permits. Even when this channel is free, the packet must first be received fully in node k before transmission to node p may be started. Given the destination d and the present node k, the selection of the next node p is made by a well-defined decision rule referred to as the routing policy. A routing policy is said to be fixed (or static) if a predetermined fraction of the packets arriving at k and directed to d is sent to each output queue; it is said to be adaptive if the selection of the output channel at each node depends on some estimate of current network traffic [27].

Traffic requirements between nodes arise at random times and the size of the requirement is also a random variable. Consequently, queues of packets build up at the channels and the system behaves as a stochastic network of queues. For routing purposes, packets are distinguished only on the basis of their destination [17], [20], [27]; thus, messages having a common destination can be considered as forming a "class of customers." The P/S network, therefore, can be modeled as a network of queues with n classes of customers where n is the number of different destinations [29].

B. Delay Analysis

A vital performance measure for a computer-communication network is the average source-to-destination packet delay T, defined as follows [27]:

$$T = \sum_{\substack{j,k \\ j \neq k}} \frac{\gamma_{jk}}{\gamma} Z_{jk} \tag{1}$$

where

γ_{jk}	average packet rate flowing from *source* j to *destination* k
Z_{jk}	average packet delay (queue and transmission) from j to k

$$\gamma = \sum_{j,k} \gamma_{jk} \tag{2}$$

A straightforward application of Little's result [28], [29] to the network of queues model leads to the following very useful expression for T:

$$T = \sum_{i=1}^{b} \frac{\lambda_i}{\gamma} T_i \tag{3}$$

where b is the number of links (arcs), λ_i is the average traffic rate, and T_i is the average queueing plus transmission delay on link i. This expression, established by Kleinrock in 1964 [27], is very general (as general as Little's result!), and extremely simple. Unfortunately, we are not able in general to evaluate λ_i and T_i. However, if we make the following assumptions: 1) external Poisson arrivals; 2) exponential packet length distribution; 3) infinite nodal storage; 4) fixed routing; 5) error-free channels; 6) no nodal delay; and 7) independence between interarrival times and transmission times on each channel, then the evaluation of (3) can be carried out analytically [26], [27]. In fact, the network of queues reduces to the model first studied by Jackson [23], in which each queue behaves as an independent $M/M/1$ queue [28]. Thus, the average delay T_i on channel i is given by

$$T_i = \frac{1}{\mu C_i - \lambda_i} \tag{4}$$

where

$1/\mu$	average packet length (bits/packet)
C_i	capacity of channel i (bits/s)
λ_i	average packet rate on channel i (packets/s).

The average rates λ_i are easily computed from the routing tables and the traffic requirement matrix [20], [27].

By substituting (4) into (3) and letting f_i be the average bit rate on channel i (bits/s), we obtain the following expression for T:

$$T = \frac{1}{\gamma} \sum_{i=1}^{b} \frac{f_i}{C_i - f_i}. \tag{5}$$

Although the delay expression (5) is sufficiently accurate for most design purposes, it is possible to obtain expressions which correspond more accurately to measured results. Kleinrock proposed in [25] the following very general formula, which includes propagation and nodal processing delay:

$$T = \frac{1}{\gamma} \sum_{i=1}^{b} \lambda_i [T_i + P_i + K_i] \tag{6}$$

where

P_i	propagation delay in channel i (seconds)
K_i	nodal processing time in the node at which channel i terminates (s/packet).

The term T_i depends on the nature of the traffic and on the packet length distribution.

Finally, assuming a general packet length distribution, with mean $1/\mu$ and variance σ^2, we obtain [26]:

$$T_i = \frac{1}{\mu C_i}(1 - \beta) + \frac{\beta}{\mu C_i - \lambda_i} \tag{7}$$

where $\beta = (1 + \mu^2\sigma^2)/2$. An even more detailed model has been recently proposed and is discussed in [31].

The validity of the above assumptions and approximations has been tested and verified through simulation and measurements by many authors in a variety of applications [17], [25], [27], [31]. The results indicate that the model is robust.

In this paper, without loss of generality, we limit our considerations to the delay expression in (5). Most of the techniques described in the sequel can be easily extended to more elaborate delay expressions. Furthermore, the solution of some of the design problems seems to be rather insensitive to the introduction of additional details in the delay formula [20].

C. The Communications Cost

With C_i the capacity of link i, we let $d_i(C_i)$ be the cost of leasing capacity value C_i for link i. The communication cost D is defined as

$$D = \sum_{i=1}^{b} d_i(C_i). \tag{8}$$

The availability and effectiveness of the design algorithms depends rather critically upon the form for $d_i(C_i)$. In most applications $d_i(C_i)$ is a discrete variable; however, for computational efficiency it is often convenient to approximate the discrete costs with continuous costs during the initial optimization phase, and to discretize the continuous values during a refinement phase [25].

D. Traffic Requirements

Average (busy-hour) traffic requirements between nodes can be represented by a requirement matrix $R = \{r_{jk}\}$, where r_{jk} is the average transmission rate from source j to destination k. In some cases, we define the requirement matrix as $R = \rho\overline{R}$, where $\overline{R}$ is a known basic traffic pattern and ρ is a variable scaling factor usually referred to as the traffic level.

In general, R (or $\overline{R}$) cannot be estimated accurately *a priori,* because of its dependence upon network parameters (e.g., allocation of resources to computers, demand for resources, etc.) which are difficult to forecast and are subject to changes with time and with network growth. Fortunately, the analysis of several different traffic situations has shown that the optimal design is rather insensitive to traffic pattern variations [1], [20]. This insensitivity property, which seems to be typical of distributed networks, justifies the use of traffic averages for network design.

E. Routing Policy

In designing network topologies, one generally assumes fixed routing, since fixed routing is easy to describe (by means of routing tables, for example) and allows the direct evaluation of channel flows and average delay as a function of routing tables and traffic requirements. Adaptive routing, on the other hand, is complex to describe, and requires simulation to evaluate channel flows and delay. Furthermore, it was shown that *at steady state,* flow patterns and delays induced by good adaptive routing policies are very close to those obtained with optimal fixed policies [18]. This fact suggests that network configurations optimized with fixed routing, are also (near) optimal for adpative routing operations [27].

F. Link Flows

The routing policy and the traffic requirements uniquely determine the vector $f \triangleq (f_1, f_2, \cdots, f_b)$ where f_i is the average data flow on link i. The evaluation of f is straightforward in the case of fixed routing; it can, at this point, only be obtained by simulation if the routing is adaptive.

Conversely, not any generic vector f corresponds to a realizable routing policy and requirement matrix. If it does, then f is a multicommodity (MC) flow for that particular requirement matrix. An MC flow results from the sum of single commodity flows f^{jk} $(j,k = 1,2, \cdots, n)$ where f^{jk} is the average flow vector generated by packets with source node j and destination node k, and n is the number of nodes. Clearly, each single commodity flow of the MC flow must separately satisfy nonnegativity and flow conservation constraints.

G. The Capacity Constraint

The presence of capacity constraints $f \leqslant C$ (where $C = (C_1, C_2, \cdots, C_b)$) makes the design problem in Section II a constrained MC flow problem. From the delay expressions (4) and (7), we notice that if the link flow approaches the link capacity, then the delay approaches infinity, thus violating the delay constraint [16]. Therefore, if both capacity and delay constraint must be satisfied the *capacity constraint is implied by the delay constraint and can be disregarded.*

H. Reliability

Links and nodes in a real network can fail with nonzero probability, thus interrupting some communication paths. It is important to evaluate the overall network reliability in the presence of such failure probabilities.

Several reliability measures have been proposed for computer-communication networks. Among them we mention: the probability of the network being connected (PC); and the fraction of communicating node pairs (FR). Such measures must be verified during the design phase.

Van Slyke and Frank developed very efficient techniques for the evaluation of PC and FR [36]. The techniques, however, are based on simulation and are too time-consuming to be included in a global design algorithm.

Roberts and Wessler [32] proposed as a reliability measure, the two-connectivity of the network (i.e., two node-disjoint paths available between each node pair). This measure is easy to include as a constraint in the topological design. Furthermore, it is adequate for networks with a relatively small number of nodes (on the order of 20-40) and relatively small component failure probability (on the order of 0.01).

For larger networks (or higher failure rates), stronger constraints must be applied to the network topology (e.g., three-connectivity, no long chains, etc.) in order to obtain adequate reliability.

This concludes our model description. In the next four sections, we discuss some of the important design problems and their solutions.

IV. THE LINK CAPACITY ASSIGNMENT (CA) PROBLEM

A. Problem Formulation

The CA problem can be formulated as follows:

Given	Topology Requirement matrix R Routing policy (and therefore link flow vector $f = (f_1, f_2, \cdots, f_b)$)	
Minimize	$D = \sum_{i=1}^{b} d_i(C_i)$	(9)
Over the design variables	$C = (C_1, C_2, \cdots, C_b)$	
Subject to	$f \leqslant C$	
	$T = \frac{1}{\gamma} \sum_{i=1}^{b} \frac{f_i}{C_i - \lambda_i} \leqslant T_{\max}$	

The optimal assignment of capacities to a distributed network with arbitrarily fixed routes is not very interesting as a stand-alone problem, since routing plays a determinant role in the optimization of network performance. Rather, the CA is of practical interest as subproblem of more general optimization problems.

The technique used for the solution of the CA problem depends on the nature of the cost-capacity functions $d_i(C_i)$. In the following, we present optimal and suboptimal algorithms for linear, concave, and discrete costs.

B. Linear Costs

Assuming that $d_i(C_i) = d_iC_i + d_{i0}$ where d_{i0} is a positive start-up cost, the optimal solution is obtained using the method of Lagrange multipliers [27]. In particular, the optimal capacity for channel i is given by

$$C_i = f_i + \frac{\sum_{j=1}^{b} \sqrt{d_jf_j}}{\gamma T_{\max}} \sqrt{\frac{f_i}{d_i}} \tag{10}$$

and the minimum cost D is given by

$$D = \sum_{i=1}^{b} \left[d_if_i + d_{i0} + \frac{\left(\sum_{j=1}^{b} \sqrt{d_jf_j}\right)^2}{\gamma T_{\max}} \right] \tag{11}$$

C. Concave Costs

The concave case can be solved iteratively by linearizing the costs and solving a linearized problem at each iteration [20]. The method leads, in general, to local minima. However, for the very important case of a power law cost function (i.e., $d_i(C_i) = d_iC_i^{\alpha} + d_{i0}$ where $0 \leqslant \alpha \leqslant 1$), Kleinrock showed that there exists a unique local minimum [25]. The iterative procedure therefore yields the optimal solution in the power law case.

D. Discrete Costs

The optimal solution to the discrete CA problem can be obtained with a dynamic programming (DP) technique. The DP algorithm, described by Gerla in [20], requires an amount of computation which, in most applications, is close to $(b)^2$. Another suboptimal technique for the solution of the discrete CA problem is the Lagrangian decomposition (LD). The LD method, first developed by Everett [9], and subsequently improved by Fox [10] and Whitney [37], is suboptimal in the sense that it determines only a subset of the set of optimal solutions corresponding to various values of the parameter $T_{\max}$. The amount of computation required by LD is slightly more than linear with respect to the number of arcs b.

The delay versus cost plot in Fig. 2 shows DP and LD solutions for a large range of values of $T_{\max}$ for a discrete CA application relative to a 26-node ARPANET topology [20]. The circles correspond to LD solutions, whereas the union of circles and dots corresponds to DP solutions. As a property of the LD method, the LD solutions belong to the convex envelope of the global set of optimal DP solutions.

V. THE ROUTING PROBLEM

A. Problem Definition

The routing problem is here defined as the problem of finding the fixed routing policy which minimizes the average delay T. A possible formulation of the problem is the following:

Given	Topology Channel capacities $\{C_i\}$ Requirement matrix R
Minimize	$T = \frac{1}{\gamma}\sum_{i=1}^{b} f_i\left[\frac{1}{C_i - f_i} + \mu(P_i + K_i)\right]$ (12)
Over the design variable	$f = (f_1, f_2, \cdots, f_b)$
Subject to	a) f is a multicommodity flow satisfying the requirement matrix R b) $f \leqslant C$

From formulation (12), we notice that the routing problem is a convex MC flow problem on a convex constraint set; therefore, there is a unique local minimum, which is also the global minimum and can be found using any downhill search technique [5].

Several optimal techniques for the solution of MC flow problems are found in the literature [8], [35]; however, their direct application to the routing problem proves, in general, to be cumbersome and computationally not efficient. Consequently, considerable effort was spent in developing heuristic suboptimal routing techniques [5], [17], [33]. Satisfactory results were obtained and computational efficiency was greatly improved. However, all of these techniques are affected by various limitations and may fail in some pathological situations.

A new downhill search algorithm, called flow deviation (FD), was recently developed by Fratta, Gerla, and Kleinrock [16]. The FD algorithm finds the optimal solution and is computationally as efficient as the heuristics. To place the FD algorithms in the proper perspective we first introduce the most popular among the heuristic algorithms—the "minimum link" algorithm—and compare it to the FD algorithm.

B. The Minimum Link Algorithm

We begin by giving an outline of the heuristic algorithm reported in [5].

Algorithm:

Step 1: For a given source j and destination k, determine all paths Π_{jk} with the minimum number of intermediate nodes. Such paths are called "feasible" paths.

Step 2: Choose, among the feasible paths, the least utilized path (or the path with maximum residual capacity).

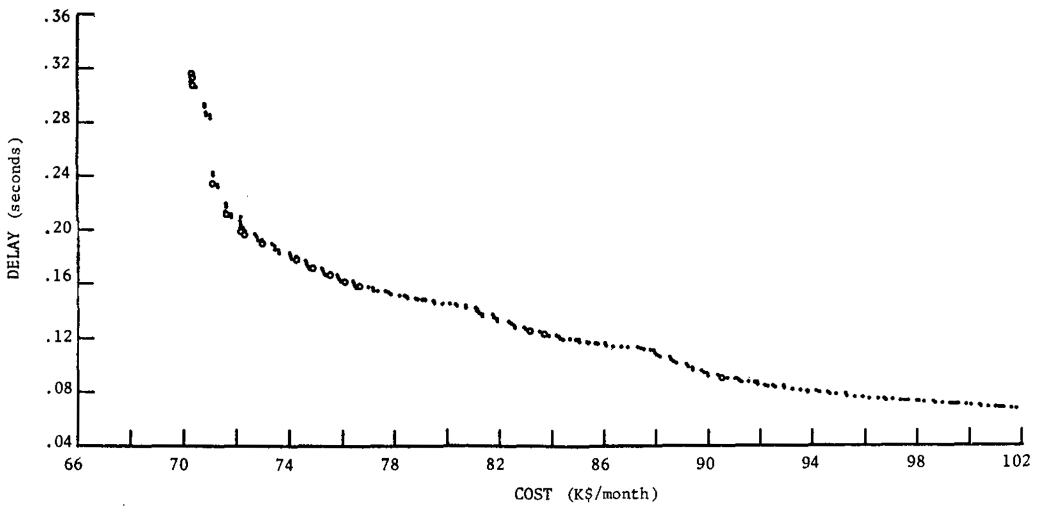


Fig. 2. Delay versus cost for discrete capacity assignments on 26-node 30-link topology.

Step 3: Route the requirement γ_{jk} along such a path.

Step 4: If all source destination pairs have been processed, stop; otherwise select a new pair and go to 1.

Step 1, repeated for all node pairs, corresponds to the evaluation of all shortest paths between all pairs of nodes, assuming unitary link length. Such a computation requires from $(n)^2$ to $(n)^2 \log(n)$ operations, depending on network connectivity. It can be shown that the total amount of computation required by the algorithm has proportion between $(n)^2$ and $(n)^2 \log(n)$.

The minimum link algorithm is conceptually simple and computationally very efficient. Its major drawback is that of being rather insensitive to queueing delays and therefore possibly far from optimum in heavy traffic situations.

C. The Flow Deviation Algorithm

Before introducing the FD algorithm, we mention the following properties of the optimal routing solution.

Property 1: The set of MC flows f satisfying the requirement matrix R is a convex polyhedron. The extreme points of such a polyhedron are called "extremal flows" and correspond to shortest route policies. A shortest route policy is a policy that routes each (j,k) commodity along the shortest (j,k) path, evaluated under an arbitrary assignment of lengths to the links. To each such assignment there corresponds an extremal flow and conversely. Any MC flow can be expressed as a convex combination of extremal flows [16].

Property 2: For a given MC flow f, let us define link length as a function of link flow of the form $l_i \triangleq \partial T/\partial f_i$. Let ϕ be the shortest route flow associated with such link lengths and let $f' = (1-\lambda)\phi + \lambda f$ be the convex combinations of ϕ and f minimizing the delay T. If $T(f') = T(f)$, then f is optimal.

Property 2 provides a way of finding a downhill direction, if it exists. Based on such property, we may now state the FD algorithm, as follows.

Algorithm:

Step 0: Let $f^{(0)}$ be a starting feasible flow. (A starting feasible flow can be obtained using a modified version of the FD algorithm [16].) Let $p = 0$.

Step 1: Compute $\phi^{(p)}$, the shortest route flow corresponding to $l_i^{(p)} = [\partial T/\partial f_i]_{f=f^{(p)}}, \forall i = 1, \cdots, b$.

Step 2: Let $\bar{\lambda}_p$ be the minimizer of $T[(1-\lambda)\phi^{(p)} + \lambda f^{(p)}]$, $0 \leqslant \lambda \leqslant 1$. Let $f^{(p+1)} = (1-\bar{\lambda}_p)\phi^{(p)} + \bar{\lambda}_p f^{(p)}$.

Step 3: If $|T(f^{(p+1)}) - T(f^{(p)})| < \epsilon$, stop: $f^{(p)}$ is optimized to within the given tolerance. Otherwise let $p = p + 1$ and go to Step 1. A geometric representation of the FD algorithm is given in Fig. 3.

Step 1 is the most time-consuming operation of the algorithm, and requires an amount of computation between $(n)^2$ and $(n)^2 \log(n)$. Therefore, the amount of computation required by the minimum link algorithm and the FD algorithm are comparable. A typical central processing unit (CPU) requirement is from 2 to 4 seconds for a 30-node application on a large computer.

VI. THE CAPACITY AND FLOW ASSIGNMENT (CFA)

A. Problem Formulation

The CFA problem can be formulated as follows:

Given	Topology Requirement matrix R Cost-capacity functions $d_i(C_i)$
Minimize	$D(C) = \sum_{i=1}^{b} d_i(C_i)$ (13)
Over the design variables	f, C
Such that	a) f is an MC flow satisfying the requirement matrix R b) $f \leqslant C$ c) $T(f,C) = \frac{1}{\gamma}\sum_{i=1}^{b} \frac{f_i}{C_i - f_i} \leqslant T_{\max}$

The CFA problem requires the simultaneous optimization of routes and line capacities. The existence of a huge number

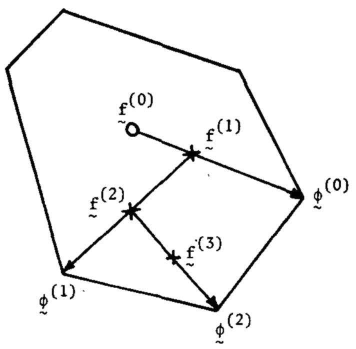


Fig. 3. Geometrical representation of the FD algorithm. Characters with tildes beneath are boldface in text.

of local minima makes the exact solution difficult to obtain. Therefore, we discuss here only suboptimal techniques.

B. Linear Costs.

In the linear cost case we can obtain, for a given f, a closed form expression of the optimal C in terms of f [see (9)]. In particular, the total cost D can be expressed in terms of f only, and problem (13) can be reformulated as follows:

Given: Topology; Requirement matrix R

Minimize

$$D(f)=\sum_{i=1}^{b}\left[d_i f_i + d_{i0} + \frac{\left(\sum_{j=1}^{b}\sqrt{d_j f_j}\right)^2}{\gamma T_{\max}}\right] \tag{14}$$

Over the design variable f

Such that f is an MC flow satisfying R

It can be shown that $D(f)$ is concave over the convex polyhedron of feasible multicommodity flows [20]. This implies that there are in general several (in fact, enormous numbers of) local minima corresponding to some corners of the polyhedron, i.e., corresponding to some extremal flows (see Fig. 4). The FD method described in the previous section can still be applied, in a properly modified form; however, it leads to local minima. More precisely, the FD algorithm performs a local search on extremal flows, until it finds a local minimum. The modified FD algorithm is next introduced.

FD algorithm (for concave objective function):

Step 0: Let $f^{(0)}$ be a feasible starting flow. Let $p = 0$.

Step 1: Let $f^{(p+1)}$ be the extremal flow corresponding to the following definition of equivalent lengths:

$$l_i^{(p)} = [\partial D/\partial f_i]_{f_i=f_i^{(p)}}, \qquad \forall i = 1, 2, \cdots, b.$$

Step 2: If $D(f^{(p+1)}) \geqslant D(f^{(p)})$, stop: $f^{(p)}$ is a local minimum. Otherwise, let $p = p + 1$ and go to 1.

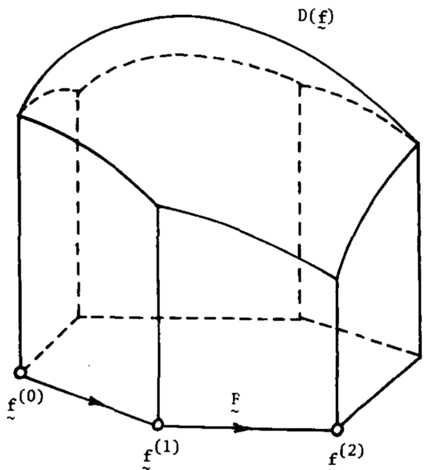


Fig. 4. Concave objective function $D(f)$. Characters with tildes beneath are boldface in text.

The convergence follows from the fact that there are only a finite (albeit large) number of extremal flows and repetitions are not allowed because of the stopping rule 2.

From (14) the equivalent length l_i defined in Step 1 has the following expression:

$$l_i = d_i\left[1 + \frac{\sum_{j=1}^{b}\sqrt{d_j f_j}}{\gamma T_{\max}}\cdot\frac{1}{\sqrt{d_i f_i}}\right] \tag{15}$$

Notice that $\lim_{f_i\to 0} l_i = \infty$. This implies that whenever the flow (and therefore the capacity) of link i is reduced to zero at the end of an iteration, flow and capacity will remain zero for all subsequent iterations, since the incremental cost of restoring the flow on a link is proportional to l_i which in this case is infinity. In other words, uneconomical links tend to be eliminated by the algorithm. This link elimination property, first observed by Yaged [38], can be utilized in the topological design, as shown in the following section.

The solution obtained with the FD algorithm is a local minimum which depends on the selection of the starting flow $f^{(0)}$. In order to obtain a more accurate estimate of the global minimum, several locals are usually explored, starting from randomly chosen flows.

C. Concave Costs

For concave channel costs there is no closed-form expression of D in terms of f. However, it has been shown by Gerla that $D(f)$ is concave over f, and that the FD algorithm can be still applied to obtain local minima [20].

An application of the FD method to the topology shown in Fig. 5 is next introduced. Channel capacities are available only in discrete sizes (see Table I); therefore, discrete channel costs are approximated with continuous power law costs, to apply the concave cost version of the FD algorithm.

A uniform traffic requirement of 1 kbit/s was assumed between all node pairs. Fifty different local minima were obtained with the FD method using randomly generated

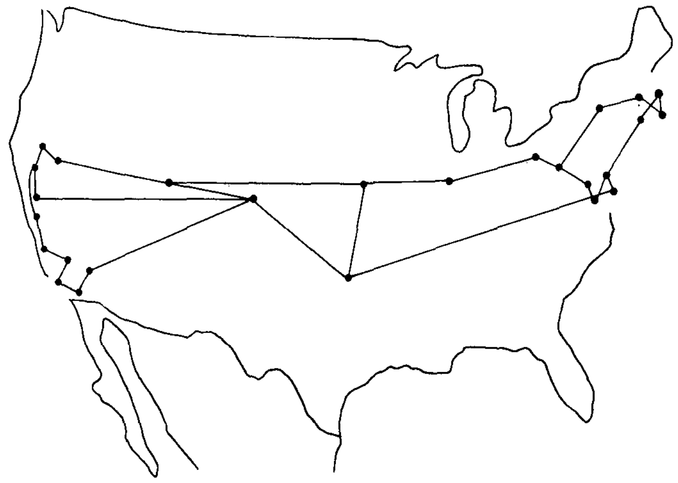


Fig. 5. ARPA-like topology with 26-nodes and 32 links.

TABLE I
LINE COST (TELEPAK RATES)

CAPACITY [K BITS/S]	TERMINATION COST [$/MONTH]	MILEAGE COST [$/MONTH/MILE]
9.6	650	.40
19.2 (2 x 9.6)*	1,300	.80
19.2	850	2.50
50.0	850	5.00
100.0 (2 x 50)*	1,700	10.00
230.4	1,350	30.00

TABLE II
DISTRIBUTION OF LOCAL SOLUTIONS

D[$/MONTH]	NO. OF SOLUTIONS
88,400 - 88,500	8
88,500 - 88,600	11
88,600 - 88,700	11
88,700 - 88,800	19
88,800 - 90,000	1

starting flows. The distribution of the local solutions in Table II shows that all the solutions fall in a very narrow range. Due to the random procedure used to select starting flows, we may conjecture that the optimal solution is also close to this cost range [20].

The execution time required to generate the 50 local minima was about 50 seconds on an IBM 360/91 (about 1 second per local solution).

D. Discrete Costs

For the discrete cost problem one of the following heuristic approaches may be used.

Approach 1: Solve, iteratively, a routing problem with fixed capacities, followed by a discrete CA problem with fixed flows, until a local minimum is obtained.

Approach 2: Interpolate discrete costs with continuous, concave costs. Solve the corresponding concave CFA problem. Adjust the continuous capacities to the smallest feasible discrete values. Reoptimize the flow assignments by solving a routing problem.

Four algorithms are next introduced, the first three following Approach 1 and the last following Approach 2.

1) Minimum Link Assignment [13]:

Step 1: Apply the minimum link routing algorithm, as described in Section V-B, to assign link flows.

Step 2: Allocate to link i the smallest feasible capacity C_i, such that

$$C_i > f_i, \qquad \forall i = 1, \cdots, b.$$

Step 3: Reoptimize the routing, so as to maximize throughput.

2) Bottom Up Algorithm [20]:

Step 1: Assign minimum available capacities to the links.

Step 2: Maximize the throughput ρ at $T \leqslant T_{\max}$, using the FD algorithm.

Step 3: If $\rho \geqslant \bar{\rho}$ (where $\bar{\rho}$ is the required traffic level) stop; we have a feasible suboptimal solution. Otherwise, increase the capacity on the "most utilized" link and go to Step 2.

3) Top Down Algorithm [20]:

Step 1: Assign maximum available capacities.

Step 2: Maximize ρ at $T \leqslant T_{\max}$, using the FD algorithm.

Step 3: If $\rho < \bar{\rho}$ (where $\bar{\rho}$ is the required traffic level) stop; routing and capacities of the previous iteration represent the feasible suboptimal solution. Otherwise, decrease the capacity on the "least utilized" channel and go to Step 2.

4) Discrete Capacity (Dis Cap) Algorithm [20]:

Step 1: Interpolate discrete costs with continuous, power law costs.

Step 2: Solve the continuous CFA problem and find a local minimum (f, C).

Step 3: Keeping f fixed, solve a discrete CA problem.

Step 4: Keeping C fixed, solve a routing problem.

Step 5: If cost D did not decrease between two successive iterations, stop. Otherwise, go to Step 3.

In order to evaluate the effectiveness of the above heuristics, the three latter algorithms were applied to the network shown in Fig. 5, and the results compared. The difference in cost between the best and the worst solution was less than 3 percent; furthermore, two of the three solutions (Top Down and Dis Cap) were identical! Considering the fact that the methods are conceptually very different, the narrow cost range of the solutions implies that they are close to optimum.

Execution time for each of the three algorithms (Top Down, Bottom Up, and Dis Cap) was about 90 seconds on an IBM 360/91.

VII. TOPOLOGICAL DESIGN

A. Problem Formulation

The topological problem can be generally designed as follows:

Given	Requirement matrix R
Minimize	$D(A, C) = \sum_{i \in A} d_i(C_i)$ where the set of arcs A specifies the topology
Such that	a) f is an MC flow satisfying the requirement matrix R
	b) $f \leqslant C$
	c) $T = \frac{1}{\gamma} \sum_{i \in A} \frac{f_i}{C_i - f_i} \leqslant T_{\max}$
	d) The set A must correspond to a 2-connected topology

There exists no efficient technique for the exact solution of this topological problem. Several heuristics, however, have been proposed and implemented and are discussed below.

B. The Branch X-Change (BXC) Method [13], [34]

This method starts from an arbitrary topological configuration and reaches local minima by means of local transformations (a local transformation, often called branch X-change, consists of the elimination of one or more old links and the insertion of one or more new links).

The BXC method has found applications in a variety of topological problems (natural gas pipelines [11], minimum cost survivable networks [34], centralized computer networks [12], etc.). In particular, BXC has been applied to the topological design of distributed computer networks [13]. The algorithm described in [13] is iterative and each iteration consists of three main steps, as follows.

Step 1: Local transformation. A new link is added and an old link is deleted in such a way that two-connectivity is maintained.

Step 2: Capacities and flows are assigned to the new topological configuration using the minimum link assignment described in Section VI, and cost and throughput are evaluated. If there is a cost-throughput improvement, then the topological transformation from Step 1 is accepted. Otherwise, it is rejected.

Step 3: If all local transformations have been explored, stop. Otherwise, go to Step 1.

C. Concave Branch Elimination (CBE) Method [20], [38]

The CBE method can be applied whenever the discrete costs can be reasonably approximated by concave curves [20]. The method consists of starting from a fully connected topology, using concave costs and applying the FD algorithm described in Section VI until a local minimum is reached. Typically, the FD algorithm eliminates uneconomical links, and strongly reduces the topology. Once a locally minimal topology is reached, the discrete capacity solution can be obtained from the continuous solution with the techniques discussed in Section VI. Since two-connectivity is required, the FD algorithm is terminated whenever the next link removal violates this constraint; the last two-connected solution is then assumed to be the local minimum. In order to obtain several local minima, and therefore several different topological solutions, the FD algorithm is applied to several randomly chosen initial flows.

D. Other Methods

Both the BXC and CBE methods have some shortcomings. For example, the BXC method requires an exhaustive exploration of all local topological exchanges and tends to be very time consuming when applied to networks with more than 20 or 30 nodes. The CBE method, on the other hand, can very efficiently eliminate uneconomical links, but does not provide for insertion of new links. In order to overcome such limitations, new methods derived from BXC or CBE have been recently proposed and are now being investigated.

The cut-saturation method, discussed in [22], can be considered as an extension of the BXC method, in the sense that, rather than exhaustively performing all possible branch exchanges, it selects only those exchanges that are likely to improve throughput and cost. In particular, at each iteration: a routing problem is solved; the saturated cut (i.e., the minimal set of most utilized links that, if removed, leaves the network disconnected) is found and a new link is added across the cut; then the least utilized link is removed. The selection of the links to be inserted or removed depends also on link cost.

The concave branch insertion method, discussed in [20], identifies and introduces links which provide cost savings under a concave cost structure. The method can be efficiently combined with the CBE method, to compensate for the inability of the latter to introduce new links.

In some applications with very irregular distributions of node locations, or with constraints which are difficult to formulate analytically (e.g., no chains longer than m hops; connectivity higher than 2, etc.), network design can be greatly enhanced using man-computer interaction. To this end, interactive design programs have been developed in which the network designer can observe (and eventually correct) the topological transformations performed by the computer and displayed iteration after iteration on a graphic terminal.

In general, the selection of the appropriate algorithm will depend on the cost-capacity structure, on the presence of additional topological constraints, on the degree of human interaction allowed and, finally, on the tradeoff between cost and precision required by the particular application.

E. Bounds

In the development and evaluation of topological design algorithms, lower bounds are investigated for the following purposes: 1) appraisal of the accuracy of heuristic algorithms; and 2) development of optimal algorithms based on branch-and-bound type approaches. Lower bounds are obtained by relaxing the topological connectivity constraint and by approximating the discrete cost-capacity curves with lower envelopes (see Fig. 6). Linear lower envelopes lead to *linearized bounds,* while concave envelopes lead to *concave bounds.*

1) Linearized bounds: The linearized bounding problem is generally formulated as a CFA problem (see Section VI) with

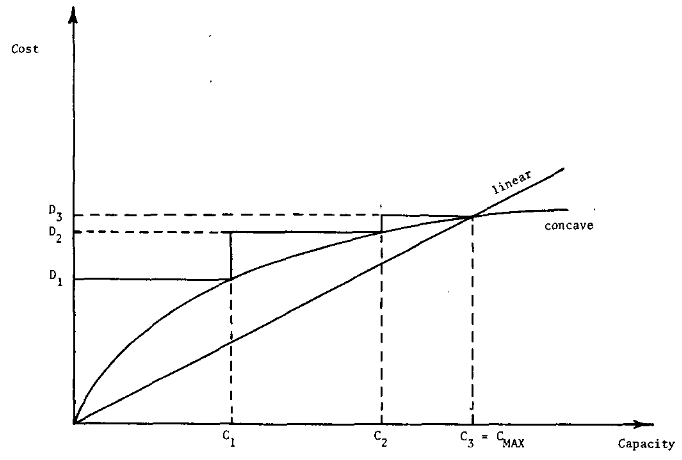


Fig. 6. Linear and concave lower envelope.

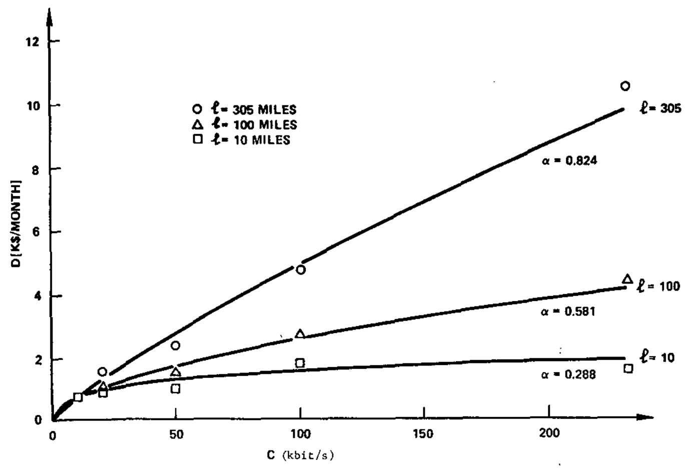


Fig. 7. Concave approximations to link costs, for various link lengths (Telpak tariff assumed).

linear line costs and fully connected topology. The direct solution of the bounding problem is difficult because of the concavity of the objective function $D(f)$ [see (14)]. Rather, the objective $D(f)$ is further bounded as follows:

$$D(f) = \sum d_i f_i + \left(\sum \sqrt{d_i f_i}\right)^2 \Big/ \gamma T_{\mathrm{amx}}$$
$$\geqslant \sum d_i f_i + \left(\sum \sqrt{d_i f_i}\right)^2 \Big/ \gamma T_{\max} C_{\max} = D_{LB}(f) \tag{16}$$

where $C_{\max}$ = max admissible link capacity option.

The lower bound $D_{LB}(f)$ in (16) is *convex*. Thus, a lower bound to the topological problem is obtained by minimizing the convex objective $D_{LB}(f)$ using the FD method.

The procedure as defined above applies to the case in which no link (and link capacity) is preassigned; but can be extended to applications in which a set of links is assigned *a priori*, and new links must be added in order to meet the requirements (e.g., network expansion problem) [39].

The linearized bound can also be applied in branch-and-bound (B-B) algorithms [39]. To this end, recall that at each step of a B-B algorithm a bound is required on the cost of a partially specified topology with a set n_a of assigned links, a set n_p of potential links, and a set n_e of excluded links. This bounding problem is similar to the topological design with some preassigned links, and can be approached with the linearized bounding procedure previously mentioned.

2) Concave bounds: Linearized bounds are simple and exact. However, they are often too loose, especially if line cost versus capacity shows strong economy of scale, or more generally, the cost-capacity structure cannot be accurately bounded with a linear envelope. In such cases, concave bounds lead to better results.

Unfortunately, the presence of concave link costs makes the solution of the bounding problem difficult, since the objective $D(f)$ cannot be expressed in closed form (see Section VI-C). One possible (but complex) approach consists of formulating linear or convex bounds for $D(f)$, and then solving the problem exactly with the FD method. A simpler approach, which we follow, consists of finding an *approximate* solution to the bounding problem with the technique indicated in Section VI-C. The lower bound is then derived from the approximate solution taking into account the accuracy of the solution method. For example, if D is the cost of the approximate solution and ϵ is the relative accuracy, the lower bound is $D_{LB} = D(1 - \epsilon)$.

VIII. APPLICATIONS

We now evaluate the efficiency of some of the heuristic techniques as applied to the topological design of a proposed 26-node ARPANET configuration (see Fig. 5). Capacity options vary from 9.6 to 230.4 kbit/s; discrete cost-capacity functions as well as concave approximations are shown in Fig. 7. Delay requirement is $T_{\max} = 200$ ms. Traffic demands are uniformly distributed between node pairs. Several levels of throughput requirement in the range from 400 to 700 kbit/s are considered.

The suboptimal solutions obtained with different techniques are displayed in a throughput versus cost diagram in Fig. 8. For each technique several solutions were generated at different throughput levels. Since each technique typically generates several locally optimal solutions, only the nondominated solutions were shown in Fig. 8. (Note: Solution A is defined to be dominated by Solution B if B has lower cost and better performance than A.) Figs. 9, 10, and 11 display some typical topologies obtained with BXC, cut-saturation, and CBE, respectively.

From Fig. 8, it is noticed that different techniques lead to solutions which fall in a narrow cost-throughput range. The resulting topological structures, on the other hand, may vary considerably from technique to technique, as can be seen by comparing cut-saturation and CBE solutions in Figs. 10 and 11, respectively. Cost and throughput of the two solutions are approximately the same, but cut-saturation yields about 30 links while CBE yields about 60 links. We note that the marginal cost [dollars/(bit/s)/month] varies over a moderate range for these three procedures.

These facts lead us to conjecture that there are a large number of low-cost solutions which may correspond to very

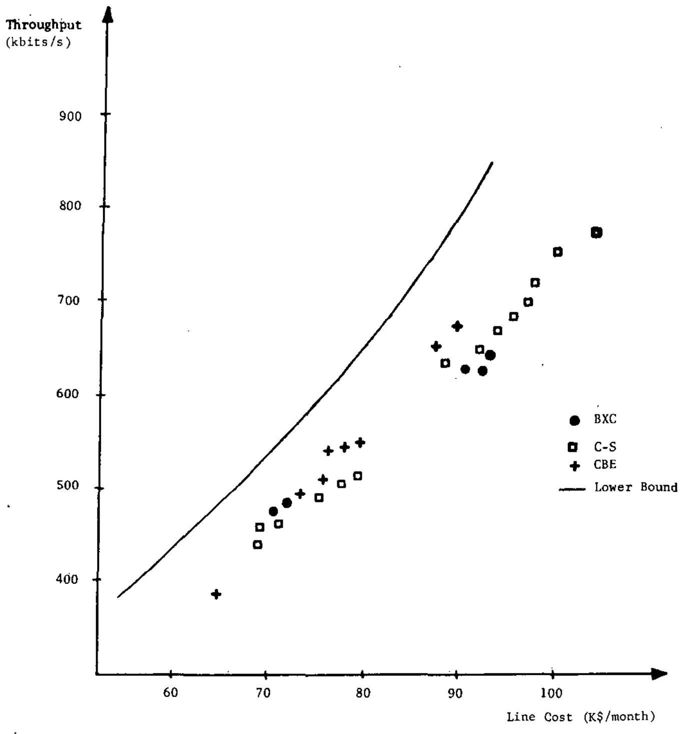


Fig. 8. Heuristic solutions and lower bound for a 26-node network design.

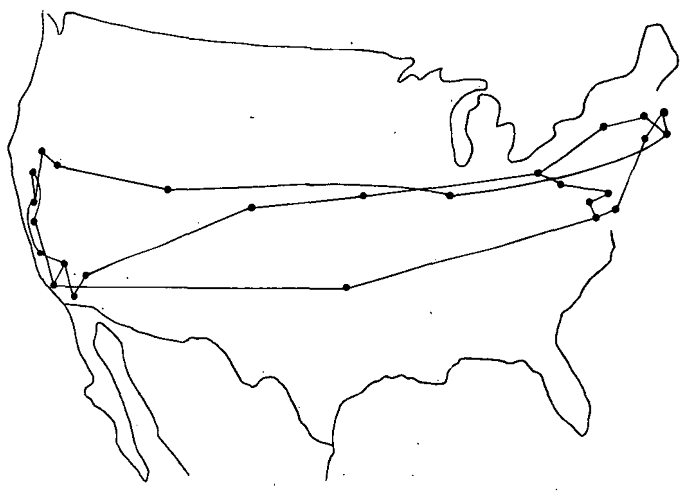


Fig. 9. Example of BXC solution. All links–50 kbits/s; throughput–480 kbits/s; cost–$72 K/month; cost/bit–$0.15/(bit/s)/month.

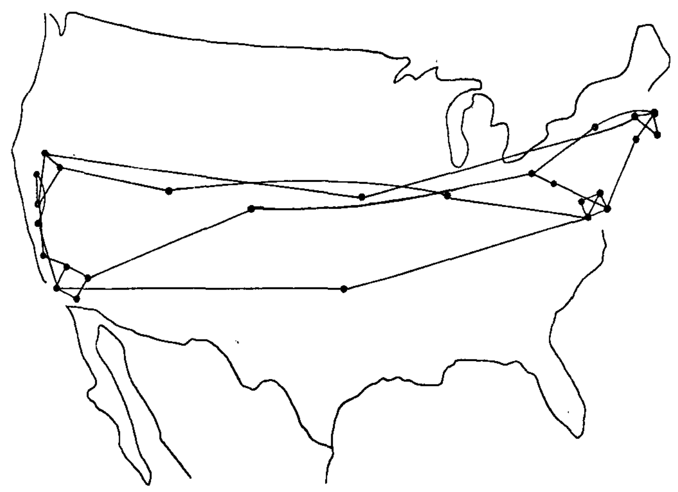


Fig. 10. Example of C-S solution. All links–50 kbits/s; throughput–650 kbits/s; cost–$83 K/month; cost/bit–$0.135/(bit/s)/month.

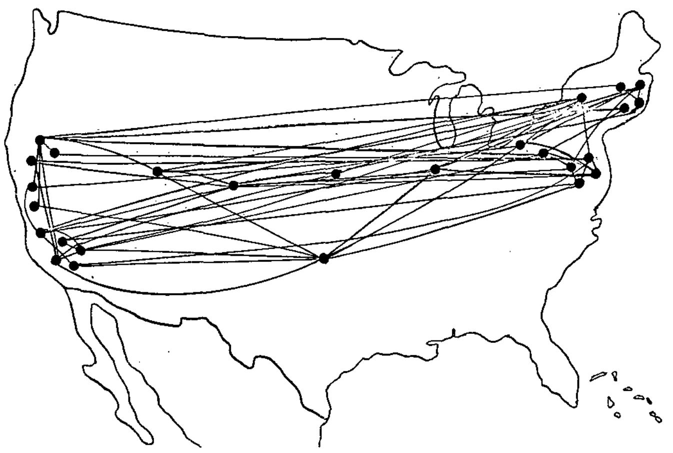


Fig. 11. Example of CBE solution. Line speeds–9.6 and 19.2 kbits/s; throughput–700 kbits/s; cost–$89 K/month; cost/bit–$0.127/(bit/s)/month.

TABLE III
CPU TIME PER NONDOMINATED SOLUTION FOR A 26-NODE NETWORK APPLICATION

BXC	25 seconds
Cut-Saturation	3 seconds
CBE	4 seconds

different topological configurations and capacity assignments. Thus well-constructed heuristics can easily identify good solutions, although the optimal solution may be elusive and extremely difficult to obtain.

In order to evaluate the computational efficiency of the various techniques, we compare in Table III, the CPU times required in the 26-node application. The runs were made on a CDC 6600 computer.

Cut-saturation and CBE methods require comparable amounts of CPU time to generate a nondominated solution while the BXC method is much more time consuming. The experience, based on a variety of topological design problems, has demonstrated that cut-saturation is consistently more efficient than BXC. This result was expected since cut-saturation can be viewed as a refinement of BXC.

As for the near optimality of the heuristics, we conjecture that since suboptimal solutions obtained with different starting topologies and different techniques all fall within a narrow cost-performance range, they are within the same range from the optimal solution as well. From the results in Fig. 8, we may, therefore, conjecture that the accuracy of the heuristics is within 5 percent.

A more rigorous appraisal of near optimality is obtained by comparing suboptimal solutions with lower bounds. For this purpose, we calculated both linearized and concave bounds for the 26-node application.

The linearized bound was used to evaluate the near optimality of the cut-saturation solution shown in Fig. 10 which is constructed exclusively with 50 kbit/s capacity options. To obtain a reasonably tight linearized bound, we generated a large number of low-cost trees (the optimal topology must contain at least a tree) and evaluated the bound on the optimal completion of each tree so as to satisfy the design requirements, following the procedure indicated in Section VII-E1. The lowest cost over all completed trees is \$72K/month, 20 percent below the cut-saturation solution.

To obtain tighter bounds we used the concave bounding approach and approximated the discrete costs with concave curves as shown in Fig. 7. We then solved the associated concave problem for various throughput levels. The resulting bound is represented by the solid curve in Fig. 8. The bound is 15 percent below the cut-saturation solution.

IX. CONCLUSION

In this paper we have surveyed some of the mathematical programming and network flow techniques that have been found useful for the design of distributed computer-communication networks. We reported some exact algorithms for the capacity assignment (CA) and routing problems, and discussed efficient suboptimal techniques that are available for the capacity and flow assignment (CFA) problem. As for the topology design, we showed that good suboptimal solutions can be obtained with the concave branch elimination (CBE) method in the case of smooth (i.e., small economy of scale) cost-capacity characteristics and relatively simple topological constraints; or with the cut-saturation method in those applications that have only a few link capacity options and require some man-machine interaction to verify and satisfy nontrivial topological constraints.

Some bounding techniques were presented, and were used to evaluate the near optimality of the topological design algorithms. The difference between the cost of the suboptimal solutions and the lower bound is typically less than 15 percent. This degree of accuracy is deemed adequate for most topological designs, especially considering that user requirements cannot be predicted with much accuracy before network implementation, or tend to change during the life of the network.

Although the present algorithms seem adequate for most applications, there is a need for better bounds and (possibly) exact topological design solutions to measure and compare the near optimality of the various techniques. More work should be devoted also to the analysis of algorithm complexity (expected and worst-case time and storage requirements) to have a better concept of the applicability and limits of the various techniques, and to identify the areas where research effort should be expended [48].

So far, we have discussed and solved a rather simple network design problem, with fixed switch locations, pure terrestrial topology, average delay constraints, etc. More general design problems can be formulated, requiring for example, the selection of backbone switch number and location; the use of both terrestrial and satellite facilities; the consideration of different traffic applications (e.g., bulk transfers, interactive traffic, digitized voice) with different priorities and performance requirements; the use of more general performance constraints (e.g., maximum end-to-end delay instead of average delay); the integration of circuit and packet-switching techniques, etc. Further work is required to extend the existing algorithms to handle the more general cases. Some preliminary results in this direction are reported in [41] for the mixed terrestrial and satellite design and in [42] for the selection of backbone nodes.

REFERENCES

[1] "Analysis and optimization of store-and-forward computer networks," NAC 4th Semiannu. Tech. Rep. for the ARPA Project, Defense Documentation Center, Dec. 1971.

[2] L. R. Bahl and D. T. Tang, "Optimization of concentrator locations in teleprocessing networks," presented at Symp. Computer-Communications Networks and Teletraffic, Polytechnic Inst. Brooklyn, Brooklyn, NY, Apr. 4-6, 1972.

[3] S. Carr, S. Crocker, and V. Cerf, "HOST-HOST communication protocol in the ARPA network," in *Conf. Rec., 1970 Spring Joint Computer Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970, pp. 589-597.

[4] W. Chou and A. Kershenbaum, "A unified algorithm for designing multidrop teleprocessing networks," in *Data Networks Analysis and Design, 3rd Data Communication Symp.*, Nov. 1973, pp. 148-156.

[5] W. Chou and H. Frank, "Routing strategies for computer network design," presented at Symp. Computer-Communications Networks and Teletraffic, Polytechnic Inst. of Brooklyn, Brooklyn, NY, Apr. 4-5, 1972.

[6] G. C. Cole, "Computer network measurements: Techniques and experiments," School of Eng. and Appl. Sci., Univ. of California, Los Angeles, rep. UCLA-ENG-7165, 1971.

[7] S. Crocker, J. Haefner, R. Metcalfe, and J. Postel, "Function-oriented protocols for the ARPA computer network," in *Conf. Rec., 1972 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 40. Montvale, NJ: AFIPS Press, 1972, pp. 271-279.

[8] G. B. Danzig, *Linear Programming and Extensions.* Princeton, NJ: Princeton Univ. Press, 1963.

[9] H. Everett, III "Generalized Lagrange multipliers method for solving problems of optimal allocation of resources," *Operations Res.*, vol. II, pp. 399-418, 1963.

[10] B. Fox, "Discrete optimization via marginal analysis," *Management Sci.*, vol. 13, pp. 210-216, Nov. 1966.

[11] H. Frank, B. Rothfarb, D. Kleitman, and K. Steiglitz, "Design of economical offshore natural gas pipeline networks," Office of Emergency Preparedness, Washington, DC, rep. R-1, Jan. 1969.

[12] H. Frank, I. T. Frisch, W. Chou, and R. Van Slyke, "Optimal design of centralized computer networks," *Networks*, vol. 1, pp. 43-57, 1971.

[13] H. Frank, I. T. Frisch, and W. Chou, "Topological considerations in the design of the ARPA computer network," in *Conf. Rec., 1970 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970.

[14] H. Frank, M. Gerla, and W. Chou, "Issues in the design of large distributed networks," presented at the IEEE Nat. Telecommun. Conf., Atlanta, GA, Nov. 1973.

[15] H. Frank, R. E. Kahn, and L. Kleinrock, "Computer communication network design–Experience with theory and practice," in *Conf. Rec., 1972 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 40. Montvale, NJ: AFIPS Press, 1972, pp. 255-270.

[16] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward computer-communication network design," *Networks*, vol. 3, pp. 97-133.

[17] G. L. Fultz and L. Kleinrock, "Adaptive routing techniques for store-and-forward computer communication networks," presented at IEEE Int. Conf. Commun., Montreal, Canada, June 14-16, 1971.

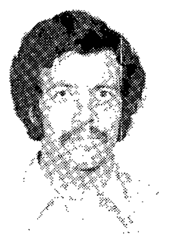
[18] M. Gerla, "Deterministic and adaptive routing policies in packet-switched computer networks," presented at the ACM-IEEE 3rd Data Commun. Symp., Tampa, FL, Nov. 13-15, 1973.

[19] M. Gerla, H. Frank, and W. Chou, "Computational considerations and routing problems for large computer networks," presented at IEEE Nat. Telecommun. Conf., Atlanta, GA, Nov. 1973.

[20] M. Gerla, "The design of store-and-forward networks for computer communications," Ph.D. dissertation, School of Eng. and Appl Sci., Univ. of California, Los Angeles, Jan. 1973.

[21] F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The interface message process for the ARPA computer network," in *Conf. Rec., 1970 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970, pp. 551-567.
[22] "Issues on large network design," Network Analysis Corp., Glen Cove, NY, ARPA rep., Jan. 1974.
[23] J. R. Jackson, "Networks of waiting lines," *Operations Res.*, vol. 5, pp. 518-521, 1957.
[24] R. E. Kahn and W. R. Crowther, "A study of the ARPA computer network design and performance," Bolt, Beranek, and Newman, Inc., rep. 2161, Aug. 1971.
[25] L. Kleinrock, "Analytic and simulation methods in computer network design," in *Conf. Rec., Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970, pp. 568-579.
[26] —, "Models for computer networks," in *Proc., Int. Conf. Commun.*, Boulder, CO, June 1969, pp. 21-9-21-16.
[27] —, *Communication Nets: Stochastic Message Flow and Delay*. New York: McGraw-Hill, 1964.
[28] —, *Queueing Systems: Volume I, Theory*. New York, Wiley-Interscience, 1975.
[29] —, *Queueing Systems: Volume II, Computer Applications*. New York: Wiley-Interscience, 1976.
[30] —, "Resource allocation in computer systems and computer-communication networks," presented at IFIP Cong., Aug. 1974.
[31] L. Kleinrock and W. Naylor, "On measured behavior of the ARPA network," in *Conf. Rec., Nat. Comput. Conf., AFIPS Conf. Proc.*, vol. 43. Montvale, NJ: AFIPS Press, 1974, pp. 767-780.
[32] L. G. Roberts and B. D. Wessler, "Computer network development to acheive resource sharing," in *Conf. Rec., 1970 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, pp. 543-599.
[33] D. J. Silk, "Routing doctrines and their implementation in message switching networks," *Proc. Inst. Elec. Eng.*, vol. 116, pp. 1631-1638, Oct. 1969.
[34] K. Steiglitz, P. Weiner, and D. J. Kleitman, "The design of minimum cost survivable networks," *IEEE Trans. Circuit Theory*, pp. 455-460, Nov. 1969.
[35] J. A. Tomlin, "Minimum cost multi-commodity network flows," *Operations Res.*, vol. 14, pp. 45-47, Jan. 1966.
[36] R. Van Slyke and H. Frank, "Network reliability analysis: Part I," *Networks*, vol. 1, pp. 279-290, 1971.
[37] V. K. M. Whitney, "Lagrangian optimization of stochastic communication systems models," presented at MRI Symp. Comput. Commun. Networks, Brooklyn, NY, Apr. 1972.
[38] B. Yaged, Jr., "Minimum cost routing for static network models," *Networks*, vol. 1, pp. 139-172, 1971.
[39] "The practical impact of recent computer advances on the analysis and design of large scale networks," Network Analysis Corp., Glen Cove, NY, 3rd Semiannu. Tech. Rep., Defense Documentation Center, June 1974.
[40] M. Gerla, H. Frank, W. Chou, and J. Eckl, "A cut-saturation algorithm for topological design of packet-switched communications networks," in *Proc. Nat. Telecommun. Conf.*, San Diego, CA, Dec. 2-4, 1974.
[41] M. Gerla, W. Chou, and H. Frank, "Cost-throughput trends in computer networks using satellite communications," in *Proc. Int. Conf. Commun.*, Minneapolis, MN, 1974.
[42] W. Hsieh, M. Gerla, P. McGregor, and J. Eckl, "Locating backbone switches in a large packet network," presented at Int. Conf. Comput. Commun., Montreal, Canada, 1976.
[43] L. Kleinrock and H. Opderbeck, "Throughput in the ARPANET—Protocols and measurement," in *Conf. Rec., Network Structures in an Evolving Operational Environment, 4th Data Commun. Symp.*, Quebec City, Canada, Oct. 1975, pp. 6-1-6-11.
[44] —, "The influence of control procedures on the performance of packet-switched networks," in *Proc. Nat. Telemetering Conf.*, Dec. 1974.
[45] L. Kleinrock and F. Kamoun, "Data communications through large packet-switching networks," presented at 8th Int. Teletraffic Cong., Australia, Nov. 1976.
[46] L. Kleinrock, W. E. Naylor, and H. Opderbeck, "A study of line overhead in the ARPANET," *Commun. Ass. Comput. Mach.*, vol. 19, pp. 3-12, Jan. 1976; also in *Proc. IIASA Conf. Comput. Commun. Networks*, Oct. 1974, pp. 87-109.
[47] L. Kleinrock, "ARPANET lessons," in *Conf. Rec. IEEE Int. Conf. Commun.*, Philadelphia, PA, June 14-16, 1976, pp. 20-1-20-6.
[48] A. Aho, J. Hopcraft, and J. Ullman, *Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.

★



Mario Gerla (M'75) received the doctorate in engineering from the Politecnico di Milano, Milan, Italy, in 1966 and the Ph.D. degree in engineering from the University of California, Los Angeles, in 1973.

From 1966 to 1968, as an officer in the Italian Navy, he participated in a NATO program on Tactical Satellite Communications. In 1969 he was with the Advanced Systems Section of GTE, Milan, and was involved with research and development of low-noise amplifiers for satellite applications. At the end of 1969, he came with a Fulbright scholarship to the University of California, Los Angeles, and participated in data network design under the ARPA project. From January 1973 to May 1976, he was with Network Analysis Corporation, Glen Cove, NY, where as Manager of Computer Networking, he was involved in several computer network design projects for both government and industry. At present, he has a joint appointment with the University of California, Los Angeles, where he performs research on satellite multiple-access techniques under the ARPA project, and with Computer Transmission Corporation, Los Angeles, CA where he is responsible for research and planning of advanced data communication techniques.

★

Leonard Kleinrock (S'55-M'64-SM'71-F'73) received the B.E.E. degree from City College of New York, New York, NY in 1957 and the M.S.E.E. and Ph.D.E.E. degrees from Massachusetts Institute of Technology, Cambridge, in 1959 and 1963, respectively.

He was an Assistant Engineer at the Photobell Co. in New York from 1951 to 1957 and a Staff Associate at the MIT Lincoln Laboratory from 1957-1963. In 1963 he joined the faculty of the School of Engineering and Applied Science, University of California, Los Angeles, where he is now Professor of Computer Science. His research spans the fields of computer networks, computer systems modeling and analysis, queueing theory and resource sharing, and allocation in general. At UCLA, he directs a large group in advanced teleprocessing systems and computer networks. He is the author of three major books in the field of computer networks: *Communication Nets: Stochastic Message Flow and Delay* (New York: McGraw Hill, 1964; also New York: Dover, 1972); *Queueing Systems, Vol. I: Theory* (New York: Wiley-Interscience, 1975); and *Queueing Systems, Vol. II: Computer Applications* (New York: Wiley-Interscience, 1976). He has published over 70 articles and contributed to several books. He serves as consultant for many domestic and foreign corporations and governments and he is a referee for numerous scholarly publications and a book reviewer for several publishers.

Dr. Kleinrock was awarded a Guggenheim Fellowship for 1971-1972 and is an IEEE Fellow "for contributions in computer-communication networks, queueing theory, time-shared systems, and engineering education."