K. Maruyama
L. Fratta
D. T. Tang

Heuristic Design Algorithm for Computer Communication Networks with Different Classes of Packets

Abstract: A typical operating environment of a packet switching (store-and-forward) computer communication network is that it is shared by many users with different classes of packets. Packets may be classified in a very general fashion by types of users, messages, applications, transactions, response time requirements, packet parameters such as packet rate and length, and by network parameters such as source-destination and path length. A well-designed network must provide access and performance assurance to all packet classes. This paper presents a heuristic algorithm for designing such a communication network. The algorithm presented contains heuristic algorithms for discrete link capacity assignment, priority assignment, and flow assignment problems with an additional feature which allows one to alter network topology interactively. Sample results from applications of the overall network design are also given.

Introduction

Since 1969, when the ARPANET Project emerged as the first major experimental store-and-forward packet switching network, many other computer communication networks have been built and are now in operation [1–4]. During this time, a great deal of research effort has been devoted to studying store-and-forward packet switching networks from various points of view including application environment, protocols, flow control, and performance analysis [5–10]. Although the technology of communication networks is still evolving, the basic notion of packet switching is certainly quite cost-effective over a wide range of computer communication network applications.

To evaluate the effectiveness of a communication network, two cost-performance parameters have been widely used [5]: the *total cost* of the network D, and the *average packet delay* Z, which is the average time experienced by a packet traveling from source to destination. With those two parameters, a general network design problem may be formulated to determine optimum values of variables such as topology, routing, link capacities, etc., while either fixing cost D and minimizing average delay Z, or minimizing cost D and satisfying certain bounds in average delay Z. Because of the complexity of this general design problem, further complicated by the interdependency among different design variables, no exact solution is yet available. Most research in this area has been devoted to solving subproblems.

There are at least four basic design subproblems [6]: the flow assignment (FA), the capacity assignment (CA), the capacity and flow assignments (CFA) and the topology, capacity and flow assignments (TCFA).

The FA problem requires one to minimize the nonlinear function Z with respect to the flows in each link while satisfying the external flow requirements. Computationally efficient algorithms for the solution of this problem exist [11, 12].

Solutions (either in closed form or as algorithms) to the CA problem are available in the literature with various degrees of completeness and computational complexities. Closed-form solutions exist for linear or logarithmic cost-capacity functions [5, 13, 14], while computationally efficient algorithms for exact or suboptimal solutions are available for concave cost-capacity functions [13] and discrete cost-capacity functions [15–18].

Solutions to the CFA problem may be obtained by combining solutions to CA and FA problems. For nonlinear cost-capacity functions, no globally optimal algorithms exist. However, heuristic algorithms are available [18].

Because of the complexity of the TCFA problem, there exist only approximate solution techniques, which iteratively apply an algorithm for the CFA problem to change, at each iteration, the network topology by eliminating and/or inserting links according to some policies [18].

The common characteristic of all of these design algorithms for the subproblems is that they evaluate the performance of a network for a single class of packets with respect to cost-performance parameters such as the network cost D and/or the average packet delay Z. In a practical communication network, it may be necessary to distinguish among different classes of packets. Packets may be classified in a very general fashion according to the types of users, applications, messages, response time requirements, packet parameters (such as packet rate and length), and network parameters (source destination, path length, etc.). For example, it is natural for the delay requirement for packets created by interactive computation to be different from the requirement for packets created by batch computation. Another aspect of the shared networks which may require class distinction among packets is the different geographical distribution of network traffic. This may become an important consideration when different networks are interconnected, for instance. The problem of distinct packet classes and requirements was first studied by Maruyama and Tang [17] in their capacity assignment algorithm; it was later extended to include another design variable which explicitly takes into account the differences among classes of packets, namely, the priority assignment among different classes of packets, for further network cost optimization [19].

The main goal of this paper is to present a heuristic algorithm for the solution of the capacity, priority and flow assignment (CPFA) problem. We define the CPFA problem in the second section. In the third section, we first give a brief description of an overall design algorithm for the CPFA problem and then describe algorithms for three basic design subproblems: capacity assignment CA, priority assignment PA, and flow assignment FA. Some numerical results are reported in the fourth section.

CPFA problem

The CPFA problem which is dealt with in this paper may be defined as follows:

Given

- traffic requirement matrices for all packet classes,
- network topology with node locations, and
- delay constraints B_k for each packet class,

minimize

$D = \Sigma_j d_j(C_j)$ and then $G = g(Z_k, B_k, \gamma_k)$

with respect to

- link capacities C_j,
- priority levels P_k, and
- flows (or routing),

under constraints

$Z_k \leq B_k$ for each packet class,

where

C_j is the jth link capacity selected from a finite set of options,
$d_j(C_j)$ is the discrete cost-capacity function of link j,
Z_k is the average delay for packet class k,
B_k is the upper bound on the average packet delay Z_k,
γ_k is the rate of packet class k entering the network,
P_k is the priority level assigned to packet class k,
D is the total network cost, the primary objective function, and
g is a secondary performance objective function discussed in the third section.

Z_k may be computed from

$$Z_k = (\Sigma_j \lambda_{jk} T_{jk})/\gamma_k + (\Sigma_s \theta_{sk} t_{sk})/\gamma_k,$$

where

λ_{jk} is the rate of packet class k on the jth link,
T_{jk} is the delay of packet class k on the jth link,
θ_{sk} is the rate of packet class k entering node s, and
t_{sk} is the average nodal processing delay, for packet class k at node s.

In the computation of T_{jk}, we assume Poisson packet arrivals at each node and the exponential distribution of packet length with the well-known independence assumption [5], which allows us to use the standard formula for an M/M/1 nonpreemptive priority (head-of-the-line) queuing system [20, 21]. T_{jk} is the sum of the mean queuing delay on link j for packets in the rth priority level (the kth packet class is assumed to be in the rth priority level), the propagation delay and the mean service time of class k packets. (If necessary, however, more accurate models may be used to compute T_{jk}.) In practice, the nodal processing delay may be a small portion of the path delay and can often be accounted for by a constant [6].

Description of algorithm

The heuristic algorithm for solving the CPFA problem, referred to as Algorithm CPFA, consists mainly of three algorithms which handle the following subproblems: discrete link capacity assignment CA, priority assignment PA and flow assignment FA. In this section, we first describe the overall composite Algorithm CPFA and then briefly discuss algorithms for capacity, priority, and flow assignment problems.

A simplified flow chart of Algorithm CPFA is illustrated in Fig. 1. CPA denotes the discrete link capacity

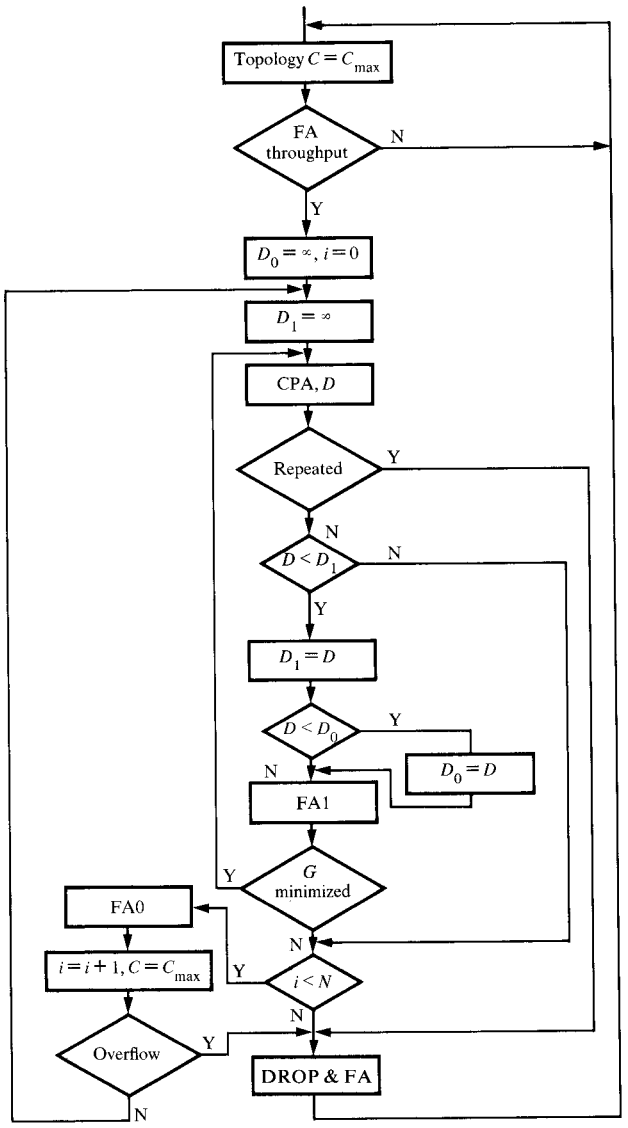

Figure 1 Algorithm CPFA.

and priority assignment algorithm; FA, FA0, FA1 and DROP&FA denote different flow assignment algorithms. Algorithm CPFA works as follows: For a given network topology, it first assigns the maximum available capacity to each link and checks for feasibility (any link flow must be smaller than the corresponding link capacity). This is performed by FA by rerouting packets until the throughput requirements are satisfied by the network (in this process, delay constraints $Z_k \leq B_k$ for all k are ignored). If no such flow assignment exists, the design problem is infeasible and the initial topology must be changed; otherwise, the optimization procedure starts. Algorithm CPA obtains the best discrete link capacity and priority assignments. If the solution given by Algorithm CPA has not been found previously and the network cost D is reduced, then the procedure FA1 will be used. This flow assignment algorithm minimizes a given performance objective function G while satisfying all of the given delay constraints. CPA and FA1 are then repeated until a local optimum solution is obtained. In order to cover a wider solution space and to find a better solution, N different initial flow assignments are obtained by Algorithm FA0, which reroutes packets while ignoring delay constraints. The final refinement of the solution is carried out by Algorithm DROP&FA, which decreases some link capacities and then tries to satisfy the delay constraints by rerouting packets.

Algorithm CPFA can be iteratively applied to solve the topology, capacity, priority and flow assignment (TCPFA) problem in a manner similar to that of the Concave Branch Elimination Method [18]. In the present algorithm, because of the PL/I implementation under CMS, Algorithm CPFA can be run interactively with a simple editing feature to delete and/or add links to the topology. One may use information such as link utilizations and average path lengths obtained from the previous design to make such changes.

• *Capacity assignment algorithm*

We assume that for each link in the network a finite set of discrete link capacities is available, and it may vary from link to link. Thus the capacity assignment problem we are dealing with consists of selecting a capacity for each link in the network from the corresponding finite set of available capacities in order to minimize the total network cost, after assuming that both flow and priority assignments are given for each class of packets. The constraints that must be satisfied are the given bounds on the average packet delays for all packet classes.

The discrete link capacity assignment algorithm [19], Algorithm CA, presented here is a composition of several heuristics and is illustrated in Fig. 2. Each heuristic is used in a particular phase of the cost optimization process. The four heuristics ADD, ADDFAST, DROP, and DROPFAST are the basic capacity assignment procedures, and each of them uses a different figure of merit for cost optimization. The heuristic ADD first computes for each link a figure of merit F, a function of path delays, bounds, traffic rates, and link cost, which represents an "overall" performance gain per unit cost. Then it increases the current capacity to the next available higher level in the link with the maximum F. This process is repeated until no further performance gain is attainable by increasing any link capacity. The heuristic ADDFAST does essentially the same things as ADD except that the figure of merit used is a function of link delay, link traffic, and link cost. To reduce computing time, this is computed only for those links that belong to the packet class with

the highest delay-bound ratio. The heuristics DROP and DROPFAST are duals to ADD and ADDFAST, respectively. These four basic capacity assignment heuristics seek a better solution by either increasing or decreasing a link capacity at each iteration. Another heuristic that often leads to further improvements on the solutions given by the above basic heuristics is EXC, which will seek to upgrade a link capacity and degrade another to further reduce network cost while satisfying all delay constraints. Procedures SETLOW and SETHIGH are used in the initial phase of the optimization process to check for problem feasibility and provide an initial capacity assignment to ADD (or ADDFAST) and DROP (or DROPFAST), respectively. Further details of these heuristics may be found in Reference [19].

We have observed that a solution obtained by any one of the capacity assignment heuristics can often be improved by running others consecutively. After experimenting with these heuristics on a number of different capacity assignment problems, we have found that a simple composition (SETLOW-ADDFAST-DROP-EXC or SETHIGH-DROP-EXC) as shown in the top part of Algorithm CA in Fig. 2 yields fairly good solutions in most cases. Another observation which we have made, especially on larger networks (over 15 nodes), is that further improvement on the solution can sometimes be obtained by adding the ADD (or ADDFAST) and DROP (or DROPFAST) heuristics, though such improvement was mostly observed to be less than five percent. To allow such alternations between two basic heuristics, interfaces RESETLOW and RESETHIGH are provided. The complete composite Algorithm CA is shown in Fig. 2.

Comparisons have been made between the solutions given by Algorithm CA and the optimum solutions given by a branch-and-bound algorithm. Because of the computational complexity involved in the latter algorithm, only solutions for small networks, up to six nodes, have been examined. In all these examples the heuristic algorithm gave the optimum solutions. For larger networks, we were not able to improve the CA Algorithm solution by extensive but incomplete runs of the branch-and-bound algorithm. The CA algorithm thus appears to produce near-optimal solutions for larger networks.

The computing time for Algorithm CA is very small compared to that required by the FA algorithm described later in the section entitled "Flow assignment algorithm." This allows one to run Algorithm CA hundreds of times while solving the priority assignment problem discussed next.

In Algorithm CA, we have also provided a procedure to estimate the nodal buffer requirement at each node from an M/M/1/k queuing model [20, 21] to ensure that the nodal buffer blocking probability does not exceed a certain given value.

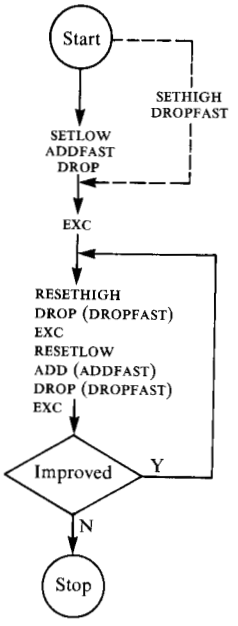

Figure 2 Algorithm CA.

• *Priority assignment algorithm*

When we design a communication network which is shared by different classes of packets as defined in the second section, it is often possible to take advantage of different characteristics and requirements among different classes of packets, while minimizing the total network cost. One approach is to use a simple priority scheme which allows different treatments on different classes of packets. We shall now consider the problem of priority assignment on different classes of packets (mapping of n packet classes into r priority levels), and describe a heuristic algorithm for the priority assignment. Although it is possible to consider different priority assignment on classes of packets "at each node," such priority assignments are not desirable from the system point of view and are not considered here.

Let n be the number of different classes of packets that share the communication network. The number of possible different priority assignments $N(n)$ is given by the following formula:

$$N(n) = \sum_{r=1}^{n} r! \left\{ {n \atop r} \right\},$$

where

$$\left\{ {n \atop r} \right\} = \left\{ {n-1 \atop r-1} \right\} + r \left\{ {n-1 \atop r} \right\},$$

and

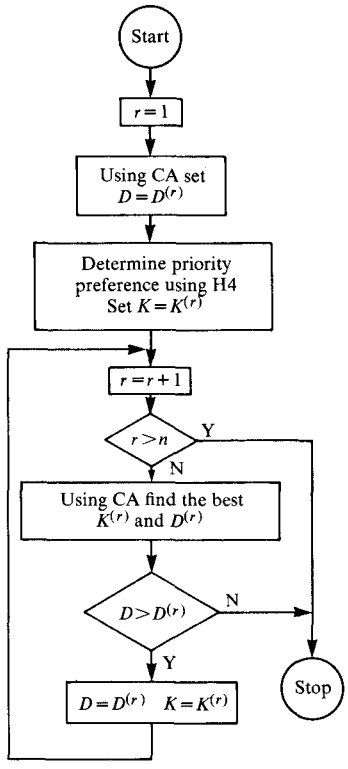

Figure 3 Algorithm CPA.

$$\left\{ {n \atop n} \right\} = \left\{ {n \atop 1} \right\} = 1,$$

and it is often referred to as one of the Stirling numbers [22]. The number of different priority assignments $N(n)$ grows very rapidly as the number of packet classes n increases. For example, $N(2) = 3$, $N(3) = 14$, $N(4) = 75$, $N(5) = 541$ and $N(6) = 4683$. Since it is clearly impractical to solve the CA problem $N(n)$ times for a CPA solution for n greater than 5, we seek to reduce the number of priority assignments examined.

Our heuristic procedure consists of the following two steps:

Step 1 Order packet classes according to priority preference.

Step 2 Determine the number of priority levels.

In Step 1, a figure of merit is used to order packet classes into a sequence $K^{(1)}$ in nonincreasing priority preference values. The figure of merit takes into account the most relevant parameters for the priority assignment, such as delay requirements B_k, average packet lengths $1/\mu_k$, packet rates γ_k and the average path lengths l_k. (Due to different traffic requirements among pairs of nodes, geographical distribution and others, the average path lengths may vary class by class. The average path length [5, 6] for packet class k, l_k, may be computed from $l_k = (\Sigma_j \lambda_{jk})/\gamma_k$.) After performing experiments on different types of figures of merit as functions of these four parameters, we have arrived at the following heuristic which we refer to as H4: Between two packet classes k and h,

$$\text{if } (B_k - Z_k)/l_k < (B_h - Z_h)/l_h,$$

then

$$PR(k) \geq PR(h).$$

Z_k and Z_h are the average packet delays for packet classes k and h, respectively, and $PR(k)$ denotes the priority preference value for class k. H4 quantifies the condition when it may be preferable to assign a priority to a packet class k that is higher than or equal to that of a packet class h.

Once the priority preference sequence $K^{(1)}$ has been determined in Step 1, the next step is to determine the best number r of partitions $K^{(r)}$, $1 \leq r \leq n$, corresponding to distinct priority levels. Since there are 2^{n-1} different partitions on a sequence of n elements, it is still impractical in most cases to exhaust all possible priority assignments to find the optimal priority assignment under the determined priority preference sequence. The heuristic used in Step 2 is a sequential partition strategy which examines at most $[n(n-1)/2] + 1$ different partitions [19]. With this strategy one establishes partition boundaries one by one until no further cost reduction can be achieved.

In Fig. 3, we show a simplified flow chart of Algorithm CPA for the problem of both capacity and priority assignments. Further details on the implementation of the algorithm can be found in [19].

• *Flow assignment algorithm*

The flow assignment problem in store-and-forward packet switching communication networks consists of finding, within a fixed capacity assignment, a set of decision rules which specifies the deterministic (or static) routing tables according to which packets should be routed in order to optimize the secondary objective functions. There are several mathematical programming techniques available [23, 24] for the flow assignment problem, which is formulated as a nonlinear multicommodity flow assignment problem [25]. However, these are typically computationally time-consuming and inappropriate, especially when the design solution requires many iterations of flow assignment computations. Consequently, a considerable amount of research effort has been spent in developing heuristic suboptimal algorithms [11, 12, 18, 25–28]. None of these, however, is capable of dealing with many classes of packets, which is the objective of the present section.

A common characteristic of all the existing flow assignment algorithms is that they usually attempt to as-

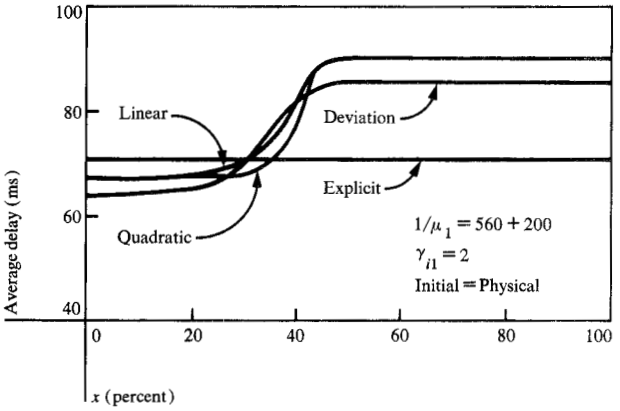

Figure 4 Model behavior with initial flow assignment using *physical* link distances.

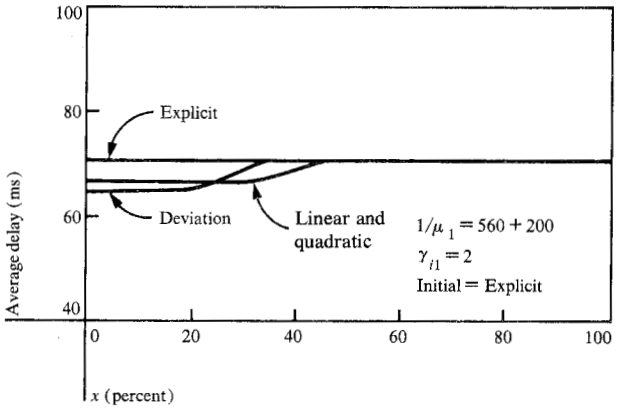

Figure 5 Model behavior with initial flow assignment using *explicit* link distances.

sign flows along the "shortest" path available. The shortest path computation is based upon "link distances" where a link distance w_j is a measure related to the packet delay in that link. In order to take different packet classes into account, we wish to compute w_{jk}, the distance of link j for the kth class packets under a fixed priority assignment. Note that once link distances are computed, flows may be fully or partially rerouted. This necessitates an update on link distances which completes a typical iteration of the flow assignment algorithm.

Flow assignment algorithms may give different results depending on the ways in which link distance is defined. In our flow assignment algorithms, we have considered the following two link distance models for computing w_{jk}:

$$w_{jk} = p_j + 1/\mu_k C_j, \text{ and} \tag{1}$$

$$w_{jk} = T_{jk} + \beta\lambda_{jk}\frac{\partial T_{jk}}{\partial \lambda_{jk}}. \tag{2}$$

The first model, which we refer to as the "explicit" link distance model, considers only the propagation delay p_j and the service time $1/\mu_k C_j$, where $1/\mu_k$ is the kth class average packet length and C_j is the current jth link capacity. The second model, which we refer to as the "selective" link distance model, considers both the current link delay T_{jk} and a nonlinear term which compensates w_{jk}, where β is a constant and λ_{jk} is the kth class packet rate on link j.

It can easily be seen that the selective link distance model is related to other link distance models which appear in the literature. When $\beta = 0$, it contains only the current link delay T_{jk}. This is known as the linear model [26]. When $\beta = \frac{1}{2}$ or 1, it is a generalization (in the case of many packet classes) of, respectively, the Quadratic Routing [26] or Flow Deviation Algorithm [11].

Again, we have experimented with these link distance models as well as with others [29] and observed that their performance was sensitive to the percentage x of the number of paths whose traffic is rerouted to shortest paths after each shortest distance update. This behavior can be observed from Figs. 4 and 5, in which the typical average delay curves are plotted as functions of x. In Fig. 4, the physical link distance (i.e., $w_{jk} = p_j$) was used for the initial flow assignment, and in Fig. 5, the explicit link distance was used. This led us to implement the following link distance model, which we refer to as the "adjusted" link distance model:

$$w_{jk} = T_{jk} + \beta(x)\lambda_{jk}\frac{\partial T_{jk}}{\partial \lambda_{jk}}, \tag{3}$$

where β is a monotonically nonincreasing function of x such that $\beta(100) = 0$ and $\beta(0) = 1$. Here the explicit link distance model is used for the initial flow assignment. Such an algorithm will automatically choose the most appropriate link distance measurement with respect to the given value of x and will find a flow assignment which corresponds to the average delay on the lower envelope of delay curves in Figs. 4 and 5.

The solution of the flow assignment problem is searched under given capacity and priority assignments. Thus, the primary objective function, total network cost, cannot be directly reduced in this process. For networks with a single class of packets, the common secondary objective function to be minimized is average packet delay [5]. However, when several classes of packets exist in the network with different delay bounds and different average packet delays, it is not clear what kind of secondary objective function should be used. Any selected secondary performance objective function should tend to result in a less costly network when the Algorithm CPA is applied again. There are many ways

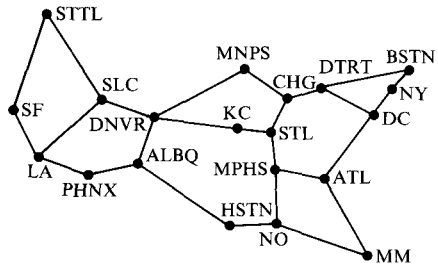

Figure 6 Mesh network.

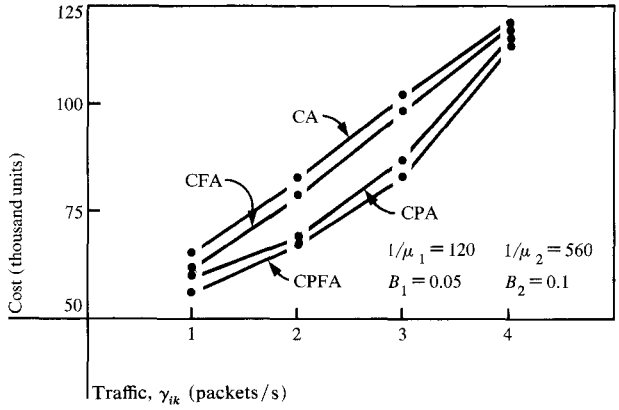

Figure 7 Behavior of different algorithms.

to define such an objective function, and we have considered and tested the functions of the form $G = g(Z_k, B_k, \gamma_k, k = 1, \cdots, n)$, where γ_k is the total traffic rate of class k packets entering the network and $\gamma = \Sigma_k \gamma_k$. We have observed that the following function,

$$G = (\Sigma_k \gamma_k Z_k \, (Z_k / B_k)) / \gamma, \tag{4}$$

where $Z_k \leq B_k$ for all k, which represents the sum of all the average packet delays weighted by the allowed delay that has actually taken place and the amount of traffic, gives the best performance among those we have considered [29].

As a concluding remark on Algorithm FA, we note that the computation time it requires depends mainly on the number of times shortest paths must be computed, and that this number is a function of both the number of packet classes in the network and the algorithm parameter x, the percentage of the number of paths whose traffic is rerouted after each pass of shortest distance calculation.

Numerical examples

Algorithm CPFA, described in the previous section, has been coded in PL/I and run on an IBM 370/168.

The numerical results discussed in this section refer to the design of the 20-node, 26-link network shown in Fig. 6. For each link in the network, the available link capacities and corresponding monthly costs are shown in Table 1. The average link delays for different classes of packets are computed using the standard formula for an M/M/1 nonpreemptive priority queuing system [20, 21], assuming Poisson packet arrivals and the exponential distribution of packet lengths with mean $1/\mu_k$. Also assumed is a one-ms nodal processing delay [6], i.e., $t_{sk} = 0.001$ s/packet for all s and k.

In order to test the effectiveness of Algorithm CPFA, four algorithms CA, CPA, CFA, and CPFA were experimentally tested in a number of different design examples, and results were compared. In the following only a limited number of design examples are reported.

Table 2 shows network costs and packet delays obtained by the four algorithms applied to the design case of two packet classes, where $\gamma_{i1} = 3$, $\gamma_{i2} = 3$, $1/\mu_1 = 120$, $1/\mu_2 = 560$, $B_1 = 0.5$, and $B_2 = 0.1$. It can be seen that Algorithm CPFA efficiently designs the network and achieves a significant cost reduction from about 103000 units to about 85000 units. It can be observed also, as expected, that as the network cost is reduced, the average path delays Z_1 and Z_2 are forced to approach the corresponding bounds B_1 and B_2, respectively. Further experimental results are shown in Fig. 7 in which network costs vs traffic requirements are plotted for the same two packet classes.

Table 3 shows the results of network design obtained by using algorithms CA, CPA, and CPFA for the case of three packet classes, where $\gamma_{i1} = 1$, $\gamma_{i2} = 1$, $\gamma_{i3} = 1$, $1/\mu_1 = 120$, $1/\mu_2 = 560$, $1/\mu_3 = 1120$, $B_1 = 0.05$, $B_2 = 0.1$ and $B_3 = 0.2$. Another design example for the case of four packet classes is shown in Table 4, where $\gamma_{i1} = 1$, $\gamma_{i2} = 1$, $\gamma_{i3} = 1$, $\gamma_{i4} = 1$, $1/\mu_1 = 120$, $1/\mu_2 = 560$, $1/\mu_3 = 780$, $1/\mu_4 = 1120$, $B_1 = 0.05$, $B_2 = 0.1$, $B_3 = 0.2$ and $B_4 = 0.3$. The network cost reductions observed in all design experiments ranged from several percent to about 30 percent of the cost given by Algorithm CA.

Algorithm CPFA can be run interactively, and its current implementation provides a simple editing feature which allows one to delete and/or add links to the topology. The alteration of the topology may thus be carried out in a dynamic fashion using the results obtained from the previous runs.

To describe the nature of network cost sensitivity with respect to the topological alteration, two examples for the case of two packet classes ($\gamma_{i1} = 1$, $\gamma_{i2} = 2$, $1/\mu_1 = 560$, $1/\mu_2 = 140$, $B_1 = 0.1$ and $B_2 = 3$) are reported here. In these examples, very simple alterations such as a single delete, a single add, and a single delete plus a single add, were used. To obtain a new network topology, we examined two to three candidate topologies de-

Table 1 Link capacities and costs.

	Capacity (Kbits/s)	*Cost/mile*	*Fixed cost*
1	9.6	0.5	750
2	19.2	2.10	850
3	50.0	4.20	850
4	108.0	420	2400
5	230.0	21.00	1300
6	460.0	60.00	1300

Table 2 Effect of priority and flow assignments.

Alg	*Net cost*	*Delay*		*Priority*	
	D	Z_1	Z_2	*CL1*	*CL2*
CA	102769	0.0483	0.0491	11	1
CFA	99556	0.0452	0.0648	11	1
CPA	88143	0.0408	0.0970	12	2
CPFA	84865	0.0411	0.0994	12	2

Table 3 An example with three classes of packets.

Alg	*Net cost*	*Delay*			*Priority*		
	D	Z_1	Z_2	Z_3	*CL1*	*CL2*	*CL3*
CA	88683	0.0488	0.0547	0.0636	1	1	1
CPA	75888	0.0500	0.0913	0.1009	1	2	2
CPFA	73567	0.0498	0.0701	0.1496	1	2	3

Table 4 An example with four classes of packets.

Alg	*Net cost*	*Delay*				*Priority*			
	D	Z_1	Z_2	Z_3	Z_4	*CL1*	*CL2*	*CL3*	*CL4*
CA	101219	0.0481	0.0334	0.0787	0.1039	1	1	1	1
CPA	85188	0.0381	0.0784	0.1850	0.2304	1	2	2	2
CPFA	79913	0.0393	0.0705	0.1492	0.1926	1	2	2	2

rived from the current topology by a simple alteration, and picked the one with the minimum cost.

The first example is the design of a topology with connectivity one (i.e., a tree). Using Algorithm CPFA, we started from the minimum spanning tree of Fig. 8(a) (the network cost is 49919 units with average delays $Z_1 = 0.0962$ and $Z_2 = 0.2002$), and ended up with the tree of Fig. 8(b) (cost is 47588 units with average delays $Z_1 = 0.0979$ and $Z_2 = 0.2204$). The second example is the design of a topology with connectivity two. Starting from the topology of Fig. 9(a) (cost is 55403 units with delays $Z_1 = 0.0988$ and $Z_2 = 0.2879$), we ended up with the topology of Fig. 9(b) (cost is 50345 units with delays $Z_1 = 0.1000$ and $Z_2 = 0.2303$).

Conclusions

We have shown an algorithm, which we call Algorithm CPFA, for the design of store-and-forward packet switching computer communication networks. The algorithm is a composition of three algorithms for the problems of discrete link capacity assignment, priority assignment, and flow assignment. It has been implemented in order to allow users to select different algorithms CA, CPA, CFA and CPFA. Also provided is some degree of control over the tradeoff between algorithm efficiency and solution optimality, especially in the flow assignment part of the algorithm.

Algorithm CPFA has been tested over a number of different design examples and is found to be very promising. It appears that one can expect a significant network cost reduction by combining both priority and flow assignment algorithms as is done in CPFA.

We have not considered an algorithm for the topological design of networks in this paper, but some experiments have been performed and some numerical results are given. In fact, because of the PL/I implementation under CMS, Algorithm CPFA can be run interactively with a simple editing feature to delete and/or add links to the topology. One may use information, such as link utilizations and average path lengths, obtained from the previous design to make such changes. Further study on this problem is needed.

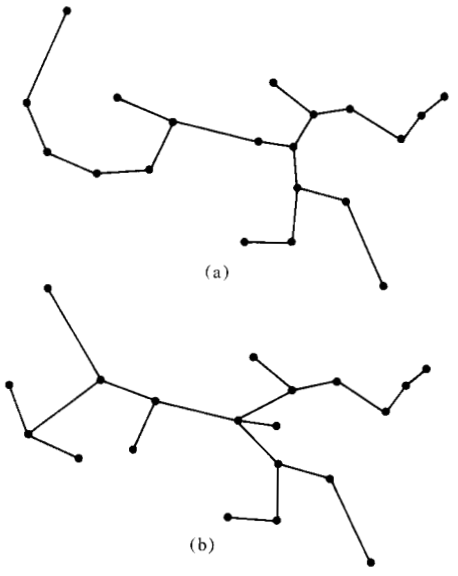

Figure 8 Topology with connectivity one.

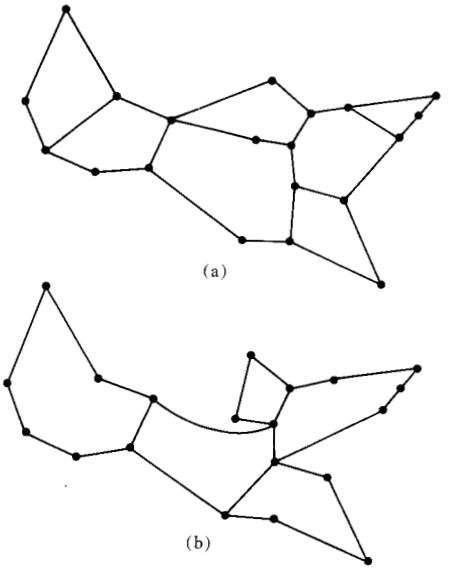

Figure 9 Topology with connectivity two.

A further application of Algorithm CPFA is that for networks that are mixed with terrestrial and satellite links. Such a design can be carried out without much complication once an explicit model for satellite links is provided for computing link distance w_{jk} in Algorithm CPFA.

References

1. R. E. Kahn, "Resource-sharing Computer Networks," *Proc. IEEE*, November 1972, p. 1397.
2. L. Pouzin, "Cigale, the Packet Switching Machine of the Cyclades Computer Network," *Proceedings of the IFIP Congress*, Stockholm, Sweden, August 1974, p. 155.
3. L. G. Roberts and B. Wessler, "The ARPA Computer Network," *Computer Communication Networks*, N. Abramson and F. F. Kuo, eds., Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972, p. 485.
4. D. L. A. Barber, "Progress with the European Informatics Network," *Proceedings of the International Computer Communication Conference*, Stockholm, Sweden, August 1974, p. 15.
5. L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill Book Co., Inc., New York, 1964.
6. L. Kleinrock, *Queuing Systems Vol. II: Computer Applications*, Wiley-Interscience Publishers, New York, 1976.
7. L. G. Roberts and B. Wessler, "Computer Network Development to Achieve Resource Sharing," *AFIPS Conf. Proc.* (Spring Joint Computer Conference, Atlantic City, NJ) **36,** 543 (1970).
8. E. Port and F. Closs, "Comparison of Switched Data Networks on the Basis of Waiting Times," *Research Report RZ405*, IBM Zurich, Switzerland, January 1971.
9. R. D. Rosher and B. Springer, "Circuit and Packet Switching; A Cost and Performance Tradeoff Study," *Computer Networks* **1,** 7 (1976).
10. K. Itoh, T. Kato, O. Hashida, and Y. Yoshida, "An Analysis of Traffic Handling Capacity of Packet Switched and Circuit Switched Networks," *Proceedings of the Third Data Communications Symposium*, St. Petersburg, Florida, 1973.
11. L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Deviation Method—An Approach to Store-and-forward Communication Network Design," *Networks* **3,** 97 (1973).
12. D. G. Cantor and M. Gerla, "The Optimal Routing of Messages in a Computer Network Via Mathematical Programming," *IEEE Computer Conference Proceedings*, San Francisco, September 1972, p. 167.
13. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design," *AFIPS Conf. Proc.* (Spring Joint Computer Conference, Atlantic City, NJ) **36,** 569 (1970).
14. B. Meister, H. R. Müller and H. R. Rudin, Jr., "New Optimization Criteria for Message Switching Networks," *IEEE Trans. Commun. Tech.* **COM-19,** 256 (1971).
15. B. Meister, H. R. Müller and H. R. Rudin, Jr., "On the Optimization of Message Switching Networks," *IEEE Trans. Commun.* **COM-20,** 8 (1972).
16. D. G. Cantor and M. Gerla, "Capacity Allocation in Distributed Computer Networks," *Proceedings of the Seventh Hawaii International Conference on System Science*, University of Hawaii, Honolulu, January 1974, p. 115.
17. K. Maruyama and D. T. Tang, "Discrete Link Capacity Assignment in Communication Networks," *Proceedings of the Third International Computer Communication Conference*, Toronto, Canada, August 1976, p. 92.
18. M. Gerla, "The Design of Store-and-forward (S/F) Networks for Computer Communications," Ph.D. Thesis, School of Engineering and Applied Science, University of California, Los Angeles, 1973.
19. K. Maruyama and D. T. Tang, "Discrete Link Capacity and Priority Assignments in Communication Networks," *IBM J. Res. Develop.* **21,** 254 (1977).
20. A. O. Allen, "Elements of Queuing Theory for System Design," *IBM Syst. J.* **14,** 161 (1975).
21. L. Kleinrock, *Queuing Systems Volume I: Theory*, Wiley-Interscience Publishers, New York, 1975.
22. D. E. Knuth, *The Art of Computer Programming Volume 1: Fundamental Algorithms*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1968.

23. G. B. Dantzig, *Linear Programming and Extension*, Princeton University Press, Princeton, NJ, 1963.
24. J. A. Tomlin, "Minimum Cost Multi-commodity Network Flows," *Oper. Res.* **14,** 45 (1966).
25. H. Frank and W. Chou, "Routing in Computer Networks," *Networks* **1,** 99 (1971).
26. C. E. Agnew, "On Quadratic Adaptive Routing Algorithms," *Commun. ACM* **19,** 18 (1976).
27. G. L. Fultz, "Adaptive Routing Techniques for Message Switching Computer-communication Networks," Ph.D. Thesis, School of Engineering and Applied Science, University of California, Los Angeles, 1972.
28. M. Schwartz and C. K. Cheung, "The Gradient Projection Algorithm for Multiple Routing in Message-switched Networks," *IEEE Trans. Commun.* **COM-24,** 449 (1976).
29. K. Maruyama, L. Fratta, and D. T. Tang, "Flow Assignment Algorithm for Computer Communication Network Design with Different Classes of Packets," *Proceedings of the Symposium on Computer Networks: Trends and Applications*, Gaithersburg, MD, November 17, 1976, p. 51.

Received December 22, 1976; revised March 2, 1977

K. Maruyama and D. T. Tang are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598; L. Fratta is located at the Istituto di Elettronica Politecnico di Milano, Milan, Italy.