K. Maruyama
D. T. Tang

Discrete Link Capacity and Priority Assignments in Communication Networks

Abstract: This paper deals with the problem of discrete link capacity assignment in store-and-forward packet switching communication networks. Our problem formulation calls for minimizing the network cost while satisfying all the average packet delay constraints specified for different classes of packets. Heuristic algorithms which give near-optimal solutions of the problem are developed.

We first describe a discrete link capacity assignment algorithm for networks with arbitrarily defined classes of packets having individual delay constraints. The problem of priority assignment on different classes of packets is then investigated, and an algorithm is developed which assigns suboptimal priorities on classes of packets based on parameters such as delay requirement, path length, packet length, and packet rate. These two algorithms for capacity assignment and priority assignment are combined and tested over a number of examples. It is found that a substantial reduction on network cost can be achieved by the use of a simple priority queuing discipline.

Introduction

This paper considers the problem of discrete link capacity and priority assignments in packet switching computer communication networks. The locations of store-and-forward nodes, network topology, the average packet rate between each pair of nodes and the routing of packets are assumed to be known.

The well-known design criterion for the capacity assignment problem first given by Kleinrock [1] is to minimize T, the average packet delay through the network, under the constraint of fixed total link cost D, where

$$D = \sum_j d_j(C_j);$$

$$T = \frac{1}{\gamma}\sum_i \gamma_i Z_i = \frac{1}{\gamma}\sum_j \lambda_j T_j,$$

where

$$\gamma = \sum_i \gamma_i,$$

γ_i is the average packet rate on the ith path,
Z_i is the average packet delay on the ith path,
λ_j is the average packet rate on the jth link,
T_j is the average packet delay on the jth link,
d_j is the cost-capacity function of the jth link, and
C_j is the jth link capacity, and is a design variable.

This is a good approach especially when the main concern of the network to be designed is overall performance. However, it has the disadvantage that some individual users may suffer average delays longer than they would prefer.

Instead of minimizing the average packet delay T, Meister, Muller, and Rudin [2, 3] suggested the design criteria which minimize $T^{(k)}$, the weighted sum of powers of the average link delays, under the constraint of fixed link cost D, where

$$T^{(k)} = \left[\frac{1}{\gamma}\sum_j \lambda_j (T_j)^k\right]^{\frac{1}{k}}.$$

Although such design procedures prevent the large spread of delays among links, they do not directly control the path delays with respect to different users and/or packet classes.

To achieve some degree of control over delays encountered by different users with different delay requirements, Maruyama and Tang [4] have suggested the design criterion of minimizing T_w, the weighted sum of the average path delays under the constraint of fixed link cost D.

$$T_w = \frac{1}{\gamma}\sum_i \gamma_i (w_i Z_i) = \frac{1}{\gamma}\sum_j \lambda_j^* T_j,$$

where $\lambda_j^* = \Sigma_i \delta_{ij}\,(w_i\gamma_i)$, δ_{ij} is 1 if the jth link is on the ith path, otherwise 0, and w_i is the weight on the average path delay Z_i. The weight w_i may be a function of packet class. For instance, one may assign large weights to "important" users.

The major advantage of any of the above three design criteria is that when linear cost-capacity functions are assumed, the capacity assignment problem becomes mathematically tractable and a closed-form analytical solution can usually be obtained by using the method of

Lagrange multipliers [5]. When other cost-capacity functions, e.g., concave or discrete, are assumed, closed-form analytical solutions no longer exist; however, reasonably simple and computationally fast procedures are available for finding either optimal or near optimal solutions [3, 6-8]. The major disadvantage of these types of design criteria is that they do not specifically satisfy the needs of individual users and the network administrator, especially when the network is used by a wide variety of users who may have different path lengths and different delay requirements. The algorithm for solving the capacity assignment problem is often referred to as Algorithm CA [7].

In order to design an optimal or near-optimal network which satisfies a set of design constraints, Maruyama and Tang [4] have suggested a new design criterion assuming a delay requirement on each priority level on each path. Their design criterion is to minimize the total link cost D, while satisfying the constraint

$Z_{ir} \leq B_{ir}$ for all i and r,

where Z_{ir} is the average class k packet delay on the ith path, and
B_{ir} is the average class k packet delay bound on the ith path.

They have developed a composite heuristic algorithm assuming discrete link cost-capacity functions, and applied the algorithm in numerical examples to study the network cost sensitivity with respect to various network parameters including delay constraints, buffer sizes, packet sizes and queuing disciplines. However, specific priority levels are assumed to be given. The problem of assigning priority levels to packet classes for further network cost reduction is not considered.

In this paper, we allow packet classes to be defined arbitrarily so as to cover problems of various complexity. The major differences between our new formulation and other formulations [1-3] are that we allow the flow of multiple packet classes on a given path and that the problem of priority assignment on packet classes for further network cost reduction is considered. Since we are dealing with both capacity and priority assignments, we shall refer to this problem as the CPA problem. The heuristic algorithm we develop will be referred to as Algorithm CPA.

CPA problem formulation

In order to design a common network for different applications, we allow a general definition of packet classes, and consider a network that satisfies a set of user constraints, rather than a single overall average packet delay constraint [1]. The problem is formulated as a discrete link capacity and priority assignment (CPA) problem as follows:

Given:

a network topology with node locations,
a traffic requirement matrix for each packet class,
a flow assignment for each packet class, and
a B_k for each packet class,

minimize:

the network cost

$$D = \sum_j d_j(C_j),$$

with respect to:

the discrete link capacities C_j and
the priority levels P_k.

The constraint is:

$Z_k \leq B_k$ for each packet class,

where

Z_k is the average delay for packet class k,
B_k is the average delay bound for packet class k,
C_j is the jth link capacity selected from a finite set of options.
d_j is the discrete link cost-capacity function of the jth link, and
P_k is the priority level assigned to packet class k.

Z_k may be computed from

$$Z_k = \frac{1}{\gamma_k} \sum_i \gamma_{ik} Z_{ik} = \frac{1}{\gamma_k} \left(\sum_j \lambda_{jk} T_{jk} + \sum_s \theta_{sk} t_{sk} \right),$$

where

$\gamma_k = \Sigma_i \gamma_{ik}$ is the rate of packet class k entering the network,
γ_{ik} is the rate of packet class k on the ith path,
Z_{ik} is the delay of packet class k on the ith path,
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 at node s for packet class k.

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 [1]. We use the standard formula for an M/M/1 nonpreemptive priority (or a head-of-the-line) queuing system [9, 10] to compute T_{jk}, which is the sum of the mean queuing delay on link j for packets in priority level r, the propagation delay and the mean service time of class k packets (priority level r is assigned to packet class k). In practice, the nodal processing delay may be a small portion of the path delay Z_{ik}, and can often be accounted for by a constant [11], e.g., 1 ms for t_{sk} at each node.

Table 1 Notation.

L_j	link j
C_j	capacity currently assigned to L_j
$C_j^{(+)}$	next higher capacity available to C_j, $C_j < C_j^{(+)}$
$C_j^{(-)}$	next lower capacity available to C_j, $C_j^{(-)} < C_j$
D_j	cost of L_j when C_j is used
$D_j^{(+)}$	cost of L_j when $C_j^{(+)}$ is used
$D_j^{(-)}$	cost of L_j when $C_j^{(-)}$ is used
λ_{jk}	class k packet rate on L_j
T_{jk}	average link delay for packet class k when C_j is used on L_j
$T_{jk}^{(+)}$	average link delay for packet class k when $C_j^{(+)}$ is used on L_j
$T_{jk}^{(-)}$	average link delay for packet class k when $C_j^{(-)}$ is used on L_j
γ_k	rate of packet class k entering the network
Z_k	average delay for packet class k when C_j is used on L_j
$Z_k^{(+)}$	average delay for packet class k when $C_j^{(+)}$ is used on L_j
$Z_k^{(-)}$	average delay for packet class k when $C_j^{(-)}$ is used on L_j
B_k	average packet delay bound on Z_k

Because of the complexity involved in finding the exact solution, we have chosen to develop heuristic algorithms for both capacity and priority assignment problems and combine them into a composite Algorithm CPA.

Capacity assignment algorithm

The best summary on the CA problem and various solutions for a single class of packets (no priority assignment) for linear and nonlinear cost-capacity functions may be found in reference [11]. However, the work described in [4] is to our knowledge the first to consider network design with multiple delay requirements.

In the following, we develop an algorithm for the CA problem of n packet classes with r ($1 \leq r \leq n$) arbitrarily specified priority levels. We first describe several heuristics to be used in various phases of cost optimization in the CA problem. These heuristics are then combined, based on the results of several experiments, to give a composite algorithm which we refer to as Algorithm CA.

In the course of network cost optimization, we construct a cost-delay table for each link. The cost-delay table gives, for each available discrete capacity C_j, a monthly link cost $d_j(C_j)$ and the resulting average link delay T_{jk} for each packet class k. We assume that in each table, entries are sorted in order of increasing cost [12–14].

• *Description of heuristics*

In this section, we describe several heuristics used in Algorithm CA. Figures of merit based on cost-delay tradeoffs are introduced to facilitate cost optimization. Here the notations listed in Table 1 will be used.

Each of the first two heuristics SETHIGH and SETLOW gives an initial capacity assignment:

SETHIGH: Assign to each link the maximum available capacity (if $Z_k > B_k$ for some k, no feasible solution exists).

SETLOW: First determine the solution feasibility using SETHIGH. If it is feasible, assign to each link the minimum available capacity which accommodates the given link traffic (if $Z_k \leq B_k$ for all k, this minimum capacity assignment is the optimal solution).

We now introduce the basic capacity assignment heuristics.

ADDFAST: Repeat the following steps until $Z_k \leq B_k$ holds for all k.

1. Find the packet class k_0 which satisfies

$$Z_{k_0}/B_{k_0} = \max_k \{Z_k/B_k\}.$$

2. Find the link L_{j_0} which carries class k_0 packets and which satisfies

$$\frac{\lambda_{j_0k_0}(T_{j_0k_0} - T_{j_0k_0}^{(+)})}{D_{j_0}^{(+)} - D_{j_0}} = \max_j \left\{\frac{\lambda_{jk_0}(T_{jk_0} - T_{jk_0}^{(+)})}{D_j^{(+)} - D_j}\right\}.$$

3. Assign the capacity of $C_{j_0}^{(+)}$ to the link L_{j_0}.

The first step determines the packet class whose delay constraint is least satisfied. The next step determines the link which tends to give the maximum performance improvement per unit cost for this class. Finally, this link capacity is increased to the next higher capacity available. The computational time required in each iteration is basically proportional to the number of links which carry class k packets.

ADD: Repeat the following steps until $Z_k < B_k$ holds for all k.

1. For each link L_j, compute its figure of merit F_j.

$$F_j = \frac{\sum_k (\gamma_k \max\{0, Z_k - \max\{Z_k^+, B_k\}\})/B_k}{D_j^+ - D_j}.$$

2. Find the link L_{j_0} which satisfies $F_{j_0} = \max_j \{F_j\}$.

3. Assign the capacity $C_{j_0}^{(+)}$ to the link L_{j_0}.

In this heuristic we first determine a link whose capacity increment gives the maximum overall performance improvement per unit cost, and then increase the link capac-

ity to the next higher capacity available. The computational time required in each iteration is no worse than proportional to the product of the number of links and the number of packet classes.

We next introduce two heuristics, DROPFAST and DROP, which attempt to decrease link capacities:

DROPFAST: Repeat the following steps as long as $Z_k^{(-)} \leq B_k$ holds for all k.

1. Find the packet class k_0 which satisfies
$$Z_{k_0}/B_{k_0} = \min_k \{Z_k/B_k\}.$$
2. Find the link L_{j_0} which carries class k_0 packets and which satisfies
$$\frac{\lambda_{j_0k_0}(T_{j_0k_0}^{(-)} - T_{j_0k_0})}{D_{j_0} - D_{j_0}^{(-)}} = \min_j \left\{\frac{\lambda_{jk_0}(T_{jk_0}^{(-)} - T_{jk_0})}{D_j - D_j^{(-)}}\right\}.$$
3. Assign the capacity C_{j_0} to the link L_{j_0}.

As with ADDFAST, this heuristic first determines the packet class whose delay constraint is most satisfied. The next step determines the link which tends to give the least performance degradation per unit cost for this class. This link capacity is then decreased to the next lower capacity available. This heuristic also requires a computational time in each iteration proportional to the number of links which carry class k packets.

DROP: Repeat the following steps as long as $Z_k^{(-)} \leq B_k$ holds for all k.

1. For each link L_j, compute its figure of merit F_j:
$$F_j = \frac{\sum_k (\gamma_k(Z_k^{(-)} - Z_k)/B_k)}{D_j - D_j^{(-)}}.$$
2. Find the link L_{j_0} which satisfies $F_{j_0} = \min_j \{F_j\}$.
3. Assign the capacity $C_{j_0}^{(-)}$ to the link L_{j_0}.

As with ADD, we first determine a link whose capacity decrement gives the least overall performance degradation per unit cost, and then decrease the link capacity to the next lower capacity available. The computational time required in each iteration is again no worse than proportional to the product of the number of links and the number of packet classes.

We describe another heuristic which we refer to as EXC.

EXC: For any two links L_i and L_j, reassign $C_i \leftarrow C_i^{(+)}$ and $C_j \leftarrow C_j^{(-)}$ if the reassignment does not violate any delay constraints and if $D_i + D_j > D_i^{(+)} + D_j^{(-)}$.

This heuristic attempts to improve further the network cost by pairwise link capacity perturbations. Since it checks for all pairs of links, the computational time required could be exceedingly large for large networks. To facilitate the computation, we find two sets of links (each with about n links), one favoring link capacity increases and another favoring link capacity decreases. The selection of these sets of links is based on figures of merit defined in ADD and DROP. Therefore, the computational time required by EXC is proportional to the square of the number of nodes.

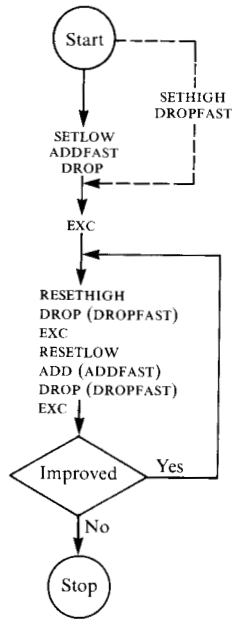

Figure 1 Discrete link capacity assignment algorithm CA for a given priority assignment.

Composite algorithm CA

After experimenting with these heuristics on a number of different capacity assignment problems, we observed that a solution given by any one heuristic can quite frequently be improved by running other heuristics consecutively. For example, a composite algorithm such as (SETLOW, ADDFAST, DROP, EXC) or (SETHIGH, DROPFAST, EXC) is very simple and yet very good. Another observation which we made, especially on larger networks, is that further improvement on a solution can be expected from running ADD (or ADDFAST) and DROP (or DROPFAST) heuristics alternatively, though such improvement is typically less than five percent.

To allow the concatenation of basic heuristics, two interfaces, RESETLOW and RESETHIGH, are provided. RESETLOW is the interface to ADD or ADDFAST and it resets the currently assigned capacity C_j to $C_j^{(-)}$ for all j. RESETHIGH is the interface to DROP or DROPFAST and it resets the currently assigned capacity C_j to $C_j^{(+)}$ for all j.

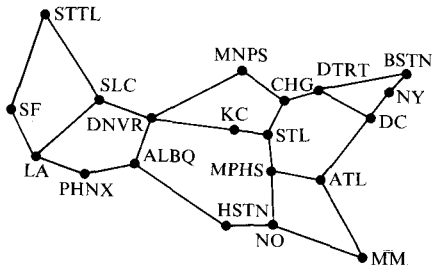

Figure 2 Mesh network.

Table 2 Link capacities and costs

Link	*Capacity* (kbits/s)	*Cost/mile*	*Fixed cost*
1	9.6	0.5	750
2	19.2	2.1	850
3	50	4.2	850
4	108	4.2	2400
5	230	21	1300
6	460	60	1300

Table 3 Network costs where $\gamma_{i1} = \gamma_{i2} = 1$, $1/\mu_1 = 120$, $1/\mu_2 = 560$, $l_1 = l_2 = 3.417$.

		Network costs for different priority assignments				
Delay bound (ms)						
B_1	B_2	*(1, 1)*	*(1, 2)*	*(2, 1)*	*H3*	*H4*
100	100	56152	56468	55402	56152	55402
"	110	54673	55402	55402	54673	54673
"	125	53411	53842	"	53411	53411
"	150	51874	52915	"	51874	51874
"	200	51874	50009	"	50009	50009
"	250	"	48788	"	48788	48788
"	300	"	47858	"	47858	47858
"	400	"	47858	"	"	"

After extensive experiments on various compositions, we observed that the composite Algorithm CA as shown in Fig. 1 gives the best overall performance, considering both solution optimality and computational efficiency.

Priority assignment algorithm

Recent work by Maruyama and Tang [4] has shown that a substantial reduction in network cost can, in general, be achieved by distinguishing delay requirements among different classes of packets and by allowing different treatments for different classes of packets using simple priority schemes. In this section, we investigate further the problem of priority assignment on different classes of packets in order to achieve further reduction of network cost. Specifically, we address the problem of mapping classes of packets into priority levels. Although it is possible to consider a 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 in this paper.

• *Examples*

Let us first observe from examples how network costs will be influenced by assignments of different priorities to classes of packets. The notation $(P_1, P_2, \cdots, P_k, \cdots, P_n)$ is used in these examples to denote a priority assignment on n classes of packets. That is, the priority level P_k is assigned to packet class k for all k. For example, (2, 3, 3, 1, 1) means that priority level 2 is assigned to the first packet class, priority level 3 is assigned to both the second and third packet classes and priority level 1 is assigned to both the fourth and fifth packet classes. In all of these examples, the mesh network of 20 nodes and 26 links in Fig. 2 with the shortest path routing is assumed. Table 2 shows six link capacities with monthly costs as used in our examples.

In Tables 3 and 4, the first three columns under the heading of priority assignments (1, 1), (1, 2) and (2, 1) show how network cost varies as priority assignment changes. In these tables, traffic requirements (packets/s) between pairs of nodes for two packet classes, 1 and 2, are denoted by γ_{i1} and γ_{i2}. The average packet lengths (bits/packet) which include both data and packet overhead are denoted by $1/\mu_1$ and $1/\mu_2$. The average path lengths [15], the average number of links traveled by a packet to reach its destination, are denoted by l_1 and l_2. In these tables, the underlined costs show the minimum costs achieved by optimal priority assignments.

In the first row of Table 3, we see that even though the delay requirements for both packet classes are the same ($B_1 = B_2 = 100$ ms), the best priority assignment is (2, 1). That is, it is better to assign the lower priority (level 2) to packet class 1 and the higher priority (level 1) to packet class 2. This is due to the fact that a packet in class 2 requires on the average a longer transmission time than that required by a packet in class 1. As the difference between the delay bounds B_1 and B_2 increases, the optimal priority assignment changes. For the next three rows where $110 \leq B_2 \leq 150$, the optimal priority assignment is (1, 1), i.e., the equal priority assignment. And for the last four rows where $B_2 \geq 200$, the optimal priority assignment becomes (1, 2); i.e., the higher priority to packet class 1 and the lower priority to packet class 2. This is due to the fact that two delay bounds B_1 and B_2

are widely separated and thus they can be satisfied by the priority assignment (1, 2) resulting in a less costly network. Table 4 may be explained similarly.

• *Complexity*

Unless an explicit priority assignment on classes of packets is prespecified, the priority assignment itself becomes a part of the network design problem. In general, there is no guaranteed method of determining an optimal priority assignment without exhaustively solving the capacity assignment problem for all possible priority assignments.

Let n be the number of packet classes which share the common communication network, and let $N(n)$ be the total number of different priority assignments on n packet classes. Let $\{{n \atop r}\}$ denote the number of different ways to partition n different items into r nonempty groups. This number $\{{n \atop r}\}$ is often referred to as a Stirling number [16] which satisfies the following recurrence equation:

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

where

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

For each such partition there are $r!$ ways to order groups and to assign numbers which correspond to distinct priority levels. Thus there are $r!\left\{{n \atop r}\right\}$ different ways to map n packet classes into r priority levels. The total number of different priority assignments $N(n)$ on n packet classes is therefore

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

The number of different priority assignments $N(n)$ grows very fast 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 $n > 5$, we seek to reduce the number of priority assignments examined. We now describe a heuristic procedure which consists of two steps. The first step is to order packet classes according to values of priority "preference," and the second step is to find a good partition for such a sequence.

• *Priority preference among packet classes*

In this section, we develop a figure of merit which is used to order classes of packets into a sequence of nonincreasing priority "preference" values among classes of packets. Let such a sequence be denoted by $K = \{k_1, k_2, \cdots, k_i, k_{i+1}, \cdots, k_n\}$ where k_i denotes a packet class whose priority preference is the ith among n packet classes. This priority preference sequence implies that it is preferable to assign a higher or equal priority to a packet class k_i than to a packet class k_j for $1 \le i < j \le n$, denoted by

$$\mathrm{PR}(k_i) \ge \mathrm{PR}(k_j) \qquad \text{if } 1 \le i < j \le n.$$

In the previous section, we allowed an arbitrary definition of packet classes. Thus, in general, parameters such

Table 4 Network costs where $\gamma_{i1} = \gamma_{i2} = 1$, $1/\mu_1 = 120$, $1/\mu_2 = 560$, $l_1 = 1.621$, $l_2 = 3.417$.

		Network costs for different priority assignments				
Delay bound (ms)						
B_1	B_2	(1, 1)	(1, 2)	(2, 1)	*H3*	*H4*
100	100	56468	56468	55802	55802	55802
"	110	54673	54673	54570	54570	54570
"	125	53355	53355	53027	53027	53027
"	150	51024	51024	51444	51024	51024
"	200	47856	47856	"	47856	47856
"	250	46066	46066	"	46066	46066
"	300	46066	44899	"	44899	44899
"	350	"	44395	"	44395	44395
"	400	"	44108	"	44108	44108
"	500	"	42976	"	42976	42976

Table 5 Comparison between Heuristics H3 and H4.

B_k	l_k	$1/\mu_k$	γ_{ik}	Z_k	$D(H3)$	Z_k	$D(H4)$
100	3.416	120	1	58.2	55621	94.3	55219
100	3.416	560	1	98.4		93.1	
70	3.416	120	1	37.5	64372	52.1	62902
70	3.416	560	1	69.2		69.8	
70	3.509	120	1	38.9	63333	56.1	62930
70	3.509	560	1	68.9		69.3	
50	3.416	120	1	23.7	95929	27.1	95818
50	3.416	840	1	49.9		49.9	
50	3.509	120	1	23.9	96224	27.7	95828
50	3.509	840	1	49.9		49.9	
70	3.416	120	2	32.5	57498	36.8	56358
70	3.416	560	0.5	69.9		69.4	
70	3.509	120	2	34.5	57190	41.5	55114
70	3.509	560	0.5	69.8		69.8	
50	3.416	120	2	20.0	89729	20.5	88676
50	3.416	840	0.5	49.9		49.8	
50	3.509	120	2	20.4	85954	21.2	84404
50	3.509	840	0.5	49.7		49.8	
100	3.416	120	1	97.7	49590	97.7	49590
100	3.416	120	5	97.7		97.7	

Table 6 Priority assignment based on priority preference where $\gamma_{ik} = 2$, $1/\mu_k = 120$, $B_k = 100$ for all k; $l_1 = 1.621$, $l_2 = 3.455$, $l_3 = 5.395$, $l_4 = 7.143$; $K^{(1)} = \{4, 3, 2, 1\}$.

Number of priority classes	*Priority assignment* Class 1	Class 2	Class 3	Class 4	*Network cost*
1	1	1	1	1	34529
2	2	1	1	1	33895
2	2	2	1	1	32860
2	2	2	2	1	34006
3	3	2	1	1	32023
3	3	3	2	1	32115
4	4	3	2	1	31798

Table 7 Priority assignment based on priority preference where $\gamma_{ik} = 2$, $1/\mu_k = 120$ for all k, $B_1 = 50$, $B_2 = 100$, $B_3 = 150$, $B_4 = 200$; $l_1 = 1.621$, $l_2 = 3{,}455$, $l_3 = 5.395$, $l_4 = 7.143$; $K^{(1)} = \{3, 4, 2, 1\}$.

Number of priority classes	*Priority assignment* Class 1	Class 2	Class 3	Class 4	*Network cost*
1	1	1	1	1	29317
2	2	1	1	1	32065
2	2	2	1	1	30407
2	2	2	1	2	30354
3	3	2	1	1	32065
3	3	3	1	2	30407
4	4	3	1	2	32065

as average delay bounds B_k, average path length l_k, average packet length $1/\mu_k$, and average rate γ_k for packets entering the network may be different for different packet classes. These four parameters are considered in our figure of merit to give priority preference values.

Let us consider two packet classes k and h, and let their sets of parameters be denoted by $(B_k, l_k, \mu_k, \gamma_k)$ and $(B_h, l_h, \mu_h, \gamma_h)$, respectively. The following heuristics are intuitively reasonable and have been substantiated by numerical examples:

H1: if $B_k < B_h$, then $\mathrm{PR}(k) \geq \mathrm{PR}(h)$,

and

H2: if $l_k > l_h$, then $\mathrm{PR}(k) \geq \mathrm{PR}(h)$.

To accommodate cases where both delay bounds and average path lengths are different, we have derived the following two heuristics:

H3: If $B_k/l_k < B_h/l_h$,

then $\mathrm{PR}(k) \geq \mathrm{PR}(h)$,

and

H4: if $(B_k - Z_k)/l_k < (B_h - Z_h)/l_h$,

then $\mathrm{PR}(k) \geq \mathrm{PR}(h)$.

Here, Z_k and Z_h are the average delays of packet classes k and h computed under the equal priority assignment.

In Tables 3 and 4, some examples are shown. Here the costs in column H3 were computed using the heuristic H3 and the costs in column H4 were computed using the heuristic H4. In Table 3, two packet classes with conditions $B_1 \leq B_2$, $l_1 = l_2$, $\mu_1 > \mu_2$ and $\gamma_1 = \gamma_2$ are assumed. Thus, H3 gives the priority preference $\mathrm{PR}(1) \geq \mathrm{PR}(2)$ for $B_1 < B_2$. For $B_1 = B_2$, H3 gives no priority preference among these two packet classes; however, H4 gives the priority preference $\mathrm{PR}(2) \geq \mathrm{PR}(1)$, which leads to the optimal priority assignment. In Table 4, both H3 and H4 give the same priority preference sequence, leading to the same network cost.

To examine the relative performance between H3 and H4, one should consider examples where parameters such as delay bounds and average path lengths are the same among packet classes but either average packet lengths or traffic rates are different. We have generated a number of such examples and have tested on both H3 and H4. Some of these are shown in Table 5. It can be observed that H4 generally gives performance better than or equal to H3. Consequently, we adopted H4 in our Algorithm CPA.

• *Determination of the number of priority classes*

Once a priority preference ordering of different classes of packets has been determined using the heuristic H4 described in the previous section, the next step is to determine the best number of partitions corresponding to distinct priority assignments. It is clear than an ordered sequence of n elements has 2^{n-1} different partitions. Since it is still impractical in most cases to exhaust all 2^{n-1} possible priority assignments to find the optimal priority assignment under the selected ordering (one should remember that a CA problem has to be solved for each priority assignment examined), we use a sequential partition strategy which examines at most $n(n-1)/2 + 1$ different priority assignments. That is, our algorithm solves up to $n(n-1)/2 + 1$ CA problems in order to find a good priority assignment on n packet classes. Our strategy as described below is to establish partition boundaries one by one until no further cost reduction can be achieved.

Let K be the priority preference sequence of n classes of packets, where

$$K = (k_1, k_2, \cdots, k_i, \cdots, k_n),$$

such that $\mathrm{PR}(k_i) \geq \mathrm{PR}(k_j)$ for all $1 \leq i < j \leq n$.

At the initial stage $K^{(1)} = K$, and $D^{(1)} =$ the network cost with equal priority assignment. And at the rth stage

$$K^{(r)} = \{K_1^{(r)}, \cdots, K_j^{(r)}, \cdots, K_r^{(r)}\},$$

where $D^{(r)} =$ the network cost when the packet classes in $K_j^{(r)}$ are assigned to priority level j for all j, and $D^{(1)} > D^{(2)} > \cdots > D^{(r)}$.

In order to find $K^{(r+1)}$, the following steps are carried out. If the minimum network cost $D^{(r+1)}$ can be achieved by partitioning $K_j^{(r)}$ into two subpartitions, $K_{j1}^{(r)}$ and $K_{j2}^{(r)}$, and if $D^{(r+1)} < D^{(r)}$, then set $K^{(r+1)}$ to

$$K_i^{(r+1)} = K_i^{(r)}, \qquad \text{for } 1 \leq i \leq j-1,$$

$$K_j^{(r+1)} = K_{j1}^{(r)},$$

$$K_{j+1}^{(r+1)} = K_{j2}^{(r)}, \text{ and}$$

$$K_{i+1}^{(r+1)} = K_i^{(r)}, \qquad \text{for } j+1 \leq i \leq n.$$

Otherwise, terminate (the best partition is $K^{(r)}$ with r priority levels and with cost $D^{(r)}$).

Let us consider the number of priority assignments examined in the above procedure. It is clear that, in the worst case, the procedure terminates after computing the network cost $D^{(r)}$ for $r = 1, 2, \cdots, n$, where r corresponds to the number of priority levels. Since the partition boundaries remain unchanged once they are chosen for each r, $2 \leq r \leq n$, $(n - r + 1)$ different priority assignments are examined. The total number of priority assignments examined for the worst case is therefore $1 + (n-1) + (n-2) + \cdots + 1 = n(n-1)/2 + 1$.

In the above procedure, one may eliminate the condition $D^{(r)} > D^{(r+1)}$ on the cost reduction at each stage and compute, instead, $K^{(i)}$ and $D^{(i)}$ for $1 \leq i \leq n$, and the minimum of $D^{(i)}$ may be obtained.

In order to see how the algorithm works, two examples with four classes of packets are shown in Tables 6 and 7. In Table 6, the priority preference sequence given by the heuristic H4 is $K^{(1)} = \{4, 3, 2, 1\}$. With no priority assignment, the network cost is $D^{(1)} = 34529$ units. The algorithm finds the best partition of two priority levels, $K^{(2)} = \{4, 3\}\ \{2, 1\}$ with $D^{(2)} = 32860$, which is the third row in the table. Since $D^{(1)} > D^{(2)}$, the algorithm continues to search for the best partition with three priority levels, and finds $K^{(3)} = \{4, 3\}\ \{2\}\ \{1\}$ with $D^{(3)} = 32023$, which is the fifth row in the table. Since $D^{(2)} > D^{(3)}$, the algorithm still continues and finds $K^{(4)} = \{4\}\ \{3\}\ \{2\}\ \{1\}$ with $D^{(4)} = 31798$. Thus, the best priority assignment found by the algorithm is (4, 3, 2, 1) with the network cost 31798 units. That is, priority level 4 (the lowest priority level) is assigned to packet class 1, priority level 3 to packet class 2, priority level 2 to packet class 3 and priority level 1 (the highest priority level) to packet class 4. Table 7 shows a similar example, in which $K^{(1)} = \{3, 4, 2, 1\}$ and $D^{(1)} = 29317$. In this example, $D^{(2)} = 30354$ with priority assignment (2, 2, 1, 2). Since $D^{(1)} < D^{(2)}$ holds, the algorithm will then terminate. The best priority assignment found by the algorithm is the equal priority assignment (1, 1, 1, 1) with a network cost of 29317 units.

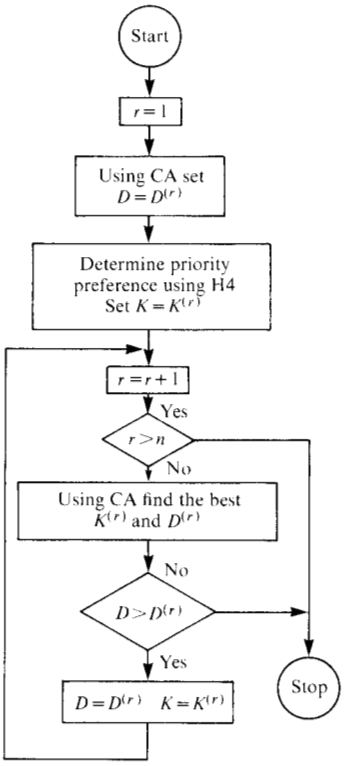

Figure 3 Capacity and priority assignment algorithm CPA.

Numerical results

The overall discrete link capacity assignment algorithm combined with the priority assignment algorithm, which we call Algorithm CPA, is diagramed in Fig. 3. The algorithm was tested on a number of design examples and some results are shown in Tables 8, 9, 10 and 11. As can be seen from these tables, the network cost may be substantially reduced by the use of appropriate priority assignments achievable with Algorithm CPA. Tables 10 and 11 show the network cost and priority assignments given by Algorithm CPA for different network loads α, $1 \leq \alpha \leq 8$. Note that the best priority assignment changes as $\alpha\gamma_{ik}$ changes, even though all other parameters are kept unchanged.

Table 8 Design examples of two classes of packets, where $\gamma_{i1} = \gamma_{i2} = 1$, $1/\mu_1 = 120$, and $1/\mu_2 = 560$.

Examples	*CA*	*CPA*
$l_1 = 4.988$, $l_2 = 3.416$ $B_1 = 50$, $B_2 = 100$	(1, 1) $Z_1 = 49.7$, $Z_2 = 68.3$ 65984	(1, 2) $Z_1 = 49.9$, $Z_2 = 85.1$ 59339
$l_1 = l_2 = 3.416$ $B_1 = 50$, $B_2 = 500$	(1, 1) $Z_1 = 49.4$, $Z_2 = 86.8$ 57917	(1, 2) $Z_1 = 49.9$, $Z_2 = 191.3$ 51122
$l_1 = l_2 = 3.416$ $B_1 = 100$, $B_2 = 500$	(1, 1) $Z_1 = 95.7$, $Z_2 = 143.1$ 52044	(1, 2) $Z_1 = 67.0$, $Z_2 = 370.4$ 46577

Table 9 Design examples of four classes of packets, where $l_1 = 1.621$, $l_2 = 2.2$, $l_3 = 3.006$, and $l_4 = 3.417$.

Examples	*CA*	*CPA*
$1/\mu_k = 120$ $B_1 = 40$, $B_2 = 200$, $B_3 = 50$, $B_4 = 100$ $\gamma_{ik} = 2$	(1, 1, 1, 1) $Z_1 = 25.3$, $Z_2 = 34.1$, $Z_3 = 45.1$, $Z_4 = 50.8$ 47946	(1, 3, 1, 2) $Z_1 = 22.9$, $Z_2 = 137.6$, $Z_3 = 39.2$, $Z_4 = 73.1$ 43677
$1/\mu_1 = 120$, $1/\mu_2 = 180$, $1/\mu_3 = 300$, $1/\mu_4 = 540$ $B_1 = 50$, $B_2 = 100$, $B_3 = 150$, $B_4 = 200$ $\gamma_{ik} = 1$	(1, 1, 1, 1) $Z_1 = 46.7$, $Z_2 = 63.0$, $Z_3 = 92.0$, $Z_4 = 110.0$ 57928	(1, 2, 2, 2) $Z_1 = 28.5$, $Z_2 = 95.3$, $Z_3 = 130.0$, $Z_4 = 158.7$ 54478
$1/\mu_k = 120$ $B_1 = 400$, $B_2 = 300$, $B_3 = 200$, $B_4 = 100$ $\gamma_{i1} = 8$, $\gamma_{i2} = 4$, $\gamma_{i3} = 2$, $\gamma_{i4} = 1$	(1, 1, 1, 1) $Z_1 = 58.4$, $Z_2 = 73.1$, $Z_3 = 89.6$, $Z_4 = 93.5$ 49029	(2, 1, 1, 1) $Z_1 = 394.0$, $Z_2 = 42.3$, $Z_3 = 59.6$, $Z_4 = 68.3$ 45499
$1/\mu_1 = 540$, $1/\mu_2 = 300$, $1/\mu_3 = 180$, $1/\mu_4 = 60$ $B_1 = 200$, $B_2 = 150$, $B_3 = 100$, $B_4 = 50$ $\gamma_{ik} = 1$	(1, 1, 1, 1) $Z_1 = 23.7$, $Z_2 = 30.6$, $Z_3 = 40.0$, $Z_4 = 45.4$ 48697	(3, 3, 2, 1) $Z_1 = 85.3$, $Z_2 = 109.3$, $Z_3 = 58.1$, $Z_4 = 45.4$ 43688

Table 10 Effect of traffic $\alpha\gamma_{ik}$ on priority assignment, where $\gamma_{ik} = 1$, $1/\mu_k = 120$ for all k; $B_1 = 50$, $B_2 = 80$, $B_3 = 150$, $B_4 = 50$; $l_1 = 1.6$, $l_2 = 2.165$, $l_3 = 2.621$, and $l_4 = 3.342$.

α	*CA*	*CPA*
1	38351	36052 (2, 2, 2, 1)
2	47526	42872 (2, 3, 4, 1)
3	55605	53262 (1, 2, 2, 1)
4	60261	58690 (2, 2, 2, 1)
5	67133	65379 (1, 1, 2, 1)
6	73051	73051 (1, 1, 1, 1)
7	83273	82088 (2, 2, 2, 1)
8	87923	87923 (1, 1, 1, 1)

Table 11 Effect of traffic $\alpha\gamma_{ik}$ on priority assignment, where $\gamma_{i1} = 1/4$, $\gamma_{i2} = 1/3$, $\gamma_{i3} = 1/2$, $\gamma_{i4} = 1$; $1/\mu_1 = 300$, $1/\mu_2 = 240$, $1/\mu_3 = 180$, $1/\mu_4 = 120$; $B_1 = 50$, $B_2 = 80$, $B_3 = 150$, $B_4 = 50$; $l_1 = 1.6$, $l_2 = 2.165$, $l_3 = 2.621$, and $l_4 = 3.342$.

α	*CA*	*CPA*
1	38036	38483 (2, 3, 3, 1)
2	45825	42638 (2, 3, 3, 1)
3	53356	49579 (2, 2, 3, 1)
4	57961	56821 (1, 2, 2, 1)
5	62560	61410 (1, 2, 2, 1)
6	68683	68683 (1, 1, 1, 1)
7	73051	73051 (1, 1, 1, 1)
8	80538	78988 (1, 2, 2, 1)

In order to check the optimality of the priority assignments given by Algorithm CPA, we also ran Algorithm CA on all possible priority assignments, and have obtained a set of optimal priority assignments for each of the examples illustrated in this paper. In these cases studied, we observed that Algorithm CPA gave optimal priority assignments. Although this does not guarantee the optimality of our priority assignment algorithm, we feel that the algorithm is near-optimal even for larger numbers of classes of packets.

Conclusion

In this paper, we first developed a generalized discrete link capacity assignment algorithm to accept arbitrarily defined classes of packets with different delay requirements. The capacity assignment algorithm CA was used to study the network cost sensitivity with respect to different priority assignments. It is observed that a substantial reduction on network cost can be achieved by an appropriate priority assignment on classes of packets.

Unless a priority assignment on classes of packets is prespecified, the priority assignment becomes a part of the network design problem. Since the number of different priority assignments for n classes of packets quickly becomes prohibitive for exhaustive examination of all priority assignments when $n > 5$, we developed a priority assignment algorithm which examines at most $n(n-1)/2 + 1$ priority assignments. This heuristic approach is based on an ordering of classes of packets according to their priority preference values, together with an add-type sequential partition strategy. In the determination of priority preference values among different classes of packets, parameters such as delay constraints, average path length, average packet length and average packet rate on different classes of packets were considered.

We have combined the priority assignment algorithm with the capacity assignment algorithm CA, and developed an algorithm CPA which gives near-optimal capacity assignments and priority assignments.

In this paper, we have not considered network cost reduction with flow assignments (packet routing) and network topologies as design variables, which they often are in practice. Some work has been done [7], but further research is needed here. Studies of this general design problem are in progress and will be reported separately.

References and notes

1. L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill Book Co., Inc., New York, 1964.
2. B. Meister, H. R. Muller, and H. R. Rudin, Jr., "New Optimization Criteria for Message Switching Networks," *IEEE Trans. Commun. Technol.* **COM-19,** 256 (1971).
3. B. Meister, H. R. Muller, and H. R. Rudin, Jr., "On the Optimization of Message Switching Networks," *IEEE Trans. Commun. Technol.* **COM-20,** 8 (1972).
4. K. Maruyama and D. T. Tang, "Discrete Link Capacity Assignment in Communication Networks," Proceedings of the Third International Computer Communication Conference, Toronto, Canada, August 3-6, 1976, p. 92.
5. G. Hadley, *Non-Linear and Dynamic Programming*, Addison-Wesley Publishing Co., Reading, MA, 1964.
6. L. Fratta and M. Gerla, "The Synthesis of Computer Networks: Properties of the Optimum Solution," presented at *ACM-International Computing Symposium*, Venice, Italy, April 1972.
7. M. Gerla, "The Design of Store-and-Forward Networks for Computer Communications," Ph.D. Thesis, School of Engineering and Applied Science, University of California, Los Angeles, 1973.
8. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design," *AFIPS Conference Proceedings*, Atlantic City, NJ, 1970, p. 569.
9. A. O. Allen, "Elements of Queuing Theory for System Design," *IBM Syst. J.* **14,** 161 (1975).
10. L. Kleinrock, *Queuing Systems Volume 1: Theory*, John Wiley and Sons, Inc., New York, 1975.
11. L. Kleinrock, *Queuing Systems Volume 2: Applications*, John Wiley and Sons, Inc., New York, 1976.
12. As a consequence of "nondominance," entries are also ordered in decreasing delay.
13. H. Frank, B. Rothfarb, D. Kleitman, and K. Steiglitz, "Design of Economical Offshore Natural Gas Pipeline Network," *Report R-1, Office of Emergency Preparedness*, Washington, DC, 1969.
14. H. Frank, I. T. Frisch, W. Chou, and R. V. Slyke, "Optimal Design of Centralized Computer Networks," *Networks* **1,** 43 (1971).
15. The average path length l_k of packet class k may be defined as $l_k = (\Sigma_j \lambda_{jk})/\gamma_k$. See references [1, 11].
16. D. E. Knuth, *The Art of Computer Programming Volume 1: Fundamental Algorithms*, Addison-Wesley Publishing Co., Reading, MA, 1968.

Received August 12, 1976; revised December 6, 1976

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.