Buffer Allocation Strategies with Blocking Requirements

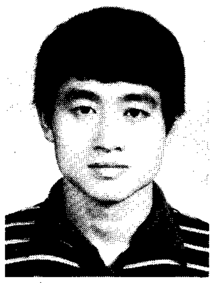
Takshing P. Yum
Dept. of Electronics, Chinese University of Hong Kong, Hong Kong

C. Dou
Dept. of Electrical Eng., National Taiwan University, Taiwan

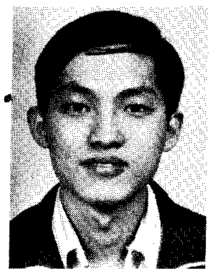
Received 22 February 1983
Revised 26 October 1983

The problem of choosing buffer allocation strategies occurs in the design of any store-and-forward computer network. A good buffer allocation strategy can reduce message blocking; and hence provide more efficient use of network storage resources. We first summarize five buffer allocation strategies and then provide algorithms for determining the minimum buffer sizes required by these strategies given that each outgoing channel must satisfy certain blocking requirements. After that, we compare them under different network conditions such as heavy or light input traffic rate, uniform or non-uniform server utilization and different blocking requirements. Guidelines on which strategy to use under different conditions are also given.

Keywords: Network Nodes, Buffering, Buffer Allocation Strategies, Optimization Algorithms, Blocking Probabilities, Blocking Constraints



Tak-Shing P. Yum was born in Shanghai, China in Feb. 1953. He received the B.S., M.S., and the Ph.D. degrees from Columbia University, New York in 1974, 1975, and 1978 respectively.
From 1978 to 1980, he was a member of the Technical Staff at Bell Telephone Laboratories, Holmdel, New Jersey and worked on the performance analysis of the Common Channel Signaling Network. In September 1980, he joined the National Chiao-Tung University and was an associate professor in the Institute of Computer Engineering. Since July 1982, he is a lecturer in the Department of Electronics in the Chinese University of Hong Kong.



Chie Dou was born in Tainan, Taiwan, Republic of China on August 11, 1956. He received the B.S. and M.S. degrees, both in communications engineering, from National Chiao-Tung University, Taiwan, in 1979 and 1981 respectively. He is currently a Ph.D. candidate at the Electrical Engineering Department in the national Taiwan University. His research interests include computer communications, queueing networks, and discrete mathematics.

1. Introduction

A computer-communication network consists of a set of nodes where computers and data-bases reside and a set of links through which the nodes communicate. The design of such a network is extremely complex due to a wide range of operating modes that have to be considered and a large number of design parameters that have to be set. In this paper, we look at the problem of how to allocate a finite buffer for storage in a computer network node. In particular, we want to know what strategy to use in different conditions and how to minimize the total buffer size such that certain blocking requirements are satisfied.

Consider a typical node in a network as shown in Fig. 1. Here, we see two buffering regions. The input buffer is used to store temporarily the received messages from other nodes and the output buffer is used as a waiting room for the messages (arrived from neighboring nodes or generated locally) destined for other nodes and are to be sent via the output channels.

Analysis of some input buffering strategies can be found in [1]–[3]. Concerning the study of output buffering strategies, Kamour and Kleinrock [4] have studied recently five output buffer sharing strategies. The five strategies will form the basis of our continued investigation. They are:

a) Complete Partitioning (CP): The entire finite storage is permanently partitioned among the L servers.

b) Complete Sharing (CS): Storage are assigned to arriving messages if any is available, independent of the servers to which these messages are directed.

c) Sharing with Maximum Queue Length (SMXQ): A limit is imposed on the number of buffers to be allocated to each server.

d) Sharing with a Minimum Allocation (SMA): A minimum number of storages is always reserved for each server and in addition, a common pool of buffer is to be shared among all servers.

e) Sharing with a Maximum Queue length and a Minimum Allocation (SMQMA): The strategy that combines SMXQ and SMA.

They derived the blocking probabilities and average time delays for the first four strategies and compared the first four strategies on a network node with two output channels, total buffer size equals to six and under symmetric loading. They then concluded that sharing with appropriate restrictions on the contention for space is very much desirable.

Earlier studies on similar strategies can be found in [5]–[8]. Lam [6], in particular, analyzed the CS strategy with nodal functions such as time-out, acknowledgement and retransmission for both a single node and in a network environment. Irland [7] proposed a square-root rule for choosing a common queue limit for all channels in the SMXQ strategy. He then obtained the optimum common queue limit by exhaustive search and showed that the loss probability obtained by using the square-root rule is very close to that obtained by using the optimum common queue limit policy. A similar square-root rule is proposed by Latouche [8] for choosing a *common* minimum allocation for all channels in the SMA strategy.

More recently, Kaufman [9] showed that the state distribution of the model obtained by previous authors assuming exponential message length distribution is in fact valid for arbitrary message length

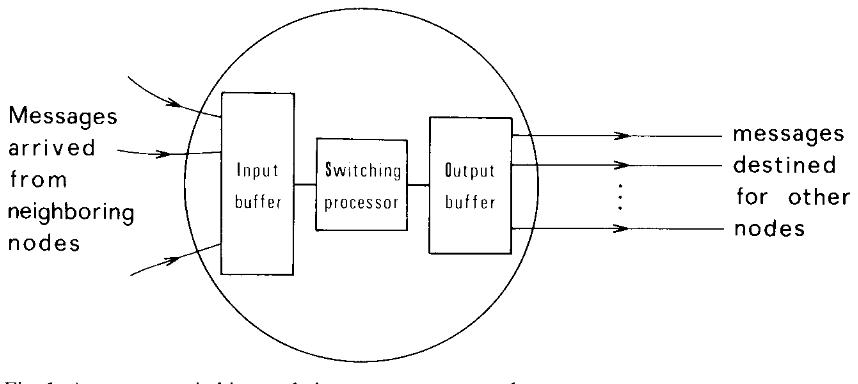

Messages arrived from neighboring nodes
Input buffer
Switching processor
Output buffer
messages destined for other nodes

Fig. 1. A message switching node in a computer network.

distributions having rational Laplace transforms. He also showed that the state distribution holds for completely arbitrary resource sharing policies. Schwartz and Kraimeche [10] and Kraimeche and Schwartz [11] treat related subjects in a more general fashion, with improved computational algorithms for system performance.

As will be shown in Section 2, we formulate the buffer allocation problem in the reverse way. That is, given the blocking requirements for each channel, we want to find the minimum buffer size required and the best strategy (among the proposed) that achieves this minimum under various conditions such as asymmetric input traffic rates and different channel capacities. For each strategy, we need also to optimize with respect to their associated parameters such as the set of maximum queue lengths $\{b_i\}$ for SMXQ and the set of minimum allocations $\{a_i\}$ for SMA.

We now present a model for the five allocation strategies. Consider L $M|M|1$ queues sharing a buffer of size B units (one unit per message). Queue i ($i = 1,,\ldots,L$) is characterized by a Poisson input stream of rate λ_i, and an exponential service time of mean $1/\mu C_i$ with C_i being the channel capacity (bits/s) and $1/\mu$ the average message length (in bits). Customers served by server i are referred to as the type-i customers, and blocked customers are lost.

The above queueing system can be described by a birth-death process with state vector $n = (n_1, n_2,\ldots,n_L)$ where n_i denotes the number of type-i customers in the system. The equilibrium state equation has the following product form solution [12]:

$$P(n) = \begin{cases} C_x \rho_1^{n_1} \cdots \rho_L^{n_L} & \text{for } n \in F_x \\ 0 & \text{otherwise} \end{cases}$$

where $\rho_i \triangleq \lambda_i/\mu C_i$, the subscript $x (x \in \{a, b, c, d, e\})$ refers to each of the five buffer allocation schemes, and F_x is the set of all possible system states with the use of scheme x.

In [4], the authors derived, for the first four schemes, 1) the normalization constant C_x, 2) the probability of blocking, 3) the system throughput and 4) the average delay. The results on SMA and SMXQ strategies have certain restrictions and cannot be used in the strategy optimization problem described above. We supply generalizations of the results in the appendix. In particular, the SMA analysis is generalized to include (1) the case where some or all the minimum allocations a_i's may be zero and (2) the case where not all ρ_i's are different. For SMXQ analysis, we derive the results without the assumption made in [4] that all $b_i \geqslant B/2$. We do this first for the case where all ρ_i's are different, and then for the case where all ρ_i's are the same. The analysis of the SMQMA strategy is not found in [4]. We found that its derivation with the above mentioned generalizations is a straight forward but tedious extension of SMXQ and SMA analyses; and so would refer the readers to [13] for full details.

2. Algorithms for optimizing buffer allocation strategies

The equations derived in [4] supplemented by those derived in the appendix allow us to calculate the performance measures $\{PB_i\}$ and $\{T_i\}$ (the set of blocking probabilities and the set of delays on each channel) given the set of parameters $\{B, \{\lambda_i\}, \{C_i\}, \{a_i\}, \{b_i\}\}$. In the actual design, however, we are usually asked to solve the reverse problem. That is, given the set of blocking and delay requirements such as $\{PB_i \leqslant PB_i^*\}$ and $\{T_i \leqslant T_i^*\}$, find the minimum buffer size required and the best strategy (CS, CP, SMXQ, SMA) that will achieve this minimum, together with the set of optimum $\{a_i\}$ or $\{b_i\}$.

Note that the delay constraints cannot generally be satisfied by the increasing or the decreasing of the buffer size. Hence we assume here that we have already solved the delay problem by choosing $\{C_i\}$ large enough to have the $\{T_i \leqslant T_i^*\}$ satisfied. Note also that the SMQMA strategy requires the minimization of B with respect to both $\{a_i\}$ and $\{b_i\}$. We have not found a good way of doing it.

a) CP optimization algorithm

For partitioned buffers, each channel is a finite storage $M|M|1$ queue. For such a queue with capacity K, it can be shown that the blocking probability PB is given by [14]:

$$PB = \frac{(1-\rho)\rho^K}{1-\rho^{K+1}}.$$

Let x_i be the solution to the following equation:

$$PB_i^* = \frac{(1-\rho_i)\rho_i^{x_i}}{1-\rho_i^{1+x_i}}.$$

Solving for x_i, we have

$$x_i = \frac{1}{\ln \rho_i}\left[\ln \frac{PB_i^*}{1-\rho_i} - \ln\left(1 + \frac{PB_i^* \rho_i}{1-\rho_i}\right)\right].$$

Let B_i^* be the required buffer size for channel i, then $B_i^* = \lceil x_i \rceil$. The total required buffer size is the sum of B_i's.

b) CS optimization algorithm

For CS, SMXQ and SMA strategies, we use the bisection method to search for the optimum total buffer size B^*. We begin with an initial feasible upper bound BU and an initial infeasible lower bound BL on B^*. If the midpoint between BU and BL satisfies the constraints, we set this midpoint as the new upper bound. Otherwise, we set it as the new lower bound. Since PB is a monotonically decreasing function of B, repeated bisection will eventually lead to a BU which barely satisfies the set of blocking constraints. This BU therefore is the required B^*.

For SMXQ and SMA strategies, $\{b_i\}$ and $\{a_i\}$ also affect the blocking probabilities. Hence they need to be optimized as well with different algorithms. These algorithms will then be subalgorithms in the above bisection algorithm.

For CS scheme, $PB_i = PB$ for all i. To satisfy $\{PB^*\}$, we need only to consider the tightest constraint. let $PB^* = \min[PB_i^*]$. The constraints $\{PB_i \le PB_i^*\}$ is then transformed to $PB \le PB^*$.

We may let $B^*(\text{CP})$ be the initial upper bound. To find the initial lower bound, consider the two queueing systems shown in Fig. 2. By the resource sharing theorem [15], we know that system B always performs better than system A. The optimum total buffer size required by system B is therefore a lower bound on that for system A.

c) SMXQ optimization algorithm

Let $\{b_i^*\}$ be the set of limits of queue size that achieves B^*. We notice that $B^* \le B^*(\text{CP}) = BU$, and $B^* \ge \max[B_i^*(\text{CP}] = BL$. Moreover, since $b_i^* \ge B_i^*(\text{CP})$, $i = 1,\dots,L$, we choose $\{B_i^*(\text{CP})\}$ as the lower bound on $\{b\}$. The initial upper bound can be set simply as $b_i^* \le B^*(\text{CP})$, $i = 1, 2,\dots,L$.

Next, we determine the feasibility of a new B value by checking if there exists a set $\{b_i\}$ that satisfies the constraints. We distinguish two cases here:

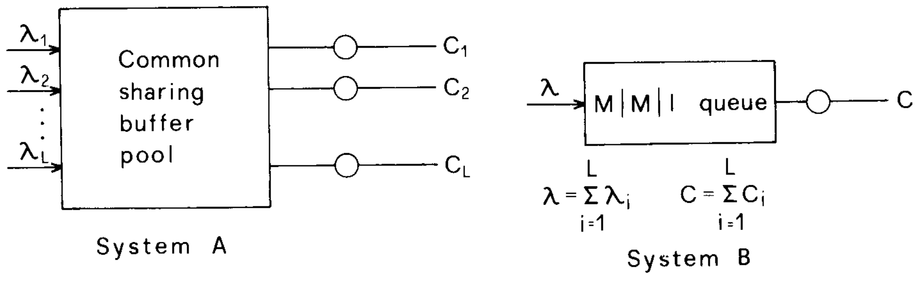


Fig. 2. The Common Sharing and the $M/M/1$ systems.

(i) If such a set $\{b_i\}$ exists, then B is feasible and becomes the new upper bound. We then turn to search for the particular set of $\{b_i\}$ which minimally (or marginally) satisfies the constraints. That set of $\{b_i\}$ is then the new lower bound on $\{b_i^*\}$.

(ii) If such a set does not exist, then B is infeasible and becomes a tighter lower bound. The bounds on $\{b_i\}$ remains the same.

We now address the question of finding the new lower bound on b_i^*. Decreasing the buffer size will cause all PB_i to increase. But a particular PB_i can be decreased by increasing its associated b_i. This increase of b_i, however, will cause all PB_k, $k \neq i$, to increase. Thus for channel k that violates the constraint, we use the bisection method to search for the minimum b_k that just satisfies $PB_k \leq PB_k^*$. We again distinguish two cases here:

1) If such a minimum b_k exists, we turn to 'fix' the next channel that violates the constraint (notice that the set of channels with constraints violated includes those induced by the increase of b_k). If all constraints are satisfied with an appropriate choice of $\{b_i\}$, we have case (i) above.

2) If such a minimum b_k does not exist for some channel, i.e., there exists k, such that when b_k is increased to B, the kth constraint is still not satisfied, we have case (ii) above.

d) SMA optimization algorithm

Bisection method is again used for the search of $B^* = B^*$ (SMA) and $\{a_i^*\}$ here. The procedure parallels the SMXQ algorithm. To determine whether all a_i^* exist for a reduced buffer size B, we increase the minimum allocations of all channels that violate the blocking requirements (obviously, PB_i decreases as a_i increases) and check if the size of the common buffer pool $(B - \Sigma_{i=1}^{L} a_i)$ is less than $\max[B_i^*(\text{CP}) - a_i]$. If yes, $\{a_i^*\}$ does not exist for this buffer size and B is deemed infeasible. Zero and $B_i^*(\text{CP})$ are obviously the upper and lower bounds on a_i. Note that when all a_i equal to zero, SMA degenerates to CS.

3. Comparisons of strategies

We now use the algorithms developed in the last section to find B^* for each of the four strategies and compare them under various conditions. The $B^*(.)$ results for equal ρ_i, $L = 4$ and various combinations of blocking probabilities are shown in Table 1. We see that when only one PB_i^* is tight, SMA will allocate a_i buffers to that channel, giving it improvement of B^* performance over CS. When half or more of the PB_i^* are tight, this improvement is diminished. The SMXQ, on the other hand, continues to give improvement over the CS strategy by limiting the queue sizes of the channels with looser bounds. We also showed the values of the $\{a_i^*\}$ and $\{b_i^*\}$ that achieve these optimum buffer sizes. We note in particular, that a_i^* is often equal to zero and b_i^* is often less than $B^*/2$, thus justifying the need for the additional derivations in the appendix.

Table 2 shows B^* for $L = 4$, asymmetric input rates and all $PB_i^* = 10^{-5}$. We found that $B^*(\text{CS}) = B^*(\text{SMXQ}) = B^*(\text{SMA})$. Thus as long as all PB_i^* are the same, the choice of strategy (except CP) is insensitive to the asymmetry of the input traffic.

For $L = 4$ and when input rates and blocking requirements are all different, we distinguish two cases:

Case A

Some channels have small ρ_i and tight constraints (others have relatively larger ρ_i and relatively looser constraints). The results for this case is shown in Table 3 and the followings are observed:

1) Only one channel has small ρ_i and tight constraint: Here SMA is indisputably better than SMXQ. In other words, SMA strategy appears to be superior when there is a single light 'user' demanding a stringent blocking requirement.
2) Half of the channels have small ρ_i and tight constraints: Here, the performance of SMA and SMXQ are very close to each other, but SMA is still better.
3) Three of the channels have small ρ_i and tight constraints and one channel has a single heavy user with a loose blocking constraint: Under this conditions, SMXQ shows its superiority over SMA. Thus we should always limit the maximum queue size of a heavy user with a loose constraint.

Table 1
Minimum buffer sizes ($L = 4$)

ρ	PB_i^*	B^*											
		CP	CS	SMA	a_1^*	a_2^*	a_3^*	a_4^*	SMXQ	b_1^*	b_2^*	b_3^*	b_4^*
	E_0	24	12	12	0	0	0	0	12	7	7	7	7
0.5	E_1	37	29	25	0	0	0	13	25	6	6	6	20
	E_2	50	29	29	0	0	0	0	26	6	6	23	23
	E_3	63	29	29	0	0	0	0	28	6	20	20	20
	E_4	76	29	29	0	0	0	0	29	20	20	20	20
	E_0	40	21	21	0	0	0	0	21	12	12	12	12
	E_1	66	55	46	0	0	0	27	46	10	10	10	39
0.7	E_2	92	55	54	0	0	1	1	50	10	10	38	38
	E_3	118	55	55	0	0	0	0	52	10	41	41	41
	E_4	144	55	55	0	0	0	0	55	39	39	39	39
	E_0	72	41	40	1	1	1	1	40	21	21	21	21
	E_1	138	116	94	0	0	0	60	94	18	18	18	81
0.85	E_2	184	116	115	0	0	2	2	103	18	18	83	83
	E_3	240	116	116	0	0	0	0	110	18	83	83	83
	E_4	296	116	116	0	0	0	0	116	84	84	84	84

E_0 = All $PB_i^* = 10^{-2}$
E_1: $PB_1^* = PB_2^* = PB_3^* = 10^{-2}$, $PB_4^* = 10^{-6}$
E_2: $PB_1^* = PB_2^* = 10^{-2}$, $PB_3^* = PB_4^* = 10^{-6}$
E_3: $PB_1^* = 10^{-2}$, $PB_2^* = PB_3^* = PB_4^* = 10^{-6}$
E_4: All $PB_i^* = 10^{-6}$

Table 2
Minimum buffer sizes ($PB^* = 10^{-5}$)

ρ_1	ρ_2	ρ_3	ρ_4	B^*	
				CP	CS, SMXQ, SMA
0.1	0.3	0.5	0.7	60	32
0.1	0.15	0.2	0.25	28	10
0.15	0.2	0.25	0.8	68	46
0.15	0.2	0.7	0.8	88	49
0.2	0.7	0.75	0.8	118	55
0.7	0.75	0.8	0.85	170	75

Table 3
Minimum buffer sizes ($L = 4$)

Case	ρ_1	ρ_2	ρ_3	ρ_4	B^*			
					CP	CS	SMA	SMXQ
	0.15	0.7	0.75	0.8	95	76	48	67
1	0.3	0.7	0.75	0.8	100	76	50	67
	0.5	0.7	0.75	0.8	109	77	55	68
	0.15	0.2	0.75	0.8	82	72	49	62
2	0.3	0.35	0.75	0.8	92	73	53	63
	0.5	0.55	0.75	0.8	112	75	64	65
	0.15	0.2	0.25	0.8	66	66	49	42
3	0.3	0.35	0.4	0.8	82	67	56	45
	0.45	0.5	0.55	0.8	104	69	67	51

Case 1): $PB_1^* = 10^{-7}$, $PB_2^* = PB_3^* = PB_4^* = 10^{-4}$
Case 2): $PB_1^* = PB_2^* = 10^{-7}$, $PB_3^* = PB_4^* = 10^{-4}$
Case 3}: $PB_1^* = PB_2^* = PB_3^* = 10^{-7}$, $PB_4^* = 10^{-4}$

Table 4
Minimum buffer sizes ($L = 4$)

Case	ρ_1	ρ_2	ρ_3	ρ_4	B^*	
					CP	CS, SMA, SMXQ
	0.15	0.2	0.25	0.8	59	46
1	0.35	0.4	0.45	0.8	67	47
	0.45	0.5	0.55	0.8	73	49
	0.2	0.25	0.75	0.8	91	52
2	0.5	0.55	0.75	0.8	101	54
	0.65	0.7	0.75	0.8	111	58
	0.2	0.7	0.75	0.8	115	55
3	0.3	0.7	0.75	0.8	116	55
	0.5	0.7	0.75	0.8	119	56

Case 1: $PB_1^* = PB_2^* = PB_3^* = 10^{-3}$, $PB_4^* = 10^{-5}$
Case 2): $PB_1^* = PB_2^* = 10^{-3}$, $PB_3^* = PB_4^* = 10^{-5}$
Case 3): $PB_1^* = 10^{-3}$, $PB_2^* = PB_3^* = PB_4^* = 10^{-5}$

Note that the amount of improvement (either SMA over SMXQ or SMXQ over SMA) depends also on the differences of the blocking requirements.

Case B

This is the reverse of case A, i.e., some channels have small ρ_i and loose constraints while the others have large ρ_i and tight constraints. Some specific cases are shown in Table 4. Here, the channels with smaller ρ_i (< 0.7) have $PB^* = 10^{-3}$ and the channels with larger ρ_i (≥ 0.7) have $PB^* = 10^{-5}$. Under this condition, we have $B^*(\text{CS}) = B^*(\text{SMXQ}) = B^*(\text{SMA})$, or the choice of strategy (except CP) is not critical. For all cases shows, $a_i^* = 0$. The channel with the largest ρ_i, because of tight blocking constraint, has b_i^* very close to B^*. The others, with looser constraints, all have very small maximum queue limits. Thus for the case corresponding to the first row of Table 4, B^* is equal to 46 and b^* is equal to (4, 5, 5, 45).

4. Summary

After reviewing the five buffer allocation strategies found in [4], we presented, in the appendix some generalizations of the SMA and SMXQ analysis. We then proposed the design problem in Section 2 and gave four algorithms for the optimization of the four strategies studied. These algorithms are then used to obtain the performance comparisons in Section 3. Among the major conclusions are:

1) As long as the blocking constraints are the same for all channels, the choice of strategy (except CP) appears to be insensitive to the asymmetry of the input traffic.

2) When there exists a heavy user with a loose blocking constraint (in relative sense), SMXQ appears to be superior to SMA.

3) When there exists a light user with a tight constraint, SMA appears to be superior to SMXQ.

4) When no user as described in 2) and 3) is present, the SMXQ, SMA and CS all appear to give comparable performance.

5) From 2) and 3) we suspect that when there exists a few heavy users with loose constraints, a few light users with tight constraints as well as some users in between, SMQMA strategy would prove to be superior. Further study is needed, however, to make this statement conclusive.

Appendix

A. SMA strategy—supplementary derivation

The SMA strategy allows the sharing of a pool of B' buffers, and with a_i buffers permanently allocated to the type-i customers (total buffer size B is $B = B' + \Sigma_{i=1}^{L} a_i$). As a result, the set of feasible states becomes

$$F_d = \left\{ n \mid \sum_{i=1}^{L} \max[0, n_i - a_i] \le B', \quad 0 \le n_i \le B' + a_i \right\}. \tag{1}$$

Let $a_i \ge 0$ (rather than $a_i \ge 1$ as assumed in [4]), we evaluate C_d^{-1} by first partitioning F_d into disjoint subsets as follows:

Let $\mathscr{L} = \{1, 2, \ldots, L\}$ and $A = \{i \mid a_i > 0, i \in \mathscr{L}\}$. Let $A' \triangleq \mathscr{L} - A$, or $A' = \{i \mid a_i = 0, i \in \mathscr{L}\}$. Further, let $\#A$ be the number of elements in set A. Then the number of all possible subsets of A is $2^{\#A}$; and we denote the set of all possible subsets of A as X where $X = \{X_1, X_2, \ldots, X_2\#A\}$. Define $Y_m \triangleq X_m \cup A'$, $m = 1, 2, \ldots, 2^{\#A}$. We associate with each Y_m a subset of F_d defined as

$$S_m = \{ n \in F_d \mid n_i \geqslant a_i \quad i \in Y_m;\ n_i < a_i \quad i \notin Y_m \}. \tag{2}$$

Note that S_m is the set of all feasible states with queues that belong to Y_m having lengths greater or equal to their respective minimum allocations a_i and the rest (i.e., $i \notin Y_m$) less than a_i. It is readily shown that 1) $S_m \cap S_n = \varnothing$ for $m \ne n$ and 2) $F_d = \cup_{m=1}^{2^{\#A}} S_m$. Therefore,

$$C_d^{-1} = \sum_{n \in F_d} (\rho_1^{n_1} \ldots \rho_L^{n_L}) = \sum_{m=1}^{2^{\#A}} \sum_{n \in S_m} (\rho_1^{n_1} \ldots \rho_L^{n_L}). \tag{3}$$

The equation for the case where all ρ_i's are different is identical to that derived in [4] with X_m replaced by Y_m and R by $\#A$. We now derive the result for all $\rho_i = \rho$ and all $a_i \ge 0$. For this case, we first define $C_m(k)$ such that

$$C_m(k) = \sum_{n \in D} \left[\Pi_{i \in Y_m} \rho^{n_i} \right], \tag{4}$$

where $D = \{ n \mid \sum_{i \in Y_m} n_i = k \}$.

Let $\#Y_m$ be the number of elements in set Y_m. Then

$$C_m(k) = \binom{k + \#Y_m - 1}{\#Y_m - 1} \rho^k. \tag{5}$$

From (3) and the definition of S_m, we have

$$C_d^{-1} = \sum_{m=1}^{2^{\#A}} \left[\Pi_{i \notin Y_m} \sum_{l=0}^{a_i - 1} \rho^l \right] \left[\Pi_{i \in Y_m} \rho^{a_i} \sum_{n \in D'} \left(\Pi_{i \in Y_m} \rho^{n_i} \right) \right],$$

where

$$D' = \left\{ n \mid \sum_{i \in Y_m} n_i \le B' \right\}.$$

Noticing that the last summation is just $\sum_{k=0}^{B'} C_m(k)$, we have finally

$$C_d^{-1} = \sum_{m=1}^{2^{\#A}} \left[\Pi_{i \notin Y_m} \left(\frac{1 - \rho^{a_i}}{1 - \rho} \right) \Pi_{i \in Y_m} (\rho^{a_i}) \sum_{k=0}^{B'} \binom{k + \#Y_m - 1}{\#Y_m - 1} \rho^k \right] \tag{6}$$

which is readily computed.

The case where only some ρ_i's are equal and others different can also be treated. Define E as the set of servers with equal ρ, or

$$E = \{ i \mid \rho_i = \rho \}$$

and let E^c and Y_m^c be the complement of E and Y_m respectively; in other words,

$$E^c = \mathscr{L} - E$$

and

$$Y_m^c = \mathscr{L} - Y_m.$$

Then, for each m in the summation of (3), we partition the set of servers into four disjoint subsets: $Y_m \cap E$, $Y_m \cap E^c$, $Y_m^c \cap E$, and $Y_m^c \cap E^c$ and partition the products in (3) according to these four subsets. Thus if there are D sets of equal ρ_i's, we can similarly define

$$E_k = \{i \mid \rho_i = d_k\}, \quad k = 1, 2, \ldots, D$$

and partition the set of servers into 2^{D+1} disjoint subsets.

B. *SMXQ strategy—supplementary derivation*

Evaluation of $Q(K)$, defined as $Q(K) = \Sigma_{n \in D} \rho_1^{n_1} \ldots \rho_L^{n_L}$ where $D = \{n \mid \Sigma_{i=1}^L n_i = K, 0 \le n_i \le b_i\}$, is central to the analysis of SMXQ strategy. In this subsection, we derive methods for calculating $Q(K)$ without assuming all $b_i \ge B/2$. We do this first for all ρ_i's different and then for all ρ_i's equal. These results will be used in the next section for searching the optimum queue size limit b for each server. Our point of departure for all ρ_i's different is (24) of [4]:

$$\sum_{K=0}^{\infty} G(K) t^K = \left(\sum_{K=0}^{\Sigma b_i} Q(K) t^K \right) \sum_{i=1}^{L} \left\{ C_i \left[1 + (\rho_i t)^{1+b_i} + \ldots + (\rho_i t)^{k(1+b_i)} + \ldots \right] \right\}. \tag{7}$$

We proceed by first collecting terms of the same powers of t of the second summation on the RHS and write it as

$$Y(t) = \sum_{k=0}^{\infty} y(k) t^k,$$

with

$$y(k) = \sum_{(i,j) \in D(k)} C_i \rho_i^{j(1+b_i)} \tag{8}$$

and

$$D(k) = \{(i,j) \mid j(1+b_i) = k\}.$$

We can easily see from (8) that $y(0) = \Sigma_{i=1}^L C_i = 1$. The remaining $y(i)$'s can be evaluated from (8) by the following algorithm.

```
BEGIN
        $k_i := \frac{B}{1+b_i}$                          $i = 1, 2, \ldots, L;$
        $y(i) := 0$                                       $i = 1, 2, \ldots, B;$
        FOR $i := 1$ to $L$ DO
           FOR $j := 1$ to $k_i$ DO
              $y(j(1+b_i)) := y(j(1+b_i)) + C_i \rho_i^{j(1+b_i)};$
END
```

With $y(k)$'s found, we can substitute $Y(t)$ into (7), equating equal powers of t and obtain the following recursive formula for $Q(K)$:

$$Q(K) = G(K) - \sum_{j=0}^{K-1} Q(j) y(K-j), \quad K = 0, 1, \ldots, B. \tag{9}$$

For the case all $\rho_i = \rho$, we have

$$Q(K) = \sum_{\substack{\Sigma n_i = K \\ 0 \le n_i \le b_i}} \rho_1^{n_1} \ldots \rho_L^{n_L} = \rho^K \sum_{\substack{\Sigma n_i = K \\ 0 \le n_i \le b_i}} 1 \triangleq \rho^K Q'(K). \tag{10}$$

Define

$$q(t) \triangleq \Pi_{i=1}^{L}(1+t+\ldots t^{b_i}). \tag{11}$$

Expanding the product, we recognize $Q'(K)$ as the coefficient of t^K, or

$$q(t) = \sum_{K=0}^{\Sigma b_i} Q'(K) t^K. \tag{12}$$

The following is a recursive procedure for representing $q(t)$ as a power series, and from that $Q'(K)$ can be picked out. Define

$$Y_0(t) = 1$$

and

$$Y_n(t) = Y_{n-1}(t) \sum_{k=0}^{b_n} t^k, \quad n = 1, 2, 3, \ldots, L. \tag{13}$$

Then

$$Y_L(t) = \Pi_{i=1}^{L} \sum_{k=0}^{b_i} t^k = q(t). \tag{14}$$

Let the power series expression for $Y_n(t)$ be

$$Y_n(t) = \sum_{l=0}^{b_i} y_{n,l} t^l, \quad n = 1, 2, \ldots, L \tag{15}$$

then from (13), we have

$$Y_n(t) = \sum_{j=0}^{\Sigma b_i} y_{n-1,j} t^j \sum_{k=0}^{b_n} t^k. \tag{16}$$

In (16), let $l = j + k$, then $k = l - j$; and upon substituting into (16), we have

$$Y_n(t) = \sum_{l=0}^{\Sigma b_i} \left[\sum_{j=\max(0,\, l-b_n)}^{l} Y_{n-1,j} \right] t^l. \tag{17}$$

Compare with (15), we have

$$Y_{n,l} = \sum_{j=\max(0,\, l-b_n)}^{l} Y_{n-1,j}, \quad n = 2, 3, \ldots, L \tag{18}$$

Starting from $y_{0,0} = 1$, we have from (18)

$$y_{1,l} = \begin{cases} 1 & l = 0, 1, 2, \ldots, b_1, \\ 0 & \text{otherwise.} \end{cases}$$

Repeated use of (18) gives $\{y_{2,l}\}, \ldots, \{y_{L,l}\}$. Finally, from (10) and (14), we have

$$Q(K) = Q'(K)\rho^K = y_{L,K}\rho^K. \tag{19}$$

References

[1] D.P. Gaver and P.A. Lewis, Probability models for buffer storage allocation problems, JACM 18 (1971) 186–198.

[2] G.D. Schultz, A stochastic model for message assembly buffering with a comparison of block assignment strategies, JACM 19 (1972) 483–495.

[3] J.H. Chang, An analysis of buffering techniques in teleprocessing systems, IEEE Trans. Commun. COM-20 (1972) 619–630.

[4] F. Kamoun and L. Kleinrock, Analysis of shared finite storage in a computer network node environment under general traffic conditions, IEEE Trans. Commun. COM-28 (1980) 992–1003.

[5] M. Rich and M. Schwartz, Buffer sharing in computer-communication network nodes, IEEE Trans. Commun. COM-25 (1977) 958–970.

[6] S. Lam, Store-and-forward buffer requirements in a packet switching network, IEEE Trans. Commun. COM-24 (1976) 394–403.

[7] M.I. Irland, Buffer management in a packet switch, IEEE Trans. Commun. COM-26 (1978) 328–337.

[8] G. Latouche, Exponential servers sharing a finite storage: comparison of space allocation policies, IEEE Trans. Commun. COM-28 (1980) 910–915.

[9] J.S. Kaufman, Blocking in a shared resource environment, IEEE Trans. Commun. COM-29 (1981) 1474–1481.

[10] M. Schwartz and B. Kraimeche, Comparison of channel assignment techniques for hybrid switching, in: Proc. ICC, Philadelphia, PA, USA, June 1982.

[11] B. Kraimeche and M. Schwartz, Circuit access control strategies in integrated digital networks, INFOCOM '84, San Francisco, CA, USA, April 1984.

[12] H. Kobayashi, Modeling and Analysis: An Introduction to System Performance Evaluation Methodology, (Addison-Wesley, 1978) 161–168.

[13] C. Dou, Performance evaluation algorithms for buffer allocation strategies in a computer network node, Master Thesis, Institute of Telecommunications, Chiao-Tung University, Taiwan, ROC, June 1981.

[14] L. Kleinrock, Queueing Systems Vol. I: Theory, (Wiley-Interscience, 1975) 104.

[15] L. Kleinrock, Queueing Systems Vol. II: Computer Applications, (Wiley-Interscience, 1976) 272–290.