The problem of allocating main storage for a message-segment buffer pool is considered. A queuing model that approximates the most typical mode of operation is formulated. Solutions for the number of buffers required by the pool are obtained. Although the solutions require iterative computational methods, they are not difficult to program.

Inasmuch as the model and solution methods both involve approximations, the validity of the approach was checked by simulating a typical set of operating conditions. Although the computational results were found to be conservatively biased, the method is clearly adequate for most design purposes.

On teleprocessing system design

Part III An analysis of a request-queued buffer pool

by J. P. Bricault and I. Delgalvis

Where each communication line transfers data at a rate well below the processing-unit rate, many lines are usually permitted to operate concurrently. Buffers then become necessary in coordinating the communication and processing functions. A buffer consists simply of a main-storage area reserved for message assembly and disassembly tasks.

A homogeneous set of buffers is called a *buffer pool*. Because a buffer pool may require a considerable portion of main storage, it is important to define a pool area that is reasonably close to the minimum permitted by application parameters. This paper postulates and analyzes a mathematical model that is applicable to a buffer pool of the class required for QTAM (Queued Telecommunications Access Method of OS/360). Methods of solving for an appropriate set of buffers are given.

Buffering principles

Storage may be allocated to a buffer on either a static or dynamic basis. In static allocation, a fixed amount of storage is assigned to each line. In dynamic allocation, the storage is pooled and each line requests a buffer from the pool as the need arises.

Static allocation is the simpler of the two schemes; however, it requires that the area assigned to a line accommodate the line's longest message. If such a message appears infrequently,

or if some lines have little traffic, static allocation leads to inefficient use of storage. Although dynamic allocation permits more efficient utilization of storage because storage is assigned to a line only when required, it involves a more elaborate control program. Thus, the choice between allocation schemes depends upon a typical kind of trade-off between allocated storage and control complexity. Systems transmitting long messages, or handling more than a few communication lines, usually employ a dynamic method of buffer allocation.

message segmentation

A common technique for reducing main-storage requirements in both allocation schemes involves *message segmentation*. Each segment of a message is then allocated a buffer in its own right. The most useful technique for keeping track of the message segments is chaining. Chaining requires of each buffer that a section be allocated to fields for a message identifier, a pointer to the buffer holding the previous segment, and other control information. Let this section be called the buffer *prefix*.

A message can be put together from its segments either in main storage or in direct-access auxiliary storage. In this paper, we make the rather reasonable assumption that main storage is to be conserved, and that, as a result, messages are reconstructed in auxiliary storage.

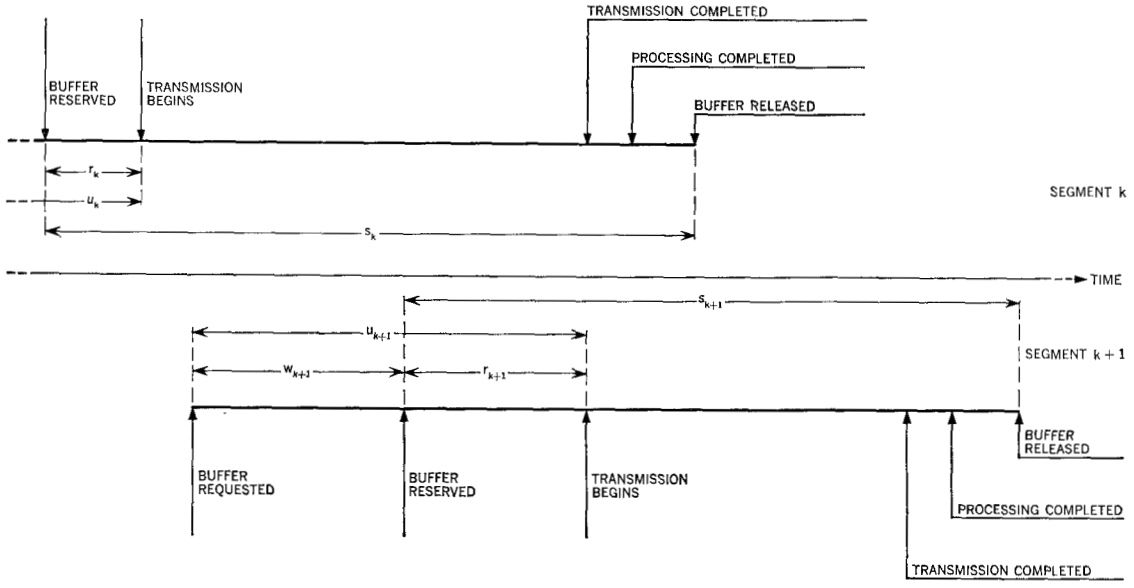
We assume that buffers are homogeneous. To ensure that a buffer is reserved for segment $k + 1$, a *buffer request* is issued during the interval in which segment k is transmitted. The request is queued and a *buffer reservation* occurs when the request is honored.

The reserved buffer lies idle from the instant of reservation to the instant at which the first character of segment $k + 1$ arrives. If u denotes the time interval from buffer request to buffer usage, and w denotes the waiting-time interval from buffer request to buffer reservation, then the assigned buffer is idle for $u - w$ units of time. Let r denote this *idle time*. One of the objectives of an efficient buffering scheme is to keep idle time reasonably close to a minimum.

One way to achieve a small r is to operate with a short, fixed u and a fixed w; Reference 1 discusses such a method. The purpose of the present paper is to discuss a more flexible method in which u can be large and w is a random variable that depends upon the arrival pattern of requests.

The *holding time*, s, of a buffer is an interval that starts when the buffer is reserved and ends when the buffer is finally liberated, i.e., returned to the subpool of available buffers. As shown in Figure 1, s is the sum of the idle time r and the application-dependent times for segment transmission, housekeeping, and movement between main storage and direct-access auxiliary storage. Because these times can vary among the first, last, and intermediate segments of a message, the average holding time, $\bar{s}$, is obtained as a weighted average over all the segments of a message. Later we show that $\bar{s}$ is related to the average buffer

Figure 1 Illustration of buffer holding time



request waiting time $\tilde{w}$ by the linear function $\tilde{s} = a - g\tilde{w}$. (The coefficients a and g are based on knowledge of the application.)

design criterion

The normal flow of data is broken if a buffer request for segment $k + 1$ is not honored by the time segment k has been transmitted. Such a *break* results in either a loss of data (for incoming transmission) or in a transmission gap (for outgoing transmission). It is seldom economically feasible to entirely eliminate all breaks, and some small probability η of a break is normally considered acceptable. Let $\hat{w}$ denote the maximum w that does not involve a break ($\hat{w} \leq u$, of course). The mathematical models to be discussed assume that

$$\Pr\{w > \hat{w}\} \leq \eta \tag{1}$$

Mathematical models

The mathematical problem is to determine P, the number of buffers (service facilities) required to serve buffer requests (customers) within a maximum waiting time $\hat{w}$ under the conditions prescribed in (1). The buffer requests are held in a single queue. Let $\bar{s}$ denote the average service time and λ the average buffer request rate. Then if the *traffic intensity* for the system is defined as $\lambda\bar{s}$, the *buffer utilization* ρ is $\lambda\bar{s}/P$. The model readily yields a solution if we assume a Poisson distribution of request arrivals and an exponential distribution of service time s (buffer holding time).

The probability β that an arriving request finds all P buffers busy is well known to be[2]

$$\beta = \frac{1 - \sigma}{1 - \sigma\rho} \tag{2}$$

where

$$\sigma = 1 - \left[\frac{(\lambda\bar{s})^P}{P!} \Big/ \sum_{i=0}^{P} \frac{(\lambda\bar{s})^i}{i!} \right]$$

The average waiting time $\bar{w}$ is likewise known to be

$$\bar{w} = \beta \frac{\bar{s}}{P - \lambda\bar{s}} \tag{3}$$

and the probability that a customer must wait more than t units of time for service to be

$$\Pr(w > t) = \beta e^{-(P - \lambda\bar{s})\, t/\bar{s}} \tag{4}$$

From (1) and (4), we have

$$\eta \geq \beta e^{-(P - \lambda\bar{s})\, \phi/\bar{s}} \tag{5}$$

and our problem is to find $P_{\min}$, the smallest P that satisfies (5).

The solution is complicated by the fact that the average service time $\bar{s}$ is related to w, which in turn is dependent upon $\bar{s}$ and P. The circularity can be handled by iteratively determining from (5) a first approximate value of P satisfying the design criterion, assuming the service time to be independent of the waiting time and reduced to its constant term $\bar{s} = a$. Then, an approximate computation of the average wait $\bar{w}$ is possible. The service is subsequently corrected by $\bar{s} = a - g\bar{w}$. Based on this corrected service, a new value of P is iteratively computed, and the process is repeated until two successive values of P converge to the correct value. This procedure is illustrated in Figure 2.

Solution 1 yields highly overconservative results. The assumption of a Poisson distribution for customer arrivals is generally sound unless the number of lines is very small, but the service times normally vary much less than expected of an exponential distribution. It would be more realistic, in fact, to assume that service time was constant. For just such cases, Everett[3] has given a method that simulates the desired effect of constant service time by introduction of a traffic reduction factor f in a relation of the form

$$f = \delta/\rho \qquad \text{where} \qquad \delta = e^{\rho[1-(1/\delta)]} \tag{6}$$

The trend in δ can be seen from Table 1.

The procedure for finding the desired P is similar to the one described previously, except that we must find f in addition. Since the necessary buffer utilization ρ is not available at this time, an iterative procedure must be employed to find a realistic

Figure 2 Flowchart for solution 1

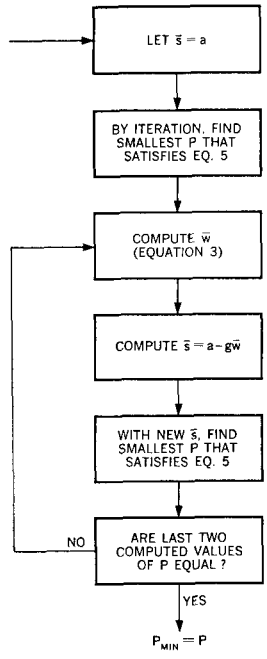
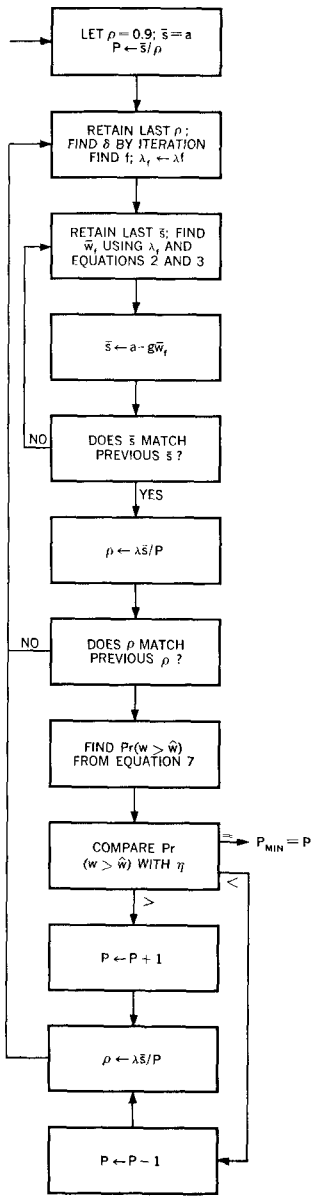


Table 1 Delta as a function of rho

ρ	δ
.6	.3877
.625	.4163
.65	.4460
.675	.4768
.7	.5089
.725	.5422
.75	.5768
.775	.6127
.8	.6499
.825	.6885
.85	.7285
.875	.7683
.9	.8128
.925	.8572
.95	.9031
.975	.9505
1.0	1.0

Figure 3 Flowchart for solution 2



number of buffers and their utilization. Then, a probability computation will show whether the proper condition on η is satisfied. The whole procedure is illustrated in Figure 3. The auxiliary equation required in the computations is

$$\Pr(w > \hat{w}) = \beta_f e^{-(P-\lambda\bar{s}_f)\hat{w}/(a-g\bar{w}_f)} \tag{7}$$

where the subscript f denotes that the waiting time in the subscribed variable is based on the reduced traffic rate $f\lambda$.

Although Solution 2 is still an approximation, experience has shown that it yields solutions of acceptable accuracy. A more accurate formulation of the problem would lead to a multiserver queuing model with a general service time—and at present an exact solution to this problem does not exist. Other approximate solutions could have been tried, but investigation showed them to be more cumbersome than desired.

An example of a message-switching system

Consider a message switching system that receives, interprets, and forwards messages to their destinations. To avoid superfluous detail, we assume that each message is sent to only one destination. Each terminal, including the processing unit, competes for line usage. The system contains L half-duplex lines and the line speed K is 14.8 characters per second. The messages are queued on a disk file. The messages in the system are characterized by an average message length $\bar{M}$ of 250 characters and an average message rate R of 2.83 messages/minute/line. The buffer size is 50 characters.

When recognized, the terminal issues a buffer request for Segment 1. The first 32 character spaces of this buffer are allocated for the Segment 1 prefix, which will be filled in the course of segment processing. The terminal starts serial transmission into the remaining 18 character spaces of the buffer. After receipt of the first character, a request is issued for another buffer. For Segment 2, only 22 characters are allocated for the prefix, which is used to link the segment to the previous segment and identify the message. Buffers for subsequent segments of the message are similarly obtained and employed. An "extra" buffer is requested by the last message segment.

For each new message, the entire allocation cycle is repeated. After a filled buffer has been processed, its content is placed on the disk file and the buffer is again made available. The extra buffer is released without being used.

For outgoing transmission, a similar procedure is used. In this case, however, the processing unit must be admitted to the line of the desired terminal. The processing unit then issues the request for the first buffer and starts the transmission-out cycle. When segments are retrieved from the disk file, they are examined to avoid a request for an extra buffer.

buffer request rate

The average buffer request rate λ is equal to $R(N + 0.5)L$, where N is the average number of segments per message and 0.5 accounts for the unused extra buffers requested by the last segments of incoming messages.

Let N_1 denote the number of characters per first segment and N_i the number per subsequent segment. Then the average number of segments per message is given by

$$N = 1 + \left\lceil \frac{\bar{M} - N_1}{N_i} \right\rceil$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. In our example, we have

$$N = 1 + \left\lceil \frac{250 - 18}{28} \right\rceil = 10$$

Since the last segment must consist of eight characters, the buffer request rate, expressed in requests per minute, is

$$\lambda = R(N + 0.5)L = 2.83(10.5)L = 29.72L$$

buffer holding times

A buffer holding time depends upon the buffer's association with either an incoming or an outgoing message and its usage as first, second, intermediate, last, or extra buffer for this message. The average holding time $\bar{s}$ of a buffer is the weighted sum of all the individual holding times. Within the context of a given message, let h_1, h_2, h_i, h_N, and h_x denote the holding times of the first, second, intermediate, last, and extra buffers, respectively. Also let I denote the number of buffers with holding time h_i. Assuming that inbound and outbound traffic are equal, $\bar{s}$ is given by

$$\bar{s} = \frac{2(h_1 + h_2 + Ih_i + h_N) + h_x}{7 + 2I} \tag{8}$$

Let p, q, and t denote processing time, direct-access move time, and transmission time, respectively. Then in light of the buffer allocation process, we can write

$$\left.\begin{aligned} h_1 &= p_1 + q_1 + t_1 \\ h_2 &= p_2 + q_2 + t_2 + [t_1 - (1/K) - w] \\ h_i &= p_i + q_i + t_i + [t_{i-1} - (1/K) - w] \\ h_N &= p_N + q_N + t_N + [t_{N-1} - (1/K) - w] \\ h_x &= \qquad\qquad [t_N \quad - (1/K) - w] \end{aligned}\right\} \tag{9}$$

where $i = 3, 4, \cdots, N - 1$, and the terms in the square brackets represent r, the idle buffer time. The w's in (9) are unknown random variables, not necessarily the same for different h's.

It is to be expected that $t_2 = t_i$. Moreover, the intermediate transmission times t_i (where $2 \leq i < N$) are all the same. In many cases it is tenable to assume that $p_1 = p_2 = p_i = p_N$ and

Table 2 Solution 2 buffer requirements

Number of lines in system	*Buffer request rate*	*Number of buffers for three values of break probability*		
		0.05	0.01	0.001
10	4.85	17	19	22
15	7.275	23	26	30
20	9.7	31	34	38
25	12.125	38	41	46
30	14.55	45	48	54
40	19.40	59	63	69
50	24.25	73	78	85
75	36.375	109	115	123
100	48.5	143	152	161
150	72.75	215	226	237

$q_1 = q_2 = q_i = q_N$; we make this assumption to simplify discussion. Substituting (9) into (8), we obtain

$$\bar{s} = a - g\bar{w} \tag{10}$$

where

$$a = \frac{(6 + 2I)(p + q) + 4t_1 + 4(1 + I)t_i + 3t_N - (5 + 2I)(1/K)}{7 + 2I}$$

and

$$g = (5 + 2I)/(7 + 2I)$$

For the example, let it be given that $I = 7$, $K = 14.8$ characters per second, and (in units of seconds) $p = 0.020$; $q = 0.0134$; $t_1 = 1.216$; $t_i = 1.892$; and $t_N = 0.5405$. Substitution yields $a = 3.153$ and $g = 0.9048$.

The solutions for P are rather tedious because of their iterative nature, but they can easily be programmed for a computer. In our example, where the buffer holding time is more nearly a constant variable than an exponential variable (most of the holding time is in fact accounted for by t), the adjusted model of Solution 2 is recommended. Using the specific values of our example, Table 2 shows the buffer requirements for differing number of lines and break probabilities.

Effects of approximations

To assess the effect of the assumptions and approximations made in the analytical model, the results for a variety of cases were checked against the results obtained using GPSS II, a general-purpose systems simulator.[4] The simulation was also extended to include some cases of the more frequently used scheme that

grants first priority to buffer requests for outgoing transmission (a priority scheme not recognized in the analytical models). Although the analytic models required more buffers, the overdesign is judged well within the limits of ordinary design practice.

A summary of the simulation results is shown in Table 3. These results show that both models overestimate the probability of break. Results are based on the following inputs:

Average message length	390 characters
Line speed	14.8 characters per second
Line utilization	75 percent
Buffer size	55 characters
First-buffer prefix size	36 characters
Other-buffer prefix size	25 characters
Processing time (1st segment in)	20 milliseconds
Processing time (1st segment out)	10 milliseconds
Processing time (other segments)	negligible
Time of buffer request	after assembly of first character

The assumed line-control procedure was based on contention (not polling), and the assumed auxiliary storage device was an IBM 2311 disk file with a special sequential ordering of file accesses that minimizes seek time.

Although the analytical models lead to conservative estimates, the appropriate way to judge their utility is to observe the effects of this bias on actual design choices. In order to show the practical effects of the bias, the results of Table 3 are shown again in the

Table 3 Summary of simulation results

	Number of lines N							
	16			32			50	
	No. of buffers			*No. of buffers*			*No. of buffers*	
	26	25	23	48	46	44	72	70
Buffer utilization	.86	.91	.944	.93	.9656	.973	.981	.987
Probability of character loss SIMULATION	.002	.006	.038	.002	.013	.031	.02	.04
Probability of transmission gap SIMULATION	.000	.000	.000	.000	.000	.000	.000	.000
Probability of character loss or transmission gap ANALYTIC MODEL	.01	.047	.14	.025	.095	.12	.08	.11

Table 4 *Comparative estimates of P*

Number of lines	16	32	50
	$\eta = 0.01$		
Simulation estimate	25	47	not computed
Analytic estimate	25	49	not computed
	$\eta = 0.05$		
Simulation estimate	23	44	70
Analytic estimate	25	47	74

form of Table 4. In the design sense, the results are in useful agreement with simulation results—within about six percent.

The question may be raised whether the analytical approach is over-conservative if applied to a nonpriority case imposing less stringent conditions on the waiting time. Table 3 shows that the domain of pool sizes is close to the saturation state (100 percent utilization). Thus, the process is very sensitive to any variation of the number of buffers in the pool. On the other hand, a change in the waiting-time requirements, such as might be occasioned by introduction of a priority scheme, has relatively little effect on the overall number of buffers required. For example, in a simulation involving $\eta = 0.01$ and 32 lines, 44 buffers were found necessary in the nonpriority case and 47 buffers (an increase of six percent) in the two-priority case.

Buffer length

If buffers are B characters in length, BP characters of storage are required by the buffer pool. Unfortunately, B and P are interdependent variables, and there is no known way of expressing them in a form that would readily permit solving for minimum BP. Moreover, there may be operational considerations that exclude consideration of the smaller buffer size. For example, a segment must be long enough to make segment-assembly time longer than the time required to allocate a buffer to a pending request. In many cases, it is also required that each segment contain the message header. For low-speed lines, the first restriction has little effect, but for high-speed lines it definitely has to be taken into account. The message-header requirement, often made to facilitate processing, is a standard requirement in QTAM.[5]

Another factor to consider is the number of required accesses to auxiliary storage. In general, the shorter the segment, the greater the number of accesses to auxiliary storage and the longer the buffers will be tied up in the auxiliary-storage access queue.

As a result of all the restrictions and interdependent relationships, buffer length is usually selected by taking the smallest length that comfortably meets all the restrictions that apply. If a better solution is desired, an integrated simulation study of the entire system is indicated. In such cases, the approach discussed here will still be useful in narrowing the ranges of the variables to be investigated.

Summary

This paper discusses an approximate procedure for estimating the amount of storage required for a buffer pool. The solution method recommended for estimating the required number of buffers was checked by simulation (for the case of a Poisson distribution of input messages) and found to agree within six percent of the estimates yielded by the simulation.

CITED REFERENCES AND FOOTNOTES

1. I. Delgalvis and G. Davison, "Storage requirements for a data exchange," *IBM Systems Journal* **3**, No. 1, 2–13 (1964).
2. P. M. Morse, *Queues, Inventory and Maintenance*, John Wiley and Sons, New York (1958).
3. The solution is based on the method of Everett, *Journal of Operations Research Society of America* **1**, 279–283 (1953), which combines the assumption of constant service time and the use of exponential-case formulas. This is an approximate solution, as there is no identity between the waiting-time distribution of the constant-service case and its "equivalent" exponential process. The iterative identity obtained by Everett is between the probabilities of finding a given number of customers in the system under two processes.
4. R. Efron and G. Gordon, "A general purpose digital simulator and examples of its application, Part I, description of the simulator," *IBM Systems Journal* **3**, No. 1, 23–34, (1964).
5. See Part I of this paper, Footnote 2, for references to manuals on QTAM.