MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Computation Structures Group Memo No. 21

A STATISTICAL MODEL FOR CONSOLE BEHAVIOR IN MULTIUSER COMPUTERS

by

Peter J. Denning

A STATISTICAL MODEL FOR CONSOLE BEHAVIOR IN MULTIUSER COMPUTERS

by

Peter J. Denning

Massachusetts Institute of Technology

Cambridge, Massachusetts

Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102 (01).

ABSTRACT

The ability of a computing system to communicate with the outside world efficiently is as important as its ability to perform computations efficiently. It is quite difficult to characterize a particular user, but rather easy to characterize the user community. Based on the properties of this community we have postulated a hypothetical "virtual console". No claim is made that a virtual console behaves like any actual console; but the entire collection of virtual consoles behaves just like the collection of actual consoles. Using the model we answer questions like "How many processes are suspended waiting for console input?", "What is the maximum rate at which a process can execute?", and "What bounds can be set on overall buffer requirements?" Answers to these questions are needed in certain aspects of operating system design. For example, it need not be a valid assumption that a sizable fraction of processes will be suspended waiting for console input. Or, by providing estimates of waiting times, the model can assist in deciding how long the program belonging to a suspended process ought to remain in core memory before being displaced, a problem in dynamic core allocation. Finally the model shows that pooled buffers are superior to private buffers, one for each console.

INTRODUCTION

The ability of a computing system to communicate with the outside world efficiently is as important as its ability to perform computations efficiently. When many processes are operating concurrently, a shared, Protected-Service-Routine (PSR) must be employed to coordinate communication between each user and his processes[1]. A PSR will be called asynchronously by many user processes. Although they can exert no direct influence on one another, these processes can affect one another via the PSR. Conceivably the PSR could be servicing users and processes beyond its capacity, so that many processes will be blocked unnecessarily waiting to pick up messages. Or, the PSR may have insufficient buffers and messages may be lost.

A model for console* behavior would be of considerable importance in the design of multiprogrammed computing systems. In this paper we set forth such a model and indicate its application to these factors relevant to operating system design:

1. The behavior of consoles is characterized.
2. One of the main difficulties in analyzing systems where a finite population is requesting service is this: as more and more units enter service, there are fewer and fewer units left in the source population to generate new requests, so the arrival rate slackens. One of our results here is, in large-scale computing systems, the source population may be regarded as infinite. If the number of users is moderately large (50 or more) very little error is

*By "console" we mean a typewriter console employed by a user to communicate with the computing system.

introduced by assuming that their number is infinite. As we will see, the expected number of processes awaiting service from the PSR normally will be small compared to the number of users. So, the arrival rate will not slacken significantly.

3. Because the expected number of processes blocked in the PSR may be a small fraction of the total, it would be bad design practice to assume that many of the active processes will be blocked waiting for console input, without first checking console behavior for the validity of this assumption.
4. The model can be used to estimate the time a particular process waits in the PSR for its message to arrive. This in turn is useful for determining the maximum rate at which a process can execute. It can assist in determining just how many processes can be active at a time without overtaxing the processor capacity of the system, an important allocation problem.
5. In a multiprogrammed computing system it is relatively expensive to move programs in and out of core memory. Yet it is also expensive to leave unused programs in core memory. It is important to know how long a process will be blocked in the PSR waiting for its message, so that core managment routines will not foolishly move a temporarily blocked program from core memory. The model can be used to predict the time a process will be blocked in the PSR. During this interval no operating system action should be taken to move that program from memory. The model, therefore, has obvious application to problems of dynamic core allocation.

6. Finally, the model can be used to estimate buffer requirements. In particular, we will show that the use of a pooled buffer is far more efficient that the use of a private buffer for each console.

DESCRIPTION OF THE MODEL

We consider only the input-string-handling function of a PSR, and not its output-string-handling function, although both functions would normally be housed in the same routine.

It is difficult to model a single user and a single console, but relatively straightforward to model the entire collection of consoles. Users tend to work in bursts, active periods following the inactive, so the character arrival rate from a given console varies erratically at best. On the other hand, with many consoles, the total character rate from all consoles should remain steady, even though contributions to the total rate from individual consoles may vary unpredictably. Consequently we will define a hypothetical entity -- the virtual console -- which will allow us to predict the behavior of the entire collection of consoles. The virtual console cannot be used to predict the behavior of any single console; its only utility is to permit us to obtain the characteristics of the consoles as a group. For it is difficult to base design assumptions on the hoped-for characteristics of some single user: there is no "typical" user.

As indicated by Figure 1, the PSR is receiving characters from each of N independent consoles, the i^{th} console transmitting at rate c_i. Incoming characters are sorted and placed into the appropriate one of N buffers. The message in the i^{th} buffer is retrieved by the i^{th} process at a rate p_i. A process making a pickup must wait for the arrival of an End of Line (EOL) character (such as carriage return) before it can claim the message in its buffer. The parameters in Figure 1 have these meanings:

c_i -- character arrival rate from the i^{th} console.

p_i -- pickup rate of the i^{th} process.

y_i -- the number of characters in the i^{th} buffer.

The total character and pickup rates are, respectively,

$$C = \sum_{i=1}^{N} c_i \qquad\qquad P = \sum_{i=1}^{N} p_i$$

Corresponding to this we define a collection of N <u>virtual</u> <u>consoles</u> and <u>processes</u> having these properties:

1. Each virtual console inputs characters at rate $c = C/N$. That is, character interarrival times are exponential with mean $1/c$.
2. Each virtual process makes pickups at rate $p = P/N$. That is, the time intervals from moment of pickup to moment of next request for pickup are exponential with mean $1/p$.
3. The virtual consoles and processes are statistically independent.

4. The total character rate $C = Nc$ does not exceed the capacity of the Sorter, which performs its function without delay.
5. Even if its buffer is non-empty, a virtual process making a pickup must wait in the PSR (it is blocked) until the arrival of the End of Line (EOL) character.

From now on, the terms "console" and "process" each refer to a virtual console or to a virtual process. All results will pertain to the collection of virtual consoles and processes and so are valid for the collection of actual consoles.

We wish to answer these questions:

1. What is the probability a process must wait when it attempts a pickup? How long must it wait? What is the maximum rate at which it can execute?
2. At any given time, how many processes will be found blocked in the PSR?
3. What bounds can be set on buffer requirements? How do separate buffers compare with a pooled buffer?

Since the answers to these questions depend on the number of characters to arrive before the EOL character (the length of the line), we will epxress answers in terms of the generating function for the EOL character position.

To simplify reading the paper, all principal symbols are listed in Table I.

TABLE I. Symbols

N	number of users, consoles, processes
c_i	character rate from i^{th} actual console
p_i	pickup rate of i^{th} actual process.
y_i	number of characters in i^{th} actual buffer
$C = \sum_{i=1}^{N} c_i$	total character rate
$P = \sum_{i=1}^{N} p_i$	total pickup rate
$c = C/N$	character rate from virtual console
$p = P/N$	pickup rate of virtual process.
y	number of characters in virtual buffer
c-event	arrival of a character
p-event	arrival of a pickup
$\rho = \frac{C}{P} = \frac{c}{p}$	ratio of characters to pickups
j	position of End of Line (EOL) character from start of line
$m = E[j]$	expected EOL character position (expected line length)
$\sigma^2 = \sigma^2[j]$	variance of EOL character position (variance of line length)
$\{q_j\}$ $j=1,2,3,...$	probabilities that EOL character is j^{th} character
$Q(x) = \sum_{j=1}^{\infty} q_j x^j$	generating function for $\{q_j\}$. Note $Q(1) = 1$.

TABLE I (cont).

$Q'(x) = \sum_{j=1}^{\infty} j\, q_j\, x^{j-1}$	Derivative of Q(x). Note that m = Q'(1).
Q^*	Approximation for quantity $Q\left(\frac{\rho}{1+\rho}\right)$.
n	number of characters to enter buffer between two pickups
$p_n(k) = Pr[n=k]$	probability distribution for n
r; $E[r]$	number of blocked processes; its expectation
w; $E[w]$	duration of blocked interval (Figure 2); its expectation.
μ	maximum processing rate
μ^*	approximation for μ
$Pr[W]$	probability a process must wait when it makes a pickup
$Pr[B]$	probability of finding a process blocked at any arbitrary time.

THE ANALYSIS

We begin by determining the number of characters to arrive from a virtual console during a computing interval of its process. Later we will extend the reasoning to include all N consoles. In Figure 2 we have indicated there is an interval τ during which the process computes. The process then attempts to claim a message in its buffer. If no EOL characters has arrived yet, the process is blocked for a time w until the character arrives. Otherwise one or more EOL characters (and so one or more lines) have accumulated in the buffer; in this case the process takes away all complete lines, leaving only an unfinished line behind for its next pickup. We do not consider the case where a process may proceed at once if it finds its buffer empty, because we assume it would not have attempted a pickup unless it needed a message. The length τ of the computing interval is an exponential random variable,

$$(1) \qquad p_T(t) = p\, e^{-pt} \qquad t \geq 0$$

and the expected length of the interval is $E[T] = 1/p$.

Number of Characters Between Pickups

Defined by equation (1), p is the pickup rate. The character rate c is exponential with mean 1/c, just like equation (1) with p=c. Let n be the number of characters to enter the buffer between two pickups. Then

$$(2) \qquad p_n(k) = Pr[n=k] = \left(\frac{1}{1+\rho}\right)\left(\frac{\rho}{1+\rho}\right)^k \qquad k=0,1,2,\ldots$$

and k=0 denotes an attempt to claim an empty buffer.

PROOF: Denote the arrival of a character by "c-event" and the arrival of a pickup by "p-event". If k c-events fall between two p-events, the interval (0,T) in Figure 3 is divided into (k+1) subintervals. Then

$$\Pr[\text{k c-events between 2 p-events}]$$
$$= \Pr\left\{ \begin{array}{l} 2^{nd} \text{ event is c-event } \underline{\text{and}} \\ 3^{rd} \text{ event is c-event } \underline{\text{and}} \\ \qquad \vdots \\ (k+1)^{st} \text{ event is c-event } \underline{\text{and}} \\ (k+2)^{nd} \text{ event is p-event} \end{array} \;\middle|\; \begin{array}{l} \text{given that} \\ 1^{st} \text{ event is} \\ \text{p-event} \end{array} \right\}$$

Since c-events and p-events are independent, this becomes

$$(3) \qquad \Pr[\text{k c-events between 2 p-events}] = \left(\Pr[\text{c-event}] \right)^k \left(\Pr[\text{p-event}] \right)$$

During the interval (0,T) a total of $(c+p)T$ events are expected. The expected number of c-events is cT, and the expected number of p-events is pT. The relative frequencies of c-events and p-events are

$$(4) \qquad \begin{array}{l} f_c = \dfrac{cT}{(c+p)T} = \dfrac{c}{c+p} = \dfrac{\rho}{1+\rho} \\[2ex] f_p = \dfrac{pT}{(c+p)T} = \dfrac{p}{c+p} = \dfrac{1}{1+\rho} \end{array} \qquad \rho = \frac{c}{p}$$

Hence we assign $\Pr[\text{c-event}] = f_c$ and $\Pr[\text{p-event}] = f_p$. Putting these into equation (3) yields equation (1).

QED

The EOL Generating Function

Throughout the remaining discussions we will need to express results in terms of the position of the EOL character. Let j be the position of the EOL character relative to the start of the line. Let q_j be the probability that the line is j characters long; that is, the probability the EOL character is the j^{th} character in the line. The probability distribution for the arrival of the EOL character is given by the sequence $\{q_1, q_2, q_3, \ldots\}$, or more compactly by the generating function

$$Q(x) = \sum_{j=1}^{\infty} q_j x^j \tag{5}$$

The expectation and variance of the EOL position are

$$\begin{aligned} m &= E[j] = \sum_{j=1}^{\infty} j\, q_j = Q'(1) \\ \sigma^2 &= E[j^2] - m^2 = \sum_{j=1}^{\infty} j^2 q_j - m^2 = Q''(1) + m - m^2 \end{aligned} \tag{6}$$

In real-life situations, Q(x) will be unknown; only σ^2 and m will be available. An approximation in terms of σ^2 and m would be valuable. Using a Taylor Series expansion,

$$Q(x) \approx Q(1) + Q'(1)(x-1) + \tfrac{1}{2}Q''(1)(x-1)^2$$

or,

$$Q(1-x) \approx Q(1) - Q'(1)x + \frac{1}{2} Q''(1)\, x^2$$

Using Q(1) = 1 and equations (6) we have

$$Q(1-x) \approx 1 - mx + \frac{\sigma^2 + m(m-1)}{2} x^2$$

We shall see the quantity

$$Q\left(\frac{\rho}{1+\rho}\right)$$

many times, so we define Q^* to be

$$Q\left(\frac{\rho}{1+\rho}\right) = Q\left(1 - \frac{1}{1+\rho}\right) \approx 1 - \frac{m}{1+\rho} + \frac{\sigma^2 + m(m-1)}{2(1+\rho)^2} = Q^* \tag{7}$$

Q^* approximates $Q\left(\frac{\rho}{1+\rho}\right)$ closely only if $\rho \gg 1$, a case not far from reality. Examination of Q(1-x) show that the expression for Q^* is too large; that is,

$$Q\left(\frac{\rho}{1+\rho}\right) < Q^*$$

Waiting Times

Let w = random variable of waiting time;

j = position of EOL character $\qquad j=1,2,3,\ldots$

n = number of characters to arrive between two pickups.

We want to determine $p_w(t)$, the probability density function for w.

When a pickup arrives, there are already n characters in the buffer, and so the wait is for the remaining $(j-n)$ characters until the EOL character. Let

(8) $$p_{w|j,n}(t) = \Pr[\text{waiting } t \text{ seconds} \mid (j-n) \text{ characters to go}]$$
$$\Pr\left[\begin{array}{c}\text{waiting for the } (j-n)^{\text{th}} \text{ event in a Poisson}^* \\ \text{Process with rate } c\end{array}\right]$$

---

*In a Poisson Process with rate c, $\Pr[\text{exactly } k \text{ events in time } t] = \frac{(ct)^k}{k!} e^{-ct}$

To determine the wait w_k for the k^{th} Poisson event, note that

(9) $$\Pr[t \le w_k < t+dt] = \Pr[(k-1) \text{ events in } (0,t) \text{ and } 1 \text{ event in } (t,t+dt)]$$
$$= \frac{(ct)^{k-1}}{(k-1)!} e^{-ct} c\, dt$$

Then

(10) $$p_{w_k}(t) = c \frac{(ct)^{k-1}}{(k-1)!} e^{-ct} \qquad t \ge 0 \qquad k > 0$$

Putting equation (8) into (10)

(11) $$p_{w|j,n}(t) = c \frac{(ct)^{j-n-1}}{(j-n-1)!} e^{-ct} \qquad t \ge 0 \qquad n < j$$

Next note that if (j-1) or fewer characters have arrived, the wait $w > 0$; and if j or more characters have arrived, $w = 0$:

$$w > 0 \qquad \text{if } n < j$$
$$w = 0 \qquad \text{if } n \geq j$$

Then

$$(12) \qquad p_w(t) = \begin{cases} \sum_{j=1}^{\infty} \left(\sum_{k=0}^{j-1} p_{w|j,n}(t)\, p_n(k) \right) q_j & t > 0 \\ \sum_{j=1}^{\infty} \left(\sum_{k=j}^{\infty} p_n(k) \right) q_j & t = 0 \end{cases}$$

so that

$$(13) \qquad p_w(t) = \begin{cases} \sum_{j=1}^{\infty} \left(\sum_{k=0}^{j-1} \frac{(ct)^{j-k-1}}{(j-k-1)!}\, c\, e^{-ct} \frac{1}{1+\rho} \left(\frac{\rho}{1+\rho}\right)^k \right) q_j & t > 0 \\ \sum_{j=1}^{\infty} \left(\sum_{k=j}^{\infty} \frac{1}{1+\rho} \left(\frac{\rho}{1+\rho}\right)^k \right) q_j & t = 0 \end{cases}$$

Without knowledge of the EOL character position probabilities $\{q_j\}$, no closed-form expression for $p_w(t)$ can be found. However the expected wait can be put into closed form, which we proceed to do.

To obtain the expected wait E[w], let j be the position of the EOL character, and suppose n characters have already arrived. Then

$$E[w \mid j,n] = 0 \qquad \text{if } n \geq j$$
$$E[w \mid j,n] = \frac{j-n}{c} \qquad \text{if } n < j$$

and so

$$E[w] = \sum_{j=1}^{\infty} \left(\sum_{k=0}^{j-1} E[w \mid j,n]\ p_n(k) \right) q_j$$

(14) $$E[w] = \sum_{j=1}^{\infty} \sum_{k=0}^{j-1} \left(\frac{j-n}{c}\right) \left(\frac{1}{1+\rho}\right) \left(\frac{\rho}{1+\rho}\right)^k q_j$$

After some manipulation not worth reproducing, this can be put into the form

(15) $$E[w] = \frac{1}{c}\left[m - \rho\left(1 - Q\left(\frac{\rho}{1+\rho}\right)\right)\right]$$

The approximation $Q^* \approx Q\left(\frac{\rho}{1+p}\right)$ from equation (7) can be used when $Q(x)$ is unknown, and $\rho >> 1$.

Maximum Processing Rate

The maximum processing rate μ is defined to be the fraction of the time the process is not waiting in the PSR. When a pickup occurs the process expects to wait E[w] before the EOL character. It then undergoes a computing interval of expected duration 1/p. Hence the maximum processing rate is

(16) $$\mu = \frac{\frac{1}{p}}{E[w] + \frac{1}{p}}$$

Using equation (15), this becomes

$$(17) \qquad \mu = \frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)}$$

Using Q^* from equation (7), and noting that $Q\left(\frac{\rho}{1+\rho}\right) < Q^*$, we have μ^*, an approximation for μ:

$$(18) \qquad \mu = \frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)} > \frac{\rho}{m + \rho\, Q^*} = \mu^*$$

The approximation is close when $\rho \gg 1$.

Waiting Probabilities

There are two similar questions about waiting, with quite different answers:

1. What is the probability that a process, upon arriving to claim a buffer-load, finds that the EOL character has not arrived and so must wait?
2. If as observers, we look into the PSR at a random time, with what probability do we find a given process blocked?

To answer the first question, note that the probability that a process must wait is just the probability that the EOL character has not arrived since the last pickup. Let j be the position of the EOL character, and Q(x) be its generating function (equation (5)). Let n be the number of characters to arrive between two pickups, and $p_n(k)$ be its probability distribution (equation (2)). If $n < j$, the process must wait, but not if $n \geq j$. Hence

$$\Pr[W \mid j] = \Pr[\text{waiting} \mid \text{EOL character is } j^{th} \text{ character}]$$

$$= \Pr[n < j]$$

$$= \sum_{k=0}^{j-1} p_n(k)$$

(19) $$\Pr[W \mid j] = \sum_{k=0}^{j-1} \left(\frac{1}{1+\rho}\right) \left(\frac{\rho}{1+\rho}\right)^k$$

Finally the probability the process must wait is

(20) $$\Pr[W] = \sum_{j=1}^{\infty} \Pr[W \mid j]\, q_j$$

where $\{q_j\}$ have been defined in equation (5). Hence

$$\Pr[W] = \sum_{j=1}^{\infty} \sum_{k=0}^{j-1} \left(\frac{1}{1+\rho}\right) \left(\frac{\rho}{1+\rho}\right)^k q_j$$

$$\sum_{j=1}^{\infty} \left(\frac{1}{1+\rho}\right) q_j \sum_{k=0}^{j-1} \left(\frac{\rho}{1+\rho}\right)^k$$

$$= \sum_{j=1}^{\infty} \left(\frac{1}{1+\rho}\right) q_j \frac{1 - \left(\frac{\rho}{1+\rho}\right)^j}{1 - \left(\frac{\rho}{1+\rho}\right)} \qquad \text{since } \sum_{r=0}^{n} x^r = \frac{1 - x^{n+1}}{1 - x}$$

$$= \sum_{j=1}^{\infty} q_j \left(1 - \left(\frac{\rho}{1+\rho}\right)^j\right) \qquad \text{since } \frac{1}{1+\rho} = 1 - \frac{\rho}{1+\rho}$$

$$= 1 - \sum_{j=1}^{\infty} \left(\frac{\rho}{1+\rho}\right)^j q_j \qquad \text{since } \sum_{j=1}^{\infty} q_j = 1$$

(21) $$\Pr[W] = 1 - Q\left(\frac{\rho}{1+\rho}\right)$$

Using the approximation Q^* from equation (7) we see that

(22) $$\Pr[W] \approx \frac{m}{1+\rho} - \frac{\sigma^2 + m(m-1)}{2(1+\rho)^2}$$

if $\rho >> 1$.

To answer the second question, note that the probability of finding the process blocked is just the fraction of time it is in fact blocked (Figure 2):

(23) $$\Pr[B] = \frac{E[w]}{E[w] + \frac{1}{p}}$$

Comparing this with equations (16) and (17),

(24) $$\Pr[B] = 1 - \mu = 1 - \frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)}$$

Of course the approximation Q^* can be used. Comparing with equation (18) we have

$$1 - \mu < 1 - \mu^*$$

or,

(25) $$\Pr[B] < 1 - \frac{\rho}{m + \rho\, Q^*}$$

Number of Blocked Processes

When we ask the question "how many processes are blocked?" we really mean "if we look in at some random time, how many processes do we expect to find blocked?" There are N processes, each blocked independently with $\Pr[B] = 1-\mu$, as given by equation (24). So the probability that r of the N are blocked is Binomial:

(26) $$\Pr[r,N] = \Pr[r \text{ of } N \text{ are blocked}] = \binom{N}{r}(1-\mu)^r \mu^{N-r} \qquad r=0,1,\ldots,N$$

where $$\binom{N}{r} = \frac{N!}{r!(N-r)!}$$

Therefore the expected number waiting is

(27) $$E[r] = N(1-\mu) \approx N(1-\mu^*)$$

$E[r]$ is an important design parameter, and approximations for it using Q^* can be obtained simply, since μ^* depends only on m and σ^2. See equation (25).

We desire an approximation for $Pr[r,N]$ of equation (24), since in its present form it is unwieldy. It is well-known that for large N, when the quantity $E[r]$ is of moderate magnitude, the Poisson Distribution approximates the Binomial [see Feller[2], p. 143]. For $E[r]$ to be only of moderate magnitude, we require that

$$1 - \mu = 1 - \frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)}$$

be small. This is the same as requiring $\rho \gg m$; if so, $\rho \gg 1$ and $Q\left(\frac{\rho}{1+\rho}\right) \approx 1$ and then $\mu \approx \rho/(m+\rho) \approx 1$, and $1-\mu$ is small. Under these circumstances

(28) $$Pr[r,N] \approx \frac{\lambda^r}{r!} e^{-\lambda} \qquad \lambda = E[r] = N(1-\mu)$$

If, in the computing system under consideration, $\rho \gg m$, it will be true that $E[r]$ is small compared to N, so that the number of blocked processes is not large. This fact leads to two important conclusions:

1. Since $E[r]$ is small, the actual number of processes that are not blocked, $N - E[r]$, is close to N. That is, the source population N has not been seriously decreased, and is behaving like an infinite population.
2. Since $E[r]$ is small, the number of processes blocked by the PSR is small, so it need not be a good operating system design assumption that significantly many processes will be held up waiting for

console messages. This is important in designing allocation policies.

Bounds on Buffer Requirements

Let y_i be the number of characters in the ith buffer, so that the total buffer requirement is, at any time,

$$\sum_{i=1}^{N} y_i$$

Suppose we want the probability that the buffer overflows to be less than some small number, ϵ. We want to find a buffer size $N\alpha$ such that

$$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] \leq \epsilon \qquad \alpha > 0$$

To simplify the analysis we make the assumption that pickups occur sufficiently frequently that the probability of being blocked is small, so that an EOL character may nearly always be found among the characters in the buffer. In this case the number of characters in the buffer is just the number to arrive between two pickups; that is, the probability that n characters are in the buffer is given by $p_n(k)$. Using the result of the Appendix, the bound is

(29) $$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] \leq \left[\frac{1+\alpha}{1+\rho}\left(\frac{\rho}{\alpha}\,\frac{1+\alpha}{1+\rho}\right)^{\alpha}\right]^{N} \qquad \alpha > 0$$

There are two ways to assign buffer:

1. Each process acquires buffer dynamically as needed, so there is a pooled buffer.
2. Each process has a private buffer.

Equation (26) gives the total buffer requirement, and so is the probability of overflowing a pooler buffer of capacity N α . If each buffer is private, we must consider the overflow for each buffer alone. For a given ϵ, use equation (26) with N = 1 and find β such that

$$\Pr[y \geq \beta] \leq \frac{1+\beta}{1+\rho}\left(\frac{\rho}{\beta}\,\frac{1+\beta}{1+\rho}\right)^{\beta} = \epsilon$$

Then the total buffer requirement is Nβ.

The following table illustrates the effects.

$\rho = 10$ $N = 50$		
	probability of overflow	
Total buffer size (characters)	pooled	private
750	.006	.90
1000	10^{-6}	.76
1500	-	.44
2000	-	.22
2500	-	.10
3000	-	.05
3500	-	.02
4000	-	.01
4500	-	.004

With $\rho = 10$, a 750-character pooled buffer overflows with the same probability as 4500 characters of private buffer. A 1000-character pooled buffer overflows with probability less than 10^{-6}.

A Special Case -- No EOL Character

In some applications, an EOL character will be be present. That is to say, one or more characters in a buffer constitute a message. An arriving process may claim the buffer load if there is at least one character present, and must wait only if the buffer is empty. To handle this case, note that we may treat each character as an EOL character; equivalently, in the sequence $[q_1, q_2, q_3, \ldots]$ we have $q_1 = 1$ and $q_j = 0,\ j \geq 2$. In this case the generating function is

(30) $$Q(x) = x$$

The approximation Q^* is now exact:

$$Q(x) = Q(1) + Q'(1)(x-1)$$

since $m = 1$ and all higher moments are zero. Then

(31) $$Q\left(\frac{\rho}{1+\rho}\right) = \frac{\rho}{1+\rho}$$

and we have the following table listing the main equations. Note that equation (13) giving $p_w(t)$ has been put into closed form.

Quantity	Expression	Equation	Result, using $Q(x)=x$.
$Q\left(\frac{\rho}{1+\rho}\right)$	$\sum_{j=1}^{\infty} q_j \left(\frac{\rho}{1+\rho}\right)^j$	(5)	$\frac{\rho}{1+\rho}$
$E[w]$	$\frac{1}{c}\left[m - \rho\left(1 - Q\left(\frac{\rho}{1+\rho}\right)\right)\right]$	(7)	$\frac{1}{c}\,\frac{1}{1+\rho}$
$p_w(t)$	--	(13)	$\begin{cases} \frac{c\,e^{-ct}}{1+\rho} & t > 0 \\ \frac{\rho}{1+\rho} & t = 0 \end{cases}$
μ	$\frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)}$	(17)	$\frac{\rho(1+\rho)}{1 + \rho(1+\rho)}$

Pr[W]	$1 - Q\left(\frac{\rho}{1+\rho}\right)$	(21)	$\frac{1}{1+\rho}$
Pr[B]	$1 - \mu$	(24)	$\frac{1}{1+\rho(1+\rho)}$
E[r]	$N(1-\mu)$	(27)	$\frac{N}{1+\rho(1+\rho)}$

To show that these results make sense, let us check with intuition. The probability a process must wait, Pr[W], is just the probability its buffer is empty, which in turn is just the probability the last event was a p-event. The expected wait is 1/c, provided it does wait; but it waits with probability Pr[W], so $E[w] = \frac{1}{c}\,\Pr[W] = \frac{1}{c}\,\frac{1}{1+\rho}$.

If $\rho^2 \gg 1$, E[r] is small; in such cases, designs expecting E[r] to be large would be in error.

<u>A Special Case -- Geometric Line Lengths</u>

A not unreasonable approximation to the EOL character distribution $\{q_j\}$ is the Geometric Distribution. Let (1-q) be the probability that a given character is the EOL character, independently of any previous characters in the line. Then

$$q_j = \Pr[\text{EOL character is } j^{th} \text{ character}]$$

$$= \Pr[1^{st}\ (j-1) \text{ characters are non-EOL, } j^{th} \text{ is EOL}]$$

(32) $$q_j = q^{j-1}(1-q) \qquad j=1,2,3,\ldots$$

Hence the generating function is

(33) $$Q(x) = \sum_{j=1}^{\infty} q_j x^j = \sum_{j=1}^{\infty} (1-q)\, q^{j-1} x^j = \frac{x(1-q)}{1-qx}$$

The expected EOL position is

$$(34)\qquad m = Q'(1) = \frac{1}{1-q}$$

The variance of the EOL position is

$$(35)\qquad \sigma^2 = Q''(1) + m - m^2 = \frac{q}{(1-q)^2}$$

We have the following table, where

$$\alpha = \rho(1-q) = \frac{\rho}{m}$$

Again note there is a closed form expression for $p_w(t)$.

Quantity	Expression	Equation	Result, Q(x) in eq.(3
$Q\left(\frac{\rho}{1+\rho}\right)$	$\sum_{j=1}^{\infty} q_j \left(\frac{\rho}{1+\rho}\right)^j$	(5)	$\frac{\alpha}{1+\alpha}$
Q^*	$1 - \frac{m}{1+\rho} + \frac{\sigma^2 + m(m-1)}{2(1+\rho)^2}$	(7)	$1 - \frac{m}{1+\rho}\left(1 - \frac{qm}{1+\rho}\right)$
$p_w(t)$	--	(13)	$\begin{cases} \frac{(1-q)c\ e^{-(1-q)ct}}{1+\alpha} & t > \\ \frac{\alpha}{1+\alpha} & t = \end{cases}$
$E[w]$	$\frac{1}{c}\left[m - \rho\left(1 - Q\left(\frac{\rho}{1+\rho}\right)\right)\right]$	(15)	$\frac{m}{c}\,\frac{1}{1+\alpha}$
μ	$\frac{\rho}{m + \rho\, Q\left(\frac{\rho}{1+\rho}\right)}$	(17)	$\frac{\alpha(1+\alpha)}{1 + \alpha(1+\alpha)}$
$Pr[W]$	$1 - Q\left(\frac{\rho}{1+\rho}\right)$	(21)	$\frac{1}{1+\alpha}$
$Pr[B]$	$1 - \mu$	(24)	$\frac{1}{1 + \alpha(1+\alpha)}$
$E[r]$	$N(1-\mu)$	(27)	$\frac{N}{1 + \alpha(1+\alpha)}$

Figure 4 shows Q^* and $Q\left(\frac{\rho}{1+\rho}\right)$ plotted for three values of m. It is clear that for $\rho >> 1$ the approximation is very good. But now we must also have $\alpha >> 1$ (that is, $\rho >> m$), which we can no longer claim to be a typical situation. In the case where $\rho \approx m$, the approximation Q^* is not too close; more complete knowledge of $Q(x)$ is required.

Figure 5 shows the processing rate μ for three values of m; we have included the case $m = 1$ for comparison. It is clear that for $\rho \approx m$, $\mu \approx \frac{2}{3}$, which means (Figure 6) that the number of blocked processes is no longer small. In fact $E[r] \approx N/3$ and so it would be permissible to allow $R/\mu = 3R/2$ processes to be active (on the average) to just occupy R processors.

CONCLUSIONS

A Model for console behavior can be of considerable importance in operating system design. Even though it is difficult to characterize a particular user, it is relatively simple to characterize the entire user population. Based on the characteristics of the user community we postulated a collection of "virtual consoles". We do not claim that a virtual console behaves like any actual console; but we do claim that the collection of virtual consoles behaves just like the collection of actual consoles. The main points we have tried to make are:

1. It may not be a particularly good design assumption that a significant portion of the processes will be blocked in the PSR. Operating systems based on this assumption may have difficulties.
2. The model can be used to estimate the maximum processing rate μ; then R/μ processes can, on the average, keep R processors busy. The model has application to resource allocation.
3. The model can be used to predict durations of blocked intervals, a factor useful in dynamic core managment.
4. The model can be used to predict buffer requirements. Pooled buffers are far superior to private buffers, especially when the number of users is large.
5. Approximations for the maximum processing rate μ can be obtained using only the knowledge of expectation and variance of line length.

APPENDIX

(I.) Let $\{y_i\}$ be a set of N identically distributed, statistically independent random variables. Then

(A.1) $$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] \leq \left(E[e^{\lambda_o y}]\, e^{-\lambda_o \alpha}\right)^N \qquad \lambda_o \geq 0$$

PROOF: Consider the step function $u(s-N\alpha)$ shown in Figure A.1. Since the step function is a binary random variable, its expectation is just the probability that it is non-zero; that is the probability that $s \geq N\alpha$, which is

$$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right]$$

Hence

(A.2) $$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] = E\left[u\left(\sum_{i=1}^{N} y_i - N\alpha\right)\right]$$

But, for any $\lambda > 0$,

(A.3) $$u(s - N\alpha) \leq e^{\lambda(s-N\alpha)}$$

Hence

(A.4) $$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] = E\left[u\left(\sum_{i=1}^{N} y_i - N\alpha\right)\right] \leq E\left[\exp\left(\sum_{i=1}^{N} y_i - N\alpha\right)\right]$$

But the $\{y_i\}$ are identically distributed and independent:

(A.5) $$\Pr\left[\sum_{i=1}^{N} y_i > N\alpha\right] \leq \left(E[e^{\lambda y}]\, e^{-\lambda\alpha}\right)^N$$

To obtain the tightest bound we must minimize with respect to λ. We want

(A.6) $$\frac{d}{d\lambda}\left(E[e^{\lambda y}]\ e^{-\lambda\alpha}\right) = E[(y-\alpha)\ e^{\lambda y}]\ e^{-\lambda\alpha} = 0$$

Solving equation (A.6) would yield λ_o, the minimum value for λ. λ_o is indeed the minimum, for the second derivative is positive:

(A.7) $$\frac{d^2}{d\lambda^2}\left(E[e^{\lambda y}]\ e^{-\lambda\alpha}\right) = E[(y-\alpha)^2\ e^{\lambda y}]\ e^{-\lambda\alpha} \geq 0$$

From equations (A.3), (A.5), and (A.6), we have these conditions for the bound to exist:

1. $\lambda_o \geq 0$
2. $E[e^{\lambda y}] < \infty$
3. $E[y\ e^{\lambda y}] < \infty$

And under these conditions we have (A.1). This bound, known as the Chernoff Bound, is discussed in detail by Wozencraft and Jacobs[3].

(II.) Let $\{y_i\}$ be the contents of the i^{th} buffer. The probability it contains k characters is

(A.8) $$p_y(k) = \frac{1}{1+\rho}\left(\frac{\rho}{1+\rho}\right)^k \qquad k=0,1,2,\ldots$$

We wish to show that

(A.9) $$\Pr\left[\sum_{i=1}^{N} y_i \geq N\alpha\right] \leq \left[\left(\frac{1+\alpha}{1+\rho}\right)\left(\frac{\rho}{\alpha}\,\frac{1+\alpha}{1+\rho}\right)^{\alpha}\right]^N$$

PROOF: Use equation (A.1):

$$E[e^{\lambda y}] = \sum_{k=0}^{\infty} e^{\lambda k} P_y(k) = \sum_{k=0}^{\infty} \left(\frac{\rho e^{\lambda}}{1+\rho}\right)^k \frac{1}{1+\rho}$$

(A.10) $$E[e^{\lambda y}] = \frac{1}{1 + \rho(1-e^{\lambda})}$$

To find λ_o, set

(A.11) $$\frac{d}{d\lambda} \frac{e^{-\lambda\alpha}}{1 + \rho(1-e^{\lambda})} = 0$$

Solving this results in

(A.12) $$\lambda_o = \ln\left(\frac{\alpha}{\rho} \frac{1+\rho}{1+\alpha} \right)$$

Putting λ_o in for λ in equation (A.10) yields $E[e^{\lambda y}]$. Then λ_o is used to obtain $e^{-\lambda\alpha}$. Both these quantities, when substituted into equation (A.1), yield equation (A.9), which is also equation (26).

QED

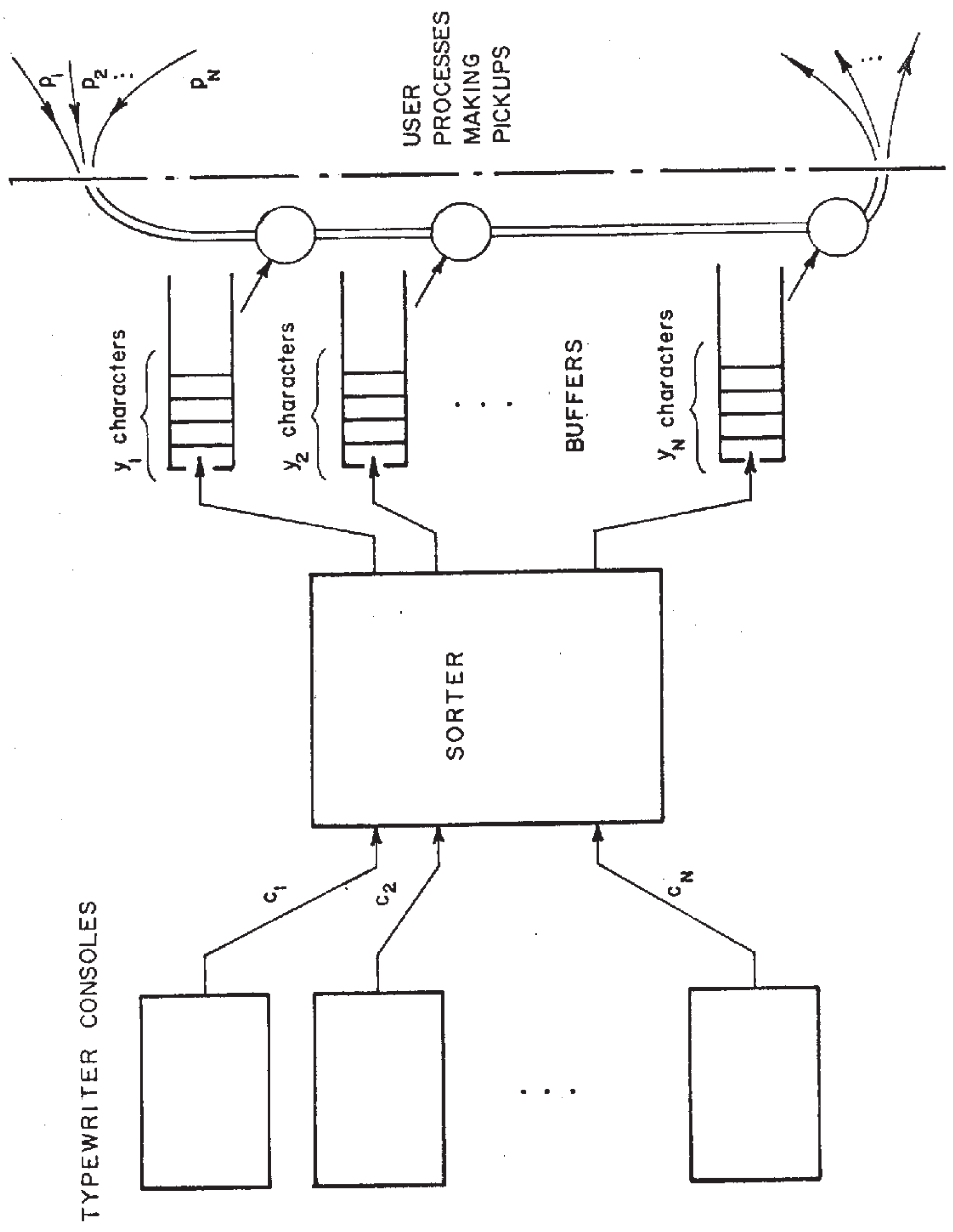
REFERENCES

[1]Dennis, J. B., and van Horn, E. C. "Programming Semantics for Multiprogrammed Computations." Comm ACM, March, 1966.

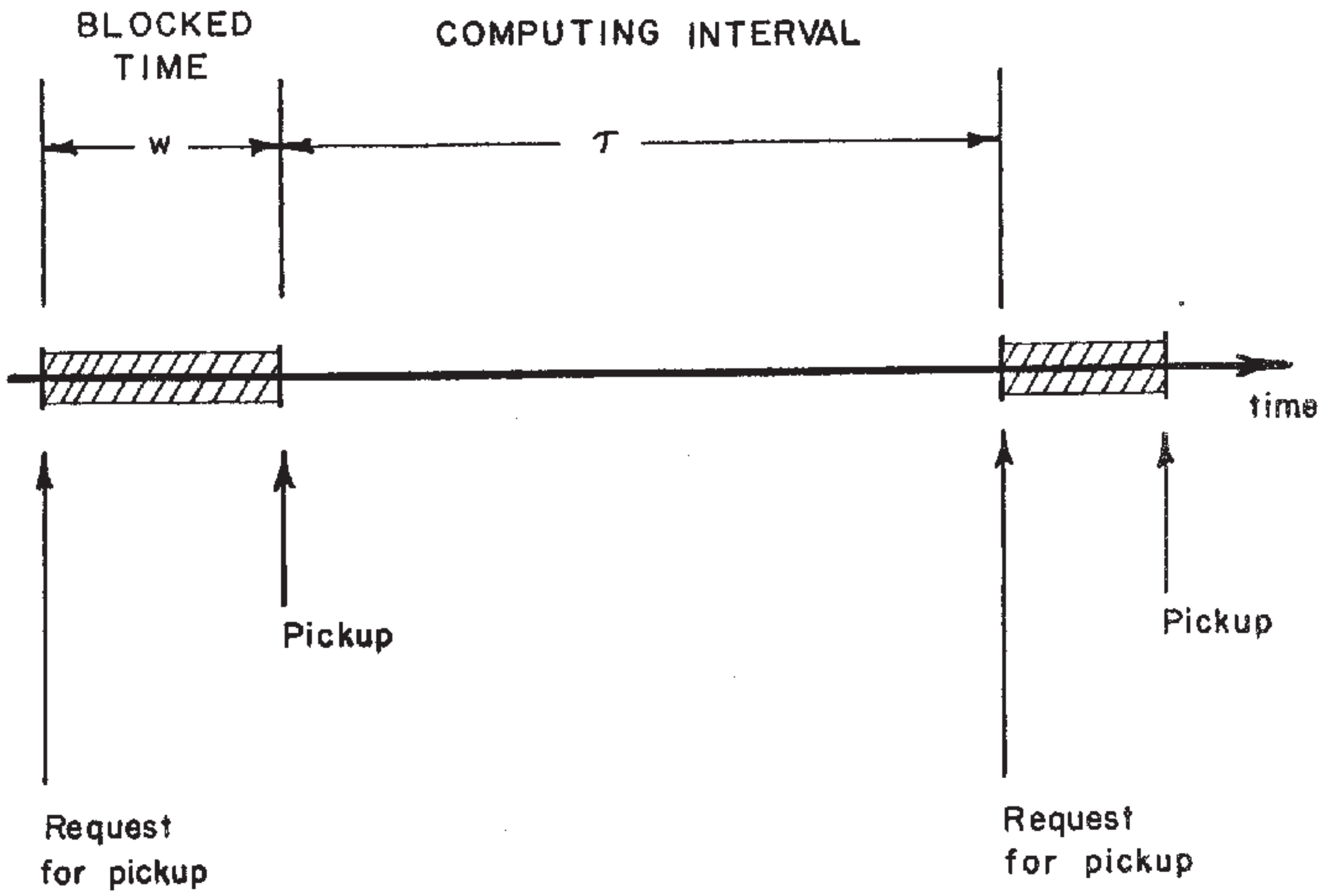
[2]Feller, W. An Introduction to Probability Theory and its Applications. New York: Wiley, 1950. (p. 143.)

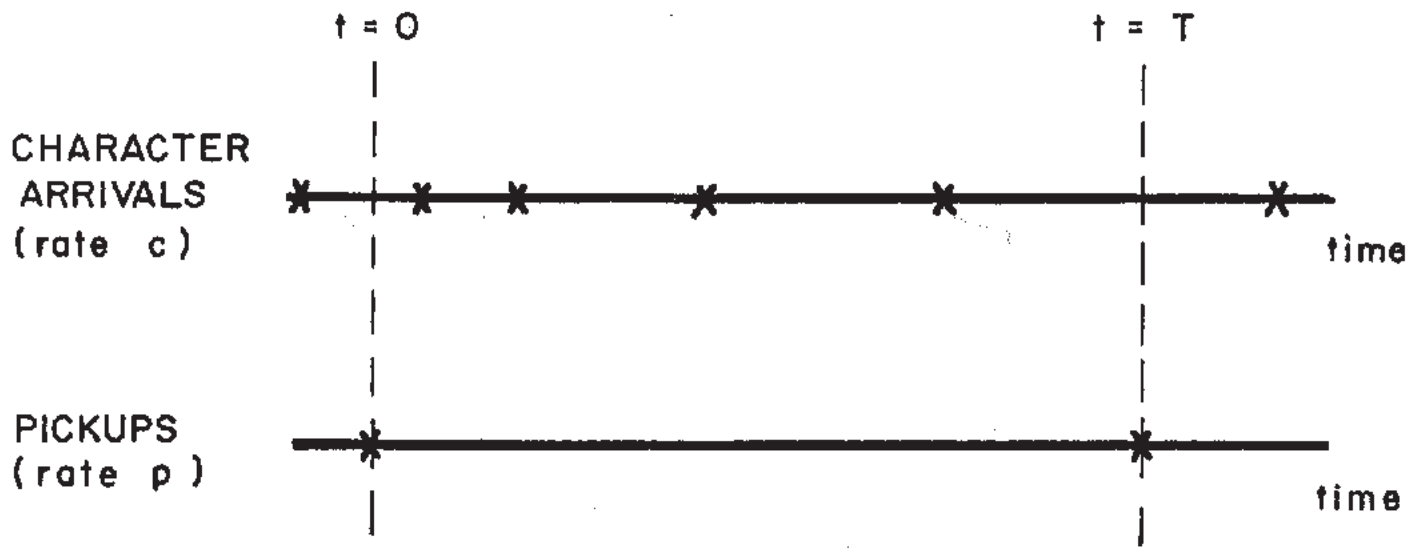
[3]Wozencraft, J. M., and Jacobs, I. M. Principles of Communications Engineering. New York: Wiley, 1965. (Chapter 1.)

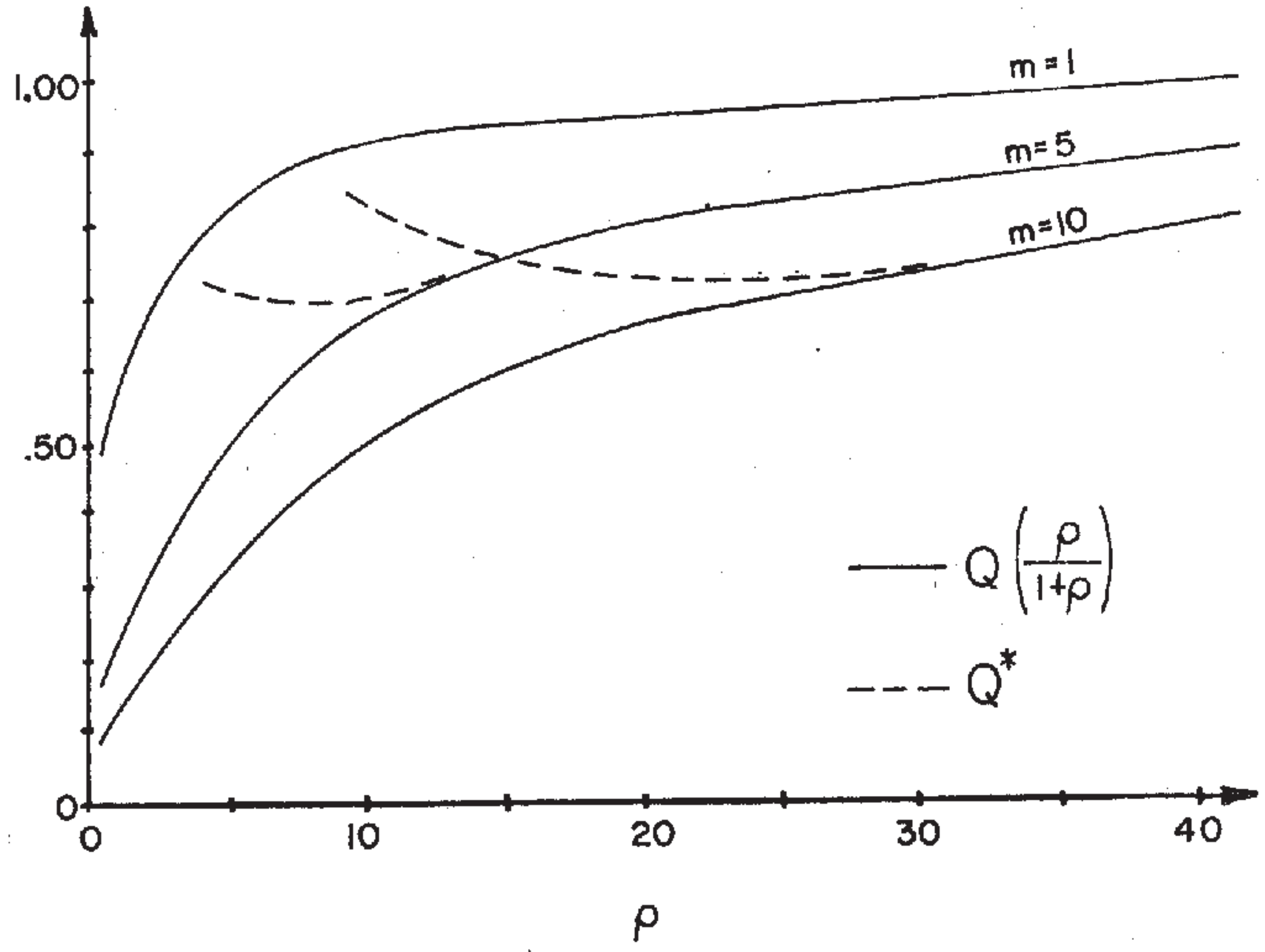
LIST OF FIGURE CAPTIONS



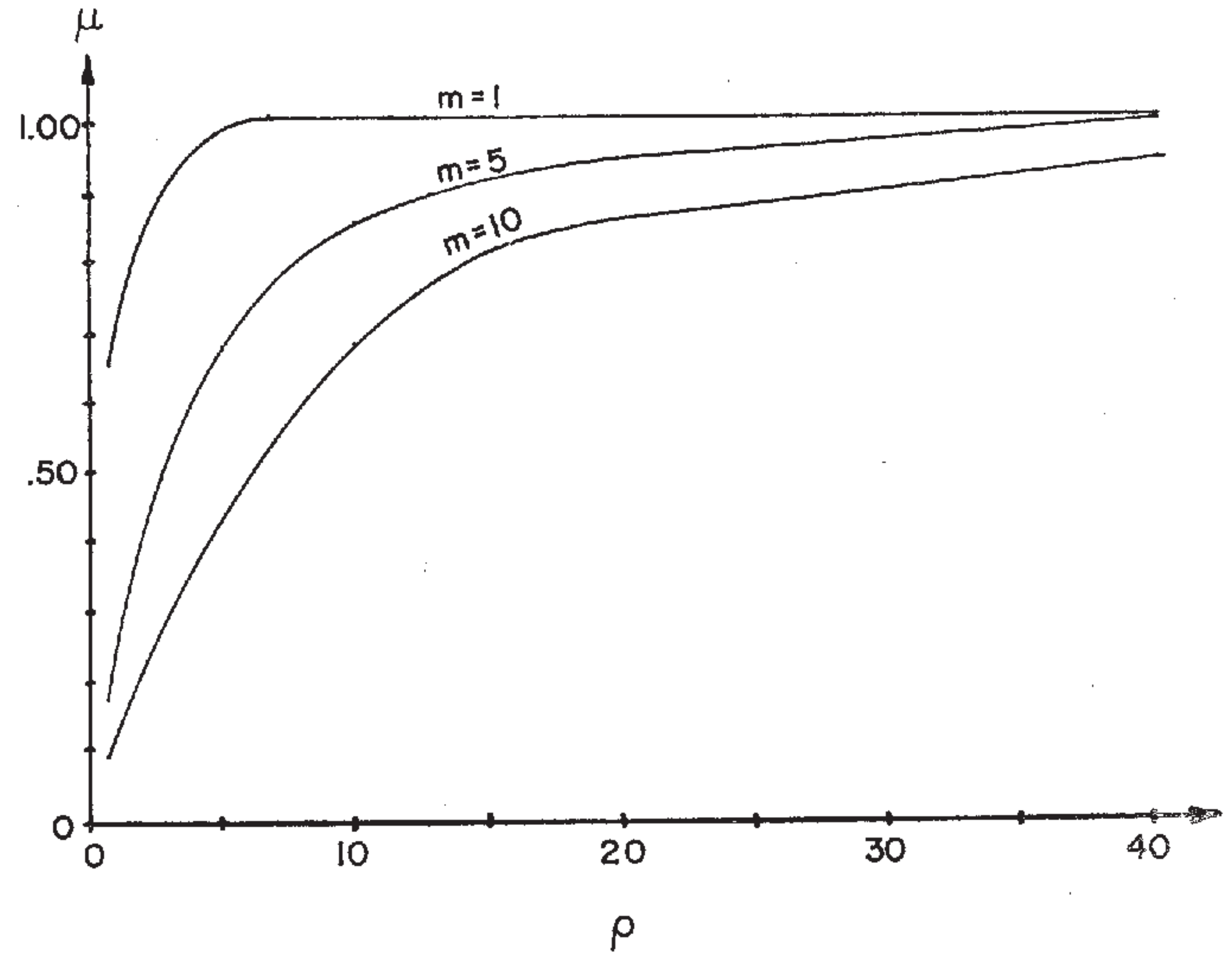
TYPEWRITER CONSOLES
c_1
c_2
c_N
SORTER
y_1 characters
y_2 characters
y_N characters
BUFFERS
USER PROCESSES MAKING PICKUPS
p_1
p_2 ...
p_N


BLOCKED TIME
COMPUTING INTERVAL
w
τ
time
Pickup
Pickup
Request for pickup
Request for pickup

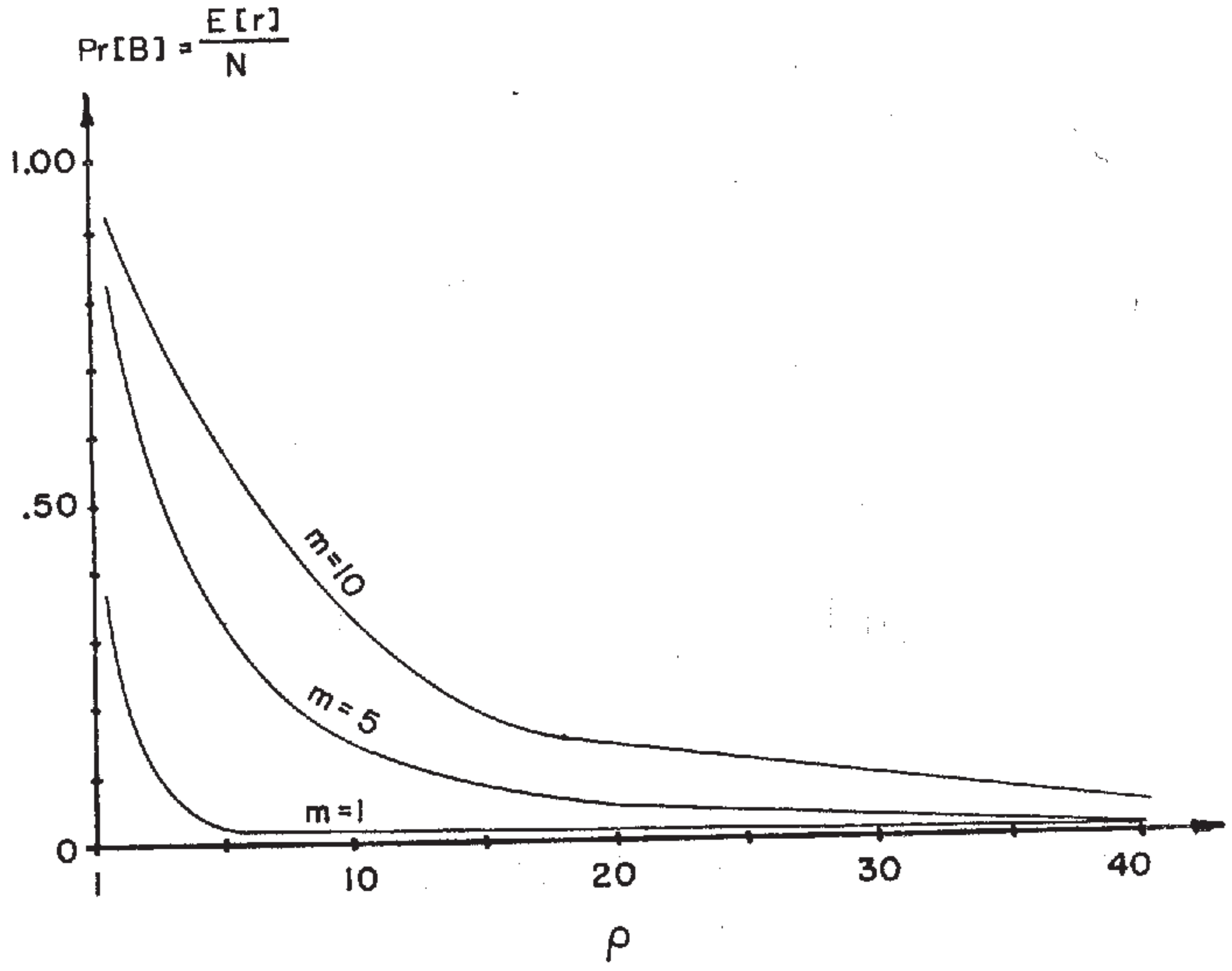

t = O
t = T
CHARACTER ARRIVALS (rate c)
PICKUPS (rate p)
time
time



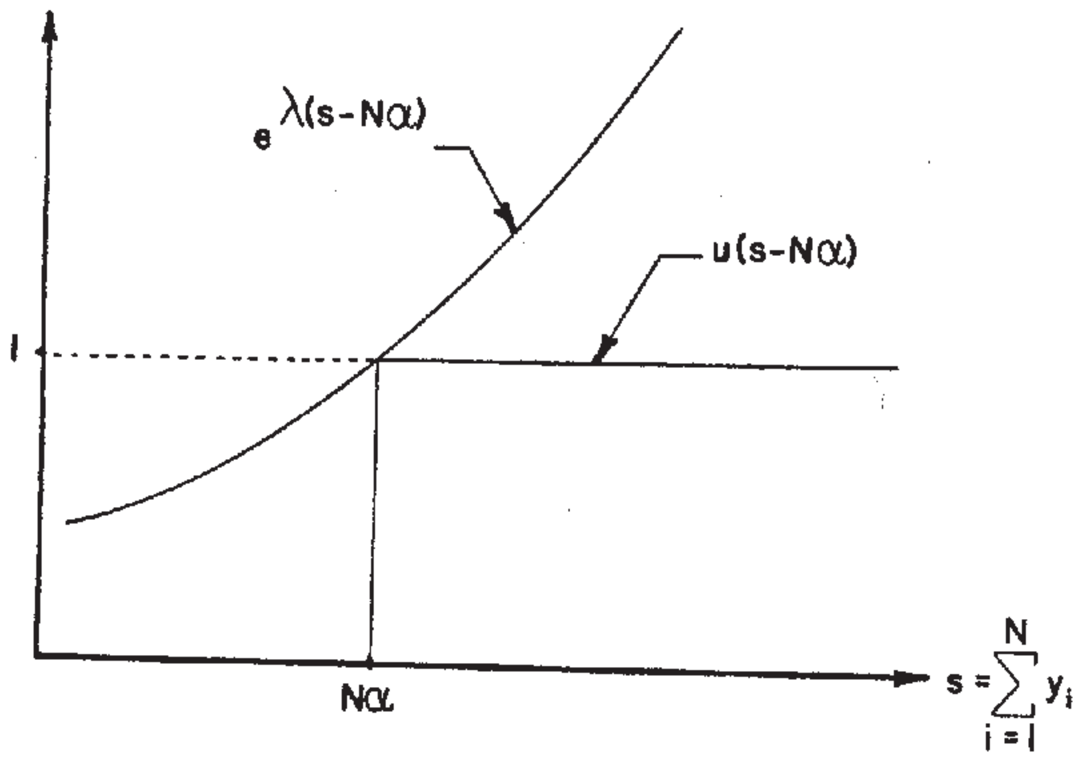
1.00
.50
0
0
10
20
30
40
ρ
m = 1
m = 5
m = 10
$Q\left(\frac{\rho}{1+\rho}\right)$
Q^*



μ
1.00
.50
0
m = 1
m = 5
m = 10
0
10
20
30
40
ρ



$Pr[B] = \frac{E[r]}{N}$
1.00
.50
0
m=10
m=5
m=1
1
10
20
30
40
ρ



$e^{\lambda(s-N\alpha)}$
$u(s-N\alpha)$
1
$N\alpha$
$s=\sum_{i=1}^{N} y_i$