Open access • Journal Article • DOI:10.1145/321707.321717

Processor Sharing Queueing Models of Mixed Scheduling Disciplines for Time Shared System — Source link

Leonard Kleinrock, Richard R. Muntz

Institutions: University of California, Los Angeles

Published on: 01 Jul 1972 - Journal of the ACM (ACM)

Topics: Processor sharing, Queueing theory, Scheduling (computing) and First-come, first-served

Processor Sharing Queueing Models of Mixed Scheduling Disciplines for Time Shared Systems

L. KLEINROCK AND R. R. MUNTZ

University of California, Los Angeles, California

ABSTRACT. Scheduling algorithms for time shared computing facilities are considered in terms of a queueing theory model. The extremely useful limit of "processor sharing" is adopted, wherein the quantum of service shrinks to zero; this approach greatly simplifies the problem. A class of algorithms is studied for which the scheduling discipline may change for a given job as a function of the amount of service received by that job. These multilevel disciplines form a natural extension to many of the disciplines previously considered.

The average response time for jobs conditioned on their service requirement is solved for. Explicit solutions are given for the system M/G/1 in which levels may be first come first served (FCFS), feedback (FB), or round-robin (RR) in any order. The service time distribution is restricted to be a polynomial times an exponential for the case of RR.

Examples are described for which the average response time is plotted. These examples display the great versatility of the results and demonstrate the flexibility available for the intelligent design of discriminatory treatment among jobs (in favor of short jobs and against long jobs) in time shared computer systems.

KEY WORDS AND PHRASES: time sharing, operating systems, queues, mathematical models

CR CATEGORIES: 3.89, 4.39, 5.5

1. *Introduction*

Queueing models have been used successfully in the analysis of time shared computer systems since the appearance of the first applied paper in this field in 1964 [1]. A recent survey of this work is given by McKinney [2]. One of the first papers to consider the effect of feedback in queueing systems was due to Takács [3].

One of the goals in a time shared computer system is to provide rapid response to those tasks which are interactive and which place frequent, but small, demands on the system. As a result, the system scheduling algorithm must identify those demands which are small, and provide them with preferential treatment over larger demands. Since we assume that the scheduler has no explicit knowledge of job processing times, this identification is accomplished implicitly by "testing" jobs. That is, jobs are rapidly provided small amounts of processing and, if they are short, they will depart rather quickly; otherwise, they will linger while other, newer jobs are provided this rapid service, etc., thus providing good response to small demands.

Authors' address: University of California, Computer Science Department, School of Engineering and Applied Science, Los Angeles, CA 90024. This work was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC-15-69-C-0285).

Generally, an arrival enters the time shared system and competes for the attention of a single processing unit. This arrival is forced to wait in a system of queues until he is permitted a quantum of service time; when this quantum expires, he is then required to join the system of queues to await his second quantum, etc. The precise model for the system is developed in Section 2. In 1967 the notion of allowing the quantum to shrink to zero was studied [4] and was referred to as "processor sharing"; in 1966 Schrage [18] also studied the zero-quantum limit. As the name implies, this zero-quantum limit provides a share or portion of the processing unit to many customers simultaneously; in the case of round-robin (RR) scheduling [4], *all* customers in the system simultaneously share (equally or on a priority basis) the processor, whereas in the feedback (FB) scheduling [5] only that *set* of customers with the smallest attained service share the processor. We use the term processor sharing here since it is the processing unit itself (the central processing unit of the computer) which is being shared among the set of the customers; the phrase "time sharing" is reserved to imply that customers are waiting sequentially for their turn to use the entire processor for a finite quantum. In studying the literature one finds that the obtained results appear in a rather complex form and this complexity arises from the fact that the quantum is typically assumed to be finite as opposed to infinitesimal. When one allows the quantum to shrink to zero, giving rise to a processor sharing system, then the difficulty in analysis as well as in the form of results disappears in large part; one is thus encouraged to consider the processor sharing case. Clearly, this limit of infinitestimal quantum[1] is an ideal and can seldom be reached in practice due to overhead considerations; nevertheless, its extreme simplicity in analysis and results brings us to the studies reported in this paper.

The two processor sharing systems studied in the past are the RR and the FB [4, 5]. Typically, the quantity solved for is $T(t)$, the expected response time conditioned on the customer's service time t; response time is the elapsed time from when a customer enters the system until he leaves completely serviced. This measure is especially important since it exposes the preferential treatment given to short jobs at the expense of the long jobs. Clearly, this discrimination is purposeful since, as stated above, it is the desire in time shared systems that small requests should be allowed to pass through the system quickly. In 1969 the distribution for the response time in the RR system was found [6]. In this paper we consider the case of *mixed* scheduling algorithms whereby customers are treated according to the RR algorithms, the FB algorithm, or first come first served (FCFS) algorithm, depending upon how much total service time they have already received. Thus, as a customer proceeds through the system obtaining service at various rates, he is treated according to different disciplines; the policy which is applied among customers in different levels is that of the FB system as explained further in Section 2. Thus, natural generalization of the previously studied processor sharing systems allows us to create a large number of new and interesting disciplines whose solutions we present.

A more restricted study of this sort was reported by the authors in [16]. Here we make use of the additional results from [11] to generalize our analysis.

[1] This limiting case is not unlike the diffusion approximation for queues (see, for example, Gaver [17]).

2. *The Model*

The model we choose to use in representing the scheduling algorithms is drawn from queueing theory. This corresponds to the many previous models studied [1, 2, 4–8, 18], all of which may be thought of in terms of the structure shown in Figure 1. In this figure we indicate that new requests enter the system queues upon arrival. Whenever the computer's central processing unit (CPU) becomes free, some customer is allowed into the service facility for an amount of time referred to as a quantum. If, during this quantum, the total accumulated service for a customer equals his required service time, then he departs the system; if not, at the end of his quantum, he cycles back to the system of queues and waits until he is next chosen for additional service. The system of queues may order the customers according to a variety of different criteria in order to select the next customer to receive a quantum. In this paper we assume that the only measure used in evaluating this criterion is the amount of attained service (total service so far received).

In order to specify the scheduling algorithm in terms of this model, it is required that we identify the following:

(a) *Customer interarrival time distribution.* We assume this to be exponential, i.e.

$$P[\text{interarrival time} \leq t] = 1 - e^{-\lambda t}, \quad t \geq 0, \tag{2.1}$$

where λ is the average arrival rate of customers.

(b) *Distribution of required service time in the CPU.* This we assume to be arbitrary (but independent of the interarrival times). We thus assume

$$P[\text{service time} \leq x] = B(x). \tag{2.2}$$

Also assume $1/\mu$ = average service time.

(c) *Quantum size.* Here we assume a processor shared model in which customers receive an equal but vanishingly small amount of service each time they are allowed into service. For more discussion of such systems, see [4, 6, 7, 18].

(d) *System of queues.* We consider here a generalization and consolidation of many systems studied in the past. In particular, we define a set of attained service times $\{a_i\}$ such that

$$0 = a_0 < a_1 < a_2 < \cdots < a_N < a_{N+1} = \infty. \tag{2.3}$$

The discipline followed for a job when it has attained service, τ, in the interval

$$a_{i-1} \leq \tau < a_i, \qquad i - 1, 2, \cdots, N + 1, \tag{2.4}$$

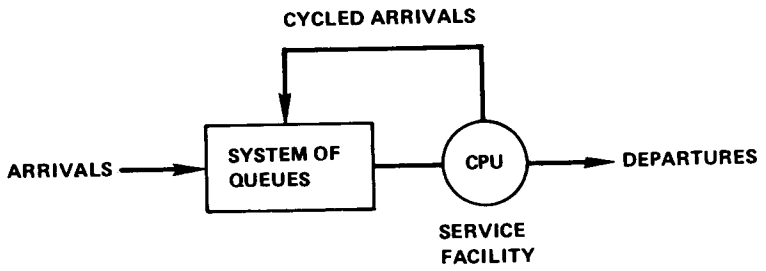

FIG. 1. The feedback queueing model.

will be denoted as D_i. We consider D_i for any given level i to be either: First Come First Served (FCFS); Processor Shared–FB$_\infty$ (FB); or Round-Robin Processor Shared (RR). The FCFS system needs no explanation, and the FB system gives service next to that customer who so far has least attained service; if there is a tie (among K customers, say) for this position, then all K members in the tie get served simultaneously (each attaining useful service at a rate of $1/K$ sec/sec), this being the nature of processor sharing systems. The RR processor sharing system shares the service facility among all customers present (say J customers) giving attained service to each at a rate of $1/J$ sec/sec. Moreover, between intervals, the jobs are treated as a set of FB disciplines (i.e. service proceeds in the ith level only if all levels $j < i$ are empty). See Figure 2. For example, for $N = 0$, we have the usual single level case of either FCFS, RR, or FB. For $N = 1$, we could have any of nine disciplines (FCFS followed by FCFS, $\cdots$, RR followed by RR); note that FB followed by FB is just a single FB system (due to overall FB policy between levels).

As an illustrative example, consider the $N = 2$ case shown in Figure 3. Any new arrivals begin to share the processor in an RR fashion with all other customers who so far have less than 2 seconds of attained service. Customers in the range of $2 \leq \tau < 6$ may get served only if no customers present have had less than 2 seconds of service; in such a case, that customer (or customers) with the least attained service will proceed to occupy the service in an FB fashion until they either leave, or reach $\tau = 6$, or some new customer arrives (in which case the overall FB rule provides that the RR policy at level 1 preempts). If all customers have $\tau \geq 6$, then the "oldest" customer will be served to completion unless interrupted by a new arrival. The history of some customers in this example system is shown in Figure 4. We denote customer n by C_n. Note that the slope of attained service varies as the number of customers simultaneously being serviced changes. We see that C_2 requires 5 seconds of service and spends 14 seconds in system (i.e. response time of 14 seconds).

So much for the system specification. We may summarize by saying that we have an M/G/1 queueing system[2] model with processor sharing and with a generalized multilevel scheduling structure.

The quantity we wish to solve for is

$$T(t) = E[\text{response time for a customer requiring a total of } t \text{ seconds of attained service}]. \tag{2.5}$$

We further make the following definitions:

$$T_i(t) = E\{\text{time spent in interval } i\ [a_{i-1}, a_i] \text{ for customers requiring a total of } t \text{ seconds of attained service}\}. \tag{2.6}$$

We note that

$$T_i(t) = T_i(t') \qquad \text{for } t, t' \geq a_i. \tag{2.7}$$

[2] M/G/1 denotes the single-server queueing system with Poisson arrivals and arbitrary service time distribution; similarly M/M/1 denotes the more special case of exponential service time distribution. One might also think of our processor sharing system as an infinite server model with constant overall service rate.

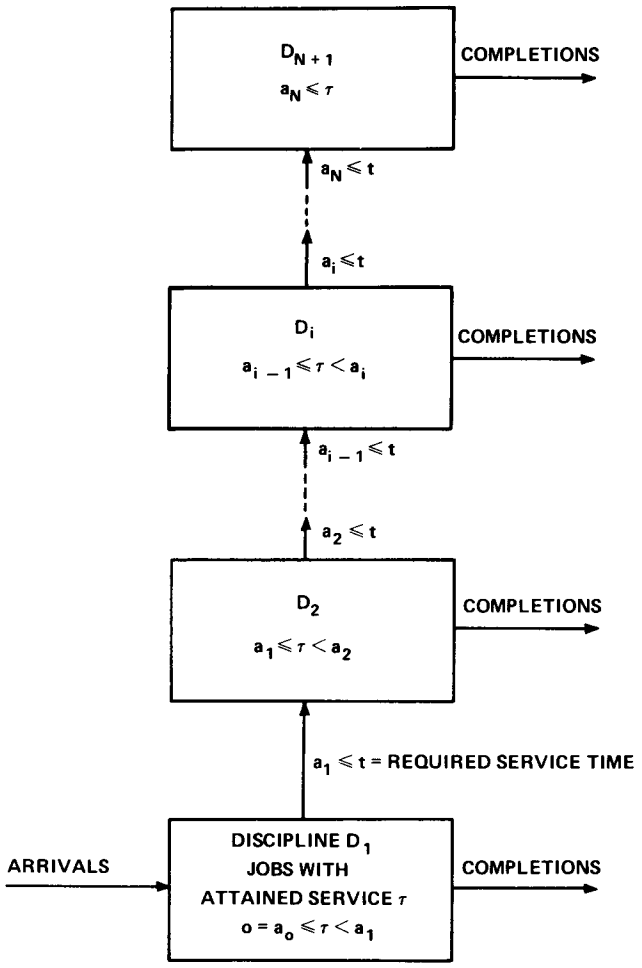

FIG. 2. The multilevel queueing structure with disciplines D_i.

Furthermore, we have, for $a_{k-1} \leq t < a_k$, that

$$T(t) = \sum_{i=1}^{k} T_i(t). \tag{2.8}$$

Also we find it convenient to define the following quantities with respect to $B(t)$:

$$\bar{t}_{<x} = \int_0^x t dB(t) + x \int_x^\infty dB(t), \tag{2.9}$$

$$\bar{t}^2_{<x} = \int_0^\infty t^2 dB(t) + x^2 \int_x^\infty dB(t), \tag{2.10}$$

$$\rho_{<x} = \lambda \bar{t}_{<x}, \tag{2.11}$$

$$W_x = \lambda \bar{t}^2_{<x} / 2(1 - \rho_{<x}). \tag{2.12}$$

Note that W_x represents the expected work found by a new arrival in the system M/G/1 where the service times are truncated at x.

3. *Results for Multilevel Queueing Systems*

We wish to find an expression for $T(t)$, the mean system time (i.e. average response time) for jobs with service time t such that $a_{i-1} \leq t < a_i$, i.e. jobs which reach the ith level queue and there leave the system. To accomplish this it is convenient to isolate the ith level to some extent. We make use of the following two facts.

(1) By the assumption of preemptive priority of lower level queues (i.e. FB discipline between levels) it is clear that jobs in levels higher than the ith level can

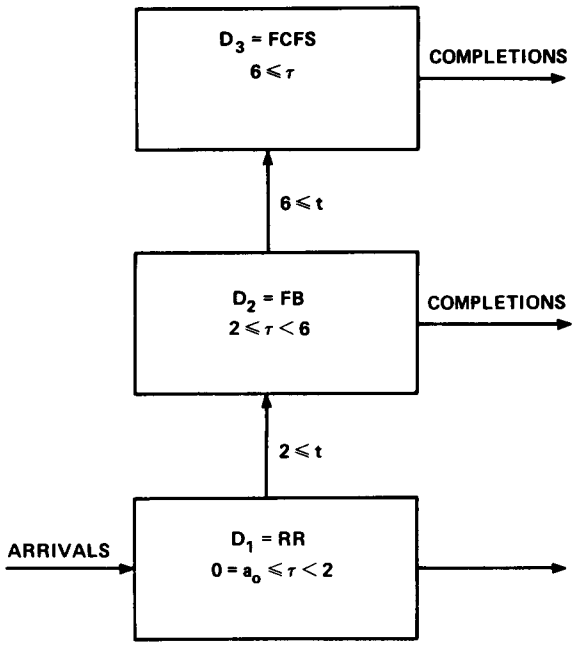

FIG. 3. Example of $N = 2$.

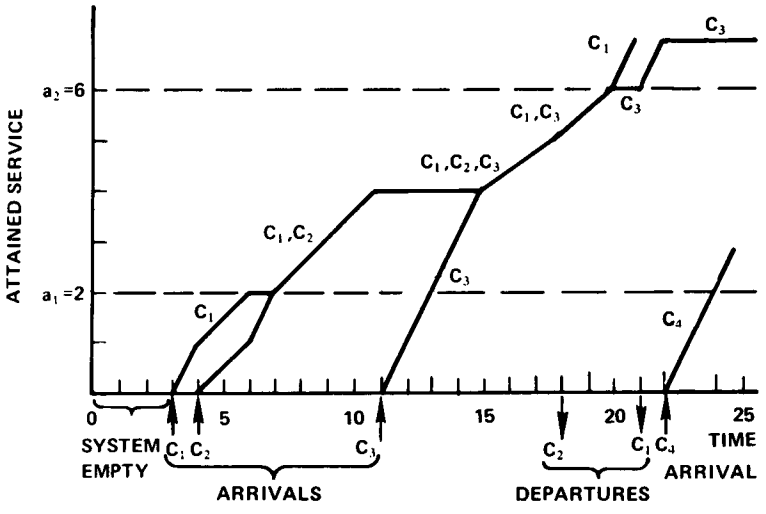

FIG. 4. History of customers in example.

be ignored. This follows since these jobs cannot interfere with the servicing of the lower levels.

(2) We are interested in jobs that will reach the ith level queue and then depart from the system before passing to the $(i + 1)$-th level. The system time of such a job can be thought of as occurring in two parts. The first portion is the time from the job's arrival to the queueing system until the group at the ith level is serviced for the first time after this job has reached the ith level. The second portion starts with the end of the first portion and ends when the job leaves the system. It is easy to see that both the first and second portions of the job's system time are unaffected by the service disciplines used in levels 1 through $i - 1$. Therefore we can assume any convenient disciplines. In fact, all these levels can be lumped into one equivalent level which services jobs with attained service between 0 and a_{i-1} seconds using any service discipline.

From (1) and (2) above it follows that we can solve for $T(t)$ for jobs that leave the system from the ith level by considering a two level system. The lower level services jobs with attained service between 0 and a_{i-1}, whereas the second level services jobs with attained service between a_{i-1} and a_i. Jobs that would have passed to the $(i + 1)$-th level after receiving a_i seconds of service in the original system are now assumed to leave the system at that point. In other words the service times are truncated at a_i.

3.1 iTH LEVEL DISCIPLINE IS FB. Consider the two level system with the second level corresponding to the ith level of the original system. Since we are free to choose this discipline used in the lower level, we can assume that the FB discipline is used in this level as well. Clearly the two level system behaves like a single level FB system with service times truncated at a_i. The solution for such a system is known [5, 9]:

$$T(t) = t/(1 - \rho_{<t}) + \lambda\overline{t^2}_{<t}/[2(1 - \rho_{<t})^2]. \tag{3.1}$$

3.2 iTH LEVEL DISCIPLINE IS FCFS. Consider again the two level system with breakpoints at a_{i-1} and a_i. Regardless of the discipline in the lower level, a tagged job entering the system will be delayed by the sum of (a) the work currently in both levels ($= W_{a_i}$) plus (b) any new arrivals to the lower level queue during the interval [average $T(t)$] this job is in the system. These new arrivals form a Poisson process with parameter λ and their contribution to the delay is a random variable whose first and second moments are $\bar{t}_{<a_{i-1}}$ and $\overline{t^2}_{<a_{i-1}}$ respectively.

Thus we have

$$T(t) = W_{a_i} + \lambda\bar{t}_{a_{i-1}} T(t) + t$$

and so

$$T(t) = (W_{a_i} + t)/(1 - \rho_{<a_{i-1}}), \tag{3.2}$$

where W_{a_i} is given by eq. (2.12). It is also possible to use these methods for solving last come first served and random order of service at any level.

3.3 iTH LEVEL DISCIPLINE IS RR. In this case, our results are limited in the ith interval to service distributions in which

$$B(x) = 1 - p(x)e^{-\beta x}, \qquad a_{i-1} \leq x < a_i, \tag{3.3}$$

$$p(x) = p_0 + p_1 x + \cdots + p_n x^n. \tag{3.4}$$

The service time distribution $F(x)$ for this ith interval is then

$$F(x) = \begin{cases} \dfrac{B(a_{i-1} + x) - B(a_{i-1})}{1 - B(a_{i-1})} = 1 - q(x)e^{-\beta x} & 0 \leq x < a_i - a_{i-1} \\ 1 & x \geq a_i - a_{i-1}, \end{cases} \tag{3.3'}$$

where

$$q(x) = \frac{e^{-\beta a_{i-1}} p(a_{i-1} + x)}{1 - B(a_{i-1})} = q_0 + q_1 x + \cdots + q_n x^n. \tag{3.4'}$$

Thus we permit in this interval service time distributions of the form: 1 minus a polynomial of degree n times an exponential. The analysis of this system appears in [11]; we make use of these results below. Nevertheless, we develop our analysis as far as possible for the case of general $B(x)$ before specializing to the class given by eqs. (3.3) and (3.4).

We start by considering the two level system with breakpoints at a_{i-1} and a_i. Consider the busy periods of the lower level. During each such busy period there may be a number of jobs that pass to the higher level. We choose to consider these arrivals to the higher level as occurring at the end of the lower level busy period so that there is a bulk arrival to the higher level at this time. We also choose to temporarily delete these lower level busy periods from the time axis. In effect we create a *virtual* time axis telescoped to delete the lower level busy periods. Since the time from the end of one lower level busy period to the start of the next is exponentially distributed (Poisson arrivals!), the arrivals to the higher level appear in virtual time to be *bulk* arrivals at instants generated from a Poisson process with parameter λ.

Consider a tagged job that required $t = a_{i-1} + \tau$ seconds of service ($0 < \tau \leq a_i - a_{i-1}$). Let α_1 be the mean *real* time the job spends in the system until its arrival (at the end of the lower level busy period) at the higher level queue. Let $\alpha_2(\tau)$ be the mean *virtual* time the job spends in the higher level queue. α_1 can be calculated as follows. The initial delay is equal to the mean work the job finds in the lower level on arrival plus the a_{i-1} seconds of work that it contributed to the lower level. Therefore

$$\alpha_1 = W_{a_{i-1}} + \lambda \bar{t}_{a_{i-1}} \alpha_1 + a_{i-1}$$

and so

$$\alpha_1 = [1/(1 - \rho_{<a_{i-1}})]\{W_{a_{i-1}} + a_{i-1}\}. \tag{3.5}$$

If $\alpha_2(\tau)$ is the mean *virtual* time the job spends in the higher level, we can easily convert this to the mean *real* time spent in this level. In the virtual time interval $\alpha_2(\tau)$ there are an average of $\lambda\alpha_2(\tau)$ lower level busy periods that have been ignored. Each of these has a mean length of $\bar{t}_{<a_{i-1}}/(1 - \rho_{<a_{i-1}})$. Therefore, the mean real time the job spends in the higher level is given by

$$\alpha_2(\tau) + \lambda\alpha_2(\tau) \cdot t_{<a_{i-1}}/(1 - \rho_{<a_{i-1}}) = \alpha_2(\tau)/(1 - \rho_{<a_{i-1}}). \tag{3.6}$$

Combining these results we see that a job requiring $t = a_{i-1} + \tau$ seconds of service has mean system time given by

$$T(a_{i-1} + \tau) = [1/(1 - \rho_{<a_{i-1}})]\{W_{a_{i-1}} + a_{i-1} + \alpha_2(\tau)\}. \tag{3.7}$$

The only unknown quantity in this equation is $\alpha_2(\tau)$. To solve for $\alpha_2(\tau)$ we must, in general, consider an M/G/1 system with bulk arrival and RR processor sharing. The only exception is the case of RR at the first level which has only single arrivals. Since the higher level queues can be ignored, the solution in this exceptional case is the same as for a round-robin single level system with service times truncated at a_1. Therefore, from [8] we have for the first level,

$$T(t) = t/(1 - \rho_{<a_1}) \qquad 0 \leq t < a_1 . \tag{3.8}$$

Let us now consider the *bulk arrival RR system* in isolation in order to solve for the virtual time spent in the higher level queue $\alpha_2(\tau)$. The service time distribution for this bulk arrival system is

$$F(x) = \begin{cases} [B(a_{i-1} + x) - B(a_{i-1})]/[1 - B(a_{i-1})], & 0 \leq x < a_i - a_{i-1} , \\ 1, & x \geq a_i - a_{i-1} . \end{cases}$$

Note that the utilization factor for this bulk system is

$$\rho = \lambda \bar{a} / \mu_i , \tag{3.9}$$

where $\bar{a}$ is the mean number of arrivals in a bulk and $1/\mu_i$ is the mean of the distribution $F(x)$. Let us begin by solving for $\bar{a}$. This we do for the general case a_{i-1}, a_i. $\bar{a}$ is just the mean number of jobs that arrive during a low level busy period and require more than a_{i-1} seconds of service. Therefore $\bar{a}$ must satisfy the equation

$$\bar{a} = \lambda \bar{t}_{<a_{i-1}} \bar{a} + [1 - B(a_{i-1})]1. \tag{3.10}$$

In this equation $\lambda t_{<a_{i-1}}$ is the mean number of jobs that arrive during the service time of the first job in the busy period. Since each of these jobs in effect generates a busy period, there are an average of $\lambda \bar{t}_{<a_{i-1}} \bar{a}$ arrivals to the upper level queue due to these jobs. The second term is just the average number of times that the first job in the busy period will require more than a_{i-1} seconds of service. Clearly then

$$\bar{a} = [1 - B(a_{i-1})]/[1 - \rho_{<a_{i-1}}]. \tag{3.11}$$

In [11], an integral equation is derived which defines $\alpha_2(\tau)$ for the RR bulk arrival system; we repeat that equation below:

$$\begin{aligned} \alpha_2'(\tau) = {} & \lambda \bar{a} \int_0^\infty \alpha_2'(x)[1 - F(x + \tau)]\, dx \\ & + \lambda \bar{a} \int_0^\infty \alpha_2'(x)[1 - F(\tau - x)]\, dx \\ & + 1 + b[1 - F(\tau)], \end{aligned} \tag{3.12}$$

where $\alpha_2'(\tau) = d\alpha_2(\tau)/d\tau$, and b is the mean number of arrivals with the tagged job. The solution to this integral equation for the restricted service time distributions as given in eqs. (3.3′) and (3.4′) is also given in [11], and for our problem takes the form

$$\begin{aligned} \alpha_2(\tau) = {} & \frac{\tau}{1 - \lambda \bar{a} \dfrac{1}{\mu_i}} - \frac{b}{\lambda \bar{a}} \sum_{m=1}^{n+1} \frac{(\beta^2 - \gamma_m^2)^{n+1}}{Q_2'(\gamma_m)} \\ & \cdot \frac{Q_0(\gamma_m)[1 - e^{-\gamma_m \tau}] - Q_1(\gamma_m) e^{-\gamma_m x_1}[e^{\gamma_m \tau} - 1]}{[Q_0(\gamma_m) + e^{-\gamma_m x_1} Q_1(\gamma_m)]\gamma_m} , \end{aligned} \tag{3.13}$$

where

$$x_1 = a_i - a_{i-1}, \tag{3.14}$$

$$Q_0(x) = (x + \beta)^{n+1} - \lambda\bar{a} \sum_{k=0}^{n} q^{(k)}(0)(x + \beta)^{n-k}, \tag{3.15}$$

$$Q_1(x) = \lambda\bar{a} \sum_{k=0}^{n} e^{-\beta x_1} q^{(k)}(x_1)(x + \lambda)^{n-k}, \tag{3.16}$$

$$Q_2(x) = Q_0(x)Q_0(-x) - Q_1(x)Q_1(-x), \tag{3.17}$$

and where eq. (3.17) has roots (occurring in pairs) $x = -\gamma_{m'}, \gamma_m$ for $m = 1, 2, \cdots, n + 1$ and the notation $f^{(k)}(\gamma)$ denotes the kth derivative of f with respect to its argument evaluated at a value γ.

In the solution for $\alpha_2(\tau)$ given in eq. (3.13), we are required to compute b (mean number of arrivals with a tagged job). This we do by first deriving an expression for

$$G(z) = \sum_{k=0}^{\infty} P[\text{bulk size} = k]z^k \tag{3.18}$$

which is the probability generating function (z-transform) for the bulk size. Either by direct arguments based upon busy periods or by use of the method of collective marks [12], we readily arrive at

$$G(z) = [1 - B(a_{i-1})]z \sum_{j=0}^{\infty} \frac{(\lambda a_{i-1})^j}{j!} e^{-\lambda a_{i-1}}[G(z)]^j + B(a_{i-1}) \left[\int_0^{a_{i-1}} \sum_{j=0}^{\infty} \frac{(\lambda t)^j}{j!} e^{-\lambda t}[G(z)]^j \frac{dB(t)}{B(a_{i-1})} \right]. \tag{3.19}$$

In eq. (3.19) the first term is conditioned on the assumption that the customer who preempts service from those at level i reaches the ith level; the second term assumes that he does not reach level i. Equation (3.19) reduces to

$$G(z) = [1 - B(a_{i-1})]ze^{-\lambda a_{i-1}[1-G(z)]} + \int_0^{a_{i-1}} e^{-\lambda t[1-G(z)]}\, dB(t). \tag{3.20}$$

For arbitrary $B(x)$, we cannot reduce this last expression any further. Nevertheless, we can obtain moments from it. In particular, from the definition of $\bar{a}$, we obtain

$$\bar{a} = G'(z)\,|_{z=1} = [1 - B(a_{i-1})]/[1 - \lambda\bar{t}_{<a_{i-1}}],$$

which is exactly as obtained by more direct arguments in eq. (3.11). However, we are seeking b. For this we must calculate

$$G''(z)\,|_{z=1} = [(\bar{a})^2/(1 - \rho_{<a_{i-1}})][2\lambda a_{i-1}(1 - \rho_{<a_{i-1}}) + \lambda^2\overline{t^2}_{<a_{i-1}}]. \tag{3.21}$$

Now since the mean group size $(1 + b)$ of a tagged customer's group is related to the bulk size distribution as the mean spread is related to the inter-event distribution (namely, the mean spread equals the second moment of the inter-event interval divided by the first moment) [13], we have

$$1 + b = (\text{second moment of bulk size})/(\text{first moment of bulk size}) \tag{3.22}$$

or

$$b = \left. \frac{G''(z)}{G'(z)} \right|_{z=1}. \tag{3.23}$$

From eq. (3.20) we get

$$b = [\bar{a}/(1 - \rho_{<a_{i-1}})][2\lambda a_{i-1}(1 - \rho_{<a_{i-1}}) + \lambda^2 \bar{t}^2_{<a_{i-1}}]. \tag{3.24}$$

Having solved for $\alpha_2(\tau)$ we may now substitute back into eq. (3.7) which solves the case when the ith level discipline is RR and service time is of the form given in eqs. (3.3′) and (3.4′). [Note that for $i = 1$, the solution given in eq. (3.8) is good for any $B(x)$.]

It is instructive to display the solution for $T(t)$ explicitly in a special case for our ith level RR discipline. We choose the multilevel system with M/M/1 and solve for $T(a_{i-1} + \tau)$ after substituting $\alpha_2(\tau)$ into eq. (3.7). Note for M/M/1 that $q(t) = q_0 = 1$. Also, from eqs. (3.14)–(3.17), and choosing $\beta = \mu$,

$$Q_0(x) = x + \mu - \lambda\bar{a}, \tag{3.25}$$

$$Q_1(x) = \lambda\bar{a}e^{-\mu x_1}, \tag{3.26}$$

$$Q_2(x) = \mu^2 - 2\mu\lambda\bar{a} + (\lambda\bar{a})^2(1 - e^{-2\mu x_1}) - x^2; \tag{3.27}$$

thus the roots of $Q_2(x)$ are

$$\pm\gamma_1 = \pm[\mu^2 - 2\mu\lambda\bar{a} + (\lambda\bar{a})^2(1 - e^{-2\mu x_1})]^{\frac{1}{2}}, \tag{3.28}$$

$$\mu_i^{-1} = \mu^{-1}(1 - e^{-\mu x_1}); \tag{3.29}$$

thus from these and eq. (3.13), we get

$$\alpha_2(\tau) = \frac{\tau}{1 - \lambda\bar{a}\mu_i^{-1}} + \frac{b(\mu^2 - \gamma_1^2)[(\gamma_1 + \mu - \lambda\bar{a})(1 - e^{-\gamma_1\tau}) - \lambda\bar{a}e^{-(\mu+\gamma_1)x_1}(e^{\gamma_1\tau} - 1)]}{2\lambda\bar{a}\gamma_1^2[\gamma_1 + \mu - \lambda\bar{a}(1 - e^{-(\mu+\gamma_1)x_1})]}. \tag{3.30}$$

Also from eqs. (2.9) and (2.10) we obtain

$$\bar{t}_{<x} = \mu^{-1}(1 - e^{-\mu x}), \tag{3.31}$$

$$\bar{t}^2_{<x} = 2\mu^{-2}(1 - e^{-\mu x} - \mu x e^{-\mu x}). \tag{3.32}$$

We may substitute these last two equations into eqs. (3.11) and (3.24) to obtain a and b explicitly. Also, we note from eqs. (2.12) and (3.32) that

$$W_{a_{i-1}} = [\lambda(1 - e^{-\mu a_{i-1}} - \mu a_{i-1}e^{-\mu a_{i-1}})]/\mu^2[1 - (\lambda/\mu)(1 - e^{-\mu a_{i-1}})]. \tag{3.33}$$

Finally, we may substitute this expression for $W_{a_{i-1}}$ and eq. (3.30) which gives $\alpha_2(\tau)$ into eq. (3.7) which gives us the explicit form for $T(\tau)$.

4. *Examples*

In this section we demonstrate through examples the nature of the results we have obtained. Recall that we have given explicit solutions for our general model in the case M/G/1 with processor sharing where the allowed scheduling disciplines within a given level may be FCFS or FB; if the discipline is RR, it may be at level 1 and if it occurs at level $i > 1$, must be of the form given in eqs. (3.3′) and (3.4′).

We begin with four examples from the system M/M/1. As mentioned in Section 2, we have nine disciplines for the case $N = 1$. This comes about since we allow any

one of three disciplines at level 1 and any one of three disciplines at level 2. As we have shown, the behavior of the average conditional response time in any particular level is independent of the discipline in all other levels; thus we have nine disciplines. In Figure 5 we show the behavior of each of the nine disciplines for the system $N = 1$. In this case we have assumed $\mu = 1$, $\lambda = 0.75$, and $a_1 = 2$. From eq. (3.1) we see that the response time for the FB system is completely independent of the values a_i and therefore the curve shown in Figure 5 for this response time is applicable to all of our M/M/1 cases. Note the inflection point in this curve and that the response time grows linearly as $t \to \infty$ [a phenomenon not observable from previously published figures but easily seen from eq. (3.1)]. As can be seen from its defining equation, the response time for FCFS is linear regardless of the level; the RR system at level 1 is also linear, but as we see from Figure 5 and from eq. (3.13) the RR at levels $i > 1$ is

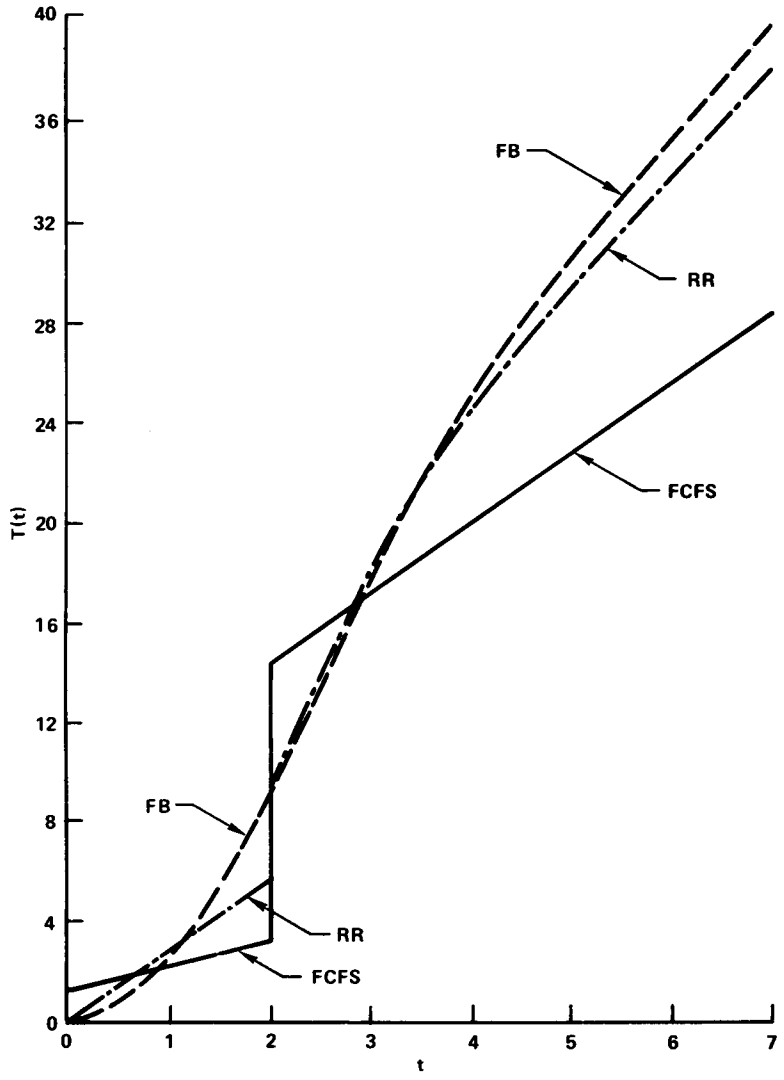

FIG. 5. Response time possibilities for $N = 1$, M/M/1, $\mu = 1$, $\lambda = 0.75$, $a_1 = 2$.

nonlinear. Thus one can determine the behavior of any of nine possible disciplines from Figure 5. Adiri and Avi-Itzhak considered the case (FB, RR) [14].

Continuing with the case M/M/1, we show in Figure 6 the case for $N = 3$ where $D_1 = \text{RR}$, $D_2 = \text{FB}$, $D_3 = \text{FCFS}$, and $D_4 = \text{RR}$. In this case we have chosen $a_i = i$ and $\mu = 1$, $\lambda = 0.75$. We also show in Figure 6 the case FB over the entire range as a reference curve for comparison with this discipline. Note (in general for M/M/1) that the response time for any discipline in a given level must either coincide with FB curve or lie above it in the early part of the interval and below it in the latter part of the interval; this is true due to the conservation law [15].

The third example for M/M/1 is for the iterated structure $D_i = \text{FCFS}$. Once again we have chosen $\mu = 1$, $\lambda = 0.75$, and $a_i = i$. Also shown in Figure 6 is a dashed line corresponding to the FB system over the entire range. Clearly, one may select any sequence of FB and FCFS with duplicates in adjacent intervals, and the be-

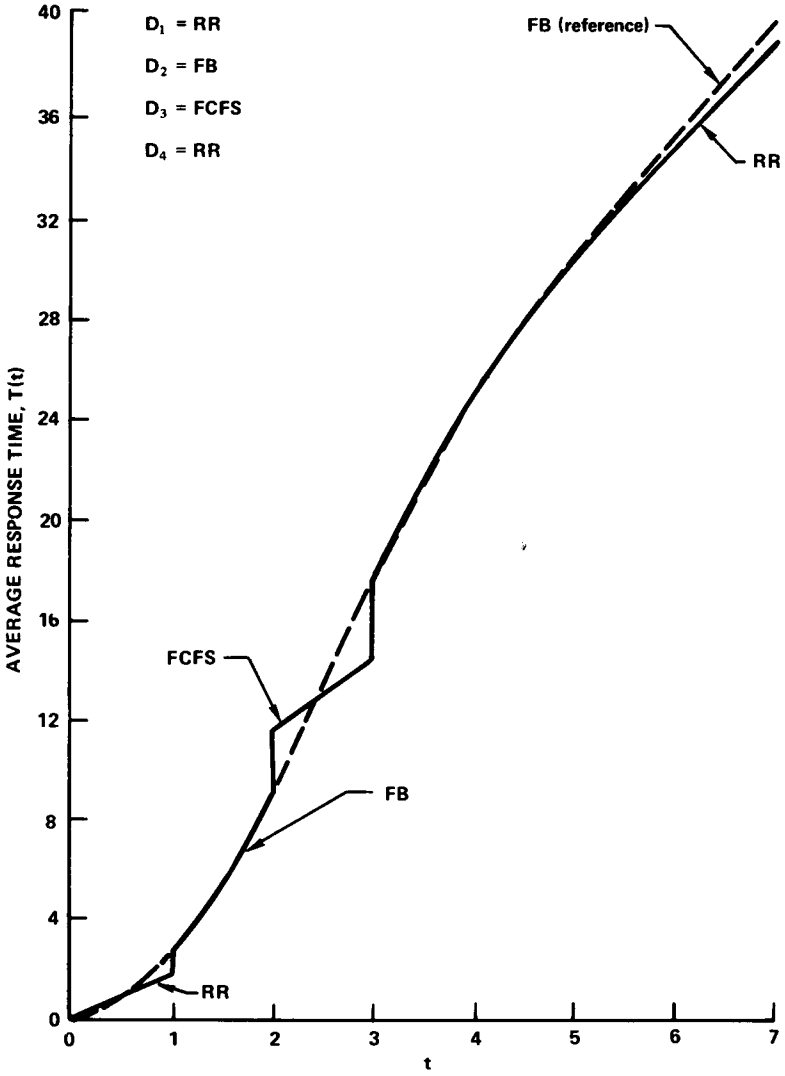

FIG. 6. Response time for an example of $N = 3$, M/M/1, $\mu = 1$, $\lambda = 0.75$, $a_i = i$.

havior for such systems can be found from Figure 7. It is interesting to note in the general M/G/1 case with D_i = FCFS that we have a solution for the FB system with *finite* quantum $q_i = a_i = a_{i-1}$ where preemption within a quantum is permitted!

Our fourth example is for an M/M/1 system with D_i = RR and is shown in Figure 8. Here we use the explicit form for $T(\tau)$ derived from eqs. (3.7), (3.30), and (3.33). We maintain the same value $\mu = 1$, $\lambda = 0.75$, $a_1 = 2$, $a_2 = 5$.

We leave M/M/1 now and give two examples for M/G/1. For the first example we choose the system $M/E_2/1$ with $N = 1$. In this system we have

$$\frac{dB(x)}{dx} = (2\mu)^2 x e^{-2\mu x}, \qquad x \geq 0 . \tag{4.1}$$

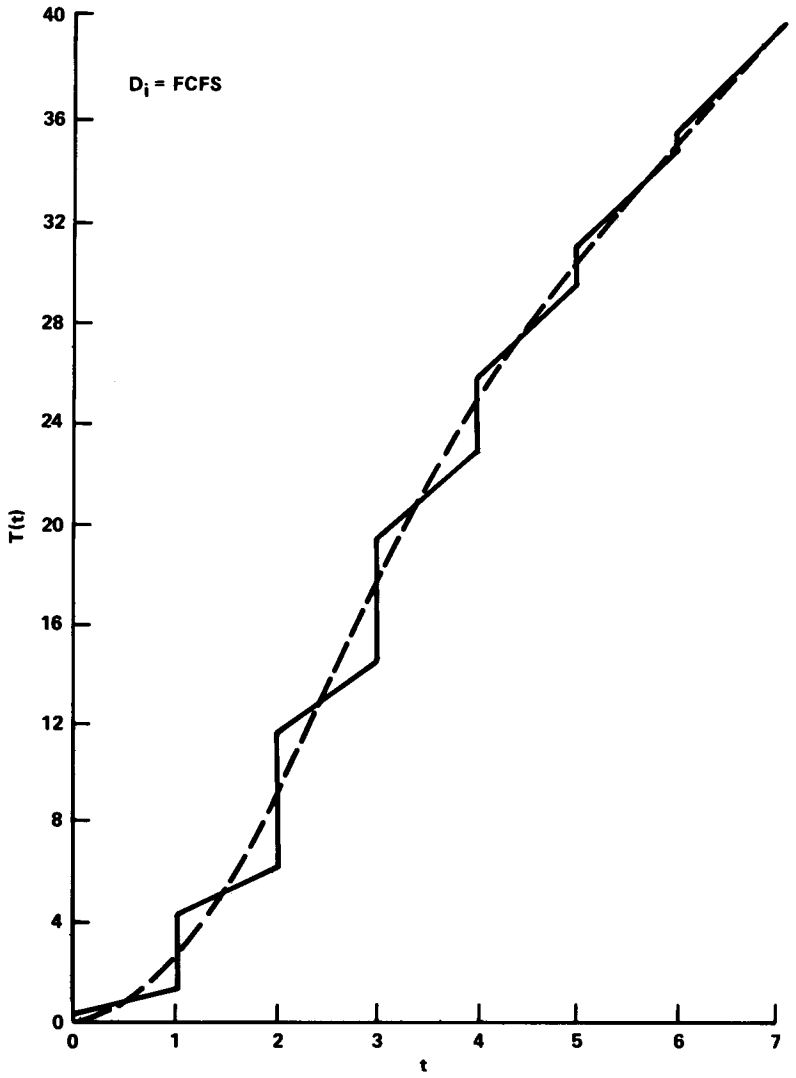

FIG. 7. Response time for the M/M/1 iterated structure, $\mu = 1$, $\lambda = 0.75$, $a_i = i$, $N = \infty$.

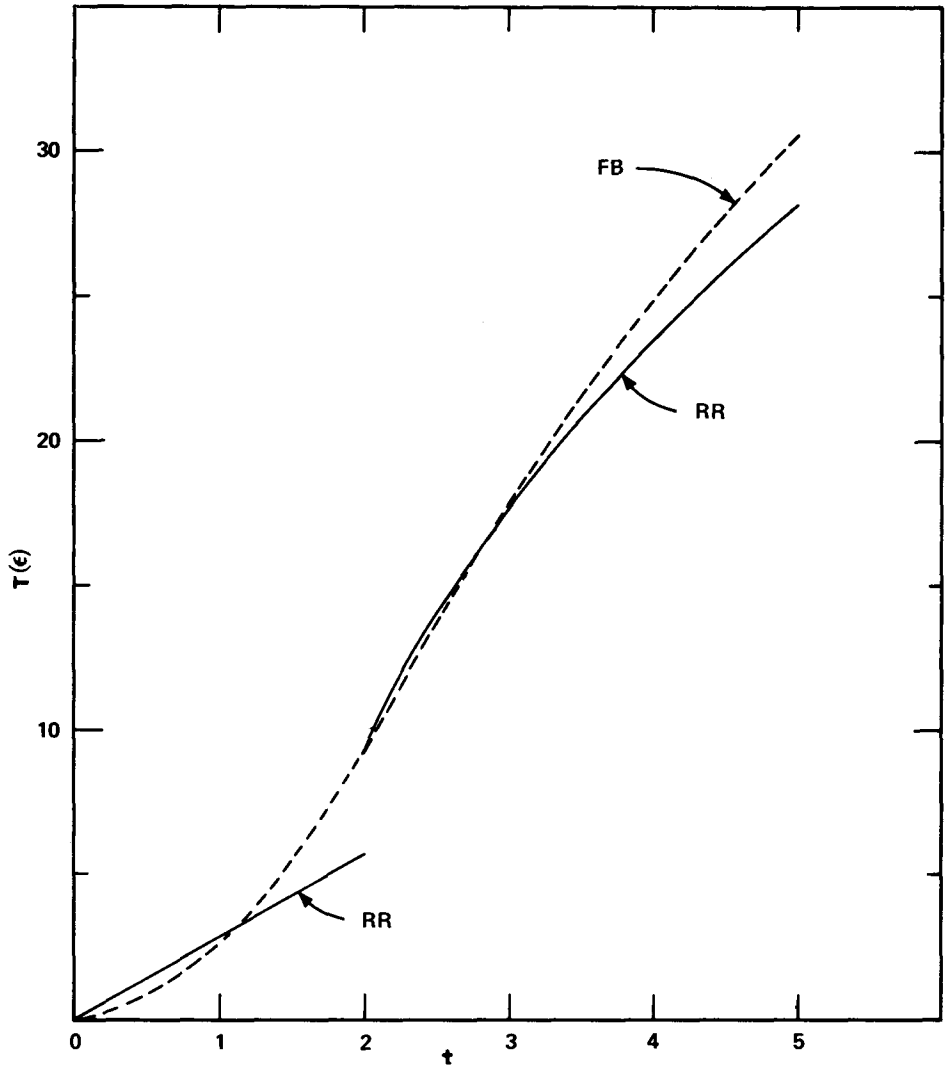

FIG. 8. Response time for example of D_i = RR, M/M/1, $\mu = 1.0$, $\lambda = 0.75$, $a_1 = 2.0$, $a_2 = 5.0$.

We note that the mean service time here is again given by $1/\mu$; the second moment of this distribution is $3/2\mu^2$. We calculate

$$\bar{t}_{<a_1} = \mu^{-1} - \mu^{-1}e^{-2\mu a_1}[1 + 2\mu a_i + 2(\mu a_1)^2]. \tag{4.2}$$

We choose the system $N = 1$ with D_1 = RR and D_2 = FCFS. For the cases $a_i = 1/2\mu$, $1/\mu$, $2/\mu$, $4/\mu$, and ∞ with $\mu = 1$ and $\lambda = 0.75$, we show in Figure 9 the behavior of this system.

The last example we use is for the following service time distribution:

$$b_1(x) \equiv \frac{dB_1(x)}{dx} = \begin{cases} 1, & 0 \leq x < \frac{1}{2}, \\ e^{-2(x-\frac{1}{2})}, & \frac{1}{2} \leq x, \end{cases} \tag{4.3}$$

as shown in Figure 10. In this case, $\bar{t}_{<\frac{1}{2}} = \frac{3}{8}$, $\bar{t}^2_{<\frac{1}{2}} = \frac{1}{6}$, $\bar{t} = \frac{5}{8}$, $\bar{t}^2 = \frac{2}{3}$. We choose the

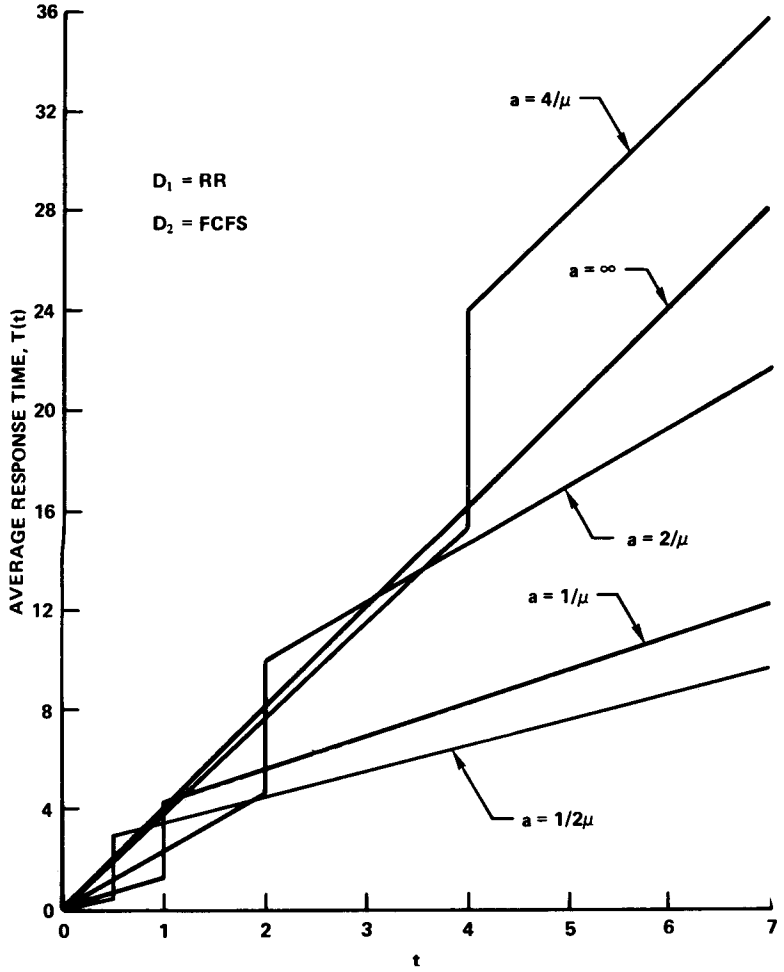

FIG. 9. Response time for RR, FCFS in $M/E_2/1$ with $\mu = 1$, $\lambda = 0.75$, and $a = \frac{1}{2}, 1, 2, 4, \infty$.

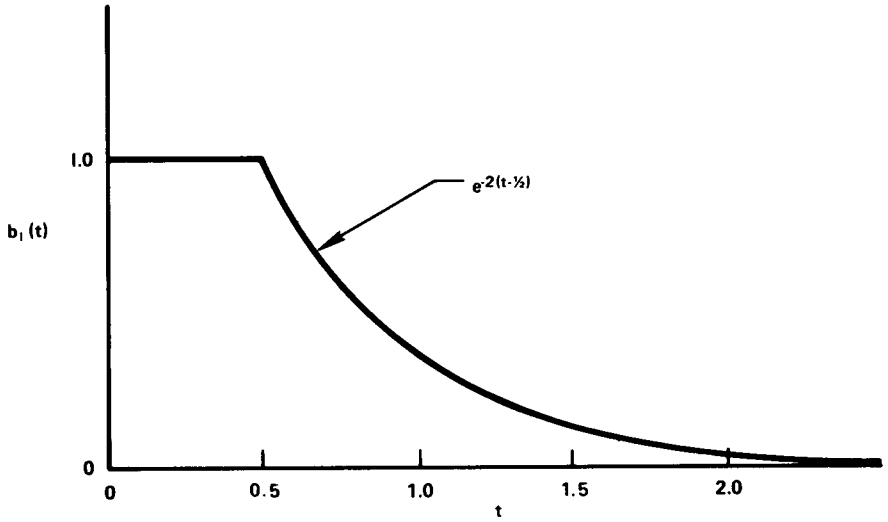

FIG. 10. Example service time density $b_1(t)$.

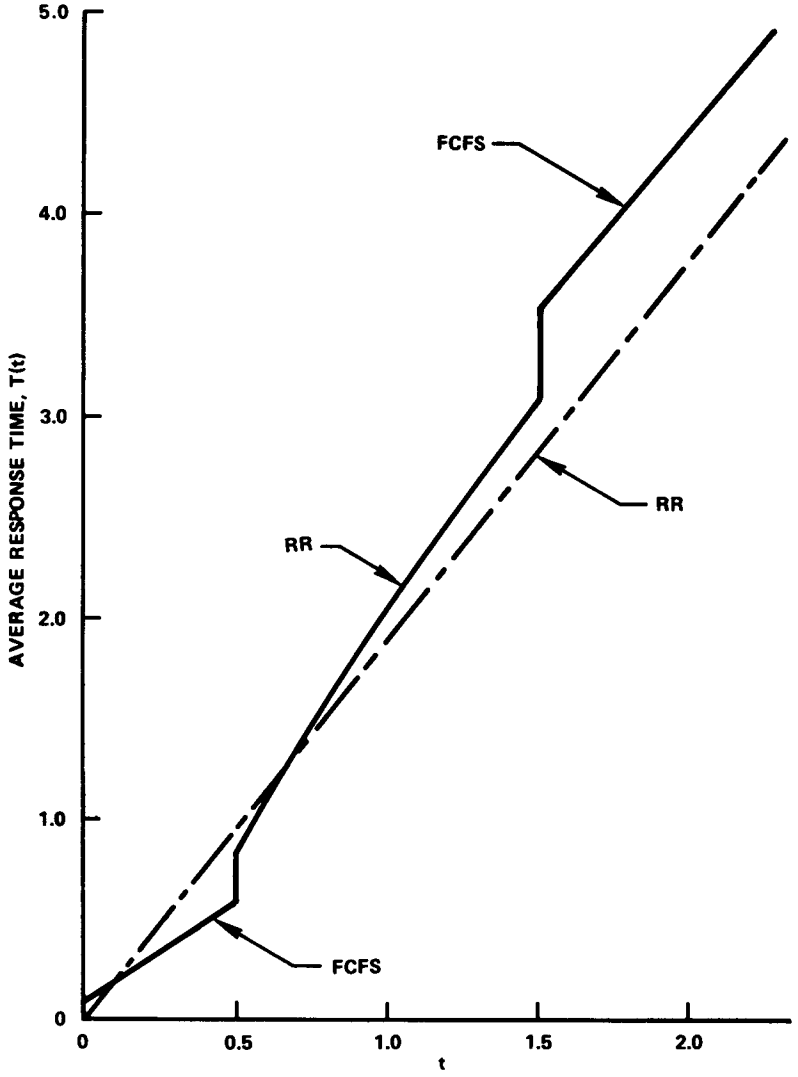

FIG. 11. Response time for an example of $N = 2$, $\lambda = 0.75$, service time density $= b_1(t)$.

system D_1 = FCFS, D_2 = RR, and D_3 = FCFS with $a_1 = \frac{1}{2}$, $a_2 = \frac{3}{2}$, and $\lambda = 0.75$. The performance of this system is given in Figure 11.

These examples demonstrate the broad behavior obtainable from our results as one varies the appropriate system parameters. In all cases the system discriminates in favor of the short jobs and against the longer jobs.

5. *Conclusion*

Our purpose has been to generalize results in the modeling and analysis of time shared systems. The class of systems considered was the processor sharing systems in which various disciplines were permitted at different levels of attained service. The principal results for M/G/1 are the following: (1) the performance for the FB

discipline at any level is given by eq. (3.1); (2) the performance for the FCFS discipline is linear with t within any level and is given by eq. (3.2); (3) the performance for the RR discipline at the first level is well known [8] and is given by eq. (3.8); (4) an integral equation for the average conditional response time for RR at any level (equivalent to bulk arrival RR) is given by eq. (3.12) and remains unsolved in general; however, for the service distribution given in eqs. (3.3′) and (3.4′), we have the general solution given in eq. (3.13) as derived in [11]. We further note that the average conditional response time at level i is independent of the queueing discipline at all other levels.

Examples are given which display the behavior of some of the possible system configurations. From these, we note the great flexibility available in the multilevel systems. From the examples in Section 4, we see that considerable variation from previously studied algorithms is possible so long as the number of levels is less than a small integer (say 5); however, we see that as N increases, the behavior of the ML systems rapidly approaches that of the pure FB system.

Examination of the envelope of the multitude of response functions available with the ML system has suggested that upper and lower bounds in system performance exist; this in fact has been established and is reported in [19].

REFERENCES

(*Note.* Reference [10] is not cited in the text.)

1. KLEINROCK, L. Analysis of a time-shared processor. *Naval Res. Logistics Quart. 2*, 1 (March 1964), 59–73.
2. MCKINNEY, J. M. A survey of analytical time-sharing models. *Comput. Surv. 1*, 2 (June 1969), 105–116.
3. TAKÁCS, L. A single-server queue with feedback. *Bell System Tech. J. 42* (March 1963), 505–519.
4. KLEINROCK, L. Time-shared systems: A theoretical treatment. *J. ACM 14*, 2 (Apr. 1967), 242–261.
5. COFFMAN, E. G., AND KLEINROCK, L. Feedback queueing models for time-shared systems. *J. ACM 15*, 4 (Oct. 1968), 549–576.
6. COFFMAN, E. G., JR., MUNTZ, R. R., AND TROTTER, H. Waiting time distributions for processor-sharing systems. *J. ACM 17*, 1 (Jan. 1970), 123–130.
7. KLEINROCK, L., AND COFFMAN, E. G. Distribution of attained service in time-shared systems. *J. Comput. Systems Sci. 3* (Oct. 1967), 287–298.
8. SAKATA, M., NOGUCHI, S., AND OIZUMI, J. Analysis of a processor-shared queueing model for time-sharing systems. Proc. 2nd Hawaii Internat. Conf. on System Sciences, Jan. 1969, pp. 625–628.
9. SCHRAGE, L. E. The queue M/G/1 with feedback to lower priority queues. *Manag. Sci. 13*, 7 (1967), 466–471.
10. CONWAY, R. W., MAXWELL, W. L., AND MILLER, L. W. *Theory of Scheduling*. Addison-Wesley, Reading, Mass., 1967.
11. KLEINROCK, L., MUNTZ, R. R., AND RODEMICH, E. The processor-sharing queueing model for time-shared systems with bulk arrivals. *Networks J. 1*, 1 (1971), 1–13.
12. COHEN, J. *The Single Server Queue*. Wiley, New York, 1969.
13. OLIVER, R. M., AND JEWELL, W. S. The distribution of spread. Research Report 20, Op. Res. Cen., U. of California, Berkeley, Calif., Jan. 25, 1962.
14. ADIRI, I., AND AVI-ITZHAK, B. Queueing models for time-sharing service systems. Technion, Mimeograph Ser. on Oper. Res., Statist. and Econ., Technion—Israel Inst. of Technol., Haifa, Israel, No. 27.

15. KLEINROCK, L. A conservation law for a wide class of queueing disciplines. *Naval Res. Logistics Quart. 12*, 2 (June 1965), 181–192.
16. KLEINROCK, L., AND MUNTZ, R. R. Multilevel processor-sharing queueing models for time-shared systems. Proc. Sixth Internat. Teletraffic Congress, Munich, Germany, Aug. 1970, pp. 341/1–341/8.
17. GAVER, D. Diffusion approximations and models for certain congestion problems. *J. Appl. Prob. 5* (1968), 607–623.
18. SCHRAGE, L. E. Some queueing models for a time-shared facility. Ph.D. dissertation, Dept. of Indust. Eng., Cornell U., Ithaca, N.Y., 1966.
19. KLEINROCK, L., MUNTZ, R. R., AND HSU, J. Tight bounds on the average response time for processor-sharing models of time-shared computer systems. Proc. IFIPS Congress, 1971 (to be published).

RECEIVED SEPTEMBER 1970; REVISED AUGUST 1971

A Stochastic Model for Message Assembly Buffering with a Comparison of Block Assignment Strategies

GARY D. SCHULTZ

IBM Research Division, Research Triangle Park, North Carolina

ABSTRACT. A stochastic model is developed for the process of dynamic buffering for inbound messages in a computer communications system. For a specific characterization of message traffic, complementary viewpoints—one considering the process as a queueing system, the other considering it as a compound process—lead to the same probability generating function. Two buffer assignment schemes, both dynamic but differing in binding strategy, are compared in terms of optimal buffer size and total buffer pool requirements for a given overflow criterion. A simple model of optimal blocking, earlier proposed by Gaver and Lewis, is extended to cover both buffer strategies and also heterogeneous message sources. Finally, the derived characterization of total storage requirements is compared by numerical example with conservative and nonconservative asymptotic treatments of the process.

KEY WORDS AND PHRASES: buffer assignment schemes, buffer block binding, computer communications model, dynamic buffer allocation, geometric binomial distribution, geometric Poisson distribution, message assembly buffering, optimal buffer size, Pólya-Aeppli distribution, shared buffer storage

CR CATEGORIES: 3.81, 4.39, 5.5, 6.29

Introduction

A design problem of interest arising in computer communication systems concerns the technique employed at the centralized computer facility for buffering inbound messages from a number of communication lines. Static assignment of private storage to each line for message assembly results in costly and inefficient usage of the memory resource by ignoring the stochastic behavior of both message generation and message length. Thus it is commonplace for system designers to capitalize on the orders of magnitude disparity between computer processing speed and line transmission rate and share buffer storage among all the lines by dynamically allocating buffers from a public pool during the message assembly process.

Such dynamic buffering techniques have the usual characteristic that the public pool is segmented into buffer blocks of equal size. As a message arrives at the computer, it is assembled by hardware and software intervention into (generally) noncontiguous blocks allocated piecemeal from the pool. This scatter assembly of a message imposes a storage penalty per block for chaining information to preserve the logical integrity of the data structure.

In this paper we establish means for determining optimal buffer block size and

Author's address: IBM Corp., P.O. Box 12275, Research Triangle Park, NC 27709.

total storage requirements for two buffer assignment schemes, both dynamic but differing in binding strategy. The differences in strategy roughly typify the varying performance constraints imposed by imbedding the line handling function within a high supervisory overhead, general purpose control program, or by placing it in a low overhead, dedicated subsystem characterized by a special purpose front end processor.

To facilitate comparison, we study the inbound message storage process *in vacuo*, formulating a stochastic model of storage usage with specific distributions imposed on message generation and message length. An interesting study of buffering by Gaver and Lewis [1] is cited, both because their approach somewhat parallels this one and because a simple blocking model they proposed lends itself to generalization and extended application here. They considered the low overhead buffering scheme in more generality with respect to distributional assumptions, and proposed a buffer storage performance criterion based on rate of exceeding buffer limits. Here we develop a model using more restrictive message traffic assumptions, but carry through the analysis of storage overflow without appeal to asymptotic approximations based on the central limit theorem.

Modeling Storage Usage

Consider a computer facility having M communication lines attached. To develop a model of storage usage for message assembly, we make the following assumptions about the input process:

(1) Each of the M lines feeding the computer has identical characteristics, with activity on one line being independent of activity on any other.

(2) Arrival of individual characters belonging to a single message proceeds at a constant rate (r characters per second) from start of message (SOM) through end of message (EOM).

(3) Each line experiences alternating periods of activity and inactivity. Considering the combined process as comprised of two alternating sequences of independent random variables, we describe both by exponential probability distribution functions with parameters μ (activity) and λ (inactivity). Thus, mean message transmission time is $1/\mu$, while the mean time between EOM and SOM is $1/\lambda$.

We are interested in comparing two buffer assignment schemes, described as follows.

Scheme 1. A buffer block is assigned instantaneously at the time of its immediate need, e.g. when the first character to enter the block begins to arrive. We call this scheme *incremental block binding* (IBB).

Scheme 2. One buffer block is preassigned to each line. During an active period, a new buffer block is allocated one "block time" prior to its (potential) need, e.g. when the first character to enter the most recently assigned buffer block begins to arrive. We call this scheme *anticipatory incremental block binding* (AIBB).

In general, the IBB scheme is possible where little overhead is incurred for performing block allocation, while the AIBB scheme characterizes higher overhead systems [2]. "Instantaneous" block assignment is, of course, a mathematical idealization, but it is realistic when considering the transmission rates of commonly used communication lines. Thus, in typical real systems, "high" overhead may entail only a significant fraction of a single character time.

Two approaches which give complementary insights into the process are suggested. One approach considers the process as a queueing system, while the other conceives it as a compound process. In either case we are interested in the process in equilibrium; that is, we want the probability distribution for the number of storage blocks currently assigned at an arbitrary time, t, to be sufficiently removed from a "zero time" so that any initial condition effects (e.g., all lines idle) have "worn off."

To fix the problem more concretely, Figure 1 shows a possible realization of the process. The message *time* realization of Figure 1(a) is contrasted with the equivalent storage *allocation* realization of 1(b), with both the IBB and AIBB schemes shown. Notice that the figure indicates that the storage blocks allocated during message assembly are freed in exact synchronism with the occurrence of EOM. This simplification is consonant with our intent to isolate the effects on storage usage due to buffering strategy from any confounding effects of post-assembly central processing unit (CPU) delays and processing. It is equivalent to assuming that the CPU pulls out the message immediately on completion of assembly, thereby making the allocated blocks again available for assignment. In certain cases, a model of

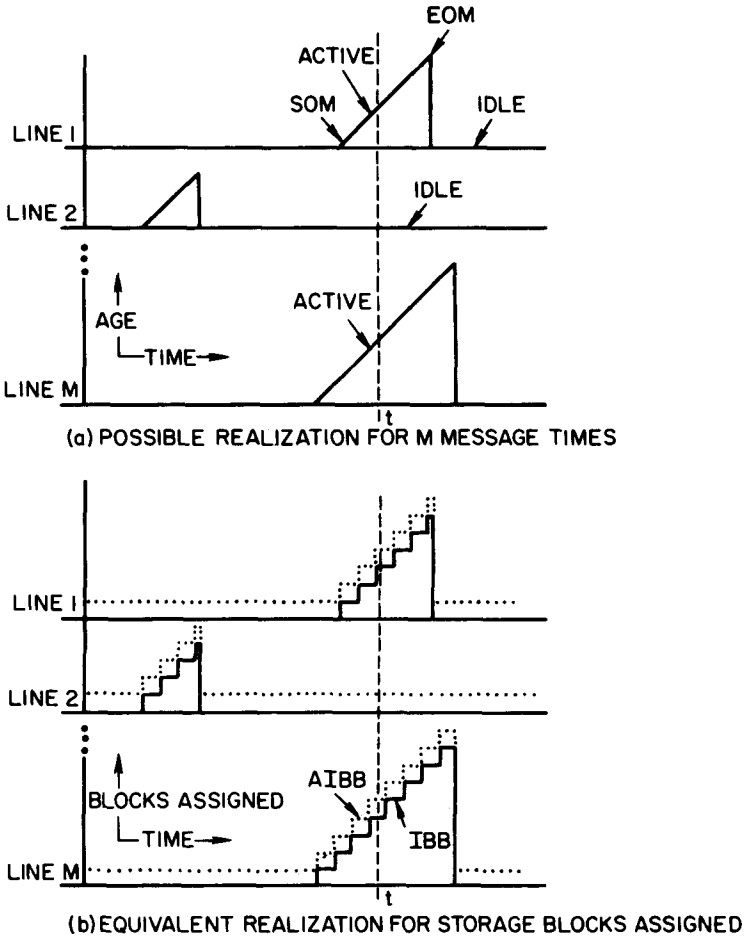

FIG. 1. Message age versus storage allocation

storage usage that goes beyond the message assembly considerations modeled here may also justifiably neglect post-assembly effects. For example, CPU processing may be negligible compared to message transmission time—a frequent case in practice. Alternatively, post-assembly processing may take place in work space independent of the buffer pool following high priority CPU transfer of the message out of the buffer blocks at EOM time.

From Figure 1(b) it is also clear that the AIBB scheme simply adds M blocks to that required for IBB. Thus, as far as storage usage is concerned, we can develop a model for both by treating the IBB scheme and extending it to AIBB in a simple fashion. This approach is used in the following derivation.

It is convenient to conceive of the entire process as consisting of M independent $M/M/1$ queueing systems (Poisson arrivals, exponential service, single server) where "service time" corresponds to the interval between SOM and EOM (message length). By defining each to be a pure loss system with no waiting room at the server, we can choose the same parameters (λ, μ) as given by our earlier assumptions to get a mathematically equivalent characterization. Since the distribution of the time between two consecutive Poisson points (SOMs) is the same as the distribution of the time between an arbitrary point (e.g. EOM) and the next Poisson point (SOM), the lossy aspect upholds the earlier assumption (3) and allows us to ignore the unreal possibility of multiple messages queued on a single line. By the independence assumption, a one-line model extends easily to the M-line process.

For an $M/M/1$ queueing system with a maximum of one in the system (i.e. the one in service), it is known [3] that in equilibrium:

$$\begin{aligned} \Pr\{\text{system is idle}\} &= \Pr\{\text{zero customers in the system}\} \\ &= 1/(1+\rho) \end{aligned} \tag{1}$$

$$\Pr\{\text{system is busy}\} = \rho/(1+\rho) \tag{2}$$

where $\rho = \lambda/\mu$.

We can use these results first to derive the equilibrium probability for the "age" or elapsed service time of the message (if one) active at an arbitrary time t, and then to obtain the probabilities for the number of storage blocks currently assigned to the line (using assumption (2)). Defining the random variables $M_a(t)$ as the message age at time t, and $M_s(t)$ as the (discrete) number of buffer blocks assigned at time t, we get

$$\Pr\{M_a(t) = 0\} = \Pr\{\text{system idle}\} = 1/(1+\rho) = \Pr\{M_s(t) = 0\}. \tag{3}$$

The presence of a message of age $v > 0$ at time t means that at time $t - v$ the line changed from idle to active by the arrival of SOM, and the resulting message has length at least as great as v. This is more precisely stated in differential notation as:

$$\begin{aligned} \Pr\{v < M_a(t) < v + dv\} &= \Pr\{\text{line idle at time } t - v - dv\} \\ &\quad \cdot \Pr\{\text{SOM occurs in } (t - v - dv,\ t - v)\} \\ &\quad \cdot \Pr\{\text{message length} > v\} \\ &= 1/(1+\rho) \cdot \lambda\, dv \cdot e^{-\mu v} \end{aligned} \tag{4}$$

where the parameters are as defined in assumption (3). To get the probability for the number of blocks assigned, we first define buffer block size $B = b + c$, where c is the number of storage spaces needed for chaining and b is the number of spaces

available to receive message characters. Then, using assumption (2) of constant character rate, r, and defining $1/a = b/r$, we use eq. (4) to get for $y = 1, 2, \cdots$,

$$\begin{aligned}\Pr\{M_s(t) = y\} &= \Pr\left\{\frac{y-1}{a} < M_a(t) < \frac{y}{a}\right\} \\ &= \frac{\lambda}{1+\rho}\int_{(y-1)/a}^{y/a} e^{-\mu v}\,dv \\ &= \frac{\rho}{1+\rho}(1 - e^{-\mu/a})e^{-(y-1)\mu/a}. \end{aligned} \tag{5}$$

The integration in (5) is a consequence of the fact that exactly y discrete blocks are assigned from the instant that $y - 1$ have completely arrived through the instant that the yth block is completely filled. Notice that for $y = 1, 2, \cdots$,

$$\Pr\{M_s(t) = y \mid \text{system busy}\} = (1 - e^{-\mu/a})e^{-(y-1)\mu/a} \tag{6}$$

and is zero otherwise. This is the geometric distribution with parameter $e^{-\mu/a}$.

Thus, using eqs. (3) and (5), the probability generating function, $P(s)$, for the single-line process is

$$\begin{aligned} P(s) &= \frac{1}{1+\rho} + \sum_{y=1}^{\infty}\frac{\rho}{1+\rho}(1-p)p^{y-1}s^y \qquad (p = e^{-\mu/a}) \\ &= (1 - ps + \rho s - \rho ps)/(1+\rho)(1-ps) \end{aligned} \tag{7}$$

and the probability generating function, $H(s)$, for the M-line process becomes

$$H(s) = [P(s)]^M = \left[\frac{1 - ps + \rho s - \rho ps}{(1+\rho)(1-ps)}\right]^M. \tag{8}$$

The queueing theory approach has the advantage that the variables of buffer size, message length, and line speed enter naturally into the treatment. Insights gained from a complementary viewpoint, however, permit more natural derivation of the desired probability distribution and also provide the basis for discussion, in a later section of the paper, of asymptotic processes.

Viewing the total process in equilibrium, then, as a compound process [4], we can relax the assumption on message (SOM) generation, and simply take the event of finding a line active as the outcome of a Bernoulli trial, with activity having probability Q. In this case, we are interested in the sum $S_N(t)$ of a sequence $\{X_k(t)\}$ of mutually independent random variables with the common distribution $\Pr\{X_k(t) = i\} = f_j$ and generating function $F(s) = \Sigma f_j s^j$. The number $N(t)$ of terms in the sum $S_N(t)$ is also a random variable independent of the $X_k(t)$ with distribution $\Pr\{N(t) = n\} = g_n$ and generating function $G(s) = \Sigma g_n s^n$. For the distribution $h_i\}$ of $S_N(t)$ we get

$$h_i = \Pr\{S_N(t) = i\} = \sum_{n=0}^{M}\Pr\{N(t) = n\}\Pr\{X_1(t) + \cdots + X_n(t) = i\}. \tag{9}$$

In terms of our process in equilibrium, we let $N(t)$ correspond to the number of active lines, and $X_k(t)$ correspond to the number of buffer blocks currently assigned to the kth active line. To preserve the discrete nature of storage allocation, we ascribe to $X_k(t)$ the geometric distribution with parameter p. Then

$$f_j = p^{j-1}(1-p), \qquad j = 1, 2, \cdots, \tag{10}$$

$$g_n = \binom{M}{n} Q^n(1 - Q)^{M-n}, \qquad n = 0, 1, \cdots, M. \tag{11}$$

For a fixed n, the distribution of $X_1(t) + \cdots + X_n(t)$ is given in Feller's notation [4] by $\{f_j\}^{n*}$, the n-fold convolution of $\{f_j\}$ with itself, giving

$$\{h_i\} = \sum_{n=0}^{M} g_n \{f_i\}^{n*}. \tag{12}$$

With

$$F(s) = \sum_{j=1}^{\infty} p^{j-1}(1 - p)s^j = \frac{s(1 - p)}{1 - ps} \tag{13}$$

and

$$G(s) = \sum_{n=0}^{M} \binom{M}{n} Q^n(1 - Q)^{M-n}s^n = (1 - Q + Qs)^M, \tag{14}$$

we find the generating function of the sum $S_N(t)$ to be

$$\begin{aligned} H(s) &= \sum_{i=0}^{\infty} h_i s^i = \sum_{n=0}^{M} g_n[F(s)]^n \\ &= G(F(s)) = \left[1 - Q + Q\frac{s(1 - p)}{1 - ps}\right]^M \\ &= \left[\frac{1 - ps - Q + Qs}{1 - ps}\right]^M, \end{aligned} \tag{15}$$

which, with $p = e^{-\mu/a}$ and $Q = \rho/(1 + \rho)$, agrees with (8).

To get the distribution, we make use of eq. (9) to derive for the zero term:

$$h_0 = \Pr\{N(t) = 0\} = (1 - Q)^M. \tag{16}$$

Using (13) we find

$$\begin{aligned} [F(s)]^n &= (1 - p)^n s^n (1 - ps)^{-n} \\ &= (1 - p)^n s^n \sum_{k=0}^{\infty} \binom{-n}{k} (-ps)^k \\ &= (1 - p)^n s^n \sum_{k=0}^{\infty} \binom{n + k - 1}{k} (ps)^k. \end{aligned} \tag{17}$$

Thus

$$\begin{aligned} \text{coefficient of } \quad s^i &= (1 - p)^n \binom{i - 1}{i - n} p^{i-n}, \qquad i \geq n \\ &= \left(\frac{1 - p}{p}\right)^n \binom{i - 1}{n - 1} p^i \end{aligned} \tag{18}$$

and is zero for $i < n$. Substituting this result, along with that of eq. (11), into eq. (12) we get for $i \geq 1$,

$$h_i = (1 - Q)^M p^i \sum_{n=1}^{\min(i,M)} \binom{M}{n} \binom{i - 1}{n - 1} \left[\frac{Q(1 - p)}{p(1 - Q)}\right]^n. \tag{19}$$

Conforming with conventions [5] for other compound processes, we call the distribution defined by (16) and (19) the *geometric binomial* distribution. Knowing $H(s)$

(or, see [4], using $G(s)$ and $F(s)$) allows us to derive the mean and second and third central moments for $S_N(t)$. Using conventional notation, where μ'_r is the rth moment (about zero) and μ_r is the rth central moment, these are:

$$\mu_1' = MQ/(1 - p), \tag{20}$$

$$\mu_2 = MQ(1 + p - Q)/(1 - p)^2, \tag{21}$$

$$\mu_3 = MQ[(2Q - 1 - p)(Q - 1 - p) + 2p]/(1 - p)^3. \tag{22}$$

Suppose, then, that the concern is to choose a finite buffer pool size such that for given parameters some performance criterion is to be met. An obvious approach is to consider the finite storage process to be well characterized by the unconstrained process modeled above. In this case, choosing *probability of overflow* as a criterion, for the idealized process we simply find a storage level whose probability of being exceeded matches the overflow criterion. The shortcoming in this matching is inherent in the departure of the real, finite storage process from the unconstrained mathematical idealization at the storage boundary. In actual systems, all blocks assigned to an overflowing message are immediately released at the time of overflow, causing congestion to fall below the critical point. An additional complication is the twofold perturbation in the input process at the boundary due to effective idleness of the source of the violating message until EOM and the subsequent retransmission event. These effects, as well as second order performance criteria based on refined analysis of the unconstrained process as treated (asymptotically) in [1], are ignored here since they are not crucial to the comparison of buffering schemes. For the purposes of the present study, probability of overflow remains an apt first order performance criterion.

Applying such a criterion, eqs. (16) and (19) can be used to find a level S, say, such that $\Pr\{S_N(t) > S\} < \epsilon$ or, equivalently, $\Pr\{S_N(t) \leq S\} \geq 1 - \epsilon$, where ϵ is the performance constraint. This yields the number of blocks required in the finite pool to satisfy requirements. For the IBB buffering scheme, the pool consists of $S \cdot B$ characters while, for AIBB, $(S + M) \cdot B$ characters comprise the pool.

If buffer pool blocking did not incur a chaining penalty, and if system overhead for dynamic block allocation were not at issue, it is obivous that a one-character block size should be chosen to minimize total storage requirements. The fact that block size, $B = B + c$, involves a chaining field which is not available for reception of message characters introduces an apparent tradeoff. With too small a block size, a large portion of the pool is devoted to space for chaining fields; while with too large a block size, the earlier binding of available storage lowers the efficiency of storage usage. The latter effect is particularly acute for the AIBB scheme, where an available buffer dangles unused an entire "block time" (b character times) prior to its actual use for incoming characters. Both schemes suffer from an approximate additional half block time early binding per allocated character (on the average) while the block is being filled.

Finding a block size that minimizes overall storage requirements is relatively simple. Holding all other parameters fixed, eqs. (16) and (19) are used with b varying. For each b, we can find the smallest n_b such that $S(n_b) = \sum_{i=0}^{n_b} h_i \geq 1 - \epsilon$, where ϵ is the overflow constraint. Multiplying n_b by B, or $(n_b + M)$ by B, gives the total buffer pool size in characters for the IBB and AIBB schemes,

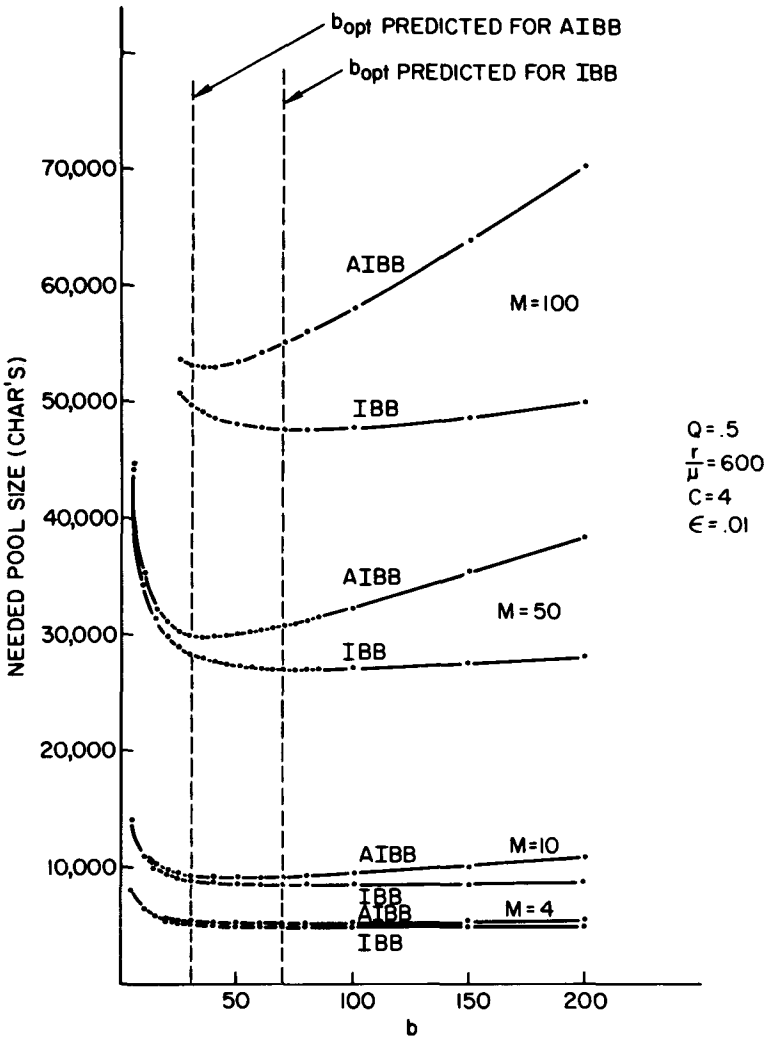

FIG. 2. Comparisons of required pool sizes and optimal block sizes for the IBB and AIBB binding strategies

respectively. Such a procedure can be used to find the block size that minimizes overall pool size for a given constraint.

This method has been used to produce the curves shown in Figure 2. The interesting feature is the difference in sensitivity to block size of the two buffering schemes. For both schemes it is unwise to choose too small a block size; however, the AIBB method is also rather sensitive in the other direction, while the IBB curves experience only mild rise beyond the minimum. This effect, as well as the differences in optimal block size, is readily explained by the differences in binding strategies of the two approaches.

Modeling Storage Blocking

A method of finding "optimal" block size that avoids the computational disadvantages of the technique described above is given by Gaver and Lewis [1]. We

generalize their model here to encompass the AIBB scheme as well as the IBB method, and we also extend it by temporarily relaxing the assumption that all lines must be identical.

Begin by considering an individual line i. Suppose on this line the average message length is l_i, while the average line loading, or fraction of time the line is active sending characters, is Q_i. While the line is inactive, there are β storage spaces wasted for buffering, where $\beta = 0$ (IBB) or $\beta = b + c$ (AIBB). While the line is active, the number of storage spaces wasted (assigned, but not holding characters) will vary according to the current age, or elapsed service time, of an observed inbound message. The relationship between average message length, l_i, and the average observed age, denoted m_i, of an inbound message (both in character times) is apparent using results (not dependent on the exponential distribution assumption) from [6] for "backward recurrence time," e.g. $m_i = \frac{1}{2}(l_i + \sigma_i^2/l_i)$, where σ_i^2 is the variance of message length for line i. In the specific case of the exponential distribution, where the mean and standard deviation of message length are equal, $m_i = l_i$.

During activity, then, the expected number of wasted spaces is equal to the scheme-dependent wastage, defined by β above, plus the expected wasted spaces in the blocks previously filled and the block currently receiving characters. Including the previously filled blocks as well as the block currently receiving characters, there are an expected $c \cdot \lceil m_i/b \rceil$ used for chaining, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x (the *ceiling* of x). For m_i large with respect to b, this is approximately cm_i/b. By the same assumption, there are also on average approximately $b/2$ storage spaces vacant in the buffer block currently being filled. Hence the expected wasted space for line i is

$$L_i(b) = Q_i\left(\frac{b}{2} + \frac{cm_i}{b} + \beta\right) + (1 - Q_i)\beta. \tag{23}$$

Considering all lines, the expected total wasted space is

$$L(b) = \sum_{i=1}^{M} L_i(b). \tag{24}$$

Differentiation with respect to b reveals that L is minimized at

$$B_{\text{opt}} = \begin{cases} c + [2c\Sigma m_i Q_i/\Sigma Q_i]^{\frac{1}{2}} & \text{(IBB)} \quad (25) \\ c + [2c\Sigma m_i Q_i/(\Sigma Q_i + 2\text{M})]^{\frac{1}{2}} & \text{(AIBB)} \quad (26) \end{cases}$$

where $B_{\text{opt}} = c + b_{\text{opt}}$.

For homogeneous lines, i.e. $m_i = m$ (because $l_i = l$ and $\sigma_i = \sigma$) and $Q_i = Q$ $(i = 1, 2, \cdots, M)$, we get

$$B_{\text{opt}} = \begin{cases} c + [2cm]^{\frac{1}{2}} & \text{(IBB)} \quad (27) \\ c + [2cmQ/(Q + 2)]^{\frac{1}{2}} & \text{(AIBB).} \quad (28) \end{cases}$$

The result in (27) agrees with that given by Gaver and Lewis in [1] where a further refinement of their model, focussing on loss in the last utilized block and using the distribution function of message length, is posed to show the quality of the approximation leading to (27). The utility of the above model is illustrated in Figure 2, where the optimal block sizes predicted by (27) and (28) are indicated.

It is interesting to compare the two buffering schemes with respect to CPU overhead for chaining operations. Let C^i denote the expected chaining overhead (number of chaining interrupts) per message for scheme i where the superscript $i = 1$

(IBB) or 2 (AIBB). For each scheme we assume that the pool is optimally blocked, with b^1_{opt} and b^2_{opt} denoting the optimal values found for IBB and AIBB, respectively. Then for l, m, c, and Q defined as above:

$$\frac{C^2}{C^1} = \frac{[l/b^2_{\text{opt}}] + 1}{l/b^1_{\text{opt}}} = \frac{l/[2cmQ/(Q+2)]^{\frac{1}{2}} + 1}{l/[2cm]^{\frac{1}{2}}} = \left(\frac{Q+2}{Q}\right)^{\frac{1}{2}} + \left(\frac{2cm}{l^2}\right)^{\frac{1}{2}}. \tag{29}$$

In the case of the exponential distribution, where $l = m$,

$$\frac{C^2}{C^1} \sim \left(\frac{Q+2}{Q}\right)^{\frac{1}{2}} \qquad \text{for} \quad l \gg c. \tag{30}$$

Asymptotic Processes

Computational considerations, as evidenced by the complexity of eq. (19), commonly lead modelers to seek approximation by a more tractable asymptotic process. It is interesting to contrast the results obtainable from the exact forms derived earlier for the modeled process with results using asymptotic characterizations. Two such approaches, one conservative in approximating storage needs and one nonconservative, are as follows.

A nonconservative approximation results from the assumption that the M-line process can be characterized as asymptotically Gaussian in distribution. From our earlier characterization of the M-line process as a compound process, with total storage allocated being the random sum of mutually independent random variables, we note that, for M sufficiently large, a central limit theorem effect applies [7]. Thus we can simplify computation by choosing a normal distribution, with mean and variance the same as for the actual distribution, to characterize the storage process. For a given overflow constraint, ϵ, this approach approximates the total buffer pool requirement, T, by (in storage blocks)

$$T = \mu_1' + \delta_{1-\epsilon}\,\mu_2^{\frac{1}{2}} \tag{31}$$

where μ_1' and μ_2 are given by eqs. (20) and (21), and $\delta_{1-\epsilon}$ is the deviation found from a standardized normal distribution table such that $1 - \epsilon$ of the distribution lies to its left.

A conservative approximation, on the other hand, results from the assumption that the number of active lines $N(t)$ with distribution given by eq. (11) can be asymptotically characterized by the Poisson distribution. Thus, with $\alpha = M \cdot Q$,

$$\Pr\{N(t) = n\} \sim \frac{\alpha^n}{n!}\, e^{-\alpha}. \tag{32}$$

This leads to characterizing the M-line process by the well-studied geometric Poisson or Pólya-Aeppli distribution [5],[1] where eqs. (16) and (19) are replaced by

$$h_0 = e^{-\alpha}, \tag{33}$$

$$h_i = e^{-\alpha} p^i \sum_{n=1}^{i} \binom{i-1}{n-1} \frac{1}{n!} \left[\frac{\alpha(1-p)}{p}\right]^n, \qquad i \geq 1, \tag{34}$$

[1] The first edition of reference [5] contains incorrect versions of eqs. (36)–(38).

TABLE I

Q	M	T_N/T_{GB}	T_{PA}/T_{GB}
.1	4	.61	1.03
.1	10	.71	1.04
.1	50	.87	1.03
.1	100	.91	1.02
.5	4	.81	1.15
.5	10	.88	1.12
.5	50	.96	1.07
.5	100	.98	1.05

Q = line loading; M = number of lines; $T_{(.)}$ = computed total storage required using the normal (N), geometric binomial (GB) or Pólya-Aeppli (PA) distribution.

while equivalent forms for (20)–(22) are

$$\mu_1' = \alpha/(1 - p), \tag{35}$$

$$\mu_2 = \alpha(1 + p)/(1 - p)^2, \tag{36}$$

$$\mu_3 = \alpha(p^2 + 4p + 1)/(1 - p)^3. \tag{37}$$

The existence of a recurrence relation due to Evans [8],

$$ih_i - [2(i - 1)p + \alpha(1 - p)]h_{i-1} + (i - 2)p^2h_{i-2} = 0 \qquad (i \geq 2), \tag{38}$$

permits ease of computation.

Table I gives a flavor for the accuracy of the two approximations compared with exact results using the geometric binomial distribution. The figures were computed choosing average message length as 600, block size $B = 79$ (with $c = 4$), $\epsilon = .01$, and assuming the IBB scheme. The results in the table show that for small Q the Pólya-Aeppli distribution gives a close approximation even for a small number of lines. Using the conventional $\mu_3/\mu_2^{3/2}$ index of skewness, the general behavior of the two approximations is easily predicted.

In general, the normal approximation must be used with caution because it nonconservatively estimates the "tail" of a distribution positively skewed like the geometric binomial. The Pólya-Aeppli distribution, on the other hand, being even more positively skewed, will yield correspondingly pessimistic results for the tail. It is interesting to note in passing that Feller [7] characterizes storage facilities in general as reducing to compound Poisson processes.

Conclusions

Using a simple model of the message assembly process, a comparison has been made between two buffering schemes. The results of the analysis indicate that for IBB the system designer has reasonable latitude in terms of block size, needing only to guard against too small a choice. For AIBB, the analysis indicates there is a greater sensitivity to larger choices of block size, as well as greater CPU overhead for chaining operations than that for IBB when for each scheme block size is optimized to reduce total storage requirements.

A scheme intermediate in strategy with respect to IBB and AIBB would have obvious appeal for the overhead-encumbered systems. We propose the following.

Scheme 3. To an inactive line assign a smaller "root" block of size d, say. Upon receipt of SOM, allocate a buffer block of (normal) size B from the public pool. Allocate each successive block from the pool when all but d characters of the currently used block have been filled. We refer to this scheme as *rooted incremental block binding* (RIBB).

The choice of the size d would depend only on the tolerance demanded by worst-case overhead considerations, which by the earlier remarks regarding character arrival times common to communication lines, implies $d \ll B$.

Analytical comparison of RIBB with IBB and AIBB is a simple matter and reveals RIBB to have the basic advantages of IBB. Notice that RIBB requires $M \cdot d$ more storage than IBB; hence storage usage curves for RIBB are those of IBB (cf. Figure 2) displaced upward everywhere, for $b \geq d$, by the amount $M \cdot d$. Similarly, in eq. (23), $\beta = d$ for RIBB, giving an optimal block size identical to that for IBB (because d is a constant). Thus, for optimal blocking, the CPU chaining overhead per message for RIBB is about the same as for IBB. An important feature of RIBB is that, like IBB, it is only mildly sensitive to larger choices of block size. This permits the system designer to make tradeoffs in favor of lower overall system overhead for message assembly without experiencing the heavier storage penalty of the AIBB method.

While the assumption on message length distribution in the paper is somewhat restrictive, it is not uncommon for modelers to employ such an assumption. Moreover, recent results reported by Fuchs and Jackson [9] show that the geometric distribution (the discrete distribution analog of the exponential distribution) can be reasonably fitted to observed statistics for short average message length systems. Similar measurements for longer average message length systems, characterized by CPU-to-CPU communication or "remote buffered" terminal usage, are not yet available in the literature.

It is hoped that the closed form result obtained herein and the perspective on asymptotic processes will be of use for other models as well as for real world applications where the assumptions used are descriptive.

ACKNOWLEDGMENTS. Thanks and acknowledgment are warmly accorded to Professor V. L. Wallace of the University of North Carolina, for whose class an earlier version of this work was a course paper. Besides originally suggesting the fruitful approach of treating each line as an $M/M/1$ queue, Dr. Wallace gave encouragement and guidance on matters of notation, style, and perspective, and pointed out some errors in the description.

The interest and discussion provided by Dr. J. Spragins of IBM and J. Whitlock Jr., of the University of North Carolina are also gratefully acknowledged.

REFERENCES

1. GAVER, D. P., JR., AND LEWIS, P. A. W. Probability models for buffer storage allocation problems. *J. ACM 18*, 2 (Apr. 1971), 186–198.
2. IBM CORP. *IBM System/360 operating system: Basic telecommunications access method.* IBM Form GC30-2004.

3. SAATY, T. L. *Elements of Queueing Theory with Applications.* McGraw-Hill, New York, 1961.
4. FELLER, W. *An Introduction to Probability Theory and Its Applications, Vol. I.* Wiley, New York, 1957.
5. JOHNSON, N. L., AND KOTZ, S. *Distributions in Statistics: Discrete Distributions.* Houghton Mifflin, Boston, 1969.
6. COX, D. R. *Renewal Theory.* Methuen, London, 1962.
7. FELLER, W. *An Introduction to Probability Theory and Its Applications, Vol. II.* Wiley, New York, 1966.
8. EVANS, D. A. Experimental evidence concerning contagious distributions in ecology. *Biometrika 40* (1953), 186–211.
9. FUCHS, E., AND JACKSON, P. E. Estimates of distributions of random variables for certain computer communications traffic models. *Comm. ACM 13*, 12 (Dec. 1970), 752–757.

RECEIVED APRIL 1971; REVISED SEPTEMBER 1971

An Approach for Finding C-Linear Complete Inference Systems

JAMES R. SLAGLE

National Institutes of Health, Department of Health, Education and Welfare, Bethesda, Maryland

ABSTRACT. An inference system is C-linear complete if it is linear (ancestry filter) complete with top clause C, where C is in the original set of clauses and has suitable satisfiability properties. C-linear completeness is important for two reasons: (1) set-of-support refutation completeness is a corollary of C-linear refutation completeness, and previous computer experiments have indicated that the set-of-support strategy is efficient; (2) the search for a C-linear deduction can be naturally represented by a goal tree, and good techniques are known for searching such trees. A theorem is proved which provides a fairly general approach which, when given only a ground complete inference system, often yields a (nonground) C-linear complete system. This approach can be combined with a previously presented approach whose object is to replace some of the axioms of a given theory by a refutation complete system. The object of the combined approach is to replace some of the axioms by a C-linear refutation complete system.

The approach of this paper is applied to six combinations of four inference rules. The rules are ordinary resolution, paramodulation, and two rules which respectively replace the transitivity axiom for $\subseteq$ and the set membership axiom. C-linear refutation complete systems are found from the six combinations. For the case of resolution alone, the stronger C-linear deduction (consequence-finding) completeness is obtained.

KEY WORDS AND PHRASES: theorem-proving, completeness theorems, inference systems, linear deduction, linear refutation, inference rules, resolution principle, paramodulation, transitivity axiom, set membership axiom, artificial intelligence, deduction, refutation, mathematical logic, predicate calculus.

CR CATEGORIES: 3.60, 3.64, 3.66, 5.21

1. *Introduction*

The purposes of programming a computer to prove theorems concern artificial intelligence [9], deduction, mathematics, and mathematical logic. See [10] for a discussion. The resolution principle [8] is an inference rule used in automated theorem-proving. Wos et al. [15, 16], Allen and Luckham [1], and others have written proof-finding programs embodying the resolution principle. Although quite general, these programs have been so slow that they have proved only a few theorems of any interest.

As a step in coping with this problem, a previous paper [10] presented a fairly general approach which, when given the axioms of some special theory, often yields complete, valid, efficient (in time) rules corresponding to some of the given axioms. The present paper takes another step. A theorem is proved which provides a fairly

Author's address: Heuristics Laboratory, Division of Computer Research and Technology, National Institutes of Health, Bethesda, MD 20014.

general approach which, when given only a ground complete inference system, often yields a (nonground) C-linear complete system. The object of the approach obtained by combining these two approaches is to replace some of the axioms by a C-linear refutation complete system. In Sections 6 through 8 the approach developed in Sections 3 through 5 is applied to six combinations of four inference rules. Previously, only two of these systems (resolution [5, 6] and resolution with paramodulation [3]) had been proved C-linear complete.

C-linear completeness is important for two reasons. First, the search for a C-linear deduction can be naturally represented by a goal tree, and good techniques are known for searching such trees [9]. Second, set-of-support refutation completeness is a corollary of C-linear refutation completeness, and previous computer experiments [16] have indicated that the set-of-support strategy is efficient. We mean set-of-support refutation completeness in the sense of Wos et al. [16] and not in the stronger sense of Slagle [11]. The difference is discussed in [18]. Table I is a key to symbols used in this paper.

2. *Clauses, Deductions, and Refutations*

We start with a vocabulary of individual variables, function symbols, and predicate symbols. Terms, atoms, and literals are introduced next. (See [8, 9] for a full de-

TABLE I. KEY TO SYMBOLS

Symbol	*Meaning(s)*	*Symbol*	*Meaning(s)*	*Symbol*	*Meaning(s)*
a	atom, (inference) rule	A	clause	μ	most general unifier
b	branch, constant, rule	B	clause	θ	substitution
c	constant, rule	C	clause	τ	substitution
d	deduction	D	clause	$\subseteq$	contained in or equal (for sets), less than or equal (for ordering)
e	expression	E	clause, there exists		
f	function	F	function which maps a countable set of clauses into a countable set of clauses, literal form, clause form, functionally reflexive axiom	$\in$	is a member of
g	function			$\notin$	is not a member of
h	index for integer			$\cup$	union
i	index for integer			$\sim$	not
j	index for integer			iff	if and only if
k	literal			$\vdash$	yields
m	nonnegative integer	G	function which maps a countable set of clauses into a countable set of clauses		
n	nonnegative integer				
p	positive integer				
q	integer				
r	(inference) rule				
s	term	H	functionally reflexive axiom		
t	term				
u	term	J	inference system		
v	term	P	predicate symbol		
w	individual variable	Q	countable set of clauses		
x	individual variable				
y	individual variable	R	set of rules		
z	individual variable	S	countable set of clauses		
		T	countable set of clauses		
		W	list		

scription.) A *clause* is a finite disjunction of zero or more literals. When convenient, we regard a clause as the set of its literals. In particular, the order and multiplicity of the literals in a disjunction is irrelevant. To facilitate matters, we take similar liberties with the nomenclature later, but what is meant will always be clear from the context. The (individual) variables in a clause are considered to be universally quantified. We shall regard a set of clauses as synonymous with a conjunction of all those clauses. By a substitution, we mean a substitution of (not necessarily ground) terms for variables. A clause C_1 *subsumes* a clause C_2 if there is a substitution θ such that $C_1\theta \subseteq C_2$. An *mgu* (most general unifier) μ for a set of expressions is a substitution with the property that for any two members e_1 and e_2 of the set, $e_1\mu = e_2\mu$, and there is no more general substitution with this property. When C is obtained by applying the inference rule r to $C_1, \cdots, C_p$, we shall say that we have the *inference* $C_1, \cdots, C_p \vdash C$ by r. We assume that every premise $C_1, \cdots, C_p$ is relevant in the inference. The symbol $\vdash$ may be read "yields." A deduction may be viewed as consisting of zero or more inferences.

Definitions. If C is a clause with literals k_1 and k_2 which have an mgu μ, then $(C-k_1)\mu$ is an *immediate factor* of C. The *factors* of C are given by the following: C is a factor of C, and an immediate factor of a factor of C is a factor of C. *Factoring* is the rule by which C yields a factor. It is assumed that factoring is a member of every set of rules used in this paper.

Definitions. $(E_1, \cdots, E_n)$ is a *list*. It is called an *ordered n-tuple* when we want to emphasize the n.

Definition. $(E_1, \cdots, E_n)*(E_{n+1}, \cdots, E_{n+m}) = (E_1, \cdots, E_{n+m})$.

Definitions. Let S be a set of clauses, and let R be a set of inference rules. A *deduction by* R from S of a clause D is a list $(C_1, \cdots, C_p, C_{p+1}, \cdots, C_{p+n})$ of clauses such that

(1) $p > 0$,
(2) $n \geq 0$,
(3) $\{C_1, \cdots, C_p\} \subseteq S$,
(4) for every $j = 1, \cdots, n$, there is a set S_j of clauses which precede C_{p+j} in the list and there is a rule r_j in R such that $S_j \vdash C_{p+j}$ by r_j,
(5) the last clause in the list is D.

Such a deduction may be thought of as a tree with D as a root (at the bottom). If C is at a node at the top of the tree and if b is a branch from that node to the root where D is, then b is a *CD-branch*. The *length of a branch* is the number of arcs (one less than the number of nodes) in it. A *refutation* is a deduction of the empty (contradictory) clause, $\square$.

Definitions. The *depth in a deduction d of a clause C* is defined recursively as follows:

(1) The depth in d of C justified by being in the original set S is zero.
(2) The depth in d of C justified by $C_1, \cdots, C_p \vdash C$ is one more than the maximum of the depths of $C_1, \cdots, C_p$.

The *depth of a deduction* is the depth of its last clause.

3. *Inference Systems, Linear Deductions, and Linear Liftability*

In this and the next section we present three interesting theorems which are used to prove the theorem (Theorem 5 of Section 5) which is the basis of our approach.

Definition. An *inference system* is an ordered pair (R, F) where R is a set of (inference) rules, and F is a function which maps a countable set of clauses into a countable set of clauses.

We shall sometimes regard R as the inference system (R, F) where F maps each countable set into the empty set. For an example of an inference system, let R be resolution with paramodulation and, for all S, let F be the $\{=\}$-reflexive axioms for S, that is, $\{x=x\}$ and the functionally reflexive axioms for S [10].

Definition. Let the inference system $J = (R, F)$. A *deduction (refutation) by* J from S is a deduction (refutation) by R from $S \cup F(S)$.

An instance of a *literal form* is a literal. We shall not bother to define literal form formally, but shall content ourselves with a few examples. Let a be a variable standing for an atom, and let s and t be variables standing for terms. The literal form $\sim a$ stands for any negative literal. An example of an instance of $\sim a$ is $\sim P(x, c, f(x))$. An instance of the literal form $s \subseteq t$ is $g(x) \subseteq f(c)$. An instance of the literal form $k[t_1, \cdots, t_n]$ is a literal in which $t_1, \cdots, t_n$ occur in particular distinct locations. Similarly, we shall speak of *clause forms*. We assume that no variable in one clause occurs in another. We are now in a position to present a very simple and uniform way to state ordinary resolution, paramodulation, and many other inference rules.

Definition. A *g.u.r.* (ground unit rule) $F_1, \cdots, F_p \vdash F_{p+1}$ (where every variable occurring in the clause form F_{p+1} occurs in at least one of the literal forms $F_1, \cdots, F_p$) is a rule r such that $C_1, \cdots, C_p \vdash C_{p+1}$ by r iff $(C_1, \cdots, C_p, C_{p+1})$ is an instance of $(F_1, \cdots, F_p, F_{p+1})$ (respectively), where C_{p+1} is a ground clause and where $C_1, \cdots, C_p$ are ground unit clauses. Note that we have taken the liberty of letting a unit clause be an instance of a literal form. The context will make it clear whether the "yields" symbol is part of an inference or part of a g.u.r. An example of a g.u.r. is $s \notin u, t \subseteq u \vdash s \notin t$.

Definitions. The *one-literal rule* corresponding to the g.u.r. $F_1, \cdots, F_p \vdash F_{p+1}$ is the following rule. From the clauses $k_1 \vee D_1, \cdots, k_p \vee D_p$, where there are most general unifiers $\mu_1, \cdots, \mu_p, \mu$ such that $(k_1\mu_1, \cdots, k_p\mu_p) = (F_1, \cdots, F_p)\mu$, infer $D_1\mu_1 \cup \cdots \cup D_p\mu_p \cup F_{p+1}\mu$. For $i = 1, \cdots, p$, k_i is the *inference literal* in $k_i \vee D_i$.

In this paper, the use of $\vee$ in $k_i \vee D_i$ implies that k_i is not in D_i. If k_i might have been in D_i, we would have used $\cup$. A similar convention is used throughout this paper.

Paramodulation [3, 10, 17, 18] is the one-literal rule corresponding to the g.u.r. $k[s], s=t \vdash k[t]$. Semantic resolution [11, 12] and, in particular, hyperresolution [7] are not one-literal rules, since more than one literal in the nucleus clause may be directly involved in an inference. From now on, when we speak of a rule (except for factoring), we shall mean a one-literal rule corresponding to some g.u.r. We shall use a g.u.r. and its one-literal rule interchangeably.

Next we give two equivalent definitions of a linear deduction. The second one uses a figure.

Definition. A *C_0C_n-linear deduction by* R from S is a deduction $W_0*(C_0)*\cdots*W_n*(C_n)$ by R from S where

(1) every member of W_0 is in S,
(2) C_0 is in S,
(3) for $j = 1, \cdots, n$, W_j is a list of factors of clauses which precede C_{j-1}, and,

if the jth inference not counting factoring is $S_j \vdash C_j$ by r_j, every member of W_j is in S_j.

Definitions. A *C_0C_n-linear deduction by R* from S is a deduction d by R from S where d has the form shown in Figure 1 and where for each $j = 0, \cdots, n-1$ and for each $i = 1, \cdots, m_j$ (where $m_j \geq 0$), C_{ji} is a factor of either some clause in S or some C_h where $h \leq j$. The branch b consisting of $C_0, \cdots, C_n$ is the *main branch* of d. The clauses $C_{j1}, \cdots, C_{jm_j}$ are the *side clauses* to C_j. If, for all j and i, the justification of C_{ji} is that it is a factor of some clause in S, the C_0C_n-linear deduction by R from S is a *C_0C_n-input deduction by R* from S. In the following, when we say that there is a C_0C_n-linear (input) deduction, we are implicitly saying, if they are not already given, that there is a C_0 or C_n or both.

Definition. A rule r is *linearly liftable* by J if, for any ground inference $D_1, \cdots, D_p \vdash D$ by r and for any clauses $C_1, \cdots, C_p$ which have $D_1, \cdots, D_p$ as their respective ground instances and for all $i = 1, \cdots, p$, there is a clause C and a C_iC-linear deduction by J from $\{C_1, \cdots, C_p\}$ such that D is a ground instance of C.

In what follows, S is a countable set of clauses. Although seldom remarked upon, the statements and sometimes the proofs of completeness theorems for automated theorem-proving can generally be extended from finite to countable sets [13]. The proofs often fail for uncountable sets.

Definition. A function G that maps countable sets of clauses into countable sets of clauses is *additive* if it maps unit sets into finite sets and, for all S, we have $G(S) = \bigcup_{C \in S} G(\{C\})$.

Definition. F is *closed* for a set R^* of inference rules if F is additive and, for every set S of clauses and for every clause D deducible by (R^*, F) from S, we have that $F(\{D\}) \subseteq F(S)$.

Roughly speaking, Theorem 1 states that, if F is closed, local linear liftability implies global linear liftability for linear deductions. In this paper we shall often pass from an individual to a set without making a formal definition. Thus, when in Theorem 1 we say that R^* is linearly liftable by J, we mean that every r in R^* is linearly liftable by J.

THEOREM 1. *If S' is a set of ground instances of clauses in S, if R^* is linearly liftable by $J = (R^*, F)$, if F is closed for R^*, and if there is a $C'D'$-linear deduction by R^* from S', then there is a CD-linear deduction by J from S where C' and D' are ground instances of C and D respectively.*

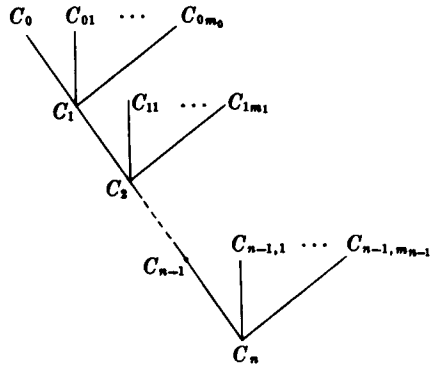

FIG. 1. The form of a C_0C_n-linear deduction

PROOF. We shall give a procedure which transforms (lifts) the given ground leduction into the required deduction. Each clause E' whose justification is that t is in S' is first transformed into E in S where E' is a ground instance of E. Next, he procedure works from top to bottom along the main branch of the given linear leduction. For $j = 0, \cdots, n-1$, let the justification of C'_{j+1} be the (ground) nference $C_j', C'_{j1}, \cdots, C'_{jm} \vdash C'_{j+1}$ by R^*. Since the procedure works from the op down, the set $T_j = \{C_j, C_{j1}, \cdots, C_{jm}\}$ of the corresponding clauses is known. ince F is closed for R^*, we have that $F(T_j) \subseteq F(S)$. Since R^* is linearly liftable y J, there is a C_jC_{j+1}-linear deduction by J from T_j where C'_{j+1} is a ground in- tance of C_{j+1}. When the procedure reaches the bottom of the given deduction, he required linear deduction has been obtained. This completes the proof.

. *Input Deductions and Rule Permutability*

'he following theorem could be extended to include semantic resolution (including yperresolution) (suitably restricted) in R, but this would bring us too far away rom our main purpose. When no confusion should result, we shall often drop races. Thus in the following theorem, $(S-C_0) \cup C_0'$ means $(S-\{C_0\}) \cup \{C_0'\}$. 'he theorem does not "lift," as can be seen from the following example. In the nput deduction $P(f(b), f(f(b)))$, $b=c \vdash P(f(b), f(f(c)))$ by paramodulation, $'(x, f(x))$ subsumes $P(f(b), f(f(b)))$, but the corresponding input deduction ould be longer. The extra inference would be a paramodulation from the func- ionally reflexive axiom, $f(y)=f(y)$.

THEOREM 2. *If there is a C_0C_n-input deduction by R with main branch b from a et S of ground clauses, then for any subclause C_0' of C_0 there is a $C_0'C_n'$-input de- uction by R with main branch b' from $S' = (S-C_0) \cup C_0'$ where C_n' subsumes $'_n$ and where b' is no longer than b.*

PROOF. The proof is by induction on the length n of b. The case when n is zero s trivial. Assume the theorem true for $n = 0, \cdots, q$. Let b have length $q+1$, and et the deduction consist of a C_0C_q-input deduction of length q from S followed by he inference $C_q, C_{q1}, \cdots, C_{qm} \vdash C_{q+1}$. By induction there is a $C_0'C_q'$-input eduction from S' where the length of the main branch does not exceed q and here C_q' subsumes C_q. There are three cases. If C_q' does not contain the inference iteral in C_q, then C_q' subsumes C_{q+1} and we are done. If at least one of the C_{qi} is '0 and, when replaced by C_0', loses its inference literal, then there is a $C_0'C_0'$-input eduction from S' where the main branch has length zero and where C_0' subsumes '$_{q+1}$. In the remaining case, none of the inference literals have been lost. Hence, 'e may infer a clause C'_{q+1} which subsumes C_{q+1}. The length of the main branch a the resulting $C_0'C'_{q+1}$-input deduction from S' does not exceed $q+1$.

Definitions. *r_1/r_2 is (unit) permutable with respect to R* if, for every deduction onsisting of the inference $C_1, \cdots, C_p \vdash C_{p+q}$ by r_1 over (followed by) the in- erence $C_{p+1}, \cdots, C_{p+q} \vdash C$ by r_2 where $C_1, \cdots, C_{p+q-1}$ are ground (unit) clauses, here is a $C_{p+1}C'$-input deduction by R from $\{C_1, \cdots, C_{p+q-1}\}$ where C' subsumes '. We shall write u.p. for unit permutable.

There is no need to prove a nonground version of the following theorem, since t is used in the proof of Lemma A, whose proof also uses (ground) Theorem 2.

THEOREM 3. *If r_1 and r_2 are in R and if r_1/r_2 is u.p. with respect to R, then r_1/r_2 s permutable with respect to R.*

PROOF. (The proof is similar to parts of those for Lemmas 1 through 4 in