AT&T Technical Journal
Vol. 64, No. 2, February 1985
Printed in U.S.A.

Approximate Analysis of a Generalized Clocked Schedule

By A. A. FREDERICKS, B. L. FARRELL, and D. F. DEMAIO*

(Manuscript received December 8, 1983)

Clocked schedules represent an important method of task scheduling for computer systems with real-time applications. In this paper we consider a generalized class of clocked schedule that includes those used in many stored program control switching systems. Key performance measures for this class are discussed, and an analytic approximation method for analyzing certain of these measures is given. This approximation method is most applicable in evaluating long-term delays. (A companion paper by Doshi addresses short-term delays for systems with extremely time-critical tasks.) Comparisons are made with exact numerical results (obtained using the method presented in a companion paper by Ackroyd), detailed simulation models, and field data.

I. INTRODUCTION

Processor scheduling concerns specifying when each task that must be done in a computer system is to be scheduled for execution, and how conflicts in task execution are to be resolved—e.g., by setting task priorities. In this paper we consider the performance analysis of a class of schedules for computer systems with real-time applications, that is, applications where at least some tasks have time-critical execution requirements. Collection of digits in a call-processing system is one of the more important examples of a time-critical task.

To introduce the concept of a clocked schedule, we consider the simple example system of Fig. 1. The processor must respond in a

* Authors are employees of AT&T Bell Laboratories.

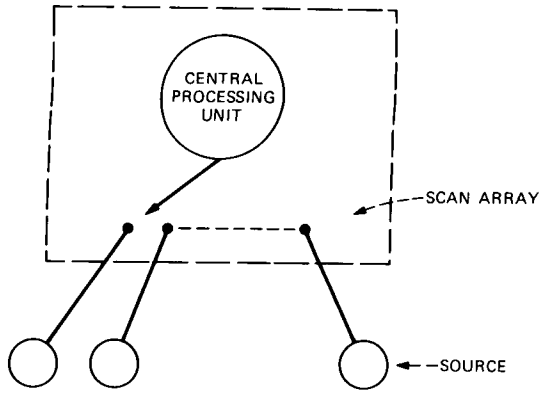


Fig. 1—Example system.

timely manner to requests from sources to send data. This consists of recognizing new requests, doing setup work (e.g., assigning buffers, sending a "ready to receive" signal), and collecting the data sent. For concreteness, consider this to be a microprocessor system handling incoming dial pulse trunks. Collecting the data corresponds to counting the number of pulses sent (number dialed). We can distinguish at least two distinct tasks: task 1, collecting pulses on active trunks; task 2, scanning inactive trunks for new requests and performing needed setup work. Task 1 is clearly the most time critical. Figure 2 shows a clocked schedule for this system. Time is divided into intervals (slots) of length T (dictated by the on-off lengths of the pulses), and in each slot, first task 1 is executed (for all active trunks), then task 2. If, at the beginning of a time slot, task 2 is executing, it is interrupted and task 1 execution is given control.

In applications of this type, the large number of stimuli (e.g., state

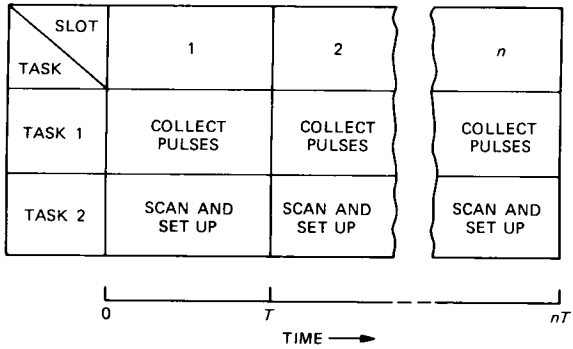


Fig. 2—Clocked schedule for example system.

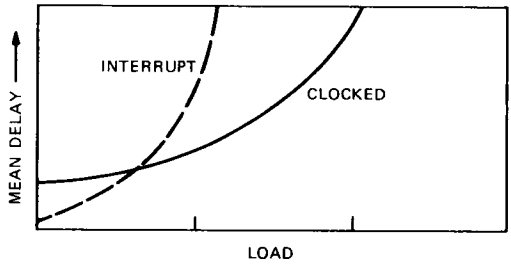


Fig. 3—Comparison of clocked and interrupt-driven schedule.

changes on active trunks) that need processing per unit time usually precludes the use of an interrupt-driven schedule having one interrupt per stimulus. Figure 3 gives a qualitative description of the relative trade-offs between clocked and interrupt-driven schedules. For a more detailed, quantitative treatment of this trade-off, see, for example, Ref. 1. For a discussion of performance trade-offs of other scheduling alternatives, see Ref. 2, where this example system is used to provide a tutorial on the analysis and design of processor schedules for computer systems with real-time applications.

Considerable work on analyzing clocked schedules similar to the basic one described above, including applications to distributed microprocessor-based systems, has been done by P. Kuehn and others (e.g., see Refs. 3, 4, and 5). Here we consider a generalized clocked schedule that forms the basis for the processor scheduling in many real-time systems, including microprocessor-based systems as well as large single- and multiprocessor call-processing systems. In Section II we define the class of schedules to be studied, discuss relevant performance measures and work-load characterizations, and use concrete examples to illustrate some scheduling variants to this class of schedules. In Section III we develop approximations for certain performance measures introduced in Section II. The approximation method is based on that given in Ref. 6. In Section IV we illustrate the accuracy of the approximation by using comparisons with exact calculations (using the methods described in Ref. 7), simulation, and field data. Section V contains some concluding remarks.

II. A GENERAL CLASS OF CLOCKED SCHEDULES

The simple clocked schedule noted above can be generalized in a variety of ways. We will consider some of the most important generalizations from an applications point of view and also note some additional variants of practical importance.

2.1 Generalized clocked schedule

Figure 4 shows the general class of clocked schedules that we will consider. Time is divided into intervals (slots) of length T. The first (labeling) column lists, in order of descending priority, all potential tasks for scheduling. A 1 in the ith row and jth column indicates that the ith listed task is scheduled for execution in the jth slot. We assume that every task is scheduled periodically and that the total period of the schedule is N_{period}. Three classes of tasks are indicated on Figure (4): High Priority (HP), Low Priority (LP), and Fill (F). The distinction between these will become clear in the following discussion.

The execution of the schedule proceeds as follows. We begin at time $t = 0$ at the beginning of the first slot. The tasks scheduled are executed in the order that they appear. Assuming all HP and LP tasks are completed before the end of the time slot ($t < T$), F work (assumed scheduled in every slot) is begun. If an LP task is still executing at the end of the time slot, it is interrupted, and it and all lower-priority LP tasks are added to the work list of the next slot *at their same priorities.* The HP tasks are *not* interrupted at the end of the time slot, that is, all HP tasks scheduled are completed before new work is scheduled. We will thus often refer to HP tasks as noninterruptable tasks and LP tasks as interruptable tasks. (F work is also interruptable.)

Thus, in summary, we have the following execution pattern. When

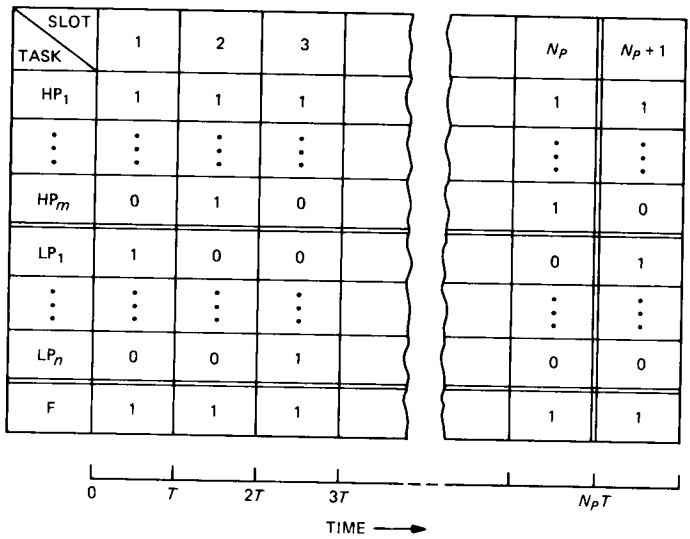


Fig. 4—General clocked schedule.

the HP list of tasks for the jth slot is begun (possibly delayed due to past scheduled HP work), the interrupt timer is inhibited. At the completion of all HP tasks, the timer is enabled. If the current time slot has already expired, the interrupt timer will immediately fire (scheduling new HP tasks); otherwise the LP work is begun. If an interrupt occurs prior to completion of all LP tasks, the remaining work is added to the next slot (the remaining rows in the jth column are "or'ed" to the $(j + 1)$st column). If all LP work completes prior to the interrupt, F work is executed until the interrupt occurs.

2.2 Performance measures

For a given task, scheduled every τ time units, we distinguish three categories of performance measures: short-term delays—delays on the order T or less; medium-term delays—delays greater than T but less than τ; and long-term delays—delays greater than τ. While this categorization is somewhat arbitrary, it has been useful in practice.

HP tasks are generally reserved for executing the most time-critical functions (e.g., digit reception in our example), and hence short-term delays are the most important performance measures. Indeed, short-term delays generally are needed to determine an appropriate time-slot length, T. A particularly important performance measure in this category is the probability that HP work scheduled for a given time slot is still executing at the end of that slot. This measure is generally referred to as the probability of (slot) overrun. Medium- and long-term delays also are often important for HP tasks; the latter is often needed to set appropriate levels for sanity timers. (In most systems, timers are set to monitor the time between successive executions of each task. If this time exceeds a given threshold [set to minimize the probability of false alarms due to work-load variability], certain maintenance routines are invoked.)

Long-term and sometimes medium-term delays are usually the most important performance measures for LP tasks, which generally are less time critical. For example, receiver attachment is typically scheduled as an LP task, possible every 100 ms, but with delays up to 500 ms acceptable.

Typical F tasks are line scanning for new originations or maintenance. For such tasks one defines a quantity T_f (e.g., time to scan all lines once and time to complete a set of maintenance tasks) such that the relevant performance measures are based on delays in completing T_f time units of F work. (Sometimes F work consists of executing a new schedule, in which case distributional information for delays can be an input to another performance analysis.) The delays of interest

for F work are generally long term, much greater than T, and often greater than $N_{\text{period}}\ T$, the entire schedule period.

2.3 Work-load characterization

Two main aspects of work-load characterization must be considered: the job-processing times associated with task execution, which generally depends on the number of jobs (stimuli) to be processed; and the characteristics of the arrival of jobs (stimuli). If we denote by X_i the (random) amount of time required to process task i, then often X_i is adequately characterized by $a_i + b_i J_i$, where a_i is the overhead associated with entering task i, J_i is the (random) number of jobs found (in our example, the number of trunks that had state changes to be processed), and b_i is the time needed to process each job found. To discuss the characterization of the arrival process, we use the queueing model description of a clocked schedule given in Fig. 5. There are two queues for each HP or LP task, an internal queue and an external queue. There is also a "never empty" single queue for F work. The arrival rate of jobs for the ith task is denoted by λ_i—often (we note some important exceptions shortly) the assumption of independent Poisson streams for these external arrivals is adequate. These external arrivals enter the external holding queue and are only allowed to pass to the internal queues, where they can be processed, when the indicated gates are opened. We distinguish four main arrival characterizations based on the gate-closing mechanism.

Gating 1: Here, at the start of each new time-slot execution, all gates corresponding to tasks scheduled for execution in this time slot are instantaneously opened, moving all existing jobs from the holding queues to the internal queues. The gates are then immediately closed, barring further movement of new jobs into the internal queues.

Gating 2: At the start of execution of a new slot, the gate for the highest-priority scheduled task is opened instantaneously and then

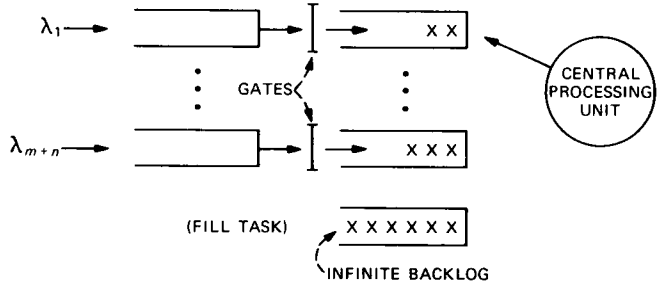


Fig. 5—Queueing model of clocked schedule.

closed. As each new scheduled task is ready for execution, its gate is correspondingly opened instantaneously and then closed.

Gating 3: Same as gating 2, but once opened, each gate is left opened until there are no more jobs for this task; the gates are then closed and the gate for the next task is opened.

Gating 4: All gates are always open.

Although in any given case, one of these gating mechanisms can often be used to characterize the arrivals, in actual systems, quite often a mixture of all are present, and indeed, there are further complications, e.g., the λ_i are not independent. (Even in our example, clearly a large number of state changes in a given slot implies a high probability of a large number of state changes in a slot that is displaced in time by a dial pulse length.) We will note some examples as we discuss some variants of the class of clocked schedules we have defined for study.

2.4 Some variants and examples

We briefly note some examples of systems that use clocked schedules as a means of demonstrating some of the complexities of clocked schedules for real systems and how they often deviate from the "ideal" models noted above. However, practical experience has indicated that these somewhat idealized models can often provide valuable insight into system performance.

2.5 Microprocessor peripheral interface system

An example of a rather basic clocked schedule is that used in the Microprocessor Peripheral Interface System (MPIS), designed to provide an intelligent interface between certain switching systems and T-carrier facilities. The main purpose of MPIS is to execute the tasks associated with the setup and tearing down of interoffice calls via a T-carrier facility. This includes digit reception/transmission/timing and interoffice signaling. For this system, the schedule for each slot is identical. After some overhead associated with each new time slot, the first task executed is to poll each of the T-carrier digroups and to process signaling change reports from or orders to them. With this polling mechanism, some work arriving after the task is initiated will be processed, while some will not. Hence the arrivals to this task behave like a combination of gating 2 and gating 3. The next task consists of unloading a First-In First-Out (FIFO) queue with orders from the switching machine and, unless the order is a high-priority maintenance order, sorting and scheduling the orders for processing by other tasks later in the slot. Gating 1 is a reasonable approximation for characterizing the arrival pattern to these latter tasks. However,

there is clearly a dependence between the work associated with these tasks and the work done in the FIFO task.

2.6 Support processor

Large 1 *ESS*™ switching equipment offices have a Support Processor (SP), which performs most of the needed input/output and timing functions. Almost all of the tasks are executed as HP, LP, or F tasks consistent with the model of Fig. 4. Typical HP tasks include digit reception and transmission, LP tasks include timing functions, and both trunk and line scanning for new originations are done as F work. However, one low-priority (interruptable) task, Peripheral Order Buffer (POB) execution, is clearly at variance with Fig. 4. This task executes orders that control peripheral equipment, e.g., open (close) relays; thus, after completing execution, it must wait a minimum period of time (20 ms) for the controllers to free and reset. Hence, this task is executed asynchronously with the rest of the schedule being rescheduled precisely 20 ms after completing. Moreover, these orders can be loaded by the main processor at any time; hence gating 4 applies.

2.7 Private branch exchange

Clocked schedules are common in many Private Branch Exchanges (PBXs). Again, HP work usually involves digit reception and transmission, and LP work includes scanning for new originations and receiver attachment, while F work is devoted to maintenance tasks.

These variants, and many others in actual systems, cause the models discussed above to be approximate at best. Nonetheless, as noted, they can often supply useful performance information. We will use these examples to look at some concrete quantification in Section IV.

III. APPROXIMATE ANALYSIS OF LONG-TERM DELAYS

For systems where short-term delays are an important performance measure, it is usually necessary to capture a considerable amount of scheduling detail (e.g., the dependencies and gating noted for MPIS in Section II). This problem has been addressed in Ref. 8, where exact solutions for realistic models are given. Here we concern ourselves with obtaining rather simple analytic approximations to long-term delays. In practice, the results have also been useful for medium- and sometimes short-term delay calculations.

3.1 Approximating task waiting time for the simplest system

Consider the simple clocked schedule depicted in Fig. 6. We assume

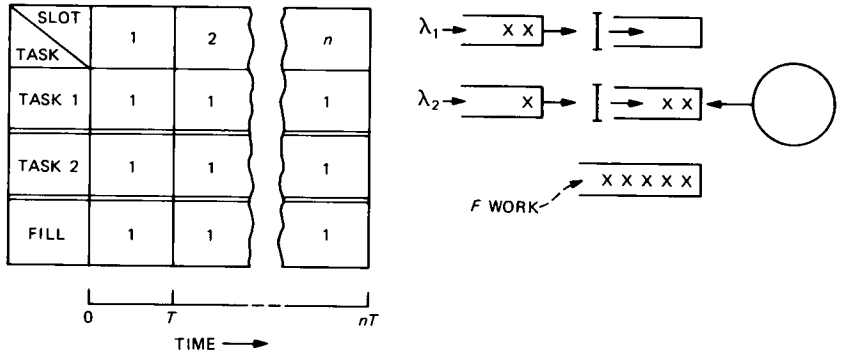


Fig. 6—Simplest clocked schedule.

that the arriving work associated with the ith task, $i = 1, 2$, forms sequences $\{X_1^k\}$, $\{X_2^k\}$ of independent, identically distributed random variables and that these sequences are mutually independent. We also assume that the work arrivals follow the gating 1 procedures described above. Denote the distribution functions for these work items by $F_i(x)$, i.e., $F_i(x) = P_r\{X_i \leq x\}$. Under these assumptions, the delays for task 1 can be obtained by studying a D/G/1 queue with (deterministic) interarrival time equal to T and service-time distribution given by $F_1(x)$. Reference 8 gives a variety of approximation methods for estimating the delays in such a system. In particular, if we denote the waiting time by t_1, then Ref. 9 shows how to determine suitable parameters a_1, C_1 such that

$$W_1(x) = 1 - C_1 e^{-a_1 x} \tag{1}$$

is a good approximation to $P_r\{t_1 \leq x\}$. (We discuss some specific choices of a_1, C_1 in Section IV.) To study the delays for task 2, we first define a random walk, $\{Y^k; y\}$ by

$$\begin{aligned} Y^{k+1} &= Y^k - (T - X_1^{k+1}) = Y^k - Z^{k+1}, \qquad k \geq 0 \\ N(y) &= \min\left\{n: \sum_{k=1}^{n} Y^k < 0, \qquad \text{given } Y^0 = y\right\}, \end{aligned} \tag{2}$$

where T is the time slot length.
Now define:

τ_N = time from arrival of an arbitrary batch of task 2 work until its processing begins (task 2 waiting time).

w = work backlog (task 1 and task 2) just before the start of an arbitrary time slot.

$W_c(n) = P_r\{\tau_N > nT\}$. (Complementary task 1 waiting-time distribution).

$G_c(n; y) = P_r\{\mathrm{N}(y) > n\}$.

$H(x) = P_r\{w \leq x\}$.

We then have that

$$W_c(n) = \int_0^\infty G_c(n; x)dH(x), \qquad n \geq 1$$
$$W_c(0) = 1 - H(0)F_1(0). \tag{3}$$

That is, if in the random walk (2), y is the backlog seen by an arriving task 2 batch of work, then Z^k is the amount of time available in the kth slot to work off this backlog (or add to it if $Z^k < 0$), Y^k is the remaining backlog at the end of the kth slot (start of the $(k+1)$st slot), and the new work for task 2 arriving at the beginning of slot 1 begins execution in the first slot that ends with $Y^k < 0$.

The usefulness of (3) depends on our ability to determine reasonable approximations for $G_c(n; x)$ and $H(x)$. First, we note that the work backlog, w, is precisely what would be seen by an arrival to a D/G/1 queue with service-time distribution equal to the convolution of F_1 and F_2, i.e., with service time $X_1 + X_2$. Again the approximation methods of Ref. 8 allow us to reasonably approximate $H(x)$ by using a simple exponential form, $\mathrm{H}'(x)$, i.e., with

$$H'(x) = W_2(x) = 1 - C_2e^{-a_2x}. \tag{4}$$

The central limit theorem implies that $G_c(n; x)$ is asymptotically Gaussian, for large n. We thus approximate $G_c(n; x)$ by $G_c'(n; x)$ defined by

$$G_c'(n; x) = \frac{1}{\sqrt{2\pi}} \int_{(n-m_N)/\sigma_N}^\infty e^{-y^2/2}dy, \tag{5}$$

where

$$m_N = \frac{x}{\overline{Z}}$$

$$\sigma_N^2 = \frac{x\sigma_z^2}{\overline{Z}^3}$$

and, as above, $Z = T - X_1$.

Equation (5) can be written as

$$G_c'(n; x) = \frac{1}{2}\left[1 - \mathrm{Erf}\left[\frac{A_1n}{\sqrt{x}} - A_2\sqrt{x}\right]\right], \tag{6}$$

where

$$A_1 = \bar{Z}^{3/2}/\sqrt{2\sigma_Z^2}$$

$$A_2 = \bar{Z}^{1/2}/\sqrt{2\sigma_Z^2}$$

and

$$\mathrm{Erf}(v) = \frac{2}{\sqrt{\pi}} \int_0^v e^{-s^2} ds.$$

Using (4) and (6) in (3) yields the approximation $W_c'(n)$ to $W_c(n)$

$$W_c'(n) = A_3 \exp(-A_4 n), \qquad n \geq 1$$

$$W_c'(0) = 1 - (1 - C_2)F_1(0), \tag{7}$$

where

$$A_3 = \frac{a_2 C_2}{2\sqrt{A_2^2 + a_2}[\sqrt{A_2^2 + a_2} - A_2]}$$

$$A_4 = 2A_1(\sqrt{A_2^2 + a_2} - A_2).$$

Equation (7) thus provides our desired approximation of the delay distribution for task 2. Note that (7) can readily be evaluated for n nonintegral yielding an interpolation formula.

While the use of (5) might seem to imply that n would need to be reasonably large for (7) to be useful, in fact, practical experience has indicated that (7) can provide a useful approximation, even for small n (e.g., $n = 1$). This will be illustrated via some examples in Section IV. We will look at how other performance measures can be approximated, e.g., task sojourn time and delays for F work. First, we look at how we can use this method to obtain approximate delays for the more general clocked schedule of Fig. 4.

3.2 Approximating low-priority task waiting time for a general clocked schedule

To treat the generalized clock schedule introduced in Section II, we need to extend the above approximation method to include an arbitrary number of tasks in a given time slot and, more important, to include more complex schedules. The former can be accounted for readily in the following manner. Assume for now that all slots have an identical schedule and that LP_j is the task to be analyzed. We then replace all tasks of higher priority than LP_j by a single task with an equivalent work load, i.e., we make the transformation

$$\sum_{i=1}^{m} \mathrm{HP}_i + \sum_{i=1}^{j-1} \mathrm{LP}_i \rightarrow \text{task 1}$$

$$\mathrm{LP}_j \rightarrow \text{task 2}$$

and use the above analysis. Two methods of obtaining the needed aggregated work-load characterization for task 1 have been found useful. The simplest method is to match the relevant means and variances. For example, if the work loads take the form $a_i + b_i J_i$ for the ith task, and λ_i is assumed Poisson, then we assume that the work load for the aggregated task, task 1, has the form $a + bJ$, where

$$a = \sum a_i$$

$$b\bar{J} = \sum b_i \bar{J}_i$$

$$b^2 \bar{J} = \sum b^2 \bar{J}_i.$$

Another alternative that is often preferable (particularly if the dominant work load is the result of a small number of jobs with large processing times) is to match the mean work load and the probability that there is no time available in a slot to execute task 2 work. That is, in addition to $a = \sum a_i$, $b\bar{J} = \sum b_i \bar{J}$, we must require that

$$P_r\{a + bJ > T\} = P_r\{\sum a_i + \sum b_i J_i > T\} = P^*.$$

The methods of Ref. 8 can be used to obtain P^*. A simpler method for estimating P^* that has proved useful is to use a "clustering" technique to replace the potentially large number of tasks that must be aggregated by precisely three, i.e.,

$$\sum a_i + \sum b_i J_i \rightarrow a_{,1} + b_{,1} J_{,1} + b_{,2} J_{,2},$$

where the $\sum a_i$ *and* tasks with small values of b_i map into $a_{,1}$, tasks with medium values of b_i map into $b_{,1}J_{,1}$, and tasks with large values of b_i map into $b_{,2}J_{,2}$. The resulting estimate of P^* can then readily be computed. This also provides a simple approximation technique for the probability of overrun, another important performance measure noted earlier.

We now consider a more complex schedule. Let $N_{\text{period},j}$ be the scheduling period for LP_j. We assume that $N_{\text{period},j}$ divides N_{period}, the period of the total schedule. We introduce a new clocked schedule with slot length $T_{,j} = N_{\text{period},j}\, T$, and a task 1 that is a suitable average of all tasks of higher priority than task j in all slots, i.e., let $K_j = N_{\text{period}}/N_{\text{period},j}$, and let $I_{\text{HP},k}$ and $I_{\text{LP},k}$ be the collection of indices for tasks of higher priority than task j in the kth scheduling interval, $k = 1, \ldots, K_j$. Then

$$\underset{k=1,\ldots,K_j}{\text{Average}} \left\{ \sum_{i \epsilon I_{\text{HP},k}} \text{HP}_i + \sum_{i \epsilon IP,k} \text{LP}_i \right\} \rightarrow \text{task 1}$$

$$\text{LP}_j \quad \rightarrow \text{task 2},$$

and we again use the above analysis. (Aggregation methods similar to those discussed above can be used here.)

3.3 Other performance measures

The method outlined above can be extended to obtain useful approximations to other performance measures. We first look at the conditional sojourn time of a task. For simplicity of notation, we consider the simple system of Fig. 6. The work-load aggregation given above can also be used here for more complex schedules.

We now define:

$\tau_S(y)$ = time from arrival of an arbitrary batch of task 2 work until completion of y units of work on that batch.

$S_c(n;\ y) = P_r\{\tau_S(y) > nT\}$ (Complementary conditional sojourn time).

We then have that

$$S_c(n;\ y) = \int_0^\infty G_c(n;\ x + y)dH(x). \tag{8}$$

Thus if $y = 0$, then $S_c(n;\ 0) = W_c(n)$.

Using (4) and (6) in (8), we obtain a rather complex integral involving the error function $\mathrm{Erf}(\cdot)$ that does not seem to have a closed form representation in terms of $\mathrm{Erf}(\cdot)$. Since our objective is to obtain, where possible, simple analytic approximations (exact numerical methods are available in Ref. 7), in addition to the Gaussian assumptions above, we assume the following (approximate) decomposition, $S_c'(n;\ y)$, of $S_c(n;\ y)$:

$$S_c'(n;\ y) = \int W_c'(n - m)dG'(m;\ y), \tag{9}$$

where

$$G'(m;\ y) = \frac{1}{\sqrt{2\pi}\sigma_{N_y}} \exp\left[-\frac{1}{2}\left[\frac{x - \bar{N}_y}{\sigma_{N_y}}\right]^2\right]$$

and

$$\bar{N}_y = \frac{y}{\bar{Z}}$$

$$\sigma_{N_y}^2 = \frac{y\sigma_z^2}{\bar{Z}^3},$$

i.e., $G'(m;\ y)$ is a (continuous in m) Gaussian approximation to $P_r\{N(y) = m\}$. This results in the following expression for $S_c'(n,\ y)$:

$$S_c'(n, y) = \frac{1}{2} W_c(n) \left[1\text{-Erf}\left[\frac{\bar{N}_y}{\sqrt{2}\sigma_{N_y}}\right]\right]$$
$$+ \frac{1}{2} A_3 \exp\left[A_4(\bar{N}_y - n) + \frac{A_4^2\sigma_{N_y}^2}{2}\right]$$
$$\cdot\left[\text{Erf}\left[\frac{\sqrt{2}A_4\sigma_{N_y}}{2} + \frac{\bar{N}_y}{\sqrt{2}\sigma_{N_y}}\right]\right.$$
$$\left. - \text{Erf}\left[\frac{\sqrt{2}A_4\sigma_{N_y}}{2} + \frac{\bar{N}_y}{\sqrt{2}\sigma_{N_y}} - \frac{n}{\sqrt{2}\sigma_{N_y}}\right]\right]$$
$$+ \frac{1}{2}\left[1 - \text{Erf}\left[\frac{n - \bar{N}_y}{\sqrt{2}\sigma_N}\right]\right] \tag{10}$$

and the A_i's are as before.

While (10) may seem complicated, it involves nothing more complex than an error function, Erf($\cdot$), whose efficient computation is available in most computer system libraries.

Equation (10) can be used to obtain approximations for a variety of important performance measures. For example, the time from arrival to completion of a batch of task 2 work is given by

$$S_c'(n) = \int_0^\infty S_c'(n; y) dF_2(y). \tag{11}$$

For the common case where X_2 is composed of individual jobs, we can use (10) to compute the average sojourn time seen by a job. Also, if, as a fill task, we schedule, say, T_f of maintenance as the next fill job starting at some time point, then $S_c'(n; T_f)$ provides the delay in finishing this task (with task 1 including *all* scheduled [nonfill] tasks).

Often, F work is repetitive, e.g., continuously scanning a set of lines. In such a case, there is no "waiting time" for fill work, i.e., by its definition, it starts immediately after completing. In this case, the elapsed time until all fill work is done can be approximated simply by $G_c'(n; T_S)$, where T_S is the time to scan all lines.

Finally, we note that in the gating model "gating 1," jobs are additionally delayed as they wait in the outer queue. Since this delay is independent of the delays internal to this system, performance measures including this delay can be computed by simply convolving this external delay with the internal job delays.

IV. EXAMPLES AND VALIDATION

The approximation method we have introduced is relatively simple, considering the complexity of the problem. Its usefulness, of course,

depends on how well it can capture the essence of the delay characteristics. We discuss here some applications and comparisons with: (1) exact results for the general class of clocked schedules introduced in Section II (using gating 1), (2) detailed simulation results that include all of the inherent work-load dependencies in an actual system, and (3) a comparison with some actual field data.

The approximation results were computed using a prototype software package, PACS (Performance Analysis of Clocked Schedules). This explicit implementation of our approximations determines C_i and a_i in the following manner. First, the service-time distribution in a D/G/1 queue is replaced by an appropriate hyperexponential, exponential, Erlangian, deterministic distribution that matches the first two moments of the service time. Then the approximation referred to as the A^* approximation in Ref. 9 is used to determine C_i and a_i. (For extreme light or heavy loads the approximations $A_{LT}, A_{H,0}$ of Ref. 9 are used.)

The first two examples considered use a mean and variance match of the aggregated work loads to a Gaussian distribution (see below). The third example used the clustering method noted above, while the fourth example again used a mean and variance match.

Example 1: As a first example we consider the simple schedule of Fig. 6 with the following parameter values:

T (Time slot length) = 10 ms
λ_i (Poisson arrival rate of jobs for task i) = 0.8/10 ms = 0.08/ms
b_1 (Time to process each job for task 1) = 5.0 ms
b_2 (Time to process each job for task 2) = 4.9 ms.

The resulting approximation for the backlog (eq. [4]) is

$W_2(x) = 1 - C_2 e^{-a_2 x} = 1 - 0.554 e^{0.0760x}$ the variable $Z = T - X_1 =$ 10 ms − 5.0 ms (J_1), where J_1 = number of jobs found for task 1. For eqs. (5) and (10) we thus have

$\bar{Z} = 6.0$ ms
$\sigma_Z^2 = 20.0$ ms^2.

The resulting waiting time (eq. [5]) and sojourn time (eq. [10]) for task 2 is shown in Fig. 7, where it is compared with the exact results obtained by the methods described in Ref. 7. Even for this simple example, the exact equilibrium solution is quite complex, being quasi-periodic in nature. Our simple method is seen to provide reasonable approximations to the waiting and sojourn time.

Example 2: Here we consider the more complex schedule shown in Fig. 8 with the parameters indicated. We look at the sojourn-time distribution for task 7. The work-load aggregation discussed above thus leads to $K = 2$, $T_{,j} = 40$ ms. Task 1 work is characterized by the

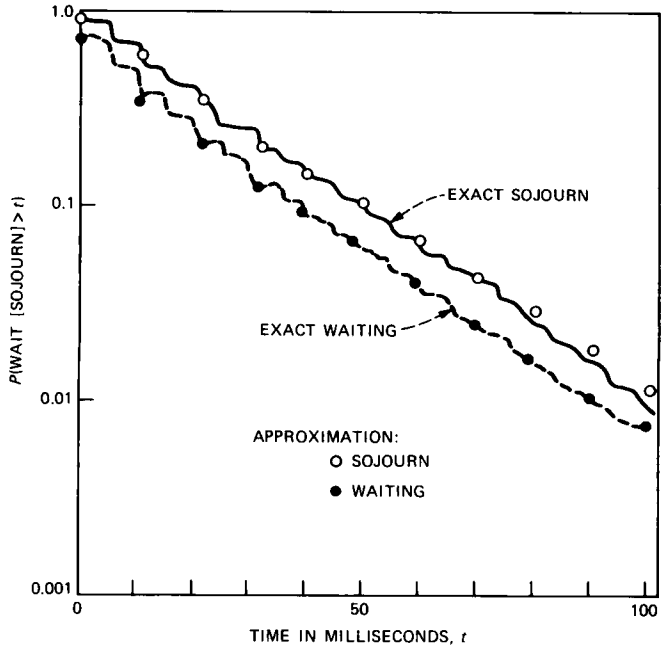
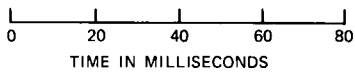


Fig. 7—Waiting and sojourn times for Example 1.

SLOT / TASK	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	0	1	0
4	0	1	0	1
5	1	0	0	0
6	0	0	1	0
7	0	1	0	1
F	1	1	1	1



λ_i	b_i
0.050	2.0
0.050	2.0
0.075	2.0
0.050	3.0
0.050	3.0
0.063	2.0
0.050	3.0

Fig. 8—Schedule for Example 2.

mean and variance of the task i ($i = 1, \ldots, 6$) work falling in each of two slots (of length 20 ms.) Figure 9 shows the resulting approximation for the sojourn-time distribution again compared with the exact results obtained from the methods of Ref. 7.

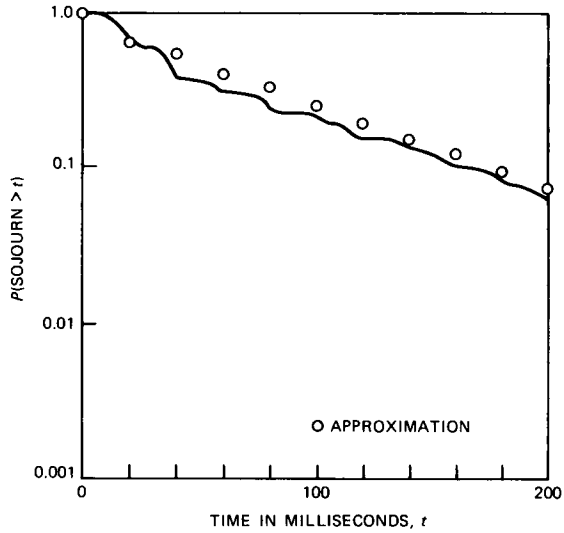


Fig. 9—Sojourn time for Example 2.

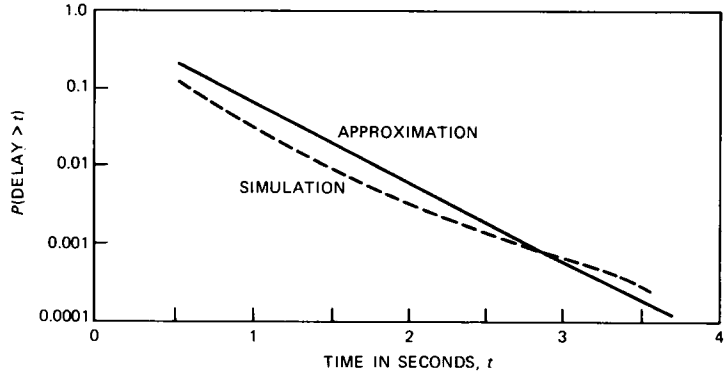


Fig. 10—Receiver attachment delay for Example 3.

Example 3: As noted in Section II, one of the LP tasks for the PBX discussed is the attachment of a digit receiver for a new line origination. Figure 10 shows the delays in receiver attachment as computed from our approximation method compared with the results of a detailed simulation model. This is a call-by-call simulation that includes most of the work-load dependencies associated with the actual system. We again see quite reasonable agreement.

Example 4: In the SP noted earlier, line scanning is a fill task. Figure 11 shows the line-scan delays predicted by our approximation as compared with data collected at a field site. We see that the approximation results fall well within the observed band of field data.

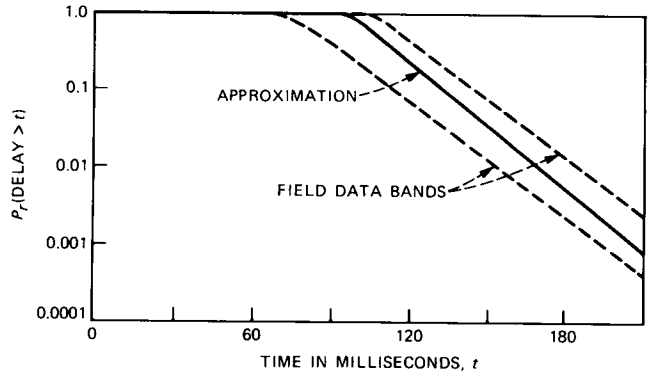


Fig. 11—Line-scan delay for Example 4.

V. CONCLUDING REMARKS

The approximation methods presented here, while quite simple in nature, have been used in the performance optimization of many systems. Often, these relatively easy-to-compute approximations are used in the preliminary design phase to determine several feasible candidate schedules, as well as to identify performance issues needing closer study. The more precise methods of Refs. 7 and 8 then have been used to address these issues. The final one or two potential schedules generally are then evaluated using detailed call-by-call simulation. In the case of modifications to existing systems, field trials are conducted for validation. For new systems, predicted performance is verified in laboratory models and early field introduction as needed. This has helped bring high-performance, high-capacity systems to fruition.

REFERENCES

1. D. R. Manfield and P. Tran-Gia, "Queuing Analysis of Scheduled Communications Phases in Distributed Processing Systems," *Performance 81, Proc. 8th Int. Symp. Comp. Perf. Mod., Meas., and Eval.*, Amsterdam, November 4–6, 1981, F. J. Kylstra, ed., New York: North-Holland, pp. 233–50.
2. A. A. Fredericks, "Analysis and Design of Processor Schedules for Real Time Applications," in *Applied Probability-Computer Science: The Interface*, Vol. 1, R. L. Disney and T. J. Ott, eds., New York: Birkhauser, 1982, pp. 433–50.
3. P. J. Kuehn, "Analysis of Switching System Control Structures by Decomposition," Proc. 9th ITC, Spain, 1979.
4. P. Tran-Gia and H. Jans, "Delay Analysis of Clocked Event Transfer in Distributed Processing Systems," ORSA/TIMS, Detroit, April 19–21, 1982.
5. H. Weisschuh, "Development of the Control Software for a Stored Program Controlled PCM Switching System," 24th Report on Studies in Congestion Theory, Univ. of Stuttgart, 1977.
6. A. A. Fredericks, "Analysis of a Class of Schedules for Computer Systems With Real Time Applications," *Performance of Computer Systems*, M. Arato, A. Butrimenko, E. Gelenbe, eds., New York: North-Holland, 1979, pp. 201–16.
7. M. H. Ackroyd, "Numerical Computation of Delays in Clocked Schedules," AT&T Technical Journal, this issue.

8. B. T. Doshi, "Analysis of Clocked Schedules—High-Priority Tasks," AT&T Technical Journal, this issue.
9. A. A. Fredericks, "A Class of Approximations for the Waiting Time Distribution in a G/G/1 Queueing System," B.S.T.J., *61*, No. 3 (March 1982), pp. 295–325.

AUTHORS

Darlene F. DeMaio, B.A. (Mathematics), 1976, Lycoming College; M.S. (Industrial and Applied Mathematics), 1981, Polytechnic Institute of New York; AT&T Bell Laboratories, 1976—. Ms. DeMaio has been involved in studying the performance of computer operating systems using both analytic and simulation models. Her current interest is in the development of user friendly performance analysis software.

Brian L. Farrell, B.S. (Mathematics, Computer Science), 1978, St. Peter's College; M.S. (Operations Research), 1982, Columbia University; AT&T Bell Laboratories, 1978—. Mr. Farrell has worked on projects involving the construction of simulation models for real-time computer systems. He has also worked on a software package for the analysis of clocked schedules and more recently has been involved in performance analysis and capacity estimation for operations systems.

Albert A. Fredericks, B.S. (Mathematics), 1962, Fairleigh Dickinson University; M.S., Ph.D. (Mathematics), 1965 and 1970, respectively, Courant Institute, New York University; AT&T Bell Laboratories, 1961—. Mr. Fredericks is Head of the Performance Analysis Department. His responsibilities include the development of methods for analyzing the performance of computer/communications systems and manufacturing systems.