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

# Numerical Computation of Delays in Clocked Schedules

By M. H. ACKROYD*

(Manuscript received December 8, 1983)

A discrete-time model for clocked schedules is described. The distributions of the waiting and sojourn times for tasks in this model can be computed by a two-stage process. The first stage involves computing the distribution of the work awaiting execution at each time slot in the schedule. The second stage yields the required delay distributions. The computations in both stages involve discrete convolutions, which can be computed via the fast Fourier transform. The availability of an exact method of analysis permits the detailed study of schedules. It also enables the accuracy of approximate methods of analysis, such as those in a companion paper by Fredericks et al., to be determined. The present paper considers just one form of scheduling mechanism; a companion paper by Doshi considers several other mechanisms.

## I. INTRODUCTION

Clocked schedules provide a means for organizing and controlling the execution of tasks in software for switching systems or other systems having strict timing requirements.[1,2] This paper describes a method for computing schedules. The method is based on the use of a discrete-time model, which makes it possible to compute the required distributions exactly, except for the effect of computational error. The availability of an exact method of analysis has some advantages, even though it can be somewhat expensive in terms of the amount of computation needed. An exact method permits the detailed study of

* AT&T Bell Laboratories.

the behavior of clocked schedules and the comparison of schedules that differ only in small details. In addition, it can be used to check the accuracy of more economical, approximate methods of analysis.

The next section of the paper describes the discrete-time model for clocked schedules that allow the methods presented here. Section III explains the method for the numerical computation of the required delay distributions. Section IV describes some aspects of an implementation of the method, and Section V gives some illustrative results computed by its use.

## II. CLOCKED SCHEDULE MODEL

There are two aspects of the model that need to be described. First, we need to describe the discipline under which the execution of the different software tasks, each having its own priority, is organized and controlled. Second, we need to specify the statistical characteristics of the execution times of the tasks.

### *2.1 Organization and control of task execution*

Figure 1 illustrates how task execution is organized. There are $N_{task}$ tasks, each with a unique priority. The priorities are numbered from

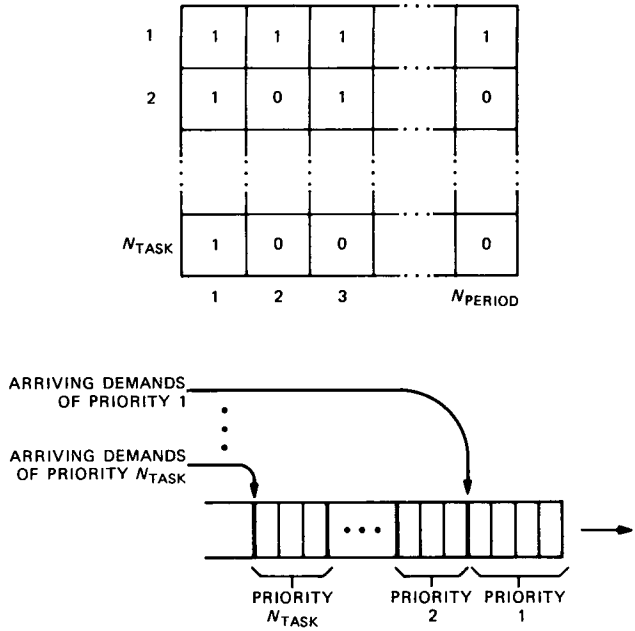



Fig. 1—Schedule table and demands for the execution of tasks waiting in a preempt resume head-of-line priority queue.

one, top priority, to $N_{task}$, lowest priority. The schedule is specified by a table whose elements are ones and zeros. The table has $N_{period}$ columns and is considered to be repeated periodically. Each tick of the system clock, at $t = 0, T, 2T, \cdots$, corresponds to one column of the schedule table, and each task corresponds to one row of the table. The periods between consecutive ticks are referred to as *slots*.

When a clock tick occurs, the system consults the schedule table to determine which tasks have become due for execution. The tasks due for execution are indicated by the presence of ones in the positions corresponding to these tasks and the current clock tick. The system instantly places demands for the execution of the newly scheduled tasks in a queue. This queue, which is illustrated in Fig. 1, is a head-of-line priority queue.[3] In the queue, demands for task execution are grouped in order of task priority, the group of demands for the highest-priority task being at the front of the queue.

All the tasks are considered, in this paper, to be of interruptable type,[1] i.e., if a task of priority $i$ is still being executed when a clock tick occurs, its execution is instantly interrupted, rather than running on to completion, through the occurrence of the clock tick. The demand for the execution of the task is put back in the queue at the head of the group of demands of priority $i$ until no demands of higher priority remain to be handled. The execution of the task of priority $i$ is then resumed where it was left off, with no overhead (extra execution time) involved in the resumption. The queue is of the head-of-line, preempt resume type.

### *2.2 Task execution times*

We assume that the time required to execute a task, given that it has exclusive use of the processor, is an independent, discrete random variable. This key assumption makes possible the numerical method of solution described in the next section. Task execution times are of the form $k\Delta$, where $k$ is a random integer and where the interval $\Delta$ is an integer submultiple of the clock period $T/\Delta = N_{subdv}$. The execution time of each task is characterized by the distribution of the random integer $k$. The method of this paper applies to arbitrary distributions for $k$, which need to be available in numerical form.

If experimental information about the task execution times is available, e.g., as histograms, these data can be used directly to provide the numerical input required by the analysis. Alternatively, the numerical input required can be provided by evaluating a formula that provides a model for the execution times of the tasks. One such model, used in Ref. 2, is based on the assumption that the task execution time consists of an initial overhead $a$, followed by $n$ executions of a job, where $n$ is Poisson distributed and where each execution of the job requires time

$b$. Provided that $a$ and $b$ are integer multiples of $\Delta$, this model fits the form assumed in this paper. Examples of the use of this model are given in Section V.

### *2.3 Performance measures*

Various measures of the performance of a system controlled by a clocked schedule may be of interest. Here we deal with two: the distribution of the waiting time of a task and the distribution of its sojourn time. The waiting time for a task is the time that elapses from the instant it is scheduled until its execution starts. The sojourn time is the time from the instant a task is scheduled until its execution is completed.

As mentioned before, the execution of a task may be considered to consist of a number of repeated executions of a single job. We do not, in this paper, deal with the waiting and sojourn times for the individual jobs that, in a batch, constitute the task. Of course, the waiting time for a task provides a lower bound on the waiting time for a job within the task. Similarly, the sojourn time for a task provides an upper bound on the sojourn time for a job.

In this paper we are concerned with the behavior of systems in equilibrium. Nonetheless, the delay distributions will generally be time dependent. Consider, for example, a task of low priority that is scheduled at several slots. The delays for the low-priority task will generally be larger at slots where much work of higher priority is also scheduled than at slots where little work of higher priority is scheduled.

In assessing the performance of a schedule to determine whether it meets a specification, the average of the delay distributions of a task will usually be of prime interest—the average being computed over all slots where the task is scheduled. Sometimes, though, the individual delay distributions will be of interest, such as when a schedule is being analyzed in detail in order to improve it. In either case, the approach described in the next section involves computing the delay distributions for a task at each slot where it is scheduled. The average distribution is obtained by averaging the individual distributions.

## III. COMPUTATION OF DELAY DISTRIBUTIONS

As shown in the previous section, a system controlled by a clocked schedule can be considered as a nonstationary multiclass priority queueing system. The analytical solution of such a system would be difficult. However, because of the form assumed for the distributions of task execution time, numerical results can readily be obtained. In this section we detail the computation of the waiting-time distribution. The computation of the sojourn distribution is very similar, so it is mentioned only briefly.

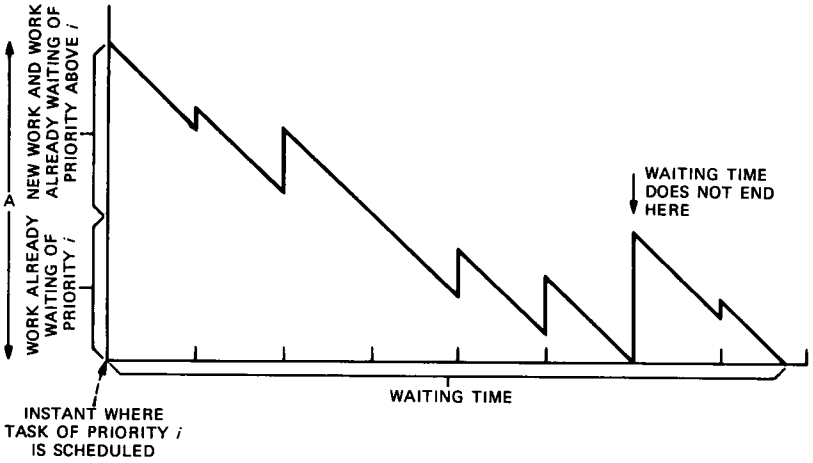



Fig. 2—Relationship between task waiting time and the work waiting in the queue.

We consider a task of priority $i$. At the instant the task is scheduled to be executed, the demand for its execution joins the queue, behind any previously scheduled demands for execution of the task still waiting to be handled. Its waiting time consists of the sum of two components, as illustrated in Fig. 2. The two parts of the waiting time are as follows:

1. The time required to clear any backlog of work, also of priority $i$, which was already waiting at the instant the task was scheduled. The wait of the task is unaffected by any arrivals of priority $i$ after its own, so these do not need to be considered.

2. The time required to clear the backlog of work, if any, of priority higher than $i$ that was already waiting at the instant it has scheduled plus the time required to deal with any work, of priority higher than $i$, that is scheduled during its wait.

We now elaborate on our definition of waiting time. The waiting time of a task is considered to have ended when the waiting work taking precedence over it becomes zero and then remains zero for a finite period. The waiting time is not considered to have ended if the waiting higher-priority work merely becomes zero for an instant—an event that can have finite probability, with the discrete distributions considered here. An example of such an event is illustrated in Fig. 2.

The situation is quite similar for sojourn times. The sojourn time of a task ends when the sum of the work in the queue taking precedence over the task plus the work remaining to execute the task itself becomes zero. In contrast to our definition of the waiting time, the sojourn is considered to have ended if this sum becomes zero for even just an instant.

The distribution of the waiting time for a task of priority $i$ can be computed in two stages. The first stage involves computing the distribution of the work, of priority $i$ and higher, awaiting execution immediately after the task is scheduled. This work is indicated by the time 'A' in Fig. 2. The second stage involves computing the distribution of the waiting time itself from the results of the first stage. This is done by a recursive process in which the $l$th stage of the recursion yields the probability that the waiting time has duration $l\Delta$. The two stages are now described in more detail.

### *3.1 Computing the distribution of the work awaiting execution*

We need to compute the distribution of the work, of priority $i$ and higher, awaiting execution just after each clock tick. $y_n$, the work of priority $i$ and higher awaiting execution at $t = nT_+$, can be expressed in terms of $y_{n-1}$ and $x_n$, the new work joining the queue at $t = nT$. The equation is

$$y_n = (y_{n-1} - T)^+ + x_n , \tag{1}$$

where $(\cdot)^+ = \max(0, \cdot)$. This expresses the fact that the work awaiting execution is diminished by $T$ during a slot, but the waiting work cannot become negative.

The right side of (1) involves the addition of a pair of independent random variables. The distribution of the sum of independent variables is obtained by convolving their distributions, which leads to a formula for the distribution of $y_n$ in terms of the distribution of $y_{n-1}$:

$$f(n, i, k) = [\Psi_{k,N_{\text{subdv}}} f(n - 1, i, k)] *_k q(n, i, k). \tag{2}$$

Terms used in (2) are defined below:

$f(n, i, k)$ = the probability that $y_n$, the work in the queue, of priority $i$ and higher, at $t = nT_+$, would require a total time $k\Delta$ for its execution. $i$, though indicated explicitly, is held constant throughout all computations.

$q(n, i, k)$ = the probability that $x_n$, the new work of priority $i$ and higher, scheduled for execution at $t = nT$, would require a total time $k\Delta$ for its execution.

$\Psi_{k,N_{\text{subdv}}}$ = the operation of shifting a distribution $N_{\text{subdv}}$ places left on the $k$ axis and then sweeping the probability on the negative $k$ axis up to the origin.[3] The shift corresponds to the subtraction of $T = \Delta N_{\text{subdv}}$ in (1), and the sweep corresponds to the fact that the waiting work can be reduced only to zero.

$*_k$ = the operation of convolution with respect to the discrete variable $k$, e.g., $u(k) *_k v(k) = \sum_{l=-\infty}^{\infty} u(l)v(k - l)$.

By starting with a known distribution $f(0, i, k)$, such as the distribution corresponding to an empty queue, (2) can be evaluated repeat-

edly to give the distributions $f(n, i, k)$ for successive values of $n$. The volume of computation needed would normally be quite unreasonable if (2) were evaluated directly. However, by using a fast Fourier transform algorithm,[3] we can reduce the amount of computation to reasonable proportions.

The distribution of the work arriving at the queue, $q(n, i, k)$ is, due to the periodicity of the schedule table, periodic in $n$. That is, $q(n, i, k) = q(n + N_{\text{period}}, i, k)$, for all $n$. Provided that the average work scheduled per clock tick is less than $T$, the queue will be stable, and $f(n, j, k)$ will converge to a periodic steady state. (Repeated evaluation of eq. [2] is analogous to a stable time-invariant linear system driven by a periodic input. After initial transients have decayed, only the periodic response remains.) On convergence, we have obtained the periodic steady-state distribution, $\hat{f}(m, i, k)$, for $m = 1, 2, \cdots, N_{\text{period}}$, i.e., for each slot in the schedule. Except in special cases, convergence requires an infinite number of iterations, so an appropriate criterion must be used, in practice, to determine when the iterations may be stopped.

### 3.2 *Computation of the waiting-time distribution*

The waiting-time distribution can be computed from the work distribution, $\hat{f}(m, i, k)$, by a recursive process. We require the computation of a sequence of probabilities $w(m, i, k)$, for $k = 0, 1, \cdots$, where $w(m, i, k)$ denotes the probability that a task of priority $i$, scheduled at slot $m$, waits a time period of $k\Delta$ until its execution starts. Equivalently, we require the computation of the probability that the work in the queue, having precedence over the task, first becomes zero at time $k\Delta$ after the task is scheduled (and remains zero for at least $\Delta$).

We define a sequence of conditional distributions $r(m, i, k, l)$, $l = 0, 1, \cdots$, where $r(m, i, k, l)$ is the probability that the work in the queue having precedence over the task of priority $i$ scheduled at slot $m$ is $k\Delta$, given that the waiting time is $l\Delta$ or more (for $l > 0$). For $l = 0$, $r(m, i, k, l)$ is the unconditional distribution of the waiting work that has precedence over the task of interest, at slot $m$. This work is the sum of the work, of priority $i$ and higher, left over from the previous slot, plus any new work, of priority higher than $i$, joining the queue at slot $m$. By convolving the distributions of the two parts of the sum, we obtain the required distribution:

$$r(m, i, k, 0) = [\Psi_{k, N_{\text{subdv}}} \hat{f}(m - 1, i, k)] *_k q(m, i - 1, k). \quad (3)$$

The convolution in (3) can be computed, as previously, by the use of a fast Fourier transform algorithm.

The probability that the waiting time is zero is given by the origin

point of the distribution computed from (3), i.e., $w(m, i, 0) = r(m, i, 0, 0)$.

We now consider the computation of the probability that the waiting time is $l\Delta$, for $l > 0$. $r(m, i, 0, l)$ is the probability that the work in the queue having precedence over the task of interest becomes zero at $l\Delta_+$ after the task is scheduled, given that the task's wait has not yet ended. The unconditional probability that this work becomes zero at $l\Delta_+$ after the task is scheduled at slot $m$ is given by

$$w(m, i, l) = \left\{ 1 - \sum_{j=0}^{l-1} w(m, i, j) \right\} r(m, i, 0, l), \tag{4}$$

i.e., the required probability value is obtained from the previously computed points in the waiting distribution and the $l = 0$ point on the conditional distribution $r(m, j, k, l)$.

To complete the $l$th stage of the recursion, the conditional distribution $r(m, j, k, l + 1)$ must be computed. It is the distribution of waiting work having precedence over the task of interest at $(l + 1)\Delta_+$ after the clock tick where the task was scheduled, given that the wait did not end at $l\Delta_+$ after the task was scheduled. (We consider $l\Delta_+$ because the wait is not considered to have ended if the waiting work becomes zero at $l\Delta_-$ and then immediately becomes nonzero again due to the arrival of new high-priority work.) $r(m, j, k, l + 1)$ is obtained by modifying $r(m, j, k, l)$ in three ways, as follows:

1. The $k = 0$ point is set to zero and the remaining points are normalized. This discounts the case where the waiting time is $l\Delta$.

2. The conditional distribution is shifted one place to the left on the $l$ axis to account for the reduction of the waiting work during an elapse of time $\Delta$.

3. The conditional distribution is convolved with the distribution of the new work, of priority higher than $i$, joining the queue at $(l + 1)\Delta$ after the task of interest was scheduled. If none is scheduled to join the queue or if $(l + 1)\Delta$ is not a clock instant, the conditional distribution is left unchanged.

The new conditional distribution is thus given by

$$r(m, i, k, l) = \left[ \Psi_{k,1} \left( \frac{r(m, i, k, l) - r(m, i, 0, l)\delta(k)}{1 - w(m, j, l)} \right) \right] *_k g(k), \tag{5}$$

where $\delta(k) = 1$, $k = 0$; $\delta(k) = 0$, otherwise, and where $g(k)$ is the distribution of arriving work having priority higher than $i$, i.e.,

$$g(k) = \begin{matrix} q(m + l/N_{\text{period}}, i - 1, k), \text{ for } l \bmod N_{\text{period}} = 0 \\ \delta(k), \text{ otherwise.} \end{matrix}$$

The recursive scheme defined by (4) and (5) is started with $r(m, i, k, 0)$, given by (3).

The normalizing division by $1 - w(m, i, l)$ in (5) is followed, at the next stage of the recursion, by a corresponding multiplication by $1 - w(m, i, l)$ in (4). In the actual computations, the divisions in (5) and the corresponding multiplications in (4) can be omitted. The computed intermediate quantities then lose their interpretation as probabilities, of course.

Evaluation of (6) gives the waiting-time distribution for a task scheduled at the $m$th slot in the schedule. As mentioned before, for a task scheduled more than once per period of the schedule, the waiting-time distribution will generally vary with $m$. The computations involving (4) and (5) can be repeated, for each slot where the task is scheduled, to obtain the individual waiting distributions. When the average of the distributions is required, this can be obtained by summing the individual distributions and normalizing the sum.

### *3.3 Sojourn distribution*

The computation of the sojourn distribution is similar to the computation of the waiting distribution, but it differs in two ways. We are concerned with when the waiting work of priority $i$ and higher becomes zero, rather than the work whose priority is higher than $i$. Also, the sojourn ends even if this work becomes zero for just an instant.

To consider work of priority $i$ and higher, (3) is replaced by

$$r(m, i, k, 0) = [\Psi_{k,N_{\text{subdv}}} \hat{f}(m - 1, i, k)] *_k q(m, i, k)$$

or, equivalently from this equation and (2),

$$r(m, i, k, 0) = \hat{f}(m, i, k). \tag{6}$$

Equation (5) is used unchanged. So that we do not disregard cases where the waiting work becomes zero for just an instant, we use the conditional distribution of the waiting work immediately prior to clock instants, rather than immediately after. $r(m, i, k, l-1)$ is the conditional distribution of work at $t = (l - 1)\Delta_+$ after the task is scheduled, so $r(m, i, k + 1, l - 1)$ is the conditional distribution at $l\Delta_-$. Equation (6) is therefore replaced by

$$s(m, i, l) = \left\{\left[1 - \prod_{j=0}^{l-1} s(m, i, j)\right]\right\} r(m, i, 1, l - 1),$$

where $s(m, i, l)$ denotes the sojourn distribution for the task or priority $i$ scheduled at slot $m$.

## IV. IMPLEMENTATION

The method given in the previous section has been implemented as a computer program in which all the convolutions are computed via Fast Fourier Transform (FFT) routines.[4] The delay distributions for a task of priority $i$ are computed in four principal steps:

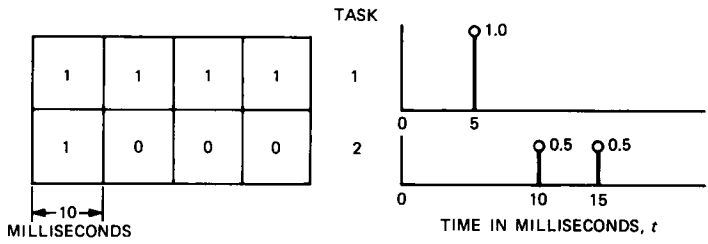
1. Initialization
2. Distribution of waiting work, via (2)
3. Waiting-time distributions, via (3, 4)
4. Sojourn-time distributions, via (6, 7).

The major part of the initialization step is the computation of the distributions $q(m, i, k)$ for each slot $m$ in the period of the schedule. This is done by convolving the individual task execution time distributions, as specified by the presence of ones in each column of the schedule table.

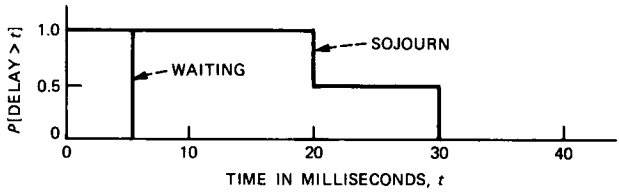
Convolutions performed via the FFT are cyclic convolutions of finite sequences, rather than the aperiodic convolutions of infinite sequences required here.[4] In principle, this introduces error, due to wraparound of the tails of the computed distributions. However, when a suitably large FFT block size is used, the magnitude of the tail extending beyond the end of the block is small by comparison with the error due to arithmetic rounding (or truncation). The effect of wraparound is then negligible, with no appreciable effect on the computed results.

The iterations in computing the distributions of the waiting work are computationally expensive. Of course, if the FFT were not used, the computation would be more expensive still—by factors of over one hundred, for large block sizes. To minimize the computational expense, only as many iterations should be used as necessary for a required level of accuracy. In using the method, the following criterion has been found useful. The computed cumulative distribution of the waiting work at the first slot in the schedule is compared with the corresponding distribution obtained $N_{\text{period}}$ iterations previously. In equilibrium, these cumulative distributions would be identical. The iterations are stopped when the computed cumulative distributions agree completely, in element-by-element comparison, to $N_{\text{place}}$ decimal places, $N_{\text{place}}$ being specified by the user. Experience shows that this is a reasonably satisfactory rule for halting the iterations. Experience with examples such as the one mentioned below suggests that the use of this criterion usually results in the computed cumulative distributions of waiting and sojourn times also having an accuracy of about $N_{\text{place}}$ decimal places in each point. If a large value of $N_{\text{place}}$ is specified, the desired accuracy may be unachievable, due to the effects of arithmetic rounding.

Testing an implementation of the method presents a difficulty because the delay distributions can be calculated manually only for




(a)




(b)

Fig. 3—Example of a simple schedule whose delay distributions can readily be calculated.

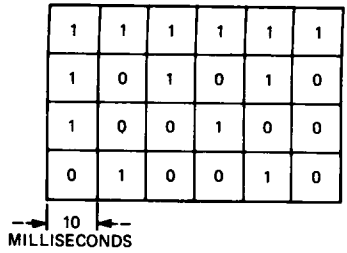
simple, somewhat atypical examples. Statistical simulations can be used to check for gross errors in the implementation, but it would be impractical to use simulation results to determine the accuracy of the extremes of the distribution tails. One example whose solution can be calculated manually is a schedule with a single task, scheduled every clock instant. If the execution time of the task is zero with probability $\alpha$, and $2T$ with probability $1 - \alpha$, it is straightforward to show that the waiting-time distribution is geometric, with parameter $(1 - \alpha)/\alpha$.

In the present form of the program, the FFTs are computed in double precision floating-point arithmetic, with the remainder of the computation being done in single precision. The accuracy of the computed delay distributions depends on the particular problem, but results are typically accurate to four decimal places when an IBM 3081 computer is used. When a *VAX** machine is used, which has the same word length but has different floating-point arithmetic characteristics, results are typically accurate to five decimal places.

## V. EXAMPLES

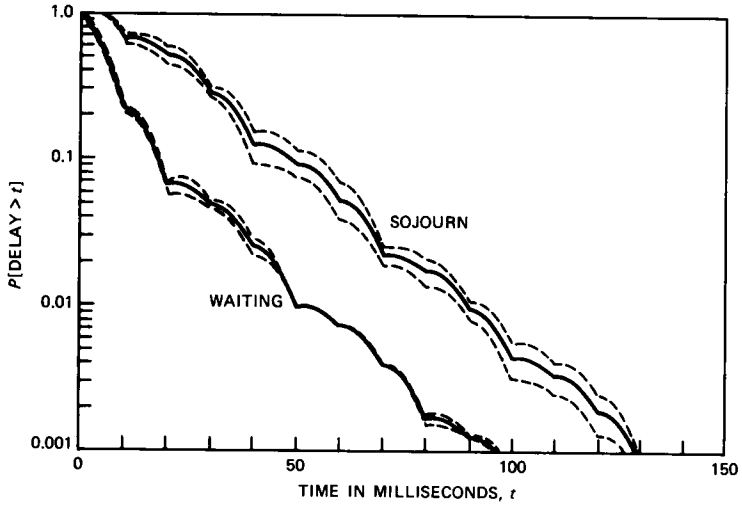
Figure 3a illustrates a simple example of a clocked schedule; the schedule table and the distributions of the execution times for the

* Trademark of Digital Equipment Corporation.




| TASK | $a$ | $b$ | $\bar{n}$ |
|---|---|---|---|
| 1 | 0.1 | 1.5 | 0.600 |
| 2 | 0.1 | 2.1 | 3.762 |
| 3 | 0.1 | 3.2 | 0.906 |
| 4 | 0.1 | 0.4 | 14.75 |

(a)

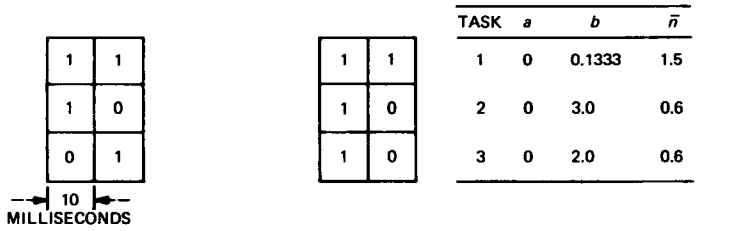



(b)

Fig. 4—Example of a schedule where the waiting and sojourn distributions depend on the slot at which the task is scheduled.

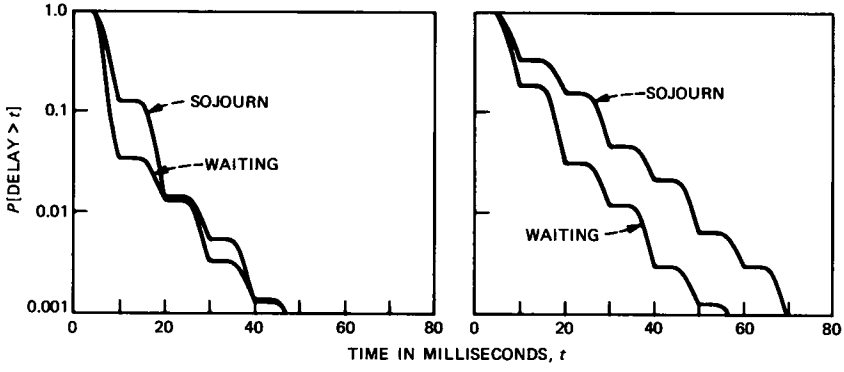
individual tasks are shown. The task of higher priority always takes 5 ms to execute. The task of lower priority, if it had sole use of the processor, would take 10 or 15 ms, with equal probability. This example is unusual in that the waiting and sojourn distributions can readily be computed manually. For the task of lower priority, the waiting time is always 5 ms—the time it waits while the higher-priority task is executed. In each slot, there are 5 ms available for the task of lower priority. Its sojourn time is therefore, with equal probability, 20 or 30 ms. The complementary cumulative distributions of waiting and sojourn time are shown in Fig. 3b.

Figure 4a illustrates another simple clocked schedule. For each task, the execution time has the form $a + nb$, where $a$ denotes the overhead




| TASK | $a$ | $b$ | $\bar{n}$ |
|---|---|---|---|
| 1 | 0 | 0.1333 | 1.5 |
| 2 | 0 | 3.0 | 0.6 |
| 3 | 0 | 2.0 | 0.6 |

(a)




(b)

Fig. 5—Two schedules, identical except for the slot at which task 3 is scheduled, illustrating how delays can depend on details of the schedule.

incurred in starting the execution of the task, $b$ denotes the time to execute a single job, and $n$ is the number of jobs in the task, $n$ being Poisson distributed. This is a form for the task execution time assumed in Ref. 1. Figure 4a shows the schedule table and, for each task, the values of $a$ and $b$, and of $\bar{n}$, the expected number of jobs in the task. Figure 4b shows the complementary cumulative distributions of the waiting and sojourn times for task 4. In this example, the delay distributions differ according to the time slot at which the task is scheduled. The individual distributions are shown hatched; their average is shown as a continuous line. The presence of cusps is quite typical of delay distributions for low-priority tasks. They exist because, if a task of low priority is still being executed when a clock tick occurs, then it will probably have to wait considerably longer, while the newly scheduled high-priority work is executed.

Figure 5a illustrates two simple schedules; they differ only in the slot at which the task of lowest priority is scheduled. Figure 5b shows the resulting delay distributions for the task of lowest priority. An apparently minor change in the schedule results in significant increases in delay.

## VI. CONCLUSION

A method has been presented, based on the use of a discrete-time model, for the numerical computation of delay distributions for interruptable tasks in clocked schedules. The method is particularly useful when a detailed examination of the characteristics of a schedule is needed, such as how the delays depend on the slot where a task is scheduled. It does not depend on assumptions that delays are long, so it gives information about delays that are smaller than a clock period, as well as for longer delays. An important use of the method is checking the accuracy of computationally more economical, though approximate, methods of analysis.

Only systems in equilibrium have been discussed here. However, the method is capable of being extended to the analysis of transient conditions such as those that occur when a system controlled by a clocked schedule is subjected to a sudden increase of traffic.

The method presented here uses an iterative method to compute the equilibrium distribution of the work awaiting execution at each slot in the schedule. An alternative approach involves the direct solution of the equilibrium equations. Levinson's method for the solution of block Toeplitz systems[5] has been used, experimentally, for the analysis of clocked schedules by the direct solution of the equilibrium equations. However, this approach must be further developed in order to become a practical alternative to the iterative method described in this paper.

## VII. ACKNOWLEDGMENTS

I am grateful to A. A. Fredericks and B. T. Doshi for introducing me to clocked schedules and for many discussions on the topic. I am indebted to B. S. Gotz for her careful reading of an earlier version of this paper.

## REFERENCES

1. A. A. Fredericks, B. L. Farrell, and D. F. DeMaio, "Approximate Analysis of a Generalized Clocked Schedule," AT&T Tech. J., this issue.
2. A. A. Fredericks, "Analysis of a Class of Schedules for Computer Systems With Real Time Applications," *Performance of Computer Systems*, M. Arata, A. Butrimenko, E. Gelenbe, eds., New York: North-Holland, 1979, pp. 201–16.
3. L. Kleinrock, *Queueing Systems*, New York: Wiley, 1975, Vol. I, Ch. 8; Vol. II, Ch. 3.
4. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975.
5. M. H. Ackroyd, "Stationary and Cyclostationary Finite Buffer Behaviour Computation via Levinson's Method," AT&T Bell Lab. Tech. J., *63*, No. 10 (December 1984), pp. 2159–70.

## AUTHOR

**Martin H. Ackroyd,** B.Sc. (Electronic and Electrical Engineering), 1966, Birmingham University, U.K.; Ph.D. (Electronic and Electrical Engineering),

1970, Loughborough University of Technology, U.K. Loughborough University, U.K.; Central Research Laboratories of EMI Limited; Aston University, U.K.; AT&T Bell Laboratories, 1982–1984; Essex University, 1984—. Mr. Ackroyd is now the Professor of Telecommunication and Information Systems in the Department of Electrical Engineering Science at Essex University, U.K. At AT&T Bell Laboratories, he was in the Performance Analysis Department of the Network Planning Division. His research interests include the numerical solution of problems in information system performance engineering. Previously he has done research in other areas, including digital signal and image processing. Senior member, IEEE.