STATISTICAL AND PATTERN BASED MODELS FOR CPU BURST PREDICTION

G. L. Lasseter, T. Lo, K. M. Chandy and J. C. Browne
Department of Computer Sciences
The University of Texas
Austin, Texas

TR-28

Abstract

CPU burst durations for several jobs were measured. The CPU bursts exhibited patterns over time. A partially observable Markov process was used to model CPU burst generation. The model seems to fit data reasonably well. A simulation study on general prediction schemes was carried out. A job's predicted burst was made to fall within a confidence region with a certain probability. The effect of the length of the confidence region and the probability of predicting within that region were studied. The device utilizations were remarkably insensitive to predictive accuracy. A pattern based predictor was developed and it yielded satisfactory schedules.

1. INTRODUCTION

The basic goals of multiprogramming scheduling systems are to maximize overlap of device usage, e.g. CPU I/O channels, and minimize mutual wait or queueing, e.g. CPU waiting for I/O completion before being able to proceed. The basic assumption on relative device speeds is that CPU processing has a higher basic unit rate than I/O processing and that a substantial level of multiprogramming is sustainable in the available memory. To obtain the goals of multiprogramming under these assumptions requires a scheduling discipline which gives priority to those tasks whose CPU service requirements between I/O activities (assuming that there is no correlation between length of I/O burst and length of CPU burst) is smallest. Conversely given these goals and assumptions giving priority to the task with the largest CPU service requirement between I/O activities will be the least productive. We are thus led to the problem of determining (or predicting) the CPU service duration (CPU burst) of the tasks executing in a computer system. A complete knowledge of the CPU burst would (in principle at least) allow the construction of an optimal CPU scheduler. Such a complete knowledge (in advance) of the CPU utilization pattern is of course not obtainable in any realistic environment. In addition if such information were available its sheer volume would probably render its use totally intractable. We are thus led to the study of simple predictive models and algorithms together with an analysis of the effects of the inevitable inaccuracies and unreliabilities of these predictions.

This paper concerns itself with recognizing and predicting the CPU burst patterns present in actual burst data recorded on a production system, the modelling of this data and with the sensitivity of predictive CPU scheduling algorithms to the reliability and accuracy of the predictive mechanism.

The principal results obtained are: (a)Most of the jobs analyzed had observable "patterns" in the duration of their CPU bursts. A partially observable Markov process was used to model this behavior. Our model seems to fit the data reasonably well. Furthermore the existing theory on partially observable Markov processes [5] can be brought to bear on the problem. Our model can be easily imbedded in queueing network models of computer systems [4] resulting in a more accurate abstraction of program behavior in computer systems. (b)Standard measures of computer system performance such as CPU utilization and throughput were found to be remarkably insensitive to variations in predictive algorithm and reliability. This result extends the work of Sherman, Baskett and Browne [1] and explains why almost any predictive scheme for CPU bursts enjoys a sizeable advantage over first-come-first-serve or random schemes. A pattern based predictor derived from the data analysis was found to be a near optimal scheduler.

2. OBSERVATION AND MODELLING OF CPU BURST PATTERNS

2.1 Pattern Analysis

CPU burst patterns were obtained for some 400 distinct jobs from analysis of trace tapes on the UT-2D operating system for the CDC 6600 at the Computation Center of The University of Texas at Austin.

Many jobs exhibited patterns in their CPU bursts. For instance, some jobs would have a sequence of several relatively short bursts separated by single bursts of relatively long duration. The probability mass function (PMF) on the number of short bursts between long ones often showed marked peaks (Fig.1). The PMF for the number of short bursts between long would be geometric ("memoryless") if the duration of successive bursts were independent; this PMF was generally not geometric. It was therefore necessary to develop models which did not assume the independence of successive CPU bursts. Such models are clearly required for analytical studies of CPU burst prediction. A model on the prediction of program behavior should preferably have two characteristics. (1)It should be based on the existing theory of program behavior and (2)It should draw upon the theory of signal prediction in the presence of noise [2].

The principle of locality states that programs have several localities and transit from locality to locality in the course of execution. It might be hypothesized that the duration of a job's CPU burst is a function of its current localities. In this case a job may have one CPU burst distribution in one locality and may change its distribution when it enters another locality. We postulate that a job has "modes" (similar to localities); a job can be in only one mode at a time and can transit from one mode to another only at the end of a CPU burst. Note that a job can change localities continuously. Restricting jobs to change modes only at the end of CPU bursts results in simpler models. A job may have different CPU burst distributions in different modes. We shall make the simplifying assumption that jobs transit between modes in a Markovian manner. Hence the number of consecutive CPU bursts generated in a single mode would be a geometric random variable. A job's CPU bursts now consist of a concatenation of strings of bursts from different modes where each string of bursts has its own distribution for burst duration. The process is only partially observable because the modes cannot be observed directly; the current mode must be inferred from the history of observed CPU bursts. The problem of prediction reduces to that of inferring the next mode. It must be emphasized that though the concept of modes was based on localities, no attempt was made to determine localities from traces of memory references because such traces were not readily available.

Two other characteristics of CPU bursts are of interest. Firstly, a job's average CPU burst duration is not generally indicative of the majority of its bursts which are generally under 16 msec regardless of their means. Thus prediction schemes which attempt to predict a job's true average burst duration are not very good. Secondly, there were a number of cases where runs of short bursts were separated by single bursts of relatively long duration. In a significant number of cases, the occurrence of a very long burst indicated that the following burst would be relatively short. In these cases the exponential smoothing predictor did not behave satisfactorily. It was hoped that the partially observable Markov process would capture this behavior and prove useful in prediction and scheduling.

A Markov diagram of a six-state model of a typical job is shown in Fig. 2. This diagram was obtained by analyzing the trace tape. At this time, we do not have an algorithm which will automatically construct the model from the trace. The process of obtaining the model is one of trial and error. This is one of the serious defects of the model as it now stands. A short burst was taken to be one which required less than one CPU quantum (eight milliseconds), while a long burst was one which required more than one quantum. The distribution of long and short bursts was obtained directly from the trace. The bursts generated by the job were divided into three groups of four hundred consecutive bursts each; the three groups are labeled a, b and c. The model was based on group a and then compared with b and c. Fig. 1 shows the PMF for the run of successive short bursts for our model, a model assuming independence, and group a. Fig. 3 compares the model with b and c. The model was compared to the data by comparing four sets of statistics:

a) the PMF for the run of short bursts between long
b) the auto correlation coefficients for burst durations
c) the autocorrelation coefficient for successive runs of short bursts and
d) the mean square error made by moving point average and exponential smoothing predictors.

Seven jobs were picked at random out of a trace tape containing four hundred jobs. Our results show that the relation between the partially observable Markov process model and group (a) is very good; but groups (a), (b) and (c) are somewhat different and hence the model fits groups (b) and (c) less satisfactorily [3].

It is interesting to study the effect of the length of a run of short bursts on the behavior of a system. Consider a two mode model for CPU bursts where mode 1 generates short bursts and mode 2 generates long bursts (Fig. 4). Let p be the probability that the job changes mode after a given burst. Clearly the number of bursts generated in mode 1 will be the same as in mode 2 for all values of p. If p=0.5 then successive bursts are independent. When p is very nearly one there will be long strings of bursts of either duration. Let us imbed this two mode model into the central server model [4]. In the modified central server model, successive CPU bursts are no longer independent but depend on the job's current mode. It is easy to show that the equilibrium behavior of the system is invariant under p, but that the transient behavior is very dependent on p. Patterns of successive bursts based on the partially observable Markov process model have no effect on equilibrium utilization and throughput. The transient utilization as a function of time for different values of p was obtained by simulation and is shown in Figs. 5(a) and (b).

2.2 Pattern Predictor

We next attempt to use the patterns in prediction.

Our predictive schemes had a very severe restriction; they had to operate in real-time. Existing theory on mode prediction from a sequence of signals in a partially observable Markov process is quite complex and cannot be used [5]. Furthermore the partially observable Markov process would have to be created on the fly. We were therefore forced to use extremely simplistic schemes. In the pattern predictor, the number of short bursts n between the last two long bursts is determined. We then predict n more consecutive short bursts before the next long burst. Note that the pattern predictor is completely different in its approach from the exponential smoothing and moving point average schemes. The pattern predictor attempts to predict the length of successive runs of short bursts rather than the duration of any given burst. A variation of this scheme is to predict a long burst either after n-1, or n, or n+1 successive short bursts where n is the length of the last run of short bursts.

Another method used was to always predict short bursts unless there were two consecutive long bursts. If there are two consecutive long bursts then we always predict long bursts until the next short one. This scheme always predicts one kind of burst unless it is switched into predicting the other kind; the switch from a prediction of short to long occurs when there are two consecutive long bursts and the switch from a prediction of long to short occurs at the first occurence of a short burst. We refer to this scheme as a continuous long/short burst predictor. This predictor was based on the observation that the majority of bursts of bursts were short, and that there were many cases in which runs of short bursts were separated by single long bursts whereas there were very few cases where runs of short bursts were separated by two consecutive long bursts. We also studied exponential smoothing, moving point average, and average burst predictors. Sherman, Baskett and Browne [1] have done a thorough study on the exponential smoothing and average burst predictors.

The five prediction schemes were compared for one hundred and twenty three jobs taken at random from a trace tape. The jobs together generated twenty-five thousand bursts. The analysis shows that the pattern predictive scheme has the highest matching percentage of long and short bursts, i.e. it recognized bursts to be short or long more accurately than the other methods. The pattern predictive scheme does give a higher root mean square error, which means that this scheme usually gives moderately accurate predictions but occasionally gives extremely inaccurate predictions. As will be seen in the next section, the pattern predictive scheme can be coupled with a bounding mechanism to yield a reasonable scheduling technique. This suggests that in circumstances where a job is to be run very many times it may be desirable to establish a partially observable Markov process model for each job and use the sophisticated scheduling techniques developed for such processes. The five methods are compared in Table 1. Details are in Lo [3].

3. THE EFFECTS OF UNCERTAINTY IN CPU BURSTS PREDICTION ON SYSTEM PERFORMANCE

It is clear that complete deterministic knowledge of the CPU bursts patterns of all the processes in a large computer installation is not possible. If it were possible, the usage or utilization of this information would be intractable. Therefore, an important if not dominating criteria for the selection of a scheduling algorithm for a CPU is its tolerance for inaccuracies or its "robustness". In the case of predictive algorithms the essential characteristic is the sensitivity of the performance of the algorithm to the accuracy and reliability of the predictor. It is convenient to define special measures for accuracy and reliability in this context. Reliability will be taken to imply the consistency with which a predictor can make qualitative decisions, for example whether a subsequent burst is "long" or "short" ("long" and "short" being perhaps arbitrarily predefined). Accuracy will be taken to be a measure of the quantitative comparison of each prediction to the actual burst.

There was designed and implemented a simulation model of a computer system. The model is represented as a queueing network in Fig. 6. The service times of the CPU were taken to have a mean of 40 milliseconds with both exponential and hypoexponential variances. The I/O devices were given exponentially distributed service times of approximately 100 milliseconds and a skewed probability of utilization. The degree of multiprogramming in the model was fixed at 5.

The variability factor in the model was the CPU scheduler. In particular, the properties of a predictive shortest burst first (PSBF) algorithm were varied. The functional nature of the algorithm was not of concern. Rather, the predicted bursts were generated with given fractional probability (confidence) of falling within a specified region (confidence region) around the true burst. The fractional probability (or confidence limit) of a predicted set of bursts is a measure of the reliability of the predictor. The width of the specified region of variation about the true burst is a constraint on the inaccuracy and thus directly on the average accuracy of the predictor. For comparison FCFS, RR true shortest burst first (TSBR) and true longest burst first (LBF) algorithms were also evaluated.

The simulator was evaluated with sets of confidence limits varying from 0.9 to 0.3 and with allowed error limits ranging from 25% to 500%. The simulator runs were extended until 4000 CPU bursts were accomplished. Three such runs with different random variable kernels were taken and the average of these is quoted as the given performance metric. For the sake of ready comparison we will restrict ourselves here to using CPU utilization as a performance metric. Full details can be found in Lasseter [6].

The results of these simulations show a remarkable insensitivity to variations in the confidence limits and/or the confidence regions of the predictor algorithms.

Table 2 shows the CPU utilization for several cases of PSBF algorithms using extreme values of the confidence limits and confidence regions and compares them to true shortest-burst-first, small quantum round-robin, FCFS, and longest burst first algorithms. There is a total 10.3% variation between the extreme cases of true shortest burst first and longest burst first. A larger total spread can be obtained by increasing the variance of the hyperexponential distribution of CPU bursts. Partial runs with a variance of 10 produced an additional spread of some 3%. However, longer runs of the simulator are required in this case. A predictive shortest burst first with a confidence region of 25% and a confidence limit of 0.9 differs from the optimal result of true shortest burst first only 0.2%. The worst predictor used in this study with a confidence limit of only 0.3 and allowed variation of 500% from the true burst still yielded a CPU utilization of 67.4%, only marginally different from that obtained with the short-quantum round-robin. Similar conclusions result from different CPU burst distributions. Clearly a predictive scheme with virtually any level of accuracy is markedly superior to a FCFS or random scheduler.

Both linear and nonlinear factor analyses were carried out on the matrix of CPU utilization for varying confidence limits and confidence regions. The results showed that the simulation model is somewhat more sensitive to changes in the confidence level than to the width of the confidence region. This suggests that it is more important to have a predictor which is reasonably reliable rather than one of a high absolute accuracy.

It seemed desirable to test the utility of the pattern predictor discussed in Section 2 with respect to performance in a total computer system environment. Therefore, the simulator of this section was modified to operate on the CPU burst data extracted from the trace tape. Again the simulator was run for approximately 4000 bursts with the original pattern predictor as given, then a pattern predictor modified to have a 250 millisecond upper bound on the amount of CPU time which could be consumed by any one job in any one burst. The imposition of such an upper bound is characteristic of circumstances where very long CPU bursts may occur. The effects of bounding have been studied by Sherman, Baskett, and Browne [7] on predictive algorithms. Other predictive algorithms have been studied by Sherman, Baskett, and Browne [1]. Table 3 shows the results in terms of CPU utilization for true shortest burst first, predictive shortest burst first without bound, 2 round-robin schedulers with short quantum, first-come-first-serve, and a bounded predictive shortest burst first. The predictive shortest burst first in fact is taken to be the pattern predictor. The pattern predictor merely predicts either that a burst is short or long. Preference is then given to those jobs for which are predicted short bursts with conflicts being resolved in favor of that task which has the earliest previously completed quantum being given the CPU.

There is approximately a 10% variation between true shortest burst first and FCFS. The unbounded predictor, while it is better than FCFS at 42% is markedly inferior to the round-robin scheduler. However, addition of the bounding procedure produces a predictor which is comparable in quality to the round-robin and less than 1-1/2% in accuracy from the optimal true shortest burst first.

This result clearly shows that merely the specification of whether a burst is long or short is adequate to return virtually all of the benefit which can be obtained from a predictor (assuming that the predictive mechanism has a bounding mechanism). We note that the predictive scheduler made approximately 25% incorrect decisions. That is 25% of the time it predicted a long or short burst when the reverse was true.

4. SUMMARY

1. A partially observable Markov process was used to model CPU burst generation. The model is based on the principle of locality. The model seems to fit the data reasonably well and it can be imbedded into queueing network models of computer systems. It is hoped that the theory of partially observable Markov processes can be applied to the analysis of program behavior by using our model.

The time patterns of bursts generated by the partially observable Markov processes do not affect equilibrium behavior though it greatly affects transient behavior.

The duration of bursts generated by a single job often do not vary slowly over time; changes in burst duration are often abrupt.

2. Predictive CPU schedulers are insensitive to the quality of predictions. The ability to correctly classify bursts as long or short is more important than absolute accuracy. The simple pattern predictor of long or short bursts is adequate since it returns 90% of the benefit which can be obtained from predictive scheduling.

Note:- This research was supported by NSF Grants GJ-1084 and GJ-35109.

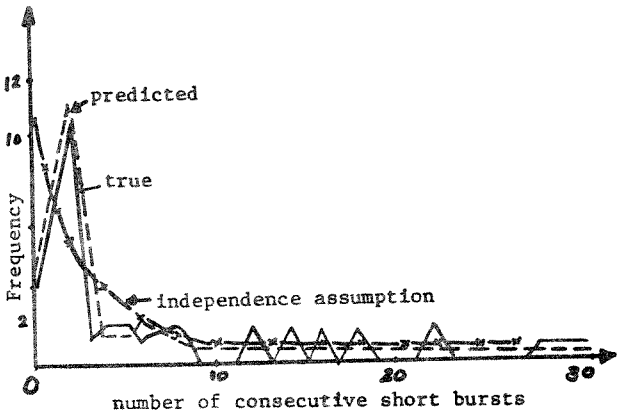

Figure 1. Comparison of data and models

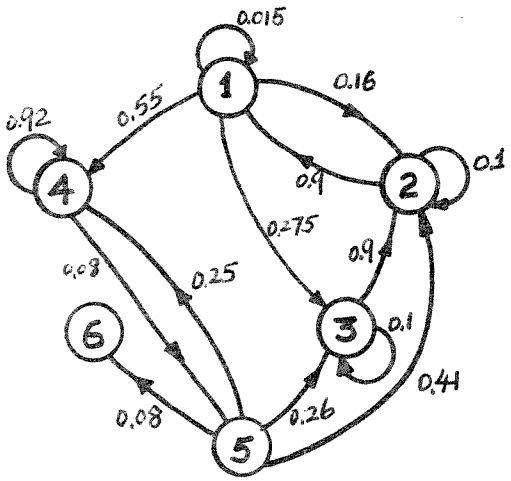

Figure 2: Markov model for a typical job

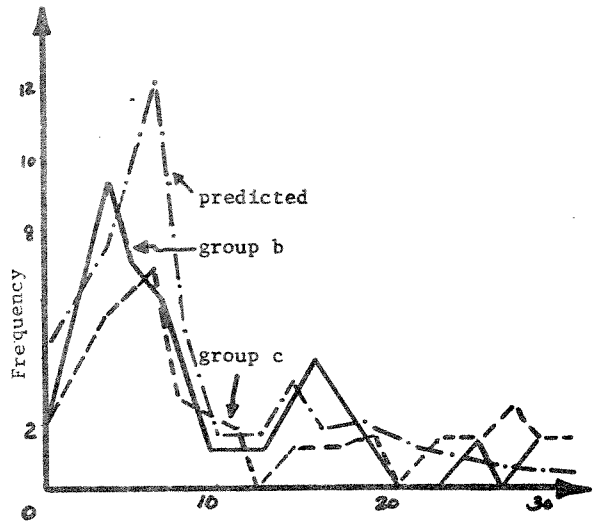

Figure 3: Comparison of data groups b,c and model

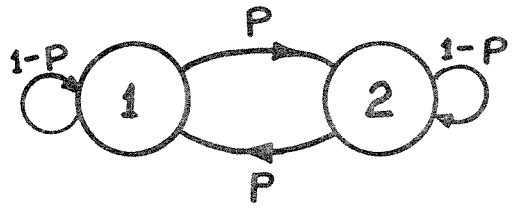

Figure 4. A Simple Markov Model

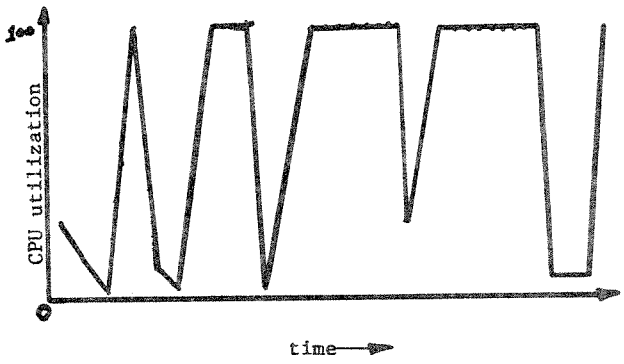

Figure 5 (a) p = 0.9, transient behavior

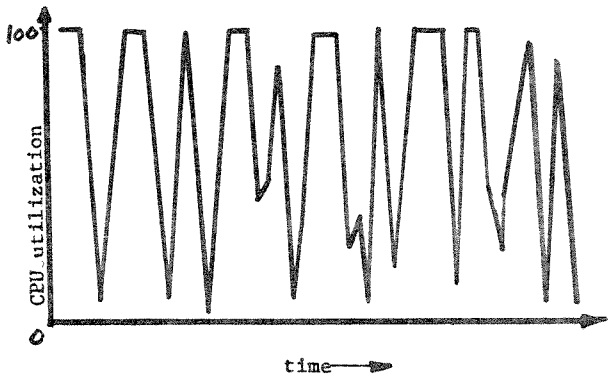

Figure 5 (b) p = 0.5, transient behavior

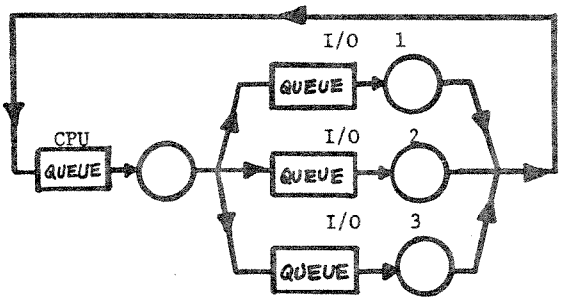

Figure 6: Simulation Model

method / data term	EXPONENTIAL	MOVING POINT AVG.	PATTERN	CONT.S/L	AVERAGE BURST
Average Absolute error	32.9697	30.7692	26.9625	26.002	33.7148
Average error squared	163.517	174.073	186.872	162.71	164.437
Matching %	67.661	72.365	77.664	69.387	62.661

TABLE 1. COMPARISONS OF PREDICTIVE METHODS

Scheduling discipline / Measure	%CPU Utilization	Throughput (Jobs/second)	CPU Switches (1000s)
Preemptive SBF	71.5	1.31	82.0
PSBF, CL = 0.9, VAR = 25%	70.8	1.28	80.6
VAR = 100%	68.8	1.25	77.5
VAR = 300%	67.3	1.21	76.1
RR, Q = 8	67.1	1.22	217.2
Q = 16	67.0	1.20	134.8
LCFS	66.7	1.21	91.4
FCFS	62.5	1.14	56.0
LBF, Q =32	59.7	1.09	103.9

TABLE 2. PERFORMANCE MEASURES OF THE MODELS WITH HYPEREXPONENTIALLY DISTRIBUTED CP BURSTS (σ^2 = 10 X MEAN)

Scheduling discipline	% CPU Utilization
True SBF	48.7
Predictive SBF	42.8
RR, Q = 8	47.9
RR, Q = 16	48.5
FCFS	38.0
Bounded PSBF	47.4

TABLE 3. CPU UTILIZATIONS OF THE MODELS

REFERENCES

1. S. Sherman, F. Baskett, and J. C. Browne, "Trace driven modeling and analysis of CPU scheduling in a multiprogramming system," Proc. SIGOPS Workshop on System Performance and Evaluation, Harvard Univ., (April, 1971) pp. 173-191.
2. Y.W. Lee, "Statistical Theory of Communication, John Wiley, New York (1966).
3. T. Lo, "Computer aids in computer systems analysis," Ph.D. dissertation, Computer Sciences, University of Texas at Austin, Austin, (1973).
4. J Buzen, "Queueing network models of multiprogramming," Ph.D. dissertation, Div. Engineering & Applied Physics, Harvard Univ., Camb., (1971).
5. J.S. Kakalik, "Optimum policies for partially observable Markov systems," MIT Operations Research Center Report No. TR-18., MIT, Cambridge, Mass (1965).
6. G. Lasseter, "A model of predictive CPU scheduling of known uncertainty," Master's thesis, University of Texas at Austin (1972).
7. S. Sherman, F. Baskett, and J. C. Browne, "Trace driven modeling and analysis of CPU scheduling in a multiprogramming system," Comm., ACM , Vol. 15, No. 12, pp. 1063-1069.

Gene Lasseter was born March 22, 1947, in Fort Worth, Texas. He graduated from Castelberry High School, in Fort Worth. He attended the University of Texas at Austin where he received the Bachelor of Arts degree in Mathematics. He received the Master of Arts degree in Computer Sciences in December, 1972, at the University of Texas, Austin. He is currently employed at Information Research Associates, Austin, Texas.

T. C. Lo was born in Chungking, China, on January 30, 1940. He received the B.S. degree from the Chinese Military Academy, Taiwan, in 1962, the M.S. degree in Mechanical Engineering, and the Ph.D. degree in Computer Sciences from the University of Texas at Austin, in 1967 and 1973, respectively. From 1967 to 1969 he worked as an instructor of Mechanical Engineering at the Chinese Military Academy. He is currently a member of the Technical Staff in the Information Services Division, Texas Instruments, Dallas, Texas.

K.M. Chandy was born in India on October 25, 1944. He received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1966, the M.S. degree in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, N.Y., in 1967, and the Ph.D. degree in Operations Research from the Massachusetts Institute of Technology, Cambridge, in 1969.

He has worked for Honeywell EDP and the IBM Cambridge Scientific Center. He is currently an Associate Professor of Computer Science, University of Texas, Austin.

Dr. Chandy is a member of the Association for Computing Machinery.

James C. Browne was born in Conway, Ark., on January 16, 1935. He received the B.A. degree from Hendrix College, Conway, Ark., in 1956 and the PhD degree in physical chemistry from the University of Texas, Austin, in 1960. From 1960 to 1964 he worked as an Assistant Professor of Physics,at the University of Texas. From 1964-1965 he was at the Queen's University of Belfast, Belfast, Northern Ireland on an NSF Postdoctoral Fellowship. From 1965-1968 he was a Professor of Computer Science and Director of the Computation Laboratory at the Queen's University of Belfast. In 1968 he rejoined the University of Texas as Professor of Computer Science and Physics and is currently Chairman of the Department of Computer Sciences.

Dr. Browne is a Fellow of the American Physical Society and the British Computer Society and a member of the Association for Computing Machinery.