PREEMPTION COSTS IN ROUND ROBIN SCHEDULING

Charles M. Shub

Computer Science Department
University of Wyoming
Box 3682, University Station
Laramie, Wyoming 82071

Round robin scheduling with preemption costs taken into consideration is examined. Both a uniprocessor configuration and a multiprocessor configuration, one processor dedicated to the scheduling, are considered. Approximation formulae to obtain effective load on the system from the actual load and the overhead parameters are derived and compared with simulation results.

Key words: Analytical model; round robin scheduling; scheduling overhead; simulation model.

1. Introduction

Coffman and Denning [5] call for treatment of optimal scheduling with preemption costs taken into consideration. A more general problem is the effect of scheduler overhead in computer systems as it relates not only to optimal processor scheduling but also to the net effects of certain peripheral scheduling algorithms. In 1970, Mullery and Driscoll [7] observed that one method of minimizing overhead was to leave each job on the processor as long as possible. They reasoned that the smaller the number of scheduling operations, the less overhead. This idea was echoed by Bernstein and Sharp in 1971 [3]. Earlier, Coffman [4] investigated overhead in switching between foreground and background using a fixed delay. More recently, Babad [2] reported results again involving a fixed constant amount of overhead.

In terms of scheduling, there are many unanswered questions. For example, what is the overall effect of a disc scheduling algorithm which maximizes data transfer rate? If the implementation of this disc scheduling algorithm requires more processing to handle disc requests than some other algorithm, is there an implication that the additional load on the processor due to the use of this particular algorithm will cause a decrease in overall throughput? Is there a point at which a processor can become so overloaded with the business of scheduling operations that system performance collapses in a manner similar to the system performance collapse due to "thrashing" [5]? Is there a point at which the increased overhead to manage additional peripheral units offsets the performance gains which those additional peripheral units should have provided?

A study at the University of Texas [9] claims that the best overall CPU throughout is obtained by a shortest execution first processor scheduling policy and presents results showing a 12.89 percent throughput increase over the worst case scheduling method. This work does not take into account the possibly significant processor overhead involved in making such a selection.

The purpose of this study is to report on the investigation of, through simulation modeling, the effect of accounting for the finite amount of time that the scheduling operations take. This study is an extension to a prior study done on an M/M/1 queueing system [11]. This study extends the work previously reported to a Round Robin environment with a fixed quantum length given a Poisson arrival process and exponentially distributed service requirements.

2. Methodology

The methodology used is perhaps unusual and merits exposition. Typically, a simulation experiment involves a number of phases including strategic planning, tactical planning, experimentation, validation of the experimental results and analysis of the valid results [8]. In terms of the strategic, or long-range plans, the goal is to develop a thorough understanding of the perhaps subtle effects that different scheduling algorithms can have on overall performance of computer systems when their costs are accurately taken into account. Normally, an experiment involving the exploration of one situation would be considered to be part of the tactical plan. However, in terms of this set

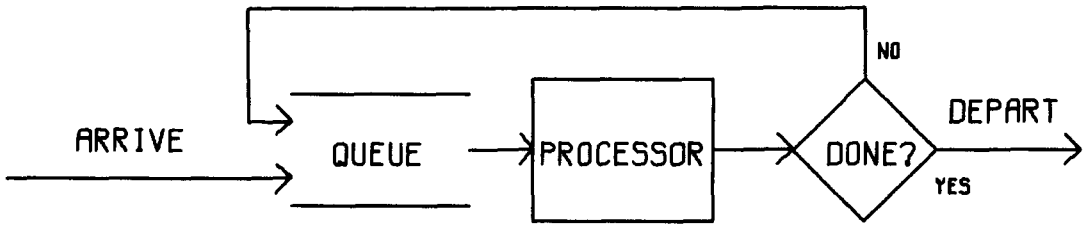


FIGURE 1. SYSTEM DIAGRAM

of experiments being one phase of the solution to the overall problem, the choice of what system to consider becomes part of the long-range plans.

The tactical planning, then, involves choosing the parameters and values to produce the desired information. Considerable detail to motivate the choices made in the tactical plan is presented in that section of the report. In general, areas of concern include fairly heavily loaded systems, so the general parameter selection process chooses a fairly heavy loading factor when overhead is not considered and then selection of overhead parameters which will yield a range of values before driving the system to total saturation.

The experimentation, then, involves one or both of two courses of action. The first, and most usual course of action, is to run the experiments on an already developed computer implementation of the model. The second involves developing a separate computer implementation for the particular experiments. In the set of experiments reported in this paper both approaches were used. The first has as its advantage the ease of carrying out the experiments and confidence in the results, but as its disadvantage, the problem that the original model may not be totally appropriate and the development of a specific model may provide additional insight to the process. Also, in a teaching environment, development of a new model can be valuable for student participation.

The validation process is normally quite complex because of the lack of analytical models which consider the scheduling delays. One can avoid the problem by validating at the no-overhead point. However, much more effective is the mechanism of analysis and actual development of analytical approximations through a thorough analysis. Once developed, the analytical approximation is, of course, not only much easier to apply but also cheaper than the simulation model. Thus, the methodology used is the development of a model, the development of a computer simulation of the model, the development of analytical techniques to validate the simulation model, and, finally, the use of the analytical results to get approximations quickly and cheaply.

The natural question at this point is why bother with the simulation in the first place. Typically, analytical modeling requires making assumptions about the stochastic form of interactions within the system. The simulation experiment can either show that this form does in fact exist or that if the form does not exist, its lack of existence is not critical to the analysis. In addition, the simulation results can dramatically show counter-intuitive interactions which the pure analyst will neither expect nor take into account.

In summary, then, the methodology is to a large degree a hybrid methodology blending both simulation and analytical techniques to obtain the desired results which are easy and cheap to use.

3. The Idealized Model

The idealized queueing situation described below evolves naturally by abstraction from a typical multiprogrammed computer system. A major simplifying assumption is that in the idealized system, peripheral usage is of peripheral interest and is thus ignored. With such an abstraction, the system can be depicted in Figure 1.

A user arrives at the system requiring an exponentially distributed service time. If the system is idle, the user receives control of the processor. Otherwise, the user enters a waiting line for the processor. Upon receiving control of the processor, the user utilizes the processor for a fixed quantum length and then goes to the end of the waiting line. Should a user complete his service requirement before ending a quantum, the user would then release the processor when he is done and depart. As has been described, the system is the classic Round Robin model. This classic model is modified by including a finite amount of time for effecting the queueing changes, and a finite amount of time to process departures. Adding these processing delays causes the idealized model to conform more closely to an actual system than the classic Round Robin model. It also transforms the model into a system for which there is not a closed form analytical solution.

Consider, as an example, a service facility which services inoperable automobiles. When an inoperable automobile arrives at the facility, some processor at the facility must expend some effort in receiving the inoperable automobile for service. This effort could involve placing the automobile in a line or a position to be worked on. The mechanic (server) will work on an automobile for at most a

fixed time (quantum) stopping before the expiration of the quantum only if the car is completely repaired prior to the end of the quantum. After completing his quantum of work on a car, the mechanic then proceeds to the next car. There will, of course, be some time involved in the mechanic transferring from one car to the next as well as time involved placing the inoperable, but partially repaired, car in a status to be worked up later. The operations of receiving a car for service and placing the car in a position for later resumption of work can be done either by a supervisor (separate resource) or by the mechanic (server) himself.

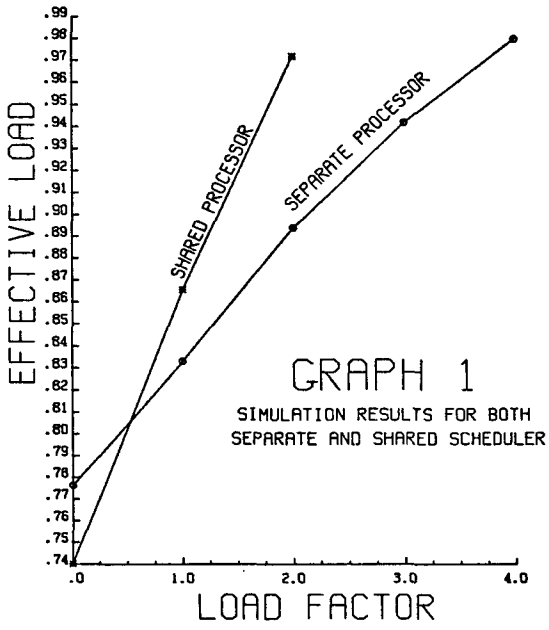
The addition of this overhead can be considered in two fashions. The first involves the server stopping production to perform the scheduling. This is the situation currently common in most multiprogramming computer systems in use today. The second, or innovative method, involves the use of a second processor to perform the scheduling operations. Intuitively, the second processor might be a very small mini or micro processor designed only to do the queue management operations involved in scheduling. It would, naturally, run concurrently with the main processor. The advantage of this method would be that the processor could do productive work without being bothered with performing the scheduling operations.

4. Simulation Experiments

Several simulation experiments were run using the idealized models described above. The simulation runs were made using a previously validated simulation model [10]. In each and every experiment, arrivals were according to a Poisson process with mean inter-arrival time set at 1.0 units. The mean service requirement was exponentially distributed with a mean of 0.75 units, and the maximum quantum length was established as 0.25 units. In addition, the system was allowed an initial bias period of 125 units, and then statistics were gathered for a period of 500 time units. Careful consideration was given to these choices. The choice of 1 unit for the mean inter-arrival time gives a (lambda) parameter of 1 with mean and variance both unity. This has the net effect of removing all (lambda) multipliers from the performance equations. The mean service requirement was chosen as .75 to provide a fairly substantial loading such that there would be a buildup in the system and equilibrium conditions would involve reasonable queue lengths. In addition, the loading had to be low enough to allow for an introduction of scheduling delays. As a basis for comparison, a M/M/1 system with identical loading would have the following performance parameters [1]; these values are summarized in table 1:

Expected Waiting Time	2.25
Expected Queue Lenght	2.25
Expected Length of Nonempty	4
Expected Number in System	3
Expected Wait Given Waiting	3
Expected Time in System	3

Table 1. Expected Results



Graph 1 provides a summary of the simulation experiments showing the differences between using a separate processor for scheduling and using the same processor for both scheduling and productive work.

5. Round Robin with Overhead

Coffman and Denning [5] provide a detailed analysis of Round Robin systems with no overhead. In their analysis, they assume an integral number of quantum lengths as the service requirement. This factor influenced the choice of .25 for the quantum length. With the selected quantum figure, the expected number of quanta would be 3, and the probability of a user requiring an additional quantum at the end of any quantum would be 0.75.

A base line experiment on the Round Robin model involving no overhead provided the results given in table 2 which compare nicely with the M/M/1 system and design considerations.

Mean Waiting Time	2.19
Mean Queue Length	2.276
Mean Length of Nonempty Queue	3.64
Mean Number in System	3.08
Mean Wait Given Waiting	2.48
Mean Time in System	2.99

Table 2. Observed Results

A batch means technique [6] was used to reduce the variance of the simulation results, and, in every case, the simulation results were statistically identical at the $\alpha = .05$ level of significance to the M/M/1 system predicted results.

The next experiment involved introducing scheduling delays as follows:

1) arrival at an idle system (AV) .026 x units
2) arrival at a productive system (AP) .031 x units
3) arrival at a system in overhead (AO) .021 x units
4) departure delay (DD) .025 x units
5) end of quantum delay (QD) .031 x units

These figures were chosen in what appears to be a rather arbitrary fashion so some justification is in order. Given a system in which the expected number of quanta is 3, one can expect, on the average, one arrival, one departure and two end of quantum delays. Each job would thus receive, on the average, four distinct scheduling services. Thus, to saturate the system, one would want total services (both production and scheduling) to take about 1 unit of time. With a .75 service requirement, about .25 is left for overhead processing. In actuality, a bit more than the .25 would be available because the end of quantum processing involves two users, the one ending the quantum and the one starting the quantum. The units of delay were chosen to attempt to reflect a total delay ascribable to each user of approximately about .11 units. That would yield, in a rather general sense, loads similar to .86 and .97 for x equals 1 and 2 respectively. A second, and concurrent, consideration is that of breaking down the delays into the primitive units as recognized by the simulation model. To establish the desired results given above, the following values were used:

1) .010 x interrupt processing time
2) .001 x time to check the queue to determine if a user was waiting for service
3) .005 x time to unlink task control block of waiting user so that he could receive the processor
4) .005 x time to link task control block into queue for start or resumption of processing later
5) .010 x time to restore processor to uninterrupted status

The ratios of these values are based upon actual instruction counts in an existing system.

6. Analysis of Round Robin Scheduling

$$\hat{\rho} = \rho + \text{arrival overhead} + (N - 1)\ \text{quantum end overhead} + \text{departure overhead} \quad (1)$$

$$\text{arrival overhead} = PO * AV + (1 - PO) * (PP * AP + (1 - PP)\ AO) \quad (2)$$

where

PO = probability the system is idle $(1 - \hat{\rho})$

PP = probability the system is productive (ρ)

AV = arrival overhead at vacant system

AP = arrival overhead at productive system

AO = additional arrival overhead at system in overhead

DD = departure delay

QD = end of quantum delay

$\hat{\rho}$ = effective load

Substituting into eq (1) gives

$$\hat{\rho} = \rho + (N - 1)\ QD + DD + (1 - \hat{\rho}) * AV + \hat{\rho}\ (\rho * AP + (1 - \rho)\ AO) \quad (3)$$

Solving eq (3) for $\hat{\rho}$ we find

$$\hat{\rho} = \frac{\rho + (N - 1)\ QD + DD + AV}{1 + AV + (\rho * AP + (1 - \rho) * AQ)} \quad (4)$$

Tables 3 and 4 summarize the results of these experiments. It can be seen that these results compare favorably with those expected. This indidates that the commonly advanced idea of, in essence, "charging" the user for a scheduling operation, while it might not be fair, is a good approximation for determining the net effect of finite scheduling time in the Round Robin situation.

The comparison of simulation results to analytical results is provided in graph 2.

PARAMETER	EXPECTED	OBSERVED
Expected Waiting Time	5.54	5.42
Expected Queue Length	5.54	6.28
Expected Length of Nonempty Queue	7.41	7.28
Expected Number in System	6.41	7.10
Expected Wait Given Waiting	6.41	5.66
Expected Time in System	6.41	6.28
Overhead Percentage	11.51	13.98
Effective Load	.8651	.8655

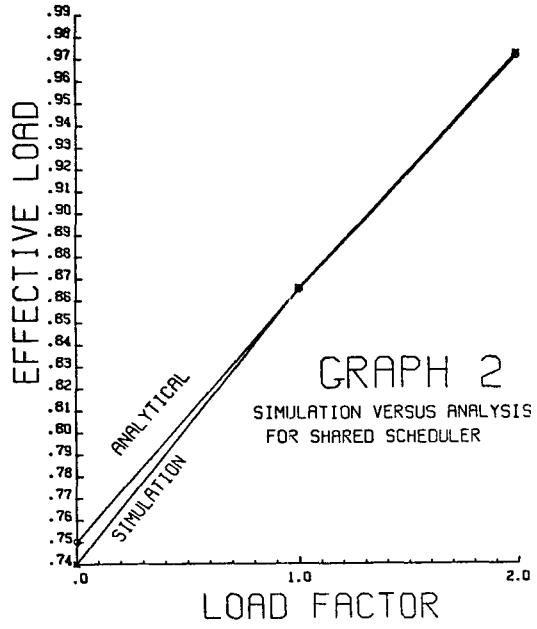
Table 3. Expected Versus Observed Parameters for Nominal Load of 0.865

PARAMETER	EXPECTED	OBSERVED
Expected Waiting Time	32.51	33.73
Expected Queue Length	32.51	32.32
Expected Length of Nonempty Queue	34.48	32.32
Expected Number in System	33.48	33.10
Expected Wait Given Waiting	33.48	33.73
Expected Time in System	33.48	34.32
Overhead Percentage	22.1	25.67
Effective Load	.971	.972

Table 4. Expected Versus Observed Parameters for Nominal Load of 0.971

7. Separate Processor

Of much greater interest and importance is the consideration of the use of an auxiliary



processor to perform scheduling operations. These operations, under a wide class of queueing disciplines, involve only simple priority calculations, some minor queue management routines and simple message sending operations [10]. There is no need for a complex modern large scale third or fourth generation processor to perform these tasks, especially when there is productive work which could otherwise occupy the processor while the scheduling is going on. In addition, the use of a small processor specifically designed to perform the scheduling would be conducive to implementing the scheduling operations with hardware rather than software.

Several simulation experiments were run with a separate processor called the scheduler to perform the scheduling operations. For comparison purposes, delay values and loading were chosen to be similar to the previous experiments.

The most interesting result is that the introduction of a separate processor to perform scheduling has little, if any, influence on performance at lower scheduling delays. At the higher delays there is a significant backup in the system due to the time the scheduling is taking. At these higher dalays, the scheduler is taking over 50 percent of the available time to do the scheduling, but the utilization statistics are no worse than with using the same processor with smaller delays for both productive work and scheduling. If the magnitude of the delay is interpreted as being due to a slower scheduling processor, we see that the scheduler can run at one-fourth the speed of the processor!

The major portion of the degradation in performance for high overhead (low speed) scheduling appears to be that scheduling has become a bottleneck. Almost 60 percent of the requests for scheduling find the scheduler busy. Thus, it is not the lack of a processor which is holding things up but the inability for requests for service to be acted upon. Also, the percentage of time that the processor is idle appears to increase with slower scheduling. This is due to the fact that the processor must sit idle while waiting for the scheduler to tell it which job to process next. There is a substantial synergistic effect from adding a scheduling device. The next section analyses the system in more detail.

8. Analysis of Separate Processor

There can be a wide range of events involved in this system because of the two processors involved. The initial step is to analyze each possible action which can occur and provide the timing data for that situation.

8.1. Job Arrival

There are three possible situations. The system can be idle when a job arrives (AV) or the scheduler can be busy (AB) or the processor can be busy and the scheduler is doing nothing (AI).

In the first case, the scheduler goes from idle mode to busy mode. This status will last for a duration of .025 x units of time. This is to create a task control block for the user and is consistent with the previous experiments. Since we have a processor in addition to the scheduler, the job will not have to wait the entire .025 x units in order to start being processed. The processor will remain idle for a period of .015 x units, experience an overhead time of .005 x units and then go into production. This is an unproductive period (UAV) of .020 x units.

In the second case, a job arrives while the scheduler is processing some other request. This arrival will have absolutely no effect upon productive use of the processor. The scheduler will spend an additional .015 x units processing this request.

The final arrival case involves an arrival when the scheduler is doing nothing but the processor is busy. In this situation, the arrival has no effect on the processing but does cause scheduler to spend .025 x units of time processing the arrival.

8.2. Job Departure

There are two possibilities when a job departs. The scheduler can be idle (DI) or the scheduler can be busy (DB). In the first case, the processor will go idle. If there is a request for service in the queue, the processor will remain idle for a period of .016 x units, then go to overhead mode for .005 x units and finally back into production. Thus, the processor will be unproductive (UDI) for a total of .021 x units. The scheduler, meanwhile, will spend a total of .026 x units of time processing the departure and scheduling the next user.

In the second case, we use D to denote the delay before the scheduler can process the departure due to being otherwise occupied. The processor will be idle for a period of D + .006 x units, and then in overhead for a total of .005 x units yielding a total nonproductive interval (UDB) for the processor of D + .011 x units. The scheduler

will require an additional .016 x units after the delay to complete the departure processing.

8.3. Quantum Expiration

At quantum termination, either the scheduler is idle (QI) or busy (QB). When the scheduler is idle, the processor goes idle, then to overhead mode .016 x units later and finally back to productive mode .005 x units after that. This is an enforced nonproductive period (UQI) of .021 x units due to quantum end. The scheduler takes a total of .031 x units of time to process the quantum expiration, actually queueing the request for the next quantum .021 x units after starting.

When the scheduler is busy, we use D to denote the delay before the scheduler can process the quantum expiration. The processor immediately goes idle. It remains idle for D + .006 x units and then goes into overhead for .005 x units before going back to production. This is a nonproductive period (UQB) of D + .011 x units. The scheduler, meanwhile, requires an additional .021 x units of time to process the end of the quantum, actually queueing the request for another quantum .011 x units after starting.

8.4. Timing Data

From the above analysis, expressions can be derived for the magnitudes of the overhead values. The following notation is used:

AO	arrival overhead
QO	quantum expiration overhead
DO	departure overhead
UQ	unproductive interval on quantum expiration
UD	unproductive interval on departure
PO	probability the system is idle
PSI	probability the scheduler is idle
PPA	probability the scheduler is processing an arrival
N	expected number of quanta a job requires
MO	mean event overhead
MU	mean unproductive interval
ESR	effective service requirement
QL	quantum length

The first step in the analysis is to determine the mean length of scheduling operation. This, coupled with the unity expected inter-arrival time, allows the calculation of the probability that the scheduler is idle. The timing data mentioned above can be used to solve the equations.

$$MO = ((N - 1) * QO + DO + AO)/(N + 1) \quad (5)$$

$$PSI = 1.0 - (N + 1) * MO \quad (6)$$

$$QO = PSI * QI + (1 - PSI) * (QB) \quad (7)$$

$$DO = PSI * DI + (1 - PSI) * (DB) \quad (8)$$

$$AO = PSI * AI + (1 - PSI) * (AB) \quad (9)$$

Some simple mathematics yields eq (10) from eqs (5) through (9).

$$MO = \frac{(N - 1)\ QI + DI + AI}{(N+1)(1+(N-1)(QI-QB)+(DI-DB)+(AI-AB))} \quad (10)$$

Next, we calculate the average forced unproductive period, due to the overhead, at quantum expiration and departure.

$$MU = \frac{(N - 1) * UQ + UD}{N} \quad (11)$$

Note that a quantum expiration can occur when the scheduler is in overhead mode only when an arrival is being processed. In this case, the unproductive period will be longer than expected because the scheduler must finish processing the arrival before it can process the quantum expiration. Both the simulation results and the fact that arrivals are exponential indicate that this delay averages out to be 36.7879 percent (1/e) of the arrival overhead. Thus, we have

$$UQ = PPA * (UQB + AO/e) + (1 - PPA)(UQI) \quad (12)$$

$$UD = PSI(UDI) + (1 - PSI)(UDB + MO/e) \quad (13)$$

and MU can be easily calculated. Finally,

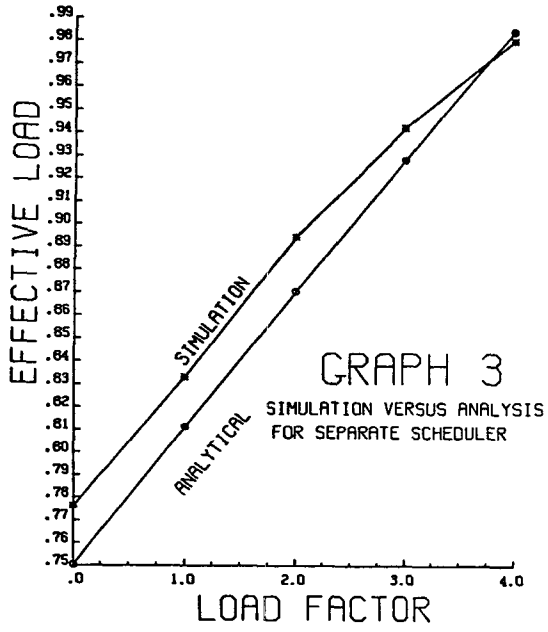
$$\hat{\rho} = \rho + N * MU. \quad (14)$$

Table 5 summarizes the comparison of the analysis and the simulation.

In all cases, the calculated $\hat{\rho}$ is within about 2 percent of the observed value of $\hat{\rho}$. At these high loadings, the measures are very sensitive to small changes in the loading. For example, in the last column a difference of less than 1/2 percent in calculated and observed loading produces a large disparity between the measures. This is depicted in graph 3 which compares the simulation results and analytical results.

9. Conclusions

The results presented above extend previous models for predicting the effect of scheduler overhead in computing systems from the single server single queue system to the Round Robin scheduling discipline. Performance calculation equations and procedures have been presented. These equations and procedures will allow the system designer to analyze the effect of adding bells and whistles to his scheduling algorithms. The analytical results derived have been validated against a valid simulation model. The effect of adding a separate processor to perform the scheduling operations has been investigated and shown to be less deleterious to system performance than the currently used mechanism. There is a need for future work in this area to extend the results to a wider variety of



DELAY FACTOR	0	1	2	3	4
Calculated $\hat{\rho}$	.75	.811	.870	.928	.984
Expected Waiting Time	2.25	3.48	5.82	11.96	60.51
Observed Waiting Time	2.69	4.14	7.48	15.27	49.74
Observed $\hat{\rho}$ Calculated from WT	.776	.832	.893	.942	.980
Expected Queue Length	2.25	3.48	5.82	11.96	60.51
Observed Queue Length	2.75	4.24	7.65	15.56	51.05
Observed $\hat{\rho}$ Calculated from QL	.779	.835	.895	.943	.981
Expected Time in System	3.0	4.29	6.69	12.89	61.5
Observed Time in System	3.42	4.96	8.41	16.32	50.93
Observed $\hat{\rho}$ Calculated from TS	.773	.832	.894	.942	.980

Table 5. Comparison of Calculation and Simulation Results for Various Scheduler Speeds

queueing disciplines. There is a further need to develop models to handle the effect of overhead in output scheduling as well. The methods presented here should be extended to produce these needed additional results.

References

1. . . . . ., Analysis of Some Queueing Models in Real Time Systems, IBM Document GF20-007-1.

2. Babad, J. M., A Generalized Multi-Entrance Time-Sharing Priority Queue, Journal of the A.C.M., Volume 22, No. 2, April 1975, pp. 231-248.

4. Bernstein, A. J. and J. C. Sharp, A Policy Driven Scheduler for a Time Sharing System, Communications of the A.C.M., Volume 14, No. 2, February 1971, pp. 74-78.

4. Coffman, Edward G., Jr., On the Trade-off Between Response and Preemptive Costs in a Foreground Background Computer Service Discipline, IEEE Transactions on Computers, Volume C 18, No. 10, October 1969, pp. 942-947.

5. Coffman, Edward G., Jr. and Peter J. Denning, Operating Systems Theory, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

6. Gordon, Geoffrey, System Simulation, (second edition), Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

7. Mullery, A. P. and G. C. Driscoll, A Processor Allocation Method for Time-Sharing, Communications of the A.C.M., Volume 13, No. 1, January 1970, pp. 10-14.

8. Shannon, Robert E., Systems Simulation: The Art and the Science, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

9. Sherman, Steven, Forest Baskett, III and J. C. Browne, Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System, Communications of the A.C.M., Volume 15, No. 12, December 1972, pp. 1063-1069.

10. Shub, Charles M. and William G. Bulgren, A Stable Time Independent Queue Model for Studying the Effects of Overhead in Computing Systems, Proceedings of the 1975 Summer Computer Simulation Conference, San Francisco, California, July 1975.

11. Shub, Charles M., Simulation Studies on the Effect of Overhead in Computing Systems, Proceedings of the Ninth Annual Simulation Symposium, Tampa, Florida, March 1976.