1975

An Endogenous Priority Model for Load Control in Combined Batch—Interactive Computer Systems

Carl E. Landwehr

Report Number:
75-157

AN ENDOGENOUS PRIORITY MODEL FOR LOAD CONTROL IN COMBINED BATCH - INTERACTIVE COMPUTER SYSTEMS

by

Carl E. Landwehr
Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

July 1975

CSD TR 157

Submitted to: International Symposium on Computer Performance Modelling, Measurement, and Evaluation
March 29 - 31, 1976

AN ENDOGENOUS PRIORITY MODEL FOR LOAD CONTROL IN COMBINED BATCH - INTERACTIVE COMPUTER SYSTEMS

by

Carl E. Landwehr
Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

July 1975

ABSTRACT

A relatively high level analytical model for computer systems serving both batch and interactive users is presented. The model is unusual in its employment of an endogenous priority scheme to represent a class of strategies for controlling service to the two types of customers. Numerical methods developed by V. L. Wallace are used to generate steady state probability distributions for the infinite state Markov chain formed by the model. Data from the Michigan Terminal System, which includes a load controlling mechanism of the type modelled, is used to validate the model. Finally, additional parameter studies indicate that the model reflects the dynamic behavior of such system in a reasonable way.

I. INTRODUCTION

An increasing number of computer systems provide both batch and interactive service to their users. In such systems, conflicts may arise between the two modes of service, since jobs of both types compete for the same set of resources. In particular, if the batch load is substantial, it can cause response times for interactive jobs to become intolerable. This effectively reduces the interactive-batch system to a batch-only one unless some control is placed on the load imposed by the batch subsystem. Conversely, if jobs in the interactive subsystem are given absolute priority, turnaround for batch jobs may become unacceptably high.

From the operating system's point of view, it is desirable that all jobs actually competing for the processor be treated equally, regardless of whether the computing request was initiated from an interactive terminal or a card reader. The actual resource requirements of jobs at a particular time provide a better basis for discriminating among them than their sources, since, for example, a heavily compute bound (or I/O bound) request may be initiated either interactively or via the batch stream. Nonetheless, it must be recognized that admitting a single additional batch job into competition for the processor will generally load the system much more heavily than admitting an additional interactive job: for the batch job, "think times" will be zero and input/output times will generally be shorter than for the interactive job. Moore [13] found that the load imposed by a single batch job was roughly equivalent to that of 5 to 15 terminal jobs. These points suggest that a reasonable way to balance service between interactive and batch jobs is to control the entry of jobs into the race for the processor at least partially on the basis of the source of the job as well as on the current level of performance of the system.

In some respects this type of control algorithm corresponds to manipulating the degree of multiprogramming in order to keep the system from becoming saturated. Previous work directed toward this end is primarily represented by the development of the working set policy (5, 6, 7) in which the degree of multiprogramming at a given time is controlled by the size of the balance set, that is, the set of jobs all of whose working sets will fit into real memory at a given time. Although a number of approximations to working set replacement policies have been implemented (8, 14, 16), currently available hardware makes precise measurements of working set size difficult. More recently, an analytical model which includes a control switch to regulate the degree of

multiprogramming has been developed (3, 4) and extended to include an adaptive control (1). This extended version was simulated, but no additional analytic results were presented. Both the working set and control switch models, however, allow only one class of jobs and both contain more low level system detail than the model investigated below.

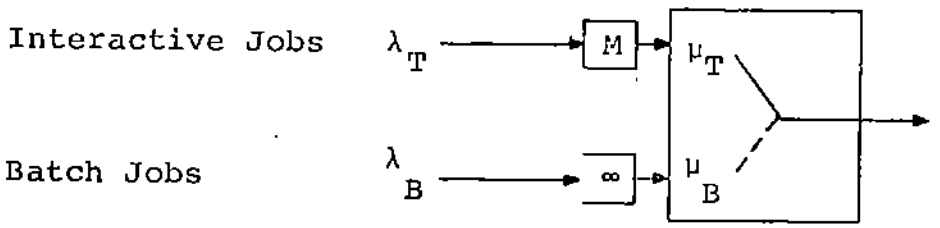
This paper presents a Markovian queueing model which allows two types of arrivals, representing batch and interactive jobs. The server discipline reflects a control algorithm such that good response to interactive requests is maintained while a minimal level of batch throughput is ensured. There is a strong emphasis on keeping the model simple and general, for purposes of wider applicability (10). Parameter values obtained from the Michigan Terminal System are used to conduct studies with the model, and the results of the studies are compared with actual system measurements.

II. STRUCTURE OF THE MODEL

The primary goal of this model is to represent a control algorithm for admitting jobs of two different types into the race for the processor, so the actual processor scheduling algorithm (that is, the algorithm for sharing the processor among the jobs which have been allowed to compete for it) will not be presented in detail. Admission of a job to the server in this model could thus correspond to the entrance of the job into the ready list of a more detailed model. All of the processing that takes place on a job after it enters the ready list will be represented here by a simple exponentially distributed service time. Since the control algorithm we will be concerned with generally maintains terminal response at the expense of batch turnaround time, the statistics of primary interest are the mean system residence time and queue length for batch jobs. Consequently, the assumption of a single server with service expontential for each job should not bias results unrealistically.

The basic structure of the model is shown in Figure 1. A finite capacity queue is used to represent the current number of terminals active in the system; the capacity of the queue can be thought of as corresponding to the (finite) number of input ports which the system supports. An infinite capacity queue is used to model batch jobs waiting for service. Since there are two queues and only one server, an algorithm is required to define from which queue the server chooses the next job to be serviced. In most queueing models, priorities are defined strictly by the type of the job (9): for example, all interactive

Single Node Model Structure



Batch Queue - infinite capacity

Maximum # terminals in system = M

Single Server, which has a state associated with it:

State	Meaning
idle	no jobs in system
0	serving batch job
1	serving 1st consecutive terminal
2	serving 2nd consecutive terminal
⋮	⋮ ⋮ ⋮
L-1	serving L-1st consecutive terminal

System State: (T,B,S)

Fig. 1 Single Host Model

jobs are serviced before all batch jobs, or vice versa. This type of priority discipline is called exogenous, since the order of service is defined strictly by the externally defined priorities of the jobs. The discipline used in this model, however, will be endogenous: the next job to be serviced will be determined on the basis of the state of the server, the lengths of the two queues, and a specified decision algorithm.

The server state is defined to be the number of consecutive interactive jobs which have been serviced (up to a finite limit, L-1). Thus, at each service completion the server is in one of L possible states, and there are n_T jobs in the interactive terminal queue and n_B jobs in the batch queue. The triple (n_T, n_B, S) (where S denotes the server state) defines the state of the entire system at a departure epoch. If we introduce the additional assumptions of Poisson arrivals to each queue with rates λ_T and λ_B and exponential service times with rates μ_T and μ_B for terminal and batch jobs, respectively, then the model defines an infinite state Markov chain.

In order to complete the specification of the model, the transition function among states must be defined. Transitions due to job arrivals are specified as:

batch arrival: $(n_T, n_B, S) \rightarrow (n_T, n_B+1, S)$

interactive terminal arrival: $(n_T, n_B, S) \rightarrow (\min(n_{T+1}, M), n_B, S)$ where M is the capacity of the terminal queue

State transitions at departures are more complicated to specify, since when a service completion occurs, the decision algorithm must be used to determine whether a terminal or batch job will be chosen for service. We will now define this algorithm and indicate the motivation for it.

Associated with each server state s, there is a breakpoint b_s. The first breakpoint, b_o, is defined to be zero, and the last, b_{L-1} has the value $+\infty$. The vector of breakpoint values is expected to be non-decreasing, although this is not a requirement. At a service completion, the server obeys the following algorithm:

1. If both queues are non-empty and the server is in state s, then
   a. If $n_T \geq b_s$, select the next job to be serviced from the interactive terminal queue and set

$S \leftarrow S + 1$. (The next state is $(n_T - 1, n_B, \min(S + 1, L - 1))$.

b. If $n_T < b_S$, choose the next job from the batch queue and set $S \leftarrow 0$. (The next state is $(n_T, n_{B-1}, 0)$).

2. If only the terminal queue is non-empty, choose a job from it and set $S \leftarrow \min(S + 1, L - 1)$. (The next state is $(n_{T-1}, 0, \min(S + 1, L - 1))$.
3. If only the batch queue is non-empty, choose a job from it and set $S \leftarrow 0$. (The next state is $(0, n_{B-1}, 0)$).
4. If both queues are empty, set $S \leftarrow 0$ and enter a distinguished idle state until the next arrival. At the time of the arrival, reapplying this algorithm.

The motivation for this scheme is the requirement that the batch stream receive varying degrees of service depending on the size of the terminal load at a given time. A minimum level of service for the batch stream is guaranteed by the fact that, even under saturated conditions, one batch job will be processed for every $L - 1$ terminal jobs (since $b_{L-1} = \infty$). The effect of the breakpoint vector and priority algorithms is to represent the server as querying and responding to the system state after each departure: if S terminals have been serviced in a row and the terminal queue still equals or exceeds b_S, the current breakpoint, (i.e., the terminal queue is "too long") service another terminal and increment S. Otherwise, service a batch job and reset S to zero, indicating that a batch job has entered service. A policy under this algorithm corresponds to fixing the number of server states, L, and the values of the breakpoints $b_0, b_1, \ldots, b_{L-1}$. By changing the policy used, different control algorithms may be modelled.

In (10) the numerical techniques developed by Wallace (17) are shown to be applicable to the determination of the steady state probability distribution of the model. The mean queue lengths for batch and terminal jobs can be determined from this distribution, and by applying Little's theorem (12) the mean system residence times can also be defined. The derivations of these results are omitted for the sake of brevity. A program has been written incorporating these techniques and this program was used to generate the results

given below.

III. APPLICATION OF THE MODEL TO THE MICHIGAN TERMINAL SYSTEM

In order to assess the usefulness of this relatively general, high-level model, it has been used to represent the Michigan Terminal System. This large scale interactive and batch system, which has been implemented on the IBM 360/67 and 370/168, is described in (2, 10, 13, 15). It includes a load leveling algorithm which controls the number of batch streams (batch job initiators) on the basis of current system performance. Measurements of CPU activity, paging activity, and disk and channel I/O activity are combined in a weighted sum which defines a current load factor for the system. This load factor is combined with weighted values of the last several load factors computed to provide exponential smoothing of the final load factor. The system decides, on the basis of this final load factor, whether to increase, decrease, or leave unchanged the number of batch initiators. (In fact, the algorithm is more complex than this, since batch initiators are not all identical: each initiator looks for a certain class of batch jobs to initiate, based on execution time estimates).

Statistics from 15 different periods, ranging from three to eight hours in length, were gathered in order to determine reasonable values for arrival and service rate parameters in the model and to provide a yardstick for the model's predictions. Details of the collection methods and values observed can be found in (10, 11) and will not be repeated here. We note that system behavior seemed to fall into three general categories, defined by mean CPU utilization and batch queue length:

1. Lightly loaded: $0\% < \overline{CPU} \leq 60\%$; $\overline{\text{Batch Queue}} = 0$
2. Moderately loaded: $60\% < \overline{CPU} < 90\%$; $0 < \overline{\text{Batch Queue}} \leq 5$
3. Heavily loaded: $90\% \leq \overline{CPU}$; $\overline{\text{Batch Queue}} > 5$

For each of the 15 measurement periods, the arrival and service rates of batch and terminal jobs were determined. Service rates were based on observed job CPU requirements only, since the system being measured was in fact generally CPU-limited. The values for M (capacity of the terminal queue), L (number of server states), and the breakpoints were determined on the basis of preliminary studies. Since the cost of computing the analytic solution is proportional to the product of M and L, there was a strong motivation to keep these two model parameters as small as possible without introducing too much

Table 1 Summary of Parameter STudies

Data From Period	Max #Term (M)	λ_T	λ_B	μ_T	μ_B	$E(W_B)$	% non-idle	$E(L_B)$	Std. Dev.
1	5	.01063	.02465	.07524	.20454	8.4	26.2	.206	.53
2	5	.01014	.02063	.05739	.1741	11.3	29.6	.233	.59
3	5	.01509	.03028	.06717	.13116	18.2	45.5	.552	1.02
4	5	.02319	.03542	.05890	.17083	30.8	59.8	1.09	1.95
5	5	.02633	.04994	.11210	.14831	18.8	57.1	.938	1.47
6	5	.03269	.06769	.1085	.18359	23.9	66.9	1.62	2.39
7	5	.0370	.03522	.08522	.10421	55.4	76.1	1.95	2.72
	10					64.5	77.4	2.27	3.26
8	5	.04926	.03111	.09279	.11338	53.4	78.2	1.66	2.35
9	5	.04324	.04056	.07724	.11227	145.2	89.1	5.89	7.09
	10					251.5	91.8	10.2	12.3
10	5	.04833	.03826	.11354	.07448	184.8	91.8	7.07	7.94
	10					290.1	93.9	11.1	12.2
11	10	.04565	.03426	.08602	.09284	180.7	89.7	6.19	7.73
12	10	.04906	.04778	.08846	.11962	368.4	95.1	17.6	18.8
13	10	.04517	.03772	.08848	.08650	280.5	94.4	13.4	15.1
14	10	.04552	.03622	.08509	.08081	911.1	97.8	33.0	32.0
15	10	.04544	.03528	.07779	.08594	1173.5	98.6	41.4	38.9

For all cases
L = 4
Breakpoints = 1,2,3,∞

Parameter Studies

Single Node Model

Table 2 Analytic Model-System Data Comparison

Collection Period	% CPU Non-idle		Mean Batch Q		Std. Dev. Batch Q	
	Data	Model	Data	Model	Data	Model
1	26.31	26.2	.374	.206	.65	.53
2	29.52	29.6	.405	.233	.63	.59
3	48.90	45.5	.823	.552	1.23	1.02
4	60.33	59.8	2.08	1.09	3.68	1.95
5	57.36	57.1	1.65	.938	1.88	1.47
6	66.91	66.9	2.27	1.62	3.51	2.39
7	79.1	77.4	3.39	2.27	5.64	3.26
8	80.51	78.2	1.80	1.66	1.76	2.35
9	91.95	91.8	12.2	10.2	9.46	12.3
10	93.95	93.9	10.34	11.1	8.84	12.2
11	90.02	89.7	19.4	6.19	16.1	7.73
12	95.32	95.1	25.8	17.6	14.4	18.8
13	94.67	94.4	27.3	13.4	26.0	15.1
14	98.30	97.8	50.1	33.0	37.8	32.0
15	99.43	98.6	13.8	41.4	9.6	38.9

Analytic Model - System Data Comparison

error into the results. The values finally chosen are shown in Table 1. For the heavily loaded periods, L = 4 and M = 10 were chosen, and a reduction to M = 5 for the lightly loaded periods proved to yield sufficiently accurate results.

The values predicted by the model for the mean and the standard deviation of the batch queue length are shown in Table 2, next to the observed values.

Since the analytical results for percent non-idle CPU depend only on the arrival and service rates (which were derived from the data) the close agreement between model and data for this parameter indicates that the finite length of the terminal queue in the model did not seriously bias the results. The mean batch queue lengths predicted by the analysis show the same trends as the observed values, although they tend to underestimate them. Figures 2, 3, and 4 detail the correspondence between the predicted and observed values. In Figure 2, arrows point from predicted to observed values. When CPU utilization is below 90%, the absolute error is not large, although there is a consistent underestimate of batch queue lengths in the predictions. In the lightly loaded periods, this tendency to underestimate is due in part to the single server assumption. Since the statistics were in fact collected from a dual-processor system, the single server in the model is defined to have twice the service rate of the actual CPU's. In lightly loaded periods, the actual system will have only one CPU busy, which will have a service rate half that of the model.

When the CPU utilization is above 90%, the observed data become much more difficult to predict, and the values projected by the model vary in both directions from observed statistics. Figure 3 shows the data points with error bars indicating distances of one standard deviation in each direction from the observed means. From this point it is clear both that the predicted means all fall within one standard deviation of the observed means and that the standard deviations observed, especially in the heavily loaded regions, are quite large.

Two explanations are possible for these observations. First, the mean batch queue length may not be describable as a simple function of the CPU utilization. This is particularly true when the load is heavy, since other bottlenecks may appear in the system. In this case, the mean batch queue length may increase while CPU utilization stays fixed. Secondly, when the system is heavily loaded, the basic existence of a steady state distribution is called into question. A look at the actual structure of MTS heavy periods

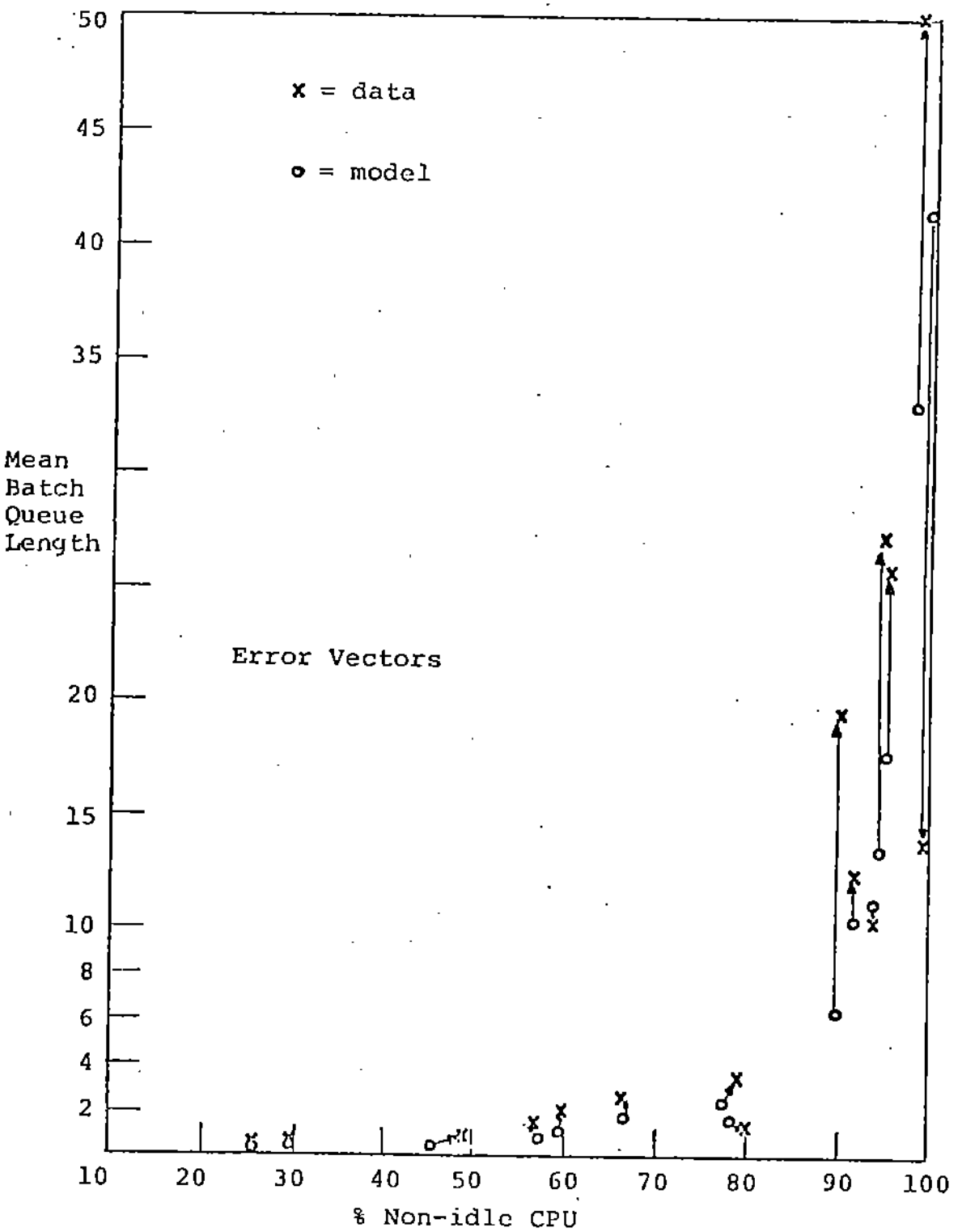


Fig. 2 Model vs. Data - Error Vectors

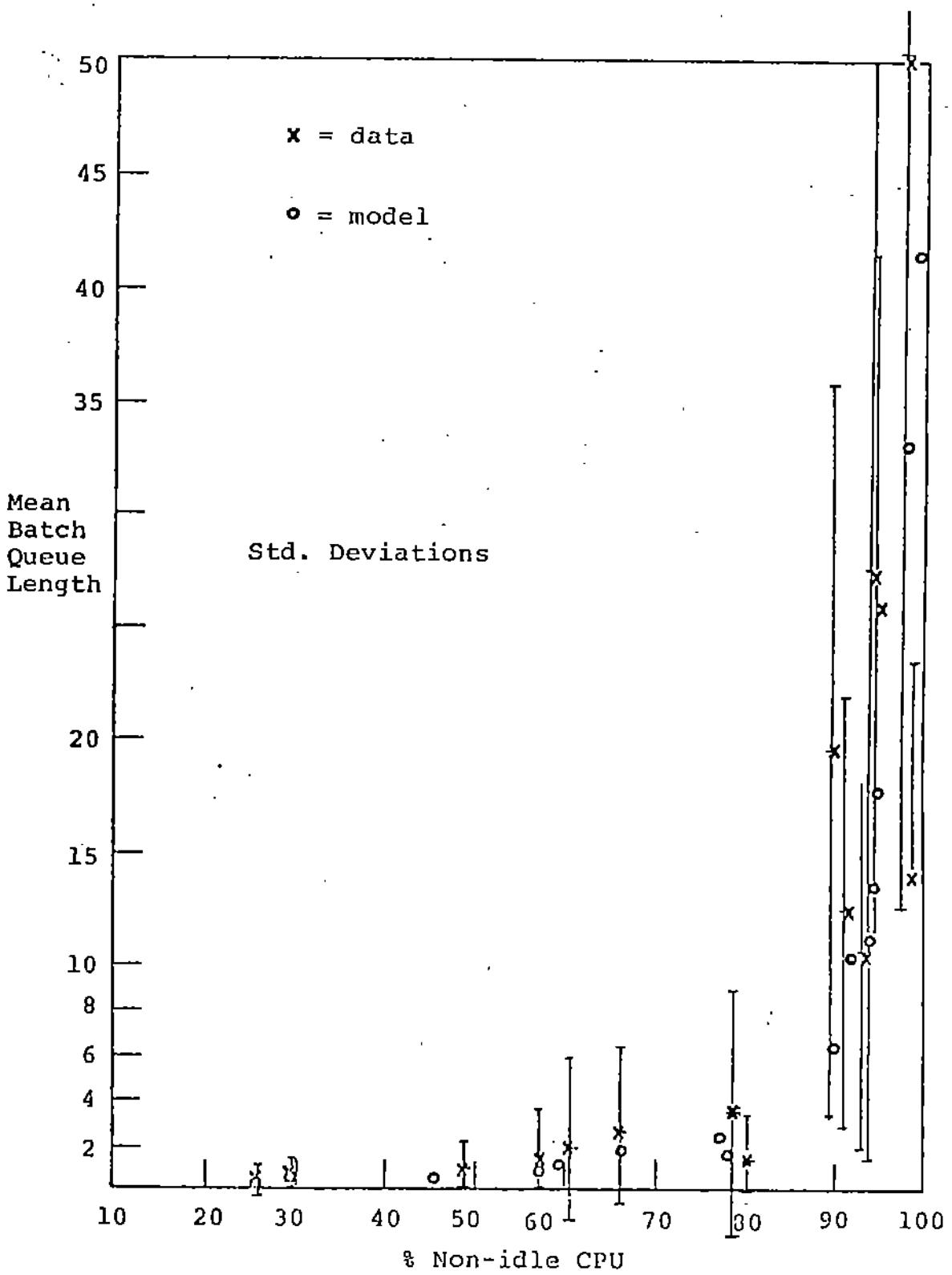


Fig. 3 Model vs. Data - Std. Deviations

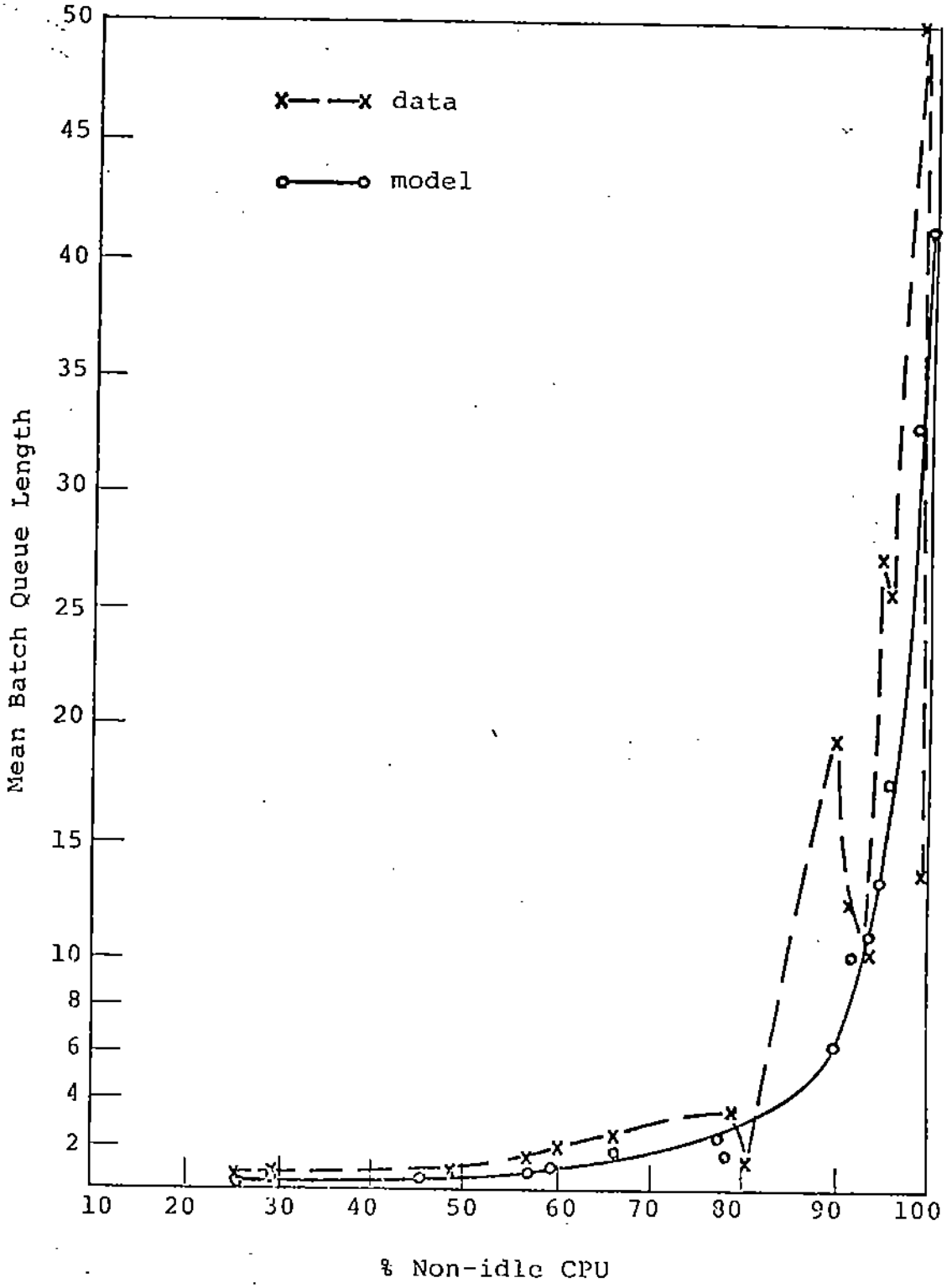


Fig. 4 Model vs Data

indicates that they are often characterized by a rising demand for interactive terminal service for several hours, during which time the batch queue grows in length, followed by a decrease in terminal use (around dinner time, for example.) As the arrival rate of terminal jobs declines, the batch queue is processed more quickly by the system, so the CPU utilization remains high until the batch queue is depleted. Thus, the heavily loaded periods may be dominated by several transient processes.

Finally, Figure 4 contrasts a smooth curve which fits the predicted data values with a jagged line produced by connecting the data points defined by the system measurements. Despite the differences in detail, the model does reflect the global characteristics of the system.

IV ADDITIONAL PARAMETER STUDIES

After validating the model, several parameter studies were made to relate the expected system residence time (turnaround) for batch jobs to the arrival rates for batch and terminal jobs. The mean system residence time for batch jobs ($E(W_B)$) can be obtained from the mean number in system via Little's Theorem ($\overline{L} = \lambda\overline{W}$). In each case, arrival and service rates were first determined from system statistics and the arrival rate for batch jobs was then varied. Statistics chosen for these studies included two light, three moderate, and two heavy periods.

Figure 5 shows the results of these studies. This graph discloses a sharp division of the system's performance into two apparent regions of operation. The first region, illustrated by the three steeply rising curves, corresponds to periods in which $E(W_B)$ is very sensitive to changes in batch arrival rate. This sensitivity is due to the relatively havey interactive load on the system; so that if the batch arrival rate is increased, the queue lengths (and hence residence times) grow rapidly. Conversely, in those cases in which $E(W_B)$ is relatively insensitive to the batch arrival rate, the interactive load is light and the system, as a whole, is underloaded. At such times, additional batch arrivals can be handled with only a small increase in $E(W_B)$.

It is also noteworthy that those cases originally classified as moderate periods fall in both regions of operation, indicating that the initial division of system states into three categories is finer than required. The resulting division of the operating region of the system into "good turnaround" and "bad turnaround" regions has considerable intuitive appeal, since MTS batch

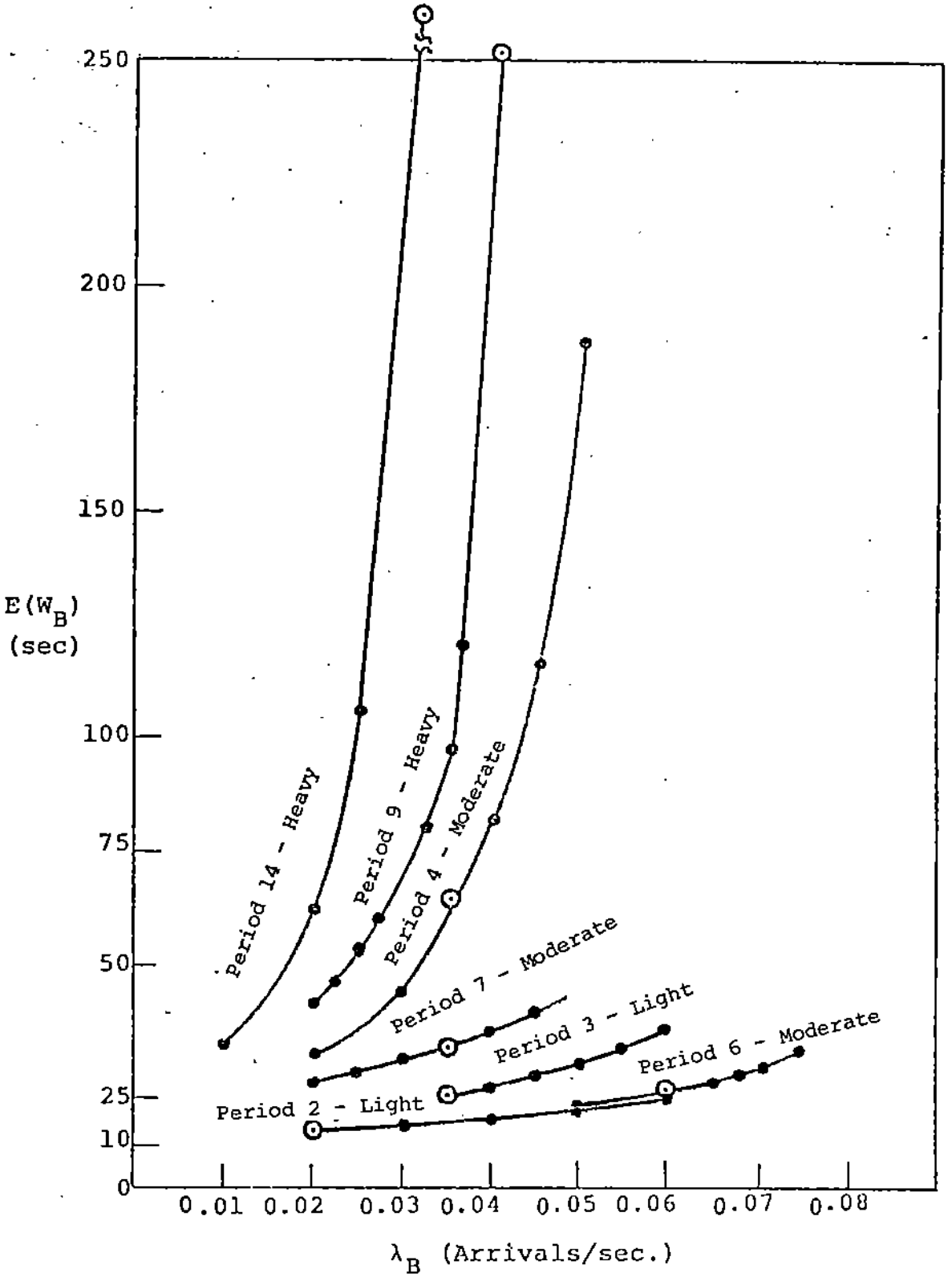


Fig. 5 Parameter Studies--Mean Batch Wait vs. Batch Arrival Rate

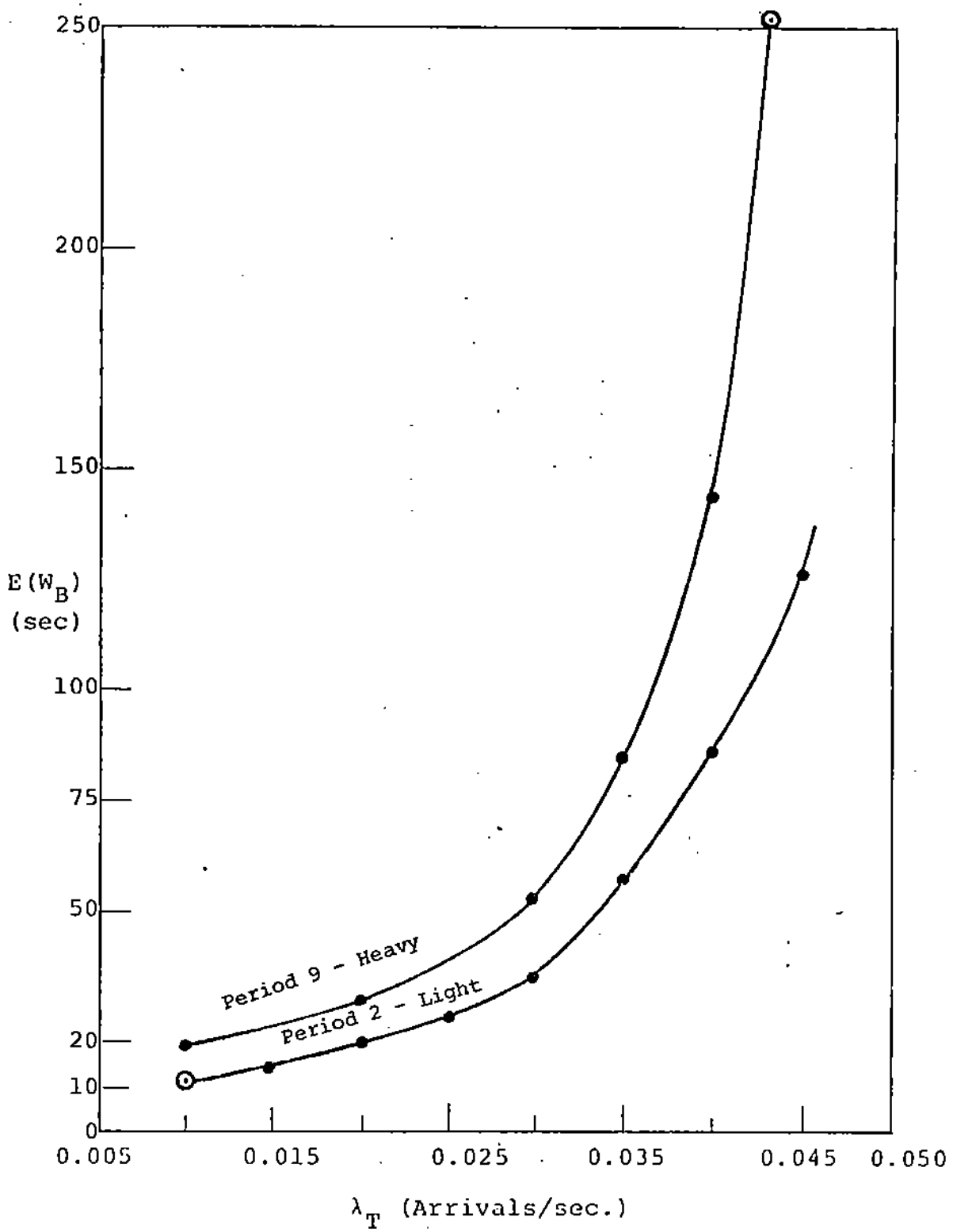


Fig. 6 Parameter Studies - Mean Batch Wait vs Terminal Arrival Rate

users often observe that turnaround is either vary short (a few minutes) or else relatively long (an hour or more).

This bifurcation of the operating region suggests that, if similar studies were run in which the terminal, rather than batch, arrival rate were varied and $E(W_B)$ observed as a function of this variation, the resulting plot would contain a knee. The knee would correspond to that point at which terminal service would begin to saturate the system; batch waiting times would rise rapidly beyond this point, because of the priority placed on terminal service.

Two such studies were made, one corresponding to a lightly loaded period during which the terminal arrival rate was gradually increased, and the other using data from a heavily loaded period, successively decreasing the terminal arrival rate. The results of these studies are shown in Figure 6. In both cases, the curve for $E(W_B)$ begins to rise sharply as the arrival rate becomes greater than .030 jobs per second. Since only the control algorithm was specified in the design of the model, this result demonstrates the model's ability to realistically reflect the system's behavior.

V. CONCLUSION

That the model developed here is capable of representing computer systems which include algorithms for controlling service delivered to batch and interactive users has been demonstrated by the validation and parameter studies. This has been shown to be true despite the model's relatively high level and the numerous assumptions made to ensure its mathematical tractability. Although the primary control algorithms portrayed in the model design are those which favor terminal service over batch as the interactive load increases, the model can easily be adapted to represent other priority schemes. Finally, the endogenous priority mechanism presented here can be used to study those lower level portions of computing hardware where more than one class of requests is served but priorities are not determined solely as a function of the request class.

REFERENCES

1. Badel, M., Gelenbe, E., Leroudier, J., and Potier, D., Adaptive optimization of a time-sharing system's performance. Proc. of the IEEE 63, 6 (June 1975), 958-965.

2. Boettner, D. W., and Alexander, M. T., The Michigan Terminal System. Proc. of the IEEE 63, 6 (June 1975), 912-918.

3. Brandwajn, A., A model of a time sharing virtual memory system solved using equivalence and decomposition methods. Acta Informatica 4, 1 (1974), 11-48.

4. Brandwajn, A., Buzen, J., Gelenbe, E., and Potier, D., A model of performance for virtual memory systems. ACM SIGMETRICS Performance Evaluation Review 3, 4 (Dec 1974), 9.

5. Denning, P. J., The working set model for program behavior. CACM 15, 5 (May 1968), 323-333.

6. Denning, P. J., Virtual memory. Computing Surveys 2, 3 (Sept 1970), 153-189.

7. Denning, P. J., Third generation computer systems. Computing Surveys 3, 4 (Dec. 1971), 175-216.

8. Fogel, M. H., The VMOS paging algorithm. ACM SIGOPS Operating System Review 8, 1 (Jan 1974), 8-17.

9. Jaiswal, N. K., Priority Queues. Academic Press, New York, 1968.

10. Landwehr, C. E., Load Sharing in Computer Networks: A Queueing Model. MERIT Computer Network MCN-1174-TR-18 (Nov. 1974), Ann Arbor, Michigan.

11. Landwehr, C. E., Usage statistics for MTS. ACM SIGMETRICS Performance Evaluation Review 4, 2 (April 1975).

12. Little, J., A proof of the queueing formula $L = \lambda W$. Operations Research 9, 3 (1961), 383-387.

13. Moore, C. G., Network Models for Large Scale Time Sharing Systems. Ph.D. dissertation, University of Michigan, Ann Arbor, 1971.

14. Morris, J. B., Demand paging through utilization of working sets on the MANIAC II. CACM 15, 10 (Oct. 1972), 867-872.

15. Pirkola, G. C., A file system for a general purpose time-sharing environment. Proc. of the IEEE 63, 6 (June 1975), 918-924.

16. Rodriguez-Rosell, J. and Dupny, J. P., The design implementation, and evaluation of a working set dispatcher. CACM 16, 4 (April 1973), 247-253.

17. Wallace, V. L., The Solution of Quasi Birth and Death Processes Arising from Multiple Access Computer Systems. Ph.D. dissertation, University of Michigan, Ann Arbor, 1969.