1975

A Queueing Model for A Multiprocessor System with Partitioned Memory

K. Omahen

V. Marathe

Report Number:
75-132

A QUEUEING MODEL FOR A MULTIPROCESSOR SYSTEM WITH PARTITIONED MEMORY

K. Omahen

Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

and

V. Marathe

Department of Business Administration
Illinois University

January, 1975

CSD-TR 132

Abstract

A general multi-resource queueing system is defined to be a single congestion point associated with a number of different resource types and having arrivals which require some combination of the system resources simultaneously for the duration of their processing times. Such a system is characterized by a variable processing rate which is a function of the combination of jobs being concurrently serviced at any instant. The capacity bound for a multi-resource queue is the smallest input rate which is guaranteed to cause saturation regardless of the scheduling rule employed, give some fixed set of job stream characteristics.

This paper examines the performance of a simple example of a multi-resource queue, a two CPU system with two memory partitions. There are two classes of Poisson arrivals, and each job class has independent and exponentially distributed service times. Resource requirements are such that Class-1 jobs need one CPU and one block of memory, while Class-2 jobs require a single CPU and both memory blocks in order to be executed. An algorithm for calculating the capacity bound is given which enables one to determine the "optimal" proportion of time that the system should spend in processing various job combinations. Seven scheduling rules are described in terms of the manner in which preference is given to different job combinations, and the notion of assigning priorities to combinations of jobs is stressed. The paper then gives an overview of the relative system performance under these rules by comparing the system capacity and average flow times for these disciplines. Finally, a sample derivation is provided for one of the scheduling rules in order to illustrate a powerful analytic technique used by the authors to obtain many of the described results.

Title: A Queueing Model for a Multiprocessor System with Partitioned Memory

Queueing Theory can provide valuable insights into various aspects of computer system performance, but existing computer systems exhibit forms of resource allocation which are not accurately represented by the queueing models analyzed to date. An examination of the literature leads to the conclusion that previously analyzed models have two common features: (i) each queue is associated with either a single resource or a number of identical resources, and (ii) arriving jobs require exactly one unit of the scarce resource. This conclusion is valid for a vast majority of the models including queues with feedback, networks of queues, and multiple-server queues.

The theses by the authors [1,2] have independently attacked a class of queueing problems involving a form of resource allocation not previously treated in the literature. In order to better define this class of problems, the term <u>multi-resource queue</u> is introduced to describe the situation in which a <u>congestion point is associated with a number of resources and where job arrivals require the simultaneous use of some combination of the system resources</u>. Computer systems provide strong motivation for examining multi-resource queues because a job or process must generally be allocated both a processor and primary memory in order for execution to take place. The notion of a multi-resource queue may also be seen in a simulation language such as GPSS [3] where users may define storage entities to handle discrete resources for which the allocation quantity may be several units. This paper presents results for an example of a multi-resource queue which, while, simple in certain respects, nevertheless exhibits a number of interesting properties which are quite different from those for the usual case of a queueing system involving a single resource type.

<u>Notation & Terminology for a General Multi-Resource Queue</u>

A general multi-resource queue is a system consisting of several different resource types and an arbitrary number of units of each resource type. Each job arriving to the system requires a combination of the system resources simultaneously for the duration of the processing time of the job. The arriving jobs fall into various classes, and each job class is characterized by an arrival process, a processing time distribution, and a fixed resource request which describes the resource requirements of each job within the class. This implies that each job within the same class has identical resource requirements; however, this fixed resource request may instead be interpreted to be the maximum resource requirements for jobs within a given class. In this latter case, the deadlock prevention technique proposed by Habermann [4] might be employed; dynamic resource allocation could take place, but deadlock would be prevented by never simultaneously processing a set of jobs for which the total of the maximum resource requirements would exceed the available system resources. There will be a queue for jobs waiting for service, and this queue will be taken to be infinite in length unless specified otherwise.

Define:

I = Number of Resource Types ($I \geq 1$).

R_i = Amount of the ith Resource in System, $i = 1,\ldots,I$.

J = Number of Job Classes.

Jobs belonging to Class-j, where $j = 1,\ldots,J$, have the following characteristics:

$\overline{V}_j = (r_{j1}, r_{j2},\ldots,r_{jI})$ = Resource Request Vector for the jth Job Class indicating that r_{j1} units of Resource Type 1 are required, r_{j2} units of Resource Type 2, etc. Furthermore, $0 \leq r_{ji} \leq R_i$ and $r_{ji} > 0$ for at least one i.

λ_j = Arrival Rate for Class-j Jobs (hence, $1/\lambda_j$ = mean interarrival time for Class-j Jobs).

$1/\mu_j$ = Expected Processing Time for Class-j Jobs.

The total input rate λ for jobs of all classes is defined to be

$$\lambda = \sum_{j=1}^{J} \lambda_j .$$

The proportion of jobs in the overall input stream which belong to Class-j will be denoted by f_j and defined as

$$f_j = \lambda_j/\lambda, \text{ where } 1 \leq j \leq J.$$

Given the basic characteristics of a multi-resource queue, it is now possible to introduce additional terminology which is useful for describing queueing systems of this type. For a multi-resource queue, it is possible to have several jobs simultaneously being processed by the system, subject to the restriction that the total of the resources required by the combination of jobs is less than or equal to the available resources in the system.

Define:

$[n_{k1}, n_{k2}, \ldots, n_{kJ}]$ = Job Combination k consisting of n_{k1} jobs of Class-1, n_{k2} jobs of Class-2, etc.

A feasible job combination k is one having the following property:

$$\sum_{j=1}^{J} n_{kj} * r_{ji} \leq R_i \quad \text{for } i = 1,2,\ldots,I.$$

A multi-resource queueing system will be said to be saturated when the expected flow time for one or more job classes is infinite.*

The capacity of a multi-resource queueing system under a scheduling rule is defined as follows. The characteristics of the job stream will be fixed in a manner to be next described. Assume a stationary distribution for the processing times associated with each job class, and take the proportion of Class-j jobs in the overall input stream, f_j, to be fixed at an arbitrary value which is subject to the following restrictions:

$$0 \leq f_j \leq 1, \text{ and } \sum_{j=1}^{J} f_j = 1.$$

Given that the type of arrival process for each job class also remains constant, we will say that the job stream characteristics are fixed and that the only parameter which can vary is the overall input rate. Under these circumstances, the capacity of the system under a specified scheduling discipline is defined to be the smallest overall input rate at which the system becomes saturated.

A capacity bound, when it exists for a multi-resource queue, will be defined to be λ_{max}=infimum of the overall input rates at which the system is guaranteed to saturate regardless of the discipline which is used, given that the job stream characteristics are fixed as described previously. The capacity bound for a multi-resource queue is useful for measuring the performance of various scheduling disciplines because, given some fixed job stream characteristics, the capacity under a particular scheduling rule may be less than the capacity bound. If the capacity under a certain scheduling discipline is equal to the capacity bound for every set of job stream characteristics, the scheduling rule is said to be a full-capacity discipline for the multi-resource queueing system.

The multi-resource queueing system will in general have a variable processing rate which is a function of both the number and type of jobs in service. The feasible job combinations are obviously important in this respect because the processing rate depends directly on the manner in which jobs are concurrently processed. Scheduling rules for multi-resource queues will greatly influence the processing rate through the choice of job combination to be processed at any instant. The capacity bound is a useful quantity because it specifies the smallest input rate which saturates the system regardless of the manner in which the scheduling rule operates; this capacity bound is therefore related to the maximum expected processing rate that can be achieved by the system when given a specified set of job stream characteristics.

This paper will be concerned with multi-resource queues for which work-conserving disciplines are employed, where the term "work-conserving" refers to situations in which there is no wasted processing time such as occurs with switchover times or with preemptive-repeat priority disciplines. For the classical single-server queue, work-conserving disciplines are full-capacity disciplines. In contrast, a discipline for a multi-resource queue may be work-conserving without being able to achieve full-capacity.

Two CPU System With Two Units of Memory

A multi-resource queueing system which will be examined under a variety of different scheduling disciplines will now be described. The system will consist of two resource types which will be interpreted to be Central Processing Units (CPUs) and blocks of primary memory. There are two CPUs (Resource Type-1), and the primary memory has been partitioned into two blocks (Resource Type-2). There are two job classes; Class-1 jobs require one CPU and one block of memory, while Class-2 jobs require one CPU and two blocks of primary memory. Class-1 and Class-2 jobs arrive in a Poisson stream at rates λ_1 and λ_2, respectively. The processing times for Class-1 jobs have a negative exponential distribution, and those for Class-2 jobs are also exponentially distributed. Using notation introduced for the general model, the system may be described as follows:

$R_1 = 2$ (Number of CPUs).

$R_2 = 2$ (Number of blocks of primary memory).

Class-1 Jobs:

λ_1 = Poisson Arrival Rate for Class-1 Jobs.

P_1 = Random Variable denoting a Class-1 Processing Time which has a negative exponential distribution.

$E(P_1)$ = Expected Class-1 Processing Time,
= $1/\mu_1$.

$\bar{V}_1$ = (1,1) = Resource Request Vector for Class-1 Jobs indicating that one CPU and one block of primary memory are required.

F_1 = Random Variable representing a Class-1 Flow Time (i.e. the interval between the arrival of a job and the completion of service for that job).

Class-2 Jobs:

λ_2 = Poisson Arrival Rate for Class-2 Jobs

P_2 = Random Variable denoting a Class-2 Processing Time having a negative exponential distribution.

$E(P_2)$ = Expected Class-2 Processing Time,
= $1/\mu_2$.

$\bar{V}_2$ = (1,2) = Resource Request Vector for Class-2 Jobs indicating that one CPU and two blocks of memory are needed.

F_2 = Random Variable used to represent a Class-2 Flow Time.

For any multi-resource queue, the <u>feasible</u> job combinations specify the various ways in which combinations of jobs may be simultaneously processed. The resource requirements of the two job classes give the following feasible combinations for this system:

C_1 =	[1,0]	One Class-1 Job
C_2 =	[0,1]	One Class-2 Job
C_3 =	[2,0]	Two Class-1 Jobs

Information concerning the resource requirements of the job classes is not needed to analyze the described system if the feasible job combinations are known. It might be pointed out that the same combinations might result for another multi-resource queueing system with different resources and changed resource requirements for the job classes; for example, a two-resource system with two job classes would have the same set of feasible job combinations for the case in which $R_1 = 2$, $R_2 = 5$, $\bar{V}_1 = (1,2)$, and $\bar{V}_2 = (1,4)$. This stresses the importance of the feasible job combinations in describing the system and implies that results of analyzing the Two-CPU System With Two Units of Memory will be applicable to other multi-resource queueing system with similar sets of feasible combinations.

The <u>capacity bound</u> λ_{max} has been previously defined as the infimum of the overall input rates at which the system is guaranteed to saturate regardless of the scheduling rule which is employed, given that the job stream characteristics are held constant. This capacity bound λ_{max} will be a function of the valid job combinations, the relative input rates, and the processing time requirements for the two job classes. The following notation will be required in the material which follows:

$\lambda = \lambda_1 + \lambda_2$ = Overall Poisson Arrival Rate for Jobs

$f_1 = \lambda_1/\lambda$ = Proportion of jobs which belong to Class-1.

$f_2 = \lambda_2/\lambda$ = Proportion of jobs which belong to Class-2.

If the job stream characteristics are held constant for this system (i.e., f_1, f_2, $E(P_1)$, and $E(P_2)$ held constant), the capacity bound λ_{max} for the Two CPU System With Two Units of Memory is given by

$$\lambda_{max} = 1/[f_1 E(P_1)/2 + f_2 E(P_2)]. \qquad (1)$$

PROOF. Let the following functions represent the steady-state probabilities that the system is in a specified state, given that the system is operating under an arbitrary scheduling rule and that the system is nonsaturated at input rate λ:

$\pi_0(\lambda)$ = Pr[system idle],
$\pi_1(\lambda)$ = Pr[job combination [1,0] in progress],
$\pi_2(\lambda)$ = Pr[job combination [0,1] in progress],
$\pi_3(\lambda)$ = Pr[job combination [2,0] in progress].

Assuming that there is zero overhead, all system states have been introduced, and it must be the case that

$$\sum_{i=1}^{3} \pi_i(\lambda) = 1 - \pi_0(\lambda), \text{ and } \pi_0(\lambda) > 0.$$

The amount of work which arrives to the system per unit of time for each of the two classes is

$\lambda f_1 E(P_1)$ = Expected amount of processing time requested by Class-1 jobs per unit of time,

$\lambda f_2 E(P_2)$ = Expected amount of processing time requested by Class-2 jobs per unit of time.

The steady-state probabilities may be related to the amount of work arriving per unit of time by applying Little's Equation [5] to the processor system; this gives the following relation:

[Expected no. of jobs in progress]
=[Arrival Rate for jobs] · [Expected time in processor].

Using the above result, the following equations are obtained for Class-1 and Class-2 jobs, respectively:

$$[\pi_1(\lambda) + 2\pi_3(\lambda)] = [\lambda f_1][E(P_1)],$$
$$[\pi_2(\lambda)] = [\lambda f_2][E(P_2)].$$

The sum of the probabilities that a valid job combination is in progress may be made arbitrarily large by increasing the input rate λ, and the system capacity $\lambda_{\max j}$ under the jth discipline will be that input rate such that

$$\lim_{\lambda \uparrow \lambda_{\max j}} \sum_{i=1}^{3} \pi_i(\lambda) = 1 \text{ and } \lim_{\lambda \uparrow \lambda_{\max j}} \pi_0(\lambda) = 0.$$

The capacity bound $\lambda_{\max}$ is the smallest input rate at which the system is guaranteed to saturate, and the bound can be found by solving the following Linear Program:

$$\lambda_{\max} = \max_{\lambda, \pi_1(\lambda), \pi_2(\lambda), \pi_3(\lambda)} \lambda$$

such that $\pi_1(\lambda) + 2\pi_3(\lambda) = \lambda f_1 E(P_1)$,

$\pi_2(\lambda) = \lambda f_2 E(P_2)$,

and $\pi_1(\lambda) + \pi_2(\lambda) + \pi_3(\lambda) = 1.$

The solution for all f_1 and f_2 is found by inspection to require $\pi_1(\lambda) = 0$, and therefore for $\lambda = \lambda_{\max}$

$$\pi_1(\lambda) + \pi_2(\lambda) + \pi_3(\lambda) = \lambda f_1 E(P_1)/2 + \lambda f_2 E(P_2),$$

$$= \lambda[f_1 E(P_1)/2 + f_2 E(P_2)],$$

$$= 1,$$

which implies that

$$\lambda_{\max} = 1/[f_1 E(P_1)/2 + f_2 E(P_2)]. \qquad \text{Q.E.D.}$$

In obtaining this result, it was not necessary to take into account the distribution types for the arrival process and processing times. The bound is valid for arbitrary distributions and depends primarily on the valid job combinations for the system.

The above derivation illustrated that the capacity bound may be calculated as the solution of a Linear Program, and the solution gives insights into the manner in which jobs should be simultaneously processed in order to achieve full-capacity. Consider the proportion of time that should be spent in processing the various job combinations as the input rate approaches the capacity bound; the derivation gave the following values:

$$\lim_{\lambda\uparrow\lambda_{\max}} \pi_1(\lambda) = 0, \quad \lim_{\lambda\uparrow\lambda_{\max}} \pi_2(\lambda) = \lambda_{\max} f_2 E(P_2), \quad \lim_{\lambda\uparrow\lambda_{\max}} \pi_3(\lambda) = \lambda_{\max} f_1 E(P_1)/2.$$

The above values will be referred to as the <u>solution set of state probabilities</u> for the Linear Program. These values suggest that the system should spend all of its time in processing either a single Class-2 job or a pair of Class-1 jobs in order to achieve the capacity bound. A single Class-1 job may be considered to be an "undesirable" job combination because it is assigned a probability of zero in the solution to the Linear Program and therefore should be avoided if system capacity is of prime importance.

References [1] and [2] independently analyzed a number of scheduling rules for the Two CPU System With Two Units of Memory. This paper is intended to be an overview which places the performance of various scheduling disciplines into perspective and which gives the reader insights into the behavior of multi-resource queueing systems.

Before describing various scheduling rules for this multi-resource queueing system, it is useful to introduce the notion of assigning priorities to combinations of jobs. In a single-resource system, priority disciplines will generally assign priorities to different classes of jobs; this situation may be regarded as an assignment of priorities to job combinations for the special case where the set of job combinations is identical to the set of different job classes (i.e., there is at most one job in service at any instant). The multi-resource queueing system is characterized by variable service rate which is a function of the job combinations in service (i.e., the degree of simultaneous processing is determined by the number of jobs in the combination). The performance of a multi-resource queueing system will be greatly influenced by the way in which the scheduling rule chooses the combination of jobs to be serviced at any instant, and the concept of priorities for job combinations follows in a natural way.

There are three feasible job combinations for the Two-CPU System With Two Units of Memory, and there are six possible priority orderings that could be assigned to these job combinations (assuming the priorities are to be different). If we consider these combinations, though, it is obvious that it makes little sense to give higher priority to a single Class-1 job than to a pair of Class-1 jobs, and priority orderings giving preference to a single Class-1 job over a pair of Class-1 jobs should be removed from consideration. Below are the three remaining priority orderings that could be given to the job combinations (top-to-bottom corresponds with highest-to-lowest priority):

$$\left.\begin{array}{ll} C_3 = [2,0] & \text{Two Class-1 Jobs} \\ C_1 = [1,0] & \text{One Class-1 Job} \\ C_2 = [0,1] & \text{One Class-2 Job} \end{array}\right\} \quad \text{Class-1 Static Priority}$$

$$\left.\begin{array}{ll} C_3 = [2,0] & \text{Two Class-1 Jobs} \\ C_2 = [0,1] & \text{One Class-2 Job} \\ C_1 = [1,0] & \text{One Class-1 Job} \end{array}\right\} \quad \text{Class-1 Conditional Priority}$$

$$\left.\begin{array}{lll} C_2 = [0,1] & \text{One Class-2 Job} \\ C_3 = [2,0] & \text{Two Class-1 Jobs} \\ C_1 = [1,0] & \text{One Class-1 Job} \end{array}\right\} \quad \text{Class-2 Static Priority}$$

When describing a priority scheduling rule for this system, one will be interested in not only the priority ordering for job combinations but also in whether a job combination has preemptive or nonpreemptive priority over some other job combination. The following notation will be used to further describe the scheduling rule:

$C_i > C_j$	denotes that job combination C_i has <u>nonpreemptive priority</u> over job combination C_j
$C_i >> C_j$	denotes that job combination C_i has <u>preemptive priority</u> over job combination C_j

Using this notation, a scheduling rule which assigns priorities to job combinations can be described in terms of the pairwise priority relationships that exist between each distinct pair of job combinations.

The authors have analyzed the following disciplines for the Two CPU System With Two Units of Memory; in each case it is assumed that there is no overhead involved in switching between job combinations.

(1) <u>First-Come-First-Served (FCFS) Discipline</u>:

Jobs go into service according to order of arrival whenever there are sufficient resources available.

(2) <u>Nonpreemptive Class-1 Static Priority Discipline</u>: $C_3 > C_2$, $C_1 > C_2$

Class-1 jobs have nonpreemptive priority over Class-2 jobs, and a Class-2 job is processed to completion upon going into service.

(3) <u>Preemptive Class-1 Static Priority Discipline</u>: $C_3 >> C_2$, $C_1 >> C_2$

Class-1 jobs always have preemptive priority over a Class-2 job.

(4) Preemptive Class-1 Conditional Priority Discipline: $C_3 >> C_2$, $C_2 >> C_1$

Class-1 jobs have preemptive priority over Class-2 jobs only when there are two or more Class-1 jobs in system. A Class-2 job has preemptive priority over a single Class-1 job.

(5) Preemptive Class-2 Static Priority Discipline: $C_2 >> C_3$, $C_2 >> C_1$

Class-2 jobs always have preemptive priority over Class-1 jobs.

(6) Mixed Class-2 Static Priority Discipline: $C_2 > C_3$, $C_2 >> C_1$

A Class-2 job has preemptive priority over a single Class-1 job but nonpreemptive priority over a pair of Class-1 jobs.

(7) Modified Alternating Priority Discipline: $C_2 >> C_1$

The relative priority of a Class-2 job and a pair of Class-1 jobs alternates as follows: If there are no Class-2 jobs and fewer than two Class-1 jobs in system, the ordering between C_2 and C_3 is undefined. Upon there being one or more Class-2 jobs and less than two Class-1 jobs in system, the ordering $C_2 > C_3$ goes into effect until no Class-2 jobs are in system. At the next epoch at which there are two or more Class-1 jobs and zero Class-2 jobs in system, the ordering $C_3 > C_2$ remains in effect until but one Class-1 job is in system. A typical busy period appears as an alternating sequence of Class-1 and Class-2 busy periods involving only job combinations of type C_3 and C_2, respectively; a single Class-1 job is processed only when it is the only job in system.

In each of the priority scheduling rules described above, it is assumed that the discipline is of the preemptive-resume type and that jobs within the same class are serviced in FCFS order.

Results will next be presented which enable the scheduling disciplines to be compared for two different measures of performance: (i) the capacity under the rule, and (ii) the average flow time for jobs when the rule is employed.

The capacity under a scheduling rule was previously defined to be the smallest input rate which causes the system to saturate for some specified set of job stream characteristics. The capacity of the jth discipline (as listed above) will be denoted by

λ_{max-j} = Capacity of discipline j, where $1 \leq j \leq 7$.

The capacity bound λ_{max} for this system has been previously given by Equation (1); it was previously stated that not every scheduling rule can achieve the capacity bound and that saturation might occur at input rates less than that given by the capacity bound for certain disciplines. The capacity for each scheduling rule is given below:

$$\lambda_{max-1} = 1/[f_1^2/(2\mu_1) + f_2^2/\mu_2 + f_1 f_2[1/\mu_1 + 1/\mu_2]], \qquad (2)$$

$$\lambda_{max-2} = [-B + \mathrm{SQRT}(B^2 - 4AC)]/(2A), \text{ where } A = f_1\mu_1[\mu_2 + 2\mu_1 f_2] \qquad (3)$$

$$B = \mu_1\mu_2[f_1\mu_2 + 2\mu_1(f_2-f_1)]$$

$$C = -2\mu_1^2\mu_2^2$$

$$\lambda_{max-3} = [-E + \mathrm{SQRT}(E^2 - 4DF)]/(2D), \text{ where } D = f_1 f_2 \qquad (4)$$

$$E = 2\mu_1 f_2 - f_1\mu_2$$

$$F = -2\mu_1\mu_2$$

$$\lambda_{max-4} = \lambda_{max-5} = \lambda_{max-6} = \lambda_{max-7} = \lambda_{max} \quad \text{(see Eqn. 1)} \qquad (5)$$

The disciplines 4, 5, 6, and 7 are full-capacity disciplines (i.e., the system saturates at the capacity bound), but the FCFS discipline, the Nonpreemptive Class-1 Static Priority rule, and the Preemptive Class-1 Static Priority discipline each saturate at an input rate less than that given by the capacity bound for certain job stream characteristics.

If the job stream consists only of Class-1 jobs (i.e., $f_1 = 1$ and $f_2 = 0$) or only of Class-2 jobs (f_1=0 and f_2=1), the capacities will obviously be identical for all of the scheduling rules. The more interesting situation is the one where both classes of jobs are present in the input stream; the FCFS discipline, the Nonpreemptive Class-1 Static Priority discipline, and the Preemptive Class-1 Static Priority discipline each has a capacity which is less than the capacity bound, but the relative ordering of these capacities is also of interest. If the Equations (1) through (4) are examined, the following partial ordering is found to exist between the capacities under the different scheduling rules:

$$\lambda_{max} \geq \lambda_{max-2}, \ \lambda_{max-2} \geq \lambda_{max-3}, \ \lambda_{max-2} \geq \lambda_{max-1}$$

where equality occurs only when the input stream contains only Class-1 or only Class-2 jobs. The capacity under the Nonpreemptive Class-1 Static Priority discipline will always be greater than or equal to the capacities of the Preemptive Class-1 Static Priority discipline and the FCFS discipline. The relative capacities of the FCFS discipline and the Preemptive Class-1 Static Priority discipline depend upon the job stream characteristics as specified below (for $f_1 > 0$ and $f_2 > 0$):

$2\mu_1 < \mu_2(1 + f_2)$ implies $\lambda_{max-1} < \lambda_{max-3}$

$2\mu_1 = \mu_2(1 + f_2)$ implies $\lambda_{max-1} = \lambda_{max-3}$

$2\mu_1 > \mu_2(1 + f_2)$ implies $\lambda_{max-1} > \lambda_{max-3}$

Let us review the capacity results which have been presented for the Two CPU System With Two Units of Memory. The FCFS discipline serves as a "benchmark" for comparing disciplines because the rule employs only the information concerning the order of arrival to the system when choosing the next job for processing. The FCFS discipline is not a full-capacity discipline in this case; this means that the processing of jobs in order of arrival to the system will not provide the same degree of concurrent processing as the full-capacity disciplines. This is not surprising since the FCFS discipline is restricted in the manner in which concurrent processing can be achieved. Both the Nonpreemptive and Preemptive Class-1 Static Priority disciplines give preference to Class-1 jobs, but better service for the Class-1 jobs is obtained at the expense of decreased system capacity. Again this multi-resource queueing system has characteristics which are counter-intuitive because it is usually desirable to give preferential service to the "small" jobs which require fewer of the system resources.

Figures 1, 2, and 3 give graphical examples of the manner in which the capacities of various disciplines vary as a function of the job stream characteristics. Each of the figures assumes that the expected processing times for Class-1 and Class-2 jobs are specified, and for each discipline graphs are given which show the points at which saturation takes place.

Another convenient measure for the performance of the system is the average flow time under each of the scheduling rules, where the average flow time $\bar{F}$ is defined as

$\bar{F} = f_1E(F_1) + f_2E(F_2)$, where $E(F_1)$ and $E(F_2)$ are the expected flow times for Class-1 and Class-2 jobs, respectively.

At high input rates, the average flow time should be anticipated to be lower for the full-capacity disciplines than for those which are not full-capacity rules. At low input rates, the non-full-capacity rules may perform slightly better than the full-capacity scheduling disciplines. Denote the average flow time under the j-th discipline by the following notation:

$\bar{F}_j = \bar{F}_j(\lambda, f_1, \mu_1, \mu_2)$,

= Average Flow Time under Discipline-j for a given set of job stream characteristics.

If parameter $f_1 = 1$ (only Class-1 jobs in the input stream) or if $f_1 = 0$ (only Class-2 jobs arriving to the system), all of the full-capacity disciplines obviously have the same average flow times. The full-capacity disciplines have an interesting property when there are both Class-1 and Class-2 arrivals to the system: the relative orderings of the average flow times under the disciplines depends only on the values for parameters μ_1 and μ_2. The relative orderings for the full-capacity disciplines are shown below for the case in which $0 < f_1 < 1$ (jobs of both classes in the arrival stream) and $0 < \lambda < \lambda_{max}$ (nonsaturated operation):

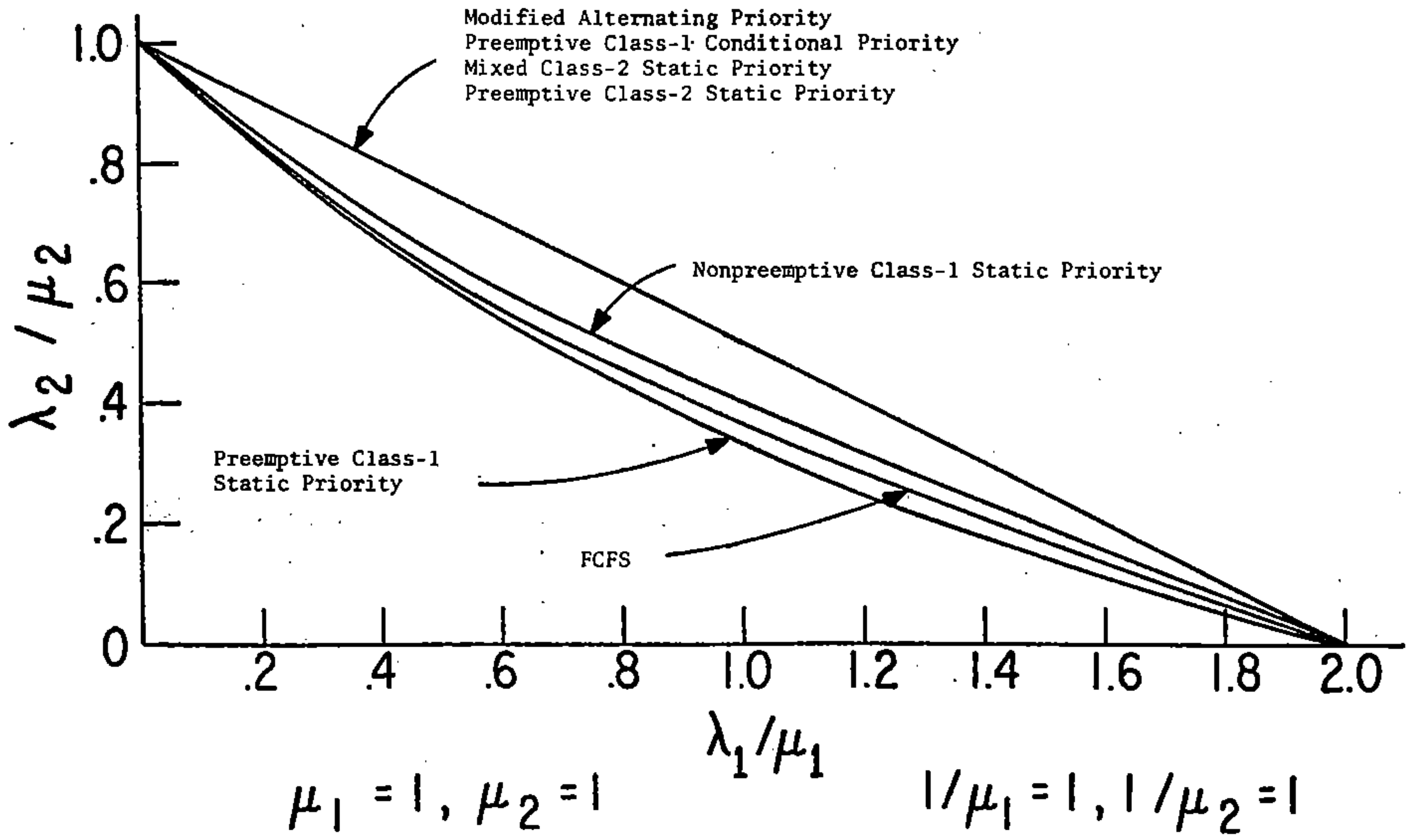


Figure 1. Saturation Points When Class-1 and Class-2 Jobs Have the same Mean Processing Times.

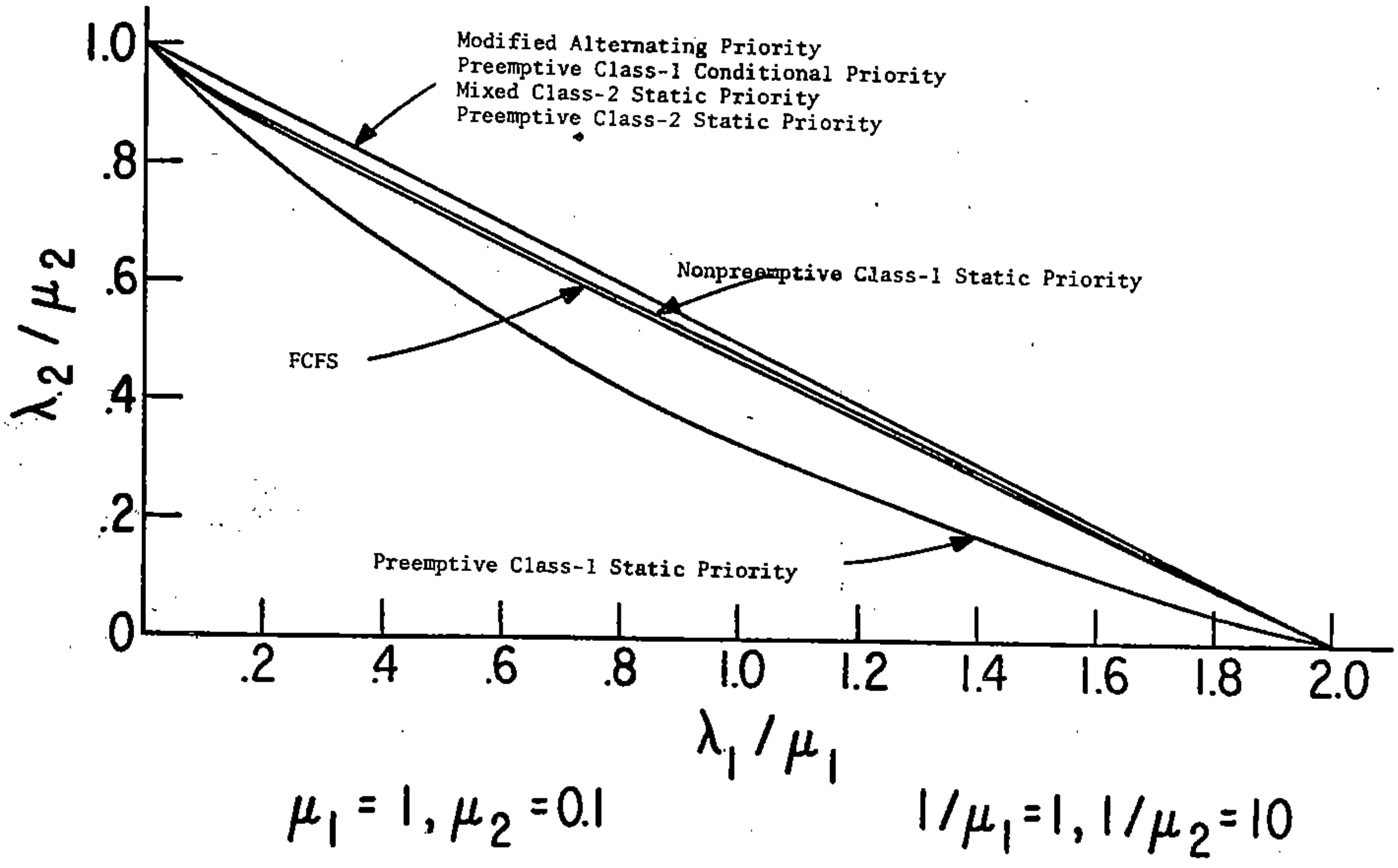


Figure 2. Saturation Points When Average Class-2 Processing Time Is Much Greater Than That For Class-1 Jobs.

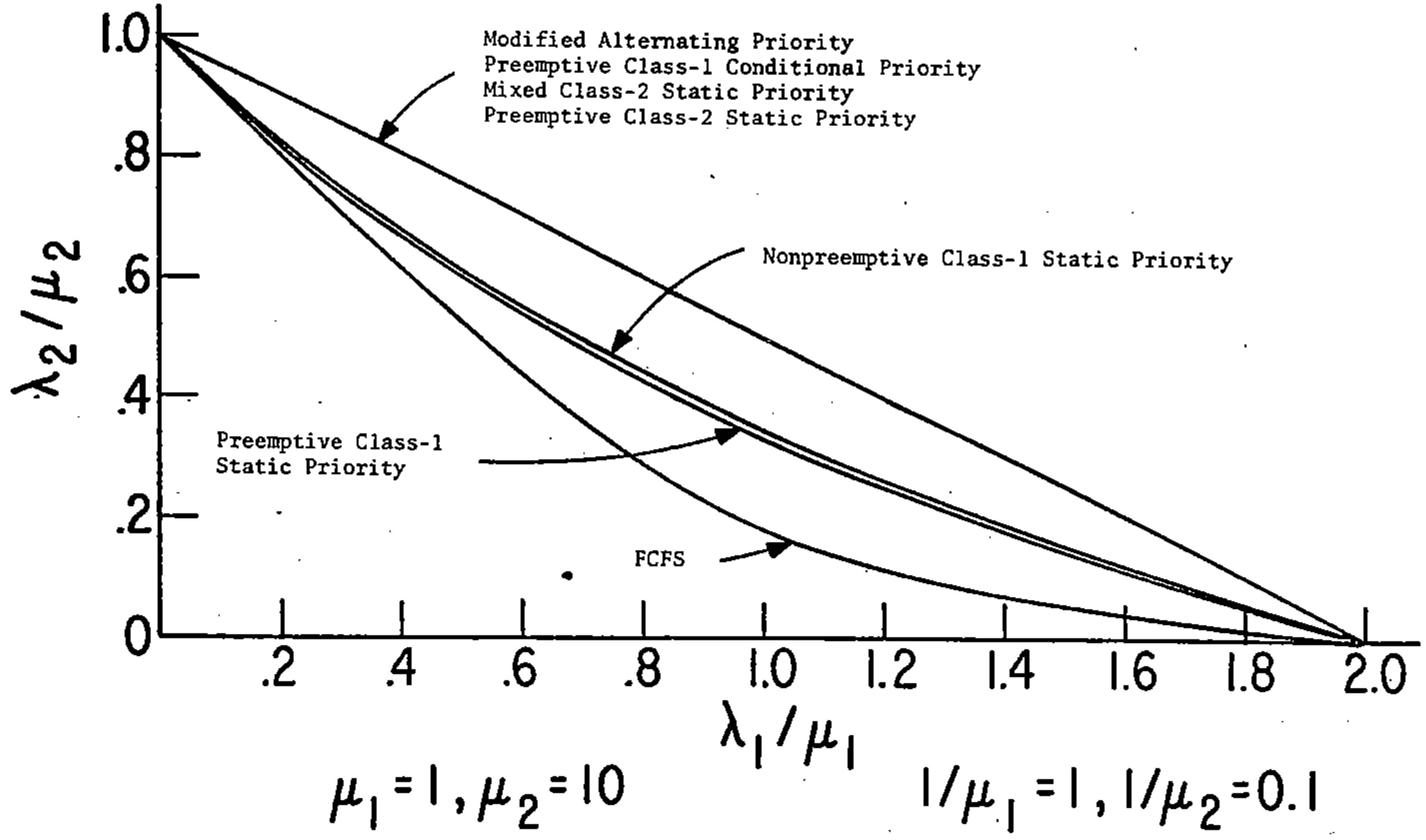


Figure 3. Saturation Points For Case In Which Expected Class-1 Service Times Are Much Larger Than Mean Class-2 Processing Times.

Condition	Ordering of Avg. Flow Times
$\mu_1/\mu_2 = 1/2$ or $E(P_1)/2 = E(P_2)$	$\bar{F}_4 = \bar{F}_5 = \bar{F}_6 = \bar{F}_7$
$\mu_1/\mu_2 < 1/2$ or $E(P_1)/2 > E(P_2)$	$\bar{F}_5 < \bar{F}_6 < \bar{F}_7 < \bar{F}_4$
$\mu_1/\mu_2 > 1/2$ or $E(P_1)/2 < E(P_2)$	$\bar{F}_4 < \bar{F}_7 < \bar{F}_6 < \bar{F}_5$

The above $\bar{F}_j$ terms assume some specified value for f_1 and input rate λ, and the meaning of each subscript j is given by:

(4) Preemptive Class-1 Conditional Priority Discipline
(5) Preemptive Class-2 Static Priority Discipline
(6) Mixed Class-2 Static Priority Discipline
(7) Modified Alternating Priority Discipline

Figures 4 and 5 show the average flow time for each discipline as a function of input rate λ for two different sets of job characteristics.

In Figure 4, the expected processing time for Class-2 jobs is three times the expected processing time for Class-1 jobs, and seventy-five per cent of the incoming jobs belong to Class-1. Figure 4 therefore corresponds to the situation in which jobs with small resource requirements have shorter running times than those with large resource requests.

Figure 5 illustrates the average flow time for the case in which Class-1 jobs on the average require three times the processing time of Class-2 jobs and where Class-1 jobs constitute only twenty-five per cent of the input stream. The job characteristics assumed in Figure 4 seem to be more realistic than those of Figure 5 but both cases are useful for the sake of comparison.

The job characteristics assumed in Figures 4 and 5 illustrate the relative loss of capacity for those disciplines which cannot achieve full capacity. In Figure 4, the maximum capacity is given by $\lambda_{max} = 2.67$; the FCFS, Nonpreemptive Class-1 Static Priority, and Preemptive Class-1 Static Priority disciplines have capacities which are respectively 92%, 93%, and 84% of full capacity for this case. The job parameters of Figure 5 result in the FCFS discipline saturating at 80% of full capacity, the Nonpreemptive Class-1 Static Priority at 86%, and the Preemptive Class-1 Static Priority at 84% of λ_{max}. It may be seen that the reduction in system capacity can be quite significant for reasonable job characteristics.

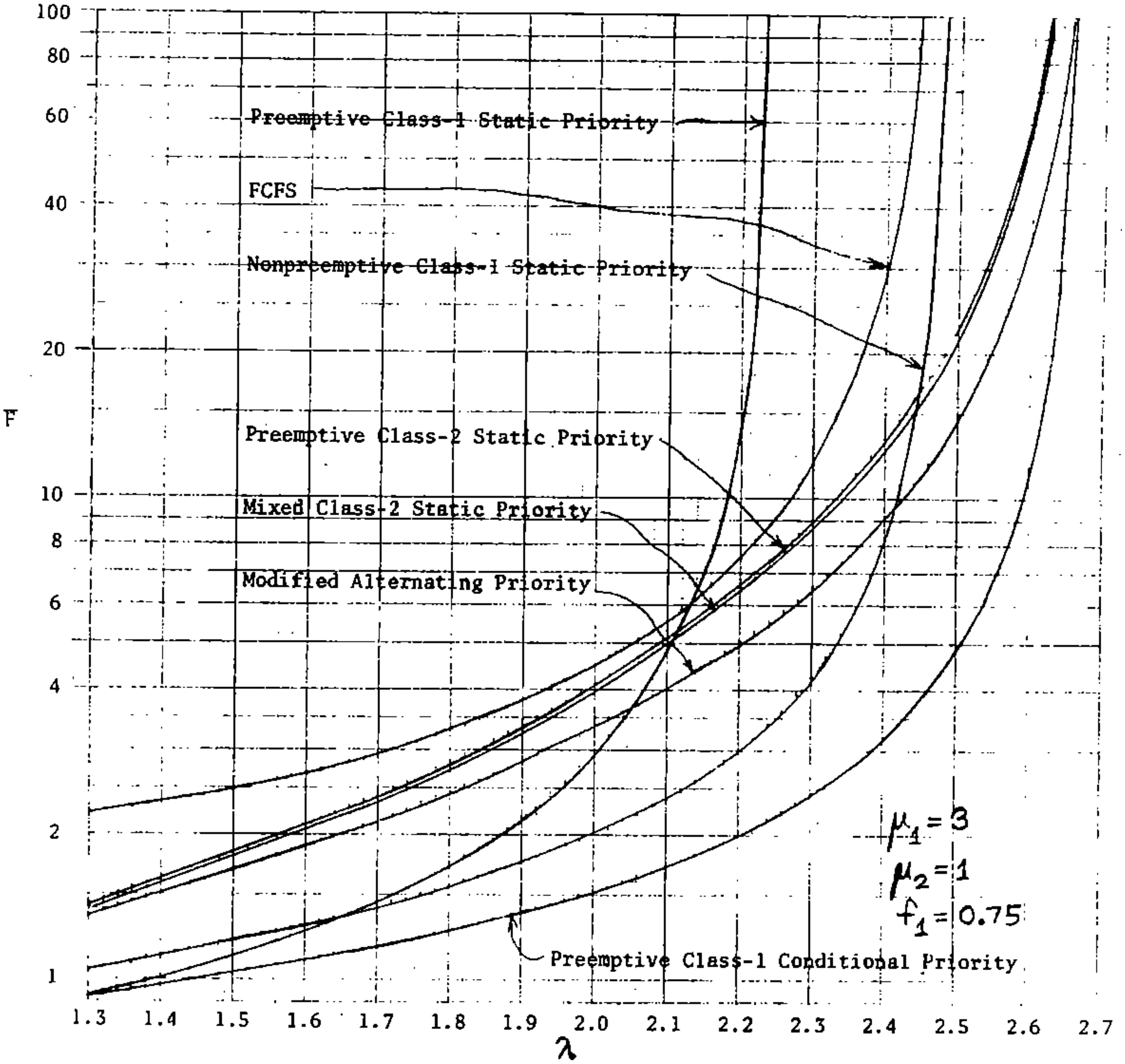


Figure 4. Average Flow Time Versus Input Rate

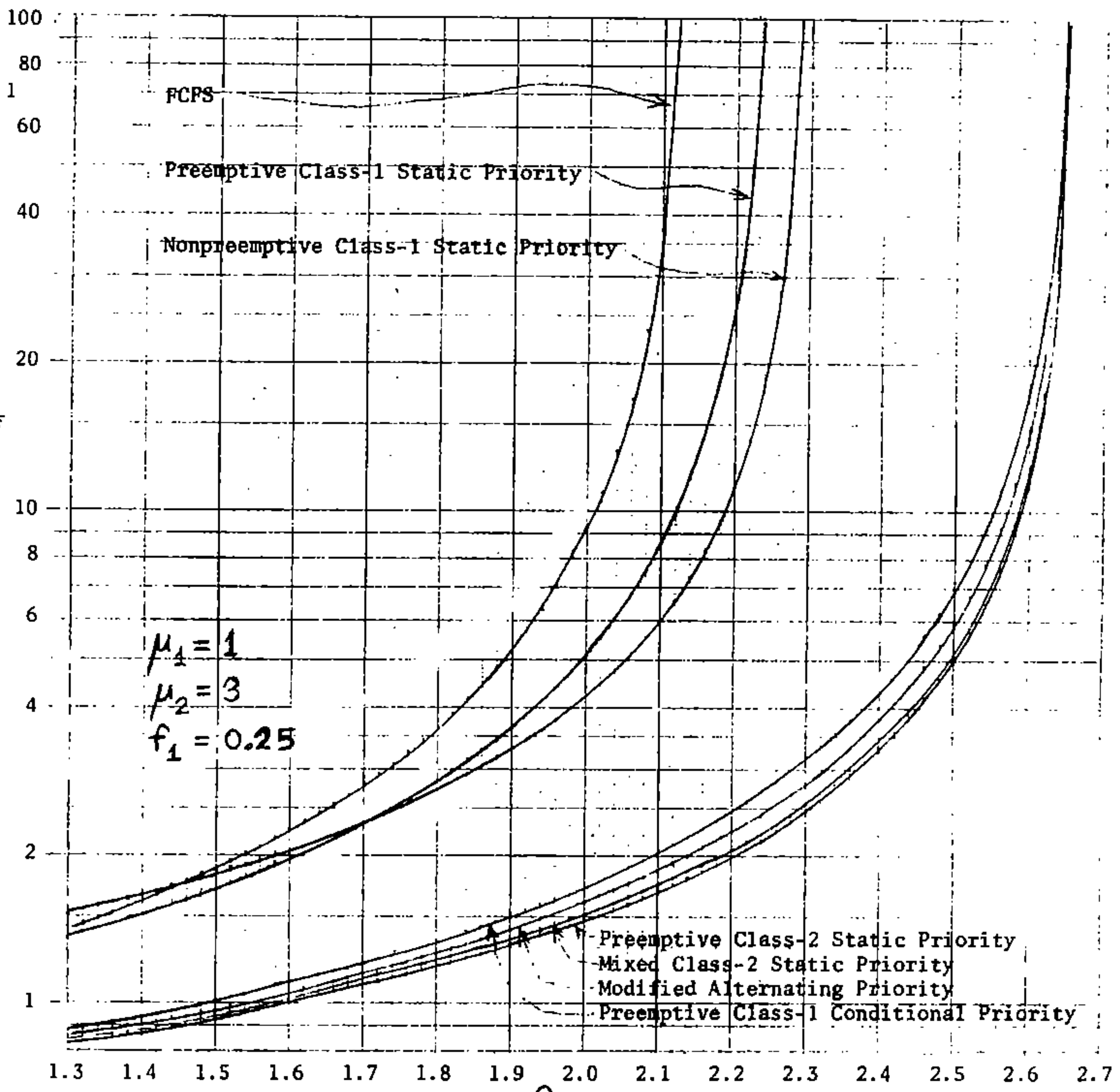


Figure 5. Average Flow Time Versus Input Rate.

The average flow time for small input rates are not shown in Figures 4 and 5 because these values are very nearly identical for all of the disciplines. At these low input rates, however, the disciplines which do not achieve full capacity may nevertheless have average flow times which are slightly better than those of the full capacity disciplines. Figures 4 and 5 demonstrate the superiority of the full capacity disciplines at the higher input rates.

Figures 4 and 5 clearly illustrate that the average flow times for the full-capacity disciplines can differ by a substantial amount. For the job stream characteristics of Figure 4, the Preemptive Class-1 Conditional Priority discipline clearly has the lowest average flow time of the full-capacity disciplines. In Figure 5 the Preemptive- and Mixed Class-2 Static Priority disciplines exhibit the lower average flow times.

The relative performance of the full-capacity disciplines may be summarized in word form as follows: The Preemptive Class-2 Static Priority rule and the Mixed Class-2 Static Priority discipline differ only slightly in average flow time, and both perform better than the other two full-capacity rules (in terms of average flow time) when the expected Class-2 processing time is less than half the average Class-1 processing time. The Modified Alternating Priority discipline is a good "compromise" rule in that it performs reasonably well regardless of the relationship between the expected processing times for the two job classes (i.e., while it never has the lowest average flow time for the general case, neither does it exhibit the highest average flow time of the full-capacity rules). The Preemptive Class-1 Conditional Priority rule is the best performer in terms of flow time when half the average Class-1 processing time is less than the expected Class-2 service time.

Observations and Conjectures

Results have been presented which describe the performance of a simple multi-resource queueing system, the Two CPU System With Two Units of Memory, under seven different scheduling rules. It was shown that there exists a capacity bound for this system which may be interpreted as the smallest input rate at which the system is guaranteed to saturate regardless of the scheduling rule that is employed. Of the seven scheduling rules examined, three were found to saturate at input rates less than that given by the capacity bound. For the four full-capacity scheduling rules, the ordering between the average flow times under each discipline was found to depend on the relationship between the average processing times of the two job classes.

An explanation will first be given for the relative orderings of the average flow times under the full-capacity disciplines. Half the expected Class-1 processing time, $E(P_1)/2$, may be considered to be the "average effective processing load" imposed on the system by a Class-1 job which is simultaneously processed with a second Class-1 job. The scheduling rules which give preference to Class-2 jobs, the Preemptive- and Mixed Class-2 Static Priority disciplines, perform better than the other two full-capacity disciplines when the average Class-2 processing time is less than the "average effective Class-1 processing load." Likewise, the Preemptive Class-1 Conditional Priority scheduling rule favors the Class-1 jobs and exhibits the lowest average flow time when the "expected effective Class-1 processing load" is smaller than the average Class-2 processing time. These results are consistent with the observed behavior of disciplines for single-server queueing systems in which the favoring of "short" jobs has the effect of reducing average flow time.

The capacity bound for the Two CPU System With Two Units of Memory was found as the solution of a Linear Program in which the constraints were found by application of Little's Theorem [5]. The solution set of state probabilities specified the proportion of time that the system should spend in processing the various job combinations in order to achieve full-capacity. It may be noted that the solution set of state probabilities was unique for this system under any given job stream characteristics and that the state probability associated with "One Class-1 Job" had a value of zero. This implies that this particular job combination is undesirable in terms of capacity, and it is intuitively reasonable to expect that a full-capacity rule will assign this lower priority than other job combinations. Consider the three priority orderings that could be given to the job combinations:

Class-1 Static Priority	Class-2 Static Priority	Class-1 Conditional Priority	
Two Class-1 Jobs	One Class-2 Job	Two Class-1 Jobs	(highest)
One Class-1 Job	Two Class-1 Jobs	One Class-2 Job	
One Class-2 Job	One Class-1 Job	One Class-1 Job	(lowest)

The Class-1 Static Priority ordering is the only one which is unable to attain full-capacity, a fact which may be explained by there being an undesirable job combination (One Class-1 Job) assigned a higher priority than some desirable job combination (One Class-2 Job).

Multi-resource queueing systems are interesting because a scheduling discipline must implement a decision rule for choosing the next combination of jobs to be processed (in effect, selecting the amount and type of concurrent processing to take place). A scheduling rule for a multiple-resource queueing system, even if it does not involve any overhead or inserted idle-time, may nevertheless be unable to attain full-capacity. Counter-intuitive behavior may result as a consequence of the characteristics of multi-resource queues; for example, the Class-1 Static Priority rules illustrated that giving better service to one job class could actually decrease system capacity.

The results for the Two CPU System With Two Units of Memory showed that more than one full-capacity scheduling rule may exist for a multi-resource queue. The Mixed Class-2 Static Priority rule and the Modified Alternating Priority rule demonstrated that a strictly preemptive rule is not needed to attain full-capacity; however, these rules strongly suggest that preemption may be needed in order to allow a desirable job combination to go into service when in fact some desirable job combination can be formed from the set of jobs in system. For example, it must be possible to preempt the lone Class-1 job left in system after the departure of a Class-1 job if there is some Class-2 job to be processed. It is obvious, however, that there do exist situations in which preemption is not needed in order for a discipline to attain full capacity; a convenient example of such a multiple-resource system is the multiple-processor queue in which each job requires the use of one of the c processors in system (where $c > 1$). The solution set of state probabilities for such a system is not unique, but every job combination involving fewer than c jobs has probability zero in the solution set of state probabilities. For this system, the "desirable" job combinations are those which involve c jobs, and all other job combinations may be thought to be "undesirable" because they have probability zero in the solution set. The job combination in progress defines the system state, and we may gain insights into why preemption is not necessary by examining the manner in which transitions between states can occur. If a desirable job combination is in service, the system will be able to make a transition to another desirable state upon the departure of a job whenever the queue is nonempty (i.e., whenever a desirable job combination can be formed from the jobs in system). Likewise, the arrival of a job to the system when an undesirable job combination is in service will be immediately processed, and once again a transition will occur to a desirable state if in fact a desirable state may be constructed using the job in system.

It is a difficult task to discover other multi-resource queues for which there exist full-capacity disciplines which do not employ preemption, and the material which appears below should be regarded as generalizations based on the experiences of the authors. The study of the Two CPU System With Two Units of Memory suggested that a full-capacity discipline should only process an undesirable job combination (i.e., one having probability zero in the solution set) when it is impossible to construct a desirable combination from the jobs in system. This conjecture, if valid, has rather strong implications for a full-capacity discipline. If a desirable job combination is being serviced and a departure occurs, it should be the case for a full-capacity discipline that (a) the jobs from the combination which still remain in system must themselves form a desirable combination, or (b) another desirable job combination must be able to go into service if in fact some desirable job combination can be formed from the set of jobs in system.

It seems likely that the necessary and sufficient conditions for achieving full capacity without preemption are related to both the solution set of state probabilities (which will not be unique in general) and the manner in which transitions can occur between various system states (i.e., job combinations) under the discipline. If we consider a discipline such as the FCFS rule, it is apparent that the rule does not allow certain state transitions which might be beneficial from the standpoint of system capacity. The value of preemption in achieving full capacity lies in the ability to make transitions between any two system states which are consistent with the set of jobs in system. One would expect that preemption is not necessary for full capacity only if the transitions which can occur without preemption are compatible in some sense with the "desirable" states in one of the solution sets of state probabilities for the system.

Notes on the Method of Analysis: An Example.

This paper summarizes results obtained independently by the authors in references [1] and [2]; the total collection of results and the associated derivations are extremely lengthy and will not be presented here. For purposes of illustration, an abbreviated analysis will be given for one of the scheduling rules, the Preemptive Class-1 Static Priority rule.

For a majority of the scheduling disciplines an approach involving a Semi-Markov process was used. Comparable approaches were used by authors such as Avi-Itzhak, Maxwell, and Miller [6] for treating single-server queueing models.

For the purpose of analysis, it is often sufficient to define system states that are more gross than those needed for a detailed description of the system at a point in time; the states, however, are so defined that the state transition process is Markovian. A particular result, for example the expected flow time for Class-1 jobs, is synthesized by conglomerating conditional results into an unconditional result by using the probabilities of finding the system in various mutually exclusive and exhaustive states. A Poisson arrival finds the system in a particular state with the same probability as the steady-state probability of the system being in that particular state (see Strauch [7]). The steady-state probabilities in turn are obtained by using results from the theory of Semi-Markov processes [8].

In each model, for certain states the system is shown to be equivalent to some other previously analyzed queueing system in some of its states. This equivalence is merely an operational one, and it is always with reference to a particular objective set forth. The 'equivalence technique' is advantageous since it allows a modular buildup of a complex system.

The Preemptive Class-1 Static Priority discipline has been chosen to illustrate this 'equivalence technique'; recall that this scheduling discipline gives Class-1 jobs preemptive priority over Class-2 jobs. This rule is somewhat easier to analyze than the remaining disciplines but nevertheless gives insights into the method of analysis. The computation of the expected flow time for Class-1 jobs is straightforward because the system operation as viewed by Class-1 jobs appears to be that of an M/M/2 queueing system under the FCFS rule. The more interesting problem is that of obtaining the expected flow time for Class-2 jobs.

Using standard queueing terminology, some results for previously analyzed systems are given below along with their defining parameters; these results will be utilized by means of the 'equivalence technique' mentioned earlier:

busy period of an M/G/1 system - Poisson input rate λ, general processing time P

busy period of an M/M/2 system - Poisson input rate λ, exponential service rate μ

delay cycle of an M/G/1 system - Poisson input rate λ, general processing time P, and initial delay period T_0 (for a more complete description, see reference [9], page 151)

The symbol '≡' will be used to define equivalences between the defining parameters given above and other chosen quantities. The system states of interest are shown below in Figure 6.

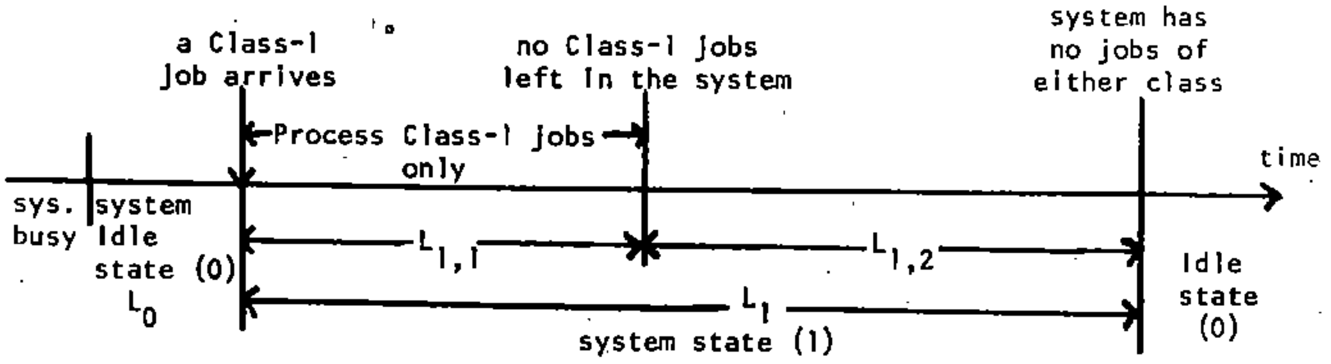


Figure 6a.

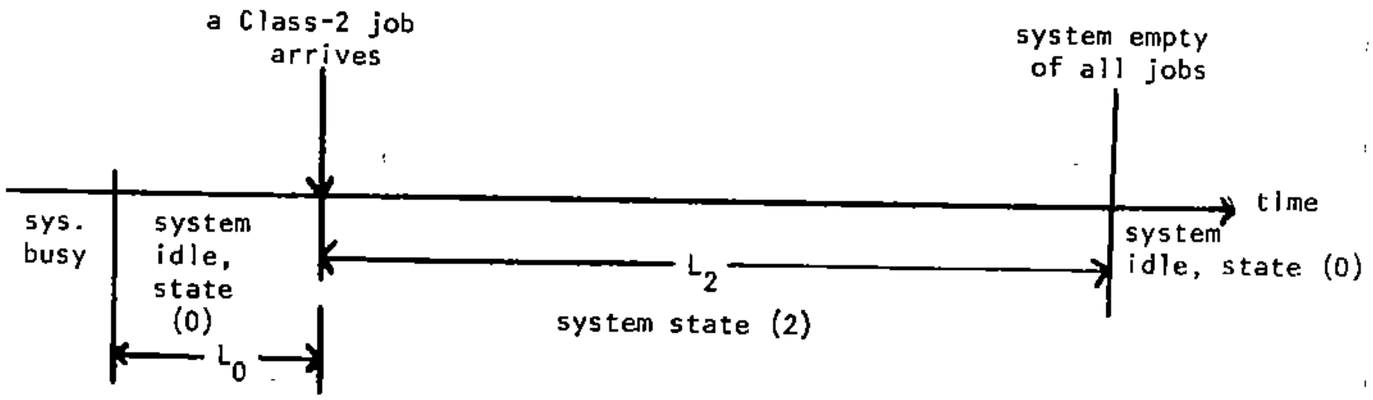


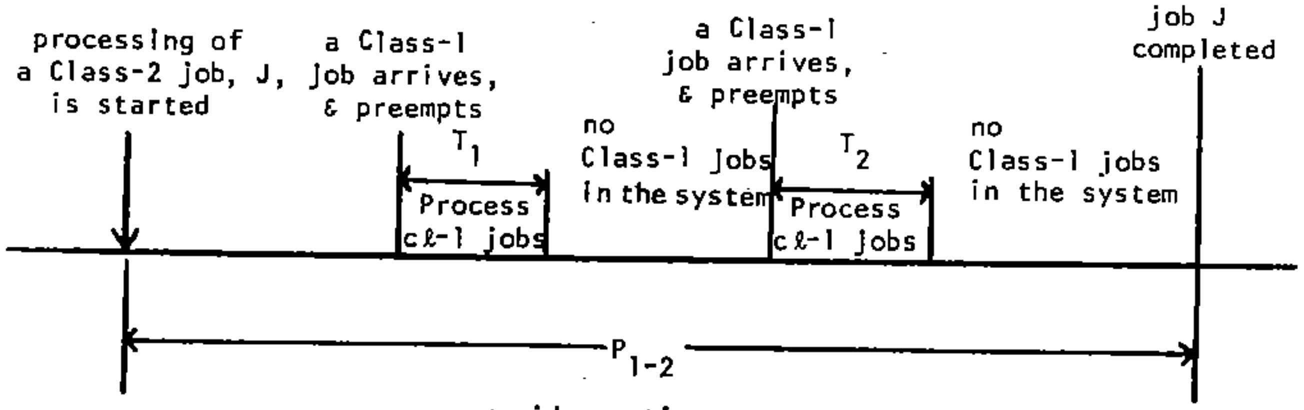
Figure 6b.

For the purpose of finding the expected Class-2 flow time $E(F_2)$, the equivalences are defined as given below:

State (1) is equivalent to the state of an M/G/1 system during a delay cycle, with parameters $\lambda \equiv \lambda_2$, $P \equiv P_{1-2}$, $T_0 \equiv L_{1,1}$,

State (2) is equivalent to the state of an M/G/1 system during a busy period, with parameters $\lambda \equiv \lambda_2$, $P \equiv P_{1-2}$,

where P_{1-2} is the residence time for a Class-2 job as illustrated in Figure 7 (for further explanation see reference [9], pages 169-173). The intervals T_i are the busy periods of an M/M/2 system with $\lambda \equiv \lambda_1$ and $\mu \equiv \mu_1$; the number of times that they occur has a geometric distribution with a complicated parameter. The first two moments of the random variable P_{1-2} are required, and these are easily obtained using some basic probabilistic arguments.


processing of a Class-2 job, J, is started
a Class-1 job arrives, & preempts
T_1
Process $c\ell-1$ jobs
no Class-1 jobs in the system
a Class-1 job arrives, & preempts
T_2
Process $c\ell-1$ jobs
no Class-1 jobs in the system
job J completed
P_{1-2}

residence time

Figure 7.

$$E(P_{1-2}) = \frac{(1+\lambda_1/\mu_1)}{\mu_2(1-\lambda_1/\mu_1)} \ .$$

$$E(P^2_{1-2}) = \frac{2(1+\lambda_1/\mu_1)^2}{\mu_2^2(1-\lambda_1/\mu_1)^2} + \frac{4\lambda_1(2-\lambda_1/\mu_1)}{\mu_1^2\mu_2(1-\lambda_1/\mu_1)^3} \ .$$

A Class-2 job, when it arrives, will find the system either in state (0), state (1), or state (2).
Let,

$E(F_2|s)$ = Expected flow time for a Class-2 arrival which finds the system in state-(s), where $s = 0, 1, 2$.

Using the system equivalencies defined earlier, we have

$$E(F_2|0) = E(P_{1-2})$$

$$E(F_2|1) = E(P_{1-2}) + \frac{\lambda_2 E(P^2_{1-2})}{2(1-\lambda_2 E(P_{1-2}))} + \frac{E(L^2_{1,1})}{2E(L_{1,1})}$$

$$E(F_2|2) = E(P_{1-2}) + \frac{E(P^2_{1-2})}{2E(P_{1-2})[1-\lambda_2 E(P_{1-2})]}$$

For the purpose of finding the first two moments associated with $L_{1,1}$, system equivalence is,

state (1,1) is equivalent to the state of a M/M/2 system during a busy period, with parameters $\lambda \equiv \lambda_1$ and $\mu \equiv \mu_1$.

Therefore,

$$E(L_{1,1}) = \frac{2}{\mu_1(1-\lambda_1/\mu_1)} \ ,$$

and

$$E(L^2_{1,1}) = \frac{4(2-\lambda_1/\mu_1)}{\mu_1^3(1-\lambda_1/\mu_1)^3} \ .$$

Various substitutions would lead to the expressions for the conditional expected flow times. These are then combined using the probabilities, $P(i)$, for finding the system in state i. The probability $P(i)$ is simply

$$\frac{E(L_i)}{E(\text{Busy Cycle length})}$$

Again using the same system equivalences as before,

$$E(L_0) = \frac{1}{\lambda_1+\lambda_2}$$

$$E(L_1) = \frac{E(L_{1,1})}{1-\lambda_2 E(P_{1-2})} = \frac{2}{\mu_1[1-\lambda_1/\mu_1-\lambda_2(1+\lambda_1/\mu_1)/\mu_2]}$$

$$E(L_2) = \frac{E(P_{1-2})}{1-\lambda_2 E(P_{1-2})} = \frac{(1+\lambda_1/\mu_1)}{\mu_2[1-\lambda_1/\mu_1-\lambda_2(1+\lambda_1/\mu_1)/\mu_2]}$$

$$E(\text{Busy Period}) = \frac{\lambda_1}{\lambda_1+\lambda_2} E(L_1) + \frac{\lambda_2}{\lambda_1+\lambda_2} E(L_2)$$

and

$$E(\text{Busy Cycle length}) = E(\text{Busy Period}) + E(\text{Idle Period}),$$

$$= \frac{(1+\lambda_1/\mu_1)}{(\lambda_1+\lambda_2)[1-\lambda_1/\mu_1-\lambda_2(1+\lambda_1/\mu_1)/\mu_2]} .$$

By taking ratios, we get the respective probabilities $P(i)$. Then, $E(F_2)$ is obtained by

$$E(F_2) = \sum_{i=0}^{3} E(F_2|i)P(i)$$

$$= \frac{\dfrac{2\lambda_1(2-\lambda_1/\mu_1)}{\mu_1^2(1+\lambda_1/\mu_1)} + \dfrac{(1+\lambda_1/\mu_1)^2\lambda_2}{\mu_2^2}}{(1-\lambda_1/\mu_1)[1-\lambda_1/\mu_1-\lambda_2(1+\lambda_1/\mu_1)/\mu_2]} + \frac{(1+\lambda_1/\mu_1)}{\mu_2(1-\lambda_1/\mu_1)} .$$

The technique outlined above is a powerful one which is relatively straightforward to apply. The experiences of the authors in utilizing this technique is that algebraic manipulations (note the cumbersome expression for $E(F_2)$) cause some difficulties but that the technique otherwise has a great deal of appeal.

Summary

Basic definitions and terminology have been given for a multi-resource queue, a type of congestion system in which arriving jobs require the simultaneous use of some combination of the system resources. Results have been presented for a simple example of a multi-resource queue, the Two CPU System With Two Units of Memory. These results illustrate that multi-resource queues exhibit a behavior which is counterintuitive in many cases and in particular that a scheduling rule for such a system must be concerned with the choice of a combination of jobs to be processed concurrently and not merely the choice of the next job to go into service. The manner in which a scheduling rule for a multi-resource queue favors the various job combinations has been shown to drastically affect both the capacity and average flow times for jobs. It is the belief of the authors that the properties of multi-resource queues may help to explain those instances in which the usual single-resource queueing models inadequately model the behavior of actual computer systems.

Bibliography

[1] Omahen, K. J. Analytic models of multiple resource systems. Ph.D. Thesis, Committee on Information Sciences, University of Chicago, June, 1973.

[2] Marathe, V. P. Priority queueing systems with simultaneous server requirements. Ph. D. Thesis, Operations Research, Cornell University, May, 1972.

[3] IBM Staff GPSS/360 User's Manual. IBM Corp., Form No. H20-0304.

[4] Habermann, A. N. Prevention of system deadlocks. Comm. ACM 12 (1969), 373-377, 385.

[5] Little, J. D. C. A proof of the queueing formula $L=\lambda W$. Operations Research 9 (1961), 383-387.

[6] Avi-Itzhak, B., Maxwell, W. L., and Miller, L. W. Queueing with alternating priorities. Operations Research 13 (1965), 306-318.

[7] Strauch, R. E. When a queue looks the same to an arriving customer as to an observer. Management Science 17 (1970), 140-141.

[8] Cox, D. R. Renewal Theory. London: Methuen, 1962.

[9] Conway, R. W., Maxwell, W. L., and Miller, L. W. Theory of Scheduling. Addison-Wesley, Reading, Massachusetts, 1967.