CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Purdue E-Pubs

Purdue University
Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1975

Capacity Bounds for Multi-Resource Queues

K. J. Omahen

Report Number:
75-137

Omahen, K. J., "Capacity Bounds for Multi-Resource Queues" (1975). *Department of Computer Science Technical Reports.* Paper 87.
https://docs.lib.purdue.edu/cstech/87

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

CAPACITY BOUNDS FOR MULTI-RESOURCE QUEUES

K. J. Omahen
Purdue University
West Lafayette, Indiana 47907

March 1975

CSD-TR 137

CAPACITY BOUNDS FOR MULTI-RESOURCE QUEUES

K. J. Omahen

Abstract

A multi-resource queueing system is a single congestion point associated with a number of resources which may be of different types. Arriving jobs require some combination of these resources simultaneously for the duration of the processing time of the job. The capacity bound for such a system, given the characteristics of the input stream, is defined as the smallest input rate which is guaranteed to saturate the system regardless of the scheduling rule which is employed. An algorithm for calculating this bound is presented which also specifies the proportion of time that the system should spend in processing various job combinations in order to achieve the capacity bound. The implications of this result are discussed for a number of resource allocation problems arising in computer systems.

CAPACITY BOUNDS FOR MULTI-RESOURCE QUEUES

K. J. Omahen

This paper examines a type of queueing model which is applicable to computer systems but which has not previously been described in the literature. This model, called a multi-resource queueing system, is characterized by a congestion point which is associated with a number of different resource types and by arriving jobs which require simultaneously some combination of the system resources. In contrast, the previously analyzed queueing models for computer systems have assumed that, even when there is a network of queues, each congestion point involves only a single resource type and jobs require exactly one unit of the scarce resource. Instances of multi-resource queueing systems have been independently treated by Omahen [1] and Marathe [2], and it has been found that these models are quite difficult to analyze when using methods from classical queueing theory. This paper focuses on one measure of performance, system capacity, and describes a method for determining a bound on the capacity of a system when given the characteristics of the job stream. In addition, this method also determines the proportion of time that the system should spend in processing various combinations of jobs if the capacity bound is to be achieved. The results are pleasing in their simplicity and offer insights into a number of different resource allocation problems arising in actual computer systems.

Notation and Terminology

This paper will first examine a basic model for a multi-resource queueing system and later demonstrate how the model may be easily extended to treat a number of realistic resource allocation problems. The presentation will begin by listing the assumptions included in the basic model for a multi-resource queueing system:

Model 1

(1) The congestion point is associated with several resource types, and an arbitrary number of units of each resource may be present in the system.

(2) Arrivals request service and require the simultaneous use of some combination of the system resources.

(3) Arriving jobs belong to one of several classes, where jobs in the same class have fixed (and identical) resource requirements and where each class has a specified interarrival time distribution and processing time distribution.

(4) There is no sharing of resources in the sense that an allocation quantity (i.e., the amount of a resource allocated to a job) is associated with at most one job at any instant in time; given that the resource requirements of a job are met, the job progresses at unit rate.

The following notation will be used to describe a multi-resource queueing system:

I = Number of Resource Types,

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

J = Number of Job Classes.

For each Job-Class-j, where $1 \leq j \leq J$, we further define:

λ_j = Arrival Rate for Class-j Jobs,

P_j = Non-negative Random Variable denoting a Class-j Processing Time,

$E(P_j)$ = Expected Class-j Processing Time,

$\overline{V}_j = (r_{j1}, r_{j2}, \ldots, r_{jI})$

= Resource Request Vector for the jth Job Class indicating that r_{j1} units of Resource-1 are required, r_{j2} units of Resource-2, etc., where $0 \leq r_{ji} \leq R_i$.

The overall arrival rate for jobs of all classes, λ, is given by

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

Given the overall arrival rate λ, the proportion of jobs which belong to a particular class is given by

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

Having given the necessary notation for describing the system resources and the characteristics of the various job classes, attention will now be focused on the manner in which combinations of jobs may be concurrently processed. 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:

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

The above requirement states that no combination of jobs can be concurrently processed if the total resource requirements exceed the number of resources present in the system.

S = Set of all (distinct) feasible job combinations for the system.

M = Number of Job Combinations contained in Set S.

If the amount of each resource type is finite and if each job class requires the use of some system resource, then M will have a finite value. The state of the system at any point in time will be described by the job combination which is being processed by the system at that moment. The feasible job combinations will be very important in the remainder of the paper because, once these combinations are known, we need not be concerned with the resource requirements of the individual jobs.

Properties of Multi-Resource Queues

Assume that stationary interarrival time and processing time distributions exist for each job class; if the system is operating under some arbitrary scheduling rule and if the system is nonsaturated at input rate λ, we define the following steady-state probabilities:

$\pi_0(\lambda)$ = Pr[System is idle],

$\pi_m(\lambda)$ = Pr[Job Combination m is in progress], where $1 \leq m \leq M$.

Suppose that certain of the job stream parameters are manipulated in the following manner: the parameters f_j, $j = 1, 2, \ldots, J$ are held constant, and the processing time distribution is also fixed for each class, but the overall arrival rate (and consequently the parameters of the interarrival time distributions) is allowed to vary. In such a situation the system will be described as having fixed job stream characteristics. For a given scheduling rule and fixed job stream characteristics, the capacity of the system will be defined as the overall input rate λ_{sat} such that

$$\lim_{\lambda \to \lambda_{sat}} \pi_0(\lambda) = 0.$$

The capacity bound λ_{max} for the multi-resource queueing system is defined as the infimum of the overall input rates at which saturation is guaranteed to occur regardless of the scheduling rule which is employed. Although it may not be obvious at this point, it is very possible that saturation will occur under a given scheduling rule at overall input rates less than that given by the capacity bound for a given set of job stream characteristics. If, however, the capacity λ_{sat} under a specified rule is equal to the capacity bound λ_{max} for all job stream parameters, the scheduling discipline is said to be a full-capacity rule.

If we consider the properties of the multi-resource queueing system, it becomes apparent that the system will be characterized by a variable-service rate which is a function of both the number and types of jobs which being concurrently processed at any instant, i. e., the system state given by the job combination in service. An interesting consequence of the above feature is that the performance of a scheduling rule is a function of the manner in which combinations of jobs are chosen for servicing; in reference [1] the concept of assigning priorities to job combinations was found to be a useful for describing various scheduling rules for multi-resource systems. Furthermore, a scheduling rule may be unable to achieve full capacity even though there is no overhead in switching between jobs. In the case of a single-server queueing system, any discipline which involves no overhead in switching between jobs and no inserted idle time will be a full-capacity rule.

A method for determining the capacity bound for a multi-resource queueing system under fixed job stream characteristics will be next presented. It will be seen that this will involve determining the proportion of time that the system should spend in processing various job combinations.

THEOREM 1. Assume that the relative input rate f_j and expected processing time $E(P_j)$ are constants for each job class-j. The capacity bound λ_{max} for the system is the solution of the following Linear Program:

$$\lambda_{max} = \underset{\pi_m(\lambda)}{\text{Max}} \sum_{m=1}^{M} C_m * \pi_m(\lambda)$$

$$\text{for } 1 \leq m \leq M$$

such that

$$\sum_{m=1}^{M} \pi_m(\lambda) = 1,$$

and

$$\sum_{m=1}^{M} A_{m,j} * \pi_m(\lambda) = 0 \qquad \text{for } 1 \leq j \leq J-1,$$

where

$$C_m = \left\{ \sum_{j=1}^{J} n_{mj} \right\} \Big/ \left\{ \sum_{j=1}^{J} f_j * E(P_j) \right\} \qquad \text{for } 1 \leq m \leq M$$

and

$$A_{m,j} = n_{mj}/[f_j * E(P_j)] - n_{m,j+1}/[f_{j+1} * E(P_{j+1})] \qquad \text{for } 1 \leq m \leq M \text{ and } 1 \leq j \leq J-1.$$

PROOF. Assuming that there is zero overhead in switching between jobs, the following equation must hold when the system is nonsaturated at input rate λ:

$$\pi_0(\lambda) + \sum_{m=1}^{M} \pi_m(\lambda) = 1, \quad \text{where } \pi_0(\lambda) \geq 0 \text{ and } \pi_m(\lambda) \geq 0 \text{ for m=1 to M.}$$

A conservation equation may be obtained for each job class by applying Little's Equation [3] to the processor system; the conservation equation may be stated as follows:

[Expected no. of Class-j jobs in progress] =

[Arrival rate for Class-j jobs] * [Expected Class-j Processing Time].

Therefore, the set of state probabilities must also satisfy the conservation equation for each Class-j, where $1 \leq j \leq J$; the Class-j equation appears below:

$$\sum_{m=1}^{M} n_{mj} * \pi_m(\lambda) = \lambda f_j E(P_j).$$

If the system is operating under some scheduling rule j, the system capacity λ_{maxj} will be the input rate at which the following equations hold:

$$\lim_{\lambda \uparrow \lambda_{maxj}} \sum_{m=1}^{M} \pi_m(\lambda) = 1, \quad \text{and} \quad \lim_{\lambda \uparrow \lambda_{maxj}} \pi_0(\lambda) = 0.$$

The capacity bound λ_{max} for this system is the infimum of the input rates at which the system is guaranteed to saturate, and this bound will be the largest

capacity that could be achieved by any scheduling rule that might be used. It follows that the capacity bound λ_{max} will be the largest input rate λ for which a set of state probabilities may be found which satisfy each of the J conservation equations and also the equation

$$\sum_{m=1}^{M} \pi_m(\lambda) = 1.$$

The capacity bound then becomes the solution of the following Linear Program:

$$\lambda_{max} = \underset{\lambda,\pi_m(\lambda),}{\text{Max}} \lambda$$

$$\text{for } 1 \leq m \leq M$$

such that

$$\sum_{m=1}^{M} \pi_m(\lambda) = 1,$$

and

$$\sum_{m=1}^{M} n_{mj} * \pi_m(\lambda) = \lambda f_j E(P_j) \quad \text{for every Class-j such that } 1 \leq j \leq J.$$

Note that λ is a dependent variable in this formulation of the problem because the value of λ is completely determined by the values assigned to the state probabilities. Therefore, the statement of the problem much be modified in order to be amenable to solutions by means of existing Linear Programming software. The procedure described below is intended to remove λ from the formulation of the Linear Programming problem. If one equates the sums of the left- and right-hand sides, respectively, of the J constraint equations obtained by application of Little's Theorem, the following equation results:

$$\sum_{m=1}^{M} \pi_m(\lambda) * \sum_{j=1}^{J} n_{mj} = \lambda * \sum_{j=1}^{J} f_j * E(P_j) .$$

Solving for λ, one obtains:

$$\lambda = \sum_{m=1}^{M} C_m * \pi_m(\lambda) \quad , \quad \text{where} \quad C_m = \left\{ \sum_{j=1}^{J} n_{mj} \right\} \Big/ \left\{ \sum_{j=1}^{J} f_j * E(P_j) \right\} \quad \text{for } 1 \leq m \leq M.$$

The above equation for λ represents the objective function to be maximized, and the problem now becomes one of maximizing the expected amount of concurrent processing in the system. Each of the J constraints can also be solved for λ, and the jth equation (from Little's Theorem) is given by

$$\lambda = [1/(f_j * E(P_j))] \sum_{m=1}^{M} n_{mj} * \pi_m(\lambda), \quad \text{where } 1 \leq j \leq J.$$

By equating the jth and (j+1)th representations of λ, where $1 \leq j \leq J-1$, and rearranging terms, one obtains the J-1 constraint equations given below:

$$\sum_{m=1}^{M} A_{m,j} * \pi_m(\lambda) = 0 \quad \text{where } A_{m,j} = n_{mj}/[f_j * E(P_j)] - n_{m,j+1}/[f_{j+1} * E(P_{j+1})]$$

$$\text{and } 1 \leq j \leq J-1$$

The statement of the Theorem then follows directly. Q.E.D.

The above theorem specifies a method for determining the capacity bound for the system, given fixed job stream characteristics. There are a number of observations which can be made concerning the implications of the theorem. Because the statement of the LP problem contains J constraint equations, at most J of the variables (i.e., the $\pi_i(\lambda)$ terms) will be non-zero in value. The set of values found for the $\pi_i(\lambda)$ terms will be referred to as <u>the solution set of state probabilities</u>, and each $\pi_i(\lambda)$ value specifies the (long-run) proportion of time that the system should spend in processing the <u>i</u>th job combination in order to achieve the capacity bound. One additional warning must be included: there is no guarantee that the solution will be unique. If the maximum occurs at more than one extreme point, it will also occur at every convex combination of those points.

The solution set of state probabilities gives insights into the manner in which the various job combinations should be chosen by the scheduling rule for the multi-resource system. The scheduling discipline should give preference to the set of job combinations which includes every combination such that $\pi_i(\lambda)$ is

nonzero in the solution to the LP problem. Conversely, those combinations assigned probabilities of zero in the solution are "undesirable" combinations which should be avoided whenever possible. The notion of assigning priorities to job combinations rather than to job classes follows naturally for a multi-resource queue since the scheduling decision involves the choice of a combination of jobs for processing. If a discipline is to achieve full-capacity, one would expect that the "desirable" combinations be given higher priority than the "undesirable" ones.

An alternative view may be taken to arrive at the representation for the capacity bound. In reference [2], Marathe suggested a measure of performance for multi-resource queues called load factor and defined as the (steady-state) proportion of time that the system is busy (non-idle). In that study of certain multi-resource queues, it was found that two disciplines under identical loads could have load factors which varied greatly. The load factor may be considered to be a measure of the concurrent processing which takes place, where smaller load factor implies greater concurrency. Suppose that we have a multi-resource queue under nonsaturated operating conditions. If the job stream characteristics are fixed as described previously, there will be some lower bound for the load factor that could be achieved by any discipline. Define:

$\rho_{min}(\lambda)$ = greatest lower bound for the load factor at input rate λ, given fixed job stream characteristics and arbitrary scheduling rule.

Using the terminology defined for the previous theorem, the above quantity can be determined as described below:

COROLLARY The quantity $\rho_{min}(\lambda)$ is found as the solution of the following Linear Programming problem:

$$\rho_{min}(\lambda) = \underset{\pi_m(\lambda)}{\text{MIN}} \sum_{m=1}^{M} \pi_m(\lambda)$$

$$\text{for } 0 \leq m \leq M$$

subject to the following constraints

$$\pi_0(\lambda) + \sum_{m=1}^{M} \pi_m(\lambda) = 1,$$

and

$$\sum_{m=1}^{M} n_{mj}*\pi_m(\lambda) = \lambda*f_j*E(P_j) \quad \text{for every Class-j such that } 1 \leq j \leq J.$$

Sketch of Proof. Essentially identical to that for the previous theorem, except that we instead want to minimize the probability of the system being busy.

Given the above result, the capacity bound λ_{max} is the input rate λ such that

$$\lim_{\lambda \uparrow \lambda_{max}} \rho_{min}(\lambda) = 1$$

The Corollary should be interpreted as follows: given the characteristics of the job stream arriving at a multi-resource queue, the steady-state system is guaranteed to be busy with probability greater than or equal to $\rho_{min}(\lambda)$. It is of course possible that the system will be busy with steady-state probability greater than $\rho_{min}(\lambda)$ for a specified scheduling rule.

An alternative multi-resource queueing model will be next presented which is particularly relevant to contemporary computer system architecture; the following assumptions form the basis for this model:

Model 2

(1) The congestion point is associated with several resource types, and an arbitrary number of units of each resource may be present in the system.

(2) Arrivals request service and require the simultaneous use of some combination of the system resources.

(3a) There is one class of jobs having some specified interarrival time distribution. Furthermore, a set of job states are defined for these jobs, where each state has fixed resource requirements and a processing time distribution to describe the time between state transitions. The transitions between states are assumed to take place immediately upon entry of the job into the system and upon the completion of processing in a particular job state; the next state is chosen via a Markov process described by a routing transition matrix.

(4) There is no sharing of resources in the sense that an allocation quantity is associated with at most one job at any instant in time; given that the resource requirements of a job-state met, the job progresses at unit rate.

By comparison with Model 1, it may be seen that Model 2 differs only in the third of the above assumptions. The notation presented for the Theorem has the following meaning for Model 2:

λ = Overall Arrival Rate for jobs,

I = Number of Resource Types,

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

J = Number of different Execution States possible for jobs.

For each Job-State-j, where $1 \leq j \leq J$, define:

P_j = Non-negative Random Variable denoting a State-j Processing Time (i.e., time between entry into State-j and the instant at which the next transition takes place),

$E(P_j)$ = Expected State-j Processing Time,

$\overline{V}_j = (r_{j1}, r_{j2},\ldots, r_{jI})$ = resource request vector for a job in State-j.

It is assumed that there are two states in addition to the J Execution States mentioned above: these correspond to an <u>initial state</u> (State-0) for arriving jobs and a terminating state (State-(J+1). Upon entering the system an arrival immediately makes a transition from State-0 to one of the other states, and a job leaves the system upon exit to State-(J+1). A <u>routing transition matrix</u> Q describes the manner in which the state transitions take place, where an element $q(k,\ell)$ of the matrix is defined to be:

$q(k,\ell)$ = Probability that a job in state-k goes into state-ℓ at the next transition, where $0 \leq k \leq J$, $1 \leq \ell \leq J+1$, $0 \leq q(k,\ell) \leq 1$, and

$$\sum_{k=1}^{J+1} q(k,\ell) = 1 \qquad k = 0, 1,\ldots, J.$$

Each job will sequentially enter a number of different execution states, and it would be desirable to know the effective rate at which jobs enter these various

execution states. Define:

f_j = Expected Number of Times that a job will enter State-j.

The set of f_j terms must satisfy the following relations:

$$f_j = q(0,j) + \sum_{k=1}^{J} f_k * q(k,j) \qquad \text{where } 1 \leq j \leq J.$$

Referring to the terms on the right-hand side of the equation, it may be seen that the expected number of times that a job enters state-j equals (a) the expected number of times that state-j is entered from the initial state, plus (b) the expected number of times that state-j is entered from each state-k, where $1 \leq k \leq J$. Given the transition matrix Q, we may determine the corresponding values for the f_j terms by solving the set of simultaneous linear equations. In contrast to Model 1, the only restriction on the f_j terms for Model 2 is that they be required to have non-negative values since they are no longer probabilities.

When specifying the combinations of jobs which may be concurrently processed, it will be necessary to describe the number of jobs in each of the possible states that are involved in the combination. A job combination may then be described by

$[n_{m1}, n_{m2}, \ldots, n_{mJ}]$ = Job-Combination-m consisting of n_{m1} jobs in State-1, n_{m2} jobs in State-2, etc.

As in Model 1, a feasible job combination is defined as a job combination whose total resource requirements can be met by the resources in system, and we again define:

M = Number of distinct feasible job combinations.

The Theorem and Corollary apply to Model 2 by merely replacing the words "Job-Class-j" with "Job-State-j". Model 2 appears to be a generalization of the Markovian Network Models which are frequently used to analyze computer systems. Model 2, however, embodies a more complex form of resource allocation than the usual Markovian Network Model in which it is assumed that a job in any particular state requires one unit of exactly one resource type.

There are two additional forms of resource allocation which will next be considered: resource-sharing and resource-multiplexing. These two cases will be discussed below in detail, but informally these two situations may be thought to correspond to the usual cases of "segment-sharing" and "processor-sharing" as take place in multiprogramming computer systems. The effect of these forms of resource allocation will be discussed with respect to Model 2.

Resource-sharing is intended to deal with the situation in which a quantity of some resource is allocated simultaneously to two or more jobs in such a way as to satisfy the resource requirements of each of those jobs. For example, reentrant code segments or read-only data segments in a multiprogramming system may be shared in this manner. In this case of resource-sharing, the jobs simultaneously allocated the segment(s) each proceed at unit rate, assuming that all other resource requirements are met.

Consider the effect of segment-sharing for jobs in some combination-m:

$[n_{m1}, n_{m2}, \ldots, n_{mJ}]$ = job-combination-m consisting of n_{m1} jobs in state-1, n_{m2} jobs in state-2, etc.

Suppose that the ith resource, say primary memory, corresponds to a resource affected by the resource-sharing. Define:

w_{mi} = Total amount of the ith resource required by the jobs in combination-m.

If resource-sharing is taking place, it should be expected that

$$w_{mi} < \sum_{j=1}^{J} n_{mj} * r_{ji} \quad ,$$

where the right-hand term represents the resource requirement with no sharing of any allocation quantity. The net effect of this form of resource-sharing is that the number of feasible job combinations may possibly increase. If this increase does take place, the additional job combinations will offer greater opportunities for concurrency (in turn, the capacity bound may then take on a larger value).

Resource-multiplexing is aimed at cases in which a quantity of some resource is allocated to two or more jobs, but where these jobs do not progress at unit rate because that allocation quantity is multiplexed (over time) among the jobs. An obvious example is the sharing of a processor by jobs in primary memory of a multiprogramming computer system; whether a round-robin discipline

or some other rule is employed, each job appears to receive only a fraction of the processor power during a given interval. At this point some additional terminology should be introduced. Define:

e_{mj} = Average rate at which each job in state-j progresses during the servicing of job-combination-m,

= Steady-state probability that a job in state-j is progressing at unit rate at a random instant, given that job-combination-m is in service.

If one considers a feasible job-combination-m, it becomes apparent that the following relation must hold when resource-multiplexing is taking place for the ith resource type:

$$\sum_{j=1}^{J} n_{mj} * r_{ji} * e_{mj} \leq R_i .$$

That is, during an interval in which the mth job combination is in service, the multiplexed resource usage over time cannot exceed the total available resources over that same interval.

The two additional forms of resource allocation will next be included in a variation of Model 2; the assumptions of Model 3 are given below. When compared to Model 2, Model 3 may be noted to have a slightly modified fourth assumption, and additional fifth and sixth assumptions to deal with resource-sharing and resource-multiplexing, respectively.

Model 3

(1) The congestion point is associated with several resource types, and an arbitrary number of units of each resource may be present in the system.

(2) Arrivals request service and require the simultaneous use of some combination of the system resources.

(3a) There is one class of jobs having some specified interarrival time distribution. Furthermore, a set of job states are defined for these jobs, where each state has fixed resource requirements and a processing time distribution to describe the time between state transitions. The transitions between states are assumed to take place immediately upon entry of the job into the system and upon the completion of processing in a particular job state; the next state is chosen via a Markov process described by a routing transition matrix.

(4a) If the resource requirements of a job are met and if the necessary quantity of each resource is allocated solely to that job, the job progresses at unit rate (i.e., no resource-sharing or resource-multiplexing).

(5) If resource sharing takes place and a quantity of some resource is simultaneously allocated to two or more jobs (thereby satisfying the resource requests of those jobs, and assuming no resource-multiplexing), those jobs progress at unit rate.

(6) If resource-multiplexing (with or without resource-sharing) takes place for certain jobs, those jobs progress at a rate (less than one in value) which is a function of the job combination in service and the job-state.

The main theorem presented earlier in the paper requires only minor modifications for Model 3. As noted earlier, resource sharing will require that the set of feasible job combinations calculated in a slightly different manner. In addition, the e_{mj} terms must be included in the statement of the Theorem as shown below:

THEOREM 2. Assume that the relative input rate f_j (calculated from the state transition matrix Q) and expected processing time $E(P_j)$ are constants for each job-state-j. The capacity bound λ_{max} for the system is the solution of the following Linear Program:

$$\lambda_{max} = \underset{\pi_m(\lambda)}{\text{Max}} \sum_{m=1}^{M} C_m * \pi_m(\lambda)$$

$$\text{for } 1 \leq m \leq M$$

such that

$$\sum_{m=1}^{M} \pi_m(\lambda) = 1,$$

and

$$\sum_{m=1}^{M} A_{m,j} * \pi_m(\lambda) = 0 \qquad \text{for } 1 \leq j \leq J-1$$

where

$$C_m = \left\{ \sum_{j=1}^{J} n_{mj}*e_{mj} \right\} \Big/ \left\{ \sum_{j=1}^{J} f_j*E(P_j) \right\} \quad \text{for } 1 \leq m \leq M$$

and

$$A_{m,j} = \left[n_{mj}*e_{mj} \right] \Big/ \left[f_j*E(P_j) \right] - \left[n_{m,j+1}*e_{m,j+1} \right] \Big/ \left[f_{j+1}*E(P_{j+1}) \right]$$

for $1 \leq m \leq M$ and $1 \leq j \leq J-1$.

Sketch of Proof. The derivation is nearly identical to Theorem 1; the only change is that the application of Little's Theorem for jobs in state-j gives the following constraint equations:

$$\sum_{m=1}^{M} n_{mj}*\pi_m(\lambda)*e_{mj} = \lambda*f_j*E(P_j) \quad \text{for every job-state-j such that } 1 \leq j \leq J.$$

In the above statement, the convention is adopted whereby the rate at which a job in state-j progresses when job-combination-m is in service is taken to be unity in the absence of resource-sharing involving that job; that is,

$e_{mj} \equiv 1$ (job progresses at unit rate) if there is no resource-sharing.

It may be seen that it is reasonable to use the interpretation of e_{mj} as the "probability that a job in state-J is progressing at unit rate at a random instant during the servicing of combination-m."

Discussion and Examples

The capacity bound for a multi-resource system may be calculated using an alternative algorithm which is simpler but which does not supply any information concerning the manner in which the bound may be achleved. Consider Model 1 introduced earlier; at saturation at least one of the resource types will be 100% utilized. For each resource type, one may find the smallest input rate which would cause that resource to be fully utilized. The capacity bound then becomes the minimum of those input rates calculated to cause full utilization for the resource types in system. Using the notation introduced for Model 1, we proceed as follows. Define:

$\lambda(i)$ = input rate which causes the $\underline{i}$th resource type to be 100% utilized, where $i = 1, 2, \ldots, I$.

The expected number of units of the $\underline{i}$th resource which are allocated to jobs at input rate $\lambda(i)$ must be less than or equal to R_i, the total number of units of resource-i present in system. That is, the following relation must hold:

$$\sum_{j=1}^{J} \lambda(i)*f_j*E(P_j)*r_{ij} \leq R_i \ .$$

It follows that

$$\lambda(i) = R_i / \left\{ \sum_{j=1}^{J} f_j*E(P_j)*r_{ij} \right\}$$

and

λ_{max} = MIN[$\lambda(1)$, $\lambda(2)$,..., $\lambda(I)$] where MIN denotes the smallest of the I arguments.

This method, however, offers no insights into how a scheduling rule should go about giving preferences to the possible job combinations which might be serviced.

Two specific systems will be examined next in order to illustrate the application of the general method described in Theorems 1 and 2. These examples will demonstrate that one may obtain a number of insights into system performance through the use of the method.

Two CPU System With K Units of Memory

A bound on the capacity of a multi-resource system of the type described by Model 1 will now be determined for the case in which the system resources consist of two CPUs and K units of memory. The system resources are therefore described by the notation given below:

$R_1 = 2$,
= Number of CPUs in the system.

$R_2 = K$,
= Number of blocks of memory in the system

The jobs arriving to the system will require the simultaneous use of one Central Processing Unit and a number of units of memory which varies as a

function of the class to which the job belongs. There are K classes of arrivals, and these classes correspond directly to the size of the memory request associated with jobs within the class. The characteristics of each job class are described by the notation given below; for jobs of Class-j, where integer j is such that $1 \leq j \leq K$, define:

λ_j = Arrival Rate for Class j Jobs,

P_j = Non-negative Random Variable denoting the Processing Time required by a Class-j Job,

$E(P_j)$ = Expected Class-j Processing Time,

$(1,j)$ = Request Vector for a Class-j Job indicating that one CPU and j units of memory are required simultaneously by each Class-j Job.

The overall arrival rate will the sum of the arrival rates for the individual job classes, and this quantity will be indicated as shown below:

$$\lambda = \sum_{j=1}^{K} \lambda_j = \text{Overall Arrival Rate for Jobs of all Classes.}$$

Given the above representation for the overall arrival rate, the proportion of jobs from the overall input stream which belong to a particular class will be designated as follows:

$f_j = \lambda_j/\lambda$

= Proportion of overall input stream which represents the contribution of Class-j Jobs, where $1 \leq j \leq K$.

Before presenting a theorem for the capacity bound of the above system, some observations will be made concerning the characteristics or major features of this problem. The resource requirements of the various job classes are such that at most two jobs may be concurrently processed, and the memory requests associated with the two jobs must not exceed the total number of units of memory in the system. In determining the bound on system capacity, the primary objective will be to maximize the amount of concurrent processing which takes place in the system. This problem, in turn, is equivalent to "balancing" the workload associated with large jobs with that for small jobs. It may be noted that the larger the memory requirements of a job class, the smaller the degree of freedom in the choice of small jobs in order to achieve concurrent processing; for example, jobs of Class (K-1) can only be processed

with Class-1 jobs, but Class-(K-2) jobs may be run with either Class-1 or Class-2 jobs. "Small" jobs which require no more than half of the total available memory pose no serious problems because any pair of these "small" jobs can always be processed concurrently.

THEOREM 3. Assume that there is a stationary distribution for the processing times associated with each job class and that the jobs of each Class-j constitute a fixed proportion f_j of the overall input stream. The capacity bound λ_{max} for the two CPU system with K units of memory is given by

$$\lambda_{max} = \left\{ \sum_{j=1}^{J} f_j E(P_j) + \sum_{j=1}^{L} \theta_{K-j} + f_K E(P_K) + \sum_{j=1}^{L} [f_{K-j} E(P_{K-j}) - \theta_{K-j}] \right\}^{-1}$$

where J = INT[K/2] = integer part of K/2,

L = INT[(K-1)/2],

and where variables θ_{K-1} through θ_{K-L} are calculated iteratively as shown below:

$$\theta_{K-1} = \text{MIN}[f_{K-1}E(P_{K-1}),\ f_1 E(P_1)],$$

= minimum of terms $f_{K-1}E(P_{K-1})$ and $f_1 E(P_1)$,

and for $2 \leq j \leq L$,

$$\theta_{K-j} = \text{MIN}[f_{K-j}E(P_{K-j}),\ f_j E(P_j) + \sum_{i=1}^{j-1} (f_i E(P_i) - \theta_{K-i})].$$

PROOF. See Appendix.

The results given in THEOREM 3 may be described in a slightly different manner using intuitive arguments. For each job-class-j, where $1 \leq j \leq K$, define variable ρ_j as follows:

$$\rho_j = \lambda f_j E(P_j),$$

= Average amount of Class-j work (i.e., processing time requests) arriving to the system per unit of time.

The capacity bound can only be achieved by a rule which results in a maximum amount of concurrent processing. This is accomplished by maximizing the sum of the probabilities that a small job and a large job are processed simultaneously and by insuring that a small job is never processed alone (more accurately, the probability that a small job is processed alone approaches zero as the input rate approaches λ_{max}). Consider the situation in which $\lambda = \lambda_{max}$, and define variable γ_{k-j} for $1 \leq j \leq L$ as given below:

$$\gamma_{K-j} = \lambda\theta_{K-j} \text{ for } 1 \leq j \leq L.$$

It then follows that

$$\gamma_{K-1} = \text{MIN}[\rho_{K-1}, \rho_1],$$

and for $2 \leq j \leq L$,

$$\gamma_{K-j} = \text{MIN}[\rho_{K-j}, \rho_j + \sum_{i=1}^{j-1} (\rho_i - \gamma_{K-i})].$$

The representation for λ_{max} in Theorem 3 implies that the following equation holds:

$$\lambda_{max} * \left\{ \sum_{j=1}^{J} f_j E(P_j) + \sum_{j=1}^{L} \theta_{K-j} + f_K E(P_K) + \sum_{j=1}^{L} [f_{K-j} E(P_{K-j}) - \theta_{K-j}] \right\} = 1$$

This equation, in turn, may be written as

$$\sum_{j=1}^{J} \rho_j + \sum_{j=1}^{L} \gamma_{K-j} + \rho_K + \sum_{j=1}^{L} (\rho_{K-j} - \gamma_{K-j}) = 1.$$

The above equation is valid at input rate λ_{max}, and the form of this equation may be explained by viewing the system operation over a unit interval of time. The details of system operation on a job-by-job basis will be ignored; instead only the ρ_j terms will be considered. The equation may be interpreted as follows:

{total work performed with two jobs simultaneously in progress}
+ {total work performed with only a single job active} = 1.

The total amount of work for which it is possible to have two jobs processed concurrently will be considered first. All work (i.e., ρ_j terms) associated with "small" jobs of Classes-1 through -J is included in this term because it should always be possible to concurrently process two small jobs, given that at least two of these jobs are present in the system. The sum of γ_{K-j} terms represents the work associated with "large" jobs for which it is possible to process concurrently with work associated with small jobs; note that:

γ_{K-1} = Maximum amount of Class-(K-1) work which can be processed simultaneously with Class-1 work per unit of time.

.

.

.

γ_{K-j} = Amount of Class-(K-j) work which can be processed concurrently with work associated with Classes-1 through -j, per unit of time.

Refer to the representations for the γ_{K-j} terms which appear above; these illustrate an optimal rule for matching work-loads presented by large jobs with those of small jobs. Class-1 jobs will be processed with Class-(K-1) jobs whenever possible, and term $(\rho_1 - \gamma_{K-1})$ represents any excess of Class-1 work above that amount needed to balance the Class-(K-1) work-load. Similarly, Class-(K-2) jobs will be run simultaneously with Class-2 jobs and the excess (if any) of Class-1 jobs; the term $(\rho_2 + (\rho_1 - \gamma_{K-1}) - \gamma_{K-2}$ is then the excess of Class-1 and Class-2 work. This process is continued in a similar fashion for the other job classes.

The total work performed with a lone job in progress includes all of the Class-K work (which must be processed alone) plus work associated with "large" jobs for which it is not possible to have concurrent processing. It may now be seen that, for $1 \leq j \leq L$,

$(\rho_{K-j} - \gamma_{K-j})$ = Class(K-j) work which cannot be matched up with work associated with "small" jobs of Classes-1 through -j.

Theorem 3 may be seen to provide, for this specific system, a solution to the Linear Programming problem of Theorem 1 which is valid for a wide range of values for the f_j and $E(P_j)$ terms which describe the job stream characteristics. Theorem 3 also is strongly suggestive of the manner in which a full-capacity rule might operate.

A Simple Multiprogramming Computer System

Consider a simple multiprogramming computer system having one CPU, one I/O File Unit, and eight pages of Primary Memory; this configuration will be shown to be an instance of a system described by Model 3. Define:

$R_1 = 1$,
= Number of Central Processing Units.

$R_2 = 1$,
= Number of I/O File Units.

$R_3 = 8$,
= Number of pages of Primary Memory.

The job stream characteristics are such that jobs may be in any one of four execution states (i.e., $J = 4$) having resource requirements and expected processing times as specified below:

State i	Resource Requirements	Expected Processing Time
0	None (Initial State)	0
1	(1,0,4)	$E(P_1)$
2	(0,1,4)	$E(P_2)$
3	(1,0,2)	$E(P_3)$
4	(0,1,2)	$E(P_4)$
5	None (Terminating State)	0

Below is a transition diagram (Figure 1) which illustrates the manner in which transitions may occur for a particular job in the system.

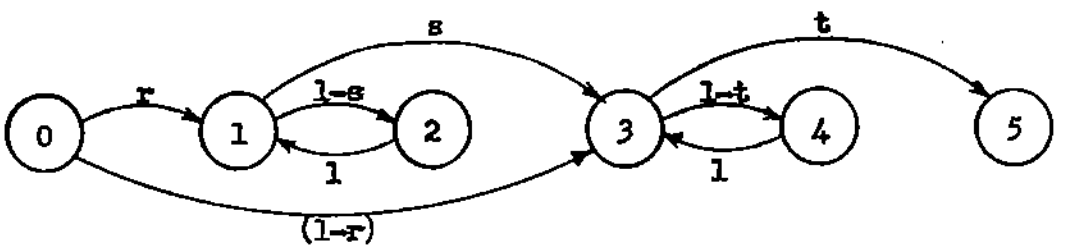

Figure 1. Transition diagram for jobs.

The routing transition matrix Q is therefore given by

$Q =$

Previous State \ Next State	1	2	3	4	5	
0	r	0	(1-r)	0	0	
1	0	(1-s)	s	0	0	$0 < r < 1$
2	1	0	0	0	0	$0 < s < 1$
3	0	0	0	(1-t)	t	$0 < t < 1$
4	0	0	1	0	0	

Recalling the notation introduced earlier, we have:

f_j = Expected number of times that a job will enter State-j, and the set of f_j must satisfy the J simultaneous linear equations given by

$$f_j = q(0,j) + \sum_{k=1}^{J} f_k * q(k,j) \qquad \text{where } 1 \leq j \leq J.$$

Because of the simple nature of the routing transition matrix Q in this problem we easily obtain:

$f_1 = r/s,$

$f_2 = r*(1-s)/s,$

$f_3 = (1-r*s)/t,$

$f_4 = (1-r*s)*(1-t)/t.$

Similarly, the feasible job combinations may be found with little difficulty; they are listed below for reference:

Combination Number	Description	Combination Number	Description
1	[0,0,0,1]	15	[0,0,0,3]
2	[0,0,1,0]	16	[0,0,1,2]
3	[0,1,0,0]	17	[0,0,2,1]
4	[1,0,0,0]	18	[0,0,3,0]
5	[0,0,0,2]	19	[0,1,0,2]
6	[0,0,1,1]	20	[0,1,1,1]
7	[0,0,2,0]	21	[0,1,2,0]
8	[0,1,0,1]	22	[1,0,0,2]
9	[0,1,1,0]	23	[1,0,1,1]
10	[0,2,0,0]	24	[1,0,2,0]
11	[1,0,0,1]	25	[0,0,0,4]
12	[1,0,1,0]	26	[0,0,1,3]
13	[1,1,0,0]	27	[0,0,2,2]
14	[2,0,0,0]	28	[0,0,3,1]
		29	[0,0,4,0]

It also follows that

M = 29 = Number of distinct feasible job combinations.

When Model 3 was described, term e_{mj} was defined as

e_{mj} = Probability that a job in state-j is progressing at unit rate during a random instant during the processing of job combination m, where $1 \leq m \leq M$ and $1 \leq j \leq J$.

The values for the e_{mj} terms in this problem have been chosen as follows. If the job combination m in service contains k(>0) jobs in states which require the CPU, each of those jobs receives (1/k) of the CPU power during that servicing interval (i.e., processor-sharing takes place). Likewise, for the n jobs in states requiring the I/O File Unit, the probability that any particular one is progressing at unit rate equals (1/n) at a random instant. Below are listed the e_{mj} terms which have values which differ from unity:

CPU Multiplexing (for jobs in state-1 and state-3)

$e_{7,3} = 1/2$, $e_{12,1} = 1/2$, $e_{12,3} = 1/2$, $e_{14,1} = 1/2$, $e_{17,3} = 1/2$,
$e_{18,3} = 1/3$, $e_{21,3} = 1/2$, $e_{23,1} = 1/2$, $e_{23,3} = 1/2$, $e_{24,1} = 1/3$,
$e_{24,3} = 1/3$, $e_{27,3} = 1/2$, $e_{28,3} = 1/3$, $e_{29,3} = 1/4$

I/O File Unit Multiplexing (for jobs in state-2 and state-4)

$e_{5,4} = 1/2$, $e_{8,2} = 1/2$, $e_{8,4} = 1/2$, $e_{10,2} = 1/2$, $e_{15,4} = 1/3$,
$e_{16,4} = 1/2$, $e_{19,2} = 1/3$, $e_{19,4} = 1/3$, $e_{20,2} = 1/2$, $e_{20,4} = 1/2$,
$e_{22,4} = 1/2$, $e_{25,4} = 1/4$, $e_{26,4} = 1/3$, $e_{27,4} = 1/2$

Given the above information, Theorem 2 may be directly applied to this problem. In order to obtain sample numerical results, it will be assumed that the parameters for the system take on the values given below:

$E(P_1) = 1$ $\qquad$ $r = 0.8$
$E(P_2) = 2$ $\qquad$ $s = 0.6$
$E(P_3) = 3$ $\qquad$ $t = 0.5$
$E(P_4) = 2$

Using these values, we immediately obtain:

$f_1 = 1.333,\ f_2 = 0.533,\ f_3 = 1.040,\ f_4 = 0.520$

There is now enough information to apply Theorem 3; however, we should make sure to use a Linear Programming software package that works properly when all constraints are equalities. Let π_i denote the proportion of time that the system should spend in processing the ith combination in order to achieve the capacity bound λ_{max}. If we solve the Linear Program for this system, we find that solution is not unique; below are two different solutions:

Solution 1		Solution 2	
π_{max}	= 0.225	π_{max}	= 0.225
π_2	= 0.467	π_{14}	= 0.299
π_4	= 0.060	π_{21}	= 0.240
π_6	= 0.234	π_{28}	= 0.234
π_{13}	= 0.240	π_{29}	= 0.228

There exist solutions in addition to the above, and it may be noted that, from the viewpoint of the Linear Program, a number of the combinations are equivalent. Job combination-m will be said to be equivalent to combination-k (denoted as "$m \equiv k$") if

$n_{mj} * e_{mj} = n_{kj} * e_{kj}$ for $j = 1,2,\ldots,J$.

If the job combinations are examined, the following "equivalences" may be noted:

$1 \equiv 5 \equiv 15 \equiv 25$

$2 \equiv 7 \equiv 18 \equiv 29$

$3 \equiv 10$

$4 \equiv 14$

$6 \equiv 16 \equiv 17 \equiv 26 \equiv 27 \equiv 28$

$9 \equiv 21$

$11 \equiv 22$

This system is CPU-bound, meaning that the CPU is the resource type whose utilization approaches 100% as the input rate nears the capacity bound. If the two solutions are examined, one may notice that they seem to differ markedly in terms of the "degree of multiprogramming" embodied in the combinations assigned non-zero values in each solution set of state probabilities. The equivalence between certain of the job combinations offers an explanation,

because equivalent combinations service identical amounts and types or work per unit of time. It is also interesting to note that no solution set of state probabilities for this problem (with the given parameters) will have non-zero values assigned combinations involving only jobs in state-2 or state-4 which require the I/O File Unit. For this reason, combinations 1, 3, 5, 8, 10, 15, 19 and 25 may be classified as "undesirable" combinations which should be avoided. If one were to examine the performance of a full-capacity scheduling rule for this system, it would be expected that the state probabilities for these undesirable combinations approach zero as the input rate increases to the capacity bound. Increasing the degree of multiprogramming for a CPU-bound system indeed has the effect of reducing the state probabilities for these undesirable states under most scheduling rules. An alternative is to use preemption to avoid processing the undesirable combinations: whenever a transition results in all jobs in service requiring the I/O File Unit (leaving the CPU idle), at least one of those jobs should be preempted with a job requiring the CPU. The derivation of the capacity bound was made under the assumption of zero overhead and arbitrary preemption; unfortunately, it is often impossible to preempt certain system resources at zero cost. It should be expected that CPU-bound multiprogramming systems will be unable to achieve 100% CPU utilization without employing the type of preemption described above.

Summary and Further Thoughts

Three different models were proposed for describing multi-resource queueing systems. For each of these models, an algorithm was given for determining the capacity bound for the system as well as a solution set of state probabilities describing an "optimal" proportion of time to be spent in processing various combinations of jobs. Two example systems were studied in order to demonstrate that the described approach can offer a number of insights into the design of full-capacity scheduling rules for such systems.

Because multi-resource queueing systems have not been previously examined using the author's approach, a number of conjectures and observations will be presented which were first discussed in reference [1].

(1) For a multi-resource queueing system, the scheduling rule should be concerned with the decision problem of choosing a combination of jobs to be serviced rather than merely the choice of the next job to go into service.

(2) First-Come-First-Served disciplines for multi-resource queueing systems will not, in general, be able to attain full capacity.

(3) Preemptive disciplines are usually required in order for a scheduling rule to be of the full-capacity type.

(4) Giving preferential service to a particular job class (or job-state) may cause the capacity under that scheduling rule to be less than that specified by the capacity bound.

Reference [4] is recommended for those desiring further information regarding instances of multi-resource queues which have been analyzed; that paper represents an overview of results independently obtained by Omahen [1] and Marathe [2]. The author believes strongly that all of the above conjectures are true and that multi-resource queues are important enough to warrant further investigation.

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

4. Omahen, K. J. and Marathe, V. P. A queueing model for a multiprocessor system with partitioned memory. Technical Report CSD-TR 132, Dept. of Computer Science, Purdue University, January, 1975.

APPENDIX

Proof of Theorem 3.

THEOREM 3. Assume that there is a stationary distribution for the processing times associated with each job class and that the jobs of each Class-j constitute a fixed proportion f_j of the overall input stream. The capacity bound λ_{max} for the two CPU system with K units of memory is given by

$$\lambda_{max} = 1/2\left[\sum_{j=1}^{J} f_j E(P_j) + \sum_{j=1}^{L} \theta_{K-j}\right] + \left[f_K E(P_K) + \sum_{j=1}^{L}\left[f_{K-j}E(P_{K-j}) - \theta_{K-j}\right]\right]^{-1}$$

where J = INT[K/2] = integer part of K/2,

L = INT[(K-1)/2],

and where variables θ_{K-1} through θ_{K-L} are calculated iteratively as shown below:

$$\theta_{K-1} = \text{MIN}\left[f_{K-1}E(P_{K-1}),\ f_1 E(P_1)\right],$$

= minimum of terms $f_{K-1}E(P_{K-1})$ and $f_1E(P_1)$,

and for $2 \leq j \leq L$,

$$\theta_{K-j} = \text{MIN}\left[f_{K-j}E(P_{K-j}),\ f_j E(P_j) + \sum_{i=1}^{j-1} (f_i E(P_i) - \theta_{K-i})\right].$$

PROOF. The state of the system at any point in time will be taken to be described by the job combination which is in progress. Assume that the system is nonsaturated under some input rate λ; the steady-state probability that the system is in a particular state will be denoted by the following terminology:

π_0 = Pr[System idle],

π_j = Pr[Class-j Job being processed along], where $1 \leq j \leq K$,

$\pi_{i,j}$ = Pr[Class-i Job and Class-j Job simultaneously in progress], where $1 \leq i \leq j \leq K-1$ and $i + j \leq K$.

The above probabilities are really functions of the input rate λ and of the scheduling rule that is employed for the system. Define two new variables J and L as (in the Theorem) follows:

$J = INT[K/2]$ = integer part of $K/2$,

$L = INT[(K-1)/2]$,

The various job combinations may be divided into four classifications, and the terminology will be adopted that a "small" job is a job belonging to a Class-j such that $1 \leq j \leq J$ and that a "large" job is one belonging to any of the classes K-L through K. The four classifications of job combinations are:

a) One small job and one large job (subject to the constraint that the two jobs fit into the available memory),

b) Two small jobs,

c) One large job,

d) One small job.

It will be convenient to define variables to represent the sums of the probabilities associated with each of the above classifications; these sums are given by:

$$A = \sum_{i=1}^{L} \sum_{j=i}^{L} \pi_{i,K-j} = \sum_{j=1}^{L} \sum_{i=1}^{j} \pi_{i,K-j} ,$$

$$B = \sum_{i=1}^{J} \sum_{j=1}^{J} \pi_{i,j} ,$$

$$C = \sum_{j=1}^{L} \pi_{K-j} ,$$

$$D = \sum_{i=1}^{J} \pi_{i} .$$

The sum of the probabilities over the possible system states must equal unity under nonsaturated operation, and the following equation obviously holds:

$$\pi_0 + A + B + C + D = 1. \qquad (1)$$

In addition to the above equation, the set of probabilities must also satisfy conservation equations which are obtained by applying Little's Equation [3] to the processor system. In this case, Little's Equation may be stated in the following manner:

$$E(N_j) = \lambda_j E(P_j) = \lambda f_j E(P_j),$$

where N_j = Non-negative Random Variable denoting the number of Class-j jobs in progress,

and integer j is such that $1 \leq j \leq K$.

The following K conservation equations, one per job class, are obtained through the use of the above equation; it may be remarked, though, that the set of K equations will be slightly different depending on whether the number of units of memory, K, has an even or an odd value.

For Class-1 Jobs:

$$\pi_1 + 2\pi_{1,1} + \sum_{j=2}^{J} \pi_{1,j} + \sum_{j=1}^{L} \pi_{1,K-j} = \lambda f_1 E(P_1) \ .$$

For Class-j Jobs, where $2 \leq j \leq L-1$:

$$\pi_j + 2\pi_{j,j} + \sum_{i=1}^{j-1} \pi_{i,j} + \sum_{i=j+1}^{J} \pi_{j,i} + \sum_{i=j}^{L} \pi_{j,K-i} = \lambda f_j E(P_j) \ .$$

If K is even, the two equations below are obtained ($L = J-1$) for Class-L and Class-J Jobs, respectively:

$$\pi_L + 2\pi_{L,L} + \sum_{i=1}^{L-1} \pi_{i,L} + \pi_{L,J} + \pi_{L,K-L} = \lambda f_L E(P_L) \ ,$$

$$\pi_J + 2\pi_{J,J} + \sum_{i=1}^{J-1} \pi_{i,J} = \lambda f_J E(P_J) \ .$$

For K odd, the single equation below is found ($L = J$) for Jobs of Class-J:

$$\pi_J + 2\pi_{J,J} + \sum_{i=1}^{J-1} \pi_{i,J} + \pi_{J,K-L} = \lambda f_J E(P_J).$$

For Class-(K-j) Jobs, where $1 \leq j \leq L$:

$$\pi_{K-j} + \sum_{i=1}^{j} \pi_{i,K-j} = \lambda f_{K-j} E(P_{K-j}).$$

For Class-K Jobs:

$$\pi_K = \lambda f_K E(P_K).$$

By finding the sums of the left-hand and right-hand sides of these K equations and equating the two results, we obtain:

$$2(A+B) + (C+D) = \lambda \sum_{j=1}^{K} f_j E(P_j). \qquad (2)$$

Under any specified discipline j, the system will first saturate at input rate λ_{maxj} (i.e., the capacity under that discipline), and it must be true that

$$\lim_{\lambda \uparrow \lambda_{maxj}} \pi_0 = 0 \qquad \text{and} \qquad \lim_{\lambda \uparrow \lambda_{maxj}} [A+B+C+D] = 1.$$

At system capacity λ_{maxj} under any particular discipline, the following equation holds in addition to the K conservation equations and Equation (2):

$$A + B + C + D = 1. \qquad (3)$$

Using Equations (2) and (3), it is found that

$$\lambda_{maxj} = \left\{ 1 + (A+B) \right\} / \sum_{j=1}^{K} f_j E(P_j). \qquad (4)$$

The notation given below will be useful in the material which follows: define:

MIN[a,b] = minimum of a and b,

max(y) = maximum value of variable y.

The capacity bound λ_{max} for this system will be an input rate at which the overall expected waiting time will be infinite for every scheduling rule. By Equation (4), it is apparent that λ_{max} can be achieved only by a rule which results in the sum A+B being maximized (alternatively, C+D being minimized). A rule which maximizes A+B is really maximizing the amount of concurrent processing which takes place in the system since A and B are sums of the state probabilities associated with the concurrent processing of two jobs.

Two additional equations may be obtained by summing both sides of the conservation equations for jobs of Classes-1 through -J and also for jobs of Classes-(K-L) through -K; this procedure gives:

$$D + 2B + A = \lambda \sum_{j=1}^{J} f_j E(P_j), \qquad (5)$$

and

$$C + A = \lambda \left\{ f_K E(P_K) + \sum_{j=1}^{L} f_{K-j} E(P_{K-j}) \right\} . \qquad (6)$$

By rearranging terms in Equation (5), it is found that

$$A + B = \left\{ \lambda \sum_{j=1}^{J} f_j E(P_j) + A - D \right\} /2, \qquad (7)$$

and Equation (6) gives

$$C = \lambda \left\{ f_K E(P_K) + \sum_{j=1}^{L} f_{K-j} E(P_{K-j}) \right\} - A. \qquad (8)$$

Equation (7) illustrates that the sum A+B is maximized when the sum A-D is maximized. It will now be demonstrated that the probabilities of the various states may always be chosen so that D is equal to zero. Recall that D is the sum of the probabilities that a "small" job is processed alone by the system. Terms π_j and $\pi_{j,j}$ appear only in the conservation equation for Class-j Jobs, where $1 \leq j \leq J$. If $\pi_j = \varepsilon > 0$ and $\pi_{j,j} = \delta \geq 0$, the probabilities may be instead chosen so that $\pi_j = 0$ and $\pi_{j,j} = \delta + \varepsilon/2$, and the conservation equation for Class-j Jobs will still be satisfied. It follows directly that the set of probabilities may always be chosen such that

$D = 0.$

Maximizing A+B is equivalent to maximizing A and minimizing D (i.e., to value zero) as can be seen from Equation (7). Substituting Equations (7) and (8) into Equation (3), the following equation is valid at $\lambda = \lambda_{max}$:

$$1/2 \left\{ \lambda_{max} \sum_{j=1}^{J} f_j E(P_j) + \max(A) \right\} + \left\{ \lambda_{max} \left[f_K E(P_K) + \sum_{j=1}^{L} f_{K-j} E(P_{K-j}) \right] - \max(A) \right\}$$

$$= 1. \qquad (9)$$

A procedure will now be given for determining the value max(A) for input rate λ; in order to simplify the description of this procedure it will be convenient to define a function which represents various partial sums of the terms included in A. For integer n such that $1 \leq n \leq L$, define:

$$S(n) = \sum_{j=1}^{n} \sum_{i=1}^{j} \pi_{i,K-j} .$$

It may immediately be seen that

$$A = S(L), \tag{10}$$

and that the following relation holds for the function S(n) when $2 \leq n \leq L$:

$$S(n) = S(n-1) + \pi_{n,K-n} + \sum_{i=1}^{n-1} \pi_{i,K-n} . \tag{11}$$

When describing the procedure for finding max(A) for some input rate λ, it will often be necessary to find the sums of both the left- and right-hand sides of certain of the conservation equations; by equating the two sums, it will then be possible to obtain inequalities which gives bounds on the values of the state probabilities.

The maximum value for $\pi_{j,K-j}$, where $1 \leq j \leq L$, may be easily found from the conservation equations for Class-j and Class-(K-j) Jobs. The conservation equation for Class-j Jobs implies that

$$\pi_{j,K-j} \leq \lambda f_j E(P_j),$$

and similarly the equation for Class-(K-j) jobs gives

$$\pi_{j,K-j} \leq \lambda f_{K-j} E(P_{K-j}) .$$

From the two inequalities above, it must be true that

$$\pi_{j,K-j} \leq \lambda * \text{MIN}[f_j E(P_j),\ f_{K-j} E(P_{K-j})]. \tag{12}$$

where $1 \leq j \leq L$.

Consider the conservation equations for Classes-1 through-n, where $2 \leq n \leq L$. By first summing the left-hand sides of these n equations, then finding the sum of the right-hand sides, and finally equating the two sums, a new equation is obtained. The resulting equation shows the inequality below to be valid:

$$\sum_{i=1}^{n-1} \pi_{i,K-n} \leq \lambda \sum_{j=1}^{n-1} f_j E(P_j) - S(n-1) .$$

The equation for Class-(K-n) jobs immediately gives

$$\sum_{i=1}^{n-1} \pi_{i,K-n} \leq \text{MIN}\left[\lambda \sum_{j=1}^{n-1} f_j E(P_j) - S(n-1),\ \lambda f_{K-n} E(P_{K-n}) - \pi_{n,K-n}\right] \tag{13}$$

Equation (11) and Inequality (13) imply that

$$S(n) \leq S(n-1) + \pi_{n,K-n} + MIN\left[\lambda \sum_{j=1}^{n-1} f_j E(P_j) - S(n-1),\ \lambda f_{K-n} E(P_{K-n}) - \pi_{n,K-n}\right] .$$

The terms on the right-hand side of the above inequality may be rewritten in the two forms shown below:

$$S(n) \leq MIN\left[\lambda \sum_{j=1}^{n-1} f_j E(P_j) + \pi_{n,K-n}\ ,\ \lambda f_{K-n} E(P_{K-n}) + S(n-1)\right] ,$$

or

$$S(n) \leq S(n-1) + MIN\left[\lambda \sum_{j=1}^{n-1} f_j E(P_j) + \pi_{n,K-n} - S(n-1),\ \lambda f_{K-n} E(P_{K-n})\right] .$$

The first of these two forms makes it obvious that $S(n)$ is maximized when both $\pi_{n,K-n}$ and $S(n-1)$ are maximized. We therefore find that, at input rate λ, the relation for the maximum value of $S(n)$ is given by (for $2 \leq n \leq L$)

$$\max(S(n)) = \max(S(n-1)) + MIN\left[\lambda \sum_{j=1}^{n-1} f_j E(P_j) + \max(\pi_{n,K-n}) - \max(S(n-1)),\ \lambda f_{K-n} E(P_{K-n})\right] , \qquad (14)$$

and Equation (12) gives

$$\max(\pi_{n,K-n}) = \lambda * MIN\left[f_n E(P_n),\ f_{K-n} E(P_{K-n})\right] \quad \text{for } 1 \leq n \leq L.$$

If one observes that

$$\max(S(1)) = \max(\pi_{1,K-1}) = \lambda * MIN\left[f_1 E(P_1),\ f_{K-1} E(P_{K-1})\right] ,$$

it is clear that the maximum values for $S(1)$ through $S(L)$ may be calculated iteratively to finally obtain $\max(A)$. Given the procedure for calculating $\max(A)$, we wish to modify the procedure slightly in order to show the manner in which $\max(A)$ varies as a function of the input rate λ. Equation (14) may be rewritten as shown below:

$$\max(S(n)) = \max(S(n-1)) + MIN\left[\lambda f_{K-n} E(P_{K-n}),\ \lambda f_n E(P_n) + \lambda \sum_{j=1}^{n-1} f_j E(P_j) - \max(S(n-1))\right] . \qquad (15)$$

Equation (15) is obtained by noting that

$$\text{MIN}[X+\text{MIN}[Y,Z],Z] = \text{MIN}[X+Y,X+Z,Z] = \text{MIN}[X+Y,Z] \quad \text{if } X \geq 0,$$

$$\text{where } X = \lambda \sum_{j=1}^{n-1} f_j E(P_j) - \max(S(n-1)),$$

$$Y = \lambda f_n E(P_n),$$

$$\text{and} \quad Z = \lambda f_{K-n} E(P_{K-n}).$$

By equating the sums of the left- and right-hand sides of the conservation equations for Classes-1 through-n, it may be verified that

$$X = \lambda \sum_{j=1}^{n-1} f_j E(P_j) - \max(S(n-1)) \geq 0.$$

The procedure for calculating max(A) may be restated; define θ_{K-j} for $1 \leq j \leq L$ as given below:

$$\theta_{K-1} = \text{MIN}\left[f_{K-1}E(P_{K-1}),\ f_1 E(P_1)\right],$$

and for $2 \leq j \leq L$,

$$\theta_{K-j} = \text{MIN}\left[f_{K-j}E(P_{K-j}),\ f_j E(P_j) + \sum_{i=1}^{j-1} (f_i E(P_i) - \theta_{K-i})\right].$$

Using these new variables, an alternative method for stating the procedure for calculating max(A) may be given. First note that

$$\max(S(n)) = \lambda \sum_{j=1}^{n} \theta_{K-j} \qquad \text{for } 1 \leq n \leq L,$$

and therefore

$$\max(A) = \lambda \sum_{j=1}^{L} \theta_{K-j}\ ,$$ where θ_{K-1} through θ_{K-1} are calculated iteratively as described above.

The value of max(A) for $\lambda = \lambda_{max}$ may be substituted into Equation (9), and a first order equation in λ_{max} results. Solving for λ_{max}, the proof is complete.

Q.E.D.