CHAPTER 6: **JACKSON NETWORK ANALYSIS**

Our goal in this section is to extend the previous performance analysis (based only on mean execution times) to include both fluctuations about mean values as well as ordering constraints or correlations between different steps and different jobs. This demands additional assumptions concerning the job arrival and execution time statistics, as well as the scheduling policy. We focus on a class of Markovian mathematical models where knowing only the present or current state of the system, not the entire history of the system operations, its future statistical behavior is determined. These are called *Jackson queueing networks* in honor of the pioneer researcher who first studied their behavior.

If all possible system states and the associated state transition rates are exhaustively enumerated, oftentimes the *number* of states in such models explodes (into tens of trillions for straightforward applications such as a single terminal word processor system, and into far far greater numbers for more sophisticated applications such as airline reservation systems with hundreds of terminals, tens of communication lines, and upwards of one hundred disk spindles) to far exceed the storage and processing capabilities of the most powerful supercomputers (both available and envisioned!): it can be virtually impossible to extract any analytically tractable formulae or numerical approximations of different measures of performance. The only credible avenues for quantifying system performance for many people for years appeared to be simulation studies or building a prototype (which is another form of simulation, using the actual system but simulating the load).

In a series of seminal papers, Jackson made the *fundamental* observation that for a wide class of Markovian models, the distribution of the number of jobs in each step queued or in service at *each* node in a network could be written as the product of functions, with each function depending only on the workload for that node. This arises in many other areas of mathematics, where the solution to a multivariate set of equations is assumed to be the *product* of functions depending only on one variable, a so called *separation of variables* decomposition. It is also in the spirit of Fourier analysis, where an arbitrary function is approximated as the sum of a number of elementary functions. Because of this property, the long term time averaged distribution for the number of jobs queued or in execution in these Jackson models are often said to obey a *product form* distribution. Just (or more) important than this analysis is the ready availability of efficient and easy to use software packages that numerically approximate a variety of performance measures for these classes of models, and handle both input and output in a flexible manner, allowing different design choices and sensitivity analyses to be quickly carried out*. There is currently a great deal of research activity into both extending this class of models and extending the range of numerical approximations.

6.1 Overview

First we describe the ingredients in a Jackson queueing network model and then we develop the outline of the exposition.

6.1.1 Ingredients The inputs for this type of analysis are a description of the total mean arrival rate for each job, the total mean time each job uses each resource with only one resource used at a time, and a description of how contention for each resource is resolved. The output of this type of analysis is the fraction of time a given *number* of jobs are queued or in execution at each resource. This in turn can be manipulated to determine the utilization of each resource, and the mean throughput rate and mean

---

* Efficiency: in one study conducted by the authors for an online transaction processing system with five hundred terminals, ten processors each with two disk spindles, and thirty data links, the entire model (including a graphics library), occupied less than ninety thousand bytes of static main memory, and could produce graphic output for seven hundred and twenty different parameter choices using less than ten seconds of processor time (for both numerical calculations and graphical output) running in a GCOS environment on a H6000.

queueing time (both waiting and execution time) for each job using the techniques developed in the earlier sections. The scheduling policies analyzed here demand that a processor must execute a job if a job is ready for execution: these are called *work conserving* policies because the total backlog of work is independent of the precise details of the scheduling policy at any instant of time. Two examples of work conserving policies are (i)servicing jobs in order of arrival, or (ii)multiplexing a single processor at a high speed among J jobs so that each job receives $1/Jth$ the total available processor cycles. The mean value analysis inputs are a subset of these inputs, so we expect the Jackson network outputs to refine the mean value analysis outputs. An example of a policy that does *not* conserve work is found in a system that executes two types of jobs, one with one second execution times and the other with one hour execution times; if it frequently occurs that ten seconds after we begin execution of a one hour job, a short one second job may arrive, then we might wait for eleven seconds after a long job arrives before starting execution, holding the processor idle, to allow short jobs not to be delayed at the expense of long jobs.

6.1.2 Program First we analyze a single parallel group of processors fed from a single queue, for a wide variety of parameter choices. Second we examine an arbitrary network of serially reusable resources where each job migrates from node to node, entering and leaving at different nodes. As a special case, we examine a pipeline of nodes or stages, each with a single processor. All of this is done for one type of job, and then extended to multiple types of jobs. These basic cases will form the foundations for a series of case studies and examples in this and subsequent chapters.

Ideas appear here that have not been encountered before. All of these new ideas are expounded for the simplest case, a single parallel group of processors: jobs can be either *delayed* (stored in a buffer) until a processor is available, or *rejected* if no storage space is available. Queueing or buffering jobs might be chosen if it were highly likely that a processor would become available an acceptable time after a job arrival; jobs would be blocked on arrival if this were not the case. A different way of thinking about this is that jobs are blocked or delayed for a finite time from being serviced; in the limit where jobs are blocked or delayed forever, the jobs are lost, i.e., never serviced. In practice, systems are designed so that in normal operation blocking or delay or defection is rare. Many communication systems adopt combinations of loss and delay mechanisms. We choose to first analyze delay alone and then loss alone before attempting to analyze systems combining both these mechanisms.

Finally, special asymptotic limiting cases give us analytic insight into system performance. One such case involves allowing the number of active terminals attached to a computer system to become larger and larger, while the total workload stays constant. As the total number of terminals grows, each terminal contributes less and less to the workload, but the total workload is constant. Another such case involves allowing the number of terminals and processors to become larger and larger, while the ratio of terminals and processors stays fixed. Each terminal contributes the same workload, so as the total number of terminals and processors grows, the total workload grows.

6.1.3 Additional Reading

[1] J.R.Jackson, *Networks of Waiting Lines,* Operations Research, **5,** 518-521 (1957).

[2] J.R.Jackson, *Jobshop-Like Queueing Systems,* Management Science, **10** (1),131-142(1963).

[3] F.P.Kelly, *Networks of Queues,* Advances in Applied Probability, **8,** 416-432 (1976).

[4] F.P.Kelly, **Reversibility and Stochastic Networks,** Wiley, Chichester, 1979.

6.2 A Single Node with P Parallel Processors

Clerks at terminals submit one type of job to a computer system that consists of P processors, then wait for a response before submitting the next job. The processors are fed jobs from a single queue. Jobs are stored in a buffer of capacity M jobs if no processor is available. If a job is submitted with all buffers filled, the job is rejected, and the clerk begins the entire process over again. The mean time spent by a clerk reading and thinking and typing is denoted by T_{think}. The mean processor execution time per job is denoted by T_{proc}. Our goal is to calculate the mean throughput rate λ and the mean delay (waiting plus execution time) that a clerk experiences which we will call the *queueing* time or *flow*

Figure 6.1.System Block Diagram

time of a job, denoted by $E(T_Q)$.

6.2.1 Functional Operation In order to describe the operation of the system, we need to describe the workload generated by each clerk and the policy for arbitrating contention for shared resources.

The workload is handled first. There are C clerks. There are at least as many clerks as buffer space, $C \geq M$. Each clerk spends a mean amount of time T_{think} reading, thinking, and typing and then submits a job for execution. The sequence of times each clerk spends reading and thinking and typing are assumed to be independent identically distributed random variables. Once a job is submitted to the system, if storage space is available, the job is accepted for service; otherwise the job is rejected and returns to the clerk who starts the entire cycle over again. The system is capable of storing a maximum of M jobs. If a job is accepted and there is an idle processor available, that job begins execution immediately. The processor execution times are independent identically distributed random variables with mean T_{proc}.

Next, the arbitration or scheduling policy is described. There are two queues here, one for the clerks and one for the processors. Once a processor finishes executing a job, an idle clerk will enter into the thinking state; since there is one clerk per job, there will never be any jobs waiting in the clerk queue for an idle clerk. The rate at which clerks submit jobs to the system depends upon the number of clerks reading and thinking and typing. Let $\Phi_{clerk}(J_{clerk})$ denote the number of seconds of work done in one second of time. For example, if $\Phi_{clerk}(J_{clerk})=J_{clerk}$, then if $J_{clerk}=0$ then no work is done, while if $J_{clerk}=1$ then one second of work is done in one second, and if $J_{clerk}=2$ then two seconds of work are done in one second, given there are J_{clerk} clerks in the thinking state. More generally, we assume that this function is positive and monotonically increasing with J_{clerk}, and equals zero if $J_{clerk}=0$. We stress that $\Phi_{clerk}(J_{clerk})$ is dimensionless or without any units. The instantaneous (i.e., the instant at which there are J_{clerk} clerks thinking) rate at which jobs are submitted by clerks is given by

$$\textit{instantaneous job submittal rate} = \frac{\Phi_{clerk}(J_{clerk})}{T_{think}} \quad J_{clerk}=0,1,...,C$$

For example, if we model the clerk behavior by

$$\Phi_{clerk}(J_{clerk}) = min(J_{clerk}, C) = J_{clerk} \quad J_{clerk}=0,1,...,C$$

note that no jobs will be submitted if $J_{clerk}=0$, i.e., if all clerks are waiting for a response. At the other extreme, if all clerks are reading, thinking and typing, this will be the maximum rate of submitting jobs.

Figure 6.2.Queueing Network Block Diagram

The processor queue operates as follows: if no processor is available, the job is inserted into a buffer, with the lower the index the closer to the head of the queue. We assume the number of buffers is greater than or equal to the number of processors, $M \geq P$. If there are J_{proc} jobs in the queue just prior to arrival of a job, the new job is inserted into the queue in position I with probability $\delta(I, J_{proc}+1)$. If a job is inserted in position I, all other jobs are moved to one index higher position. The rate at which all processors execute jobs depends upon the number of jobs in the queue. Let $\Phi_{proc}(J_{proc})$ denote the amount of work done, measured in seconds, during one second of time. For example, if $\Phi_{proc}(J_{proc}) = min[J_{proc}, P=2]$, then for $J_{proc}=0$ no work is done, for $J_{proc}=1$ one second of work is done in one second, for $J_{proc}=2$ two seconds of work are done in one second, and for $J_{proc}>2$ two seconds of work are done in one second. The instantaneous rate at which the processor subsystem is executing jobs is given by

$$processor\ \ job\ \ execution\ \ rate = \frac{\Phi_{proc}(J_{proc})}{T_{proc}}$$

For example, we might choose

$$\Phi_{proc}(J_{proc}) = min[J_{proc}, P] \quad J_{proc}=0,...,M$$

In words, if there are P or fewer jobs present, then each processor can execute one job at a rate of one job every T_{proc} seconds, while if there are more jobs than processors, then all processors are completely busy executing jobs.

The rate at which job I is executed in the queue when there are J_{proc} jobs total either in execution or waiting is given by $\gamma(I, J_{proc})\Phi_{proc}(J_{proc}) \leq 1$, where

$$\sum_{I=1}^{J_K} \gamma_{proc}(I, J_{proc}) = 1$$

We stress that $\gamma(I, J_{proc})$ is measured in seconds of service received during one second of actual time.

>From this point on, we assume either of two conditions hold governing the operation of the processor system:

- The processor service time obeys an exponential distribution

$$fraction\ \ of\ \ time\ \ job\ \ executed\ \ in\ \ X\ \ seconds = 1 - exp[-X/T_{proc}]$$

- The service time distribution is arbitrary, and the fraction of time that a job is inserted into position I with J_{proc} jobs in the queue equals the rate at which that job will receive service:

$$\gamma(I, J_{proc}) = \delta(I, J_{proc})$$

One example of a scheduling rule that meets these conditions is processor sharing, where all jobs are multiplexed by the processor so quickly that effectively each receives a fraction of processor capacity proportional to the number of jobs at the processor queue. If J_{proc} jobs are present, each receives $1/J_{proc}$ the fraction of available processing capacity:

$$\gamma(I, J_{proc}) = \delta(I, J_{proc}) = \frac{1}{J_{proc}}$$

A second example is processor scheduling such that the last arrival is the first to receive service, with subsequent later arrivals preempting the job currently in service, and service is resumed (eventually, after all later arrivals are completely executed) at the point of interruption:

$$\gamma(I, J_{proc}) = \begin{cases} 1 & I = J_{proc} \\ 0 & otherwise \end{cases}$$

A third example is where there is always a processor available to execute a job, so

$$\gamma(I, J_{proc}) = \delta(I, J_{proc}) = 1$$

so that when a job arrives it immediately begins execution.

6.2.2 State Space What is the state of the system? The complete state space would involve specifying the state of each clerk (busy with a transaction or waiting for the system to respond), and the state of each job in the system (where it is in the queue and how much service it had received). Suppose we only keep track of the *total* number of clerks busy with jobs, and the *total* number of jobs in the system? The system state at any instant is given by the pair $\underline{J} = (J_{clerk}, J_{proc})$ where $J_{clerk} = 0,...,C$ denotes the number of clerks with jobs, i.e., reading, thinking and typing, and $J_{proc} = 0,...,M$ denotes the number of jobs in the system, both queued and in execution. A key constraint is that each clerk is either busy reading, thinking and typing, or waiting for a response; put differently, there are always C jobs somewhere:

$$J_{clerk} + J_{proc} = C$$

The state space, the set of admissible pairs $\underline{J}$ is denoted by Ω:

$$\Omega = \{\underline{J} = (J_{clerk}, J_{proc}) \mid J_{clerk} = 0,...,C; J_{proc} = 0,...,M; J_{clerk} + J_{proc} = C\}$$

In describing the operation of the system, we might measure the following items:

- The times at which jobs are submitted
- Whether a job is accepted or rejected because storage is not available
- The time an accepted job waits without being executed in the system
- The total time an accepted job spends in the system

For each of these, we could summarize the measurements with averages, carried out over a sufficiently long time interval such that the mean values stabilized.

6.2.3 Goals What customer oriented criteria are we interested in here?

- the mean waiting time $E(T_W)$ and mean queueing time $E(T_Q)$
- the fraction of time a task waits at all $PROB[T_W = 0]$
- the fraction of time a task waits greater than X time units $PROB[T_W > X]$
- the fraction of time a task has a queueing time greater than X time units $PROB[T_Q > X]$

What system oriented criteria are of interest?

- the mean throughput rate of executing jobs

- the mean number of busy processors
- the fraction of time each processor is busy, its utilization
- the fraction of time more than one processor and more than one clerk are busy

We want to operate in states with acceptable levels of customer and system oriented criteria. States with acceptable customer oriented criteria may result in unacceptable system oriented criteria: delays may be acceptable at the expense of idle processors and low throughput, while high utilization of each available processor and high throughput rates may result in long queues and long delays.

6.2.4 Mean Value Analysis The first level of analysis is based on mean value inputs, with the outputs being mean throughput rate and mean delay. The fraction of attempts that are blocked or rejected because the available storage is filled is denoted by B. The mean throughput rate is

$$\lambda_{offered} = \frac{\lambda_{accepted}}{1-B} \qquad \lambda_{offered}(1-B) = \lambda_{accepted}$$

On the other hand, each clerk spends a mean amount of time reading and thinking and typing, T_{think}, and then submits a job; if the job is accepted, i.e., the fraction of attempts that are accepted is 1–B, then the job will spend a mean time $E(T_Q)$ inside the system either waiting or in execution, and then the cycle starts over again. Since there are C clerks, the mean throughput rate is given by

$$\lambda_{accepted} = \frac{C}{(1-B)E(T_Q) + T_{think}}$$

In order to determine mean throughput rate and mean delay, we need to determine the fraction of requests that are blocked. If there is no blocking, $M{=}C$, then we can handle this by the methods in the previous sections; if there is blocking, $C{>}M$, then the earlier analysis will only give a lower bound on mean throughput rate and an upper bound on mean delay.

If $M{=}C$, the system has three bottlenecks:

- If clerks are limiting the mean throughput rate, then

$$\lambda \equiv \frac{C}{T_{think}+T_{proc}}$$

- If processor time is limiting the mean throughput rate, then

$$\lambda \equiv \frac{P}{T_{proc}}$$

- If storage or buffering is limiting the mean throughput rate, then

$$\lambda \equiv \frac{M}{T_{proc}}$$

Provided that $M{=}C$, we can write down upper and bounds on the mean throughput rate and mean queueing time:

$$\lambda_{lower} \le \lambda \le \lambda_{upper}$$

$$\lambda_{lower} = \frac{C}{T_{think} + \frac{C}{min[M,C]}T_{proc}}$$

$$\lambda_{upper} = \min\left[\frac{P}{T_{proc}}, \frac{C}{T_{think}+T_{proc}}\right]$$

$$\max\left[T_{proc}, \frac{C}{\lambda_{upper}} - T_{think}\right] \le E(T_Q) \le \frac{C}{min[M,C]}\ T_{proc}$$

In order to proceed further, we need to make use of the assumptions outlined earlier concerning arrival and execution time statistics, and scheduling policies.

6.2.5 Jackson Network Analysis Averaged over a suitably long time interval, the fraction of time the system is in a given state is denoted by $\pi(\underline{J})$ which is the product of three terms, one dependent only on clerks, one dependent only on the parallel processor group, and one that normalizes $\pi_C(\underline{J})$ so that the system is in *some* state all the time:

$$\pi_C(\underline{J}) = \frac{1}{G}\ (clerk\ \ dependent\ \ term) \times (processor\ \ dependent\ \ term) \quad \underline{J} \in \Omega$$

$$\sum_{\underline{J} \in \Omega} \pi_C(\underline{J}) = 1$$

and the second factor depends only on the source (here clerks reading and thinking and submitting jobs), while the final factor depends only on the system (which here is a group of parallel processors).

$$\pi_C(\underline{J}) = \pi_C(J_{clerk}, J_{proc}) = \frac{1}{G} \prod_{I=0}^{J_{clerk}-1} \frac{\Phi_{clerk}(I)}{T_{think}} \prod_{K=0}^{J_{proc}} \frac{T_{proc}}{\Phi_{proc}(K)} \quad \underline{J} \in \Omega$$

The function G is also called the *system partition function,* and is chosen such that

$$\sum_{\underline{J} \in \Omega} \pi_C(\underline{J}) = 1$$

The fraction of arrivals that are blocked or rejected is denoted by B, where

$$B = B(M) = \begin{cases} \pi_{C-1}(J_{proc}=M) & C>M \\ 0 & C=M \end{cases}$$

Note that the fraction of time an *arrival* finds J_{proc} jobs in the system is *different* from the long term time averaged distribution of jobs in the system, because when an arrival occurs it *cannot* be in the system, i.e., effectively there are a total of $C-1$ jobs in the system:

$$fraction\ \ of\ \ arrivals\ \ when\ \ \ system\ \ state\ \ is\ \ \underline{J} = \pi_{C-1}(\underline{J})$$

The fraction of time that a job enters service immediately is given by

$$fraction\ \ of\ \ time\ \ job\ \ enters\ \ service\ \ immediately = \frac{\sum_{K=0}^{P-1} B(K)}{1 - B(M)}$$

The mean number of jobs in execution in the system is given by

$$E[min(J_{proc}, P)] = \sum_{K=0}^{M-1} min(K, P)\ \frac{B(K)}{1 - B(M)}$$

The mean throughput rate is the mean number of jobs in execution divided by the mean time per job:

$$\lambda = \frac{E[min(J_{proc}, P)]}{T_{proc}}$$

The mean queueing time per job, including both waiting and execution, is the ratio of the total mean number of jobs in the processor system divided by the mean throughput rate:

$$E[T_Q] = \frac{E[J_{proc}]}{\lambda} = T_{proc}\ \frac{E(J_{proc})}{E[min(J_{proc}, P)]}$$

The mean queueing time is equal to the mean execution time of a job, multiplied by a stretching factor:

$$stretching\ \ factor = \frac{E(J_{proc})}{E[min(J_{proc}, P)]}$$

If $J_{proc} << P$ then the stretching factor is one, i.e., there is zero mean waiting time, while if $J_{proc} >> P$ then the stretching factor grows as J_{proc}/P.

The difference between the total number of jobs either queued or in execution, J_{proc}, and the number of jobs actually in execution, $min[J_{proc}, P]$, is defined as the number of jobs waiting. We can rewrite this

as follows

$$J_{proc} - min[P, J_{proc}] = max[0, J_{proc} - P]$$

The mean waiting time per job is the ratio of the mean number of waiting jobs divided by the mean throughput rate:

$$E(T_W) = \frac{E[max(J_{proc} - P, 0)]}{\lambda}$$

In general, although we can obtain an expression for the joint distribution of number of jobs at each stage in the system, we do **not** have an expression for the *delay distribution* and must infer mean delay via Little's Law. The exception is where jobs are executed in order of arrival, first come, for one case: jobs are executed in order of arrival, first come, first serve. For this case, when a job arrives it must wait for all jobs ahead of it to be serviced and then it will be executed. For this case, we can explicitly calculate the delay statistics for a job.

$$PROB[T_W > X \mid job\ accepted] = \frac{\sum_{K=P}^{M-1} B(K)[1 - \tilde{E}_{K-P+1}(X)]}{1 - B(M)}$$

$$\tilde{E}_K(X) = 1 - \exp(-X/T_{proc}) \sum_{I=0}^{K-1} \frac{(X/T_{proc})^I}{I!} \quad K = 1,2,...$$

EXERCISE. Calculate how quickly the mean value asymptotes are approached, as $P \to \infty$, with all other parameters held constant. Is the rate of convergence exponentially fast? as a power of P?

EXERCISE. Determine the rate of convergence to the mean value asymptotes with $C \to \infty$ but all other parameters fixed. Is the rate of convergence exponentially fast? as a power of C?

6.2.6 Additional Reading

[1] R.Syski, *Markovian Queues*, pp. 170-227 (Chapter Seven) in **Proceedings Symposium on Congestion Theory,** W.L.Smith, W.E.Wilkinson (editors), 24-26 August 1964, University of North Carolina Press, Chapel Hill, North Carolina, 1965.

6.2.7 Numerical Approximation Just because we have an analytic expression does *not* mean that we can get numbers of interest for engineering actual systems. A key advantage of Jackson network models is that a wide variety of performance measures can be quickly and efficiently (storage and execution time) approximated. This is not to be dismissed lightly: these numbers often suggest design tradeoffs, areas that require further study, in a very short period of time, rather than spending months (or years!) finding out the obvious. Since it does not take long to get numbers, little has been invested, but perhaps a lot is gained!

There exist several classes of algorithms that efficiently use storage for numerically approximating the partition function and related quantities. We will explore these in this and later sections. For the problem at hand, a recursion for approximating the system partition function appears to be most appropriate, provided C is small, e.g., $C < 10$.

We assume the total number of jobs in the system is fixed at M in what follows. The partition function can be viewed as a function of two variables, the total number of nodes N and the total number of jobs inside the system M. For this case, the mean throughput rate λ is given by

$$\lambda = \frac{G(M-1,N)}{G(M,N)} \qquad \sum_{K=1}^{N} J_K = M$$

The fraction of time there are L jobs at node N, i.e., $J_N = L$, is given by

$$\pi(J_N = L) = \sum_{\underline{J} \in \Omega; J_N = L} \pi(\underline{J}) = \frac{G(M-L, N-1)}{G(M,N)} \frac{(T_N)^L}{L!} \prod_{I=0}^{L} \max\left[1, \frac{I}{P_L}\right]$$

Here is an example written in FORTRAN for calculating these recursions:

```
C SYSTEM PARTITION FUNCTION VIA RECURSION
C CLOSED JACKSON NETWORK MODEL
C N NODES, M JOBS IN SYSTEM
C
DOUBLE PRECISION PART(MEAN,PROB,UTIL,IP,N,IC,T)
C
C  NODE 1    = CLERK NODE, IC CLERKS=M JOBS
C  NODE 2    = SYSTEM PROCESSOR NODE
C  T(K)      = MEAN TIME PER JOB AT NODE K,K=1,...,N
C  IP(K)     = NUMBER OF SERVERS AT NODE K,K=1,...,N
C  MEAN(K)   = MEAN NUMBER OF JOBS AT NODE K,K=1,...,N
C  PROB(K,J) = PROBABILITY J JOBS AT NODE K,K=1,...,N
C
DOUBLE PRECISION PROB,MEAN,SRATE,FRATE,T,G,SUM,TEMP
DIMENSION IP(N),MEAN(N),SRATE(N,IC+1),FRATE(N,IC+1)
DIMENSION PROB(N,IC+1),T(N)
C
C  INITIALIZATION
C
 DO 100 J=1,N
 SRATE(1,J)=1.0d0
 G(1,J)=0.0d0
 DO 110 I=2,IC+1
 G(I,J)=0.0d0
 IMIN=I-1
 IF(IMIN.GT.IP(J))IMIN=IP(J)
 SRATE(I,J)=SRATE(I-1,J)/(DBLE(FLOAT(IMIN))*T(J))
 110 CONTINUE
 110 CONTINUE
C
C  PERMUTATION OF NODE NUMBERS
C  IPERM DENOTES FINAL NODE IN ONE PERMUTATION
C  CYCLIC PERMUTATION: FIRST NODE 1, THEN NODE 2, ON UP
C  TO NODE N CHOSEN FOR FINAL NODE IN RECURSION
C
 IPERM=0
 150 CONTINUE
 IPERM=IPERM+1
 DO 200 J=1,N
 JCOM=J-IPERM
 IF(JCOM.LE.0)JCOM=N+JCOM
 DO 210 I=1,IC+1
 FRATE(I,JCOM)=SRATE(I,J)
 210 CONTINUE
 200 CONTINUE
C
C INITIALIZE FIRST ROW,COLUMN OF PARTITION FUNCTION MATRIX
C
 G(1,1)=1.0d0
 DO 300 I=1,IC+1
 G(I,1)=FRATE(I,1)
 300 CONTINUE
 DO 310 J=2,N
 G(1,J)=1.0d0
```

```
 310 CONTINUE
C
C FILL OUT BODY OF PARTITION MATRIX
C
 DO 400 J=2,N
 DO 410 I=2,IC+1
 G(I,J)=G(I,J-1)
 DO 420 K=2,I
 G(I,J)=G(I,J)+FRATE(K,J)*G(I-K+1,J-1)
 420 CONTINUE
 410 CONTINUE
 400 CONTINUE
C
C PROBABILITY DISTRIBUTION OF NUMBER AT NODE IPERM
C
 DO 500 K=1,IC+1
 PROB(K,IPERM)=FRATE(K,N)*G(IC+1-K+1,N-1)/G(IC+1,N)
 500 CONTINUE
 TEMP=0.0d0
 DO 510 K=1,IC+1
 TEMP=TEMP+PROB(K,IPERM)
 510 CONTINUE
 DO 520 K=1,IC+1
 PROB(K,IPERM)=PROB(K,IPERM)/TEMP
 620 CONTINUE
C
C MEAN NUMBER AT NODE IPERM
C
 MEAN(IPERM)=0.0d0
 DO 530 K=1,IC+1
 MEAN(IPERM)=MEAN(IPERM)+PROB(K,IPERM)*DBLE(FLOAT(K-1))
 530 CONTINUE
C
 IF(IPERM.LT.N)GO TO 150
 RETURN
 END
```

Figure 6.3.Partition Function FORTRAN Program

Two observations are in order:

- The recursion actually involves two aspects, the total number of jobs in the system, and the number of nodes
- The recursion developed here will handle more than two nodes; we will use it later for a three node system.

Here is one way to numerically approximate the following performance measures:

- the fraction of time greater than K jobs are buffered (waiting or in execution) at the processor stage

$$\pi[J_{proc} > K] = \sum_{J_{proc}=K+1}^{C} \pi[J_{clerk} = C - J_{proc}, J_{proc}]$$

- the fraction of time exactly K jobs are at the processor node

$$\pi[J_{proc}=K] = \pi[J_{clerk}=C-K, J_{proc}=K]$$

- the mean queue length at the processor stage

$$E[J_{proc}] = \sum_{J_{proc}=0}^{C} J_{proc}\,\pi[J_{clerk}=C-J_{proc}, J_{proc}]$$

- the mean number of jobs in execution

$$E[min(J_{proc}, P)] = \sum_{J_{proc}=0}^{C} min(J_{proc}, P)\,\pi[J_{clerk}=C-J_{proc}, J_{proc}]$$

- the mean number of jobs waiting

$$E[max(J_{proc}-P, 0)] = \sum_{J_{proc}=0}^{C} max(J_{proc}-P, 0)\,\pi[J_{clerk}=C-J_{proc}, J_{proc}]$$

- the fraction of time processors are busy doing work, their utilization

$$U_{processor} = \frac{1}{P}\left[1 - \sum_{K=0}^{P-1} \pi[J_{clerk}=C-K, J_{proc}=K]\right]$$

- the fraction of time both one or more clerks and one or more processors are simultaneously busy

$$\pi[J_{clerk}>0, J_{proc}>0] = 1 - \pi[J_{clerk}=C, J_{proc}=0] - \pi[J_{clerk}=0, J_{proc}=C]$$

- the mean throughput rate at the processor stage

$$mean\ \ throughput\ \ rate = \sum_{K=0}^{C} \pi[J_{clerk}=C-K, J_{proc}=K]\frac{min[K, P]}{T_{proc}}$$

In the plots below the mean throughput rate and mean delay per transaction as well as upper and lower bounds are calculated versus the number of clerks.

Figure 6.4.Mean Throughput Rate vs Number of Clerks

For the numbers chosen here, the mean value upper bound on mean throughput rate and lower bound on mean delay closely approximates the Jackson network analysis. Put differently, the Jackson network analysis refines our earlier analysis, showing in more detail (provided the underlying assumptions are valid) how great contention is for processors or clerks. The mean value bounds arise from *bottlenecks* that have a natural interpretation; oftentimes the designer can choose where the bottleneck should be, or move it from one point to another as needs warrant it. Jackson network analysis results in a smooth curve; mean value analysis results in a piecewise linear function. The *interpretation* of Jackson network analysis results in bottlenecks being identified, just as in mean value analysis. A key contribution of the

Figure 6.5.Mean Delay vs Number of Clerks

Jackson network analysis is to provide an independent check on what mean value bounds provide, because the underlying *assumptions* are more stringent or different, and our findings are more robust.

6.2.8 Additional Reading

[1] S.C.Bruell, G.Balbo, **Computational Algorithms for Closed Queueing Networks,** Elsevier North Holland, NY, 1980.

[2] J.P.Buzen, *Computational Algorithms for Closed Queueing Networks with Exponential Servers,* C.A.C.M., **16(9),** 527-531 (1973).

[3] R.B.Cooper, **Introduction to Queueing Theory,** pp.65-77, MacMillan, NY, 1972.

[4] K.G. Ramakrishnan, D. Mitra, *An Overview of PANACEA: A Software Package for Analyzing Markovian Queueing Networks,* Bell System Technical Journal, **61,** (10), I, 2849-2872 (1982).

[5] J.McKenna, D.Mitra, *Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage,* Bell System Technical Journal, **61,** (5), 661-683 (1982).

[6] J.McKenna, D.Mitra, K.G.Ramakrishnan, *A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, and Generalizations,* Bell System Technical Journal, **60** (5), 599-641 (1981).

6.3 Asymptotics

In order to gain insight into these formulae, we wish to examine two types of special cases. Both arise from examining the formulae presented earlier. In all cases, we assume

$$\Phi_{clerk}(J_{clerk}) = J_{clerk} \qquad \Phi_{proc}(J_{proc}) = min[J_{proc}, P]$$

In the first type of special case, the total workload offered is fixed, measured in jobs arriving per second, but the number of clerks becomes infinite. This is called an *infinite source* approximation, and it is also called a *Poisson* approximation.

$$\lambda \equiv \frac{C}{T_{think}} = fixed \quad C \to \infty \;\; T_{think} \to \infty$$

Since $T_{think} \to \infty$, each clerk spends more and more time reading and thinking and typing, but since we have more and more clerks, the total workload is constant.

A second type of limit is to fix *ratios C/M* and *C/P* and allow $C \to \infty$, with T_{think} and T_{proc} fixed. Should we have five clerks per processor, or ten clerks for two processors? Should there be three buffers for every five clerks, or six buffers for every ten clerks? Since the mean time intervals T_{think} and T_{proc}, are fixed, the activity of each clerk stays the same, unlike the first case. The advent of inexpensive processors may make this type of analysis of great practical import in times to come.

6.3.1 Infinite Source Asymptotics If we fix $\lambda \equiv C/T_{think}$ and allow $C \to \infty$, then we find that the fraction of time J_{proc} jobs are in the system is given by

$$\pi(J_{proc}) = \frac{1}{G} \frac{(\lambda T_{proc})^{J_{proc}}}{J_{proc}!} \prod_{K=0}^{J_{proc}} \max\left[1, \frac{K}{P}\right] \quad J_{proc}=0,...,M$$

The distribution of jobs in the system at the instant a job arrives is now *identical* to the long term time averaged number of jobs in the system. If $M=C \to \infty$, the fraction of time that a task must wait greater than X time units is

$$PROB[T_W > X] = \begin{cases} C(P,A) exp[-(1-U)XP/T_{proc}] & A<P \\ \infty & A \geq P \end{cases} \quad M \to \infty$$

$$A \equiv E[J_{proc}] = \lambda T_{proc} \quad U \equiv A/P$$

The units of A are called *Erlangs* in honor of the Scandinavian teletraffic engineer A.K.Erlang, who was a pioneer in engineering voice telephone transmission and switching systems. Since A is the mean number of jobs in the processor queue, either waiting to run or in execution, if the mean number of jobs is less than the actual number of processors then the processors can keep up with the work, and otherwise they cannot. U is the fraction of time each processor is busy, its' *utilization.*

The expression $C(P,A)$, called *Erlang's delay function* or *Erlang's function of the second kind*, is given by

$$C(P,A) = \frac{A^P/(P-1)!(P-A)}{\sum_{K=0}^{P-1} \frac{A^K}{K!} + \frac{A^P}{(P-1)!(P-A)}}$$

This is a very important function, because we can compute many performance measures directly once we know it:

- The fraction of time a task waits at all is

  $$PROB[T_W > 0] = C(P,A) \qquad A<P$$

  For a single processor, the fraction of time a task waits is the fraction of time the processor is busy; for multiple processors, the fraction of time a task waits at all depends upon finding all the processors busy first, and hence is more complicated.

- The mean waiting time for a task is given by

  $$E(T_W) = \frac{C(P,A)T_{proc}}{P-A} = \frac{C(P,A)T_{proc}}{(1-U)P} \qquad A<P$$

- The mean queueing time for a task is the sum of its mean waiting and service time:

  $$E(T_Q) = E(T_W) + T_{proc}$$

For one processor, we see

$$C(P=1,A) = A \quad \rightarrow \quad E(T_W) = \frac{AT_{proc}}{1-A} \quad \rightarrow \quad E(T_Q) = \frac{T_{proc}}{1-A}$$

The mean queueing time equals the mean execution time, multiplied by a *stretching* factor of $1/1-A$. Under light loading $A<<1$ the stretching factor is one, while as $A \to 1$ the stretching becomes larger and larger than one.

6.3.2 Modem Pooling In a dial up online computer system, users dial up the computer over voice telephone lines. Each user will need a modem to connect to the computer. The mean holding time per call is one minute. The mean arrival rate of calls is five calls per minute. We wish to have sufficient modems so that the fraction of time that a user waits at all for a modem is under ten per cent. How many modems are required?

The offered load to the system is given by

$$A = 5 \; calls \; per \; minute \times one \; minute \; per \; call = 5 \; Erlangs$$

We wish to find P such that

$$C(P,A{=}5) < 0.10$$

If we simply try various numbers, starting with P=5 since we expect to have five busy modems on the average most of the time, we see that eight or more modems is sufficient to meet the delay criterion.

6.3.3 P=M Loss System A typical problem in voice telephony is to determine how many circuits are needed to handle the load. Telephone calls arrive at random instants of time, and will tie up or hold a telephone line for three to five to ten minutes. If all lines are busy, it is highly unlikely a line will become free within the next few seconds, and the new arrival is blocked or rejected. A natural model of this situation is calls or jobs arrive for service at a group of $P{=}M$ parallel servers or processors, with one buffer per processor. The mean service time of a job is denoted by $E(T_S)$. If all servers are busy when an arrival occurs, the new arrival is rejected or blocked and presumably will retry later.

What goals are to be achieved? >From the point of view of a customer, the fraction of time an arrival is rejected should be acceptably low. What system oriented criteria are of interest? The mean number of busy processors, and the fraction of time each processor is busy, its utilization, should be acceptably high.

The interarrival times between jobs are assumed to be independent identically distributed exponential random variables, i.e., the arrival statistics can be modeled by a Poisson distribution. The mean throughput rate of completing tasks is denoted by the carried mean throughput rate, to distinguish it from the offered load, which will be bigger than the carried load because some offered load was rejected. The carried mean throughput rate is

$$\lambda_{carried} = \lambda_{offered}[1 - B(P,A)]$$

where as above A is the mean *offered* load to the system $A = \lambda E(T_S)$ to distinguish it from the *carried (actual)* load $\hat{A}$ given by

$$\hat{A} = A[1 - B(P,A)] = mean \; number \; of \; busy \; servers$$

In some cases, only the carried load can be measured, and then we must work backwards to infer the offered load. As long as the fraction of attempts blocked is negligible, the above expression provides a convenient rough cut for doing so. $B(P,A)$ is called *Erlang's blocking function* or *Erlang's function of the first kind*, and is given by

$$B(P,A) = \frac{A^P/P!}{\sum_{K=0}^{P} A^K/K!}$$

$B(P,A)$ is the fraction of time that all P processors are busy, or it is the fraction of time a new arrival is blocked or rejected.

If a task is accepted, it waits no time at all to be processed. If a task is rejected, it waits forever and is never processed. The queueing time, the sum of the waiting time and the service time, is either equal to the service time if the task is accepted, or is infinite if the task is rejected.

For values of $\lambda_{offered}$ that result in small blocking the mean throughput rate $\lambda_{carried}$ equals $\lambda_{offered}$; for $\lambda_{offered} \to \infty$ the mean throughput rate approaches $P/E(T_S)$. For intermediate values, the exact expressions must be used. The mean waiting time of accepted tasks is zero, while the mean queueing

time of accepted tasks is $E(T_S)$.

6.3.4 Circuit Switching We must configure a data computer communication system with sufficient private communication lines so that the fraction of time call attempts are blocked because no line is available is sufficiently small. Measurement data suggest a typical call lasts for forty five minutes, with an average of ten calls per hour being attempted. We wish to examine two levels of service, one where a call attempt is blocked no more than ten per cent of the time, and a second where a call attempt is blocked no more than one per cent of the time. How many circuits are needed?

The mean number of calls in progress is

$$A = (3/4)\ hours/call \times 10\ calls/hour = 7.5\ calls\ or\ Erlangs$$

Hence, at least 7.5, i.e., eight circuits are needed just to carry the load.

Using an Erlang blocking analysis, in order to meet the goal of calls being blocked no more than ten per cent of the time, we need ten circuits. In order to meet the goal of calls being blocked no more than one per cent of the time, we need fourteen circuits.

6.3.5 Relationships Between Loss and Delay Systems There are many relationships between loss and delay systems. Here is one such formula relating Erlang's loss formula and Erlang's delay formula:

$$C(P,A) = \frac{A\ B(P-1,A)}{P + A[B(P-1,A)-1]}$$

Thus, if we know how to calculate or numerically approximate Erlang's loss formula, we can also numerically approximate Erlang's delay formula.

6.3.6 Numerical Approximation of Erlang's Blocking Function We now present an algorithm for numerically approximating the Erlang blocking function, due to D.Jagerman, that can be readily programmed on a pocket calculator or other machine. The implementation presented here is in FORTRAN, and requires two constants, one for underflow and one for overflow.

```
      DOUBLE PRECISION FUNCTION BERL(N,A)
C
C INPUTS
C
C   P--NUMBER OF PROCESSORS
C   A--OFFERED LOAD IN ERLANGS
C
C OUTPUT
C
C   BERL--ERLANG BLOCKING FUNCTION
C
C  ALGORITHM--ITERATE ON THE INVERSE OF BERL
C
      DOUBLE PRECISION BERL,A,XN,XA,TERM,SUM,DMIN,DMAX
C
C  DMIN--CHOSEN TO MACHINE ACCURACY
C  DMAX--RECIPROCAL OF SMALLEST BLOCKING OF INTEREST
C
      DMIN=1.0D-10
      DMAX=1.0d9
C
C  PARAMETER INITIALIZATION
C
      XN=DBLE(FLOAT(N))
      XA=1.0d0/A
```

```
      TERM=XN*XA
      SUM=1.0d0
C
C  MAIN LOOP
C
   10 CONTINUE
      SUM=SUM+TERM
      XN=XN-1.0d0
      TERM=TERM*XN*XA
      IF((TERM.LT.DMIN).OR.(SUM.GT.DMAX)) GO TO 20
      GO TO 10
C
C  TO REACH HERE MUST HAVE COMPLETED
C
   20 CONTINUE
      BERL=1.0d0/SUM
      RETURN
      END
```

Figure 6.6.Erlang Blocking Function FORTRAN Program

6.3.7 Additional Reading

[1] D.L.Jagerman, *Some Properties of the Erlang Loss Function,* Bell System Technical Journal, **53,** (3), pp.525-551, 1974.

6.3.8 C/M and C/P Fixed, $C \to \infty$ The remaining asymptotic case, where the ratios are fixed for *C/M* and *C/P* but $C \to \infty$, is summarized by the following formulae:

- The fraction of time an attempt is blocked is given by

$$B = 1 - \min\left[1, \frac{C}{T_{think} + \min\left[\frac{CT_{proc}}{P} - T_{think}, T_{proc}\,\frac{M}{P}\right]}\right]$$

- The fraction of time a processor is busy is given by

$$U = utilization/processor = \min\left[1, \frac{CT_{think}}{P\,(T_{think}+T_{proc})}\right]$$

- The mean rate of submitting jobs is given by

$$\lambda_{offered} = mean\ \ job\ \ submission\ \ rate = \frac{PU}{T_{proc}}$$

- The mean throughput rate of jobs is given by

$$\lambda_{carried} = \frac{\lambda_{offered}}{1-B} = \frac{C}{(1-B)E(T_Q) + T_{think}}$$

- The mean delay per job is given by

$$E(T_Q) = \max\left[T_{proc}, \min\left[\frac{CT_{proc}}{P} - T_{think}, \frac{M}{P}T_{proc}\right]\right]$$

6.4 Teletraffic Engineering of an Automatic Call Distributor

An automatic call distributor, as the name implies, automatically distributes incoming calls among various available agents. Customers dial a telephone number, and the calls are routed through the voice telephony network to the call distributor. The call distributor either blocks or rejects a call attempt if all incoming circuits or trunks to the call distributor are busy (either with calls waiting for an agent or calls between agents and customers). Customers are queued for service by an agent. If all agents in a given group are busy, the customer is queued for a receiver that plays a recorded message asking the customer to be patient until an agent becomes available; in many cases, music is played while the customer is waiting. After the agent handles the customer request there is often some clean up work associated with that particular task, after which the agent is available for handling the next incoming call.

Automatic call distributors are currently in wide use. Some examples are

- airlines--for handling reservations, time of arrival and departure of flights, travel agent activity, charter and tour operators, charging or billing information, internal administrative use
- hotels and motels--for handling reservations at a nationwide chain, billing questions, travel agent activity
- credit authorization and verification--for a variety of bank credit cards, which require authorization if the purchase price exceeds a given limit
- message service--major companies frequently have a centralized group of agents handle all messages rather than tie up a secretarial pool with this job

6.4.1 Model The model inputs required here would be the mean arrival rate of calls, the mean time a customer would wait for an agent, and the mean time to handle a customer plus any related work (such as logging the result of the customer inquiry).

How can the number of trunks or links and agents be chosen such that the system meets both a given blocking criterion and a mean delay criterion for each customer?

Here is a summary of model parameters for two illustrative cases, one for credit authorization, and one for airline reservations and flight information.

Table 6.1.Model Parameters

Attribute	*Credit Authorization*	*Airline*
Announcement Time	10-20 Seconds	20-30 seconds
Talking Time	30-60 seconds	100-200 seconds

Goals are specified for a peak business hour load. During the first year of operation, the peak business hour mean call arrival rate is four per minute. This will increase to eight calls per minute during the second year, and twenty calls per minute during the third year. Each call that is accepted can wait for fifteen seconds, on the average, and each inquiry will require a mean of one minute per agent. We denote by T_{wait} the mean waiting time (e.g., to hear an announcement), and T_{talk} the mean time to handle a customer request.

6.4.2 State Space The system state space is denoted by Ω. Elements are integer valued pairs denoted by $\underline{J}=(J_{wait},J_{talk})$, where J_{wait} denotes the number of calls waiting for an agent, and J_{talk} denotes the number of busy agents (busy handling customer requests). Since there are T trunks and S agents, the state space is given by:

$$\Omega = \{(J_{wait},J_{talk}) \mid J_{wait}+J_{talk} \leq T, J_{talk} \leq S\}$$

6.4.3 Mean Value Analysis The mean number of trunks held equals the mean arrival rate of accepted calls multiplied by the mean trunk holding time. Since trunks are held for an announcement (if no agent is available), and for handling customer requests, we can write

$$E[min(J_{wait}+J_{talk},T)] = \lambda(T_{wait}+T_{talk})$$

The mean number of busy agents is given by

Figure 6.7.Call Distributor Block Diagram

Figure 6.8.Call Distributor Queueing Network

$$E[min(J_{talk}, S)] = \lambda T_{talk}$$

This suggests the following two bottlenecks:

- If trunks are the bottleneck, then

$$\lambda = \frac{T}{T_{wait} + T_{talk}}$$

For the number here, the required mean number of trunks is summarized in the table below:

Table 6.2.Number of Required Trunks

Year	*Trunks*
1	5 trunks
2	10 trunks
3	40 trunks

- If agents are the bottleneck, then

$$\lambda = \frac{S}{T_{talk}}$$

For the numbers described above, the mean number of required agents is summarized in the table below:

Table 6.3.Number of Required Agents

Year	*Agents*
1	4 agents
2	8 agents
3	20 agents

This analysis suggests that 1.25 trunks are required for every agent, i.e., $(T_{talk}+T_{wait})/T_{wait}$ trunks per agent.

6.4.4 Jackson Network Analysis What if we use an Erlang blocking analysis to determine the number of trunks required? This means that we are looking for the largest value of T such that

$$B[T,\lambda(T_{talk}+T_{wait})] \leq \varepsilon_{trunk}$$

where ε_{trunk} is the fraction of blocked or rejected call attempts. The table below summarizes these calculations:

Table 6.4.Number of Trunks Required

	Fraction of Attempts Rejected		
Year	*0.001*	*0.01*	*0.1*
1	14 trunks	11 trunks	8 trunks
2	21 trunks	18 trunks	13 trunks
3	41 trunks	36 trunks	28 trunks

The interpretation of the data is that during year one 8-14 trunks suffice, during year two 13-21 trunks, and during year three 28-41 trunks. Since trunks may come in bundles or integral multiples of one trunk (such as six trunks or twenty four per trunk group, with all purchases in multiples of trunk groups) economic considerations must be addressed here. For example, two trunk groups of six trunks each would suffice for the first year, four trunk groups for the second year, and seven trunk groups for the third year. On the other hand, the *increment* in carried load is simply the difference in blocking for $T+1$ trunks versus T trunks, multiplied by the carried load.

We next use Erlang's delay formula to calculate the number of agents required to meet a given delay criterion. This means we are looking for the largest value of S such that

$$C(S,\lambda T_{talk}) \leq \varepsilon_{agent}$$

where ε_{agent} is the agent delay criterion. The table below summarizes these calculations:

Table 6.5.Number of Agents Required

	Fraction of Time Call Waits > 0		
Year	*0.1*	*0.2*	*0.3*
1	8 agents	7 agents	6 agents
2	13 agents	12 agents	11 agents
3	22 agents	20 agents	19 agents

This table makes it clear fewer agents are needed than trunks, typically one agent for every 1.1 to 1.5 trunks. In all cases, the number of agents and trunks needed is larger than the mean value analysis suggests. On the other hand, the mean value analysis appears to be a reasonable first cut at answering how many trunks and agents are needed, one that can be refined as additional information comes to light.

6.4.5 Additional Reading

[1] L.Kosten, **Stochastic Theory of Service Systems,** pp.26-40, Pergamon Press, London, 1973.

[2] P.A.Brown, D.W.Clark, *Automatic Call Distribution System-ASDP 162,* Telecommunications Journal of Australia, **29** (3), 245-255 (1979).

6.5 Teletraffic Engineering of an Automatic Call Distributor

An automatic call distributor, as the name implies, automatically distributes incoming calls among various available agents. Customers dial a telephone number, and the calls are routed through the voice telephony network to the call distributor. The call distributor either blocks or rejects a call attempt if all incoming circuits or trunks to the call distributor are busy (either with calls waiting for an agent or calls between agents and customers). Customers are queued for service by an agent. If all agents in a given group are busy, the customer is queued for a receiver that plays a recorded message asking the customer to be patient until an agent becomes available; in many cases, music is played while the customer is waiting. After the agent handles the customer request there is often some clean up work associated with that particular task, after which the agent is available for handling the next incoming call.

Automatic call distributors are currently in wide use. Some examples are

- airlines--for handling reservations, time of arrival and departure of flights, travel agent activity, charter and tour operators, charging or billing information, internal administrative use
- hotels and motels--for handling reservations at a nationwide chain, billing questions, travel agent activity
- credit authorization and verification--for a variety of bank credit cards, which require authorization if the purchase price exceeds a given limit
- message service--major companies frequently have a centralized group of agents handle all messages rather than tie up a secretarial pool with this job

6.5.1 Model The model inputs required here would be the mean arrival rate of calls, the mean time a customer would wait for an agent, and the mean time to handle a customer plus any related work (such as logging the result of the customer inquiry).

How can the number of trunks or links and agents be chosen such that the system meets both a given blocking criterion and a mean delay criterion for each customer?

Here is a summary of model parameters for two illustrative cases, one for credit authorization, and one for airline reservations and flight information.

Table 6.1.Model Parameters

Attribute	*Credit Authorization*	*Airline*
Announcement Time	10-20 Seconds	20-30 seconds
Talking Time	30-60 seconds	100-200 seconds

Goals are specified for a peak business hour load. During the first year of operation, the peak business hour mean call arrival rate is four per minute. This will increase to eight calls per minute during the second year, and twenty calls per minute during the third year. Each call that is accepted can wait for fifteen seconds, on the average, and each inquiry will require a mean of one minute per agent. We denote by T_{wait} the mean waiting time (e.g., to hear an announcement), and T_{talk} the mean time to handle a customer request.

6.5.2 State Space The system state space is denoted by Ω. Elements are integer valued pairs denoted by $\underline{J}=(J_{wait},J_{talk})$, where J_{wait} denotes the number of calls waiting for an agent, and J_{talk} denotes the number of busy agents (busy handling customer requests). Since there are T trunks and S agents, the state space is given by:

$$\Omega = \{(J_{wait},J_{talk}) \mid J_{wait}+J_{talk} \leq T, J_{talk} \leq S\}$$

Figure 6.7.Call Distributor Block Diagram

Figure 6.8.Call Distributor Queueing Network

6.5.3 Mean Value Analysis The mean number of trunks held equals the mean arrival rate of accepted calls multiplied by the mean trunk holding time. Since trunks are held for an announcement (if no agent is available), and for handling customer requests, we can write

$$E[min(J_{wait}+J_{talk},T)] = \lambda(T_{wait}+T_{talk})$$

The mean number of busy agents is given by

$$E[min(J_{talk},S)] = \lambda T_{talk}$$

This suggests the following two bottlenecks:

- If trunks are the bottleneck, then

$$\lambda = \frac{T}{T_{wait}+T_{talk}}$$

For the number here, the required mean number of trunks is summarized in the table below:

Table 6.2.Number of Required Trunks

Year	*Trunks*
1	5 trunks
2	10 trunks
3	40 trunks

- If agents are the bottleneck, then

$$\lambda = \frac{S}{T_{talk}}$$

For the numbers described above, the mean number of required agents is summarized in the table below:

Table 6.3.Number of Required Agents

Year	*Agents*
1	4 agents
2	8 agents
3	20 agents

This analysis suggests that 1.25 trunks are required for every agent, i.e., $(T_{talk}+T_{wait})/T_{wait}$ trunks per agent.

6.5.4 Jackson Network Analysis What if we use an Erlang blocking analysis to determine the number of trunks required? This means that we are looking for the largest value of T such that

$$B[T,\lambda(T_{talk}+T_{wait})] \leq \varepsilon_{trunk}$$

where ε_{trunk} is the fraction of blocked or rejected call attempts. The table below summarizes these calculations:

Table 6.4.Number of Trunks Required

	Fraction of Attempts Rejected		
Year	*0.001*	*0.01*	*0.1*
1	14 trunks	11 trunks	8 trunks
2	21 trunks	18 trunks	13 trunks
3	41 trunks	36 trunks	28 trunks

The interpretation of the data is that during year one 8-14 trunks suffice, during year two 13-21 trunks, and during year three 28-41 trunks. Since trunks may come in bundles or integral multiples of one trunk (such as six trunks or twenty four per trunk group, with all purchases in multiples of trunk groups) economic considerations must be addressed here. For example, two trunk groups of six trunks each would suffice for the first year, four trunk groups for the second year, and seven trunk groups for the third year. On the other hand, the *increment* in carried load is simply the difference in blocking for T+1 trunks versus T trunks, multiplied by the carried load.

We next use Erlang's delay formula to calculate the number of agents required to meet a given delay criterion. This means we are looking for the largest value of S such that

$$C(S,\lambda T_{talk}) \leq \varepsilon_{agent}$$

where ε_{agent} is the agent delay criterion. The table below summarizes these calculations:

Table 6.5.Number of Agents Required

	Fraction of Time Call Waits > 0		
Year	*0.1*	*0.2*	*0.3*
1	8 agents	7 agents	6 agents
2	13 agents	12 agents	11 agents
3	22 agents	20 agents	19 agents

This table makes it clear fewer agents are needed than trunks, typically one agent for every 1.1 to 1.5 trunks. In all cases, the number of agents and trunks needed is larger than the mean value analysis suggests. On the other hand, the mean value analysis appears to be a reasonable first cut at answering how many trunks and agents are needed, one that can be refined as additional information comes to light.

6.5.5 Additional Reading

[1] L.Kosten, **Stochastic Theory of Service Systems,** pp.26-40, Pergamon Press, London, 1973.

[2] P.A.Brown, D.W.Clark, *Automatic Call Distribution System-ASDP 162*, Telecommunications Journal of Australia, **29** (3), 245-255 (1979).

6.6 General Single Class Jackson Network Model

Clerks at terminals submit one type of job to a computer communication system.

Figure 6.9.System Block Diagram

A queueing network model of this system is as follows: A source submits one type of job to a system. There are a total of P_0 sources. Each job enters a staging queue: if there are less than M jobs in the system, the job immediately enters the system, and otherwise waits for the number of jobs in the system to drop below M. Once a job enters the system, it visits a number of nodes or stations within the network, and after being completely processed exits the system, where the cycle begins anew. The number of processors or servers at node K is denoted by P_K.

6.6.1 State Space The number of jobs either queued or in execution at time t at node $K=1,...,N$ is denoted by $J_K(t)$. Elements $J_K(t)$ are nonnegative integers, and are members of a state space denoted by Ω. The number of sources with a job is denoted by J_0; if there is staging, then

$$number\ of\ jobs\ in\ staging\ queue\ = min\,[0,M\ -\ J_0]$$

Three cases will be dealt with:

[1] Jobs can be submitted to the system sufficiently fast such that the system *always* contains M jobs. The state space for this case is given by

$$\Omega = \{(J_1,...,J_N)\mid J_K \geq 0, 1 \leq K \leq N; \sum_{K=1}^{N} J_K = M\}$$

[2] The system always has room for a job, i.e., $M \geq P_0$, so there is never a job in the staging queue. The state space for this case is given by

Figure 6.10.Queueing Network Model Block Diagram

$$\Omega = \{(J_1,...,J_N) \mid J_K \geq 0, 0 \leq K \leq N, \sum_{K=0}^{N} J_K = P_0\}$$

[3] One limiting case that is analytically tractable is a so called *open* network, where only the total job arrival rate is known, denoted by $\lambda = P_0/T_0$, with $P_0 \rightarrow \infty$. The state space is now given by

$$\Omega = \{(J_1,...,J_N) \mid J_K \geq 0, 1 \leq K \leq N\}$$

It remains to specify the workload in greater precision, and the scheduling or arbitration rule for each node.

6.6.2 Workload The system at any instant of time contains $|\underline{J}|$ jobs, where

$$|\underline{J}| = \sum_{K=1}^{N} J_K$$

Given the system contains $|\underline{J}|$ jobs at time t and that the number of jobs at each node is observed over time, the instantaneous job arrival rate at time t is given by

$$\textit{instantaneous job arrival rate} = \frac{\Phi_0(|\underline{J}(t)|)}{T_0} > 0$$

Each distinct visit to node K requires a possibly different amount of execution time by a server at that node. The random variable τ_K denotes the service time per visit to node K, which is a sequence of independent identically distributed random variables drawn from a common distribution denoted by $H_K(X)$:

$$PROB[\tau_K \leq X] = H_K(X)$$

The mean of τ_K is denoted by $E(\tau_K)$.

Each job is routed through the network probabilistically: once a job is serviced at node K, it migrates to node J with probability R_{KJ}, where $K=1,...,N$ and $J=0,...,N$, i.e., a job can exit the system and return to the source node when $J=0$. An equivalent description of this process is that each job makes V_K visits to node K, and hence the total mean amount of service at node K, denoted by T_K, is $V_K E(\tau_K)$.

6.6.3 Scheduling Policy for Arbitrating Contention When a job arrives at queue K and finds J_K jobs present, it is inserted into place I with probability $\delta_K(I, J_K+1)$ where $I=1,...,J_K+1$. All tasks in position $I,...,J_K$ are shifted to the position with the next higher index. When a job finishes execution at position I, all jobs at position $I+1,...,J_K$ are shifted to the position with the next lower index.

When J_K jobs are present at node K, work is executed at a total rate (measured in seconds of processing per second, so that it is dimensionless), of $\Phi_K(J_K)$. A job in position I out of J_K at node K is processed at a rate of $\gamma_K(I,J_K)\phi_K(J_K) \le 1$, again measured in seconds of processing per second, such that

$$\sum_{I=1}^{J_K} \gamma_K(I,J_K) = 1$$

or in words that *some* jobs are receiving *all* the available processing capacity.

We will not allow all possible scheduling rules, but only *two* classes. The first class of rules allows the service time distribution to be *arbitrary* but the scheduling policy must be *balanced* in the sense that

$$\gamma_K(I,J_K) = \delta_K(I,J_K)$$

Examples of scheduling rules that meet these conditions are

- Processor sharing: if J_K jobs are present, each receives $1/J_K$ the fraction of available processing capacity:

$$\gamma_K(I,J_K) = \frac{1}{J_K}$$

- Last come first serve preemptive resume: the last arrival is serviced first, with subsequent arrivals preempting service, and service is resumed at the point of interruption:

$$\gamma_K(I,J_K) = \begin{cases} 1 & I=J_K \\ 0 & otherwise \end{cases}$$

- An infinite server group, i.e., there is always available one server to execute a job, with no waiting or queueing:

$$\gamma_K(I,J_K) = 1$$

The second type of scheduling policy is to allow jobs to be executed in order of arrival, but now the service time distribution must be *exponential:*

$$H_K(\tau_K \le X) = 1 - exp[-X/E(\tau_K)]$$

6.6.4 Finite Source Arrival Statistics with Staging Analysis Granted the previous assumptions, the long term time averaged fraction of time the system is in state $\underline{J}$ is denoted by $\pi(\underline{J})$, where

$$\pi(\underline{J}) = \pi(J_1,...,J_N) = \frac{1}{G} \prod_{I=0}^{|\underline{J}|-1} \frac{\Phi_0(I)}{T_0} \prod_{K=1}^{N} \prod_{I=1}^{J_K} \frac{T_K}{\Phi_K(I)}$$

$$\sum_{\underline{J}\in\Omega} \pi(\underline{J}) = 1 \qquad \pi(\underline{J}) \ge 0$$

If $\Phi_K(I)=min(I,P_K)$, and the system always contains M jobs, this can be written as

$$\pi(\underline{J}) = \frac{1}{G} \prod_{K=1}^{N} \frac{T_K^{J_K}}{J_K!} \prod_{I=0}^{J_K} \max\left[1, \frac{I}{P_K}\right] \qquad \underline{J}\in\Omega$$

For the case where there is a staging queue with P_0 sources, this can be written as

$$\pi(\underline{J}) = \frac{1}{G} \prod_{K=0}^{N} \frac{T_K^{J_K}}{J_K!} \prod_{I=0}^{J_K} \max\left[1, \frac{I}{P_K}\right]$$

6.6.5 Asymptotics One natural type of asymptotic analysis is to fix the total arrival rate at λ while allowing the number of sources to become infinite:

$$\lambda \equiv \frac{P_0}{T_0} = fixed \quad P_0, T_0 \to \infty$$

The resulting form of the long term time averaged distribution simplifies:

$$\lim_{\lambda\ fixed} \pi(\underline{J}) = \begin{cases} \prod_{K=1}^{N} \prod_{I=1}^{J_K} \frac{\lambda T_K}{J_K!} \prod_{I=0}^{J_K} \max\left[1, \frac{I}{P_K}\right] & \lambda < \min_K \frac{P_K}{T_K} = \lambda_{\max} \\ 0 & \lambda \geq \min_K \frac{P_K}{T_K} = \lambda_{\max} \end{cases}$$

This last condition needs some elaboration: if $\lambda<\lambda_{max}$, the mean value upper bound on the maximum mean throughput rate, then all nodes can keep up with the workload, and otherwise the number of jobs at the slowest or bottleneck nodes exceeds any threshold. This can be seen from evaluating the partition function: the partition function is finite provided $\lambda<\lambda_{max}$, and otherwise is infinite.

The total mean queueing delay of a job is the sum of the individual queueing delays at each node:

$$E(T_Q) = \sum_{K=1}^{N} E(T_{Q,K})$$

where the mean queueing delay (waiting plus execution) at node K is given by

$$E(T_{Q,K}) = T_K \; D(P_K, U_K) \quad K=1,...,N$$

U is the utilization, the fraction of time, *each* server or processor at node K is busy, and $D(P_K, U_K)$ is the *stretching factor* for node K, i.e., it shows how much T_K is inflated or stretched to account for queueing delays.

$$D(P, U) = \frac{C(P, P\ U)}{P(1-U)} + 1$$

$C(P, P\ U)$ is the Erlang delay function discussed earlier. This can be readily approximated numerically. For $U<<1$, light loading, $D(P,U)\approx 1$, i.e., there is virtually no stretching of the execution time, while under heavy loading $U\to 1$, $D(P,U)\to\infty$, i.e., the waiting time can swamp the execution time.

What if we plot the mean throughput rate versus the degree of multiprogramming defined to be $|\underline{J}|$? We see for $|\underline{J}|=1$ the mean throughput rate is simply the reciprocal of the *total* time to complete one job, and hence increasing the degree of multiprogramming will *always* be less than $|\underline{J}|$ times the mean throughput for one job. At the other extreme, as the degree of multiprogramming becomes larger and larger, one or more resources will be completely utilized or busy with work, and we see

$$\lambda = \min_I \left[\frac{P_I}{T_I}\right] \quad |\underline{J}| \to \infty$$

The nodes that reach complete utilization first are called *bottleneck* nodes. The queueing network analysis allows us to interpolate between these two asymptotic regions, as shown in the figure below. One figure of merit discussed earlier is the *stretching factor* which we defined as the ratio of the upper bound on mean throughput rate due to the number of jobs being a bottleneck over the actual mean throughput rate:

$$stretching\ \ factor = \frac{job\ \ bottleneck\ \ mean\ \ throughput\ \ bound}{actual\ \ mean\ \ throughput\ \ rate} = \frac{J\lambda(J=1)}{\lambda(J>1)}$$

We want to operate with a stretching factor close to one, i.e., as we add more jobs, the mean throughput rate scales or increases *linearly* with J, and we stay close to the ideal bound. The design question is to intelligently choose the bottlenecks so that this in fact happens!

6.7 Pipeline Processing

The configuration studied here is a pipeline of N processors, one processor per stage. Jobs arrive at the first stage, are processed, and move on to the second stage, until completely executed. What impact

Figure 6.11.Mean Throughput Rate vs Degree of Multiprogramming

Figure 6.12.Pipeline Queueing Network Block Diagram

might fluctuations have on the performance of such a system?

6.7.1 Analysis The model ingredients are

- Jobs arrive according to simple Poisson statistics with mean arrival rate λ jobs per unit time
- Each job requires $T_{S,K}$ service at node K, where the service times are independent identically distributed random variables
- Jobs are serviced in order of arrival

If there is no contention at all, to completely execute a job will require on the average $E(T_1)+ E(T_2)+ ...+ E(T_N)$ time units.

We now wish to study the impact that fluctuations about the mean value of the processing times can have on the time required to completely execute a job. First, if the processing times at each stage are exponentially distributed, which might be the case if sometimes one branch were followed and another time another branch were followed, then the mean time that a job is waiting while other jobs are being executed is denoted by $E(T_W)$ and is given by

$$E(T_{W,exponential})=\sum_{k=1}^{N}\frac{\lambda E^2(T_{S,K})}{1-\lambda E(T_{S,K})}$$

For example, if $E(T_{S,1})=...=E(T_{S,N})\equiv E(T_S)$, i.e., the processing times at each stage had the same mean value, then

$$E(T_{W,exponential})=\frac{N\lambda E^2(T_S)}{1-\lambda E(T_S)} \quad E(T_S)\equiv E(T_{S,1})=...=E(T_{S,N})$$

On the other hand, if the processing times at each stage are constant, i.e., no fluctuations about the mean, then the mean time that a job is waiting while other jobs are being executed is denoted by $E(T_{W,constant})$ and is given by

$$E(T_{W,constant})=\frac{\lambda E^2(T_{S,max})}{2(1-\lambda E(T_{S,max}))} \quad E(T_{S,max})=max(E(T_{S,1}),...,E(T_{S,N}))$$

For example, if $E(T_{S,1})=...=E(T_{S,N})$, then

$$E(T_{W,constant})=\frac{\lambda E^2(T_{S,max})}{2(1-\lambda E(T_{S,max}))}$$

Note that this can be N/2 times that for the exponential case if the fluctuations about the mean processing times are reduced from the exponential spread to the constant processing time case, with similar much more dramatic payoff if one wishes to cut the ninetieth percentile of the waiting time distribution function, averaged over a long term time interval.

EXERCISE: Show that the result for constant service at each stage is true, by showing that all the queueing or waiting occurs as if all jobs were queued at the stage with the longest service time, independent of the order of nodes in the pipeline.

It is interesting to note that although the delay statistics in pipeline processing can be quite sensitive to the processing time statistics at each stage of execution, the maximum mean throughput rate is the same for either example above:

$$maximum\ mean\ throughput\ rate\ = 1/max(E(T_{S,1}),E(T_{S,2}),...,E(T_{S,N}))$$

This means that *small* changes in the mean service rate result in *small* changes in the maximum mean throughput rate but **not** in the mean delay as the underlying execution time distributions at each stage are varied.

6.7.2 An Example Suppose a computer communication system consists of N steps for each job, and each step requires a mean of one second. If jobs arrive at the first stage of the pipeline at the rate of one job every two seconds, then we can compare the performance of this system for different number of stages in the pipeline and to fluctuations about the mean service time, and the results are summarized in the table below:

Table 6.6.Mean Delay with λ=0.5 *jobs/sec*

Number	*Mean Waiting Time* $E(T_W)$		*Mean Queueing Time* $E(T_Q)$	
of Stages	*Constant*	*Exponential*	*Constant*	*Exponential*
3	0.5 sec	3.0 sec	3.5 sec	6.0 sec
4	0.5 sec	4.0 sec	4.5 sec	7.0 sec
5	0.5 sec	5.0 sec	5.5 sec	8.0 sec

This makes it evident that the mean waiting grows with the number of stages if the fluctuations are severe. As a check on this study, we allow the first stage to require two units of service, the second stage one half unit of service, and all the rest of the stages require one unit of service, with the calculations summarized below:

Table 6.7.Mean Delay with λ=0.5 *jobs/sec*

Number	*Mean Waiting Time* $E(T_W)$		*Mean Queueing Time* $E(T_Q)$	
of Stages	*Constant*	*Exponential*	*Constant*	*Exponential*
3	2.25 sec	5.67 sec	5.25 sec	8.67 sec
4	2.25 sec	7.67 sec	6.25 sec	10.67 sec
5	2.25 sec	9.67 sec	7.25 sec	12.67 sec

This makes evident that the mean waiting is significantly impacted by the imbalance at the first two

stages in the service time, for both the constant and exponential distribution. However, the constant service case is significantly *less* impacted than the exponential case.

6.7.3 Link Level Flow Control Consider a transmitter and a receiver that are connected via a data network of N stages or nodes, with messages at stage K requiring a constant amount of processing, $E(T_{S,K}), K=1,...,N$. In order to insure that no messages are lost along the way, because one stage may be transmitting while the receiving stage may have no storage or buffer space available, a flow control policy regulates message flow between neighboring nodes. Each node has a buffer with two limits, its maximum capacity or *high water mark* and a lower threshold or *low water mark.* When a buffer is filled to its high water mark, the transmitting node is stopped, and the buffer is drained of messages until it reaches its low water mark, at which point the transmitter node can continue to send messages. From the previous discussion, it is straightforward to show that if the high water marks for all nodes are greater than or equal to two messages, then the maximum mean throughput rate is given by

$$maximum\ \ mean\ \ throughput\ \ rate\ = 1/max(E(T_{S,1}),...,E(T_{S,N}))$$

In other words, the flow control strategy under these assumptions has *no* impact on the maximum data mean throughput rate. No messages will be lost due to buffer space being unavailable, and the slowest stage will be the bottleneck. Furthermore, all of the queueing occurs at the stage with the greatest processing time, and all of the delay analysis sketched earlier carries over immediately.

6.7.4 Additional Reading

[1] Gene M. Amdahl, *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities,* AFIPS Spring Joint Computer Conference Proceedings, pp.483-485, 1967.

[2] H.D.Friedman, *Reduction Methods for Tandem Queuing Systems,* Operations Research, **13** (1), 121-131 (1965).

[3] Wolfgang Kramer, *Investigations of Systems with Queues in Series,* Institute of Switching and Data Technics, University of Stuttgart, Report #22(1975).

6.8 One Processor, One Paging Drum, One Moving Head Disk

The hardware configuration of a computer system consists of one processor, one moving head disk, and one paging drum. From the earlier mean value analysis, we know that the best concurrency we might achieve is to keep all three devices simultaneously busy. A job will undergo some processing, some input/output, some processing, and so forth, and then leave, but at the instant it leaves another job will enter to take its place, fixing the total number of jobs in the system at M at all times. Each job accesses the paging drum a mean of seven times, the moving head disk a mean of twice, and has a mean of ten processor visits. The total mean time spent using each resource without contention is:

$$T_{processor} = 280\ msec \qquad T_{drum} = 280\ msec \qquad T_{disk} = 560\ msec$$

As we increase the total number of jobs, the disk will reach complete utilization first, i.e., the disk is the system bottleneck (why?). It also happens to be the slowest device, but it is really the *total* load per node that determines the bottleneck, not the speed or number of visits individually.

The mean throughput rate for M=1 is

$$\lambda(M=1) = \frac{1}{1120\ msec} = 0.88\ jobs/sec$$

The table below summarizes the results of calculating the processor utilization versus M:

Table 6.8.Degree of Multiprogramming vs Utilization

M	*1*	*2*	*3*	*4*	∞
$U_{processor}$	1/4	4/11	11/23	26/57	1/2

Most of the gain in maximizing mean throughput rate is achieved for increasing M from one to three, and the increase beyond two buys little:

$$mean\ \ throughput\ \ rate\ = \frac{U_{processor}}{T_{processor}}$$

Since we have three devices, the best we could ever do is to keep all three simultaneously busy. For the numbers here, with the disk time per job twice as great as the processor or drum time per job, the disk will reach completely utilization first. If the disk is completely busy, the processor and the drum will each be utilized fifty per cent of the time. Since the mean number of jobs in execution equals the fraction of time each device is busy, the total mean number of jobs in execution with $M \rightarrow \infty$ is *two,* one on the disk, and one half each for the drum and processor. The Jackson network analysis refines this mean value asymptotic analysis, quantifying the mean throughput rate more precisely: 0.3636 jobs per second (M=2) versus 0.4783 jobs per second (M=3), a net gain of 23.9 per cent.

EXERCISE. Graph the mean throughput rate versus M. Plot upper and lower bounds on mean throughput rate versus M that were derived earlier, and compare the results.

6.8.1 Additional Reading

[1] H.Kobiyashi, **Modeling and Analysis: An Introduction to System Performance Evaluation Methodology,** Addison Wesley, Reading, Mass, 1978.

6.9 Prototype Directory Assistance System Case Study

A prototype of an online directory assistance system was built to handle telephone number directory assistance queries. In a typical cycle of operation, a person at a terminal would be involved in the following steps

[1] Receive a query from a customer via voice telephone

[2] Enter the given information into a computer terminal while talking to the customer

[3] Wait for the system to respond with the answer to the query

[4] Tell the customer the reply over the voice telephone

[5] Close out customer interaction

[6] Wait to receive the next customer query

The hardware configuration for the system consisted of C terminals, a single processor, a single disk controller, and a single disk spindle. An operating system coordinated scheduling and management of these devices, while a set of prototype application programs handled query processing.

Measurements on the prototype system in operation showed that

- The mean time spent by a person talking, reading, and thinking, denoted by T_C, was twenty seconds
- The mean processor time per query was broken down into three sets of application programs
  - The operator interface front end programs consumed 180 milliseconds of processor time per query on the average
  - The index manipulation application programs consumed 420 milliseconds of processor time per query on the average
  - The data retrieval application programs consumed 330 milliseconds of processor time per query on the average
  - Miscellaneous application programs that were invoked for accounting, system administration, and other purposes consumed one hundred and forty milliseconds (140 msec) per query

  Hence, the total mean processor time per query, T_P, was 1.07 seconds
- The mean number of disk accesses per query was twenty six (26), with the disk capable of making one access every twenty five milliseconds (25 msec) which results in a mean time the disk is busy per query, denoted T_D, of six hundred fifty milliseconds (650 msec)

The above measurements on total mean processor time and disk access counts were based on examining the mean resources required for one hundred different queries to the system; the measurement error on the processor time was felt to be under ten milliseconds, while the measurement error on the number of disk accesses was felt to be under one access.

6.9.1 State Space The system state space Ω is given by

$$\Omega = \{\underline{J} = (J_C, J_P, J_D) \mid J_C + J_P + J_D = C\}$$

J_C clerks are active entering a query, J_P jobs are queued or running on the processor, and J_D jobs are queued or running on the disk.

6.9.2 Mean Value Analysis The upper and lower mean value bounds on mean queueing time or response time are given by

$$\max\left[T_P + T_D, \frac{C}{max[T_P, T_D]} - T_{think}\right] \leq E(T_Q) \leq C(T_P + T_D)$$

while the associated upper and lower mean value bounds on mean throughput rate are given by

$$\frac{C}{T_C + C(T_P + T_D)} \leq \lambda \leq \min\left[\frac{C}{T_C + T_P + T_D}, \frac{1}{max[T_P, T_D]}\right]$$

6.9.3 Jackson Network Analysis Provided a Jackson network model adequately fits the data, the distribution of number of jobs at each node is given by

$$\pi(\underline{J}) = \pi(J_C, J_P, J_D) = \frac{1}{G} \frac{{T_C}^{J_C}}{J_C!} {T_P}^{J_P} {T_D}^{J_D}$$

$$G = \sum_{\underline{K} \in \Omega} \frac{{T_C}^{K_C}}{K_C!} {T_P}^{K_P} {T_D}^{K_D}$$

The mean number of jobs inside the computer system is

$$E[J_P + J_D] = \sum_{\underline{K} \in \Omega} [K_P + K_D] \pi(\underline{K})$$

The mean throughput rate is simply the fraction of time the processor is busy multiplied by the maximum rate, in jobs per unit time, that the processor can execute jobs:

$$mean\ throughput\ rate = \sum_{\underline{K} \in \Omega} [1 - \pi(K_C, K_P = 0, K_D)] / T_P = \frac{E[min(1, J_P)]}{T_P}$$

The mean queueing time per job is the mean number of jobs in the system divided by the mean throughput rate:

$$mean\ queueing\ time\ per\ query = \frac{E[J_P + J_D]}{mean\ throughput\ rate}$$

For infinite source arrival statistics, this simplifies:

$$mean\ throughput\ rate = \lambda = \frac{C}{T_C}$$

$$mean\ queueing\ time\ per\ query = \frac{E(T_P)}{1 - \lambda E(T_P)} + \frac{E(T_D)}{1 - \lambda E(T_D)}$$

These bounds are plotted in the figures below, along with observed data gathered over an eight hour time interval with twelve $C=12$ operators and calculations based upon a closed queueing network model with $M=C$ obeying product form type solution. The goodness of fit of the closed queueing network model to actual data was felt to be acceptable for the purposes at hand; the mean value lower bound on mean delay and upper bound on mean throughput rate were also felt to give an indication of performance limitations at an early stage of development, which the data gathering and refinement via a

Figure 6.13.Mean Throughput Rate vs Mean Processor Time/Query

Figure 6.14.Mean Response Time vs Mean Processor Time/Query

closed queueing network model only strengthened further. Note that the system is achieving a great deal of concurrency, because the actual mean throughput rate is much closer to the upper bound, not the single thread lower bound, as is the mean delay.

6.10 Multiple Class Jackson Network Model

C types of jobs arrive to a system for execution. Each job stream is submitted by P_0 sources. $F_I, I=1,...,C$ is the fraction of job submissions of type I. Each job takes a route through the network, specified by $R_I(M), M=1,...,S(I)$ where M denotes the step for job type I. The job step execution time is denoted by τ_{IK} time units of service at node $K=1,...,N$. The total amount of service required at node K by a type I job is given by aggregating the total service time for each step, denoted by T_{IK}. Each node consists of P_K processors.

6.10.1 State Space The state of the system at any time instant t is denoted by $\underline{J}$ where

$$\underline{J} = (J_{11},...,J_{C1},...,J_{1N},...,J_{CN})$$

with $J_{IK}(t), 1 \le I \le C, 1 \le K \le N$ denoting the number of jobs either waiting or in execution at node K of type I. We denote by J_{K*} the total number of jobs at node K and by J_{*I} the total number of jobs in the system of type I:

$$J_{K*} = \sum_{I=1}^{C} J_{IK} \quad J_{*I} = \sum_{K=1}^{N} J_{IK}$$

Finally, the total number of jobs in the system at time t, denoted by $|\underline{J}(t)|$, is given by:

$$|\underline{J}(t)| = \sum_{K=1}^{N} \sum_{I=1}^{C} J_{IK}(t)$$

6.10.2 Scheduling Policy The instantaneous total throughput rate at node K is given by

$$r_K = \frac{\Phi_K(J_{K*})}{\sum_{I=1}^{C} F_I T_{IK}}$$

If the queue discipline is balanced with an arbitrary service time distribution at each node, then the fraction of time the system is in state $\underline{J}$ is given by a product of three terms, a partition function term, a source term, and a term that is the product of one term for each node:

$$\pi(\underline{J}) = \frac{1}{G} (\textit{source term}) \times \prod_{K=1}^{N} (\textit{node } K \textit{ term})$$

$$\textit{source term} = \begin{cases} \dfrac{T_0^{J_0}}{J_0!} & J_0 = P_0 - |\underline{J}| \\ 1 & |\underline{J}| = M \\ \lambda^{|\underline{J}|} & \dfrac{P_0}{T_0} = \lambda, P_0 \to \infty \end{cases}$$

$$\textit{node } K \textit{ term} = J_{K*}! \prod_{l=1}^{J_{K*}} \frac{1}{\Phi_K(l)} \prod_{I=1}^{C} \frac{(F_I T_{IK})^{J_{IK}}}{J_{IK}!} \quad K = 1, \ldots, N$$

6.10.3 Infinite Source Asymptotics For the infinite source case, this can be simplified:

$$\pi(\underline{J}) = \frac{1}{G} \prod_{I=1}^{C} \prod_{K=1}^{N} J_{K*}! \frac{(\lambda F_I T_{IK})^{J_{IK}}}{J_{IK}!}$$

$$G = \begin{cases} \displaystyle\prod_{K=1}^{N} \frac{1}{1 - \lambda \sum_{I=1}^{C} F_{IK} T_{IK}} & \lambda < \min_K \left[\dfrac{P_K}{\sum_{I=1}^{C} F_I T_{IK}} \right] \\ \infty & \lambda \ge \min_K \left[\dfrac{P_K}{\sum_{I=1}^{C} F_I T_{IK}} \right] \end{cases}$$

6.10.4 Delay Analysis For either the finite source or infinite source model, the mean queueing delay is simply the sum of the mean queueing delay at each node:

$$E[T_{I,Q}] = \sum_{K=1}^{N} E[T_{IK,Q}] \quad I = 1, .., C$$

The mean queueing delay at each node is the mean queueing delay for all jobs at that node, apportioned according to the *utilization* of the node by that type of job:

$$E[T_{IK,Q}] = \frac{F_I T_{IK}}{\sum_{I=1}^{C} F_I T_{IK}} E[T_{K,Q}]$$

where

$$E[T_{K,Q}] = \frac{E[J_K]}{\lambda} = \left[\sum_{I=1}^{C} F_I T_{IK}\right] \left[D(P,U) \equiv 1 + \frac{C(P,PU)}{P(1-U)}\right]$$

$C(P,P\ U)$ is the Erlang delay function discussed earlier, and $D(P,U)$ is the stretching factor that multiplies the total mean job execution time at node K. We stress that this can be readily approximated numerically using the Erlang delay numerical approximations discussed earlier. Here the multiple class analysis involves analyzing a *single* class system, with the *total* load on each node apportioned among the different jobs, and then calculating the *multiple* class delays by apportioning the delay at each node according to loading.

6.10.5 An Example As an example, suppose a single processor node N=1 executes two types of jobs C=2. Jobs arrive according to Poisson statistics, with total job arrival rate λ; F_K denotes the fraction of arrivals of type K=1,2. The state space is given by

$$\Omega = \{(J_{11},J_{21}) \mid J_{I1} \geq 0, I=1,2\}$$

The fraction of time the system is in state $\underline{J}=(J_{11},J_{21})$ is given by

$$\pi(J_{11},J_{21}) = \frac{1}{G}\lambda^{J_{1*}} J_{1*}! \frac{(F_1 T_{11})^{J_{11}}}{J_{11}!} \frac{(F_2 T_{21})^{J_{21}}}{J_{21}!}$$

The partition function is given by

$$G = \frac{1}{1 - \lambda(F_1 T_{11} + F_2 T_{21})}$$

In order to calculate the mean throughput rate, we calculate the mean number of total jobs at the node, $E(J_{1*})$, and then apportion the mean number of each type job according to the workload:

$$E(J_{I1}) = \frac{F_I T_{I1}}{F_1 T_{11} + F_2 T_{21}} E[J_{1*}]$$

If there is only one processor, the mean number of jobs at the node is given by

$$E[J_{1*}] = \frac{\lambda(F_1 T_{11} + F_2 T_{21})}{1 - \lambda(F_1 T_{11} + F_2 T_{21})}$$

The mean throughput rate is λ which equals the total mean number of jobs divided by the total mean execution time per job:

$$\lambda = \frac{E[min(J_{1*},P_1)]}{F_1 T_{11} + F_2 T_{21}}$$

The total mean delay (waiting plus execution) is given by

$$E[T_Q] = \frac{E[J_{1*}]}{\lambda} = D(P,U)(F_1 T_{11} + F_2 T_{21})$$

which for one processor is simply

$$E[T_Q] = \frac{F_1 T_{11} + F_2 T_{21}}{1 - \lambda(F_1 T_{11} + F_2 T_{21})}$$

The mean throughput rate for each type of job is $\lambda_I = \lambda F_I$. Finally, the total mean delay for each type of job is apportioned according to the load:

$$E[T_{I,Q}] = \frac{F_I T_{I1}}{F_1 T_{11} + F_2 T_{21}} E[T_Q]$$

6.11 Bounds on Mean Throughput Rate in Product Form Network Distributions

Our goal is to obtain upper and lower *bounds* on mean throughput rate *without* explicitly evaluating the mean throughput rate based on the product form formula. These bounds will in general be *tighter* than the mean value bounds obtained earlier, but unfortunately at present are only known for the case where there is *one* processor or serially reusable resource at each step of execution, unlike the mean value bounds which hold for *multiple* processors (and resources) held at each execution step.

6.11.1 Model The mathematical model dealt with here consists of

- One type of job that migrates amongst N stations or stages
- A *single* processor available to execute a job at stage $K=1,...,N$
- M tasks or jobs circulate among the nodes
- T_K denotes the total mean amount of service required by a job summed over all its visits to stage $K=1,...,N$

The system state is denoted by Ω:

$$\Omega = \{(J_1,...,J_N) \mid \sum_{K=1}^{N} J_K = M\}$$

At any given instant of time, the system is in state $\underline{J} = (J_1,...,J_N)$ where $J_K, K=1,...,N$ denotes the number of jobs at node K (both waiting and in execution). The long term time averaged distribution of number of jobs at each node at an arbitrary instant of time can be adequately modeled by *product form* or separation of variables formula

$$PROB[J_1=K_1, \ldots, J_N=K_N] = \frac{1}{G_M} \prod_{I=1}^{N} T_I^{K_I} \qquad (K_1,...,K_N) \in \Omega$$

$$G_M = G_M(T_1,...,T_N) = \sum_{\underline{J} \in \Omega} \prod_{I=1}^{N} T_I^{J_I}$$

G_M is the *system partition function* chosen to normalize the probability distribution. Granted these assumptions, we observe that the mean throughput rate of jobs making a complete cycle of the system is given by

$$\lambda = \frac{PROB[J_K>0]}{T_K} = \frac{G_{M-1}(T_1,...,T_N)}{G_M(T_1,...,T_N)}$$

Straightforward manipulations of this expression yield the desired bounds:

$$\frac{M}{\sum_{K=1}^{N} T_K + (M-1)T_{\max}} \le \lambda \le \min\left[\frac{1}{T_{\max}}, \frac{M}{\sum_{K=1}^{N} T_K + (M-1)T_{average}}\right]$$

$$\frac{M}{T_{cycle} + (M-1)T_{\max}} \le \lambda \le \min\left[\frac{1}{T_{\max}}, \frac{M}{T_{cycle} + (M-1)T_{average}}\right]$$

where T_{cycle} is the total mean execution time for a job to cycle through all nodes

$$T_{cycle} = \sum_{K=1}^{N} T_K$$

and $T_{average}$ is the average execution time per node

$$T_{average} = \frac{1}{N} \sum_{K=1}^{N} T_K = \frac{T_{cycle}}{N}$$

and $T_{\max}$ is the largest mean execution time per node:

$$T_{\max} = \max_{K=1,\ldots,N} T_K$$

If the *average* time per node spent in execution by one job during a cycle and the *maximum* time per node per job are roughly comparable to one another, these bounds will be quite close to one another.

6.11.2 Example The hardware configuration for a computer system consists of a single processor and a single disk. The computer system is assumed to have M jobs resident in main memory at any one time, the so called *degree of multiprogramming.* How should M be chosen to maximize mean throughput rate?

This can be modeled by a two stage queueing network, $N=2$, with the total mean processor time per job denoted by T_{proc} while the total mean number of disk accesses per job multiplied by the mean time per disk access gives the total mean disk time per job, denoted by T_{disk}. The table below summarizes mean value upper and lower bounds on mean throughput rate for this system as a function of M, the degree of multiprogramming, i.e., the number of jobs in the system.

$$\frac{M}{M(T_{proc} + T_{disk})} = \frac{1}{T_{proc} + T_{disk}} \le \lambda \le \min\left[\frac{1}{max[T_{proc}, T_{disk}]}, \frac{M}{T_{proc} + T_{disk}}\right]$$

Table 6.9.Mean Value Bounds on λ with $T_{disk}=1.0$

M	$T_{proc}=0.3$		$T_{proc}=1.0$	
(Jobs)	λ_{lower}	λ_{upper}	λ_{lower}	λ_{upper}
1	0.77 jobs/sec	0.77 jobs/sec	0.50 jobs/sec	0.50 jobs/sec
2	0.77 jobs/sec	1.00 jobs/sec	0.50 jobs/sec	1.00 jobs/sec
3	0.77 jobs/sec	1.00 jobs/sec	0.50 jobs/sec	1.00 jobs/sec

All of benefit occurs in increasing the degree of multiprogramming from one to two, because only two resources can be kept busy at any time; going beyond two buys nothing at this level of analysis.

For comparison, Table 6.10 summarizes queueing network upper and lower bounds for the same system:

$$\frac{M}{T_{cycle} + (M-1)T_{\max}} \le \lambda \le \min\left[\frac{1}{T_{\max}}, \frac{M}{T_{cycle} + (M-1)T_{average}}\right]$$

$$T_{cycle} = T_{proc} + T_{disk} \quad T_{average} = \frac{T_{cycle}}{2} \quad T_{\max} = max[T_{proc}, T_{disk}]$$

Table 6.10.Queueing Network Bounds on λ with $T_{disk}=1.0$

M	$T_{proc}=0.3$		$T_{proc}=1.0$	
(Jobs)	λ_{lower}	λ_{upper}	λ_{lower}	λ_{upper}
1	0.77 jobs/sec	0.77 jobs/sec	0.50 jobs/sec	0.50 jobs/sec
2	0.87 jobs/sec	1.00 jobs/sec	0.67 jobs/sec	0.67 jobs/sec
3	0.91 jobs/sec	1.00 jobs/sec	0.75 jobs/sec	0.75 jobs/sec

The numbers show that the queueing network bounds are considerably tighter than the mean value bounds, for the numbers chosen here. Furthermore, for the case of equal job time balance, $T_{proc}=T_{disk}$, the queueing network upper and lower bounds on mean throughput rate are *identical* and hence equal the *exact* mean throughput rate, something that might not be obvious a priori! Finally, most of the benefit occurs in going from $M=1$ to $M=2$ for the imbalanced case, $T_{proc}=0.3, T_{disk}=1.0$, while the gain for the balanced case in increasing M are much more gradual, provided the Jackson network assumptions are valid. This is a refinement beyond the information provided in the mean value bounds.

6.11.3 Additional Reading

[1] P.J.Denning, J.P.Buzen, *The Operational Analysis of Queueing Network Models,* Computing Surveys, **10** (3), 225-261 (1978).

[2] C.Sauer, K.Chandy, **Computer Systems Performance Modeling,** Prentice-Hall, Englewood Cliffs, NJ, 1981.

[3] J.Zahorjan K.C.Sevick, D.L.Eager, B.Galler, *Balanced Job Bound Analysis of Queueing Networks*, Communications of the ACM, **25** (2), 134-141 (1982).

Problems

1) An online transaction processing system consists of one processor and C clerks at terminals. The mean time a clerk spends reading, thinking and entering the transaction is denoted by T_{think}. The mean time required for the processor to execute the transaction is denoted by T_{proc}.

A. What is the state space for this system?

B. What are the resources required for each step of execution?

C. What are the bottlenecks?

D. Plot bounds on mean throughput rate and mean delay to execute a transaction versus number of clerks using mean value analysis.

E. Plot the mean throughput rate and mean delay versus number of clerks using a closed Jackson queueing network, and compare with the mean value analysis.

F. Suppose the total mean arrival rate of jobs is fixed at λ such that

$$\lambda = constant = \frac{C}{T_{think}} \quad C \to \infty, T_{think} \to \infty$$

which is the infinite source model for this system. Plot the mean throughput rate and mean delay versus arrival rate, and compare with the closed Jackson network analysis and mean value bounds.

2) An online transaction processing system requires dedicated links from front end terminal controllers to back end data base management systems. Each transaction holds a circuit for a mean time interval of one tenth of a second. The arrival statistics are assumed to be simple Poisson. During a normal business hour the system must process ten transactions per second with the fraction of transactions blocked due to all links being busy of less than one transaction in ten thousand. During a peak business hour the system must process thirty transactions per second with the fraction of transactions blocked due to all links being busy of less than one transaction in one hundred.

[1] If all circuits are busy, a transaction will be queued until a circuit becomes available.

A. How many trunks are needed to meet design goals?

B. Repeat the above if the mean transaction holding time is one fifth of a second.

C. Repeat the above if the mean transaction holding time is one twentieth of a second.

[2] If all circuits are busy, a transaction is rejected or blocked, and presumably will retry later.

A. How many trunks are needed to meet design goals?

B. Repeat the above if the mean transaction holding time is one fifth of a second.

C. Repeat the above if the mean transaction holding time is one twentieth of a second.

[3] Repeat both A) and B) for the following two new design goals

A. During a normal business hour, no more than one transaction in one hundred is delayed due to trunks not being available. During a peak business hour, no more than one transaction in five is delayed due to trunks not being available.

B. During a normal business hour, no more than one transaction in one million is delayed due to trunks not being available. During a peak business hour, no more than one transaction in ten thousand is delayed due to trunks not being available.

3) Insurance agents dial up through the voice telephone network a central computer, and then use a terminal connected to a modem to query the central computer about different types of insurance policies for potential customers. A five year plan has been proposed. During the first two years of operations, the system should handle five queries per second during a normal business hour, and ten queries per second during a peak business hour. During the next three years, the system should handle twenty queries per second during a normal business hour, and thirty queries per second during a peak business hour. A query session has a mean duration of five minutes. If all ports are busy when a query attempt occurs, a busy tone is generated, and the query is delayed.

[1] Assume all blocked queries are lost or rejected and will retry later:

A. How many modems are needed if the fraction of queries blocked due to no modem being available is no more than one query in ten?

B. Repeat if the criterion is no more than one query in five is blocked?

C. Repeat if the criterion is no more than one query in one hundred is blocked?

D. Repeat all the above if the mean holding time per query is increased to ten minutes.

[2] Assume all blocked queries are queued or delayed until a port becomes available. Repeat all the above.

4) Jobs arrive for service at a parallel group of P processors. The arrival statistics are Poisson with mean arrival rate λ. The holding time or service time per job forms a sequence of independent identically distributed exponential random variables with mean service time $E(T_S)$. The *offered load* is the mean number of busy servers provided we had an *infinite* number of servers, and this is measured in Erlangs, and denoted by A*:*

$$A = \lambda E(T_S) = offered\ \ load\,(Erlangs)$$

The fraction of time P or more servers are busy with an *infinite* number of available servers is denoted by $F(P,A)$ and is given by

$$F(P,A) = fraction\ \ of\ \ time\ \ P\ \ or\ \ more\ \ busy\ \ servers = e^{-A}\sum_{K=P}^{\infty}\frac{A^K}{K!}$$

A. Show the following relationships between Erlang's blocking function,

$$B(P,A) = \frac{A^P/P!}{\sum_{K=0}^{P}\frac{A^K}{K!}} \quad A = offered\ \ load\,(Erlangs) \quad P = number\ \ of\ \ servers$$

which is also called Erlang's function of the first kind and Erlang's delay function,

$$C(P,A) = \frac{A^P/(P-1)!(P-A)}{\sum_{K=0}^{P-1}\frac{A^K}{K!} + A^P/(P-1)!(P-A)}$$

which is also called Erlang's function of the second kind:

$$B(P,A) < F(P,A) < C(P,A)$$

Provide a physical interpretation for why this result is reasonable: draw some pictures!

B. Show the following:

$$C(P,A) = \frac{(P/A)B(P,A)}{(P/A)-1+B(P,A)}$$

C. Show the following:

$$C(P,A) = \frac{1}{\frac{P-A}{A\ B(P-1,A)} + 1}$$

D. Show the following:

$$B(P,A) = \frac{AB(P-1,A)}{P + A\ B(P-1,A)}$$

E. Show the following:

$$C(P,A) = \frac{1}{1 + \frac{P-A}{A}\ \frac{P-1-A\ C(P-1,A)}{(P-1-A)C(P-1,A)}}$$

5) The hardware configuration for an online transaction processing system consists of one processor and one disk. Each transaction requires a total mean amount of processor time denoted by T_P and a total mean number of disk accesses N each of which requires τ_D seconds, so the total disk access time per job is denoted by T_D. The system state is given by $\underline{J} = (J_P, J_D)$ where J_P denotes the number of jobs waiting or in execution at the processor queue and J_D denotes the number of jobs waiting or in execution at the disk queue. An infinite source open queueing network is felt to adequately model the distribution of the number of jobs at each queue; if λ is the total mean arrival rate, in transactions per unit time, and $\pi(\underline{J})$ denotes the fraction of time the system is in state $\underline{J}$, then

$$\pi(\underline{J}=(J_P,J_D)) = \frac{1}{G}\ [\lambda T_P]^{J_P}\ [\lambda T_D]^{J_D}$$

A. Calculate the system partition function G

$$G = \sum_{I=0}^{\infty}(\lambda T_P)^I \sum_{J=0}^{\infty}(\lambda T_D)^J$$

B. Calculate the total mean number of jobs in system from first principles:

$$E[J_P + J_D] = \sum_{J_P=0}^{\infty}\sum_{J_D=0}^{\infty}[J_P + J_D]\pi(J_P,J_D)$$

C. Show that the mean delay (waiting plus execution time) per transaction equals

$$E[T_{delay}] = \frac{E[J_P + J_D]}{\lambda} = \frac{T_P}{\lambda}\frac{\partial G}{\partial T_P} + \frac{T_D}{\lambda}\frac{\partial G}{\partial T_D}$$

D. For N=10,20 disk accesses, with one processor and one disk, and τ_D=50 *msec* while τ_P=300 *msec*,700 *msec* calculate the mean throughput rate and the mean delay.

6) The hardware configuration for a computer system consists of a single processor and a single disk. The system state at any time is denoted by $\underline{J}=(J_P,J_D)$ where J_P denotes the number of jobs waiting or in execution on the processor, and J_D denotes the number of jobs waiting or in execution on the disk. Each job requires some processor time and some disk access time, in order to be completely executed. The fraction of time a computer system spends in state $\underline{J}$ is denoted by $\pi(\underline{J})$. We are given some mean value or average performance metric *E[PM]* which is a functional of the system state $\underline{J}$, denoted by $PM(\underline{J})$:

$$E[PM] = \sum_{\underline{J}\in\Omega} PM[\underline{J}]\pi(\underline{J})$$

A. Show that the mean delay per transaction is an example of such a performance metric by exhibiting an explicit formula for *E[PM]*

B. Show that the processor utilization is an example of such a performance metric by exhibiting an explicit formula for *E[PM]*

C. If we wish to assess the sensitivity of the performance metric to a change in *speed* of the processor, show that

$$\frac{\partial E[PM]}{\partial(1/T_P)} = E\left[\frac{\partial PM(\underline{J})}{\partial(1/T_P)}\right] - T_P\, covar[J_P, PM(\underline{J})]$$

where $covar(I,J)$ denotes the covariance between I,J, and illustrate what this is explicitly for the mean delay and processor utilization performance metrics

D. Suppose that the time per transaction to access the disk is state dependent, such that it equals the product of two terms, the first a disk speed dependent but state independent factor, and the second a state dependent dimensionless function (which accounts for the disk scheduling algorithm):

$$T_D = T_{D,speed}\ f(J_D) \qquad f(J_D) = \begin{cases} 1 & J_D = 0 \\ (2J_D - 1/J_D) & J_D > 0 \end{cases}$$

Write an analytic expression for the fraction of time the system is in state (J_P, J_D), denoted by $\pi(J_P, J_D)$, which is a modification of the expression shown earlier.

E. We wish to assess the sensitivity of changing the *speed* of the disk while keeping the same scheduling algorithm. Show that

$$\frac{\partial E[PM(\underline{J})]}{\partial(1/T_D)} = E\left[\frac{\partial PM(\underline{J})}{\partial(1/T_D)}\right] - T_D\, covar[J_D, PM(\underline{J})]$$

Evaluate this for a total mean delay performance measure and a disk utilization performance measure

F. If T_D=1 explicitly evaluate all these formulae for T_P=1,=0.5,=0.2.

G. Suppose you can get *one* fast disk or *two* slower disks such that

$$T_{D,one\ fast\ disk} = \tfrac{1}{2} T_{D,one\ slow\ disk}$$

What is the maximum mean throughput rate of the disk for either system? Repeat all of the above (including substitution of numbers into all formulae).

7) The hardware configuration for a computer system consists of a central processor unit (CPU) and two direct memory access (DMA) disk controllers each handling a single spindle. Jobs arrive for execution to this system in such a manner that the staging queue is never empty. The staging queue limits the degree of multiprogramming to a constant level of *M* jobs always in the system. Measurements are carried out on this system, and are summarized in the table below:

Table 6.11.Measurement Data Summary

Attribute	*Symbol*	*Quantity*
Processor Time/Job	T_{proc}	2 sec
Mean Time/Disk Access	T_{access}	50 msec
Mean Number of Disk Accesses/Job:		
To Spindle I	$N_{disk,I}$	50
To Spindle II	$N_{disk,II}$	20

Answer the following questions:

A. What is the state space for the operations of this system?

B. Plot upper and lower bounds on mean throughput rate of executing jobs versus *M* and state what the bottleneck resource is as $M \to \infty$

C. Suppose that the long term time averaged fraction of time the system is in a given state can be adequately modeled by a product form distribution. What is an explicit expression for this formula?

D. Plot the mean throughput rate of executing jobs versus M assuming the product form distribution adequately models the system operation on the same plot as the mean value upper and lower bounds on mean throughput rate.

E. Plot upper and lower bounds on the mean throughput rate of executing jobs versus M assuming the a product form adequately models system operation. Do this on the same plot as the mean value upper and lower bounds on mean throughput rate.

F. Suppose that the file access load is redistributed so that the mean number of disk accesses per job stays the same but is *equal* to each spindle. Repeat all the above.

8) A data network consists of three nodes, A, B, C. At each node there are two Poisson streams of arriving messages of equal intensity intended for each of the other nodes, with each stream having mean arrival rate λ messages per unit time. Each message consists of a random number of bits with a mean of B bits per message. Each node is connected to every other node by two one way links, with one link for transmission and the other for reception. All links have capacity C bits per second.

A. What is the state space for this system?

B. What is an upper bound on the total number of messages per unit time carried by the network versus the individual message stream arrival rate λ?

C. Assuming a Jackson network model is statistically valid, what is an explicit expression for the fraction of time the system is in a given state?

D. Assuming the validity of a Jackson network model, what is an explicit expression for mean message delay as a function of model parameters?

E. If A can only transmit to C, C can only to transmit to B, and B can only transmit to A, i.e., we have a ring, with each link having capacity $2C$ bits per second, repeat all of the above analysis. Which system should be chosen under what conditions?

9) N clerks each submit a single type of job to a computer system. The hardware configuration for the computer system consists of a single processor and a single disk. Each clerk spends a variable amount of time reading, thinking, and entering the job, and then waits for the system to respond before starting the cycle over again. Each job undergoes processor execution, following by a single disk access, followed by more processor execution, followed by a single disk access, until eventually the processor completes execution of the job. The table below summarizes the quantitative data for this system:

Table 6.12.Measurement Data

Attribute	*Symbol*	*Quantity*
Mean Time Thinking	T_{think}	15 sec
Total Processor Time/Job	T_{proc}	3 sec
Probability Job Migrates CPU->Disk	R	360/361
Probability Job Migrates CPU->Clerk	$1-R$	1/361
Mean Time per Disk Access	T_{access}	50 msec

Answer the following questions:

A. What is a suitable state space to describe the operation of this system?

B. What is the mean number of disk accesses per job?

C. Plot upper and lower bounds on mean throughput and mean response time per job versus the number of active clerks.

D. What are the potential bottlenecks in this system?

E. Suppose that the long term time averaged fraction of time the system is in a given state can be adequately modeled by the *product form* probability distribution. What is an explicit expression for this probability distribution in terms of measurement data?

F. Plot the mean throughput rate and mean response time per job versus the number of active clerks assuming the product form distribution adequately models the system operation. Do this on the same plot as the mean value upper and lower bounds.

G. Suppose that the total rate at which jobs are submitted to the system is fixed at $\lambda \equiv N/T_{think}$ while the number of clerks is allowed to become infinite. What is the state space for this system? If a product form distribution adequately models the long term time averaged operational statistics, plot the mean throughput rate and mean response time per job versus the arrival rate λ.

10) Students want to use N terminals attached to a computer system that consists of a single processor. The statistics for students requesting or demanding a terminal are adequately modeled by a Poisson process with mean arrival rate λ which has the numerical value of two (2) sessions per hour. If all terminals are occupied with a student when a request occurs, the requesting student goes away. Once a student gets a terminal, the student keeps the terminal busy for a *session.* Each session consists of a series of interactions between the student at the terminal and the computer. Each interaction consists of a time interval with mean $T_{think}=4$ sec where the student is reading and thinking and typing, and a time interval for the system to respond. With only one student at one terminal using the computer, the mean time for the system to respond is denoted by $T_{system}=2$ sec. There are a mean number of $I=100$ interactions per student during a session. Once a student completes a session, the student leaves, freeing the terminal for another student.

A. What is the state space for describing the operation of this system?

B. Plot an upper bound on the mean number of sessions per unit time that the system can service versus $N=1,2,3$.

C. If the system statistics for the number of students at terminals and the number of requests in the computer can be adequately modeled by a product form distribution, write an explicit expression for this distribution as a function of N.

D. Using the above product form distribution, plot the fraction of student requests that are blocked or rejected versus $N=1,2,3$.

E. Using the above product form distribution, plot the mean number of sessions per unit time that the system services versus $N=1,2,3$.

F. Plot the mean time per session for students that are not rejected for $N=1,2,3$.

11) A computer system executes one type of job. The system hardware configuration consists of one processor, one fixed head disk (called a drum), and one moving head disk. The file system is configured with some files physically located on the drum and others on the disk. Each job requires some processor time, followed by a secondary storage access, and so on until the job is completely executed. There are N secondary storage accesses, and $N+1$ processor interactions per job. The system holds M jobs at any one time, either waiting or running on the processor, or waiting and accessing secondary storage. The secondary storage subsystem consists of a moving head disk and a fixed head disk (also called a drum). The file system is configured so that a fraction F of the secondary storage requests are made to the drum, and the remainder to the disk. The mean time per processor interaction is T_{cpu}. The mean time for a disk access is T_{disk}, while the mean time for a drum access is T_{drum}. In what follows,

use the following numbers:

Table 6.13.Numerical Parameters

T_{cpu}	10 msec
T_{drum}	20 msec
T_{disk}	30 msec
N	25 accesses
M	5 jobs

Answer the following questions:

A. What is the system state space?

B. Plot an upper bound on the mean rate of executing jobs as a function of F.

C. What choice of F maximizes the upper bound on the mean rate of executing jobs?

D. If the system statistics for the number of jobs queued or in execution at the processor, drum and disk can be adequately modeled by a product form distribution, write an explicit expression for this formula.

E. Using the above product form formula, plot the mean throughput rate of executing jobs versus F on the same plot as the upper bound.

12) A communication system administrator wishes to determine the benefits of pooling resources. A single type of message holds a link for a mean time interval of two seconds. There are two groups of links that can handle this type of message. The first group has three links and messages arrive at a rate of one message per second. The second group has four links and messages arrive at a rate of three messages every two seconds. The administrator can also pool the links into one group of seven links, with a total message arrival rate of five messages every two seconds.

A. What is the state space for each configuration?

B. Based on a mean value analysis, which configuration should be chosen?

C. If messages are blocked if no link is available, which configuration will achieve the lowest total blocking?

D. If messages are delayed if no link is available, which configuration will achieve the lowest total mean delay?

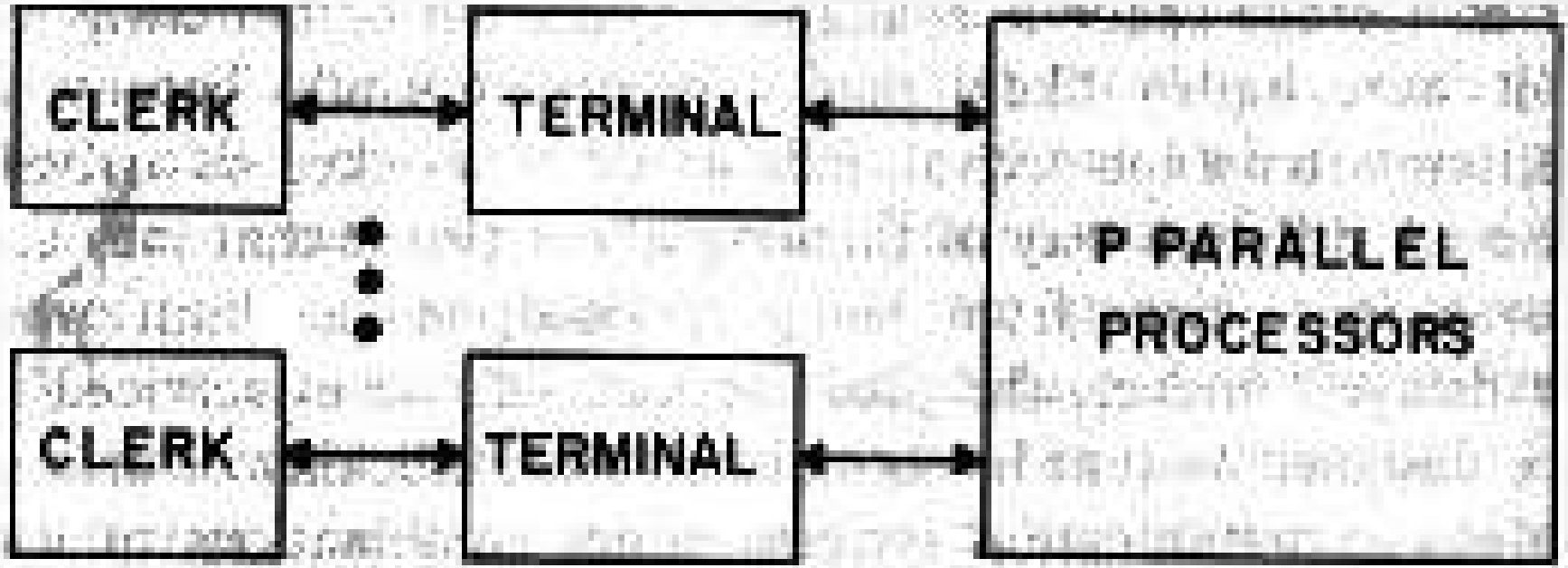

Figure 6.1.System Block Diagram

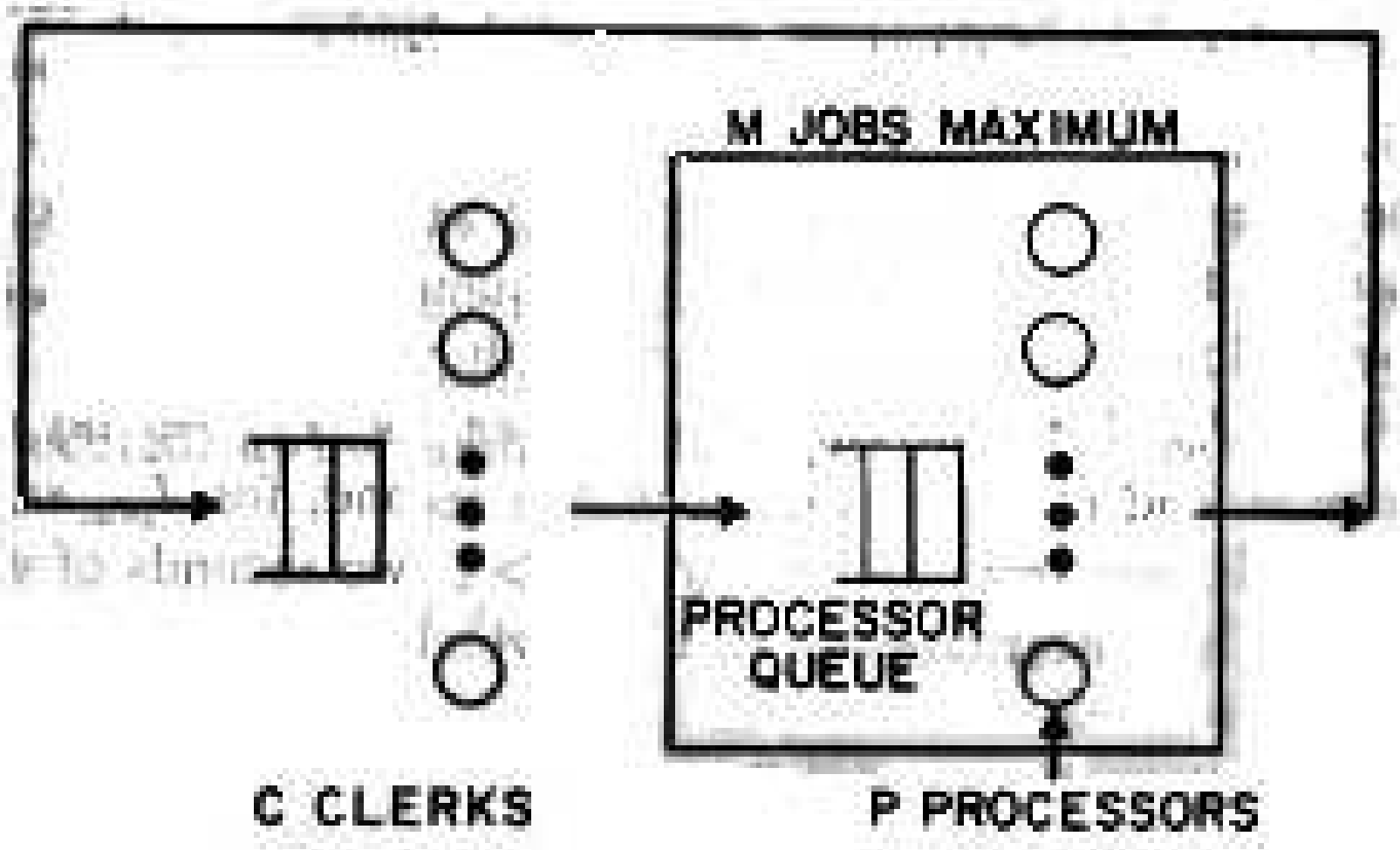

Figure 6.2.Queueing Network Block Diagram

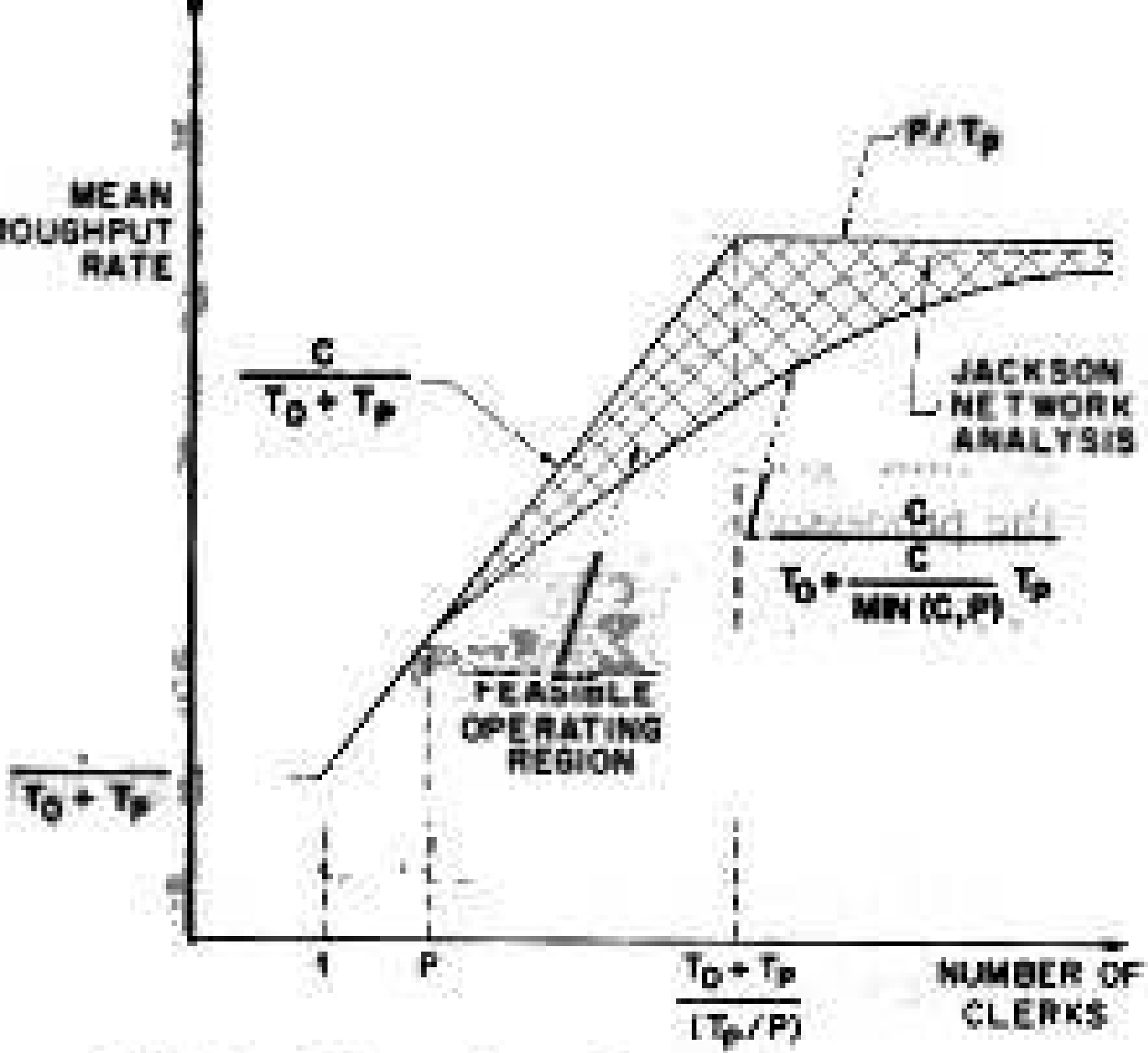

Figure 6.4.Mean Throughput Rate vs Number of Clerks

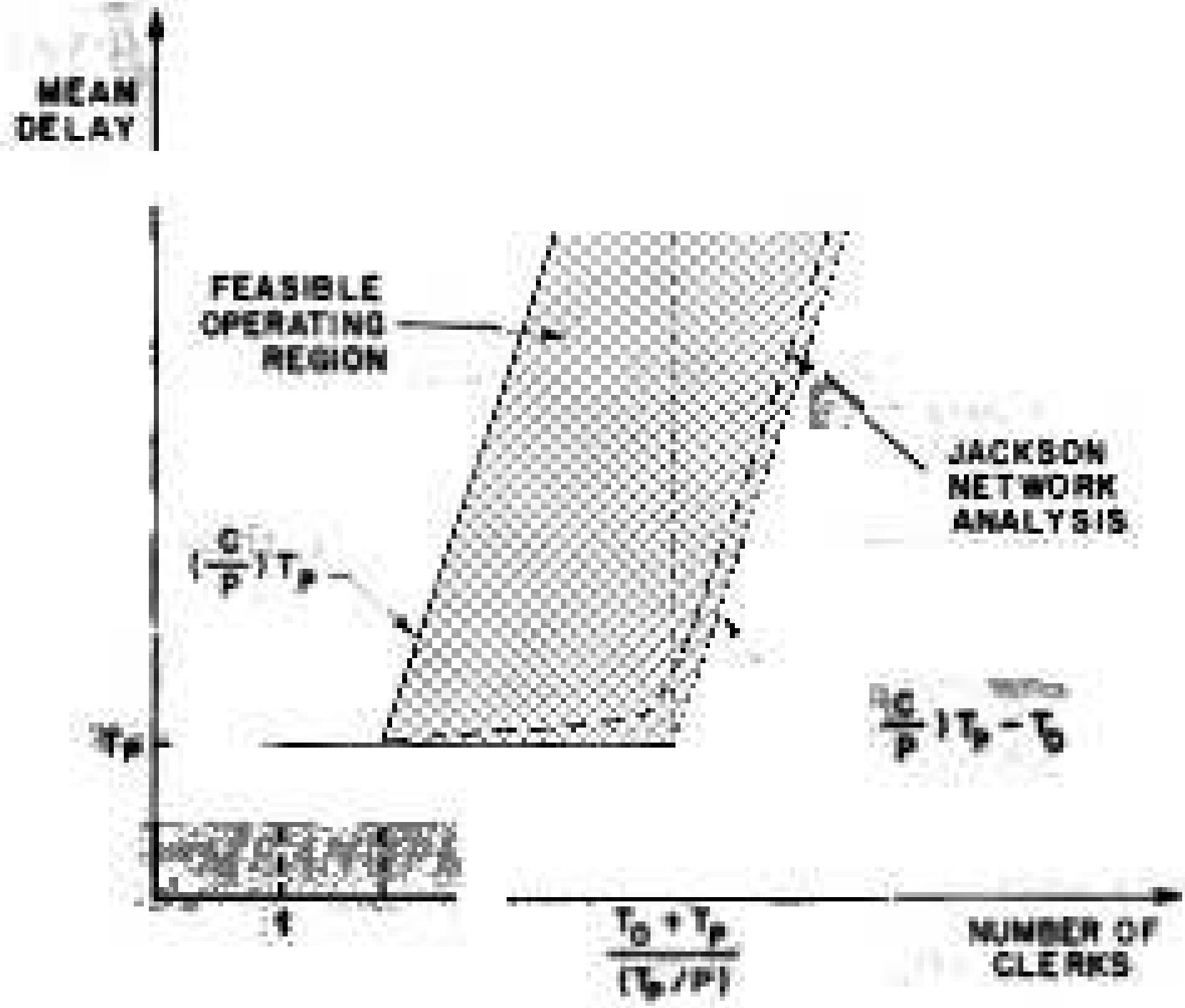

Figure 6.5.Mean Delay vs Number of Clerks

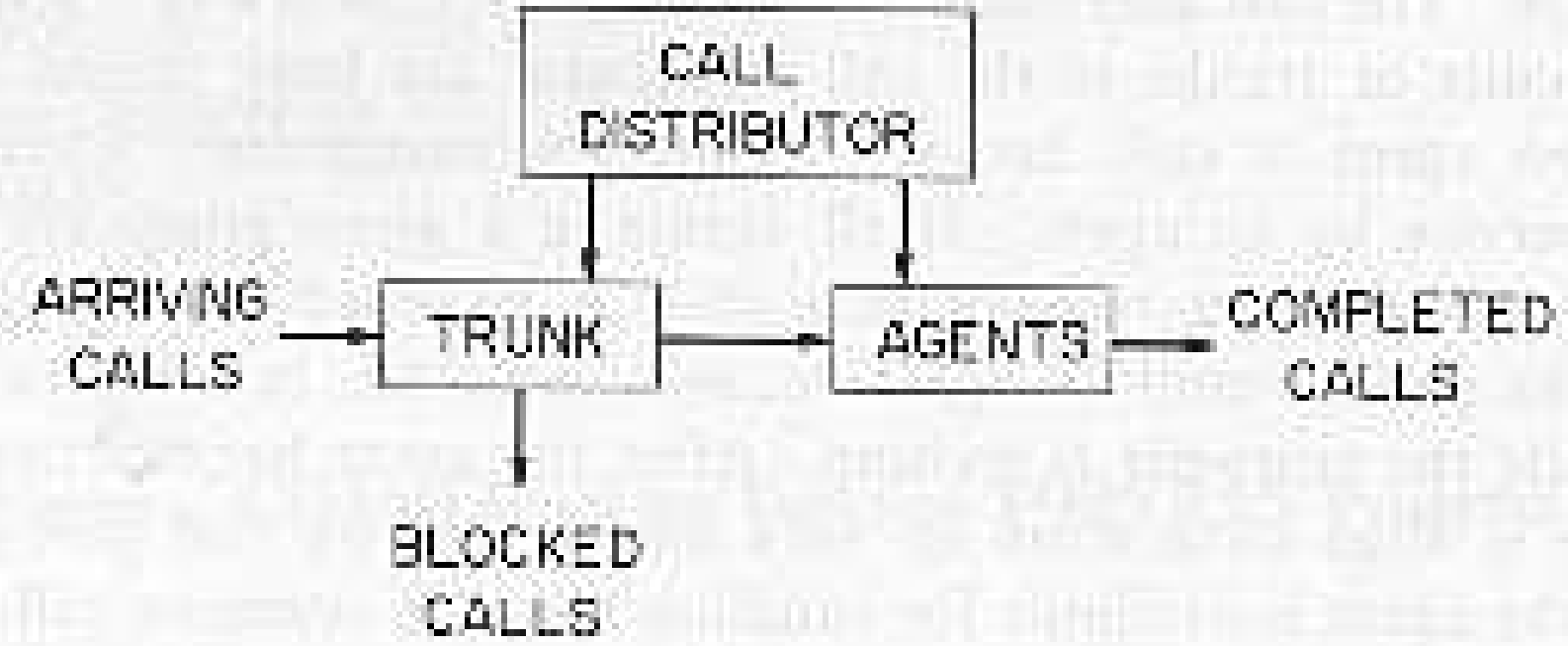

Figure 6.7.Call Distributor Block Diagram

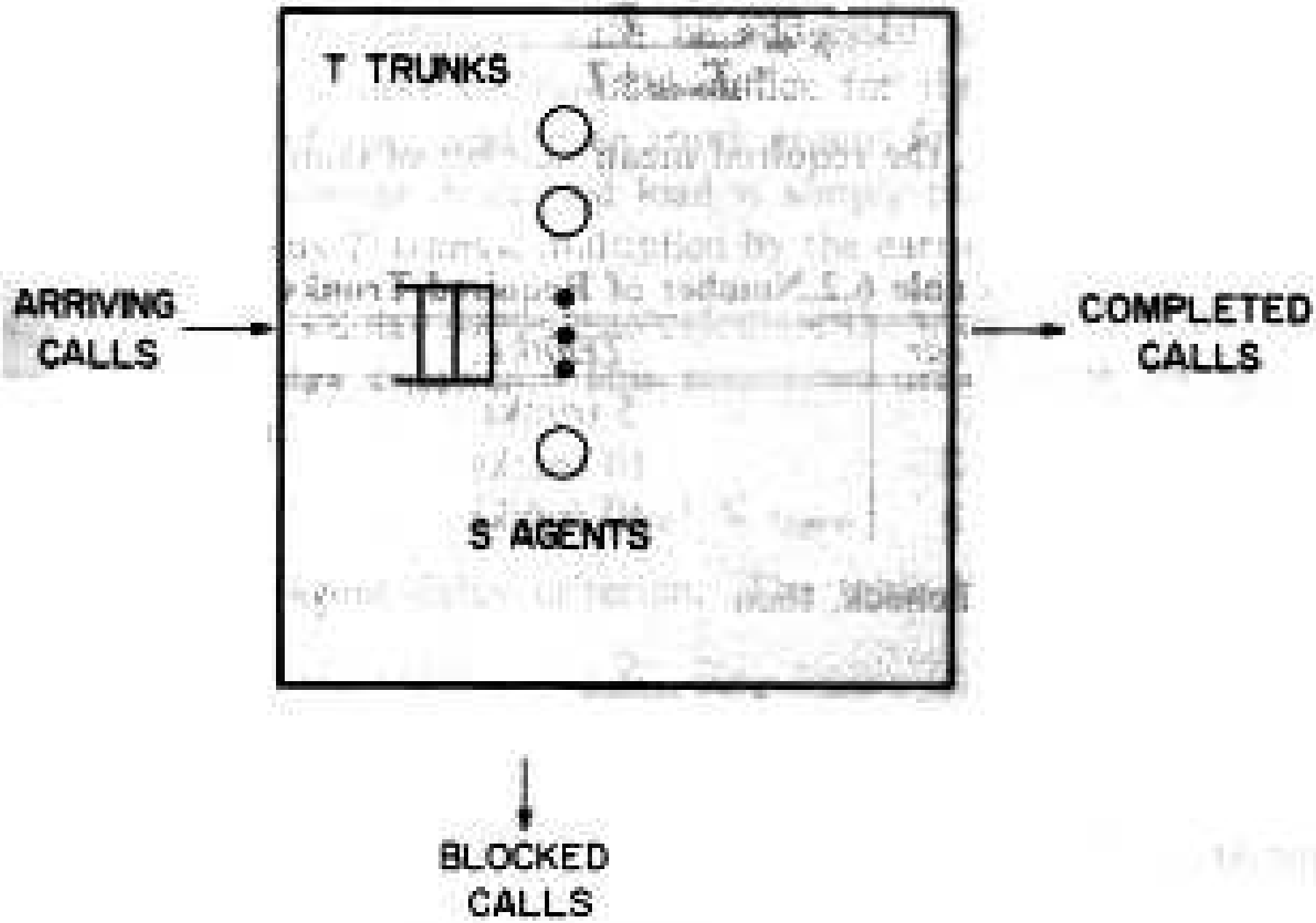

Figure 6.8.Call Distributor Queueing Network

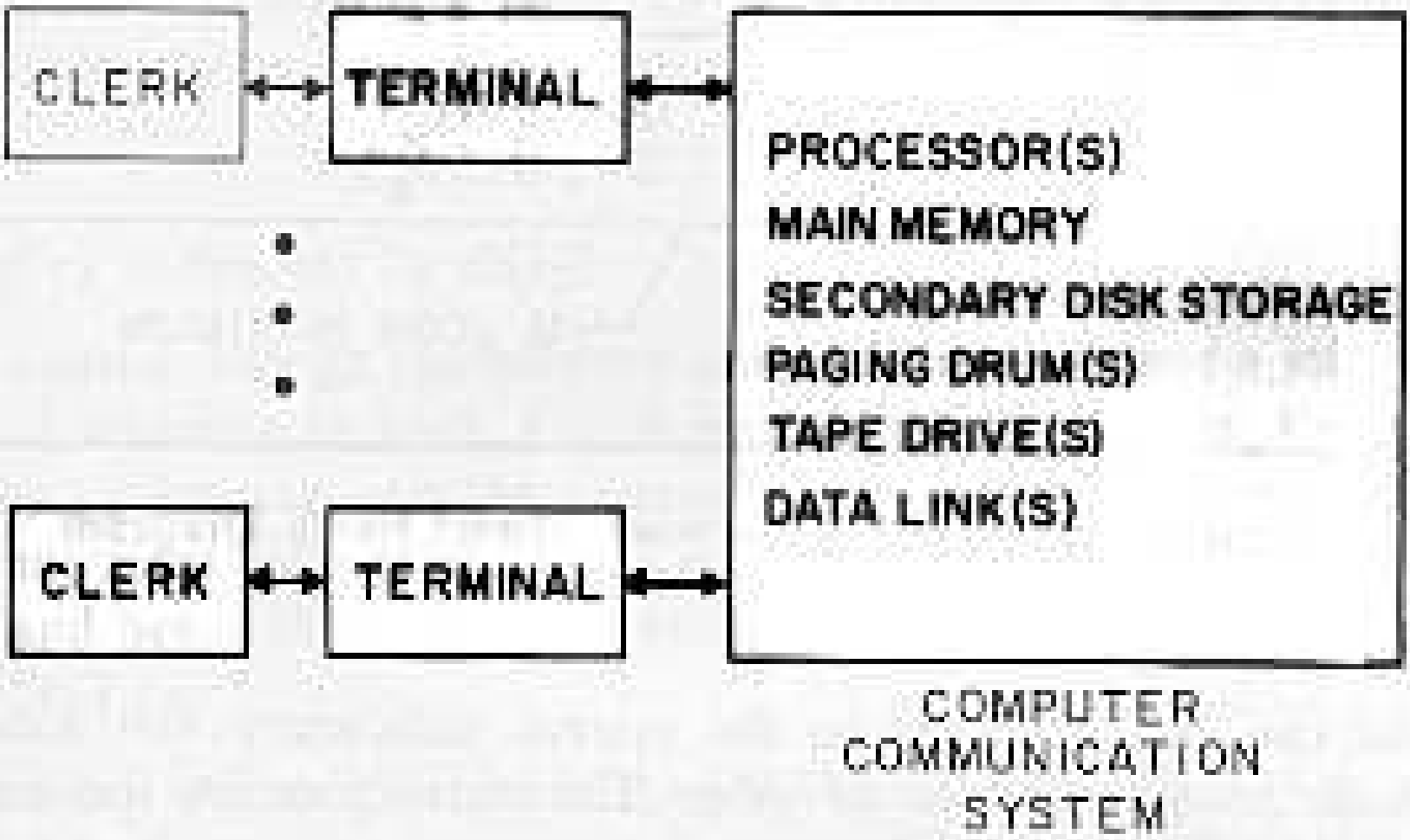

Figure 6.9.System Block Diagram

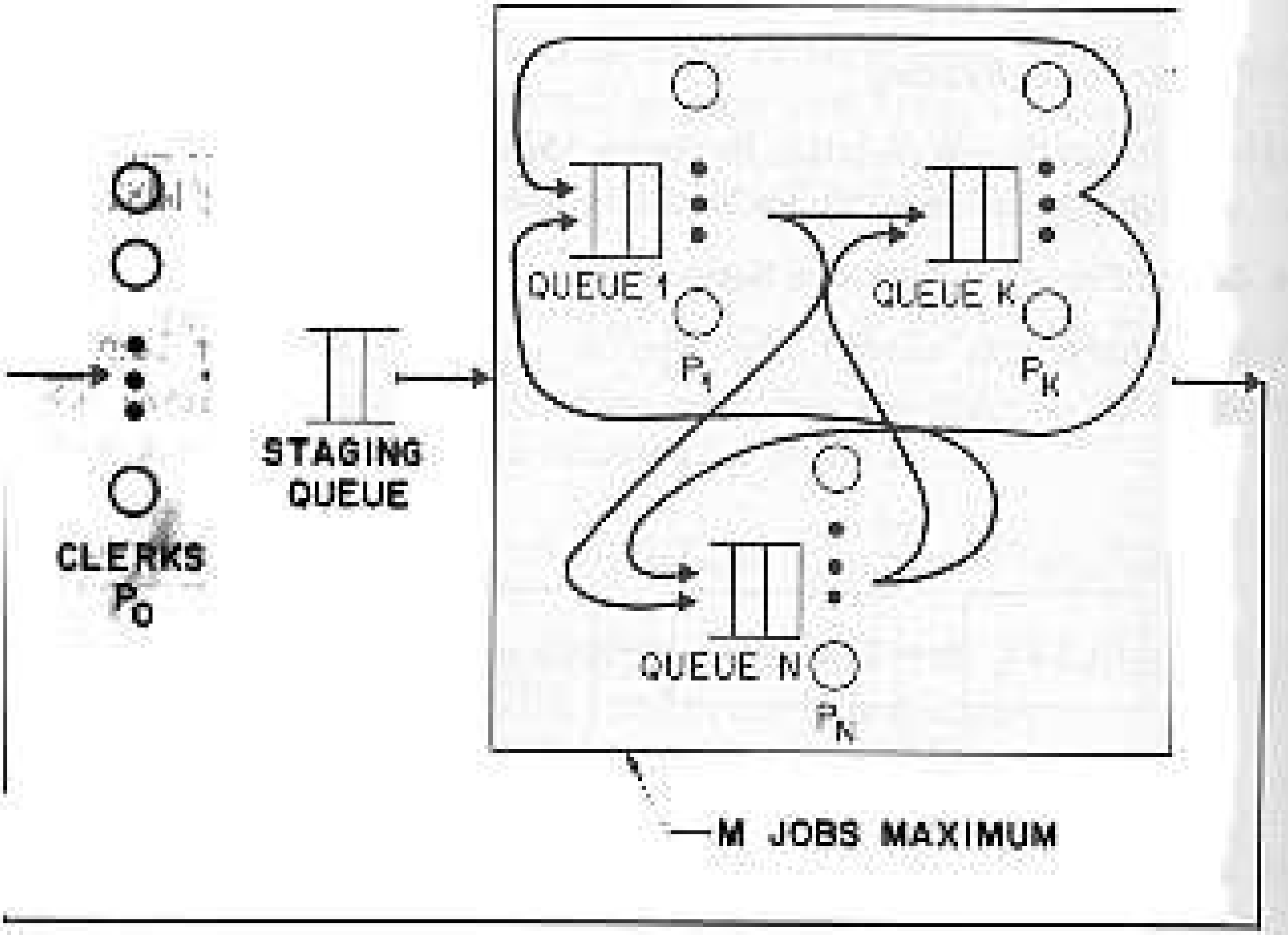

Figure 6.10.Queueing Network Model Block Diagram

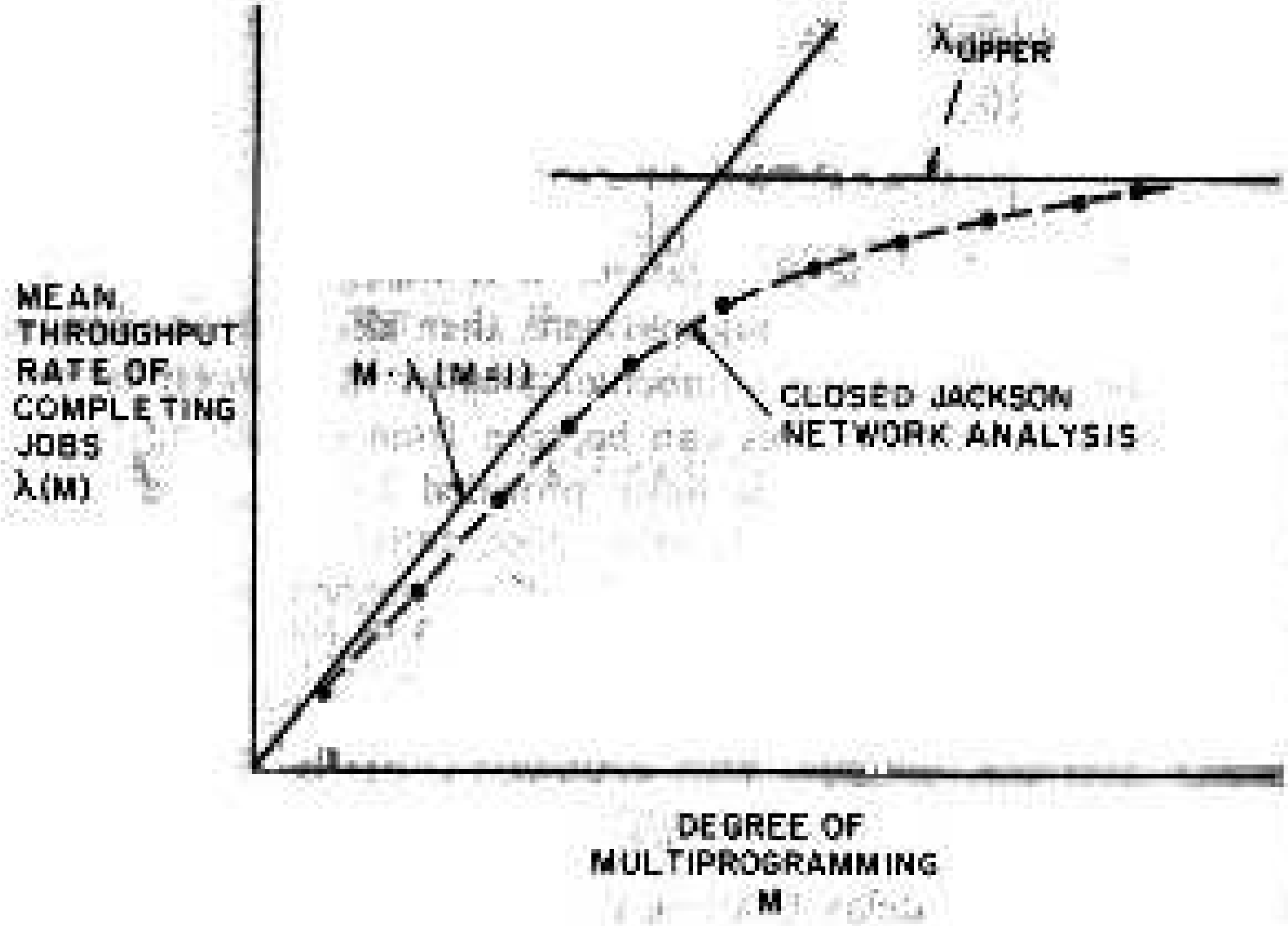

Figure 6.11.Mean Throughput Rate vs Degree of Multiprogramming

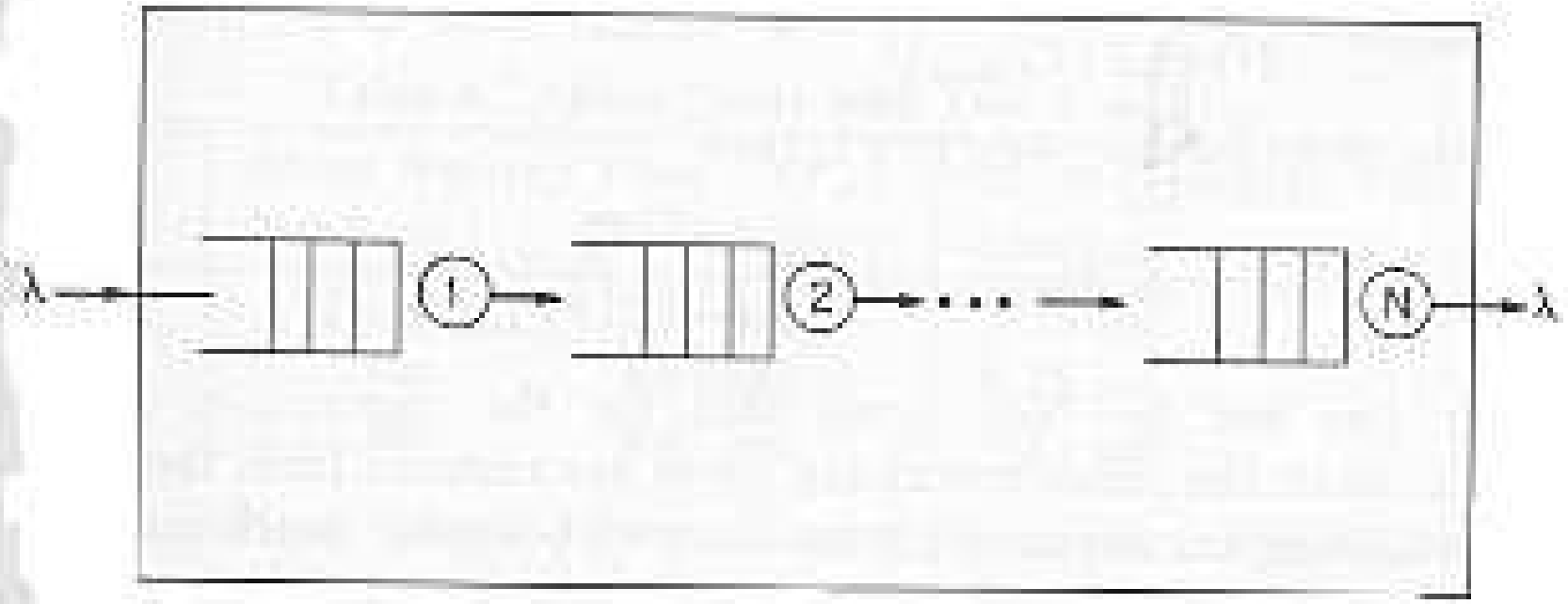

Figure 6.12.Pipeline Queueing Network Block Diagram

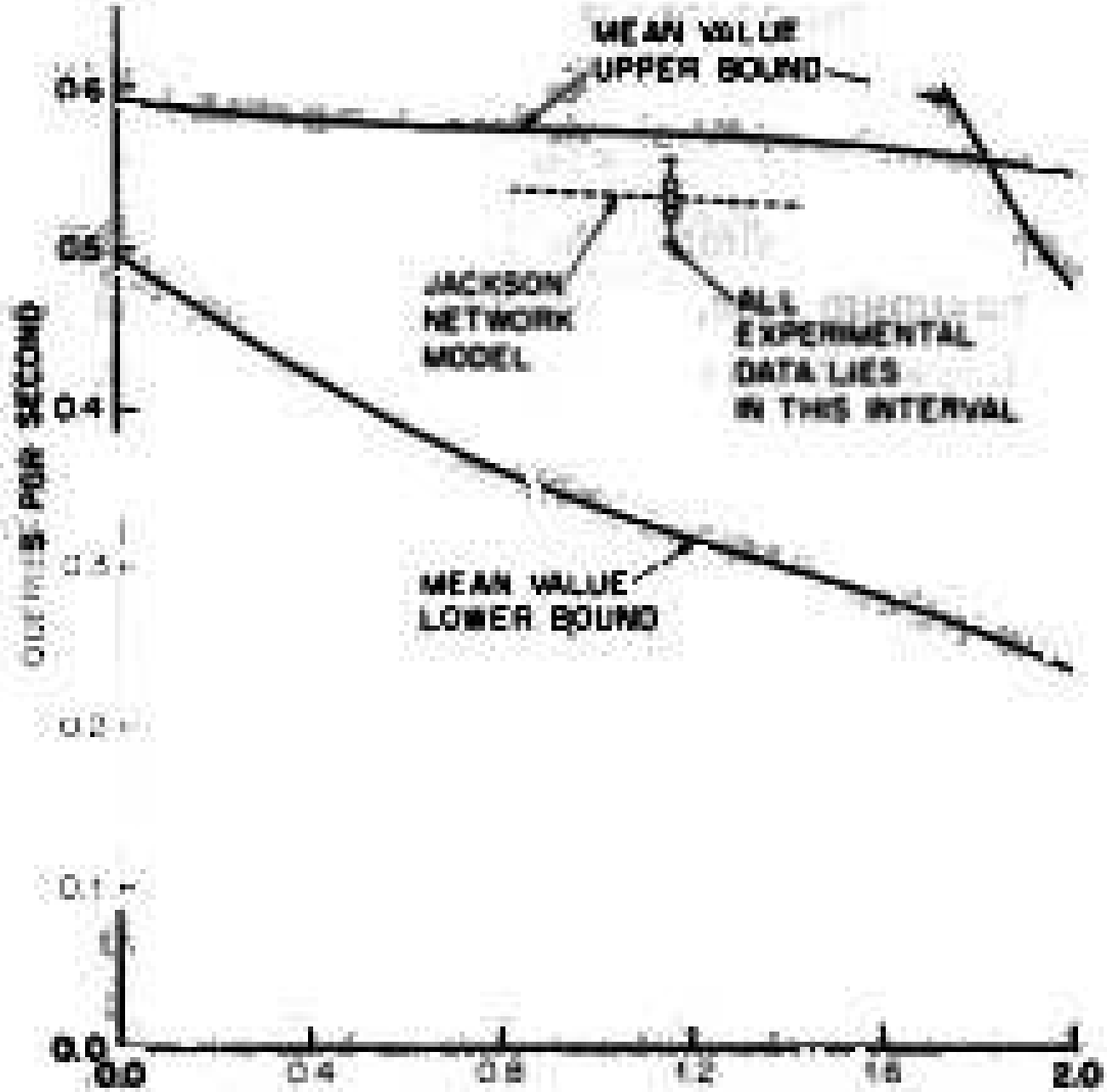

Figure 6.13.Mean Throughput Rate vs Mean Processor Time/Query

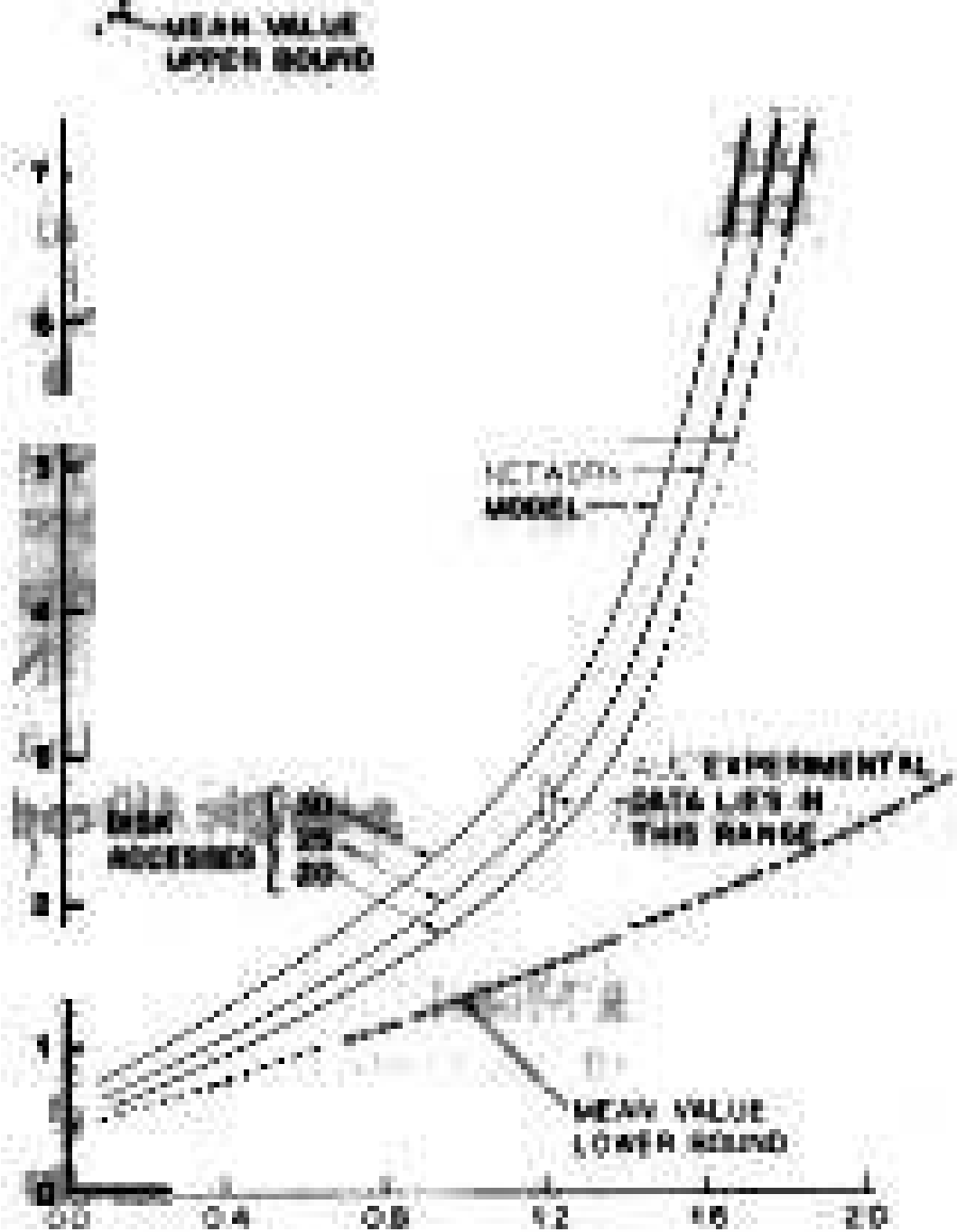

Figure 6.14.Mean Response Time vs Mean Processor Time/Query