CHAPTER 4: **MEAN VALUE ANALYSIS**

Up to this point we have examined detailed operational descriptions of models of computer communication systems. Much of the complexity in analysis *seemingly* disappears when the long term time averaged behavior of these types of computer system models is analyzed. We now focus on mean value analysis, long term time averaged behavior of mean throughput rate and mean delay. In our experience with operational systems and field experience, this is possibly the single most important fundamental result in analyzing computer communication system performance.

Our program here is to first analyze systems where there is only one type of job that takes only one step and one resource. Once we understand that case, we will generalize to systems where multiple types of jobs take multiple steps and demand multiple resources at each step.

4.1 Little's Law

In the technical literature, a key result is attributed to Little. In order to motivate the result, we return to a static model of a computer system. A system must process N jobs. All jobs are ready for processing at time t=0. The figure below plots the number in system process *J(t)* versus time t for one possible scenario.

Figure 4.1.Number in of Jobs in the System vs Time
Initial Work Present at T=0/No Arrivals for T>0

We wish to determine the mean number of jobs in the system over the time interval starting at zero until the system becomes empty. C_K denotes the time at which the Kth job completes execution and leaves the system. Since there are a total of N jobs, the last job will leave at C_N. The area under the function $J(t)$ over the time interval $(0,C_N)$ must by definition equal the mean number of jobs in the system, denoted by $E(J)$, multiplied by the observation interval duration, C_N:

$$E(J) = \frac{1}{C_N} \int_0^{C_N} J(t)dt$$

Since C_K is the completion time of the Kth transaction, K=1,...,N we see

$$E(J) = \frac{1}{C_N}\,[NC_1 + (N-1)(C_2-C_1) + \cdots + (C_N-C_{N-1})]$$

$$E(J) = \frac{1}{C_N}\sum_{K=1}^{N} C_K$$

F_K denotes the total time spent in the system by job K. Since job one spends C_1 amount of time in the system, while job two spends $C_1 + C_2$ amount of time in the system, and so forth, we see that

$$E(J) = \frac{1}{C_N}\sum_{K=1}^{N} F_K$$

We now rearrange this expression, by multiplying and dividing by N, the total number of jobs:

$$E(J) = \frac{N}{C_N}\,\frac{1}{N}\sum_{K=1}^{N} F_K$$

We recognize the first term as the mean throughput rate, by definition the total number of jobs divided by the observation interval:

$$\textit{mean throughput rate} = \frac{N}{C_N} \equiv \lambda$$

The second term is the mean time for a job to flow through the system, denoted by $E(F)$:

$$E(F) = \frac{1}{N}\sum_{K=1}^{N} F_K$$

Combining all this, we obtain the desired result:

$$\boxed{E(J) = \lambda \times E(F)}$$

The above result, that the mean number in system equals the mean throughput rate multiplied by the mean time in system, is called *Little's Law.*

We have obtained one equation in three unknowns, the mean number of jobs in system, the mean arrival rate, and the mean time in system. Our program is to *bound* each of these quantities, using the best available information. The more restrictive the information, the better the bounds; conversely, if little information is available, the bounds are still valid, but may not be as tight as desired.

4.1.1 Dynamic Arrivals Our intent is to relax the assumptions to obtain as general a result as possible. We first allow arrivals to occur at arbitrary points in time, rather than having all jobs present at time zero: A_K denotes the arrival time or ready time of the Kth job. $F_K \equiv C_K - A_K$ denotes the flow time of the Kth job, the time that elapses while job K flows through the system, from arrival (A_K) to departure (C_K). The figure below is an illustrative plot of the number of jobs in the system. Since each job contributes *one* unit of height for the duration of time it is in the system, since the area due to each job is its height (one) multiplied by its flow time, and since the total area is the sum of the areas contributed by each job, we see:

$$E(J) = \frac{N}{C_N}\,\frac{1}{N}\sum_{K=1}^{N}(F_K = C_K - A_K)$$

If we identify $\lambda \equiv N/C_N$ and $E(F)$ as before, then

$$E(J) = \lambda \times E(F)$$

What have we accomplished? We have one equation or relationship, and three parameters. In order to use this, we need to quantify two of the three parameters. An example is in order.

4.1.2 A Computer Time Sharing System One of the first applications of this type of analysis to computer systems was for time sharing program development, for MULTICS. In such a system, program developers spend some time reading and thinking and typing, with mean T_{think}, and then submit

Figure 4.2.Number in System vs Time
Initial Work Present at T=0/Arrivals for T>0

a job (an editor command, a compilation request, and so forth) that requires a mean amount of time T_{system} to be executed with no other load on the system.

With more than one request being executed, there will be contention, and hence some delay to execute each job. The mean response time for each job is measured from the time a job is submitted until the system finishes its response, denoted by R. The mean throughput rate, measured in jobs per unit time, is denoted by λ. There are a total of N program developers using the system actively at any one time. How can we use Little's Law to analyze performance?

The figure below is a hardware block diagram of this system:

Figure 4.3.Hardware Block Diagram of Time Sharing System

The key first step in using Little's Law is to determine what the system is: is it the terminals, is it the computer, or does it include both? If we define the total system as including both terminals and the system, then the total number of jobs is fixed at N, the number of active users, with each job either being in the think state or the processing state. If we define the system as including only terminals or only the system, we do not know how many jobs are in either.

For $N=1$ user, Little's Law says the mean number of jobs equals the mean throughput rate multiplied by the mean time in the system. The mean time in the system is the sum of the thinking time plus the response time with no load:

$$N=1 = \lambda(N=1)[T_{think} + R(N=1)]$$

We have written this to stress the fact that the mean throughput rate and the mean response time depend upon the number of active terminals. We could simply rewrite this by solving for the mean throughput rate:

$$\lambda(N=1) = \frac{N=1}{T_{think} + R(N=1)}$$

Next, as $N \to \infty$, the mean throughput rate will grow no faster than linearly with N, i.e., if we put on two terminals, the mean throughput rate is at best double that of one terminal:

$$\lambda(N) \leq \frac{N}{T_{think} + R(N=1)} \quad N>1$$

The mean response will be no better than that for one terminal:

$$R(N) \geq R(N=1) \quad N>1$$

At some point, the mean throughput rate does not increase any further, but rather saturates at a limiting value characteristic of the system, denoted by λ_{max}:

$$\lambda(N) \leq \lambda_{max}$$

Combining both these upper bounds on mean throughput rate, we see

$$\lambda(N) \leq \min\left[\lambda_{max}, \frac{N}{T_{think} + R(N=1)}\right]$$

>From Little's Law, the mean response time is given by

$$R(N) = \frac{N}{\lambda(N)} - T_{think}$$

Since the mean throughput rate eventually saturates at λ_{max}, the mean response time eventually grows linearly with N:

$$R(N) = \frac{N}{\lambda_{max}} - T_{think} \quad N \to \infty$$

We can combine all this in the following expression:

$$R \geq \max\left[R(N=1), \frac{N}{\lambda_{max}} - T_{think}\right]$$

There is a clear breakpoint here as N increases:

$$N_{breakpoint} = \lambda_{max}[T_{think} + R(N=1)]$$

For $N<N_{breakpoint}$ the terminals are unable to keep the system busy, and are a bottleneck. For $N>N_{breakpoint}$ the system is completely busy and is a bottleneck.

The worst that the mean response time could be would be to always wait for every other job to be processed immediately ahead of your execution:

$$R(N) \leq N \; R(N=1)$$

This in turn gives a lower bound on mean throughput rate:

$$\lambda(N) \geq \frac{N}{T_{think} + NR(N=1)}$$

Note that as $N \to \infty$, the lower bound on mean throughput rate approaches the rate of executing one job at a time:

$$\lambda(N) \geq \frac{1}{R(N=1)} \qquad N \to \infty$$

This suggests defining speedup as the ratio of the mean throughput rate for executing one job at a time over the actual mean throughput rate:

$$speedup = \frac{1}{R(N=1)\lambda(N)} \to \frac{1}{\lambda_{max} R(N=1)} \qquad N \to \infty$$

Figure 4.4 plots the upper and lower bounds on mean throughput rate versus number of active terminals.

Figure 4.4.Mean Throughput Rate Bounds versus Number of Active Users

Figure 4.5 plots the upper and lower bounds on mean response time versus number of active terminals.

Figure 4.5.Mean Response Time Bounds versus Number of Active Users

4.1.3 Additional Reading

[1] R.W.Conway, W.L.Maxwell, L.W.Miller, **Theory of Scheduling,** Addison-Wesley, Reading, Massachusetts, 1967; Little's formula, pp.18-19.

[2] J.D.C.Little, *A Proof of the Queueing Formula* $L = \lambda W$, Operations Research, **9,** 383-387 (1961).

[3] A.L.Scherr, **An Analysis of Time-Shared Computer Systems,** MIT Press, Cambridge, Mass, 1967.

4.1.4 Little's Inequality The only valid justification for *Little's Law* follows from controlled experimentation: testing the operation of a system to see how well it fits the hypothesis. Little's Law holds exactly when the system initially is empty or idle with no work, and after a period of observing its operation we stop gathering data when the system is once more entirely idle or empty. In practice, this may not be true: observations may be gathered over a finite time interval, and the state at the start of observation and the state at the end of observation may be different from the all empty or all idle state. Under a variety of technical conditions, which must be checked in practice, Little's Law holds *in some sense.* However, there is a weaker statement that we call *Little's Inequality* that holds :

$$mean\ number\ in\ system \geq mean\ arrival\ rate \times mean\ time\ in\ system$$

To make this precise, consider a system observed in operation from some time T to some later time $T + C_N$. We choose the interval of observation so that N jobs are observed to both start and finish; C_N is the completion instant of the Nth job. The jobs are denoted by $J_K, K=1,...,N$ with $J(.)$ denoting the total number of jobs in the system at any point in time, and $F_K, K=1,...,N$ denoting the time in system or flow time for job K. >From these definitions, and following the earlier development, we see

$$mean\ number\ of\ jobs = E(J) = \frac{1}{C_N} \int_{T}^{T+C_N} J(\tau) d\tau$$

Note that a job may have entered the system prior to T but not yet left, entered prior to T and left, or entered during the measurement time but not yet left. This implies that

$$E(J) = mean\ number\ in\ system \geq \frac{N}{C_N} \frac{1}{N} \sum_{K=1}^{N} F_K$$

We identify the mean throughput rate with λ

$$\lambda \equiv \frac{N}{C_N}$$

while we identify the mean time in system as

$$mean\ time\ in\ system = \frac{1}{N} \sum_{K=1}^{N} F_K$$

If you don't believe all this, draw a picture! On the other hand, if these *end* effects are negligible (which can only be checked with controlled experimentation), then

$$\boxed{E(J) \simeq \lambda \times E(F)}$$

>From this point on, we assume this holds with *equality.* This has been proven to be prudent and reasonable on numerous occasions, but must always be tested for *your* particular problem!

4.1.5 Jobs With Multiple Steps What if a job has more than one step? Consider the following analysis: Each job has S types of steps, and requires one or more resources at each step for a mean time $T_K, K=1,...,S$. The system is observed over an interval with N job completions occurring at time instants $C_K, K=1,...,N$. We denote with an S tuple $\underline{J} = (J_1,...,J_S)$ the number of jobs in execution in each step, i.e., $J_K, K=1,...,S$ denotes the number of jobs in execution in step K. The state space is denoted by Ω and is the set of feasible S tuples $\underline{J}$. The fraction of time the system is in each state $\underline{J}$ over the observation interval is denoted by $\pi(\underline{J})$. From Little's Law, we can write

$$E(J_K) = \sum_{\underline{J} \in \Omega} J_K \pi(\underline{J}) \geq \lambda \tilde{T}_K \qquad K=1,...,S$$

where the mean throughput rate is simply the total number of jobs divided by the observation time

interval,

$$\lambda \equiv \frac{N}{C_N}$$

and $\tilde{T}_K$ is given by averaging T_K over the fraction of time the system is in each state:

$$\tilde{T}_K \equiv \frac{1}{N} \sum_{K=1}^{N} T_K \pi(J_K)$$

EXERCISE: Derive this for N jobs present at time zero and no further arrivals.

EXERCISE: Derive this with the system idle at some initial time, and with N arrivals and departures occurring at random points in time, so that the system is once again idle.

4.2 Scheduling a Single Processor to Minimize the Mean Flow Time

Suppose that N jobs are present at an initial time, say zero, and a single processor executes these jobs until no jobs are left. Let C_K denote the completion time of the Kth job. Let $J(t)$ denote the number of jobs still not completely executed at time t. The area under $J(t)$ is given by its integral:

$$\int_0^{C_N} J(t)dt = \sum_{K=1}^{N} C_K$$

The mean time a job spends in the system is given by

$$E(F) = \frac{1}{N} \sum_{K=1}^{N} C_K$$

Our problem is to find a schedule for a single processor that minimizes the average or mean flow time of a job. Note that the schedule cannot change C_N, the completion time of the last job, because the processor will always be busy doing *some* job until it finishes them all, but we *can* control the completion times of the jobs. Once we realize this, we see that

$$E(F) = \frac{1}{N} \int_0^{C_N} J(t)dt$$

Hence we wish to minimize the area under $J(t)$. Intuitively, we wish to drive $J(t)$ as close to zero as quickly as possible, i.e., we want to schedule jobs with the *shortest* processing time first.

To make this concrete, suppose there are two jobs, one with processing time equal to 10, and one with processing time equal to 1. One schedule is to run the short job first and then the long job, with mean time in system given by

$$E(F) = ½(1+11) = 6$$

while a second schedule is to run the long job first and then the short job, with mean time in system given by

$$E(F)average = ½(10+11) = 10.5$$

What we are trying to accomplish is to make sure that short jobs are not delayed by long jobs.

4.3 Telephone Traffic Engineering

Voice telephone calls are made between two locations. The only data available is that during a peak busy hour of the day, on the average $\lambda_{telephone}$ is the mean rate of calls arriving per minute that are successfully completed, with a mean holding time per successful call of $T_{telephone}$. A total of C circuits are installed. How many links L are actually needed, where one link can handle one voice telephone call?

The mean number of calls in progress during this busy hour is given by Little's Law:

Figure 4.6.Telephone System Block Diagram

$$mean\ number\ of\ calls\ in\ progress = \lambda_{telephone} T_{telephone}$$

More formally, the number of calls in progress at any instant of time say t is denoted by $J(t)$. The state space is the set of admissible nonnegative integer values for $J(t)$ denoted by Ω. The system is observed over a time interval of duration T and the fraction of time the system is in a given state is assumed to have stabilized at $\pi(J)$. The maximum available number of circuits is C, and hence

$$E[min(J,C)] = \sum_{J \in \Omega} \pi(J) min(J,C) = \lambda_{telephone} T_{telephone}$$

Since each link can handle one call, we see that we need (roughly)

$$L \approx \lambda_{telephone} T_{telephone}$$

The units associated with the mean number of calls in progress are called *Erlangs* in honor of the Danish teletraffic engineer A.Erlang who pioneered much of the early work to quantify and answer questions such as these. In practice, we would put in more than L links as given by Little's Law because we will have fluctuations about the mean value and some call attempts will be blocked or rejected because all links are busy. We will return to this topic later.

4.4 Serially Reusable versus Concurrently Shared Resources

In many applications, some resources must be used serially, one task at a time, and others can be shared simultaneously or in parallel with many tasks. In the illustrative figure below, a job consists of two steps: the first step requires a shared resource such as reentrant application code, while the second step requires a serially reusable resource such as operating system code. Multiple simultaneous uses can be made of the shared resource, but only one job at a time may use the serially reusable resource.

For application programs executing on IBM OS/360, typically 40% of the processor time was found to be devoted to serially reusable resources, and with a great deal of effort this might be reduced to 20% of the processor time (Amdahl, 1967). This suggests that identical multiple processor configurations with one processor devoted to serial tasks and P devoted to the parallel tasks might find little benefit in going to more than two processors (60%/40% < 2) for a 1967 implementation of OS/360, while going to more than four processors (80%/20% =4) for an optimistic scenario. Furthermore, since the serial tasks are a fundamental bottleneck, every effort should be made to make these execute as quickly as possible.

Figure 4.7.Hardware Block Diagram of Serial and Concurrent Processors

Here is a somewhat more quantitative approach to these intuitive notions. A job consists of two steps. The first step must be done using a serially reusable resource (e.g., a critical region of the system), and will execute in time T_{serial}. The second step can be done concurrently, and requires $T_{concurrent}$ units of time to be executed. We might think of the serial portion being the work associated with classifying a step, readying it for subsequent execution, while the second portion might involve execution using a read only storage for example.

The system state is a pair, denoted $(J_{serial}, J_{concurrent})$, describing the number of jobs in execution on the serially reusable resource J_{serial} and on the concurrently shared resource $J_{concurrent}$.

The mean number of tasks in execution and requiring the serially reusable resource is given by

$$\lambda T_{serial} = E(J_{serial}) \leq 1$$

while the mean number of tasks in execution and requiring the shared resource is given by

$$\lambda T_{concurrent} = E(J_{concurrent}) \leq P$$

These relations give us the following upper bound on the mean throughput rate:

$$\lambda = \min\left[\frac{1}{T_{serial}}, \frac{P}{T_{concurrent}}\right]$$

A second type of upper bound arises from a limitation on the total number of jobs, denoted M, in the system. With one job, M=1, in the system, the mean throughput rate is given by

$$\lambda \leq \frac{1}{T_{serial} + T_{concurrent}} \quad J=1$$

and hence as more jobs are allowed in the system, M>1, the best that could be done is

$$\lambda \leq \frac{M}{T_{serial} + T_{concurrent}}$$

Combining this with the other upper bound, we see

$$\lambda \leq \min\left[\frac{M}{T_{serial} + T_{concurrent}}, \frac{1}{T_{serial}}, \frac{P}{T_{concurrent}}\right]$$

There are three types of bottleneck possible:

- The number of jobs is a bottleneck

$$\lambda_{\max} = \frac{M}{T_{serial} + T_{concurrent}}$$

- The serially reusable resource is a bottleneck

$$\lambda_{\max} = \frac{1}{T_{serial}}$$

- The concurrently shared resource is a bottleneck

$$\lambda_{\max} = \frac{P}{T_{concurrent}}$$

Here is a different example to illustrate the importance of this phenomenon in doing tradeoffs. A processor spends 25% of its time on a representative job doing input/output over various communication links, and 75% of its time actually executing work. A front end processor is procured to offload the input/output work, and the front end processor is twice as fast as the original processor. What is the potential gain in maximum mean throughput rate? If the original system required $T_{i/o}$ seconds for each job and $T_{process}$ to execute each job, while the new front end processor requires T_{fe} seconds for each job, then the mean throughput rate is upper bounded for the old system by

$$mean\ \ throughput\ \ rate \leq \frac{1}{T_{i/o} + T_{process}} \quad old\ \ system$$

while the mean throughput rate for the new system is governed by the slower of the two subsystems

$$mean\ \ throughput\ \ rate \leq \frac{1}{max[T_{fe}, T_{process}]} \quad new\ \ system$$

Suppose the front end processor is twice as fast as the original processor, so

$$T_{fe} = \tfrac{1}{2}\ T_{i/o}$$

Finally, suppose that 25% of the time the original processor is doing input/output, and 75% of the time it is doing work, so

$$T_{process} = 3\ T_{i/o}$$

Combining all this, the mean throughput rate for the old system is upper bounded by

$$mean\ \ throughput\ \ rate = \frac{1}{4T_{i/o}} \quad old\ \ system$$

while the mean throughput rate for the new system is upper bounded by

$$mean\ \ throughput\ \ rate = \frac{1}{max[3T_{i/o}, \tfrac{1}{2}T_{i/o}]} \quad new\ \ system$$

and hence the maximum gain is (4/3) or thirty three per cent. In fact, as long as the front end processor is as fast or faster than the original processor, the maximum gain will be thirty three per cent. However, the delay in getting through from start to finish ignoring delays due to congestion is

$$T_{execution} = T_{i/o} + T_{process} = 4T_{i/o} \quad old\ \ system$$

$$T_{execution} = T_{i/o} + T_{process} = 3\tfrac{1}{2}T_{i/o} \quad new\ \ system$$

and this time depends directly on the speed of the original processor and the front end processor.

What if a different front end processor can be procured that is one half the speed of the original processor, but costs one third what the other front end processor cost? Now the time per job for a front end processor equals

$$T_{fe} = 2T_{i/o} \quad new\ \ front\ \ processor$$

and hence if only one front end processor is added to the system, the mean throughput rate is upper

bounded by

$$mean\ \ throughput\ \ rate \leq \frac{1}{max[T_{fe} = 2T_{i/o}, T_{process} = 3T_{i/o}]}$$

which is just as good as the more expensive front end processor, but the execution time is greater for one inexpensive but slower front end processor:

$$T_{execute} = T_{fe} + T_{process} = 5T_{i/o}$$

This is a typical finding: the mean throughput rate can be increased, at the expense of delay.

4.4.1 Additional Reading

[1] G.M.Amdahl, *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*, AFIPS Conference Proceedings, **30,** 483-485, AFIPS Press, Montvale, NJ, 1967.

4.5 Packet Computer Communications Network

The block diagram below shows nodes in a packet computer communications network:

Figure 4.8.Packet Network Node Block Diagram

Messages enter or leave the network via one of N ports. There are S nodes or packet switches within the network. At node K we measure R_K packets per second for a mean throughput rate. The mean flow time of a packet, waiting plus switching time, at node K is given by $E(F_K)$. The mean flow time of a packet through the network is denoted by $E(F)$, while the total mean external arrival rate at port J is denoted by λ_J, and we denote by λ the aggregate total network throughput rate:

$$\lambda = \sum_{J=1}^{N} \lambda_J$$

>From Little's Law we see

$$\lambda E(F) = \sum_{K=1}^{S} R_K E(F_K)$$

Since packets can be switched through more than one internal node, the packet throughput rate within the network can exceed the external packet arrival and departure rate:

$$R \equiv \sum_{K=1}^{S} R_K \geq \lambda = \sum_{J=1}^{N} \lambda_J$$

The mean flow time can be written as

$$E(F) = \sum_{K=1}^{S} \frac{R_K}{\lambda} E(F_K)$$

If we rewrite this as

$$E(F) = \frac{R}{\lambda} \sum_{K=1}^{S} \frac{R_K}{R} E(F_K)$$

then we can identify the mean number of nodes visited per packet V with

$$V = \frac{R}{\lambda}$$

while we recognize the mean flow time per node is

$$mean\ \ flow\ \ time\ \ per\ \ node = \sum_{K=1}^{S} \frac{R_K}{R} E(F_K)$$

What have we learned? The mean time a packet spends in the network equals the mean number of nodes visited per packet multiplied by the mean time per node.

4.6 A Model of a Processor and Disk System

Jobs that require more than one step are quite common. This section analyzes a model of a computer system consisting of processors and disks, with the first step of any job requiring a processor, the second a disk, the third a processor again, and so forth, until the job is completely executed.

4.6.1 Model The figure below shows a hardware block diagram of the system, while the next figure shows a queueing network block diagram.

Figure 4.9.Processor and Disk Hardware Block Diagram

The system consists of P processors and D disks connected by a common switch. The switch is

Figure 4.10.Queueing Network Block Diagram

assumed to be much faster than any step of job execution involving either a processor or a disk, and is ignored from this point on. The jobs or transactions are generated from people or operators at terminals. Each operator spends a mean amount of time reading, thinking, and typing, denoted by T_{think} and then submits a job to the system and waits for a response before repeating this process. The mean response time is denoted by R and is the sum of four time intervals, due to waiting or being executed on a processor or disk.

Each job involves execution on a processor, data retrieval from disk, and so on, until the job is completely executed. Each job requires a total mean amount of time denoted by $T_{processor}$ and T_{disk} on a processor and disk respectively. No job is assumed to be capable of executing in parallel with itself. The operating system multiplexes available jobs among available processors and disks to achieve some degree of concurrent use of resources.

4.6.2 Analysis The system state is given by a triple, where $(J_{operator}, J_{processor}, J_{disk})$ where $J_{operator}$ denotes the number of operators reading, thinking and typing, $J_{processor}$ denotes the number of jobs waiting or executing on a processor, and J_{disk} denotes the number of jobs waiting to use a disk or retrieving data from a disk.

>From Little's Law, we see that the mean number of tasks executing on processors is equal to the mean arrival time multiplied by the mean time spent using a processor:

$$E(J_{processor}) = \lambda T_{processor} \leq P$$

Similarly, the mean number of tasks using disks is equal to the mean arrival rate multiplied by the mean time spent using a disk:

$$E(J_{disk}) = \lambda T_{disk} \leq D$$

>From the above relations, we see that the mean throughput rate of jobs is upper bounded by

$$\lambda \leq \min\left[\frac{P}{T_{processor}}, \frac{D}{T_{disk}}\right]$$

A second type of upper bound arises from considering how work flows through the system: if there is only one job in the system at any time, then the mean throughput rate is simply given by

$$\lambda \leq \frac{1}{T_{processor} + T_{disk}} \quad one\ job\ in\ system$$

and hence with J jobs in the system the mean throughput rate is upper bounded by J times that for one job:

$$\lambda \leq \frac{J}{T_{processor} + T_{disk}} \quad J\ jobs\ in\ system$$

Combining this upper bound with the previous upper bound we see

$$\lambda \leq \min\left[\frac{P}{T_{processor}}, \frac{D}{T_{disk}}, \frac{J}{T_{processor} + T_{disk}}\right]$$

Since from Little's Law or the definition of mean throughput rate the mean response time and mean throughput rate are related by

$$\lambda = \frac{J}{T_{think} + R}$$

we obtain the following lower bound on mean response time:

$$R \geq \max\left[T_{processor} + T_{disk}, \frac{J}{max[T_{processor}/P, T_{disk}/D]} - T_{think}\right]$$

To lower bound the mean throughput rate, we realize that the J jobs could all be waiting to run or be running on a processor, or waiting to run to be retrieving data from a disk, and hence

$$\frac{J}{T_{think} + \frac{JT_{processor}}{P} + \frac{JT_{disk}}{D}} \leq \lambda$$

This allows us to upper bound the mean response time:

$$R \leq \frac{JT_{processor}}{P} + \frac{JT_{disk}}{D}$$

These bounds define an admissible or feasible region of operation and are plotted in the figures below for the case of one processor and one disk.

Figure 4.11.One Processor/One Disk Mean Throughput Bounds vs Number of Terminals.

Figure 4.12.One Processor/One Disk Mean Response Time Bounds vs Number of Terminals.

4.6.3 Speedup Here are two possible scheduling policies:

- *single thread* scheduling, where one job at a time is allowed in the system and executed until completion before allowing the next job in
- *multiple thread* scheduling, where more than one job at a time is allowed in the system and executed until completion

For the first policy, we see

$$\lambda_{single\ thread} = \frac{1}{T_{processor} + T_{disk}} \quad J=1$$

The ratio of the two different upper bounds is an indication of the gain due to scheduling:

$$\frac{\lambda_{multiple\ thread}}{\lambda_{single\ thread}} = \frac{T_{processor} + T_{disk}}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}, \frac{T_{processor} + T_{disk}}{J}\right]}$$

For one processor and one disk, this gain due to scheduling can be at most two, no matter what $T_{processor}$ or T_{disk} are! Moreover, this will only be achieved when $T_{processor}$ equals T_{disk}, but in general these two mean times will *not* be equal and hence the gain will *not* be as great as a factor of two; for example, if T_{disk} were ten times as great as $T_{processor}$, then the gain would be at most ten per cent, and other factors may swamp this *upper* bound. J is called the *degree of multiprogramming* and the two cases we have examined are degree of multiprogramming one and two, for single and multiple thread operation, respectively. If we allow multiplexing of the processor and disk amongst transactions, then $J>1$ is allowed, but now one or the other of the two serially reusable resources will become completely utilized for J sufficiently large.

4.6.4 Bottlenecks Bottlenecks can arise from several sources:

- If the number of jobs or degree of multiprogramming is a bottleneck, then

$$\lambda_{\max} = \frac{J}{T_{processor} + T_{disk}}$$

- If the number of processors is a bottleneck, then

$$\lambda_{\max} = \frac{P}{T_{processor}}$$

- If the number of disks is a bottleneck, then

$$\lambda_{\max} = \frac{D}{T_{disk}}$$

The design problem is to choose where the bottleneck should be; remember, there will always be *some* bottleneck!

4.6.5 Asymptotics One type of asymptotic analysis is to let all parameters be fixed except one, and the final one becomes progressively larger and larger. Here a natural candidate for such a parameter is the number of operators or jobs circulating in the system J, and we see

$$\frac{1}{\frac{T_{processor}}{P} + \frac{T_{disk}}{D}} \leq \lambda \leq \frac{1}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}\right]} \qquad J \to \infty$$

This yields the following asymptotic behavior for the mean response time:

$$T_{processor} + T_{disk} \leq R \leq \infty \qquad J \to \infty$$

This is quite instructive by itself: the mean response time can lie between a *finite* and *infinite* limit, showing how great the variation *can* be, given only mean value information.

A second type of asymptotic analysis is to fix the ratio of two parameters, and allow them both to become progressively larger, fixing all other parameters. Here, a natural candidate is the ratio of the number of jobs divided by the mean think time per operator, which we fix at α

$$\alpha \equiv \frac{J}{T_{think}} \qquad J \to \infty \quad T_{think} \to \infty$$

which is a measure of the *total* offered rate of submitting jobs, and we allow the number of jobs or terminals to become large as well as the mean intersubmission time of jobs from each terminal, thus weakening the contribution to the total offered rate of each terminal. If we do so, we see

$$\frac{\alpha}{1 + \alpha\left[\frac{T_{processor}}{P} + \frac{T_{disk}}{D}\right]} \leq \lambda \leq \frac{1}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}\right]}$$

$$R \geq \begin{cases} \infty & \alpha > \dfrac{1}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}\right]} \\ T_{processor} + T_{disk} & \alpha < \dfrac{1}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}\right]} \end{cases}$$

where in the above summary, we have fixed α, but allowed both $J \to \infty$ and $T_{think} \to \infty$.

Additional (distributional) information must be available to allow us to handle the case where

$$\alpha = \frac{1}{\max\left[\frac{T_{processor}}{P}, \frac{T_{disk}}{D}\right]}$$

Intuitively we see that the if the total mean arrival rate is less than the upper bound on the mean throughput rate, then the system is capable of having a *finite* lower bound on mean response time; when the total mean arrival rate is greater than the upper bound on the mean throughput rate, then the mean response time lower bound is *infinite.* The remaining case, an upper bound on the mean response time,

is trivial

$$R \leq \infty \quad \alpha\ fixed, J \rightarrow \infty\ T_{think} \rightarrow \infty$$

As in the first case, the mean response time can lie between a *finite* and *infinite* value, given only mean value information, i.e., the mean response time is *not* well bounded given only means but no information about fluctuations. *Mean* delay depends not only on the *mean* processing times but also *second* moments of the processing time distribution: mean value information does *not* specify the mean delay in such systems by itself.

4.7 Mean Throughput and Mean Delay Bounds in Multiple Step Single Job Networks

In this section we present the analysis that leads to *upper* as well as *lower* bounds on mean throughput and mean delay for a particular type of computer communication system that handles only one type of job. The utility is that the bounds are in fact *achievable,* and hence *sharp,* or the best possible bounds, given *only* the mean duration of each step.

4.7.1 Model Each job consists of one or more steps. At each step, a given amount of a serially reusable resource is required for a given mean time interval. Here the first step of each job involves entering the job into the system via an operator at a terminal, the second step of each job involves placing the job in a staging queue where it will wait if there are more than a given maximum number of jobs already in the system and otherwise will enter the system immediately, and one or more additional steps inside the system where the job holds a single serially reusable resource for each step of execution and then moves on, until the job is completely executed and control returns to the operator at the terminal. For each step of each job, we are given the amount of each resource and the *mean* time required to hold that set of resources. We denote by T_K the total mean time spent by a job holding resource type K, which we stress is the sum total execution time of all visits to that stage by a job.

The mathematical model consists of

- N+2 stages of stations: station 0 is associated with operators at terminals, station 1 is the staging station, and stations 2,...,N+1 (N total) are associated with a single serially reusable resource
- Stage K=0,2,...,N+1 has P_K identical parallel servers or processors
- A maximum of M jobs can be held at all stages K=2,...,N+1
- Each job moves from station to station, and requires T_K total mean amount of service time at stage K=0,2,...,N+2

The figure below is a queueing network block diagram of this system.

We denote by λ the total mean throughput rate of completing jobs; R denotes the total mean response time (queueing or waiting time plus execution time) per job. The system state space is denoted by Ω. Elements in the state space are denoted by $\underline{J}=(J_0, \ldots, J_{N+1})$. J_K,K=0,2,...,N+1 denotes the number of jobs either waiting or in execution at stage K. Feasible elements in the state space obey the following constraints:

[1] The total number of tasks in the system is fixed at P_0

$$P_0 = |\underline{J}| = \sum_{K=0}^{N+1} J_K$$

[2] There can be at most a maximum of M jobs inside the system:

$$\sum_{K=2}^{N+1} J_K = \min[M, P_0 - J_0]$$

Combining all these, we see that elements $\underline{J}$ in Ω are nonnegative integer valued tuples where Ω is given by the set of all N+2 tuples $\underline{V} = (V_0, \ldots, V_{N+1})$ such that

Figure 4.13.Block Diagram of Memory Constrained Queueing Network

$$V_K \geq 0 \;\; K=0,...,N+1; \sum_{K=0}^{N+1} V_K = P_0; \sum_{K=2}^{N+1} V_K = min\,[M, P_0 - V_0]\}$$

The number of jobs *in execution* at stage $K=0,2,...,N+1$ is given by $\min[J_K, P_K]$ at any given instant of time. From the previous section, Little's Law allows us to write:

$$mean\;\; number\;\; in\;\; execution\;\; at\;\; stage\;\; K \equiv E\,[\min(J_K, P_K)] = \lambda T_K \qquad K=0,2,...,N+1$$

where $E(.)$ denotes the time average of the argument. Our goal is to find upper and lower bounds on λ subject to the state space constraints on $J_K, K=0,...,N+1$. Since mean throughput rate and mean response time or delay are related via

$$\lambda = \frac{P_0}{T_0 + R}$$

we will also obtain associated lower and upper bounds on mean delay.

4.7.2 Lower Bound on Mean Throughput Rate We first divide both sides of the following equation

$$E\,[min\,(J_0, P_0)] = \lambda T_0$$

by P_0. In like manner, we divide both sides of the following equations

$$\lambda T_K = E\,[min\,(J_K, P_K)] \quad K=2,...,N+1$$

by $min\,[M, P_0, P_K]$. Now we add up these $N+1$ equations:

$$\frac{E\,[min\,(J_0, P_0)]}{P_0} + \sum_{K=2}^{N+1} \frac{E\,[min\,(J_K, P_K)]}{min\,[M, P_0, P_K)]} = \lambda \left[\frac{T_0}{P_0} + \sum_{K=2}^{N+1} \frac{T_K}{min\,[M, P_0, P_K]} \right]$$

Now we interchange the mean value with the summation on the left hand side:

$$E\left[\frac{min\,[(J_0, P_0)]}{P_0} + \sum_{K=2}^{N+1} \frac{min\,[J_K, P_K]}{min\,[M, P_0, P_K]} \right] = \lambda \left[\frac{T_0}{P_0} + \sum_{K=2}^{N+1} \frac{T_K}{min\,[M, P_0, P_K]} \right]$$

Our goal is to lower bound the left hand side by one, which will yield a lower bound on λ.

Two cases can arise. First, there can exist one $I=2,...,N+1$ such that $P_I \leq J_I$. Since all the terms on the left hand side are nonnegative, we can lower bound the left hand side by ignoring all of these terms except term $I=2,...,N+1$:

$$\frac{min[J_0,P_0]}{P_0} + \sum_{K=2}^{N+1} \frac{min[J_K,P_K]}{min[M,P_0,P_K]} \geq \frac{min[J_I,P_I]}{min[M,P_0,P_I]} \geq \frac{P_I}{min[M,P_0,P_I]} \geq 1$$

Second, for all $K=0,2,...,N+1$, $P_K > J_K$ and hence

$$min[J_K,P_K] = J_K \quad K=2,...,N+1$$

Two subcases arise: if $P_0 - J_0 \leq M$ then there is no waiting by any job in the staging queue, and

$$\frac{J_0}{P_0} + \sum_{K=2}^{N+1} \frac{J_K}{min[M,P_0,P_K]} \geq \frac{J_0}{P_0} + \frac{P_0 - J_0}{P_0} = 1$$

The other subcase is if $P_0 - J_0 > M$ and then there is waiting in the staging queue, so

$$\frac{min[J_0,P_0]}{P_0} + \sum_{K=2}^{N+1} \frac{J_K}{min[M,P_0,P_K]} \geq \sum_{K=2}^{N+1} \frac{J_K}{min[M,P_0,P_K]} = \frac{M}{M} = 1$$

Hence, we see that

$$\lambda \left[\frac{T_0}{P_0} + \sum_{K=2}^{N+1} \frac{T_K}{min[M,P_0,P_K]} \right] \geq 1$$

and we obtain the desired lower bound:

$$\lambda_{lower} = \frac{P_0}{T_0 + \sum_{K=2}^{N+1} \frac{P_0}{min[M,P_0,P_K]} T_K}$$

The total mean time to execute a job at each stage in the system has been stretched from $T_K, K=2,...,N+1$ to $\tilde{T}_K, K=2,...,N+1$ where

$$\tilde{T}_K = \frac{P_0}{min[M,P_0,P_K]} T_K \geq T_K \quad K=2,...,N+1$$

$$\lambda_{lower} = \frac{P_0}{T_0 + \sum_{K=2}^{N+1} \tilde{T}_K}$$

which is one way of quantifying the slowdown at each node due to congestion.

4.7.3 Upper Bound on Mean Throughput Rate >From the definition of λ we see

$$\lambda = \frac{E[\min(J_K,P_K)]}{T_K} \leq \frac{min[P_K,P_0,M]}{T_K} \quad K=0,2,...,N+1$$

>From this same identity, we obtain a second upper bound:

$$\lambda T_K \leq E[J_K] \quad K=0,2,...,N+1 \quad \rightarrow$$

$$\lambda[T_0 + \sum_{K=2}^{N+1} T_K] \leq E[J_0 + \sum_{K=2}^{N+1} J_K] = P_0$$

The constraint on the maximum number of jobs inside the system can be written as

$$\sum_{K=2}^{N+1} J_K \leq min[P_0,M]$$

If we use Little's Law, we see

$$\lambda \sum_{K=2}^{N+1} T_K = \sum_{K=2}^{N+1} E[J_K] \leq min[M, P_0]$$

In summary, we have shown

$$\lambda \leq \min \left[\min_{K=0,2,...,N+1} \left[\frac{min[P_0, P_K, M]}{T_K} \right], \frac{P_0}{T_0 + \sum_{K=2}^{N+1} T_K}, \frac{min[M, P_0]}{\sum_{K=2}^{N+1} T_K} \right]$$

4.7.4 Interpretation One intuitive explanation for these bounds is the following. To achieve the upper bound on mean throughput rate, each step of job execution has little fluctuation relative to its mean value, and jobs interleave with one another. The mean throughput rate can be upper bounded via the following mechanisms:

- The total number of jobs circulating in the system is limiting the mean throughput rate; in this regime, as we increase the number of jobs, the mean throughput rate increases in roughly the same proportion
- One stage is executing jobs at its maximum rate, limiting the mean throughput rate; in this regime, as we increase either the speed of each processor at that stage, or the number of processors with the same speed, the mean throughput rate increases in roughly the same proportion
- The constraint on the maximum number of jobs in the system is limiting the mean throughput rate; in this regime, as we increase the allowable maximum number of jobs in the system, the mean throughput rate increases accordingly

To achieve the lower bound on mean throughput rate, each step of job execution has large fluctuations relative to its mean value, so that all jobs in the system are congested at one node. A different way of gaining insight into this lower bound is to replace the service or processing time distribution at each node with a bimodal distribution with the same mean as the old distribution, where $(1-\varepsilon_K)$ denotes the fraction of jobs at stage K that are executed in *zero* time and ε_K are the fraction of jobs at stage K that are executed in time $1/\mu_K$ such that $T_K = \varepsilon_K/\mu_K$. Here in normal operation two things can occur: the mean time for a job to cycle through the network will be roughly zero, since most stages will take zero time, and hence the number of jobs in circulation will limit the mean throughput rate, or one stage of execution will take a time that is much longer relative to all the other times, and hence all but one or two jobs will be congested at one node, thus limiting the mean throughput rate.

4.7.5 Additional Reading

[1] R.W.Conway, W.L.Maxwell, L.W.Miller, **Theory of Scheduling,** Addison-Wesley, Reading, Massachusetts, 1967; Little's formula, pp.18-19.

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

[3] J.D.C.Little, *A Proof of the Queueing Formula* $L = \lambda W$, Operations Research, **9,** 383-387 (1961).

[4] W.L.Smith, *Renewal Theory and Its Ramifications*, J.Royal Statistical Society (Series B), **20(2),** 243-302(1958).

4.8 A Mean Throughput Rate Upper Bound for Multiple Class Single Resource Models

Now we sketch how to extend the analysis in the previous section that led to an upper bound on mean throughput rate to a system that processes multiple types of jobs, not just one.

4.8.1 Model A computer communication system must execute C classes of jobs. Each job consists of one or more steps, and each step requires a single resource for a mean amount of time. $T_{IK} \geq 0$ denotes the total mean amount of time, summed over all steps, that class K job requires resource $I=1,...,N$.

The system configuration is $P_I, I=1,...,N$ servers or processors for resource I, where $P_I=1,2,3...$ Each resource is fed by a single queue of jobs, i.e., any processor can execute any job.

At any given instant of time, say t, the number of jobs either waiting to be executed or in execution at resource I from class K is denoted by $J_{IK}(t)$. We denote by $\underline{J}(t) \in \Omega$ the vector

$$\underline{J}(t) = [J_{IK}(t), I=1,...,N; K=1,...,C]$$

where the set of feasible states for $\underline{J}(t)$ is denoted by Ω.

Work is scheduled for each resource such that if a server or processor is idle, it will search the list of jobs ready to be run and execute one if at all possible. This means that at any instant of time the number of busy servers or processors at resource $I=1,...,N$ is given by

$$number\ of\ busy\ servers\ at\ node\ I = min[P_I, \sum_{K=1}^{C} J_{IK}(t)] \quad I=1,...,N$$

We assume that each job type requires a constant amount of storage or memory, denoted by M_K for class $K=1,...,C$. This introduces a constraint on the allowable set of states Ω, because only a given number of each type of job can be stored in the system at any one time. For example, if there is a pool of memory of M blocks, then this constraint takes the form

$$\sum_{I=1}^{N} \sum_{K=1}^{C} M_K J_{IK} \leq M$$

while if we dedicate say M_K units of memory to each type of job, then this constraint takes the form

$$\sum_{I=1}^{N} M_K J_{IK} \leq M_K \qquad K=1,...,C$$

4.8.2 Analysis The previous discussion implies that the set of feasible mean throughput rates for each type of job, $\lambda_K, K=1,...,C$ forms a convex set. In fact, this convex set is a simplex, with the extreme points yielding the maximum possible mean throughput rate vectors. One way to explore the geometry of this model is to fix the *mix* of job types, say $F_K, K=1,...,C$ is the fraction of jobs of type K, and hence the mean throughput rate for job type K is

$$\lambda_K = \lambda_{total} F_K \qquad K=1,...,C$$

where λ_{total} is a scalar denoting the total mean throughput rate of jobs through the system. In order to find the largest such permissible λ_{total}, we would start at zero and increase λ_{total} until we would violate one of the state space or mean value constraints; the point at which this occurs would be the largest possible λ_{total}.

4.9 A Mean Throughput Rate Upper Bound for Multiple Class Single Resource Models

Now we sketch how to extend the analysis in the previous section that led to an upper bound on mean throughput rate to a system that processes multiple types of jobs, not just one.

4.9.1 Model A computer communication system must execute C classes of jobs. Each job consists of one or more steps, and each step requires a single resource for a mean amount of time. $T_{IK} \geq 0$ denotes the total mean amount of time, summed over all steps, that class K job requires resource $I=1,...,N$.

The system configuration is $P_I, I=1,...,N$ servers or processors for resource I, where $P_I=1,2,3...$ Each resource is fed by a single queue of jobs, i.e., any processor can execute any job.

At any given instant of time, say t, the number of jobs either waiting to be executed or in execution at resource I from class K is denoted by $J_{IK}(t)$. We denote by $\underline{J}(t) \in \Omega$ the vector

$$\underline{J}(t) = [J_{IK}(t), I=1,...,N; K=1,...,C]$$

where the set of feasible states for $\underline{J}(t)$ is denoted by Ω.

Work is scheduled for each resource such that if a server or processor is idle, it will search the list of jobs ready to be run and execute one if at all possible. This means that at any instant of time the

number of busy servers or processors at resource $I=1,...,N$ is given by

$$number\ of\ busy\ servers\ at\ node\ I = min[P_I, \sum_{K=1}^{C} J_{IK}(t)] \quad I=1,...,N$$

We assume that each job type requires a constant amount of storage or memory, denoted by M_K for class $K=1,...,C$. This introduces a constraint on the allowable set of states Ω, because only a given number of each type of job can be stored in the system at any one time. For example, if there is a pool of memory of M blocks, then this constraint takes the form

$$\sum_{I=1}^{N}\sum_{K=1}^{C} M_K J_{IK} \leq M$$

while if we dedicate say M_K units of memory to each type of job, then this constraint takes the form

$$\sum_{I=1}^{N} M_K J_{IK} \leq M_K \qquad K=1,...,C$$

4.9.2 Analysis The previous discussion implies that the set of feasible mean throughput rates for each type of job, $\lambda_K, K=1,...,C$ forms a convex set. In fact, this convex set is a simplex, with the extreme points yielding the maximum possible mean throughput rate vectors. One way to explore the geometry of this model is to fix the *mix* of job types, say $F_K, K=1,...,C$ is the fraction of jobs of type K, and hence the mean throughput rate for job type K is

$$\lambda_K = \lambda_{total} F_K \qquad K=1,...,C$$

where λ_{total} is a scalar denoting the total mean throughput rate of jobs through the system. In order to find the largest such permissible λ_{total}, we would start at zero and increase λ_{total} until we would violate one of the state space or mean value constraints; the point at which this occurs would be the largest possible λ_{total}.

4.10 Voice Message Mail and Electronic Mail Storage System

A digital system is designed to store two different types of messages

Figure 4.14.Integrated Communication Storage Subsystem Block Diagram

- Voice messages originate at a rate of λ_V messages per unit time, are stored for an average of T_V time units before being removed, and require B_V bytes of storage
- Electronic mail messages originate at a rate of λ_E messages per unit time, are stored for an average of T_E time units, and require B_E bytes of storage

The storage subsystem has a capacity of M bytes. What evidence is there that the storage is adequate?

We denote by $m(t)$ the number of bytes of storage filled with either a voice or electronic mail message at time t. If we ask what the mean amount of occupied storage is, we see

$$mean\ storage = \lim_{T\to\infty} \frac{1}{T}\int_0^T m(t)dt$$

Over an interval of duration T we will find N messages total of either type in the storage subsystem. We denote by $B(K)$ the amount of storage (in bytes) of the Kth message, and $T(K)$ denotes the storage time of the Kth message. Given the above, we see

$$\frac{1}{T}\int_0^T m(t)dt = \frac{1}{T}\sum_{K=1}^{N} B(K)T(K) = \frac{N}{T}\ \frac{1}{N}\sum_{K=1}^{N} B(K)T(K)$$

We identify the total mean throughput rate of messages over the observation interval with N/T and we identify the other term with the mean number of byte-seconds per message:

$$\lambda = \frac{N}{T} \qquad mean\ byte{-}seconds\ per\ message = \frac{1}{N}\sum_{K=1}^{N} B(K)T(K)$$

However, we know that the total mean throughput rate equals the sum of the two types of messages:

$$\lambda = \lambda_V + \lambda_E$$

Next, we realize the mean byte-seconds per message is given by the fraction of arrivals of each type weighted by the mean byte-seconds per each message type:

$$mean\ byte\ seconds\ per\ message = F_V B_V T_V + F_E B_E T_E$$

The fraction of messages of each is simply the fraction of arrivals of each type:

$$F_V = \frac{\lambda_V}{\lambda_V + \lambda_E} \qquad F_E = \frac{\lambda_E}{\lambda_E + \lambda_E}$$

Combining all this, we see

$$M \geq \frac{1}{T}\int_0^T m(t)dt = \lambda_V B_V T_V + \lambda_E B_E T_E$$

Notice we have generalized Little's Law here: the jumps up and down in $m(t)$ are no longer all of size one, but equal the amount of storage associated with each arrival or departure. As in the previous case, we might wish to choose M larger than this mean value due to fluctuations about the mean.

EXERCISE: Plot upper and lower bounds on mean throughput rate and mean delay for this system as a function of total mean arrival rate for a given mix of voice and data arrivals.

4.11 Multiple Processor/Multiple Memory Systems

A hardware block diagram of a multiple resource system is shown in the figure below.

Records are retrieved from auxiliary storage, processed, and stored. The system consists of one processor, and main storage capable of holding R records simultaneously. Main storage is connected to P peripheral processors. Each peripheral processor is connected to a switching fabric, as are all the auxiliary storage devices. Each peripheral processor can be connected to any of A auxiliary storage devices via the switching fabric, but only one peripheral processor can access one auxiliary storage device at a time. Each access to an auxiliary storage device retrieves one block of data; each record comprises an integral number of blocks. For simplicity, we assume that all records are one block in size from this point on.

The steps involved in processing a transaction with the associated resources are summarized in the table below:

Figure 4.15.Hardware Block Diagram of Record Processing System

Table 4.1.Step Resource Table

Step	*Processor*	*Buffer*	*Peripheral*	*Aux Storage*	*Time*
Input	0	1	1	1	T_{input}
Processing	1	1	0	0	T_{proc}
Output	0	1	1	1	T_{output}

The time required for input or output is the time required to access or transfer the data from auxiliary storage, transfer it to or from a buffer, with the associated time required for controlling these actions via the peripheral processor; this total mean time is denoted by T_{input} and T_{output}, for input and output respectively. The mean time involved in record processing on the processor is T_{proc}.

Since a job can be in one of three steps, we denote by $\underline{J}$ the three tuple whose elements denote the number of jobs in execution in each step:

$$\underline{J} = (J_{input}, J_{proc}, J_{output})$$

How do the finite resources limit the admissible state space? Since we have only one processor, we see

$$J_{proc} \leq 1$$

Since we have P peripheral processors with A auxiliary storage devices, the lesser of these two will limit the total number of transactions involved in input or output:

$$J_{input} + J_{output} \leq \min(P, A)$$

Finally, we have a total of R records, and hence

$$J_{input} + J_{proc} + J_{output} \leq R$$

The state space Ω is the admissible set of *integer* valued components of $\underline{J}$ that satisfy these constraints. The *convex hull* of the state space Ω, denoted $C(\Omega)$, is the set of *real* valued three tuples that satisfy these constraints. If we substitute into these constraints, if the processor is the bottleneck, then an upper bound on mean throughput rate λ is given by

$$\lambda T_{proc} \leq 1$$

On the other hand, if the peripheral processor or auxiliary storage is the bottleneck, then an upper bound on mean throughput rate λ is given by

$$\lambda(T_{input} + T_{output}) \leq min(P, A)$$

$$\lambda(T_{input} + T_{proc} + T_{output}) \leq R \qquad buffer\ bottleneck$$

Summarizing all this, we see

$$\lambda \leq \min\left[\frac{1}{T_{proc}}, \frac{min(P, A)}{T_{input} + T_{output}}, \frac{R}{T_{input} + T_{proc} + T_{output}}\right]$$

Four configurations are under investigation

- Main memory capacity of one record, with one peripheral processor and one auxiliary storage unit resulting in no concurrency between the processing and input/output
- Main memory capacity of two records, with one peripheral processor and one auxiliary storage unit with the potential for processor/input or processor/output concurrency
- Main memory capacity of two records, with two peripheral processors and two auxiliary storage units, with the potential for processor/input, processor/output, or processor/input-output concurrency
- Main memory capacity of three records, with two peripheral processors and two auxiliary storage units, and the potential for input/output, processor/input, processor/output concurrency

For each configuration, our goal is to calculate an upper bound on mean throughput rate of completing transactions, and to describe the set of parameter values that achieves this upper bound on mean throughput rate.

For the first configuration, we see

$$\lambda \leq \frac{1}{T_{input} + T_{output} + T_{proc}}$$

For the second configuration, we see

$$\lambda \leq \frac{1}{max[T_{proc}, T_{input} + T_{output}]}$$

For the third configuration, we see

$$\lambda \leq \frac{1}{\frac{1}{2}[T_{proc} + max(T_{proc}, T_{input} + T_{output})]}$$

For the fourth configuration, we see

$$\lambda \leq \frac{1}{max[T_{proc}, \frac{1}{2}(T_{input} + T_{output})]}$$

EXERCISE: Show that the third configuration is least sensitive to the workload assumptions.

4.12 Transaction Processing System

In a transaction processing, clerks spend a given mean amount of time reading, thinking and typing a transaction via a terminal and then wait for the system to respond before repeating this process. Transaction processing involves:

- Input data validation and logical consistency checking
- Data base management access to modify records appropriately

4.12.1 System Configuration The system hardware configuration consists of C clerks at terminals, one terminal per clerk, plus a single processor and a single disk.

4.12.2 Steps Required Per Transaction The steps and resources required for each step of transaction processing are summarized below:

Table 4.2.Resources Required Per Step of Transaction Processing

Step	*Clerk*	*Processor*	*Disk*	*Mean Time*
1	Yes	No	No	T_{think}
2	No	Yes	No	T_{input}
3	No	Yes	No	$T_{proc,dbm}$
4	No	No	Yes	$T_{disk,dbm}$

All the steps involved in transaction processing that require the data base manager have been aggregated into two steps, with one step involving only the processor, and the other step involving only the disk.

4.12.3 System State Space The state at a given instant of time, say t, is given by a tuple denoted by $\underline{J}(t)$ where

- $J_{clerk}(t)$ denotes the number of clerks busy with reading, thinking and typing transactions
- $J_{input}(t)$ denotes the number of jobs either waiting for the processor or running on the processor
- $J_{proc,dbm}(t)$ denotes the number of jobs requiring data base management activity either waiting or running on the processor
- $J_{disk,dbm}(t)$ denotes the number of jobs requiring data base management activity either waiting or running on the disk

The system state space is given by Ω which is the set of all admissible $\underline{J}(t)$ tuples:

$$\Omega = \{\underline{J}(t) \mid J_{clerk}(t), J_{input}(t), J_{proc,dbm}(t), J_{disk,dbm}(t) = 0,1,2...\}$$

We wish to assess the impact of two different modes of operation, involving different structuring of application software:

- In the first mode, a clerk waits for the transaction submitted to be completely executed before entering the next transaction. This results in a constraint on the admissible values of $\underline{J}(t)$, because the total number of transactions *anywhere* in the system at any given instant of time is given by the total number of clerks:

$$\Omega\ \{\underline{J}(t) \mid J_{clerk}(t) + J_{input}(t) + J_{proc,dbm}(t) + J_{disk,dbm}(t) = C\}$$

- In the second mode, a clerk waits for the transaction data validation stage to be completed, before entering the next transaction; a clerk can have a maximum number of transactions, say S transactions, in the data validation phase of system operation at any one time. If this threshold is exceeded, then that clerk is blocked from submitting a new transaction until less than S transactions are in the input validation stage. In practice, S would be chosen to be no impediment on a clerk under normal operations, while under heavy loading, when the system performance becomes unacceptable, S would be a throttle or limit on the maximum amount of work that could ever be in the system, a type of overload control. The design problem is to find an appropriate value of S. This results in a different type of state space constraint:

$$\Omega = \{\underline{J}(t) \mid J_{clerk}(t) + J_{input}(t) \leq S\}$$

4.12.4 Mean Resource Utilization Transactions are executed by the system at a total mean rate of λ transactions per unit time. On the average, we will find the mean number of clerks busy reading, thinking and typing given by

$$E[min(J_{clerk}(t), C)] = \lambda T_{think}$$

The fraction of time the processor is busy is given by

$$E[min(J_{input}(t) + J_{proc,dbm}(t), 1)] = \lambda(T_{input} + T_{proc,dbm})$$

The fraction of time the disk is busy is given by

$$E[min(J_{disk,dbm}(t),1)] = \lambda T_{disk,dbm}$$

4.12.5 Upper Bounds on Mean Throughput Rate To calculate an upper bound on mean throughput, we examine each resource:

- If the clerks are completely busy, then an upper bound on mean throughput rate is

$$C \geq E[min(J_{clerk}(t),C)] = \lambda T_{think} \rightarrow \lambda \leq \frac{C}{T_{think}}$$

- If the processor is completely busy, then an upper bound on mean throughput rate is

$$1 \geq E[min(J_{input}(t)+J_{proc,dbm}(t),1)] = \lambda(T_{input} + T_{proc,dbm}) \rightarrow \lambda \leq \frac{1}{T_{input} + T_{proc,dbm}}$$

- If the disk is completely busy, then an upper bound on mean throughput rate is

$$1 \geq E[min(J_{disk,dbm}(t),1)] = \lambda T_{disk,dbm} \rightarrow \lambda \leq \frac{1}{T_{proc,dbm}}$$

- If the policy of demanding that a clerk wait until a transaction is completely executed until entering a new transaction is the bottleneck, then

$$C = \lambda(T_{think} + T_{input} + T_{proc,dbm} + T_{disk,dbm})$$

  The mean throughput rate is upper bounded by

$$\lambda \leq \frac{C}{T_{think} + T_{input} + T_{proc,dbm} + T_{disk,dbm}}$$

- If the policy of allowing a clerk to enter a maximum of S transactions into the system, with only the data input validation being completed before control is under, is the bottleneck, then

$$SC \geq E[J_{clerk}(t) + J_{input}(t)] = \lambda(T_{think} + T_{input}) \rightarrow \lambda \leq \frac{SC}{T_{think} + T_{input}}$$

4.12.6 Interpretation Let's substitute some numbers to gain insight into what all this means:

Table 4.3.Illustrative Numerical Values

T_{think}	*15 sec*
T_{input}	*0.1 sec*
$T_{proc,dbm}$	*0.5 sec*
$T_{disk,dbm}$	*0.75 sec*
C Clerks	*10*
S Job Limit	*5*

The upper bounds on mean throughput rate are as follows:

- If clerks are the bottleneck then

$$\lambda \leq \frac{10}{15 \text{ sec}} = 0.6667 \text{ jobs/sec}$$

- If the processor is a bottleneck then

$$\lambda \leq \frac{1}{0.1 \text{ sec} + 0.6 \text{ sec}} = 1.4286 \text{ jobs/sec}$$

- If the disk is a bottleneck then

$$\lambda \leq \frac{1}{0.75 \text{ sec}} = 1.3333 \text{ jobs/sec}$$

- If the policy of demanding a clerk wait for a transaction to be completely executed before entering a new transaction is the bottleneck then

$$\lambda \leq \frac{10}{15 \text{ sec} + 0.1 \text{ sec} + 0.6 \text{ sec} + 0.75 \text{ sec}} = 0.6079 \ jobs/sec$$

- If the policy of allowing a clerk to enter a maximum of five (S=5) transactions into the system, with control being returned after data input, is the bottleneck then

$$\lambda \leq \frac{10 \times 5}{15 \text{ sec} + 0.1 \text{ sec}} = 3.3112 \ jobs/sec$$

Demanding that a clerk wait for a transaction to be completely processed before entering a new transaction results in the mean throughput rate being upper bounded by

$$\lambda \leq 0.6079 \ jobs/sec$$

Here, the bottleneck is the work scheduling policy.

Allowing a clerk to immediately enter a transaction after the data input and validation phase is completed, up to a maximum of five, results in the mean throughput rate being upper bounded by

$$\lambda \leq 0.6667 \ jobs/sec$$

We have changed the bottleneck from the policy of work scheduling to the clerks as the limit on maximum mean throughput rate.

If we put twenty clerks on the system (C=20) rather than ten clerks (C=10), then after substituting into the above formulae we find the policy of demanding a clerk wait for a transaction to be completely executed before allowing a new transaction to enter the system results in a maximum mean throughput rate of

$$\lambda \leq 1.2160 \ jobs/sec$$

while if we allow each clerk to enter a maximum of five (S=5) transactions into the system, with control being returned after the data input phase, then the bottleneck is now the clerk's:

$$\lambda \leq 1.3333 \ jobs/sec$$

Finally, if we put thirty clerks on the system (C=30) then we find that the disk is the bottleneck, not the scheduling policy or number of clerks.

4.13 A Distributed Data Communications System

A communications system receives messages from two sources labeled A and B, and transmits them to a common source, labeled C. The system configuration consists of four subsystems, each with its own processor and communications facilities, labeled *1,2,3,4* respectively. A block diagram is shown below.

Figure 4.16.Distributed Data Communications System Block Diagram

Conceptually each processor receives messages from a source, and copies them into output buffers. A queueing network block diagram of the system is shown below:

Figure 4.17.Queueing Network Block Diagram

The operation of the system is as follows:

- Processor 1 receives messages from external source A and buffers them into an internal buffer; it then copies the messages into a set of buffers shared with processor 2. In order that no records be lost in copying because a buffer is not available, critical region code governed by two semaphores *P* and *V* is used.

  The pseudo code executed by processor 1 is shown below:

```
initialize records(1)=buffer(1),space(1)=0 /*B1,max =buffer(1)*/

process 1
loop
        P(records(1)) /*test if records(1)>0;
         yes->decrement by one; no->wait*/
        P(space(2)) /*test if space(2)>0;
        yes->decrement by one; no->wait*/
        copy(space(1),space(2)) /*copy contents of space(1)
         into space(2)*/
        V(records(2)) /*increment records(2) by one*/
end loop
```

- Processor 2 receives messages from processor 1 and copies them into a set of buffers shared with processor 3 and processor 4. The pseudo code executed by processor 2 is shown below:

```
initialize records(2)=buffer(2), space(2)=0 /*B2,max =buffer(2)*/

process 2
loop
        P(records(2)) /*test if records(2)>0;
        yes->decrement by one; no->wait*/
        P(space(3)) /*test if space(3)>0;
         yes->decrement by one; no -> wait*/
        copy(space(2),space(3)) /*copy contents of space(2)
         into space(3)*/
        V(records(3)) /*increment records(3) by one*/
end loop
```

- Processor 4 receives messages from external source B and copies them into a set of buffers shared with processor 3 and with processor 2. The pseudo code executed by processor 4 is shown below:

```
initialize records(4)=buffer(4), space(4)=0 /*B4,max =buffer(4)*/

process 4
loop
        P(records(4)) /*test if records(4)>0;
        yes->decrement by one;no->wait*/
        P(space(3)) /*test if space(3)>0;
        yes->decrement by one;no->wait*/
        copy(space(4),space(3)) /*copy contents of space(4)
         into space(3)*/
        V(records(3)) /*increment records(3) by one*/
end loop
```

- Processor 3 receives messages from processors 2 and 4 and copies them into an external buffer shared with external sink C. The pseudo code executed by processor 3 is shown below:

```
initialize records(3)=buffer(3),space(3)=0 /*B3,max =buffer(3)*/

process 3
loop
        P(records(3)) /*test if records(3)>0;
        yes->decrement by one;no->wait*/
        P(space(5)) /*test if space(5)>0;
        yes->decrement by one;no->wait*/
        copy(space(3),space(5)) /*copy contents of space(3)
         into space(5)*/
        V(records(5)) /*increment records(5) by one*/
end loop
```

The steps and resources required at each step are summarized in the table below:

Table 4.4.Resources Required for Each Step of A->C Transmission

Step	*Processor*			*Buffer*			*Time*
Number	*1*	*2*	*3*	*1*	*2*	*C*	*Interval*
1	Yes	No	No	Yes	No	No	T_1
2	No	Yes	No	Yes	Yes	No	T_2
3	No	No	Yes	No	Yes	Yes	T_3

Table 4.5.Resources Required for Each Step of B->C Transmission

Step	*Processor*		*Buffer*		*Time*
Number	*3*	*4*	*2*	*C*	*Interval*
1	No	Yes	Yes	No	T_4
2	Yes	No	Yes	No	T_3

4.13.1 System State Space What is the state of this system at any given time instant, say *t?* We denote by $M_K(t),K=1,2,3,4,C$ the number of messages either waiting to be transmitted or being transmitted at station K at time t, while we let $B_K(t),K=1,2,3,4,C$ denote the number of buffers filled with a message at node K. The stage space Ω is given by

$$\Omega = \{M_K(t)=0,1,2,...;B_K(t)=0,1,...,B_{max,K};K=1,2,3,4,C\}$$

4.13.2 Long Term Time Averaged Utilization of Resources The fraction of time we will find processor K busy, averaged over a suitably long time interval (we will denote the average by E()) is simply the mean arrival rate of messages to processor K, denoted by λ_K, multiplied by the mean time spent copying a message, denoted by T_K:

$$E[min(M_K(t),1)] = \lambda_K T_K \qquad K=1,2,3,4,5$$

The mean number of buffers we will find filled with a message at node K, averaged over a suitably long time interval, is the mean arrival rate of messages to the buffer pool, denoted by $\tilde{\lambda}_K$, multiplied by the mean time spent copying a message into the buffer and out of the buffer, denoted by $\tilde{T}_K$:

$$E[min(B_K(t),B_{K,max})] = \tilde{\lambda}_K \tilde{T}_K \qquad K=1,2,3,4,C$$

4.13.3 Upper Bounds on Mean Throughput Rate Two types of bottlenecks can arise, one due to the processors becoming completely busy copying messages, the other due to the buffers becoming completely filled. How can we quantify these ideas?

First we focus on processor bottlenecks:

- Processor 1 can be completely utilized:

$$1 \geq E[min(J_1(t),1)] = P\lambda T_1 \qquad \lambda \leq \frac{1}{PT_1}$$

- Processor 2 can be completely utilized:

$$1 \geq E[min(J_2(t),1)] = P\lambda T_2 \qquad \lambda \leq \frac{1}{PT_2}$$

- Processor 4 can be completely utilized:

$$1 \geq E[min(J_4(t),1)] = (1-P)\lambda T_4 \qquad \lambda \leq \frac{1}{(1-P)T_4}$$

- Processor 3 can be completely utilized:

$$1 \geq E[min(J_3(t),1)] = \lambda T_3 \qquad \lambda \leq \frac{1}{T_3}$$

Next, we concentrate on potential buffer bottlenecks:

- The buffer pool connecting processor 1 to processor 2 can become completely filled:

$$B_{1,max} \geq E[min(B_1(t),B_{1,max})] = P\lambda(T_1+T_2) \qquad \lambda \leq \frac{B_{1,max}}{P(T_1+T_2)}$$

- The buffer pool connecting processors 2 and 4 to processor 3 can become completely filled:

$$B_{2,max} \geq E[min(B_2(t),B_{2,max})] = P\lambda(T_2+T_3)+(1-P)\lambda(T_3+T_4)$$

$$\lambda \leq \frac{B_{2,max}}{P(T_2+T_3) + (1-P)(T_3+T_4)}$$

- The buffer pool connecting processor 3 to output sink C can become completely filled:

$$B_{3,max} \geq E[min(B_3(t),B_{3,max})] = \lambda(T_3+T_C) \qquad \lambda \leq \frac{B_{3,max}}{T_3+T_C}$$

The mean throughput rate is upper bounded by finding the minimum of the upper bound on the processor bounds and the minimum of the upper bound on the buffer bounds:

$$\lambda \leq \min[\lambda_{processor,max}\ ,\ \lambda_{buffer,max}]$$

$$\lambda_{processor,max} = \min\left[\frac{1}{PT_1}, \frac{1}{PT_2}, \frac{1}{(1-P)T_4}, \frac{1}{T_3}\right]$$

$$\lambda_{buffer,max} = \min\left[\frac{B_{1,max}}{P(T_1+T_2)}, \frac{B_{2,max}}{P(T_2+T_3)+(1-P)(T_3+T_4)}, \frac{B_{3,max}}{T_3+T_C}\right]$$

4.13.4 Special Cases Two cases are of interest:

- Single buffering messages so $B_{K,max}=1$ which is a test of the logically correct operation of the system. Can you see that the *buffers* will *always* be the bottleneck, and *never* the processors?
- Double buffering messages to allow for concurrent operation of a transmitter and receiver pair, so that $B_{K,max}=2$. Depending upon the time spent in copying messages into and out of shared buffers, either a processor or a buffer pool may be the bottleneck

4.14 Different Data Structures for Multiple Processors and Multiple Memories

In this section we examine the performance of a computer system with multiple processors and multiple memories interconnected via a high speed bus, that must execute one application program that can have two different data structures. >From this point on, we will assume the time the bus is used to transfer bits to and from the processors and memories is negligible compared to the processing time required.

The two different types of data structures are labeled *1* and *2*. To be concrete, we will fix the hardware configuration at seven units of memory and three processors. For data structure 1, four units of memory and (1/19) seconds of execution time are required on one processor, on the average. For data structure 2, two units of memory and (1/10) seconds of execution time are required on one processor. Put differently, data structure 1 uses more memory than data structure 2, but takes less time to execute. The table below summarizes this information:

Table 4.6.Resources Required For Each Data Structure

Type	*Processor*	*Memory*	*Mean Time*
1	1	4	1/19
2	1	2	1/10

The state of the system is given by an ordered pair (J_1,J_2), where $J_K,K=1,2$ is the total number of jobs in execution at a given instant of time using data structure K. Due to the hardware configuration constraints, we see

$$4J_1 + 2J_2 \leq 7 \quad \textit{memory constraint}$$

$$J_1 + J_2 \leq 3 \quad \textit{processor constraint}$$

$$J_K = 0,1,2,...;K=1,2 \quad \textit{nonnegative integers constraint}$$

The state space, Ω, is given by

$$\Omega = \{(J_1,J_2) \mid (0,0),(1,0),(0,1),(1,1),(0,2),(0,3)\}$$

The mean number of jobs in execution, averaged over a suitably long time interval, is denoted by $E(J_K),K=1,2$. The total mean throughput rate of executing jobs is

$$\textit{mean throughput rate} = \lambda = 19E(J_1) + 10E(J_2)$$

Suppose we wish to maximize the mean throughput rate of completing jobs: *which data structure do we choose?*

The fraction of time the system is in state (J_1,J_2), averaged over a suitably long time interval, is denoted by $\pi(J_1,J_2)$. We have six possible states, and we would attempt to maximize the mean throughput rate subject to the constraint

$$\sum_{(J_1,J_2)\in\Omega} \pi(J_1,J_2) = 1 \quad \pi(J_1,J_2)\geq 0 \quad (J_1,J_2)\in\Omega$$

If we do so, by direct substitution we find

$$\lambda \leq 30 \quad \textit{when } J_1 \equiv 0, J_2 \equiv 3$$

At this point, all three processors are completely busy, while we only have six out of seven memory units busy; based on this evidence, one might naively expect that the processors are a *bottleneck*, i.e., the processors are completely utilized but not memory.

As an aside, it is interesting to note that the point (J_1=1,J_2=1) results in a *local* optimum for the mean throughput rate as we explore all of its nearest neighbors, but this is not the *global* optimum for the mean throughput rate. To see this, we evaluate the mean throughput rate of completing jobs at state (J_1=1,J_2=1), and compare this mean throughput rate with that for all its nearest neighbors. Unfortunately, the state (J_1=0,J_2=3) is *not* a nearest neighbor, and this state results in a higher mean throughput rate. This suggests that we might be forced to exhaustively enumerate all system states in much more complicated (realistic) systems, and evaluate the mean throughput rate in order to determine an upper bound on mean throughput rate. The time required to carry out this type of analysis, even with a high speed digital computer, can easily become excessive relative to our willingness to pay for this type of analysis.

On the other hand, what if we allow the values for J_K,K=1,2 to be continuous, not simply integer valued? This would allow us to apply linear programming techniques, where once we find a *local* maximum we are done, because this is also a *global* maximum. If we do so, we find the maximum mean throughput rate is *higher* (we have dropped the constraint that the pairs (J_1,J_2) be integers, so fewer constraints presumably *increase* the maximum mean throughput rate), and in fact is

$$\textit{maximum mean throughput rate} = 37\ 3/4 = \max \lambda \geq \lambda$$

$$E(J_1) = 7/4 \qquad E(J_2) = 0$$

At this maximum, memory is completely utilized, and we have (5/4) processors idle on the average. This suggests that memory is the *bottleneck*, not the *processors*. This is *completely* different from what we just saw.

Notice that we can get *diametrically* opposite answers depending upon our assumptions: using the assumption that the mean number of jobs must be continuous leads to using only data structure 1, not data structure 2, while assuming that the mean number of jobs must be discrete leads to employing data structure 2, not data structure 1.

What happens if we increase memory from seven to eight units? For this case, keeping three processors, we find that the maximum mean throughput rate is the same whether we assume J_K,K=1,2 is continuous or discrete, with

$$\textit{maximum mean throughput rate} = 38 = \max \lambda \geq \lambda$$

which occurs at J_1=2,J_2=0.

Since the maximum mean throughput rate *increases* as we add memory, we would say naively that memory was a bottleneck. On the other hand, we can remove one processor now: we cannot use it! For the seven unit memory configuration, we saw that either processors or memory could be a bottleneck, but adding one unit of memory (to relieve the bottleneck) led to only two thirds processor utilization (i.e., we can get rid of a processor).

4.15 Two Types of Jobs

A single processor computer system must execute two different types of jobs, submitted by two different people. The first person spends a mean amount of time reading, and thinking, and typing, with a mean duration of $T_{think,1}$, and then waits for a response before repeating this process. The second person spends a different amount of time reading, and thinking, and typing, with a mean duration of $T_{think,2}$, and then waits for a response before repeating this process. The mean processing times for a job submitted by person K=1,2 are denoted by $T_{proc,K}$. Our goal is to determine the mean rate of executing jobs submitted by each person.

4.15.1 Analysis The system at any instant of time can be in the following states:

[1] Both operators actively thinking, reading and typing, with no work being processed

[2] Operator one is waiting for a response while operator two is still reading and thinking and typing

[3] Operator two is waiting for a response while operator one is still reading and thinking and typing

[4] Both operators are waiting for a response; the job submitted by operator one is being executed, while the job submitted by operator two is being queued until the processor becomes available

[5] Both operators are waiting for a response; the job submitted by operator two is being executed, while the job submitted by operator one is being queued until the processor becomes available

We will index the states by J=1,2,3,4,5. The system spends a total amount of time T_J in state J during a measurement interval of duration T. The fraction of time the system spends in state J is denoted by $\pi(J)$ where

$$\pi(J) = \frac{T_J}{T} \geq 0 \qquad \sum_{J=1}^{5} \pi(J) = 1$$

The mean throughput rate of executing jobs for person K=1,2 is denoted λ_K. Little's Law allows us to relate the mean throughput rate to the fraction of time spent in each state.

First, the fraction of time clerk one is reading and thinking and typing is

$$\lambda_1 T_{think,1} = \pi(1) + \pi(3)$$

Next, the fraction of time clerk two is reading and thinking and typing is

$$\lambda_2 T_{think,2} = \pi(1) + \pi(2)$$

Next, the fraction of time the processor is busy executing jobs submitted by operator one is given by

$$\lambda_1 T_{proc,1} = \pi(2) + \pi(4)$$

Finally, the fraction of time the processor is busy executing jobs submitted by operator two is given by

$$\lambda_2 T_{proc,2} = \pi(3) + \pi(5)$$

If we add the first and third equations, we see

$$\lambda_1(T_{think,1} + T_{proc,1}) = \pi(1) + \pi(2) + \pi(3) + \pi(4) \leq 1$$

while if we add the second and fourth equations we see

$$\lambda_2(T_{think,2} + T_{proc,2}) = \pi(1) + \pi(2) + \pi(3) + \pi(5) \leq 1$$

On the other hand, if we add the last two equations, we see

$$\lambda_1 T_{proc,1} + \lambda_2 T_{proc,2} = \pi(1) + \pi(2) + \pi(3) + \pi(4) + \pi(5) \leq 1$$

This defines a convex set, with boundaries determined by the upper bounds on λ_1,λ_2.

On the other hand, if we add the first two equations plus twice the third plus twice the fourth equation, we find

$$\lambda_1(T_{think,1} + 2T_{proc,1}) + \lambda_2(T_{think,2} + T_{proc,2}) = 2 + \pi(2) + \pi(3) \geq 2$$

This gives us a lower bound on the admissible mean throughput rate. All of this is summarized in the figure below.

4.15.2 An Alternate Approach An alternative approach to this is to attempt to maximize the total mean throughput rate λ subject to the constraints implied by Little's Law.

4.16 Multiple Class Multiple Resource Mean Value Analysis

We close with an analysis of an upper bound on mean throughput rate for a system processing multiple types of jobs, with each job requiring multiple resources for a mean duration for each step.

4.16.1 Model There are C types of jobs to be executed. A job consists of $S(J)$,J=1,...,C steps. Each step is defined as requiring a *fixed* set of resources for a certain average or mean amount of time.

Figure 4.18.Admissible Mean Throughput Rates

The system resources available are given by a vector $\underline{R}$ which has M components, one for each type of resource, with each component denoting the *number* of resources of each type available:

$$\underline{R} = (R_1,...,R_M) \quad R_K = amount\ of\ resource\ K, K=1,...,M$$

For a given job step of a class $J=1,...,C$ job, say step $I, 1 \le I \le S$, a set of resources is demanded, denoted by $D_{IJ}(K)$, where $K=1,...,M$ denotes the amount of resource K. The mean duration of step I for jobs in class J is denoted by T_{IJ}.

In what follows we let $N_{IJ}(t)$ denote the number of jobs in class J in the system at time t in step $I=1,...,S(J)$.

The system state space is denoted by Ω with elements denoted by $\underline{N}$ with each element denoting the number of type $J=1,...,C$ jobs in step $I=1,...,S(J)$. Not all states are feasible; states must be nonnegative integers, subject to the constraint that the amount of resources of each used at every instant of time must be less than the total available amount of that resource:

$$\Omega = \{N_{I,J} \mid N_{I,J}=0,1,...; 1 \le J \le C; 1 \le I \le S(J);$$
$$\sum_{J=1}^{C} \sum_{I=1}^{S(J)} N_{IJ} D_{IJ}(K) \le R_K, K=1,...,M\}$$

4.16.2 Analysis Little's Law allows us to write that the mean number of jobs in execution equals the mean throughput rate of each job type multiplied by the mean execution time:

$$E[N_{IJ}] = \lambda_{IJ} T_{IJ} \quad 1 \le J \le C; 1 \le I \le S(J)$$

Rather than deal with the individual mean arrival rates of each type of job, we will denote by F_{IJ} the fraction of jobs of type I,J while λ denotes the *total* mean throughput rate of jobs:

$$\lambda F_{IJ} = \lambda_{IJ} \quad 1 \le J \le C; 1 \le I \le S(J)$$

The system will spend a given fraction of time $\pi(\underline{N})$ in each state $\underline{N} \in \Omega$.

Which states will maximize the total mean throughput rate for a fixed set of resources over all possible job mixes F_{IJ}? Substituting into the above, it follows that

$$\frac{1}{\lambda} = \frac{F_{IJ} T_{IJ}}{\sum_{\underline{N} \in \Omega} \pi(\underline{N}) N_{IJ}} \quad 1 \le J \le C; 1 \le I \le S(J)$$

must be maximized over all F_{IJ}:

$$F_{IJ} \geq 0 \qquad \sum_{J} F_{IJ} = 1$$

and subject to the state space constraint on $\underline{N}$.

If resource K is a bottleneck, then

$$\lambda \leq \max_{K} \frac{R_K}{D_{IJ} F_{IJ} T_{IJ}}$$

Let $U_K(t), K=1,...,M$ denote the total amount of type K resources allocated at time t so that it follows that

$$E[U_K(0,T)] = \frac{1}{T} \int_0^T U_K(t)dt = \textit{time averaged value of } U_K(t) \textit{ in } (0,T) \quad K=1,...,M$$

Over the time interval of duration T a total of N_J jobs of type $J=1,..,C$ are executed, so that

$$E[U_K(0,T)] = \sum_{L=1}^{N_J} F_{KJ} D_{KJ} T_{KJ} = \frac{N_J}{T} \frac{1}{N_J} \sum_{L=1}^{N_J} F_{KJ} D_{KJ} T_{KJ}$$

$$= \lambda_J E[D_{KJ} T_{KJ}] \qquad K=1,...,M; J=1,...,C$$

Put differently, an upper bound on the mean throughput rate is given by

$$\lambda_J = \frac{E[U_K(0,T)]}{E[D_{KJ} T_{KJ}]}$$

The numerator is the utilization of resource K, while the denominator is the cross product or cross correlation of the demand for resource K by job type J with the holding time of resource K by job type J. To maximize the mean throughput rate for job type J, the utilization must be increased, while the cross correlation must be decreased.

Problems

1) A computer system consists of N clerks at terminals. Each clerk spends a constant amount of time T_{think} reading and typing and thinking to submit a job. The system executes one job at a time, and each job requires T_{system} time units to be completely executed.

A. Assume all clerks begin to read and think and type at time zero. For N=1,2,3 draw time lines of the activity of each clerk through three job submission cycles.

B. What is the mean throughput rate for executing jobs beyond the first job submission cycle? What is the mean response time each clerk experiences beyond the first job submission cycle?

C. If T_{think}=15 sec and T_{system}=1 sec, what is the breakpoint number of clerks? What is the mean response time for N=10 and for N=20? What is the mean throughput rate for N=10 and N=20?

2) An interactive transaction processing system handles one type of transaction. A clerk submits transactions and waits for the system to respond before submitting the next transaction. There is a single processor that must execute each transaction. The following data were reported during a four hour time interval of operation:

- Six clerks were signed on during the measurement interval
- The mean *think* time for a clerk was twenty (20) seconds
- $F(K)$ denotes the fraction of time that K clerks are simultaneously waiting for a transaction to complete, and is summarized below

Table 4.7.Number of Clerks vs F(K)

K	*0*	*1*	*2*	*3*	*4*	*5*	*6*
F(K)	0.40	0.25	0.15	0.10	0.05	0.03	0.02

Estimate the mean transaction processing throughput rate and the mean response time for a transaction.

3) A widget retailer wishes to purchase an online point of sale computer communications system which will

- Check the credit at the point of sale of each potential customer
- Update and control inventory at the point of sale

The retailer wishes to purchase a system capable of handling one transaction per minute per terminal, with a total of two hundred (200) terminals attached to the system. The hardware configuration that the retailer can afford consists of one processor and one disk controller for a single spindle. Each transaction goes through the following steps:

[1] At the start of each transaction, a log file is updated to show that a transaction has entered the system; this is useful for recovery and accounting

[2] A credit check is made by first retrieving a regional customer index and then an accounts receivable index to find the location of the customer accounts receivable file

[3] An application program processes the credit check information and sends an approval or disapproval signal to the point of sale terminal originating the transaction; from this point on we assume the fraction of transactions that are not approved is negligible.

[4] An application inventory control program first accesses a model index and then a color index to determine if the widget is in stock. >From this point on we assume all but a negligible fraction of items are in stock.

[5] An application program generates an order for the appropriate item and has it spooled for subsequent printing at the appropriate warehouse.

[6] An application program updates an inventory file.

[7] An audit trail file is written.

[8] A log file is written showing the final status of the transaction. This is useful for recovery and accounting.

The total mean time per disk access is fifty milliseconds. The total mean time the processor is busy per transaction is two hundred fifty milliseconds.

Is this design feasible? Will the system meet its' performance goals? You might wish to structure your answer as follows:

A. Make a table showing the resources consumed at each step of execution. What is the total mean amount of each type of resource consumed per transaction?

B. What are the bottlenecks in the system? What is the maximum mean throughput rate of each bottleneck?

C. Two operating systems are under investigation. One operates in a mode of executing one transaction at a time from start to finish, while the second allows concurrent processor and disk activity. What is the maximum number of terminals the system can support for either operating system?

D. What if the system is configured with two disk spindles, not just one?

4) An interactive transaction processing system handles one type of transaction. A clerk submits transactions and waits for the system to respond before submitting the next transaction. The clerical reading and thinking and typing time interval has a mean of thirty (30) seconds. There is a single processor and a single input/output controller for one or more disks. A transaction involves some processing followed by some input/output followed by some processing and so forth until completion. For each configuration below we assume the software (application code and operating system) is unchanged, and we wish to investigate the performance of various hardware configurations on total system traffic handling characteristics. We assume each transaction requires twenty (20) accesses to secondary disk storage. For simplicity, assume the disk controller requires zero time to transfer data to and from disks.

- For each configuration below, clearly state what hardware resource reaches complete utilization first (this is called the *bottleneck* resource) as the number of clerks is increased toward infinity.
- For each configuration below, plot upper and lower bounds associated with the mean throughput rate of completing transactions and the mean response time versus the number of clerks on the system; clearly label each plot.

Here are the different configurations under consideration:

A. The hardware consists of a slow processor with a single slow speed disk. The total mean processor time per transaction is 2.0 seconds. The slow speed disk requires seventy (70) milliseconds to execute one disk access to one block of data.

B. The hardware consists of a slow processor with a special purpose *cache* memory attached, and a single slow speed disk. The total mean processor time per transaction is reduced to 1.25 seconds due to the attached cache.

C. The hardware consists of a slow speed processor with a cache memory, and a single medium speed disk. The medium speed disk requires fifty (50) milliseconds to execute one disk access to one block of data.

D. The hardware consists of a high speed processor with a single medium speed disk. The total mean processor time per transaction is 0.60 seconds.

E. The hardware consists of a high speed processor with either two medium speed disks or a single high speed disk. The high speed disk requires thirty (30) milliseconds to execute one disk access to one block of data.

5) An online transaction processing consists of hardware, a set of application programs, and an operating system that manages resources. The hardware configuration consists of a single processor, a disk controller that controls half duplex communication between the processor and disks, and three disk spindles. The steps involved in transaction processing are

[1] Wait for the disk arm on the requisite spindle to become available

[2] Initiate a head seek to the proper disk cylinder

[3] Wait for the channel to become available

[4] Wait for the proper block to rotate under the head

[5] Read from one file on one disk spindle and release the channel

[6] Process the transaction in the processor

[7] Complete the transaction by following one of the following two branches

A. Update the file and unlock the disk spindle

B. Do not update the file and unlock the disk spindle

The following information is available

- Six (6) milliseconds is required for every read or write transfer over the half duplex communication network arbitrated by the disk controller.
- Twenty five per cent (25%) of the file accesses are to the first disk spindle, forty per cent (40%) of the file accesses are to the second disk spindle, and thirty five per cent (35%) of the file accesses are to the third disk spindle.
- Each disk access involves time to move the head to the correct cylinder, called *seek* time, followed by time for the correct file to rotate around to the head, called *rotational latency time.* The seek time is uniformly distributed from forty (40) to one hundred twenty (120) milliseconds. The rotation time is uniformly distributed from zero (0) to thirty four (34) milliseconds.
- The processing time of a transaction in the single processor is uniformly distributed from four (4) to fourteen (14) milliseconds.
- Seventy five per cent (75%) of the transactions require a file update.

Answer the following questions:

A. What is the step resource table for transaction execution?

B. What are upper bounds on the mean throughput rate of executing transactions for each resource? What are the bottlenecks?

C. The degree of multiprogramming is defined as the mean number of jobs in execution at any one instant in time. What is the maximum degree of multiprogramming for this system?

D. What if the disk accesses are equally balanced among all the spindles? What if all the disk accesses are intended for one spindle?

6) Here is a model of a *link level flow control* protocol for exchanging information from a transmitter to a receiver. A data link consists of a transmitter, a receiver, a full duplex noiseless channel (consisting of two one way channels, one from the transmitter to the receiver, and one from the receiver to the transmitter) and a limited amount of memory for buffering messages at the receiver. After initialization procedures, the transmitter can send at most B unacknowledged messages. Every time the transmitter sends a message, it decrements a counter which is initialized at B by one, and if this counter is at zero the transmitter can send no messages. Every time the transmitter receives an acknowledgement, it increments this same counter by one, up to a maximum value of B. The maximum value B associated with this counter is sometimes called a *window* because it portrays the maximum number of unacknowledged messages streaming through the channel at any given instant of time, i.e., it is a window on the message stream. Here are the steps involved in message transmission:

[1] The transmitter decrements its buffer counter by one, hence reserving a buffer at the receiver, and transmits the message; this step requires a mean T_T. If all available buffers are reserved, i.e., if the counter is at zero, the transmitter waits until one becomes available.

[2] The message is transmitted over the link from the transmitter to the receiver; this step requires a mean time T_{T-R} and only involves the time for a message to propagate from the transmitter to the receiver.

[3] The receiver processes the message, empties the buffer and sends an acknowledgement back; this step requires a mean time T_R.

[4] The message is transmitted over the link from the receiver to the transmitter; this step requires a mean time T_{R-T} and only involves the time for a message to propagate from the receiver to the transmitter.

[5] The transmitter processes the acknowledgement; this step requires a mean time T_A and the transmitter marks the appropriate message buffer available or free for new messages, i.e., the counter is incremented by one.

A. Make a table showing the resources required at each step of message transmission and the mean time to hold these resources.

B. Find an upper and lower bound on the mean throughput rate of successfully transmitting messages over this system, as a function of the number of buffers B at the receiver, and other model parameters. Plot these upper and lower bounds vs B assuming $T_T=T_R$ and $T_{T-R}=T_{R-T}$ for three cases: $T_T=T_{T-R}/10$, $T_T=T_{T-R}$, and $T_T=10T_{T-R}$.

C. Suppose that $T_{T-R}=T_{R-T}=0$. What is the change in the upper and lower bounds on mean throughput rate in going from $B=1$ to $B=2$ to $B=\infty$ for $T_T=T_R$? Repeat the above but now assume $T_R=10T_T$.

7) A transaction processing system retrieves a record from one disk spindle, processes it, and stores it back on a second disk spindle. The hardware configuration consists of a central processor connected via a switch to multiple direct memory access disk controllers, with one spindle or moving head disk per controller. The processor has a limited amount of main memory (so called buffers) for storing records. S denotes the number of system buffers. Each buffer can hold one record. The total mean time to retrieve a record is T_{input}, the total mean time to process a record is T_{proc}, and the total mean time to store a record is T_{output}.

A. Make a table showing the resources used at each step of transaction processing and for what mean amount of time.

B. What is the state space for this sytem?

C. If $T_{proc}=1$, $T_{input}=T_{output}=2$, what is the maximum mean throughput rate for $S=1$ versus $S=2$ versus $S=3$?

D. Repeat the above for $T_{proc}=5$, $T_{input}=T_{output}=1$, and $S=1$ versus $S=2$ versus $S=3$?

E. Repeat the above for $T_{proc}=T_{input}=T_{output}=1$ for $S=1$ versus $S=2$ versus $S=3$?

F. The mean time to execute a record is $T_{input}+T_{proc}+T_{output}$. With no concurrency, the mean throughput rate is the reciprocal of this:

$$mean\ throughput\ rate_{no\ concurrency} = \frac{1}{T_{input}+T_{proc}+T_{output}}$$

Compute the reciprocal of the maximum mean throughput rate for each of the above parts, normalized by the mean throughput rate with no concurrency (this displays the gain due to concurrency).

G. Compute the degree of multiprogramming for each of the above configurations, where this is defined as the mean number of tasks simultaneously in execution, assuming the system is executing work at its maximum mean throughput rate.

8) A transaction processing system retrieves a record from storage on one disk spindle, processes it, and stores it on a second disk spindle. The hardware configuration consists of a single central processor connected via a switch to two direct memory access disk controllers, with one spindle for each controller; the direct memory access capability allows the central processor to execute work while the disk controller is busy accessing a record. All of the text required to process records is resident in main memory. The processor can buffer at most S blocks of data at a time. Each disk accesses retrieves or stores one block at a time. N denotes the number of records per block. Each record consists of B bytes. T_{access} denotes the mean time to move the disk head from an arbitrary point to the start of a block. $T_{transfer}$ denotes the mean time to transfer one byte of data from an input/output device to a disk controller. T_{proc} denotes the mean processor time per record.

A. What are the steps and resources held and time to execute each step of each job?

B. Find an expression in terms of model parameters and plot it for the maximum mean throughput rate (records per unit time) versus the number of records per block.

C. Find the ratio of the reciprocal of the maximum mean throughput rate divided by the mean time to process a record from start to finish (this is the reciprocal of the mean throughput rate with no concurrency).

D. What are the bottlenecks in this system?

E. Suppose that the following numbers specify the model parameters:

Table 4.8.Model Parameters

Disk Access Time	T_{access}	50 msec
Disk Transfer Time	$T_{transfer}$	1 microsecond
Buffer Size	B	4096 bytes
Processor Time/Record	T_{proc}	30 msec
Processor Buffers	S	8

For $N=1,2,4,8$ what is the numerical value of the maximum mean throughput rate in records per unit time?

9) Two types of jobs arrive at random instants to be executed by a computer system. The first type of job requires one processor and one memory partition, and will hold these two resources for a mean time of T_{small} to complete its execution with no interruptions. The second type of job requires one processor and two memory partitions, and will hold these resources for a mean time of T_{big} to complete its execution, again with no interruptions. The system consists of P processors and M memory partitions connected by a switching system. The time involved with data transfers through the switching system is

assumed to be negligible compared to the time associated with holding a processor and one or more memory partitions.

A. At any given instant of time, there are $J_{small}(t)$ small jobs and $J_{big}(t)$ big jobs in execution in the system. Find the set of feasible pairs $(J_{small}(t), J_{big}(t))$ for this system, which defines the *state space* of operations.

B. We denote by λ_{small} and λ_{big} the mean throughput rate of each type of job. Find the set of feasible mean throughput rates $(\lambda_{small}, \lambda_{big})$. Clearly label all set boundaries in terms of model parameters.

C. List all the potential bottlenecks. For each bottleneck in the system, what is an upper bound on its mean throughput rate?

D. We now fix the *mix* or *fraction* of jobs executed of each type, denoted by F_{small}, F_{big} respectively, where

$$0 \le F_{small} \le 1 \quad 0 \le F_{big} \le 1 \quad F_{small} + F_{big} = 1$$

The mean or average time to process a job is denoted by $T_{average}$ and is given by

$$T_{average} = F_{small} T_{small} + F_{big} T_{big}$$

The mean arrival rate for each type of job is given by

$$\lambda_{small} = \lambda F_{small} \qquad \lambda_{big} = \lambda F_{big}$$

where λ is the *total* mean throughput rate of executing jobs. Find upper and lower bounds on the total mean throughput rate λ as a function of model parameters for the following cases

[1] Two processors $P=2$ and two memory partitions $M=2$

[2] Two processors $P=2$ and three memory partitions $M=3$

[3] Three processors $P=3$ and three memory partitions $M=3$

For each case, compare your calculation with the maximum mean throughput rate for a system that has a *processor* bottleneck, i.e., where each processor can execute one job every $T_{average}$ seconds on the average, and memory is *not* a bottleneck. What interactions arise between the processor and memory?

10) We consider an abstraction of a communications system consisting of a transmitter (or producer) and receiver (or consumer). The transmitter generates a record, stores it in an internal transmitter buffer, and tests to see if a buffer from a pool of buffers shared by the transmitter and receiver is available. If a buffer is not available, it waits until a buffer is available, and stores the record in a buffer. The transmitter increments the number of records outstanding, waiting to be handled by the receiver, by one, and repeats the entire process. The receiver tests to see if a record is waiting to be read; if not, the receiver waits until a record is available, and removes it from its buffer in the shared buffer pool into an internal receiver buffer. The receiver then decrements the number of records outstanding, and processes it, and repeats the entire process. There are two processes each in their own physical processor, one for the producer and one for the consumer, plus a pool of buffers of size B records, i.e., one record can fit into one buffer. In order to insure that no records are lost due to a speed mismatch between the producer and consumer, there is a semaphore with *P* and *V* primitives. The *P()* primitive involves testing its argument to see if it is positive; if it is, access to the associated field is locked, and if it is not, wait until it becomes nonzero. The *V()* primitive involves testing its argument to see if it is nonzero; if it is, the argument is decremented by one, and access is allowed to the associated array; if it is not, wait. The pseudo code (with comments) shown below gives a succinct description of system operation:

*initialize space=B, records=0 /*initialize empty buffers=B,ready records=0*/*

```
process producer
loop
 generate record
 P(space)   /*test to see if space>0; if so, decrement space by 1; if not wait*/
 deposit record  /*deposit record in available space*/
 V(records)  /*increment number of records by one*/
end loop{producer}

process consumer
loop
 P(records)  /*test to see if records>0; if so, decrement records by 1; if not wait*/
 read record
 V(space)  /*increment amount of space by one*/
 process record
end loop{consumer}
```

The mean time to execute each of these actions without contention on the respective processors is as follows:

- $T_{generate}$ --mean time to generate a record
- $T_{deposit}$ --mean time to deposit a record in a buffer
- T_{read} --mean time to read a record
- $T_{process}$ --mean time to process a record

We assume that the P and V primitives require zero processor time to be executed.

A. Make a table showing the resources required at each step.

B. The pair $J_{producer}(t), J_{consumer}(t)$ denote the number of messages at the producer and consumer respectively at an arbitrary time t. Find the set of admissible values for these pairs, which is called the state space of this system.

C. What are potential bottlenecks limiting the maximum mean throughput rate?

D. What is the maximum mean throughput rate as a function of model parameters?

11) An online transaction processing consists of hardware, a set of application programs, and an operating system that manages resources. The hardware can be configured in two ways

[1] High performance configuration--One high speed processor, disk controller, and high speed disk at a total system cost of two hundred thousand dollars

[2] Low performance configuration--One low speed processor, disk controller, and low speed disk at a total system cost of one hundred thousand dollars

Clerks enter transactions into the system and wait for the system to respond before entering the next transaction. Each clerk is paid a total salary of twenty five thousand dollars per year, fully burdened to reflect salary, benefits, and general and administrative overhead.

The steps involved in a transaction are as follows

[1] A clerk reads an order form, and types this into the system from a terminal; this step has a mean time of thirty seconds

[2] A front end screen manager checks the data and formats it for the next step of processing; this step has a mean time of T_{fe}

[3] A scheduler logs the transaction and passes it onto the next step; this step has a mean time of T_{sched}

[4] A data base manager accepts the transaction and the data base is modified accordingly; this step has a mean time of T_{db}

[5] The scheduler logs the transaction and passes it onto the next step; the mean duration of this step is T_{sched}

[6] A back end transmits a message to another system; the mean time for this step is T_{be}

[7] The scheduler logs the transaction and closes it out; the mean duration of this step is T_{sched}

A prototype system was constructed and benchmarked on the high performance hardware configuration, with the results tabulated below:

Table 4.9.Benchmark Summary

Transaction Step	*Processor Time*		*Disk Accesses*
	Application	*System*	
Front End	0.75 sec	0.25 sec	5
Scheduler	0.05 sec	0.05 sec	0
Data Base Manager	2.10 sec	0.40 sec	25
Scheduler	0.05 sec	0.05 sec	0
Back End	0.50 sec	0.50 sec	10
Scheduler	0.05 sec	0.05 sec	0

The difference in performance between the two hardware configurations is summarized below:

Table 4.10.Hardware Performance Summary

Type	*Processor Speed*	*Time/Disk Access*
High Performance	0.7 MIPS	30 msec
Low Performance	0.5 MIPS	50 msec

MIPS refers to *millions of assembly language instructions executed per second* by a single processor.

The following configurations are under investigation:

[1] One high performance configuration

[2] One low performance configuration

[3] Two low performance configurations running identical application programs and operating system code

[4] Two low performance configurations, one running operating system code and one running application programs

[5] Two low performance configurations running the same operating system code, with one running all application programs except data base management, the other running data base management

For each configuration

A. Plot upper and lower bounds on mean throughput rate and mean delay or response time versus number of clerks. Clearly label the breakpoint number of clerks for the upper bound on mean throughput rate and lower bound on mean delay.

B. Assuming the breakpoint number of clerks, calculate the mean throughput rate for executing jobs per hour.

C. Calculate the total cost to operate each configuration for five years, and the ratio of the cost divided by throughput. Use a straight line five year depreciation for the hardware, assume a fifty per cent tax rate with no tax shelters, and assume the software is developed at a cost of two hundred and fifty thousand dollars with a support cost of one and a half per cent per month of the

development cost over the five year period.

12) A virtual memory system consists of a processor with attached memory plus a much larger external memory. We assume that there is a nonempty queue of jobs waiting for access to this system, so that whenever one job completes execution and departs, another job immediately takes its place. There is only one type of job, and each job requires a certain number of *pages* of memory for execution on the processor. If a program is in execution in the processor and finds that all the required pages are not present in the attached memory of the processor, a page fault is said to occur, and a request is made to the external memory to load the missing page. The mean time to load one page is $T_{external\ storage}$ and equals ten (10) milliseconds. We measure the behavior of an individual program in this system by gathering measurements for the mean time between page faults with the main processor memory configured for a given number of pages, and we then repeat this process for a different number of pages. The total number of jobs in execution in the system we call the *degree of multiprogramming* of this system. Our goal is to determine the mean throughput rate of completing jobs for a given total number of memory pages and degree of multiprogramming and other parameters.

We let K denote the degree of multiprogramming and M denote the total number of memory pages. Hence, each job has an average of M/K pages available to it at any time. The mean time between page faults for an individual program can be approximated by AK^2, where A=9 μsec.

Answer the following questions:

A. What is the state space for this system?

B. Find an upper bound on the mean time the processor is busy as a function of the degree of multiprogramming K. For M=100, plot this versus K. Interpret your results.

13) Twenty five operators at terminals carry out interactive work with a computer system. The computer system also executes batch work. The hardware configuration for the computer system is one processor and one disk. Each operator at a terminal undergoes the same basic cycle: some time is spent reading and thinking and typing, followed by some time waiting for the system to respond. The thinking time has a mean of thirty seconds, denoted by T_{think}. The response time has a mean of R. Each interactive job undergoes some processing and some disk access activity over and over until each job is completely executed. The mean processor time per visit to the processor for each interactive job is ten milliseconds, and each interactive job visits the processor ten times on the average. The mean disk access time for each interactive job is ninety milliseconds, and each interactive job requires ten disk accesses on the average. Each batch job on the other hand requires one disk access followed by one second of processing time, on the average.

The following measurements are carried out on the system in operation:

- The processor is found to be completely busy or saturated throughout the measurement interval
- The mean response time for interactive jobs is four seconds

Answer the following questions:

A. Make a table showing the resources held by each step of each job and the mean time for each step

B. What is the bottleneck?

C. What is the mean throughput rate of executing batch jobs?

D. What is the disk utilization due to interactive and to batch work?

E. Suppose a new processor replaces the old processor, and is found to be five times as fast as the old processor. Answer the following questions:

- What is the bottleneck now?
- What is the disk utilization due to interactive and to batch work?
- What is a lower bound on the mean response time for interactive work?
- What is an upper bound on mean throughput rate for interactive work?

14) A computer system consists of a CPU, a drum with a direct memory access (DMA) controller, and eight pages of memory. A sequence of jobs is executed on this system. The mean throughput rate of executing jobs is denoted by λ, measured in jobs per millisecond. Each job consists of a sequence of steps, with each step of each job indexed by K=1,2,3,4. A job need not execute every step; however, the average number of type K steps per job, denoted by V_K, is given in the table below:

Table 4.11.Mean Number of Steps/Job

Symbol	*Steps/Job*
V_1	4/3
V_2	8/15
V_3	2
V_4	1

The resource requirement table for the four steps are shown in the table below:

Table 4.12.Resources/Step and Mean Holding Time/Step

Step	*CPU*	*Drum*	*Pages*	*Time*
1	1	0	4	10 msec
2	0	1	4	20 msec
3	1	0	2	30 msec
4	0	1	2	20 msec

Answer the following questions:

A. Define the state space of this system for *running* jobs. How many states are in the state space?

B. Use Little's Law to write the conservation equations relating the mean throughput rate λ to state space averages.

C. Find an upper bound λ_{max} on the mean throughput rate λ. What resource is the bottleneck if $\lambda=\lambda_{max}$?

D. Assuming that $\lambda=\lambda_{max}$, what is the

- Percentage of time the CPU is busy
- Percentage of time the drum is busy
- Average memory utilization

for *running* jobs?

15) A transaction processing system retrieves a record from storage on one disk spindle, processes it, and stores it on a second disk spindle. The hardware configuration consists of a single central processor connected via a switch to two direct memory access disk controllers, with one spindle for each controller; the direct memory access capability allows the central processor to execute work while the disk controller is busy accessing a record. All of the text required to process records is resident in main memory. The processor can buffer at most S blocks of data at a time. Each disk access retrieves one block at a time. N denotes the number of records per block. Each record consists of B bytes. T_{access} denotes the mean time to move the disk head from an arbitrary point to the start of a block. $T_{transfer}$ denotes the mean time to transfer one byte of data from an input/output device to a disk controller.

T_{proc} denotes the mean processor time per record.

A. Find an expression in terms of model parameters and plot it for the maximum mean throughput rate (records per unit time) versus the number of records per block.

B. Find the ratio of the reciprocal of the maximum mean throughput rate divided by the mean time to process a record from start to finish (this is the reciprocal of the mean throughput rate with no concurrency).

C. What are the bottlenecks in this system?

D. Suppose that the following numbers specify the model parameters:

Table 4.13.Model Parameters

Attribute	*Symbol*	*Time*
Disk Access Time	T_{access}	50 msec
Disk Transfer Time	$T_{transfer}$	1 μsec
Buffer Size	B	4096 bytes
Processor Time/Record	T_{proc}	30 msec

For N=1,2,4,8 what is the numerical value of the maximum mean throughput rate in records per unit time?

16) A computer systems consists of a central processing unit (CPU), processor memory, and two disks (labeled I and II). Each disk is connected to the CPU by its own channel (DMA controller). A hardware monitor is available for measuring the system performance, which has as output three signals, $X_{CPU}(t)$ for the processor, $X_I(t)$ and $X_{II}(t)$ for disks I and II respectively, where

$$X_{CPU}(t) = \begin{cases} 1 & \text{if } CPU \text{ busy at time } t \\ 0 & otherwise \end{cases}$$

$$X_K(t) = \begin{cases} 1 & \text{if } channel\ K \text{ busy at time } t \\ 0 & otherwise \end{cases} \qquad K=I,II$$

The following measurement data is available concerning the execution of workload over a time interval beginning at time t=0 and ending at time t=T:

$$X_{CPU}(t) + X_I(t) + X_{II}(t) > 0 \quad 0 < t < T \tag{i}$$

$$\int_0^T X_{CPU}(t)dt = 100 \text{ seconds} \tag{ii}$$

$$\int_0^T X_I(t)dt = 100 \text{ seconds} \tag{iii}$$

$$\int_0^T X_{II}(t)dt = 125 \text{ seconds} \tag{iv}$$

$$\int_0^T X_{CPU}(t)X_I(t)dt = 50 \text{ seconds} \tag{v}$$

$$\int_0^T X_{CPU}(t)X_{II}(t)dt = 25 \text{ seconds} \tag{vi}$$

$$\int_0^T X_I(t)X_{II}(t)dt = 75 \text{ seconds} \tag{vii}$$

$$\int_0^T X_{CPU}(t)X_I(t)X_{II}(t)dt = 25 \text{ seconds} \quad \text{(viii)}$$

Answer the following questions:

A. What is the fraction of time the CPU is busy during (0,T)?

B. Evaluate

$$\frac{1}{T}\int_0^T [X_{CPU}(t) + X_I(t) + X_{II}(t)]dt$$

C. What fraction of the *potential* speedup was in fact realized?

17) Terminals are connected over data links to a front end processor to a computer system. Each terminal has its own dedicated three hundred bit per second line. Each operator at a terminal does the same job over and over again: each operator spends a certain amount of time reading and thinking and typing, denoted T_{think}, and then strikes a *send* key. At that point the screen data is transmitted over the link to the front end processor; on the average four hundred bits of data are input. The front end processor is connected to the computer system by a very high speed data link. Each job enters a staging queue where it waits until it can enter the main computer system for processing. The computer system can hold at most five jobs at any one time: if there are less than five jobs in the system, a job in the staging queue will enter the system immediately, otherwise jobs are queued for entry in order of arrival. The system consists of a single processor and a single disk, and each job requires an average of T_{proc}=2 seconds of processor time and N_{disk}=30 disk accesses, with each disk access requiring a fifty millisecond access time. Once the job completes execution, it is transmitted back over the high speed link to the front end processor, and the terminal displays the output. On the average each screen has 4800 bits of information for output. Assume that the time for signals to propagate from the terminal to the front end processor and back are negligible compared to any other time interval. Assume that the time for signals to propagate to and from the system and the front end processor are negligible compared to any other time interval.

A. Make up a table showing each step of processing a job and the resources required for that step and the mean time duration of that step.

B. What is the bottleneck resource in this system for N=1 terminals? What is the bottleneck resource in this system as $N \to \infty$?

C. Plot an upper bound on the mean throughput rate versus the number of active terminals. Clearly label all regions and breakpoints in terms of model parameters.

D. Mean response time is defined as the time interval from when the operator initiates transmission of a screen of data until the start of output on the screen. Plot a lower bound on mean response time versus the number of active terminals. Clearly label all regions and breakpoints in terms of model parameters.

E. Repeat all the above assuming that the system can now hold twenty jobs at once, not just five.

F. Suppose that the terminals' links are replaced with 56,000 bits per second links connected to a front end that is now connected to a space satellite earth station. The propagation time of signals between the terminals and front end is negligible compared to all other time intervals. The one way propagation time of signals from the front end to the computer system is one fourth second. Repeat all the above.

18) A packet switching system consists of N=10 identical nodes connected in a ring, with index K=0,...,N−1=9. Each node transmits to only one other node, and each node receives from only one other node; all nodes are connected by identical one way transmission links with transmission rate

$C=10^6$ bits per second. All packets are routed counterclockwise around the ring. Packets can arrive at any node and are transmitted to any of the other remaining $N-1$ nodes, and thence depart from the system. All packets are fixed in size at $B=1000$ bits. The fraction of the *total* network packet load entering at node $I=0,...,N-1$ and departing at a node that is J nodes away (counterclockwise) is denoted by $F_{I,I+J}$ with the sum $I+J$ being modulo N. We assume that

[1] All stations are statistically identical or symmetric, i.e., $F_{I,I+J}$ is independent of I.

[2] No station sends packets to itself (i.e., $F_{II}=0, I=0,...,N-1$).

[3] All packets are sent to *some* node:

$$\sum_{I=0}^{N-1}\sum_{J=1}^{N-1} F_{I,I+J} = 1$$

The total mean packet arrival rate to the system, i.e., summed over all nodes, is denoted by λ.

A. If $F_{I,I+J} = 1/N(N-1)$ find an upper bound on the maximum mean packet switching rate for this system.

B. What choice of $F_{I,I+J}, 0 \le I,J \le N-1$ permits the largest mean packet switching rate?

C. What choice of $F_{I,I+J}$ permits the smallest mean packet switching rate?

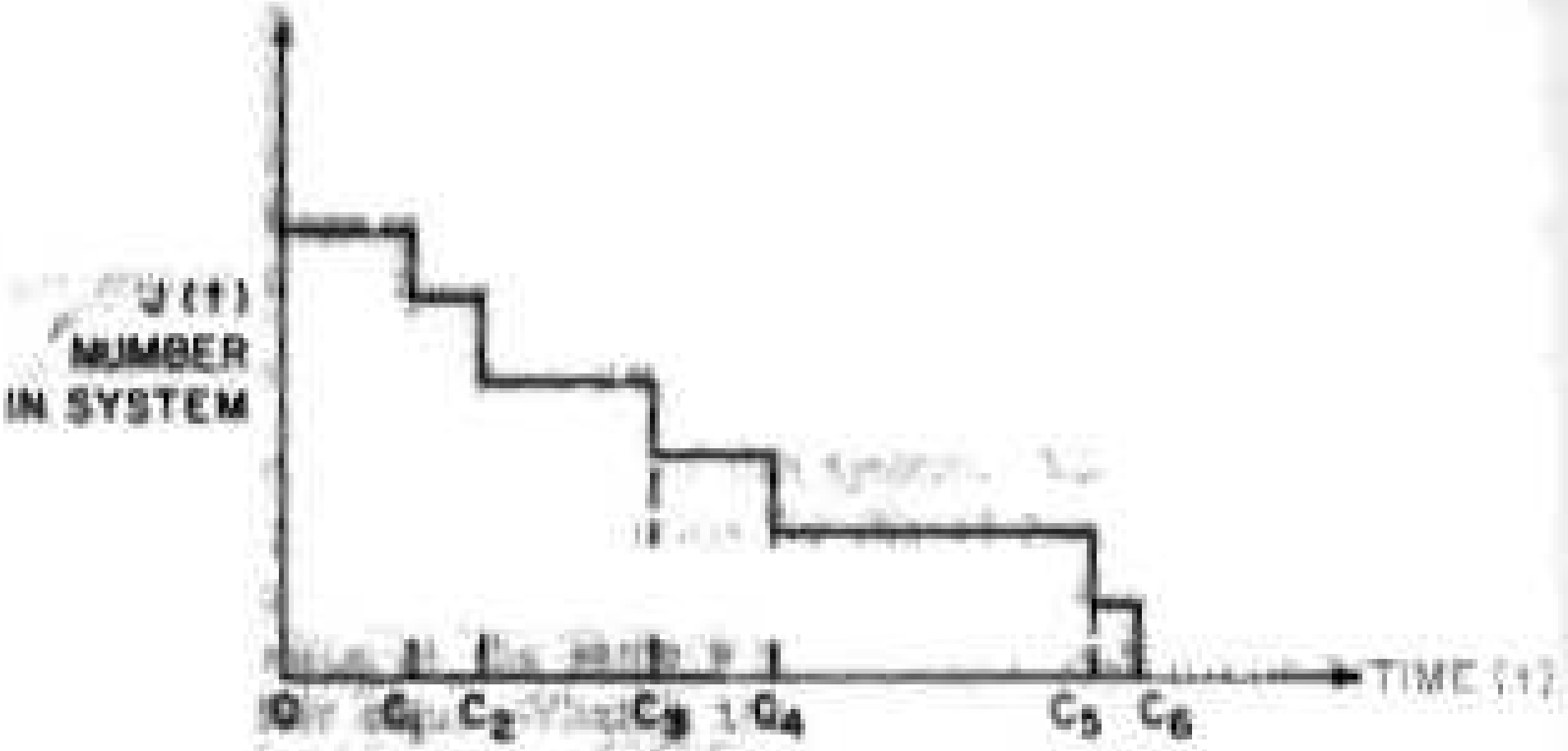


Figure 4.1.Number in of Jobs in the System vs Time Initial Work Present at T=0/No Arrivals for T>0

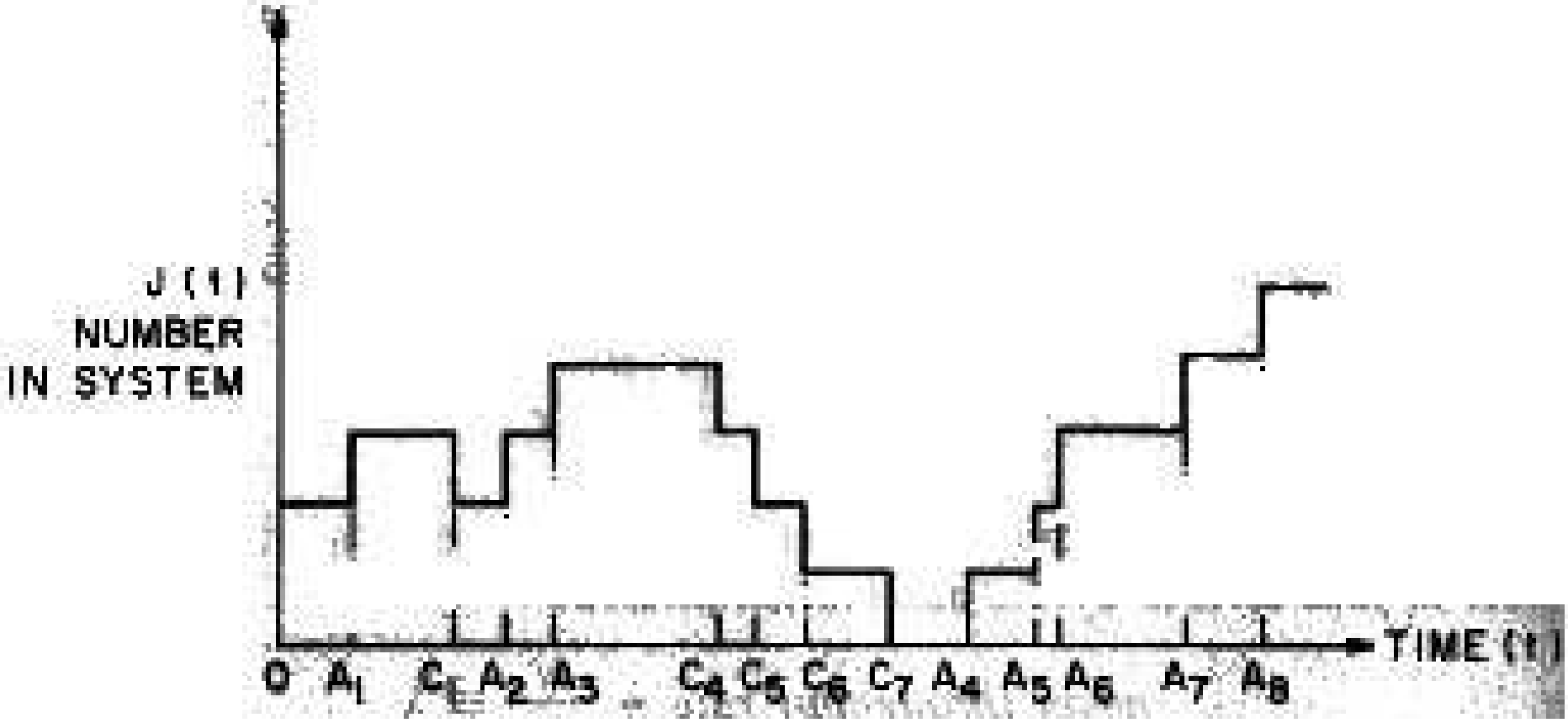
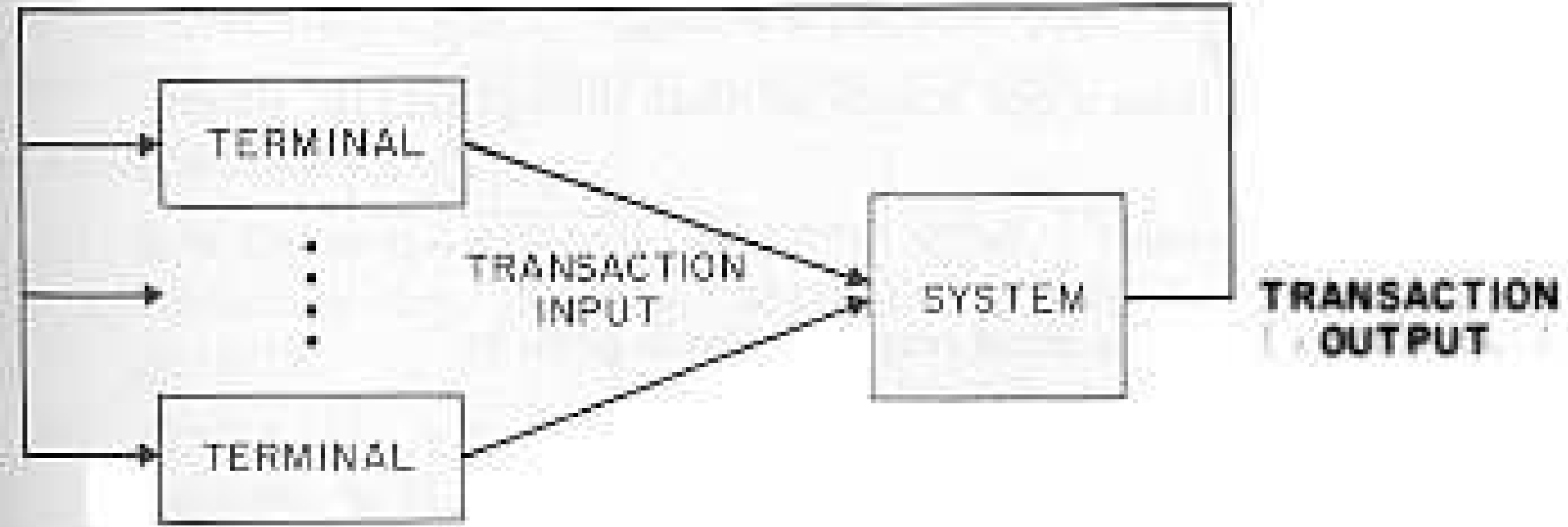


Figure 4.2.Number in System vs Time
Initial Work Present at T=0/Arrivals for T>0



INTERACTIVE ONLINE SYSTEM BLOCK DIAGRAM

Figure 4.3.Hardware Block Diagram of Time Sharing System

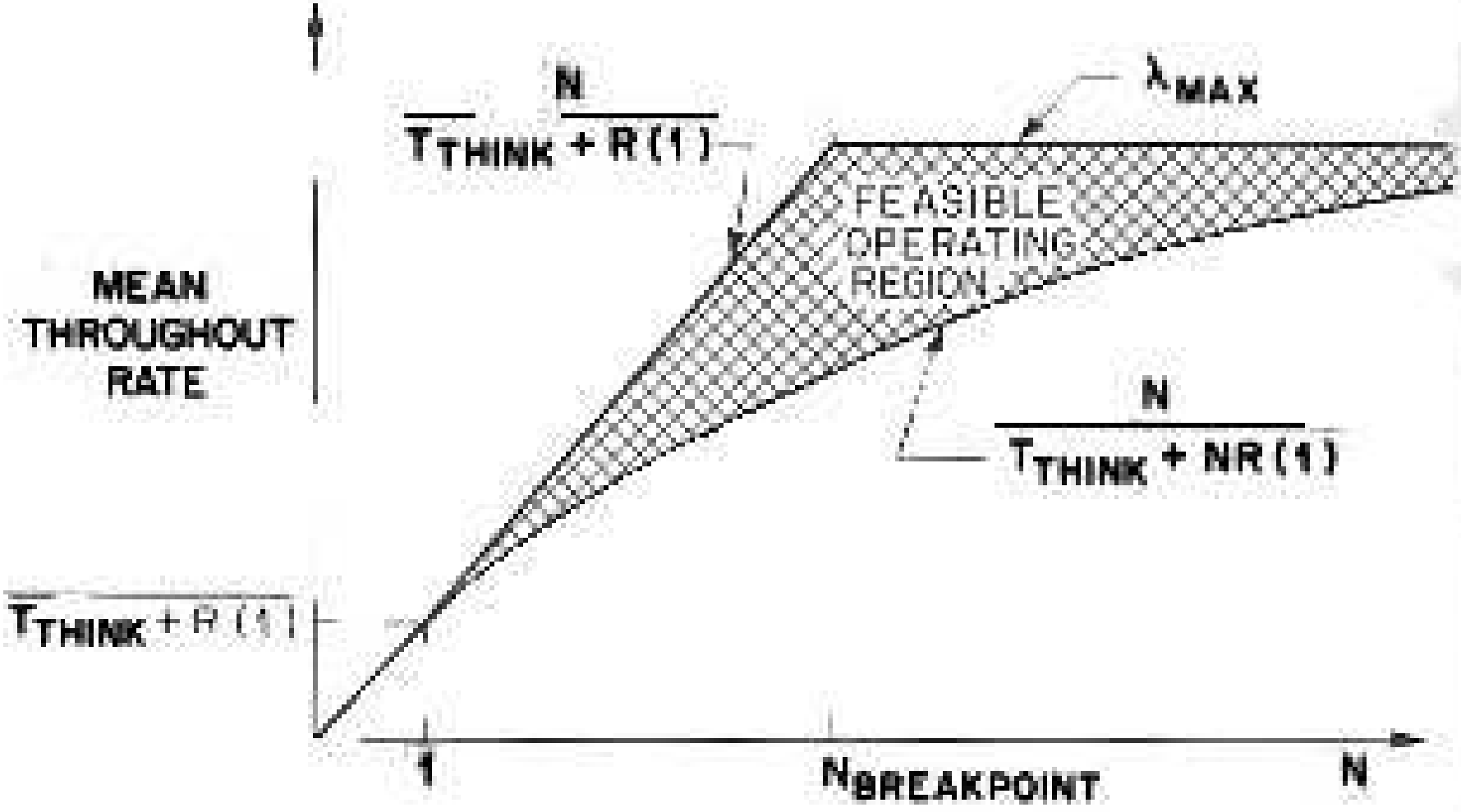


Figure 4.4.Mean Throughput Rate Bounds versus Number of Active Users

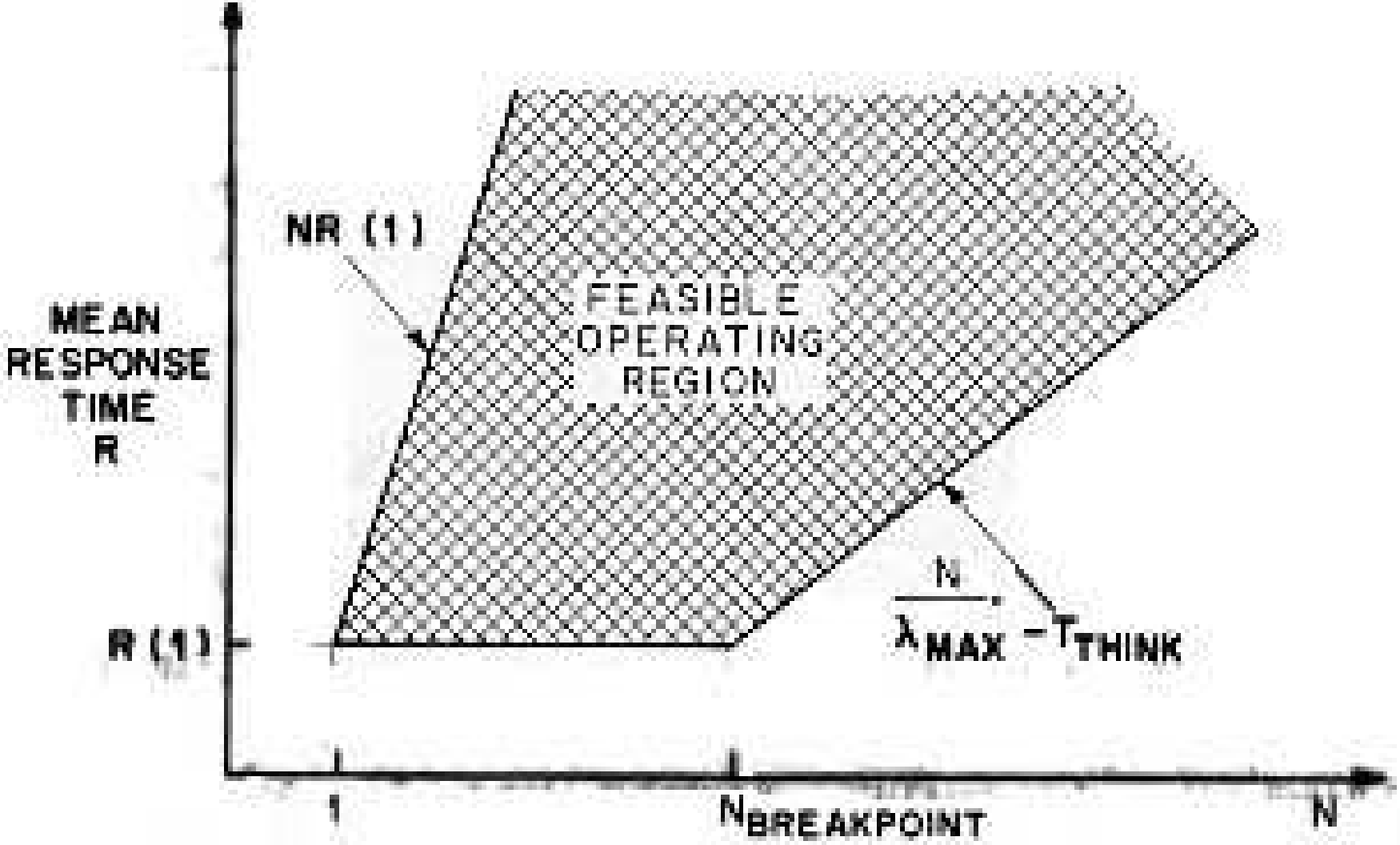


Figure 4.5.Mean Response Time Bounds versus Number of Active Users

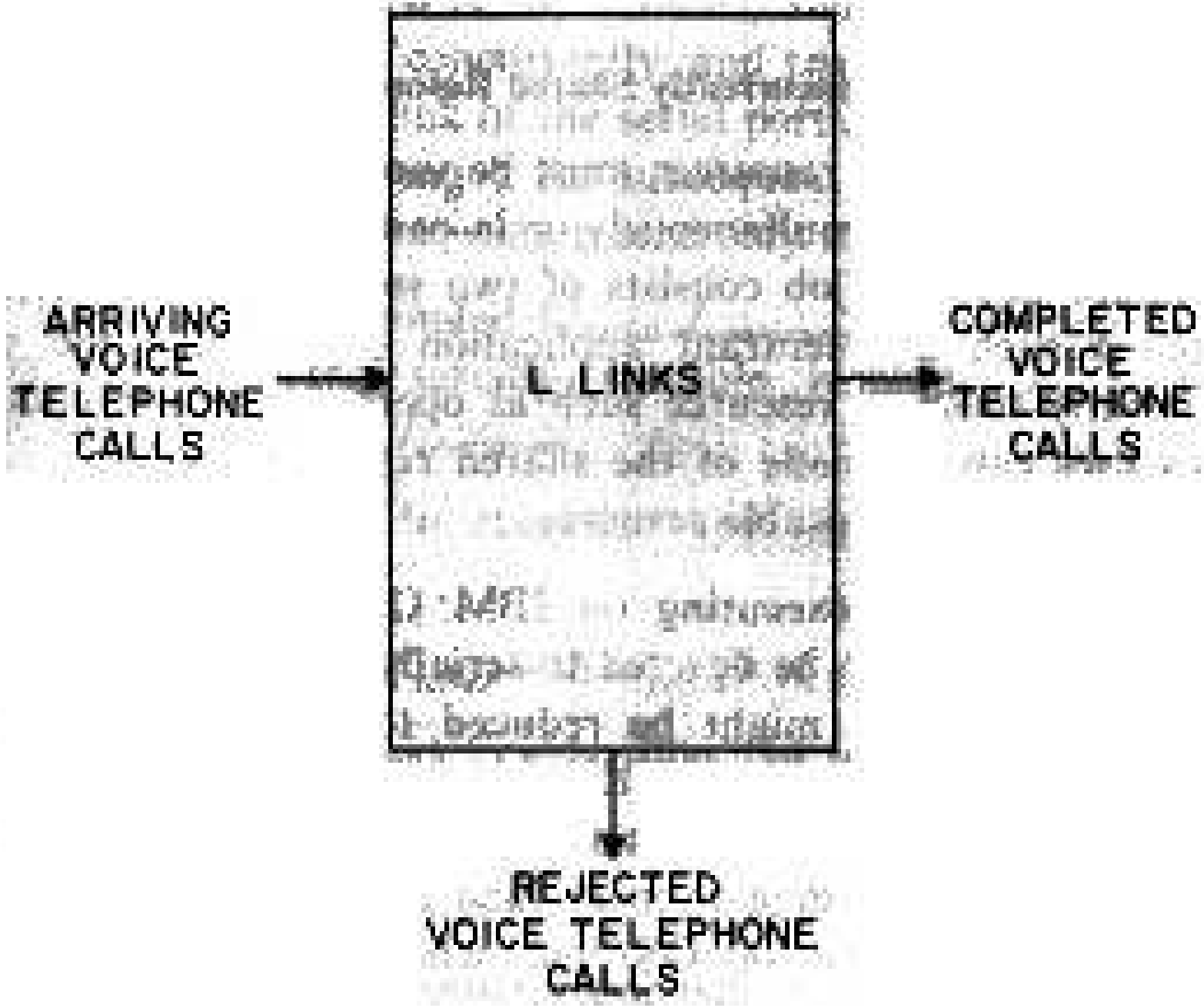


Figure 4.6.Telephone System Block Diagram

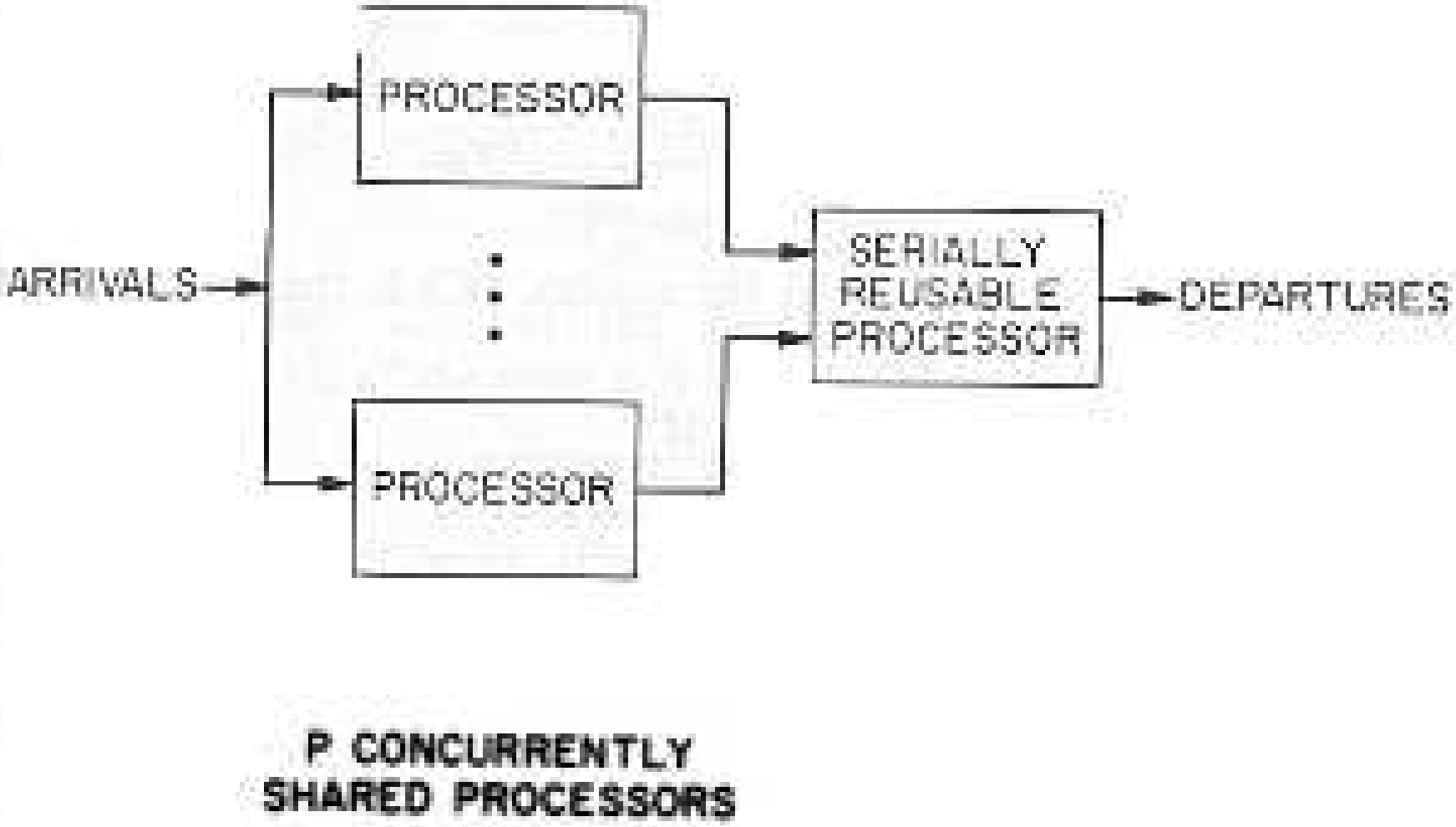


Figure **4.7.Hardware Block Diagram of Serial and Concurrent Processors**

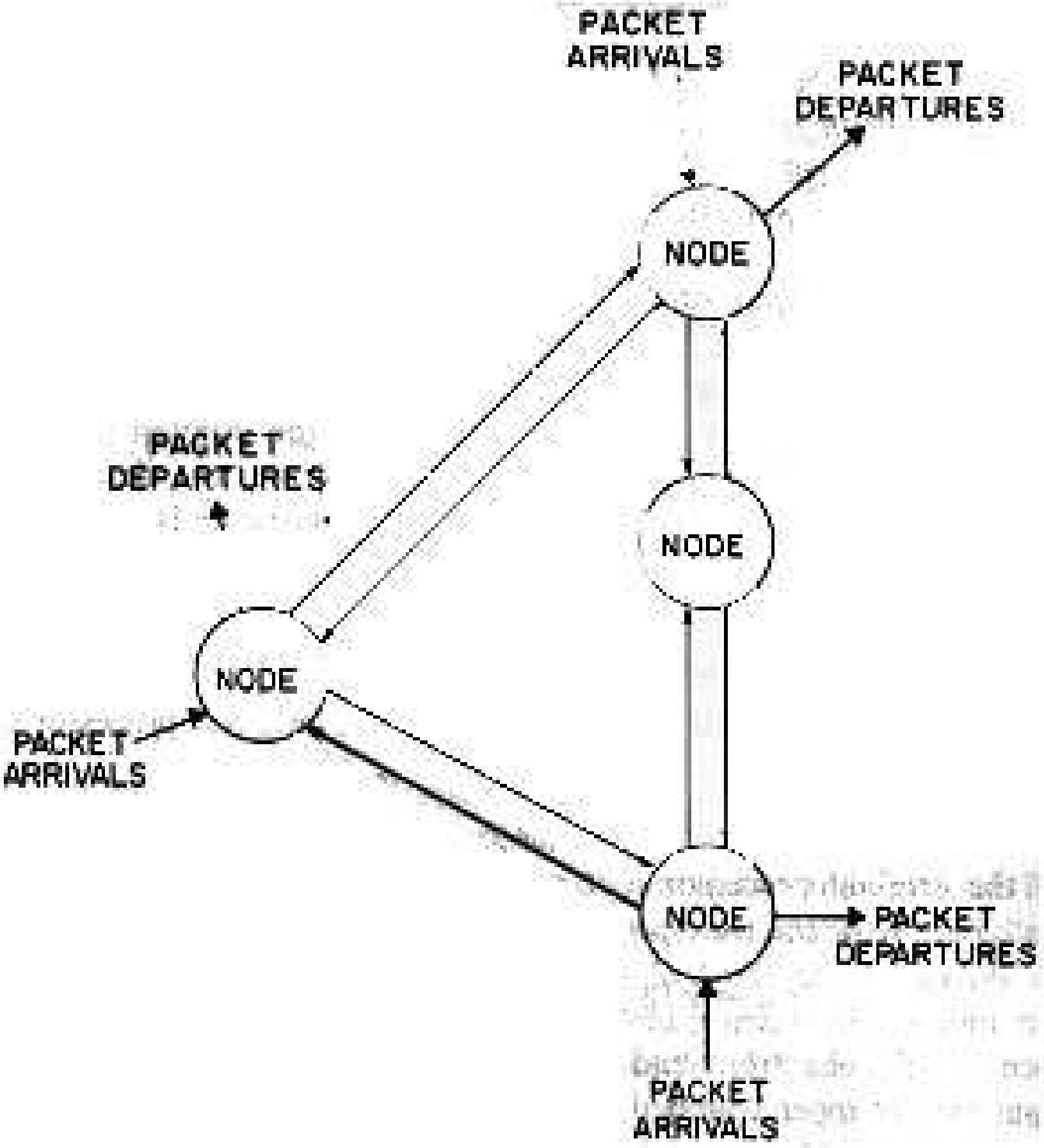


Figure 4.8.Packet Network Node Block Diagram

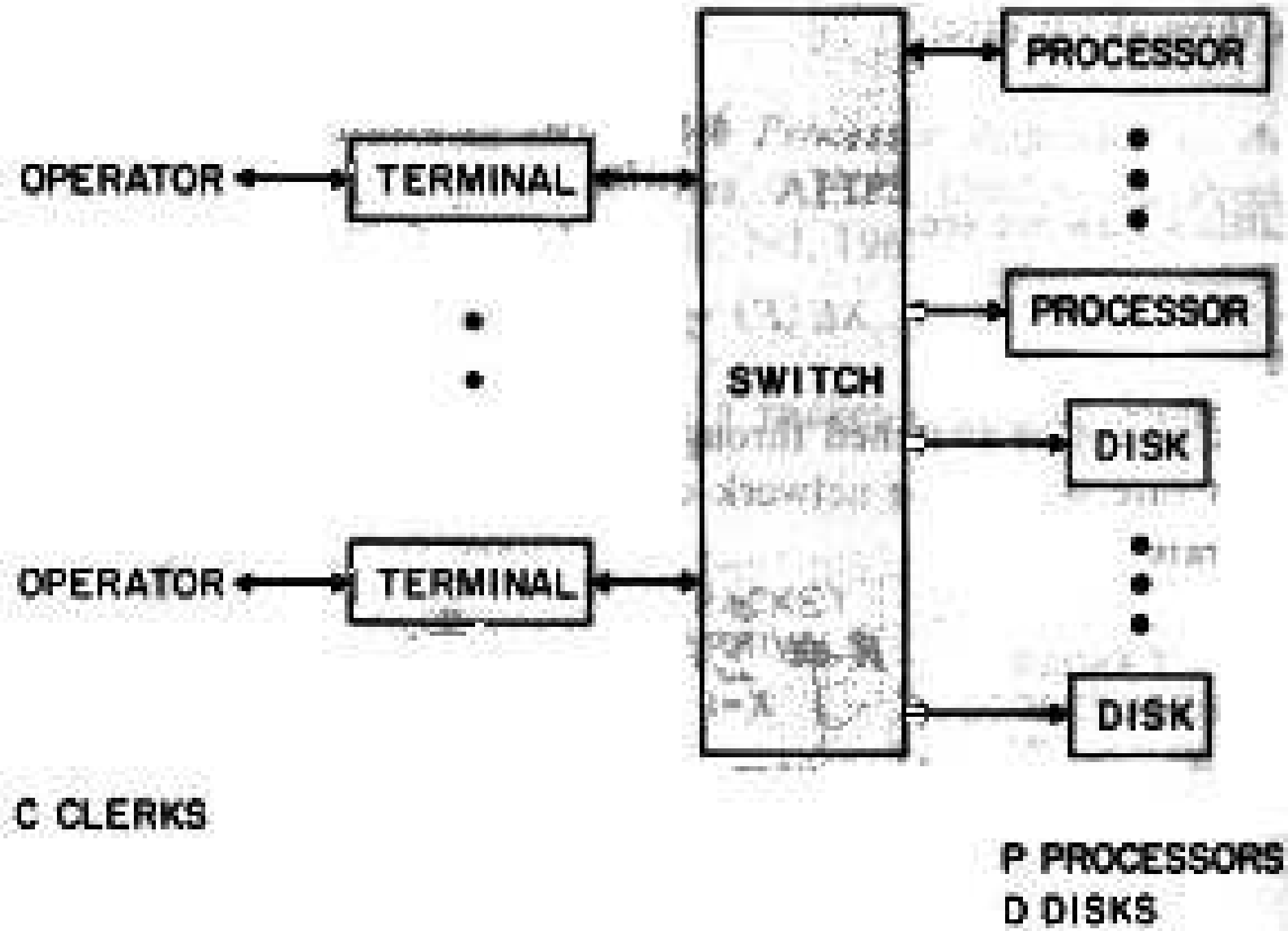


Figure 4.9.Processor and Disk Hardware Block Diagram

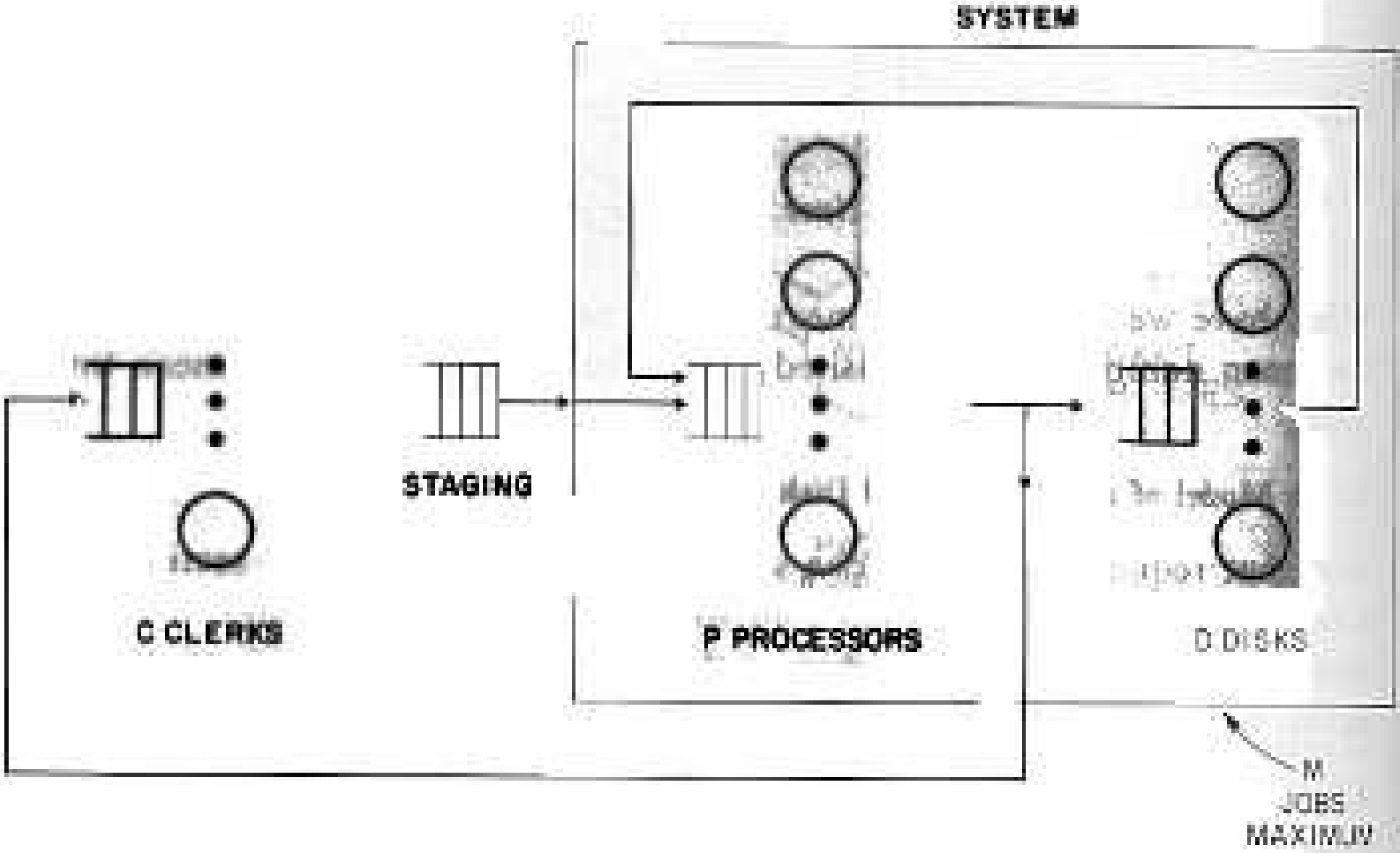


Figure 4.10.Queueing Network Block Diagram

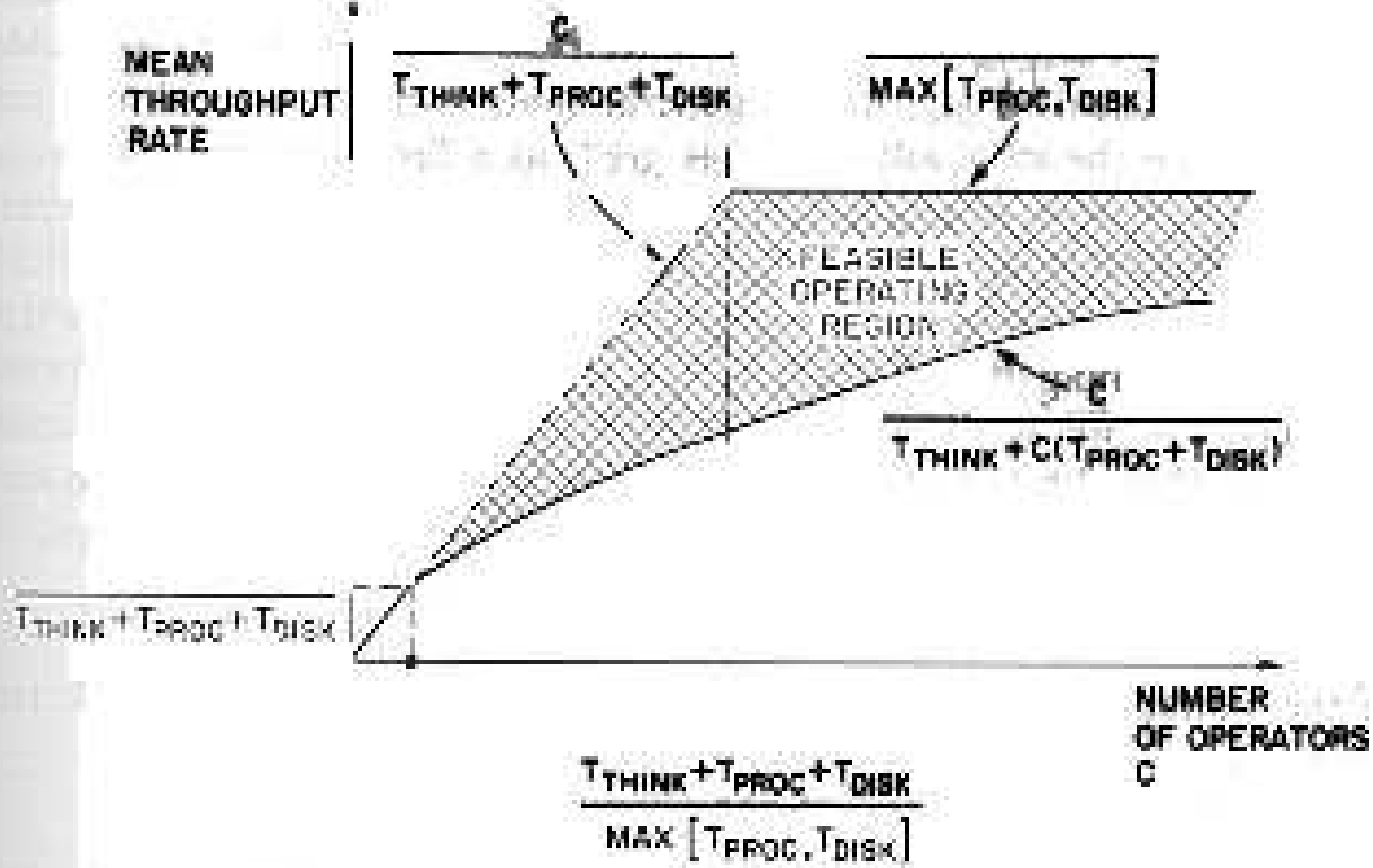


Figure 4.11.One Processor/One Disk Mean Throughput Bounds vs Number of Terminals.

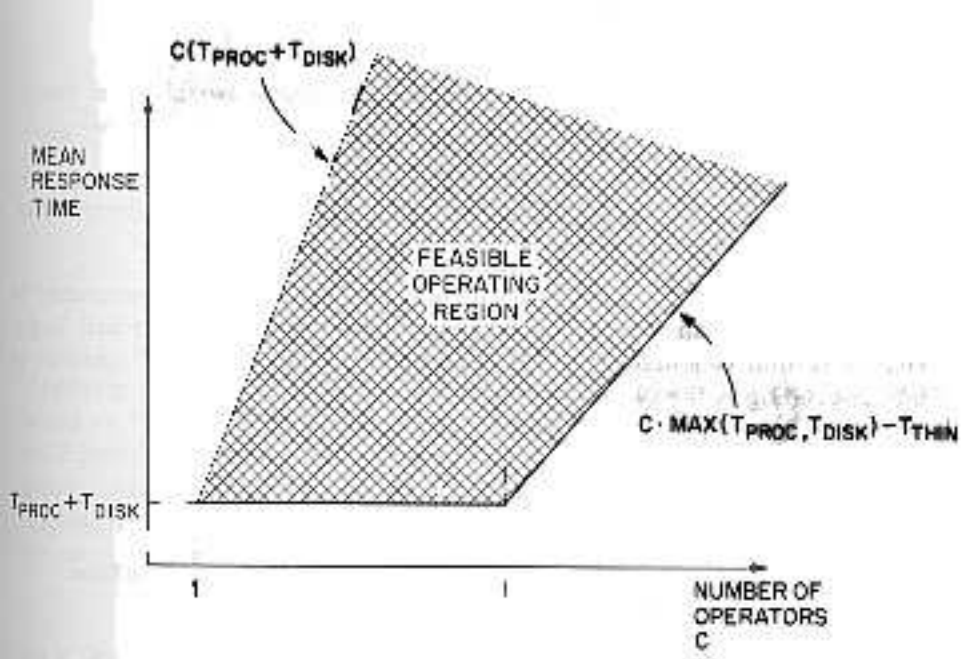


Figure 4.12.One Processor/One Disk Mean Response Time Bounds vs Number of Terminals.

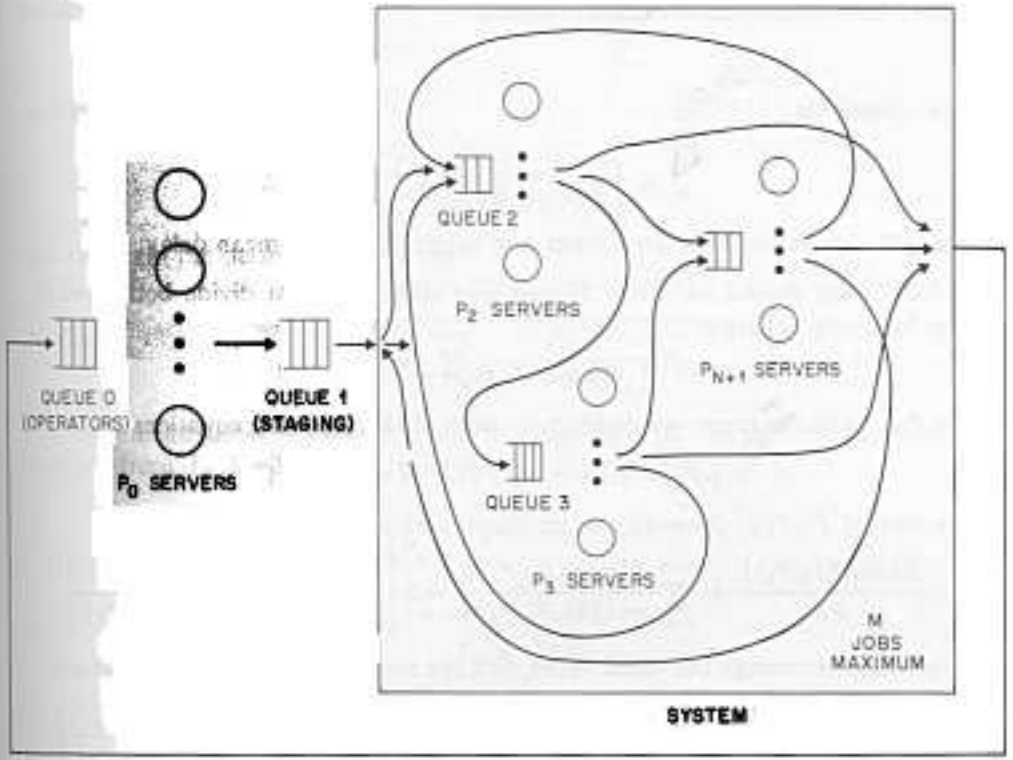


Figure **4.13.Block Diagram of Memory Constrained Queueing Network**

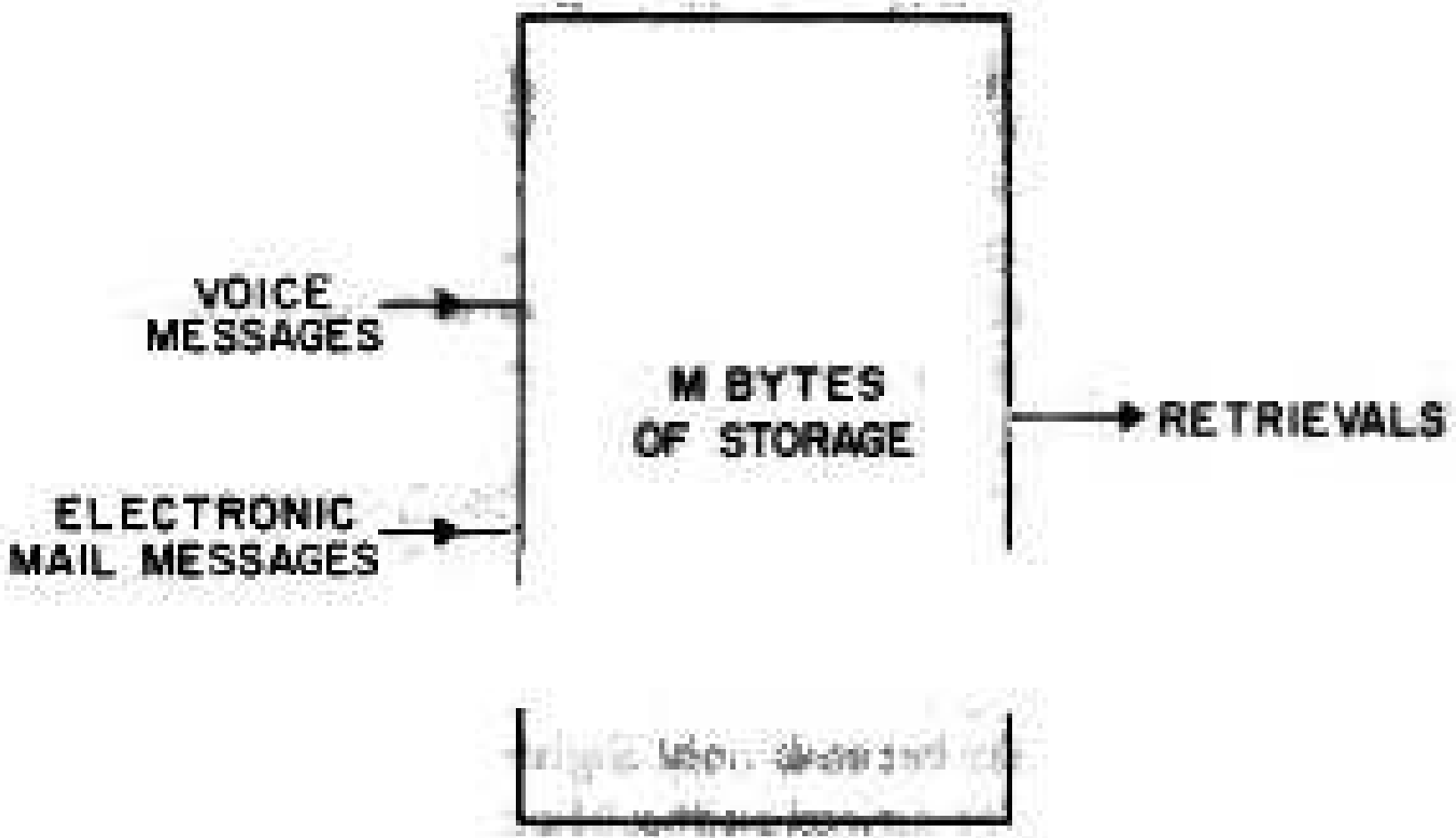
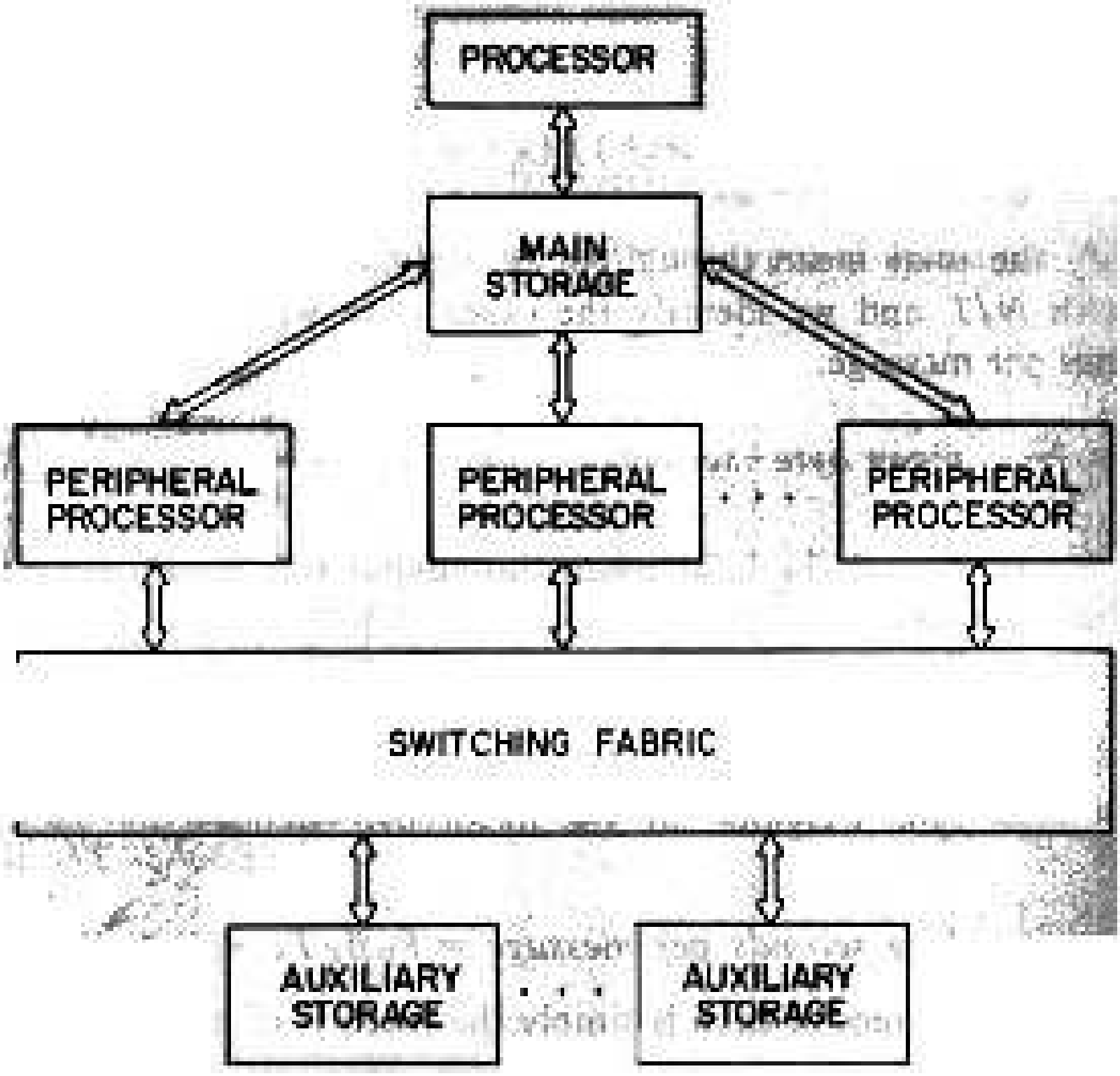


Figure 4.14.Integrated Communication Storage Subsystem Block Diagram


PROCESSOR
MAIN STORAGE
PERIPHERAL PROCESSOR
PERIPHERAL PROCESSOR
PERIPHERAL PROCESSOR
SWITCHING FABRIC
AUXILIARY STORAGE
AUXILIARY STORAGE

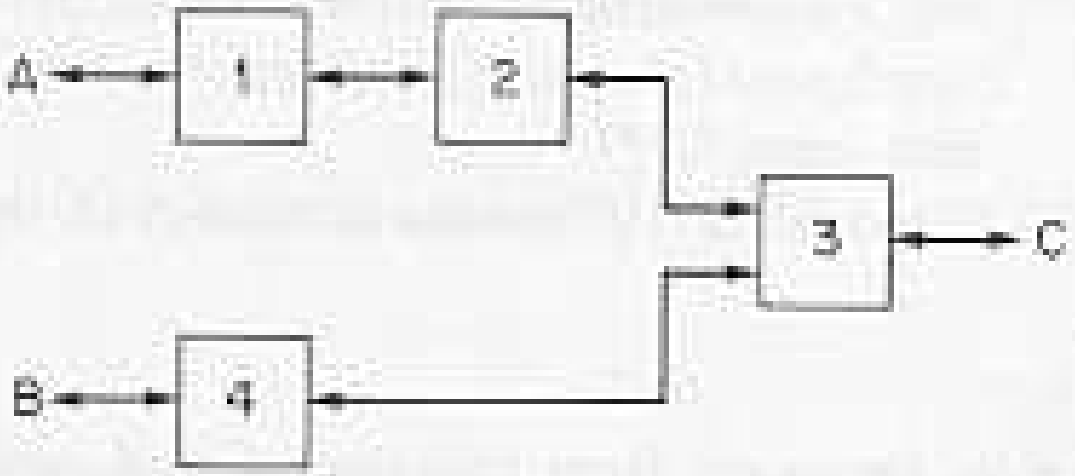


Figure 4.16.Distributed Data Communications System Block Diagram

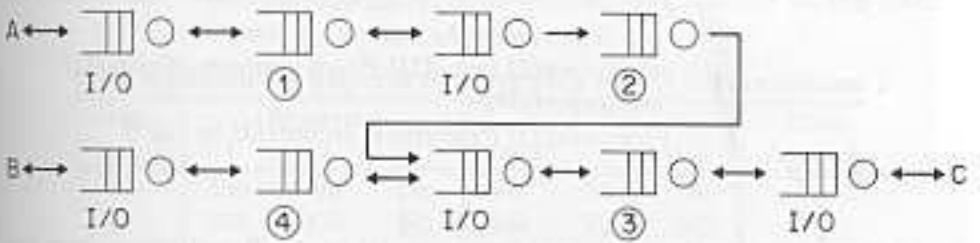


Figure 4.17.Queueing Network Block Diagram

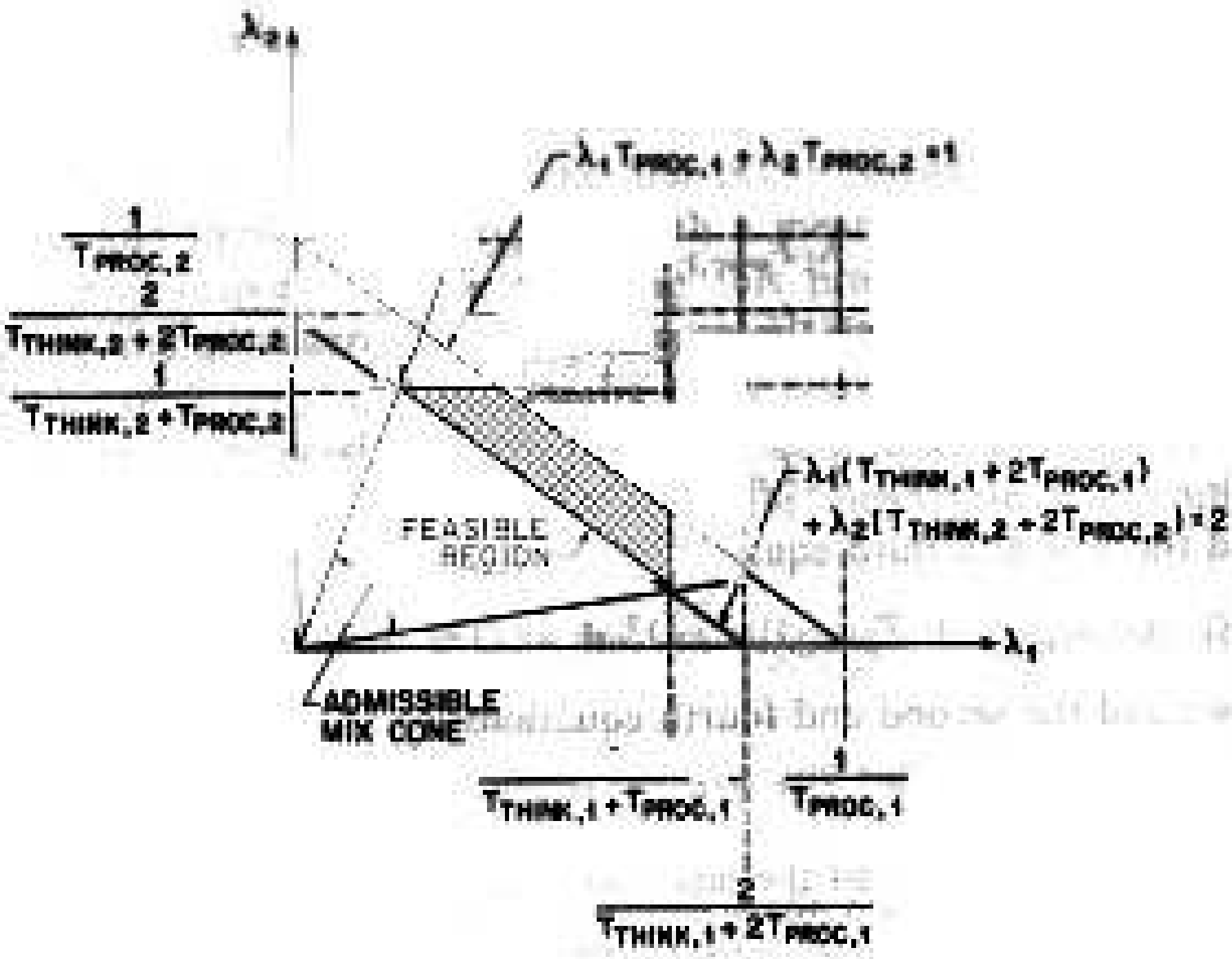


Figure 4.18.Admissible Mean Throughput Rates