Copyright © 1983 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 2, February 1983
Printed in U.S.A.

Upper and Lower Bounds on Mean Throughput Rate and Mean Delay in Memory-Constrained Queueing Networks

By E. ARTHURS and B. W. STUCK

(Manuscript received April 27, 1982)

Operators use terminals to enter transactions into a system and then wait for the system to respond. The system contains serially reusable resources, and can hold a maximum number of jobs. Each job requires a total mean amount of service at each stage. We calculate upper and lower bounds on the mean throughput rate and mean delay as a function of model parameters, and present examples that show these bounds are sharp, *in the sense that they are achievable given only mean values. We also present partial results for closed queueing networks where the long-term, time-averaged distribution of number of jobs at each node in the network obey so-called* product form *separation of variable type of probability distributions. Examples and data from actual systems illustrate the utility of the work.*

I. INTRODUCTION

At present there is great interest in modeling the traffic-handling characteristics of computer and communication systems using queueing networks.[1–4] The change in cost of electronic solid-state circuitry[5,6] and rising personnel costs[7,8] offers strong incentive to design cost-effective digital systems.

Computer communications systems can often be modeled quite naturally by a network of queues, where a job receives service at one stage or queue and then migrates to another stage, until it is completely serviced. Examples of actual systems and associated models are presented in later sections of this report. This class of models captures several fundamental phenomena of such systems, including asynchronous and concurrent execution of different jobs and different amounts of service required at each stage of execution. To answer whether this type of model is valid, controlled experimentation and measurement must be carried out, and goals or criteria must be set for judging goodness of fit. Finally, one would like to use these models to predict or extrapolate behavior into unknown regions of operation to guide decision making.

Here we focus on one technique for bounding the mean throughput rate and mean delay of an abstraction of a computer communications system. This is only one factor among many others, such as cost, flexibility, and reliability, that must be considered in choosing one design over another for a given application. We drop these other factors from consideration after this point in the interest of brevity.

Broadly speaking, there are two reasons for wanting to quantify the traffic-handling capabilities in a computer communication system:

(*i*) Cost reduction of an existing service or product:

(a) In an existing system, it is often possible to modify existing scheduling policies to improve performance at an acceptable cost. An example would be to change from one memory partition per application program to a memory pool shared among all application programs.

(b) In a system handling a fixed set of job types, different equipment configurations can accomplish these jobs at different costs. Which should be chosen? An example would be to compare using two slow disks versus one fast disk.

(*ii*) Comparisons are often desired between current operations and wholly new modes of operations. An example would be using an existing batch computer system for time sharing, using the existing time sharing system for electronic mail, or using existing word processors for voice-annotated text services.

To quantify these issues, typically two stages are involved: the first is *synthesis*, where goals are stated along with different alternatives for reaching those goals, while the second is *analysis*, where the performance (here the mean throughput rate of finishing jobs and the mean delay for each stage of job execution) is quantified. Goals may be either oriented toward the total system, such as total number of jobs of a given type that are handled during an hour, or toward an individual, such as the mean delay to handle one or more stages of a job; along with goals such as these there should be some measure of the *sensitivity* of the goals to different operating points, and so forth.

Analysis often begins by postulating a set of parameters that carry or capture specific operational aspects, drawing inferences based on these parameters (either by mathematical analysis or by discrete event simulation),[9–11] measuring actual or simulated operation, and then repeating this process until it is felt that additional work is no longer warranted.

Our goal here is to demonstrate how to carry out part of this process by a straightforward technique for obtaining bounds on mean throughput rate and mean delay, given only mean value information for the service required at each stage of job execution. In our opinion, there are three principal contributions:

(*i*) A new technique for obtaining a *lower* bound on mean throughput rate and an associated *upper* bound on mean delay. Earlier workers (e.g., Ref. 1, pp. 212–25) obtained an *upper* bound on mean throughput rate and an associated *lower* bound on mean delay. Furthermore, we present an example that shows that given *only* mean values for the amount of service required at each visit to each stage in the queueing network model, *either* bound can be approached arbitrarily close, depending upon on the amount of fluctuation present about the mean service times. This shows that these bounds are *sharp*, much as was done earlier for loss systems.[12,13] The interested reader is referred to related works.[14,15]

(*ii*) A new technique for calculating both *upper* and *lower* bounds on the mean throughput rate and mean delay for a class of closed queueing networks whose long-term, time-averaged distribution for the number of jobs in system obeys a so-called *product form* or separation of variables decomposition[2,3] (for an application case study, see Ref. 16). The upper bound on mean throughput rate is the reciprocal of the total mean time to execute the transaction plus the *average* time spent in execution per node, while the lower bound on mean throughput rate is the reciprocal of the total mean time to execute the transaction plus the *maximum* mean time spent in execution per node. The tightness of these bounds will thus depend on how close these factors are to one another (Zahorjan et al., developed these results independently;[17] our derivation is felt to be more straightforward).

(*iii*) Data from controlled experimentation on actual computer and computer communication systems is presented to validate the approach presented here.

The examples presented are deliberately elementary, chosen for tractability. Everything of interest can be represented by formulas. Furthermore, this approach is a natural starting point for virtually any study of traffic-handling performance, can be refined in a variety of ways, used to check and bound much more complex analyses or simulations, and can be immediately related to measurements in an actual system. Often data are simply not available to describe the arrival statistics and service required for each step of each job, such as would be needed in simulation studies; this suggests using a mean value (distribution free) analysis, rather than more stringent distributional assumptions, and then assessing performance sensitivity by varying the mean value, rather than investing effort in simulation studies. We advocate *synthesis* via *analysis* of the performance of a given configuration. The approach adopted here is not exhaustive, but it is fundamental. The examples show that only two avenues are available for improving computer communication system performance, reducing the time to handle a given task (i.e., speedup) and handling two or more tasks simultaneously [i.e., concurrency, either real (multiplexing multiple resources) or apparent (via scheduling a single resource)].

II. A MODEL OF A PROCESSOR AND DISK SYSTEM

In this section we deal with a mathematical abstraction of an on-line transaction processing system.

2.1 Model description

Clerks at terminals spend a mean amount of time reading, thinking, and entering the transaction, and then wait for the system to respond before repeating this cycle. Each transaction requires a mean amount of processor time and disk secondary storage access time to be completely executed. The system is configured with a finite amount of memory, and hence can hold a maximum number of jobs at any one time. Figure 1 shows a hardware block diagram of the system. The cycle that a job or transaction follows can be described by a path through a network of queues. The first stage or queue is associated with operators at terminals entering each transaction. Next, each job enters a *staging* queue, where it waits if there are already the maximum number of allowable jobs in the system, and otherwise it immediately enters the system. Once inside the system, a job will receive some processing, then require accessing some data from secondary disk storage, then some processing, and so forth until it is completely executed. Finally, control will return to the operator at the terminal and the process begins anew. Figure 2 shows a queueing network block diagram for this system, consisting of four queues: one for operator jobs, one for staging, one for processors, and one for disks.

The ingredients we need are

(*i*) The number of clerks, C, actively submitting transactions to the system

(*ii*) The number of processors, P, and disks, D, connected by a common switch. (The switch is 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.)

(*iii*) The maximum number of jobs allowed inside the system at any one time, M

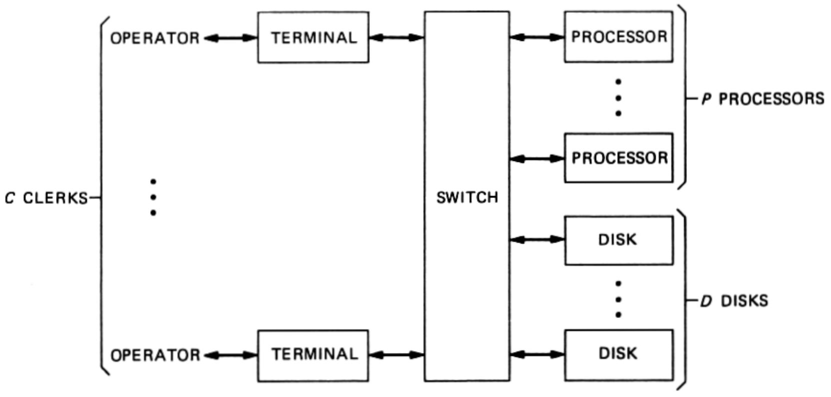


Fig. 1—Block diagram of the system hardware.

(*iv*) The mean time each operator spends reading, thinking, and entering each transaction, denoted T_{think}

(*v*) The mean processor time, $T_{\text{processor}}$, and mean disk access time, T_{disk}, per transaction.

The outputs of the analysis are the mean throughput rate of executing transactions and the mean response time (as seen by operators, including both execution time plus waiting time) as a function of model parameters. 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.

From the vantage point of an operator at a terminal, we see that each transaction undergoes two stages of processing:

(*i*) A stage spent preparing the transaction for execution, with mean time interval, T_{think}

(*ii*) A stage spent waiting for the transaction to execute, with mean time interval, R.

For one operator at one terminal, the mean cycle time per transaction is simply the sum of the mean preparation time and mean delay. Hence, when C operators are active, the mean throughput rate equals simply C times the mean throughput rate for one operator:

$$\text{mean throughput rate} = \lambda = \frac{C}{T_{\text{think}} + R}.$$

If we rewrite this equation to find the mean response time, we see

$$R = \frac{C}{\text{mean throughput rate}} - T_{\text{think}}.$$

These two relationships will be fundamental in determining feasible operating regions for mean throughput rate and mean delay.

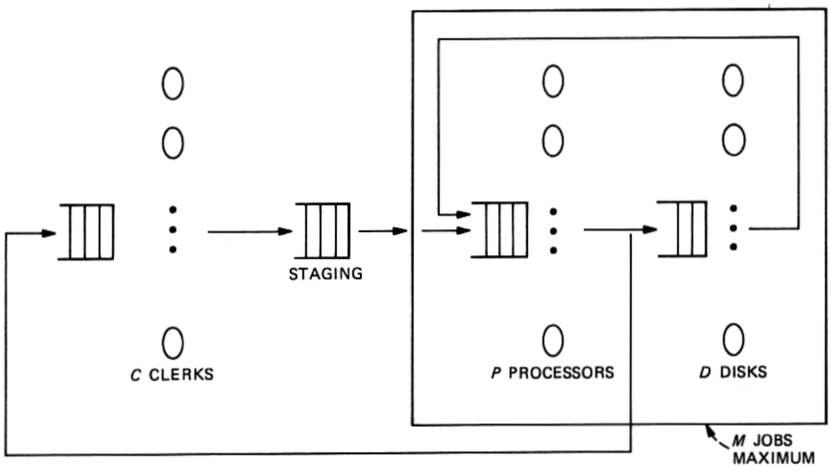


Fig. 2—Block diagram of the system queueing network.

2.2 Mathematical problem statement

The system state space is denoted by Ω, where

$$\Omega = \{(J_{\text{operator}}, J_{\text{stage}}, J_{\text{processor}}, J_{\text{disk}}) \mid J_{\text{operator}} + J_{\text{stage}} + J_{\text{processor}} + J_{\text{disk}} = C; J_{\text{processor}} + J_{\text{disk}} = \min[M, C - J_{\text{operator}}]\}.$$

At any instant of time, the system is in a state given by $(J_{\text{operator}}, J_{\text{stage}}, J_{\text{processor}}, J_{\text{disk}})$. Each component is integer valued, and refers to the number of jobs or transactions either in execution or waiting to be executed at that stage. The admissible state space is constrained because

(*i*) There are at most C jobs being worked on at any one time

(*ii*) The system can hold at most M jobs at any one time.

In a later section, we will show that the mean number of tasks in execution with operators, processors, and disks, averaged over a suitably long time interval, equals the mean throughput rate, λ, multiplied by the total mean execution time for that stage. This is summarized in the following equations, where $E(\cdot)$ denotes a time average of the argument:

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

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

$$E[\min(J_{\text{disk}}, D)] = \lambda T_{\text{disk}}.$$

In a later section, we show that λ can be upper and lower bounded, given only this information, in terms of model parameters, as follows:

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

$$\lambda_{\text{lower}} = \frac{C}{T_{\text{think}} + \dfrac{C}{\min(C, M, P)} T_{\text{processor}} + \dfrac{C}{\min(C, M, D)} T_{\text{disk}}}$$

$$\lambda_{\text{upper}} = \min\left[\frac{\min(C, M, P)}{T_{\text{processor}}}, \frac{\min(C, M, D)}{T_{\text{disk}}}, \frac{\min(C, M)}{T_{\text{processor}} + T_{\text{disk}}}, \frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}\right].$$

Each of the upper bounds on mean throughput rate has a physical interpretation, as follows:

(*i*) The processors are limiting the maximum mean throughput rate

$$\lambda_{\text{upper}} = \frac{\min(C, M, P)}{T_{\text{processor}}}$$

(*ii*) The disks are limiting the maximum mean throughput rate

$$\lambda_{\text{upper}} = \frac{\min(C, M, D)}{T_{\text{disk}}}$$

(*iii*) The clerks are limiting the maximum mean throughput rate

$$\lambda_{\text{upper}} = \frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}$$

(*iv*) Memory is limiting the maximum mean throughput rate

$$\lambda_{\text{upper}} = \frac{\min(C, M)}{T_{\text{processor}} + T_{\text{disk}}}.$$

The lower bound on mean throughput rate has the physical interpretation of executing jobs one at a time on each processor/disk pair. The upper bound on mean throughput rate is associated with the best possible concurrency, while the lower bound on mean throughput rate is associated with the worst possible parallelism. The upper bound on mean throughput rate yields a lower bound on mean delay; the lower bound on mean throughput rate yields an upper bound on mean delay:

$$\frac{C}{\lambda_{\text{upper}}} - T_{\text{think}} \le R \le \frac{C}{\lambda_{\text{lower}}} - T_{\text{think}}.$$

These bounds define an admissible or feasible region of operation and are plotted in Figs. 3 and 4 for the case of one processor and one disk, versus $C = M$.

Two regimes are evident: a *lightly loaded* regime, where the number

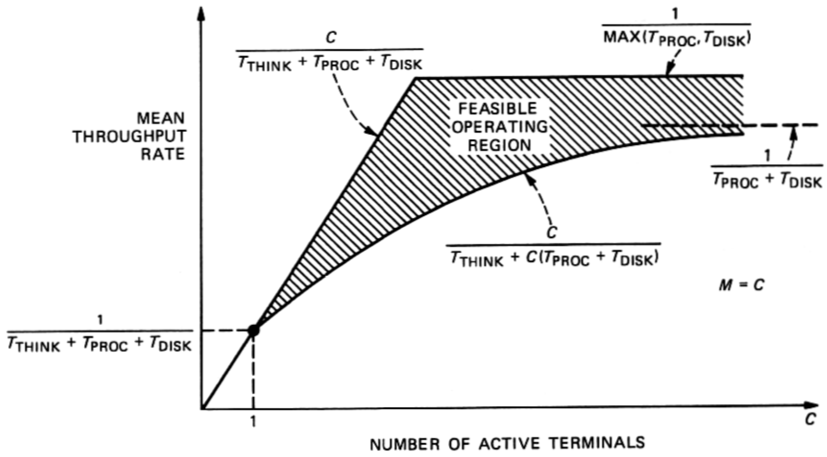


Fig. 3—Mean throughput rate versus number of active terminals.

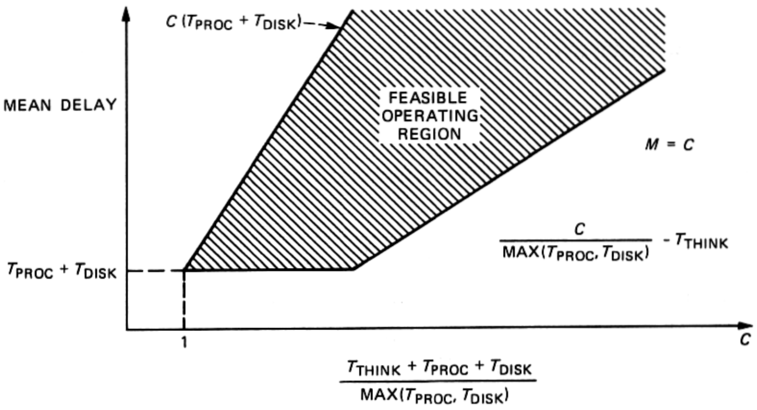


Fig. 4—Mean delay versus number of active terminals.

of *clerks* is directly proportional to the mean throughput rate and the mean delay is independent of the number of clerks, so the clerks are a bottleneck; and a *heavily loaded* regime, where the on-line computer communication system is the bottleneck, with the mean throughput rate independent of the number of clerks and the mean delay directly proportional to the number of clerks.

If we vary the number of clerks, there is a natural breakpoint between these two regimes:

$$\text{breakpoint number of operators} = \lambda_{\text{upper}}[T_{\text{processor}} + T_{\text{disk}} + T_{\text{think}}].$$

As long as the number of clerks is well below this breakpoint, the clerks and not the system will be limiting the mean throughput rate, and the mean delay per transaction will be approximately the mean delay to execute a transaction with no contention. With the number of clerks well above this breakpoint, the system will be limiting the mean throughput rate, and the mean delay per transaction will be well in excess of that time to execute a transaction with no contention. Analysis suggests measurements to determine where these two regimes lie; synthesis would involve choosing which regime we wish to operate in (remember, we will always have *some* bottleneck!) and designing the system accordingly.

2.3 Impact of memory constraint for one processor and one disk

Here are two possible scheduling policies for a system with one processor and one disk:

(*i*) Only one job is allowed into the system to be executed at any one time. This is called *single-thread* scheduling, and corresponds to the maximum number of jobs in the system equal to one, $M = 1$.

(*ii*) More than one job is allowed in the system to be executed at any one time. This is called *multiple-thread* scheduling, and corresponds to $M > 1$. For $M = 1$ we see

$$\frac{C}{T_{\text{think}} + C(T_{\text{processor}} + T_{\text{disk}})} \le \lambda_{\text{single thread}}$$

$$\le \min\left(\frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}, \frac{1}{T_{\text{processor}} + T_{\text{disk}}}\right) \quad M = 1$$

If we allow multiplexing of the processors and disks amongst transactions, then $M > 1$ is allowed, but now one or the other of the two serially reusable resources will become completely utilized for M sufficiently large:

$$\frac{C}{T_{\text{think}} + C(T_{\text{processor}} + T_{\text{disk}})} \le \lambda_{\text{multiple thread}}$$

$$\le \min\left(\frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}, \frac{1}{T_{\text{processor}}}, \frac{1}{T_{\text{disk}}}\right) \quad M > 1$$

In either case, the lower bound on mean throughput rate is identical, but the upper bound can be different, owing to different bottlenecks:

(*i*) The number of clerks is a bottleneck

$$\lambda_{\text{single thread}}, \lambda_{\text{multiple thread}} \le \frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}$$

(*ii*) The processor is a bottleneck

$$\lambda_{\text{multiple thread}} \le \frac{1}{T_{\text{processor}}}$$

(*iii*) The disk is a bottleneck

$$\lambda_{\text{multiple thread}} \le \frac{1}{T_{\text{disk}}}$$

(*iv*) Memory is limiting the maximum mean throughput rate

$$\lambda_{\text{single thread}} \le \frac{1}{T_{\text{processor}} + T_{\text{disk}}}.$$

Provided that the clerks or operators are not limiting the maximum mean throughput rate, the ratio of the two different upper bounds is an indication of the gain owing to scheduling or allowing more than one job inside the system at any one time:

$$\frac{\lambda_{\text{multiple thread}}}{\lambda_{\text{single thread}}} = \frac{T_{\text{processor}} + T_{\text{disk}}}{\max(T_{\text{processor}}, T_{\text{disk}})}.$$

For one processor and one disk, this gain owing to scheduling can be at most two, no matter what $T_{\text{processor}}$ or T_{disk} are! Moreover, this will only be achieved when $T_{\text{processor}}$ 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 $T_{\text{processor}}$, then the gain would be at most ten percent, and other factors ignored in this analysis may swamp this.

2.4 Asymptotics

We close with an investigation of asymptotic behavior of this system. 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 C. As the number of operators or clerks becomes large, $C \to \infty$, we see

$$\frac{1}{\dfrac{T_{\text{processor}}}{\min(P, M)} + \dfrac{T_{\text{disk}}}{\min(D, M)}} \le \lambda$$

$$\le \min\left(\frac{P}{T_{\text{processor}}}, \frac{D}{T_{\text{disk}}}, \frac{M}{T_{\text{processor}} + T_{\text{disk}}}\right), \quad C \to \infty.$$

This in turn will yield upper and lower bounds on mean response time:

$$\left(\frac{C}{\lambda_{\text{lower}}} - T_{\text{think}}\right) \to \infty \le R \le \left(\frac{C}{\lambda_{\text{upper}}} - T_{\text{think}}\right) \to \infty \quad C \to \infty.$$

In other words, the mean throughput rate lies between two *finite* bounds, while the mean response time is *infinite* (will exceed any finite threshold as we add more and more clerks).

A second type of asymptotic analysis is to fix the ratio of two parameters, and allow them both to become progressively larger, holding all other parameters constant. Here, a natural candidate is the ratio of the number of jobs over the mean think time per operator, which we denote by α, which is a measure of the *total* offered rate of submitting jobs:

$$\alpha \equiv \frac{C}{T_{\text{think}}},\ C \to \infty,\ T_{\text{think}} \to \infty.$$

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. In the literature, an analogous procedure is called passing from the so-called *finite source* to *infinite source* arrival process (e.g., see Ref. 18, pp. 102–3), granted certain additional distribution assumptions that we do *not* make here (e.g., see Ref. 18, pp. 80–2). If we fix α while allowing C, $T_{\text{think}} \to \infty$, we see

$$\frac{\alpha}{1+\alpha\left[\dfrac{T_{\text{processor}}}{\min(P,M)}+\dfrac{T_{\text{disk}}}{\min(D,M)}\right]} \le \lambda \le \min\left(\frac{P}{T_{\text{processor}}},\frac{D}{T_{\text{disk}}},\frac{M}{T_{\text{processor}}+T_{\text{disk}}}\right).$$

This in turn yields the following lower bound on mean delay:

$$R \ge \begin{cases} \infty & \alpha > \dfrac{1}{\max\left[\dfrac{T_{\text{processor}}}{\min(P,M)},\dfrac{T_{\text{disk}}}{\min(D,M)},\dfrac{M}{T_{\text{processor}}+T_{\text{disk}}}\right]} \\ T_{\text{processor}}+T_{\text{disk}} & \\ & \alpha < \dfrac{1}{\max\left[\dfrac{T_{\text{processor}}}{\min(P,M)},\dfrac{T_{\text{disk}}}{\min(D,M)},\dfrac{T_{\text{processor}}+T_{\text{disk}}}{M}\right]}. \end{cases}$$

The remaining case, an upper bound on mean delay or mean response time, is trivial:

$$R \le \infty,\ \alpha \textit{ fixed},\ C \to \infty,\ T_{\text{think}} \to \infty.$$

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

$$\alpha = \frac{1}{\max\left[\dfrac{T_{\text{processor}}}{\min(P,M)},\dfrac{T_{\text{disk}}}{\min(D,M)},\dfrac{T_{\text{processor}}+T_{\text{disk}}}{M}\right]}.$$

Inituitively we see that 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 for 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.*

Note that the mean throughput rate lies between two *finite* limits, while 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 this amount of information. This is well known in other types of queueing systems, such as the M/G/1 system (e.g., see Ref. 18, pp. 189–92), where the *mean* delay depends not only on the *first* moment of the service time distribution but also the *second* moment of the service time distribution: mean value information does *not* specify the mean delay in such systems by itself, but rather we need the actual distribution of service time to deal with this issue.

III. PROTOTYPE DIRECTORY ASSISTANCE SYSTEM CASE STUDY

Here is a case study in using these techniques. A prototype of an on-line transaction processing system was built to handle telephone number directory assistance queries. In a typical cycle of operation, a person at a terminal would

(*i*) Receive a query from a customer via voice telephone

(*ii*) Enter the given information into a computer terminal while talking to the customer

(*iii*) Wait for the system to respond with the answer to the query

(*iv*) Tell the customer over the voice telephone the reply

(*v*) Close out customer interaction

(*vi*) Wait to receive the next customer query.

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

Measurements on the prototype system in operation showed that

(*i*) The mean time spent by a person talking, reading, and thinking, denoted by T_{think}, was twenty seconds

(*ii*) The mean processor time per transaction was broken down into three sets of application programs

(a) The operator interface front-end programs consumed 180 milliseconds of processor time per query on the average

(b) The index manipulation application programs consumed 420 milliseconds of processor time per query on the average

(c) The data retrieval application programs consumed 330 milliseconds of processor time per query on the average

(d) Miscellaneous application programs that were invoked for accounting, system administration, and other purposes consumed one hundred and forty milliseconds (140 ms) per query

Hence, the total mean processor time per query, $T_{\text{processor}}$, was 1.07 seconds

(*iii*) The mean number of disk accesses per query was twenty six (26), with the disk capable of making one access every twenty five milliseconds (25 ms), which results in a mean time the disk is busy per query, denoted T_{disk}, of six hundred fifty milliseconds (650 ms).

The above measurements on total mean processor time and disk access counts were based on examining the mean resources required for one hundred different transactions to the system; the measurement error on the processor time was felt to be under ten milliseconds, while the measurement error on the number of disk accesses was felt to be under one access. For this level of analysis, the upper and lower mean value bounds on mean response time are given by

$$\max\left[T_{\text{processor}} + T_{\text{disk}}, \frac{C}{\max(T_{\text{processor}}, T_{\text{disk}})} - T_{\text{think}}\right] \le R \le C(T_{\text{processor}} + T_{\text{disk}}),$$

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

$$\frac{C}{T_{\text{think}} + C(T_{\text{processor}} + T_{\text{disk}})} \le \lambda \le \min\left[\frac{C}{T_{\text{think}} + T_{\text{processor}} + T_{\text{disk}}}, \frac{1}{\max(T_{\text{processor}}, T_{\text{disk}})}\right].$$

These bounds are plotted in Figs. 5 and 6, along with observed data gathered over an eight-hour time interval with twelve $C = 12$ operators and calculations based upon a closed queueing network model obeying product-form-type solution. The goodness of fit of the closed queueing network model to actual data was felt to be acceptable for the purposes at hand; the mean value lower bound on mean delay and upper bound on mean throughput rate were also felt to give an indication of performance limitations at an early stage of development, which the data gathering and refinement via a closed queueing network model only strengthened further. Note that the system is achieving a great deal of concurrency, because the actual mean throughput rate is much closer to the upper bound, not the single-thread lower bound. Similar observations hold for mean delay.

IV. PROTOTYPE DATA BASE ADMINISTRATION SYSTEM CASE STUDY

A transaction processing system administers the data base for a

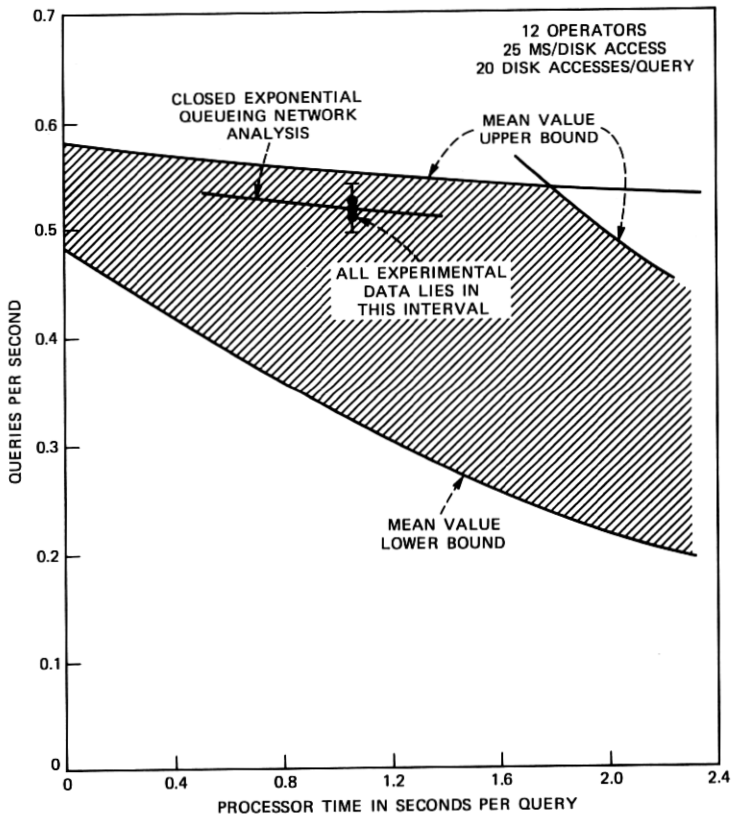


Fig. 5—Mean throughput rate (bounds and data) versus $T_{\text{processor}}$.

second system that switches telephone calls; hence this system is called a *front-end* system to the *back-end* telephone call switching system. Transactions involve additions, deletions, and changes to existing telephone numbers in the switching system files. A prototype system had a hardware configuration consisting of a single processor, a single disk controller, and a single disk spindle, with a fixed number of asynchronous terminals. This same prototype had an operating system to coordinate and schedule these resources, while application programs handled the transaction processing. The application programs were structured into a front end for handling operator terminal interactions, a data base management system, and a back-end communications system for interacting with various switching systems.

The same formulas for upper and lower mean value bounds on mean response time and mean throughput rate hold as in the previous example, except for a change in the numbers.

Two sets of measurements were gathered, one at the start of per-

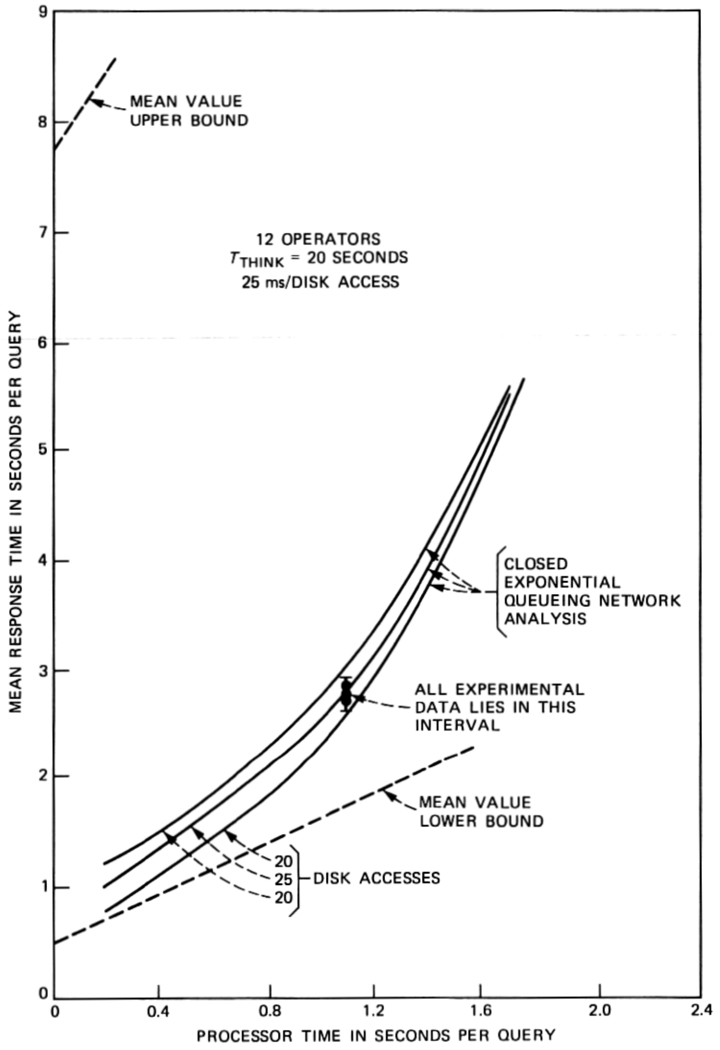


Fig. 6—Mean delay (bounds and data) versus $T_{\text{processor}}$.

formance analysis, labeled *initial* in Table I, and one after completing two months of performance analysis, which involved recoding application programs to take better advantage of operating system features, with the same hardware configuration, labeled *final* in Table I. Measurements were carried out in a controlled environment where the actual hardware, operating system, and application programs were used, but the operator behavior was simulated by a second computer. The behavior of each operator was modeled by a script, involving a time for reading, thinking, and typing, followed by a time waiting for the system to respond. After an initial startup transient the measurements of response time were quite predictable for all operations, with the measurement error being one second at most. Each operator submitted tens of jobs, and the results were averaged over all operators and all jobs, so the final statistics were felt to be statistically reproducible, to within a fraction of a second.

Figures 7 and 8 plot the mean value upper and lower bounds as well as data from these measurements for the mean response time and mean throughput rate as a function of number of operators. The goodness of fit to mean value bounds was felt to be acceptable for the purpose. Unlike the first case study, the data here clearly shows that a great deal of fluctuation was encountered in system operation under load: for the initial system, the fluctuations were so great that the system apparently was always executing only one transaction at a time, while for the final system, as load built up, the system effectively moved from a regime of two tasks making use of both serially reusable resources to a regime where only one task at a time was in execution. This is in contrast to the other set of data, where the system is always achieving a great deal of concurrency under load. A closed exponential

Table I—Prototype system measurements

Quantity	Initial (seconds)	Final (seconds)
T_{think}	15.0	15.0
$T_{\text{processor}}$	8.2	3.5
T_{disk}	5.0	0.5

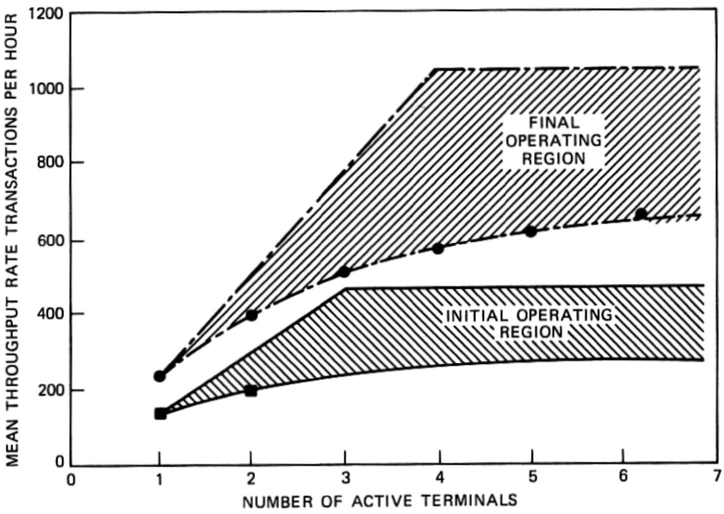


Fig. 7—Mean throughput rate (bounds and data) versus number of active clerks.

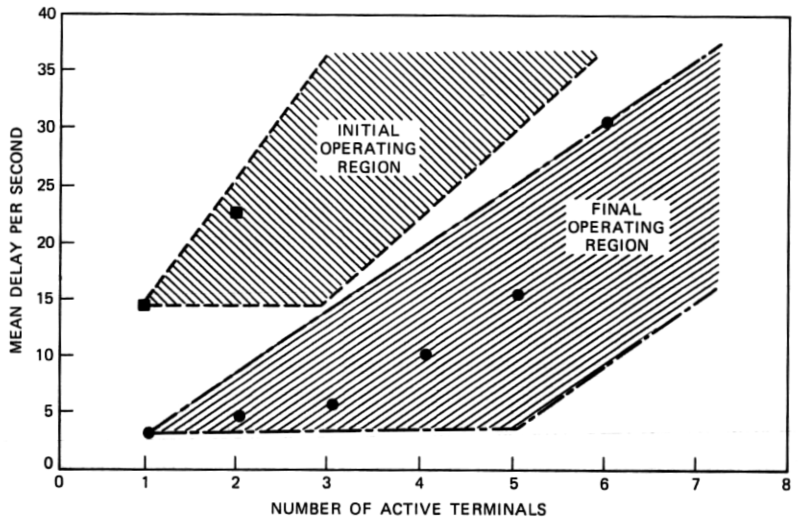


Fig. 8—Mean delay (bounds and data) versus number of active clerks.

queueing network model of this system would predict behavior that closely tracked the upper bound on mean throughput rate and lower bound on mean delay, and would simply not allow for sufficient fluctuation to drive operation into a mode of operation of executing one task at a time. In fact, this suggested a problem with *memory management* that was forcing the system into this mode of operation; an obvious test that was not carried out owing to lack of time was to add more main memory to see if more concurrency might be achieved.

V. A MODEL OF FLOW CONTROL OVER A SINGLE LINK

An on-line communications system consists of operators at terminals who send messages to one another. The system consists of a transmitter and a receiver, with communication channels connecting the transmitter and receiver. The receiver is capable of buffering only a maximum number of messages at any one time, which is a memory constraint.

5.1 Model description

A communications system is composed of a transmitter processor, a receiver processor, a set of buffers each capable of holding one message at the receiver, and a noiseless communications link. Here are the steps involved in sending a message from the transmitter to the receiver:

(*i*) The transmitter processes a message. This step has a mean duration T_{trans} at the transmitter, and it requires both the transmitter *and* a buffer at the receiver.

(*ii*) The message is sent over the link from the transmitter to the receiver. This step has a mean duration $T_{\text{trans-rec}}$.

(*iii*) The receiver processes the message. This step has a mean duration T_{rec}.

(*iv*) An acknowledgment of correct receipt of the message is sent from the receiver to the transmitter. This step has a mean duration $T_{\text{rec-trans}}$. At the start of this step, the receiver marks the buffer free.

(*v*) The transmitter processes the acknowledgment. This step has a mean duration of T_{ack}.

At the end of this step, the transmitter marks the buffer free.

We assume from this point on that the time required by the transmitter to process the acknowledgment is zero. Figure 9 shows a hardware block diagram of the system. Figure 10 shows a queueing network block diagram of the system. The system state is denoted by Ω where

$$\Omega = \{(J_{\text{trans}}, J_{\text{trans-rec}}, J_{\text{rec}}, J_{\text{rec-trans}}) \mid J_{\text{trans}} + J_{\text{trans-rec}} + J_{\text{rec}} + J_{\text{rec-trans}} = B\}$$

At any instant of time, the system is in a state given by a four tuple, $(J_{\text{trans}}, J_{\text{trans-rec}}, J_{\text{rec}}, J_{\text{rec-trans}})$, where each component is nonnegative and integer valued, and the state space constraint is obeyed.

The mean throughput rate is denoted by λ. The mean number of jobs in execution in the transmitter and in the receiver equals the mean throughput rate multiplied by the total mean execution time, as shown in a later section. We denote by $E(\cdot)$ the time average of the argument, and write:

$$\lambda T_{\text{trans}} = E[\min(J_{\text{trans}}, P_{\text{trans}} = 1)]$$

$$\lambda T_{\text{rec}} = E[\min(J_{\text{rec}}, P_{\text{rec}} = 1)].$$

Our goal is to find upper and lower bounds on mean throughput rate, subject to meeting state space constraints.

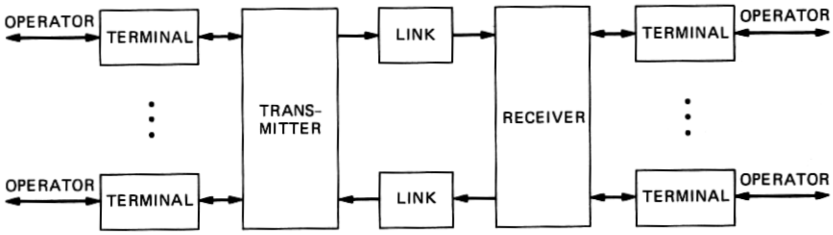


Fig. 9—Hardware block diagram.

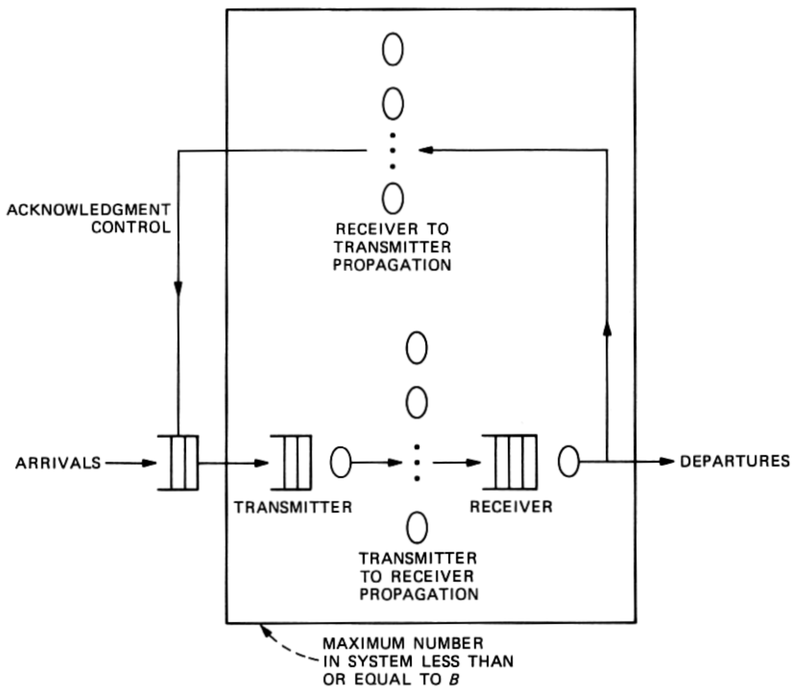

ACKNOWLEDGMENT CONTROL
RECEIVER TO TRANSMITTER PROPAGATION
ARRIVALS
TRANSMITTER
RECEIVER
DEPARTURES
TRANSMITTER TO RECEIVER PROPAGATION
MAXIMUM NUMBER IN SYSTEM LESS THAN OR EQUAL TO *B*

Fig. 10—Queueing network model.

In a later section, we show

$$\frac{B}{BT_{\text{trans}} + T_{\text{trans-rec}} + BT_{\text{rec}} + T_{\text{rec-trans}}} = \lambda_{\text{lower}} \leq \lambda$$

$$\lambda \leq \lambda_{\text{upper}} = \min\left(\frac{1}{T_{\text{trans}}}, \frac{1}{T_{\text{rec}}}, \frac{B}{T_{\text{trans}} + T_{\text{trans-rec}} + T_{\text{rec}} + T_{\text{rec-trans}}}\right).$$

The physical interpretation of the upper bound on mean throughput rate is as follows

(*i*) The transmitter is the bottleneck

$$\lambda_{\text{upper}} = \frac{1}{T_{\text{trans}}}$$

(*ii*) The receiver is the bottleneck

$$\lambda_{\text{upper}} = \frac{1}{T_{\text{rec}}}$$

(*iii*) The number of buffers is the bottleneck

$$\lambda_{\text{upper}} = \frac{B}{T_{\text{trans}} + T_{\text{trans-rec}} + T_{\text{rec}} + T_{\text{rec-trans}}}.$$

The physical interpretation of the lower bound is that at most one message at a time is being handled by the system.

Figures 11 through 13 plot these upper and lower bounds, as well as the results of an exponential queueing network analysis,[1-3] for the special case where

$$T_{\text{trans}} = T_{\text{rec}},\ T_{\text{trans-rec}} = T_{\text{rec-trans}}$$

for three different cases, where the propagation delay is much smaller, equal, and much larger than the mean processing time at either end of the link. The fraction of time the queueing network model predicts the system to be in state $\boldsymbol{J}$ is denoted by $\pi(\boldsymbol{J})$, where

$$\pi(\boldsymbol{J}) = \frac{1}{G} T_{\text{trans}}^{J_{\text{trans}}} \frac{T_{\text{trans-rec}}^{J_{\text{trans-rec}}}}{J_{\text{trans-rec}}!} T_{\text{rec}}^{J_{\text{rec}}} \frac{T_{\text{rec-trans}}^{J_{\text{rec-trans}}}}{J_{\text{rec-trans}}!}.$$

The system partition function denoted G is chosen to normalize the probability distribution:

$$\sum_{\boldsymbol{J} \in \Omega_B} \pi(\boldsymbol{J}) = 1.$$

5.2 Negligible link propagation delay

We now restrict attention to the special case where the propagation delay is negligible compared to the processing at either end of the link, from this point on. For one buffer, the mean throughput rate is upper bounded by

$$\lambda \le \lambda_{\text{upper}} = \frac{1}{T_{\text{trans}} + T_{\text{trans-rec}} + T_{\text{rec}} + T_{\text{rec-trans}} + T_{\text{ack}}} = \frac{1}{T_{\text{trans}} + T_{\text{rec}}}.$$

There is no concurrency or parallel execution of messages, and the

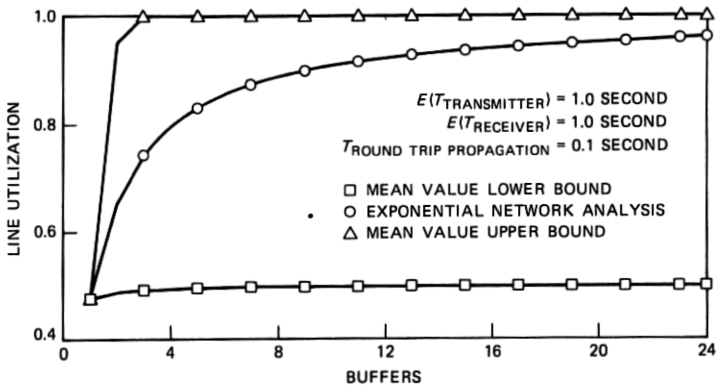


Fig. 11—Line utilization vs. number of buffers ($T_{\text{trans-rec}} = T_{\text{trans}}/10$).

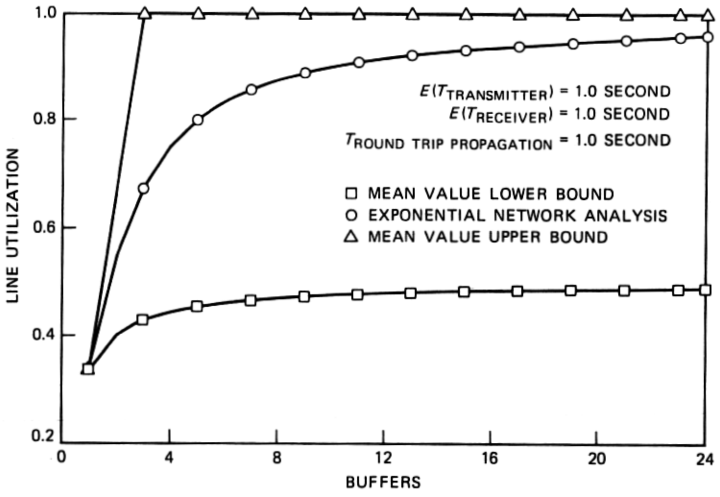


Fig. 12—Line utilization versus number of buffers ($T_{\text{trans-rec}} = T_{\text{trans}}$).

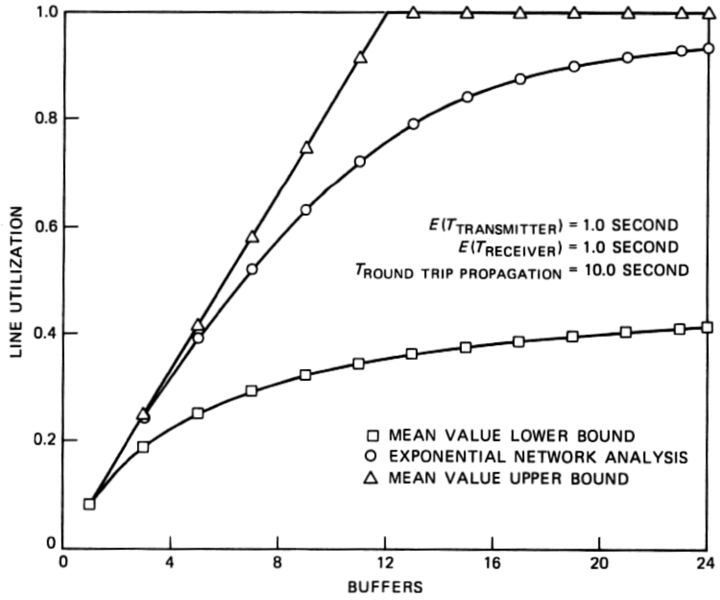


Fig. 13—Line utilization versus number of buffers ($T_{\text{trans-rec}} = 10\ T_{\text{trans}}$).

total time required for message handling is the sum of the individual steps.

For more than one buffer, this will yield an upper bound on the mean throughput rate of simply B times the mean throughput rate for one buffer:

$$\lambda \le \lambda_{\text{upper}} = \frac{B}{T_{\text{trans}} + T_{\text{trans-rec}} + T_{\text{rec}} + T_{\text{rec-trans}} + T_{\text{ack}}} = \frac{B}{T_{\text{trans}} + T_{\text{rec}}}.$$

On the other hand, as the number of messages increases, then either the transmitter or the receiver (or both) will become completely busy, yielding different upper bounds on mean throughput rate:

(*i*) The transmitter is a bottleneck

$$\lambda \le \lambda_{\text{upper}} = \frac{1}{T_{\text{trans}} + T_{\text{ack}}} = \frac{1}{T_{\text{trans}}}$$

(*ii*) The receiver is a bottleneck

$$\lambda \le \lambda_{\text{upper}} = \frac{1}{T_{\text{rec}}}.$$

Combining all this, we see

$$\lambda \le \lambda_{\text{upper}} = \min\left(\frac{1}{T_{\text{trans}} + T_{\text{ack}}}, \frac{1}{T_{\text{receiver}}}, \frac{B}{T_{\text{trans}} + T_{\text{trans-rec}} + T_{\text{rec}} + T_{\text{rec-trans}} + T_{\text{ack}}}\right)$$

$$\lambda \le \lambda_{\text{upper}} = \min\left(\frac{1}{T_{\text{trans}}}, \frac{1}{T_{\text{rec}}}, \frac{B}{T_{\text{trans}} + T_{\text{rec}}}\right).$$

Increasing the number of buffers from one to two, $B = 1$ to $B = 2$ always increases the maximum mean throughput rate, and now we see

$$\lambda \le \lambda_{\text{upper}} = \min\left(\frac{1}{T_{\text{trans}}}, \frac{1}{T_{\text{rec}}}\right) \qquad B > 1.$$

Furthermore, this increase is maximized for $T_{\text{trans}} = T_{\text{rec}}$, and then the upper bound *doubles* in going from one buffer to more than one buffer. Why is this so? By having more than one buffer, both the transmitter and receiver can simultaneously be filling and emptying a buffer, allowing greater concurrency or parallelism compared with the single-buffer case. We also note that allowing more than two buffers, e.g., *infinite* buffers, will not increase the upper bound on the maximum mean throughput rate any further. This is because there are only two serially reusable resources, a transmitter and a receiver, so once they are concurrently busy, no further gains can be achieved.

For the lower bound on mean throughput rate, we see that

$$\lambda \ge \lambda_{\text{lower}} = \frac{B}{BT_{\text{trans}} + BT_{\text{rec}}} = \frac{1}{T_{\text{trans}} + T_{\text{rec}}},$$

which is identical to the upper bound for $B = 1$. Why is this so? There may be significant fluctuation about the mean values shown above, and in the limit of one big swing about the mean value all of the messages will pile up at one stage in the network and nothing will be transmitted until buffers become available.

5.3 Impact of fluctuations

We now examine one special case of this problem in detail, where $T_{\text{trans-rec}} = T_{\text{rec-trans}} = T_{\text{ack}} = 0$, and we wish to study the impact of fluctuations about mean values on system performance. We assume the transmitter processing times are sequences of independent identically distributed exponential random variables with mean T_{trans}. We assume the receiver processing times are sequences of independent identically random variables with common hyperexponential distribution $G_{\text{receiver}}(X)$:

$$G_{\text{receiver}}(X) = (1 - \alpha) + \alpha(1 - e^{-X\mu_{\text{rec}}}).$$

In words, a fraction $1 - \alpha$ will require zero processing time at the receiver, while a fraction α will require an exponentially distributed amount of processing time with mean $1/\mu_{\text{rec}}$. The parameter α gives us an additional degree of freedom to model fluctuations in the receiver processing times. For this case, we choose to fix the *squared coefficient of variation* denoted by C^2, which for the random variable X is defined as the ratio of the variance to square of the mean (the standard deviation, measured in units of mean value, squared):

$$\text{squared coefficient of variation} = \frac{\text{variance}(X)}{E^2(X)} \equiv C^2.$$

When this is zero, the variance is zero, and there is zero fluctuation about the mean. When this is one, we have an exponential distribution, where the standard deviation equals the mean. When this is greater than one, the standard deviation is greater than the mean. For this particular case, we see $0 < \alpha \leq 1$ and hence

$$C^2 = \frac{2}{\alpha} - 1 \geq 1.$$

If the mean is fixed but α is varied from one (the exponential distribution case, where the fluctuations are the order of the mean) to zero (increasing fluctuations about the mean), with most jobs taking zero time but a few taking a very long time, we can gain insight into the impact on performance. Since we have fixed the squared coefficient of variation, the mean is also fixed, since

$$T_{\text{rec}} = \frac{\alpha}{\mu_{\text{rec}}}.$$

The distribution of the number in the receiver subsystem at the completion of processing at the receiver of a message is denoted by $F(K)$, $K = 0, \cdots, B$. If none are left behind, then the mean time to the next completion epoch is $T_{\text{trans}} + T_{\text{rec}}$. If more than zero are left behind at the receiver, then the mean time to the next completion epoch is T_{rec}. The mean throughput rate is given by

$$\lambda = \frac{1}{F(0)(T_{\text{trans}} + T_{\text{rec}}) + [1 - F(0)](T_{\text{rec}})} = \frac{1}{F(0)T_{\text{trans}} + T_{\text{rec}}}.$$

Once we find the distribution of the number of messages in the system at completion epochs, we are done. However, this is a well-known result (see Ref. 18, pp. 235–40), and we merely summarize the known formulas here for the sake of completeness:

$$F(0) = \frac{Q(0)}{\sum_{K=0}^{B-1} Q(K)}.$$

The terms $Q(K)$, $K = 0, \cdots, B - 1$ are given implicitly via the following moment-generating function $\zeta(X)$:

$$\zeta(X) = \sum_{K=0}^{\infty} Q(K)X^K = \frac{(1 - X)E[e^{-\lambda(1-X)T_{\text{rec}}}]}{E[e^{-\lambda(1-X)T_{\text{rec}}}] - X}.$$

Illustrative numerical results are plotted in Figs. 14 through 16 assuming the transmitter and receiver service times are independent, identically distributed, exponential random variables. We note that for the special case where $T_{\text{rec}} = T_{\text{trans}} = 1$, the mean throughput rate is given by

$$\lambda = \frac{1 + \dfrac{2(B + 1)}{C^2 + 1}}{2 + \dfrac{2(B - 1)}{C^2 + 1}} \qquad C^2 \geq 1.$$

Here we see that as $C^2 \to \infty$ that the mean throughput rate approaches the lower bound of 1/2 arbitrarily close, i.e., there is no concurrency or gain in going to more than one buffer if the fluctuations are too great. On the other hand, as $B \to \infty$ for C^2 fixed, the mean throughput rate approaches one, which is the best possible. The numerical plots show in which regime which phenomenon (the fluctuations or the buffering and concurrency) dominates the actual mean throughput rate. The impact of speed mismatch (i.e., as the transmitter and receiver mean message execution times start to differ) tends to swamp the impact of fluctuations: the greater the speed mismatch, the greater concurrency is achieved, because the exact mean throughput rate approaches the upper bound closer and closer as the speed mismatch increases be-

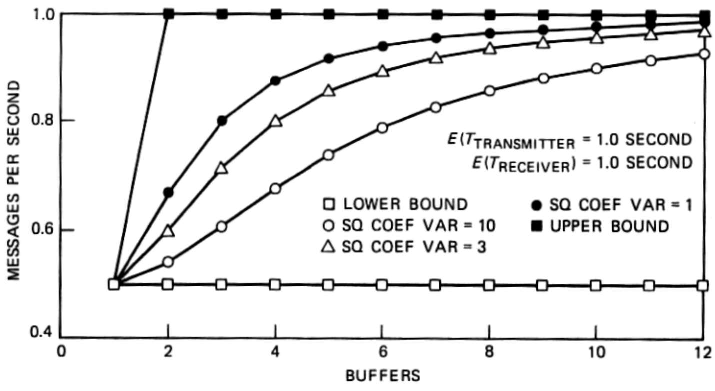


Fig. 14—Mean throughput rate vs. number of buffers for a closed queueing network model ($T_{trans} = 1.0$ and $T_{rec} = 1.0$).

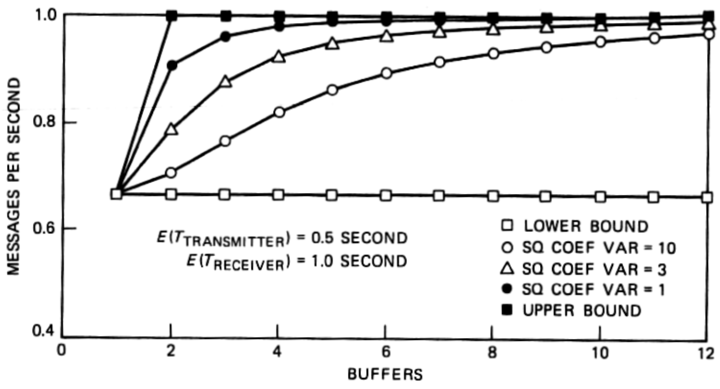


Fig. 15—Mean throughput rate vs. number of buffers for a closed queueing network model ($T_{trans} = 0.5$ and $T_{rec} = 1.0$).

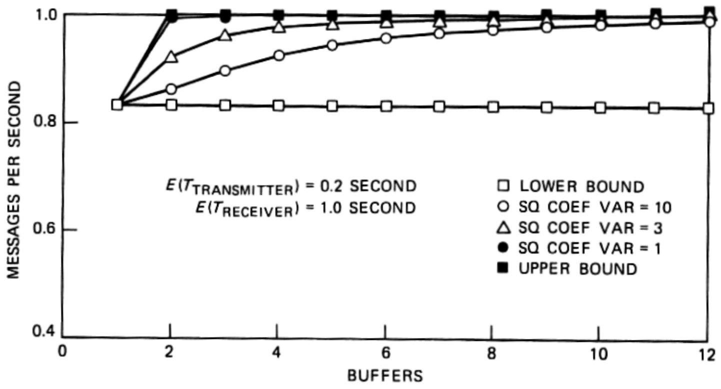


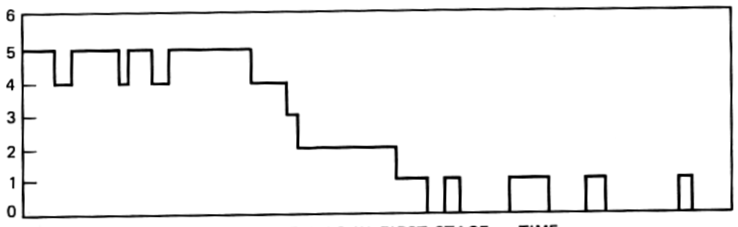
Fig. 16—Mean throughput rate vs. number of buffers for a closed queueing network model ($T_{trans} = 0.2$ and $T_{rec} = 1.0$).

tween transmitter and receiver. Note that the upper bound on mean throughput rate corresponds to a squared coefficient of variation of zero, while the lower bound corresponds to a squared coefficient of variation that becomes infinite.

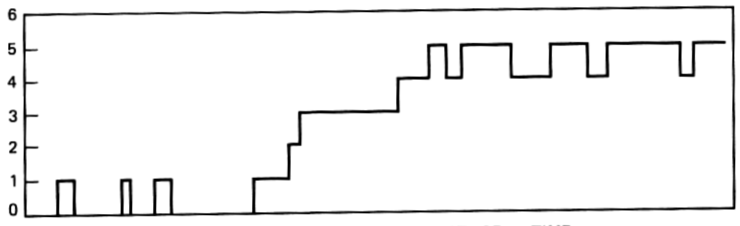
We now discuss this phenomenon in more detail, because the formulas give only one way of understanding this model. Figure 17 shows a sample path generated from a simulation of the model, for a total number of five jobs in the system. In the initial part of the simulation, the first stage fluctuates between four and five jobs, while the second stage fluctuates between zero and one job; in the final part of the simulation, the situation is reversed; after sufficiently long time, we would return to the first case. When most of the jobs are at one stage, the mean throughput rate is roughly the reciprocal of the time to execute one job from start to finish, and there is no concurrency. The other cases, where there are multiple jobs at each stage, are transient and the system spends relatively little time in these states.

The analysis developed above can make these intuitive notions more precise. For example, the mean fraction of time that there are zero jobs at the receiver is

$$\text{fraction of time zero jobs at receiver} = \frac{F(0)}{F(0) + \dfrac{T_{\text{rec}}}{T_{\text{trans}}}},$$



(a)



(b)

Fig. 17—Sample path generated from a simulation of the model for (a) first stage vs. time and (b) second stage vs. time.

while the fraction of time all the jobs are at the receiver is

$$\text{fraction of time all jobs at receiver} = 1 - \frac{1}{F(0) + \dfrac{T_{\text{rec}}}{T_{\text{trans}}}}.$$

As we allow $\alpha \to 0$, i.e., as the fluctuations and squared coefficient of variation become larger, while the mean time spent at the transmitter and receiver stay fixed, we see that the sum of these two fractions can be made *arbitrarily* close to one, which is what the simulation result in Fig. 17 shows. At the same time, we see that the mean sojourn time in the state where the receiver is empty is given by

mean sojourn time in idle receiver state

$$= \sum_{K=1}^{\infty} (1-\alpha)^{K-1} \alpha K T_{\text{trans}} = \frac{T_{\text{trans}}}{\alpha} \to \infty \quad \alpha \to 0.$$

Put differently, if one were to measure the operation of this system, the system might be in the receiver idle state for the entire duration of the observation process, and the other state of the receiver having all jobs (which will also become successively longer and longer as $\alpha \to 0$) will never be observed, or vice versa! In Fig. 17, this would correspond to gathering data in the first part of the simulation, never in the second part, or vice versa.

5.4 Queueing network analysis for negligible propagation delay

In a later section, we show that the mean throughput rate is upper and lower bounded by

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

$$\lambda_{\text{lower}} = \frac{1}{T_{\text{trans}} + T_{\text{rec}} + \max(T_{\text{trans}}, T_{\text{rec}})}$$

$$\lambda_{\text{upper}} = \frac{1}{T_{\text{trans}} + T_{\text{rec}} + 1/2(T_{\text{trans}} + T_{\text{rec}})}.$$

The mean value bounds, the queueing network upper and lower bounds, and exact queueing network analysis mean throughput calculations are all plotted in Figs. 18 through 20 for $T_{\text{rec}} = 1.0$ and $T_{\text{trans}} = 1.0, 0.5, 0.2$. The queueing network bounds are identical to the exact analysis when the transmitter and receiver execute messages at the same rate. When the transmitter becomes faster than the receiver, the bounds and exact analysis tend to track the upper bound on mean throughput rate; in other words, the speed mismatch is of greater importance than the impact of fluctuations.

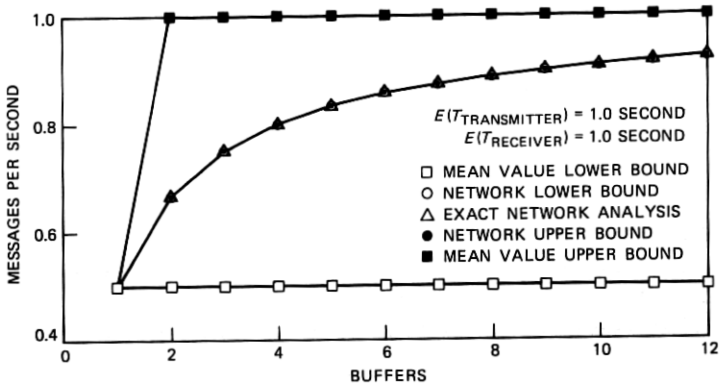


Fig. 18—Mean throughput rate vs. number of buffers for zero propagation time ($T_{\text{trans}} = 1.0$ and $T_{\text{rec}} = 1.0$).

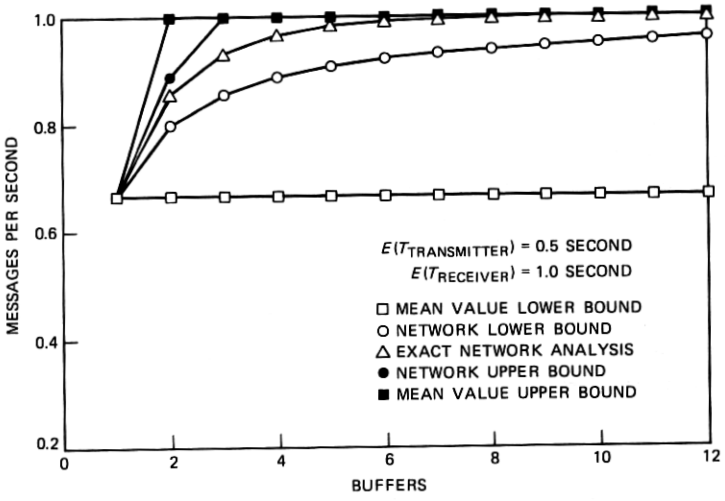


Fig. 19—Mean throughput rate vs. number of buffers for zero propagation time ($T_{\text{trans}} = 0.5$ and $T_{\text{rec}} = 1.0$).

5.5 Experimental data

To test predictions of this analysis against actual operations, a series of experiments were carried out to determine the mean maximum throughput rate of a data communications link constructed with two computers, one transmitting and one receiving, over a data link where the link propagation time was negligible compared to the data communications processing at either end of the link or the data transmission time of a packet over this link. The test described here involved sending 51,200 bytes of data over a 9600-b/s data link; similar results were found for a 1200-b/s data link. The source data were encoded

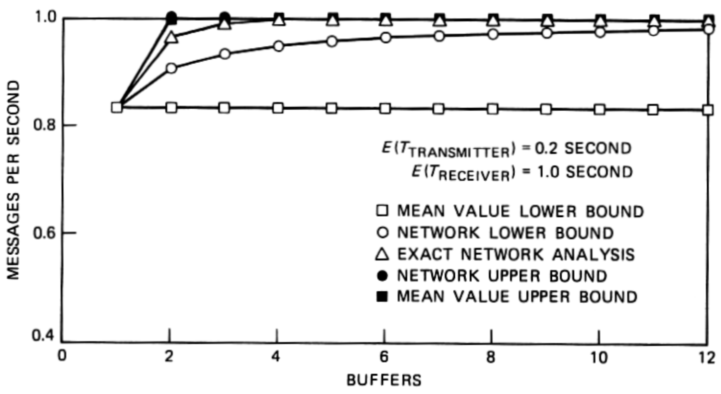


Fig. 20—Mean throughput rate vs. number of buffers for zero propagation time (T_{trans} = 0.2 and T_{rec} = 1.0).

into packets containing either 32, 64, 128, or 256 bytes (one byte equals eight bits) of data. The system and numerous other details of the experiment will be described elsewhere in a different report.

We wish to test the gain in going from start-stop or single buffering to double buffering and to greater than double buffering; our previous analysis assumes that a mean value of data communications processing time at the transmitter and receiver adequately characterizes the system performance.

The experiment involved simply measuring the time required to transmit 51,200 bytes of data over each link for each size packet. No processing was done on the data at either the transmitter or receiver other than to do the data communications processing required for correct operation. The transmitter and receiver processes resided in the same PDP 11/45 computer with a *UNIX**-like operating system environment.

Table II summarizes the results of that experiment. The time required to send each of 51,200 bytes of data plus two additional bits (for parity and control) over a serial 9600-b/s data link is 53.3 seconds; thus, the data link transmission speed and not the transmitter or receiver is limiting data flow here. This can also be seen directly by noting that the link utilization is approaching one hundred per cent in Table II. This table shows that double buffering at the receiver offers substantial improvement in mean message throughput over single buffering, and there is no apparent advantage in terms of throughput in choosing a receiver buffer larger than two (e.g., seven was tried). Finally, this suggests that for this purpose this level of analysis is

* Trademark of Bell Laboratories.

Table II—PDP 11/45 loop-around experiment—maximum mean throughput rate for transmission of 51,200 bytes over 9600-b/s data link

Number of Buffers	Packet Size (bytes)	Time (seconds)	Maximum Throughput (b/s)	Link Utilization (percent)
1	32	160.0	3200	33
1	64	125.0	4096	43
1	128	90.0	5688	59
1	256	80.0	6400	67
2	32	80.0	6400	67
2	64	58.5	8752	91
2	128	55.5	9225	96
2	256	55.0	9309	97
7	32	64.0	8000	83
7	64	58.0	8827	92
7	128	55.0	9309	97
7	256	54.5	9412	98

appropriate, i.e., that other phenomena that are present are in fact negligible for these purposes, as shown by the data.

VI. CONCLUSIONS

A performance study of an computer communication system may be carried out in at least one of three ways:

(*i*) Mean value analysis, as described here[14,15]
(*ii*) Jackson queueing network analysis[3]
(*iii*) Discrete event simulation model.[9–11]

In this paper we have demonstrated the ability of the mean value analysis to present a clear picture of the dependence of computer communication system performance on the values of the model parameters. The mean value analysis is a simple, flexible, inexpensive approach to performance analysis and should always be used, even if it is required to supplement the analysis with one or both of the other techniques. The other approaches quantify the impact of fluctuations about mean values on performance, refining the mean value analysis.

The utility or validity of any of these approaches cannot be judged in the abstract: whichever approach or combination of methods is most appropriate must be judged in terms of the data gathering and measurements, and how the data is used to draw inferences concerning cause and effect phenomena, coupled with the spectrum of practical feasible alternatives. The mean value approach presented here is simply one tool for carrying out this complex decision-making process.

VII. ACKNOWLEDGMENT

The authors are indebted to numerous colleagues throughout Bell Laboratories for stimulating this work, people involved with the development of over one hundred diverse systems who have shared with us their insight into design and analysis considerations over the past five years. The authors are also indebted to Bell Laboratories for providing such a stimulating work environment.

REFERENCES

1. L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, Chapters Five and Six, New York: Wiley Interscience, 1976.
2. F. P. Kelly, "*Networks of Queues*," Advances in Applied Probability, *8* No. 2 (June 1976), pp. 416–32.
3. F. P. Kelly, *Reversibility and Stochastic Networks*, Chichester: Wiley, 1979.
4. C. Sauer and K. Chandy, *Computer Systems Performance Modeling*, Englewood Cliffs, NJ: Prentice Hall, 1981.
5. M. Phister, Jr. *Data Processing: Technology and Economics*, Second Edition, Bedford, MA: Digital Press, 1979.
6. A. F. Shackil, "*Design Case History: Wang's Word Processor*," IEEE Spectrum, *18*, No. 8 (August 1981), pp. 29–33.
7. G. H. Engel, J. Groppuso, R. A. Lowenstein, and W. G. Traub, "*An Office Communications System*," IBM Systems Journal, *18*, No. 4 (1979), pp. 402–31.
8. R. P. Uhlig, D. J. Farber, and J. H. Bair, *The Office of the Future: Communication and Computers*, Amsterdam: North-Holland, 1979.
9. G. S. Fishman, *Concepts and Methods in Discrete Event Digital Simulation*, New York: Wiley, 1973.
10. G. S. Fishman, *Principles of Discrete Event Simulation*, New York: Wiley, 1978.
11. H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*, Reading, MA: Addison Wesley, 1978.
12. J. M. Holtzman, "*The Accuracy of the Equivalent Random Method with Renewal Inputs*," B.S.T.J., *52*, No. 9 (September 1973), pp. 1673–9.
13. A. E. Eckberg, "*Sharp Bounds on Laplace-Stieltjes Transforms, with Applications to Various Queueing Problems*," Mathematics of Operations Research, *2*, No. 2 (1977), pp. 135–42.
14. K. Omahen, "*Capacity Bounds for Multiresource Queues*," J.A.C.M., *24* (1977), pp. 646–63.
15. P. J. Denning and J. P. Buzen, "*The Operational Analysis of Queueing Network Models*," Computing Surveys, *10*, No. 3 (1978), pp. 225–61.
16. S. S. Lam, "*Queuing Networks with Population Size Constraints*," IBM J. Research and Development, *21*, No. 7 (July 1977), pp. 370–8.
17. J. Zahorjan, K. C. Sevick, D. L. Eager, and B. Galler, "*Balanced Job Bound Analysis of Queueing Networks*," Communications of the ACM, *25*, No. 2 (February 1982), pp. 134–41.
18. R. B. Cooper, *Introduction to Queueing Theory*, Second Edition, New York: North Holland, 1981.
19. J. D. C. Little, "*A Proof of the Queueing Formula $L = \lambda W$*," Operations Research, *9* (1961), pp. 383–7.
20. W. S. Jewell, "*A Simple Proof of: $L = \lambda W$*," Operations Research, *15* (1967), pp. 1109–16.
21. R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Reading, MA: Addison-Wesley, 1967; Little's formula, pp. 18–19.
22. W. L. Smith, "*Renewal Theory and Its Ramifications*," J. Royal Statistical Society (Series B), *20*, No. 2 (February 1958), pp. 243–302.
23. G. P. Klimov, *An Erogodic Theorem for Regenerating Processes*, Theory of Probability and Its Applications, *21*, No. 2 (June 1976), pp. 392–5.

APPENDIX A

Little's Law

Jobs enter a system, spend time within the system, and depart. The system attributes of interest here are:

(*i*) $L(t)$ denotes the number of jobs in the system at time t

(*ii*) $C(t)$ denotes the number of completions in the time interval $(0, t]$

(*iii*) Every job that enters the system leaves the system.

Our goal is to relate the mean throughput rate of jobs, the mean time a job spends in the system, and the mean number of jobs in the system.

The total mean time spent by all the jobs in the system is simply the area underneath the function $L(t)$:

$$\text{total mean time in system by all jobs} = \int_0^t L(\tau)d\tau.$$

The total mean time spent in the system by any one job is given by

$$\text{mean time in system per job in } (0, t] \equiv \frac{\int_0^t L(\tau)d\tau}{C(t)}.$$

We multiply and divide by t as follows:

$$\text{mean time in system per job} = \frac{\int_0^t L(\tau)d\tau}{t} \times \frac{1}{C(t)/t}.$$

The first term is simply the mean number of jobs in the system, averaged over a time interval of duration t:

$$\text{mean number of jobs in system in } (0, t] \equiv \frac{\int_0^t L(\tau)d\tau}{t}.$$

The second term is simply the mean throughput rate:

$$\text{mean throughput rate in } (0, t] \equiv \frac{C(t)}{t}.$$

Hence, we have shown that the mean number of jobs inside the system equals the mean throughput rate multiplied by the mean time in system per job, all over an interval of duration $(0, t]$:

$$\text{mean number of jobs in system in } (0, t]$$
$$= \text{mean throughput rate in } (0, t] \times \text{mean time in system/job in } (0, t].$$

If the observation interval becomes infinite, $t \to \infty$, and the mean values defined here in fact stabilize and do not fluctuate, then we have what is called Little's Law.[19-21] These other derivations rely on averaging over an ensemble of equally likely experiments, and draw on deep results from the theory of stochastic processes;[22,23] the difficulty is in showing that the limits in fact exist in a meaningful mathematical sense. Here we focus exclusively on time averages of quantities of interest, since these can be readily measured.

We close with an application of this result that we will use in the following section. Jobs arrive for processing by a system. Each job requires a total mean amount of service T. The system consists of a single queue feeding P identical processors. At any given instant of time, there are J jobs in the system, either running or waiting to run. The mean throughput rate of jobs is denoted by λ.

We now restrict attention to a subsystem of the total system, the subsystem of jobs in execution. Since we have P processors, the number of jobs in execution at any instant of time is $\min[J, P]$. Hence, we see that the mean number of jobs in execution, averaged over a time interval, equals the mean throughput rate multiplied by the mean time a job spends in execution:

$$\text{mean number of jobs in execution} \equiv E[\min(J, P)] = \lambda T.$$

The actual service pattern of the job is not of interest here: a job may actually consist of a series of steps with different processing at each step, and at the conclusion of each step of execution the job returns to the end of the queue (or to some point in the queue based upon the step) until it is completely executed.

APPENDIX B

Mean Value Analysis

We now present the mathematical analysis to justify assertions in earlier sections. The model we deal with is a system handling only one type of job or transaction. Each transaction consists of one or more steps; at each step, a given amount of a serially reusable resource is required for a given time interval. A resource is any entity that is required for subsequent execution of a transaction; examples of physical hardware resources are processors, memory, disk spindles, disk controllers, communication links, local backplane buses and so forth; example of logical software resources are files, tables, messages, semaphores, and so forth. Here the first step of each transaction involves entering the transaction into the system via an operator at a terminal; the second step of each transaction involves placing the transaction 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 it will go through 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 transaction, 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 transaction holding resource type K, which we stress is the sum total execution time of all visits to that stage by a transaction.

The mathematical model consists of

(*i*) $N + 2$ stages of stations: station 0 is associated with operators at terminals, station 1 is the staging station, and stations $2, \cdots, N + 1$ (N total) are associated with a single, serially reusable resource

(*ii*) Stage $K = 0, 2, \cdots, N + 1$ has P_K identical parallel servers or processors

(*iii*) A maximum of M jobs can be held at all stages $K = 2, \cdots, N + 1$

(*iv*) Each job moves from station to station, and requires T_K total mean amount of service time at stage $K = 0, 2, \cdots, N + 2$.

Figure 21 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 $\boldsymbol{J} = (J_0, \cdots, J_{N+1})$. J_K, $K = 0, 2, \cdots, N + 1$ denotes the number of jobs either waiting or in execution at stage K.

Feasible elements in the state space obey the following constraints:

(*i*) The total number of tasks in the system is fixed at P_0

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

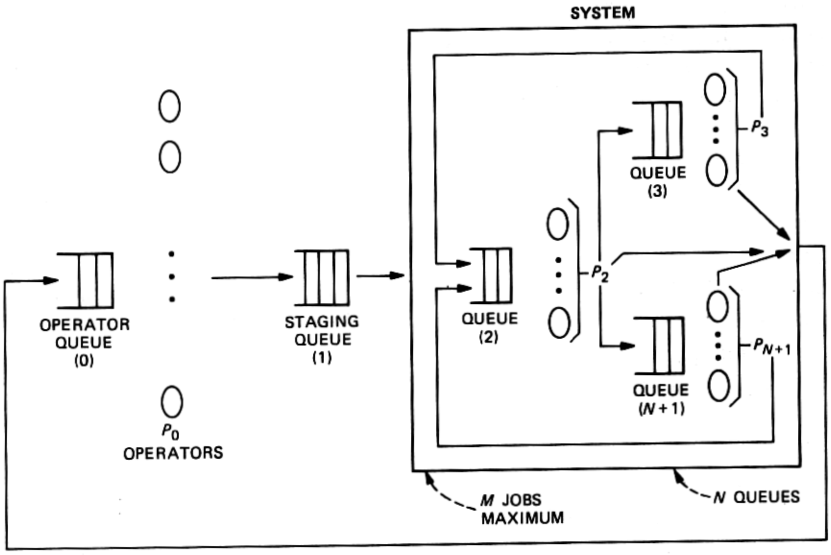


Fig. 21—Block diagram of the queueing network model.

(*ii*) 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 $\boldsymbol{J}$ in Ω are nonnegative integer-valued tuples such that

$$\boldsymbol{J} \in \Omega = \{V \mid V = (V_0, \cdots, V_{N+1}); \qquad V_N \geq 0 \; K = 0, \cdots, N+1;$$
$$\sum_{K=0}^{N+1} V_K = P_0; \qquad \sum_{K=2}^{N=1} V_K = \min(M, P_0 - V_0)\}.$$

The number of jobs *in execution* at stage $K = 0, 2, \cdots, N+1$ 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

$$\equiv E[\min(J_K, P_K)] = \lambda T_K \quad K = 0, 2, \cdots, N+1,$$

where $E(\cdot)$ 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, $K = 0, \cdots, N+1$.

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.

B.1 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

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

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 $I = 2, \cdots, N+1$ such that $P_I \le J_I$. Since all the terms on the left-hand side are non-negative, we can lower bound the left-hand side by ignoring all of these terms except term I:

$$\frac{\min(J_0, P_0)}{P_0} + \sum_{K=2}^{N+1} \frac{\min(J_K, P_K)}{\min(M, P_0, P_K)} \ge \frac{\min(J_I, P_I)}{\min(M, P_0, P_I)} \ge \frac{P_I}{\min(M, P_0, P_I)} \ge 1 \quad I = 2, \cdots, N+1.$$

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

$$\min(J_K, P_K) = J_K \quad K = 2, \cdots, N+1.$$

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

$$\frac{J_0}{P_0} + \sum_{K=2}^{N+1} \frac{J_K}{\min(M, P_0, P_K)} \ge \frac{J_0}{P_0} + \frac{P_0 - J_0}{P_0} = 1.$$

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

$$\frac{\min(J_0, P_0)}{P_0} + \sum_{K=2}^{N+1} \frac{J_K}{\min(M, P_0, P_K)} \ge \sum_{K=2}^{N+1} \frac{J_K}{\min(M, P_0, P_K)} = \frac{M}{M} = 1.$$

Hence, we see that

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

and we obtain the desired lower bound:

$$\lambda_{\text{lower}} = \frac{P_0}{T_0 + \sum_{K=2}^{N+1} \frac{P_0}{\min(M, P_0, P_K)} T_K}.$$

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

$$\tilde{T}_K = \frac{P_0}{\min(M, P_0, P_K)} T_K \geq T_k \quad K = 2, \cdots, N+1$$

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

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

B.2 Upper bound on mean throughput rate

From the definition of λ we see

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

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

$$\lambda T_K \leq E(J_K) \quad K = 0, 2, \cdots, N+1$$
$$\rightarrow \lambda\left(T_0 + \sum_{K=2}^{N+1} T_K\right) \leq E\left(J_0 + \sum_{K=2}^{N+1} J_K\right) = P_0.$$

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

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

B.3 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:

(*i*) 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

(*ii*) 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 the stage, or the number of processors with the same speed, the mean throughput rate increases in roughly the same proportion

(*iii*) 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 $(1 - \epsilon_K)$ denotes the fraction of jobs at stage K that are executed in "zero" time and ϵ_K is the fraction of jobs at stage K that are executed in time $1/\mu_K$ such that $T_K = \epsilon_K/\mu_K$. 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.

APPENDIX C

Product Form Distribution Results

The mathematical model considered in this section is a special case of that considered in the previous section:

(*i*) One type of job that migrates amongst S stations or stages

(*ii*) A *single* processor available to execute a job at stage $K = 1, \cdots, S$

(*iii*) N tasks or jobs circulate among the nodes

(*iv*) T_K denotes the total mean amount of service required by a job summed over all its visits to stage $K = 1, \cdots, S$.

The system state is denoted by Ω:

$$\Omega = \left\{ (J_1, \cdots, J_S) \mid \sum_{K=1}^{S} J_K = N \right\}.$$

At any given instant of time, the system is in state $\boldsymbol{J} = (J_1, \cdots, J_S)$, where J_K, $K = 1, \cdots, S$ denotes the number of jobs at node K (both waiting and in execution). The long-term time-averaged distribution of number of jobs at each node at an arbitrary instant of time is assumed from this point on to obey a so-called *product form* or separation of variables formula

$$\text{PROB}(J_1 = K_1, \cdots, J_S = K_S) = \frac{1}{G_N} \prod_{I=1}^{S} T_I^{K_I} \qquad (K_1, \cdots, K_S) \in \Omega$$

$$G_N = G_N(T_1, \cdots, T_S) = \sum_{J \in \Omega} \prod_{I=1}^{S} T_I^{J_I}.$$

The interested reader is referred to the literature[3] for probabilistic assumptions that lead to this type of probability measure on Ω, the admissible state space. G_N is the *system partition function* chosen to normalize the product form to a probability measure.[3] Granted these assumptions, we observe that the mean throughput rate of jobs making a complete cycle of the system is given by

$$\lambda = \frac{\text{PROB}(J_K > 0)}{T_K} = \frac{G_{N-1}(T_1, \cdots, T_S)}{G_N(T_1, \cdots, T_S)}.$$

Our goal is to obtain *tighter* upper and lower bounds on mean throughput rate and hence mean delay than we obtained in the previous section, using this additional information. We begin by observing that

$$\lambda \sum_{K=1}^{S} T_K = \frac{G_{N-1}(T_1, \cdots, T_s) \sum_{K=1}^{S} T_K}{G_N(T_1, \cdots, T_S)}$$

is a symmetric function of the S variables T_K, i.e., we do not change the value of the function when we interchange any two variables. This property allows us to show that this function has its maximum when all the variables are equal to one another. This follows from calculating first the gradient of the function at that point and showing that it is zero, and second showing that the Hessian, the determinant of all partial second derivatives, is negative definite at that point. An alternate way of seeing this holds is to realize that the gradient is zero (from the symmetry of the function) at the point where all coordinates are unity, so this point must either be a minimum or a maximum; we then evaluate the function at a neighboring point, say the point where all coordinates except one equal zero, and see that this is *less* than at the point where all coordinates are one, so this must be a maximum. Hence, we see

$$\lambda \sum_{K=1}^{S} T_K \leq \frac{S G_{N-1}(T_1 = 1, \cdots, T_S = 1)}{G_N(T_1 = 1, \cdots, T_S = 1)}$$

$$\leq \frac{S\binom{S+N-2}{N-1}}{\binom{S+N-1}{N}} = \frac{SN}{S+N-1}.$$

We now rearrange this upper bound to see

$$\lambda \leq \frac{N}{\sum_{K=1}^{S} T_K + (N-1)T_{\text{average}}}.$$

The first term in the denominator is the mean time for a job to make a complete cycle through the network:

$$T_{\text{cycle}} = \sum_{K=1}^{S} T_K.$$

The second term in the denominator is the mean amount of time per node spent by a job in one cycle of the network:

$$T_{\text{average}} = \frac{1}{S} \sum_{K=1}^{S} T_K.$$

The same method is now used to obtain a lower bound on the mean throughput rate, by observing that

$$\frac{1}{\lambda} = \frac{G_N(T_1, \cdots, T_S)}{G_{N-1}(T_1, \cdots, T_S)}.$$

Without loss of generality, we number the nodes such that node S is the node that will do the greatest amount of processing on a job on the average during one cycle:

$$T_S = \max_{K=1,\cdots,S} T_K.$$

This can be used to rewrite the above expression:

$$\frac{1}{\lambda} = T_S + \frac{G_N(T_1, \cdots, T_{S-1})}{G_{N-1}(T_1, \cdots, T_S)}.$$

Since the second term is positive, this immediately gives us an *upper* bound on the mean throughput rate:

$$\lambda \leq \frac{1}{T_S}.$$

In words, node S is the *bottleneck* node, in this sense.

We now rewrite our expression for the reciprocal of the mean throughput rate:

$$\frac{1}{\lambda} = T_S + \sum_{K=1}^{S-1} T_K F(T_1, \cdots, T_S)$$

$$F(T_1, \cdots, T_S) = \frac{G_N(T_1, \cdots, T_{S-1})}{G_{N-1}(T_1, \cdots, T_S) \sum_{K=1}^{S-1} T_K}.$$

We now manipulate this expression as we did above:

$$F(T^1, \cdots, T^S) \leq \frac{G^N(T^1 = 1, \cdots, T^{S-1} = 1)}{(S-1)G^{N-1}(T^1 = 1, \cdots, T^S = 1)}$$

$$= \frac{\begin{pmatrix} S+N-2 \\ N \end{pmatrix}}{(S-1)\begin{pmatrix} S+N-2 \\ N-1 \end{pmatrix}} = \frac{\dfrac{(S+N-2)!}{N!(S-2)!}}{\dfrac{(S-1)(S+N-2)!}{(S-1)!(N-1)!}} = \frac{1}{N}.$$

Combining all this, we see

$$\frac{1}{\lambda} \leq T_S + \frac{1}{N} \sum_{K=1}^{S-1} T_K.$$

Rearranging, we see

$$\frac{N}{\sum_{K=1}^{S} T_K + (N-1)T_{\max}} \leq \lambda.$$

The first term in the denominator is the mean time for one job to make one complete cycle of the network:

$$T_{\text{cycle}} = \sum_{K=1}^{S} T_K.$$

The second term is the maximum mean time a job spends at any one node in the network:

$$T_{\max} = \max_{K=1,\cdots,S} T_K.$$

In summary, we see

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

$$\frac{N}{T_{\text{cycle}} + (N-1)T_{\max}} \leq \lambda \qquad \leq \min\left[\frac{1}{T_{\max}}, \frac{N}{T_{\text{cycle}} + (N-1)T_{\text{average}}}\right].$$

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