The Operational Analysis of Queueing Network Models*

PETER J. DENNING

Computer Sciences Department, Purdue University, West Lafayette, Indiana 47907

JEFFREY P. BUZEN

BGS Systems, Inc., Box 128, Lincoln, Massachusetts 01773

Queueing network models have proved to be cost effective tools for analyzing modern computer systems. This tutorial paper presents the basic results using the operational approach, a framework which allows the analyst to test whether each assumption is met in a given system. The early sections describe the nature of queueing network models and their applications for calculating and predicting performance quantities The basic performance quantities—such as utilizations, mean queue lengths, and mean response times—are defined, and operational relationships among them are derived Following this, the concept of job flow balance is introduced and used to study asymptotic throughputs and response times. The concepts of state transition balance, one-step behavior, and homogeneity are then used to relate the proportions of time that each system state is occupied to the parameters of job demand and to device characteristics Efficient methods for computing basic performance quantities are also described. Finally the concept of decomposition is used to simplify analyses by replacing subsystems with equivalent devices. All concepts are illustrated liberally with examples

Keywords and Phrases: balanced system, bottlenecks, decomposability, operational analysis, performance evaluation, performance modeling, queueing models, queueing networks, response times, saturation.

CR Categories: 8.1, 4.3

INTRODUCTION

Queueing networks are used widely to analyze the performance of multiprogrammed computer systems. The theory dates back to the 1950s. In 1957, Jackson published an analysis of a multiple device system wherein each device contained one or more parallel servers and jobs could enter or exit the system anywhere [JACK57]. In 1963 Jackson extended his analysis to open and closed systems with local load-dependent service rates at all devices [JACK63]. In 1967, Gordon and Newell simplified the notational structure of these results for the special case of closed systems [GORD67]. Baskett, et al. extended the results to include different queueing disciplines, multiple classes of jobs, and nonexponential service distributions [BASK75].

The first successful application of a network model to a computer system came in 1965 when Scherr used the classical machine repairman model to analyze the MIT time sharing system, CTSS [SCHE67]. However, the Jackson-Gordon-Newell theory

* This work was supported in part by NSF Grant GJ-41289 at Purdue University

CONTENTS

lay dormant until 1971 when Buzen introduced the central server model and fast computational algorithms for these models [BUZE71a, BUZE71b, BUZE73]. Working independently, Moore showed that queueing network models could predict the response times on the Michigan Terminal System (MTS) to within 10% [MOOR71]. Extensive validations since 1971 have verified that these models reproduce observed performance quantities with remarkable accuracy [BUZE75, GIAM76, HUGH73, LIPS77, ROSE78]. Good surveys are [GELE76a, KLEI75, KLEI76, and MUNT75].

Many analysts have experienced puzzlement at the accuracy of queueing network results. The traditional approach to deriving them depends on a series of assumptions used in the theory of stochastic processes:

- The system is modeled by a *stationary stochastic process*;
- Jobs are *stochastically independent*;
- Job steps from device to device follow a *Markov chain*;
- The system is in *stochastic equilibrium*;
- The service time requirements at each device conform to an *exponential distribution*; and
- The system is *ergodic*—i.e., long-term time averages converge to the values computed for stochastic equilibrium.

The theory of queueing networks based on these assumptions is usually called "Markovian queueing network theory" [KLEI75]. The italicized words in this list of assumptions illustrate concepts that the analyst must understand to be able to deploy the models. Some of these concepts are difficult. Some, such as "equilibrium" or "stationarity," cannot be proved to hold by observing the system in a finite time period. In fact, most can be disproved empirically —for example, parameters change over time, jobs are dependent, device to device transitions do not follow Markov chains, systems are observable only for short periods, service distributions are seldom exponential. It is no wonder that many people are surprised that these models apply so well to systems which violate so many assumptions of the analysis!

In applying or validating the results of Markovian queueing network theory, analysts substitute operational (i.e., directly measured) values for stochastic parameters in the equations. The repeated successes of validations led us to investigate whether the traditional equations of Markovian queueing network theory might also be relations among operational variables, and, if so, whether they can be derived using different assumptions that can be directly verified and that are likely to hold in actual systems. This has proved to be true [BUZE76a,b,c; and DENN77].

This tutorial paper outlines the operational approach to queueing network modeling. All the basic equations and results are derived from one or more of three operational principles:

- All quantities should be defined so as

to be *precisely measurable*, and all assumptions stated so as to be *directly testable*. The validity of results should depend only on assumptions which can be tested by observing a real system for a finite period of time.

- The system must be *flow balanced*—i.e., the number of arrivals at a given device must be (almost) the same as the number of departures from that device during the observation period.
- The devices must be *homogeneous*—i.e., the routing of jobs must be independent of local queue lengths, and the mean time between service completions at a given device must not depend on the queue lengths of other devices.

These operational principles, which will be discussed at length in later sections, lead to the same mathematical equations as the traditional Markovian assumptions. However, the operational assumptions can be tested, and there are good reasons to believe that they often hold. This is why operational queueing network analysis explains the success of validation experiments. It is now possible to use the queueing network technology with much more confidence and understanding.

1. THE BASIS FOR OPERATIONAL ANALYSIS

Throughout this paper we will be concerned with deriving equations that characterize the performance of actual computer systems during given time periods. To do this, we need a mathematical framework in which we can define formal variables, formulate hypotheses, and prove theorems.

The theory of stochastic processes has traditionally been used as such a framework. Most analyses of performance begin with the

> *Stochastic Hypothesis:* The behavior of the real system during a given period of time is characterized by the probability distributions of a stochastic process.

Supplementary hypotheses are usually also made. These hypotheses, which concern the nature of the stochastic process, typically introduce concepts such as steady state, ergodicity, independence, and the distributions of specific random variables. All these hypotheses constitute a *stochastic model.*

Observable aspects of the real system—e.g., states, parameters, and probability distributions—can be identified with quantities in the stochastic model, and equations relating these quantities can be derived. Although formally applicable only to the stochastic process, these equations can also be applied to the observable behavior of the system itself under suitable limiting conditions [BUZE78a].

Stochastic models bestow bountiful benefits. Independent and dependent variables can be defined precisely, hypotheses can be stated succinctly, and a considerable body of theory can be called on during analysis. However, stochastic modeling has certain disadvantages, the most important being the impossibility of validating the Stochastic Hypothesis and the supplementary hypotheses that depend on it.

The Stochastic Hypothesis is an assertion about the causes underlying the behavior of a real system. Because one cannot prove asserted causes by studying observed effects, the truth or falsehood of the Stochastic Hypothesis and its dependent supplementary hypotheses—for a given system and time period—can never be established beyond doubt through any measurement.[1] This is true even if measurement error is assumed to be zero and every conceivable measurement is assumed to be taken.

Thus, an analyst can never be certain that an equation derived from a stochastic model can be correctly applied to the observable behavior of a real system.

Operational Variables, Laws, and Theorems

Hypotheses whose veracity can be established beyond doubt by measurement will be called *operationally testable.* Operational analysis provides a rigorous mathematical discipline for studying computer system performance based solely on operationally testable hypotheses.

[1] For example, one can never establish through measurement that a set of observed service times is or is not a sample from a sequence of independent exponentially distributed random variables.

In operational analysis there are two basic components to every problem: a system, which can be real or hypothetical, and a time period, which may be past, present, or future. The objective of an analysis is equations relating quantities measurable in the system during the given time period.

The finite time period in which a system is observed is called the *observation period.* An operational variable is a formal symbol that stands for the value of some quantity which is measurable during the observation period. It has a single, specific value for each observation period.

Operational variables are either *basic quantities,* which are directly measured during the observation period, or *derived quantities,* which are computed from the basic quantities. Figure 1 shows a single-server queueing system with four basic quantities:

- T—the length of the observation period;
- A—the number of arrivals occurring during the observation period;
- B—the total amount of time during which the system is busy during the observation period ($B \leq T$); and
- C—the number of completions occurring during the observation period.

Four important derived quantities are

- $\lambda = A/T$, the *arrival rate* (jobs/second);
- $X = C/T$, the *output rate* (jobs/second);
- $U = B/T$, the *utilization* (fraction of time system is busy); and
- $S = B/C$, the *mean service time* per completed job.

The basic quantities (A, B, C) are typical of "raw data" collected during an observation, and the derived quantities (λ, X, U, S) are typical of "performance measures." All these quantities are variables which may change from one observation period to another.

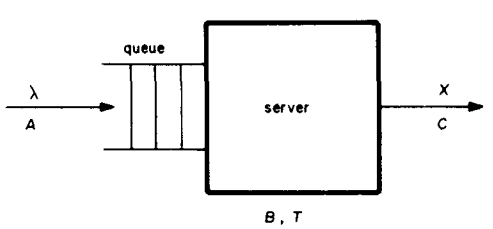

FIGURE 1 Single server queueing system.

It is easy to see that the derived quantities satisfy the equation

$$U = XS.$$

Thus, if the system is completing 3 jobs/second, and if each job requires 0.1 second of service, then the utilization of the system is 0.3 or 30%. An equation such as this, which expresses an identity among operational quantities, is called an *operational law* or *operational identity.* This is because the relation must hold in every observation period, regardless of the values observed. The identity $U = XS$ is called the *utilization law.* We will encounter various other operational laws later.

Now, suppose that we assume that the number of arrivals is equal to the number of completions during the observation period. That is, we assume

$$A = C.$$

This assumption is called *job flow balance* because it implies $\lambda = X$. Job flow balance holds only in some observation periods. However, it is often a very good approximation, especially if the observation period is long, because the ratio of unfinished to completed jobs, $(A - C)/C$, is typically small. Job flow balance is an example of an operationally testable assumption: it need not hold in every observation period, but an analyst can always test whether or not it does—or how much error is made by assuming it does.

Under the assumption of job flow balance, it is easy to see that

$$U = \lambda S.$$

This is an example of an *operational theorem*: a proposition derived from operational quantities with the help of operationally testable assumptions.

In a stochastic analysis of Figure 1, λ would be interpreted as the reciprocal of the mean time between arrivals, S as the mean amount of service requested by jobs, and U as the steady-state probability that the system has at least one job in it. The statement $U = \lambda S$ is a limit theorem for stochastic steady state [KLEI75]. In general, a steady-state stochastic theorem is a statement about a collection (ensemble) of

possible infinite behavior sequences, but it is not guaranteed to apply to a particular finite behavior sequence. An operational theorem is a statement about the collection of behavior sequences, finite or infinite, that satisfy the given operational assumptions: it is guaranteed to apply to every behavior sequence in the collection. (For detailed comparisons between stochastic and operational modeling, see [BOUH78, BUZE78a].)

Application Areas

There are three major applications for operational results such as the utilization law:

- *Performance Calculation.* Operational results can be used to compute quantities which were not measured, but could have been. For example, a measurement of U is not needed in a flow-balanced system if λ and S have been measured.
- *Consistency Checking.* A failure of the data to verify a theorem or identity reveals an error in the data, a fault in the measurement procedure, or a violation of a critical hypothesis. For example, $U \neq \lambda S$ would imply an error if observed in a flow-balanced system.
- *Performance Prediction.* Operational results can be used to estimate performance quantities in a future time period (or indeed a past one) for which no directly measured data are available. For example, the analyst can estimate λ and S for the future time period, and then predict that U will have the value λS in that time period. (Although the analyst may find ways of estimating U directly, it is often easier to calculate it indirectly from estimates of λ and S.)

The first two applications are straightforward, but the third is actually a two-step process. The first step is *estimating* the values of λ and S for the future time period; the second step is *calculating* U. Our primary concern in this paper is deriving the equations which can be used for performance calculation, consistency checking, and the second step in performance prediction.

Parameter estimation, the first step in performance prediction, is a problem of induction—inferring the characteristics of an unseen part of the universe on the basis of observations of another finite part. Gardner has an interesting discussion of why no one has found a consistent system of inductive mathematics [GARD76]. Various techniques for dealing with the parameter estimation problem will be discussed throughout this paper.

Prior Work in Operational Analysis

Many textbooks illustrate the ideas of probability with operational concepts such as "relative frequencies" and "proportions of time." In addition, the derivations of many well-known results in the classical theory of stochastic processes are based, in part, on operational arguments. However, the explicit recognition that operational analysis is a separate branch of applied mathematics—quite apart from the theory of stochastic processes—is a more recent development.

The concept of operational analysis as a separate mathematical discipline was first proposed by Buzen [BUZE76b], who characterized the real-world problems that could be treated with operational techniques, and derived operational laws and theorems giving exact answers for a large class of practical performance problems. At about the same time, operational arguments leading to upper and lower bounds on the saturation behavior of computer systems were presented by Denning and Kahn [DENN75a]. These arguments were the operational counterpart of similar results developed by Muntz and Wong [MUNT74]. The only operational assumption used at this point was job flow balance.

These early operational results dealt exclusively with mean values of quantities such as throughput, response time, and queue length. The theory was soon extended so that complete operational distributions—as well as mean values—could be derived for operational analogs of the "birth-death process" and the "M/M/1 queueing process" [BUZE76a, BUZE78a]. These extensions introduced two new analysis techniques: the application of

"flow balance" in the logical state space of the system (as contrasted with the physical system itself) and the homogeneity assumptions, which are the operational counterparts of Markovian assumptions in stochastic theory. These techniques form the basis for the operational treatment of many problems which are conventionally analyzed with ergodic Markovian models.

The results in [BUZE76a and BUZE78a] applied only to single-resource queueing systems. The same analysis techniques were applied to multiple-resource queueing networks by Denning and Buzen [DENN77a], who showed that the "product form solution," encountered in Markovian queueing networks, holds in general queueing networks with flow balance and homogeneity; this result is more general than can be derived in the Markovian framework. This work also introduced a new operational concept, "online = offline behavior," which characterizes the way analysts use decomposition to estimate parameters of devices and subsystems. The operational treatment of queueing network models is discussed in detail in the rest of this paper. Additional points about the theory and applications of operational analysis have been given in [BOUH78, BUZE77, BUZE78a].

2. VALIDATION AND PREDICTION

We have noted three uses of models in studying computer performance: calculation, consistency-checking, and prediction of performance measures. *Validation* refers to extensive testing of a model to determine its accuracy in calculating performance measures. *Prediction* refers to using a validated model to calculate performance measures for a time period (usually in the future) when the values of parameters required by the model are uncertain.

Figure 2 illustrates the steps followed in a typical validation. First, the analyst runs an actual workload on an actual system. For the observation period, he measures performance quantities, such as throughput and response time, and also the parameters of the devices and the workload. Then the analyst applies a model to these parameters, and compares the results against the measured performance quantities. If, over many different observation periods, the computed values compare well with actual (measured) values, the analyst will come to believe that the model is good. Thereafter, he will employ it confidently for predicting future behavior and for evaluating proposed changes in the system.

The scheme of Figure 2 is used to validate many types of models, including highly detailed deterministic models, simulation models, and queueing network models. In general, the more parameters used by the model, the greater is its accuracy in such validations.

Performance prediction typically follows the scheme of Figure 3. The analyst begins with a set of workload and device parameters for a particular observation period, known as the *baseline period.* He then carries out a modification analysis to estimate the values these parameters are expected to have in the *projection period,* which is another time period for which he

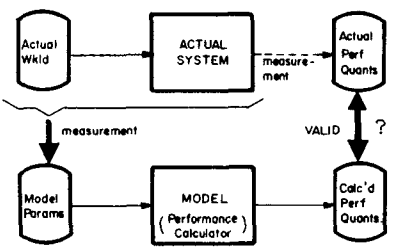

FIGURE 2. Typical validation scheme.

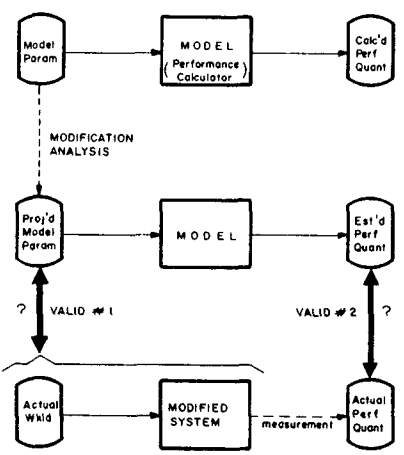

FIGURE 3. Typical performance prediction scheme.

desires to know performance quantities. (In the projection period, the same system may be processing a changed workload, or a changed system may be processing the same workload, or both.) The analyst applies the validated model to calculate performance quantities for the projection period. If the modification is ever implemented, the predictions can be validated by comparing the actual workload and system parameters against the project values (#1) and the actual performance quantities against the projected quantities (#2).

A variety of *invariance assumptions* are employed in the modification analysis. These assumptions are typically that device and workload parameters do not change unless they are explicitly modified—the analyst may assume, for example, that the mean disk service time will be invariant if the same disk is present in both the baseline and projection periods, or that the mean number of requests for each disk will be invariant if the same workload is present in both periods. Though usually satisfactory, such assumptions can lead to trouble if a given change has side effects—for example, increasing the number of time-sharing terminals may unexpectedly reduce the batch multiprogramming level even though the batch workload is the same.

The wise analyst will make all his invariance assumptions explicit. Otherwise, he will have difficulty in explaining a failure in Validation #1, which will cause a failure in Validation #2—even though previous tests of the model were satisfactory (Figure 2).

In some prediction problems there is no explicit baseline period. In these cases, the analyst must estimate parameters for the projection period by other means. For example, he can estimate the mean service time for a disk from published specifications of seek time, rotation time, and data transfer rate; and he can estimate the mean number of disk requests per job from an analysis of the source code of representative programs. Usually, however, the modification analysis is more accurate when it begins with a measured baseline period.

A model's quality depends on the number of parameters it requires. The more information the model requires about the workload and the system, the greater the accuracy attainable in its calculations. However, when there are many parameters, there may be a lot of uncertainty about whether all are correctly estimated for a projection period; the confidence in the predictions may thereby be reduced. Queueing network models isolate the few critical parameters. They permit accurate calculation and credible prediction.

Additional issues of performance calculation and parameter estimation will be discussed as they arise throughout the paper. (See also [BUZE77, BUZE78a].)

3. OPERATIONAL MEASURES OF NETWORKS

Figure 1 illustrated a "single resource" queueing model consisting of a queue and a service facility. This model can be used to represent a single input/output (I/O) device or central processing unit (CPU) within a computer system. A model of the entire computer system can be developed by connecting single-resource models in the same configuration as the devices of an actual computer system. A set of interconnected single-resource queueing models comprises a multiple-resource queueing network.

Types of Networks

Figure 4 shows two of K devices in a multiple-resource network. A job enters the system at IN. It circulates around in the network, waiting in queues and having service requests processed at various devices. When done, it exits at OUT. The network is *operationally connected* in that each device is visited at least once by some job during the observation period.

The model assumes that no job overlaps its use of different devices. In practice, few applications programs ever achieve more than a few per cent overlap between CPU and I/O devices: the error introduced by this assumption is usually not significant.[2] The model also assumes that a device is

[2] Measurements taken at the Purdue University Computer Center reveal that the average overlap of CPU and I/O within a job is between 4 and 6 per cent.

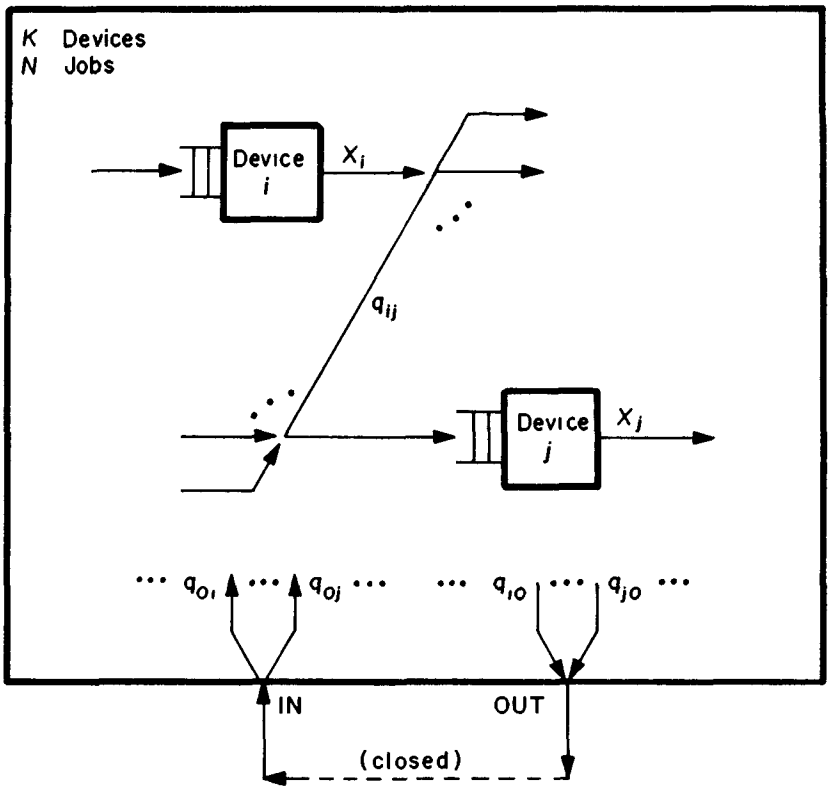

FIGURE 4. Two devices in a queueing network.

busy if a request is pending there—no part of the system can block progress in another part. This assumption is not met by all real systems; for example, the CPU might be unable to continue if an I/O buffer is full.

A job is "in queue" at device i if it is waiting for or receiving service there. We let n_i denote the number of jobs in queue at device i, and $N = n_1 + \cdots + n_K$ denote the total number of jobs in the system. The *system output rate*, X_0, is the number of jobs per second leaving the system. If the system is *open*, X_0 is known and N varies as jobs enter or leave the system. If the system is *closed*, the number of jobs N is fixed; this is modeled by connecting the output back to the input, as suggested by the dashed arrow in Figure 4.

An analysis of an open system assumes that X_0 is known and seeks to characterize the distribution of N. An analysis of a closed system begins with N given and seeks to determine the resulting X_0 along the OUT/IN path. Other quantities such as queue lengths and response times at the devices may be sought in both cases.

Example: Figure 5 shows a common type of network, the "central server." Device 1 is the CPU, devices 2, · , K are I/O stations. A job begins with a CPU service interval (burst) and continues with zero or more I/O service intervals which alternate with further CPU bursts. The quantities q_{1i} are called the "routing frequencies" and the S_i the "mean service times." Definitions for these quantities will be given shortly.

In the closed central server network of Figure 5, a new job enters the system as soon as an active job terminates. This be-

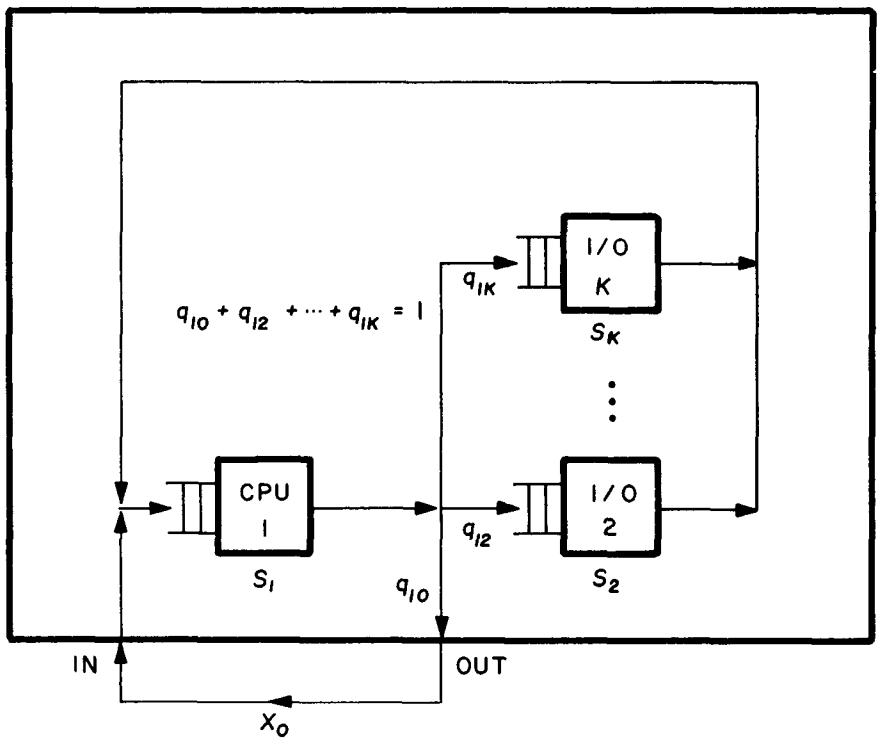

FIGURE 5. Central server network.

havior typically occurs in a batch processing system operating under a backlog. The throughput of the system under these conditions is denoted by X_0.

Time sharing systems which are driven by interactive terminals can also be represented as closed networks. Figure 6 depicts the structure. The model is separated into two (open) subnetworks: the central subsystem, which consists of I/O devices and the CPUs, and the terminal subsystem. Each terminal is manned by a user who alternates between *thinking* and *waiting*. In the thinking state, the user is contemplating what job next to submit, and the central subsystem is performing no work for him. On submitting a next job, the user enters the waiting state, where he remains until the central subsystem completes the job for him. The mean time a user spends in a thinking interval is called the *think time*; we denote it by Z. The mean time a user spends in a waiting interval is called the *response time* (of the central subsystem); we denote it by R. Since users think independently, the think time Z is independent of M. Because jobs delay each other while contending for resources in the central subsystem, R is a function of M.

It is also possible to define mixed systems which are open for some workloads and closed for others. Figure 7 illustrates a typical case. The *interactive workload* comprises the jobs associated with the M interactive terminals. The *batch workload* comprises jobs submitted by other means, for example, remote job entry stations. The number of interactive jobs in the network (including the terminal subnetwork) is fixed at M, but the number of batch jobs may be variable. The batch throughput (X_0) is given, but the interactive throughput (X_0') depends on X_0 and on the other parameters of the network.

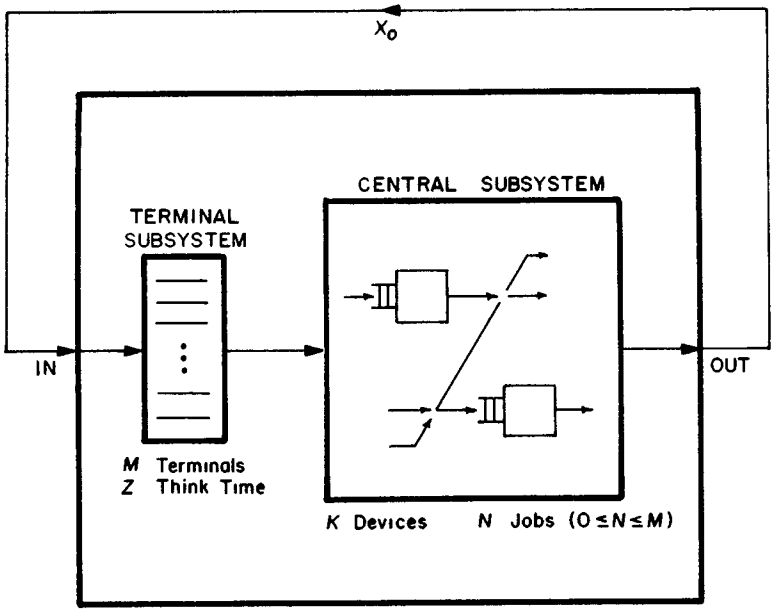

FIGURE 6. Terminal-driven system.

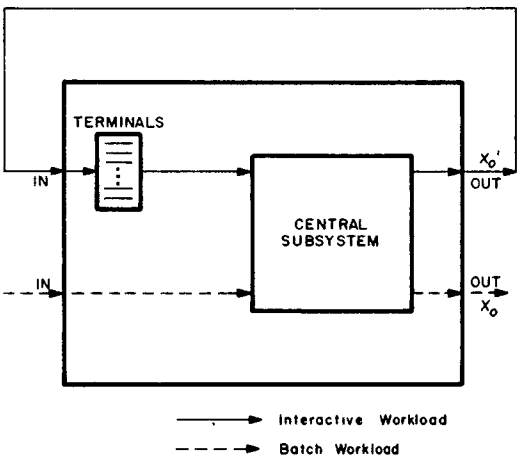

FIGURE 7. Mixed system.

Basic Operational Quantities

Suppose that the system is measured during an observation period of T seconds and that these data are collected for each device $i = 1, \ldots, K$:

A_i—number of arrivals;

B_i—total busy time (time during which $n_i > 0$);

C_{ij}—number of times a job requests service at device j immediately after completing a service request at device i.

These are similar to data specified in Figure 1, but here we are not requiring device i to be a single server. If we treat the "outside world" as device "O", we can define also

A_{0j}—number of jobs whose first service request is for device j;

C_{i0}—number of jobs whose last service request is for device i.

We will assume that $C_{00} = 0$, because otherwise there would be jobs that used no resources before departing. However, it is possible that $C_{ii} > 0$ for any device i since a job could request another burst of service from a device which had just completed a

request for that job. The number of completions at device i is

$$C_i = \sum_{j=0}^{K} C_{ij}, \quad i = 1, \ldots, K.$$

The number of arrivals to, and departures from, the system are, respectively,

$$A_0 = \sum_{j=1}^{K} A_{0j}, \quad C_0 = \sum_{i=1}^{K} C_{i0}.$$

From Figure 4 it is clear that $A_0 = C_0$ in a closed system.

In terms of these basic data, four derived operational quantities are defined:

U_i = utilization of device i
$= B_i/T$.

S_i = mean service time between completions of requests at device i
$= B_i/C_i$

X_i = output rate of requests from device i
$= C_i/T$

q_{ij} = routing frequency, the fraction of jobs proceeding next to device j on completing a service request at device i

$$= \begin{cases} C_{ij}/C_i, & \text{if } i = 1, \ldots, K \\ A_{0j}/A_0, & \text{if } i = 0. \end{cases}$$

Note that, for any i, $q_{i0} + q_{i1} + \ldots + q_{iK} = 1$. Note that q_{i0} is an output routing frequency (fraction of completions from device i corresponding to the final service request of some job) and q_{0j} is an input routing frequency (fraction of arrivals to the system which proceed first to device j). Note also that the system output rate is defined as $X_0 = C_0/T$. It is easy to deduce the operational law

$$X_0 = \sum_{i=1}^{K} X_i q_{i0}.$$

Note that $X_0, X_1, \ldots, X_K$ cannot be interpreted as "throughputs" because no assumption of job flow balance has been made. It is clear that the utilization law

$$U_i = X_i S_i$$

holds at every device.

We let n_i denote the queue length at device i; it includes jobs waiting for *and* receiving service. Sometimes we write $n_i(t)$ to make explicit the time dependence. (An example $n_i(t)$ appears in Figure 8.) To calculate mean queue length and response time at a device, analysts usually introduce the basic measure W_i, which is the area under the graph of $n_i(t)$ during the observation period. Since $\bar{n}_i$, the mean queue length at device i, is the average height of this graph,

$$\bar{n}_i = W_i/T.$$

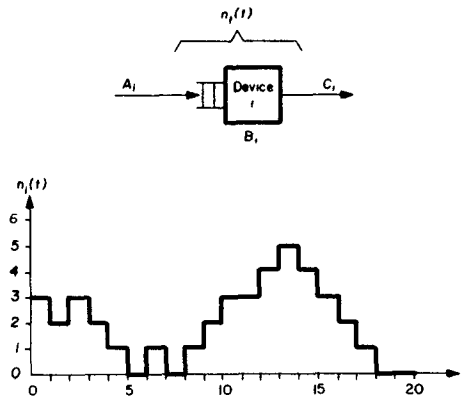

FIGURE 8. Example of a device's operation.

The mean response time at device i, denoted by R_i, is also related to W_i in a simple way. Note that W_i can be interpreted as the total number of "job-seconds" accumulated at device i during the observation period (if j jobs are at a device for s seconds, js job-seconds accumulate). R_i is defined as the average amount of time accumulated at device i per completed request. Thus

$$R_i = W_i/C_i.$$

An immediate consequence of these definitions is the operational law

$$\bar{n}_i = X_i R_i,$$

which is called *Little's Law*.

Example: Figure 8 shows device i and a possible observation of its queue length for a period of 20 seconds. The basic measures are

$A_i = 7$ jobs, $B_i = 16$ seconds, $C_i = 10$ jobs.

Note that $n_i(0) = 3$ and that

$$n_i(20) = n_i(0) + A_i - C_i = 0.$$

The basic operational performance measures are

$U_i = 16/20 = 0.80$ $S_i = 16/10 = 1.6$ seconds $X_i = 10/20 = 0.5$ jobs/second

The total area under $n_i(t)$ in the observation period is

$$W_i = 40 \text{ job-seconds.}$$

Thus the mean queue length is

$$\bar{n}_i = W_i/T = 2 \text{ jobs,}$$

and the mean response time per service completion is:

$$R_i = W_i/C_i = 4 \text{ seconds.}$$

4. JOB FLOW ANALYSIS

Given the mean service times (S_i) and the routing frequencies (q_{ij}), how much can we determine about overall device completion rates (X_i) or response times (R_i)? These questions are usually approached through the operational hypothesis known as the

Principle of Job Flow Balance: For each device i, X_i is the same as the total input rate to device i.

This principle will give a good approximation for observation periods long enough that the difference between arrivals and completions, $A_i - C_i$, is small compared to C_i. It will be exact if the initial queue length $n_i(0)$ is the same as the final $n_i(T)$. Choosing an observation period so that the initial and final states of every queue are the same is not as strange as it might seem. This notion underlies the highly successful "regeneration point" method of conducting simulations [IGLE78].

When job flow is balanced, we refer to the X_i as *device throughputs*. Expressing the balance principle as an equation,

$$C_j = A_j = \sum_{i=0}^{K} C_{ij}, \quad i = 0, \ldots, K$$

(Note that job flow balance allows us to substitute C_{0j} for A_{0j}.) With the definition $q_{ij} = C_{ij}/C_i$, we may write

$$C_j = \sum_{i=0}^{K} C_i q_{ij}.$$

Employing the definition $X_i = C_i/T$, we obtain

JOB FLOW BALANCE EQUATIONS

$$X_j = \sum_{i=0}^{K} X_i q_{ij}, \quad j = 0, \ldots, K$$

If the network is open, the value of X_0 is externally specified and these equations will have a unique solution for the unknowns X_i. However, if the network is closed, X_0 is initially unknown, and the equations have no unique solution; this can be verified by showing that the sum of the X_j-equations for $j = 1, \ldots, K$ reduces to the X_0-equation. In a closed network, there are K independent equations but $K + 1$ unknowns. Nonetheless, the job flow balance equations contain information of considerable value.

Visit Ratios

The "visit ratio," which expresses the mean number of requests per job for a device, can always be calculated uniquely from the job flow balance equations. With the mean service times, they can be used to determine the throughputs and response times of systems under very light or very heavy loads. Define

$$V_i = X_i/X_0;$$

V_i is the job flow through device i relative to the system's output flow. Our definitions imply that $V_i = C_i/C_0$, which is the mean number of completions at device i for each completion from the system. Since V_i can be interpreted as the mean number of visits per job to device i, we call it the *visit ratio.*

The relation $X_i = V_i X_0$ is an operational law, called the *Forced Flow Law.* It states that the flow in any one part of the system determines the flows everywhere in the system.

Example: Consider the performance question: "Jobs generate an average of 5 disk requests and disk throughput is measured as 10 requests/second; what is the system throughput?" This question seems momentarily frivolous, since nothing is stated about the relation between the disk and any other part of the system. Yet the forced flow law gives the answer precisely. Let subscript i refer to the disk:

$$X_0 = X_i/V_i = \frac{10 \text{ requests/second}}{5 \text{ requests/job}} = 2 \text{ jobs/second.}$$

On replacing each X_i with $V_i X_0$ in the job flow balance equations, we obtain the

VISIT RATIO EQUATIONS

$$V_0 = 1$$

$$V_j = q_{0j} + \sum_{i=1}^{K} V_i q_{ij}, \quad j = 1, \ldots, K$$

These are $K + 1$ independent equations with $K + 1$ unknowns: a unique solution is always possible assuming the network is operationally connected. These equations show the relation between the network's "connective structure," represented by the q_{ij}, and the visit ratios. Although $V_i = X_i/X_0$ is an operational law, the V_i satisfy the visit ratio equations only if job flow is balanced in the network.

Example: The central server network (Figure 5) has these job flow equations:

$$X_0 = X_1 q_{10}$$

$$X_1 = X_0 + X_2 + \ldots + X_K$$

$$X_i = X_1 q_{1i}, \quad i = 2, \ldots, K.$$

Setting $X_i = V_i X_0$, these equations reduce to

$$1 = V_1 q_{10}$$

$$V_1 = 1 + V_2 + \ldots + V_K$$

$$V_i = V_1 q_{1i}, \quad i = 2, \ldots, K.$$

It is easy to see that

$$V_1 = 1/q_{10}$$

$$V_i = q_{1i}/q_{10}, \quad i = 2, \ldots, K.$$

In this case, only K of the possible routing frequencies q_{ij} are nonzero; these q_{1i} can be determined uniquely if the V_i are given. This is not so in a general network, where K visit ratios are insufficient to determine the $(K + 1)^2$ unknown routing frequencies.

As we shall see, all the performance quantities can be computed using only the visit ratios and the mean service times S_i as parameters. The visit ratio equations are used to prove that this is so. In practice, the analyst may be able to extract the visit ratios directly from workload data, thereby avoiding computing a solution of the visit ratio equations.

System Response Time

One method of computing the mean response time per job, R, for an open or closed system is to apply Little's law to the system as a whole,

$$R = \bar{N}/X_0,$$

where $\bar{N} = \bar{n}_1 + \ldots + \bar{n}_K$. If $\bar{N}$ or X_0 are not known, an alternate method can be used. Since $\bar{n}_i = X_i R_i$ from Little's law at device i, and $X_i = V_i X_0$ from the forced flow law, we have $\bar{n}_i/X_0 = V_i R_i$. This reduces $\bar{N}/X_0$ to the *General Response Time Law*:

$$R = \sum_{i=1}^{K} V_i R_i.$$

This law holds even if job flow is not balanced.

Little's law can be used to compute the central subsystem's response time R in the terminal driven system of Figure 6. The mean time for a user to complete a think-wait cycle is $Z + R$. When job flow is balanced, X_0 will denote the rate at which cycles are completed. By Little's law, $(Z + R)X_0$ must be the mean number of users observed to be in a think-wait cycle; but all the users are in such cycles, hence, $M = (Z + R)X_0$. Therefore,

$$R = M/X_0 - Z.$$

This is called the *Interactive Response Time Formula.*

Examples

This section's three examples illustrate performance calculation and performance prediction using the operational laws summarized in Table I. The first example illustrates a simple performance calculation; a few measured data are used to find the mean response time. The second example illustrates a performance calculation for a system with an interactive and a batch workload; it also illustrates a performance prediction, estimating the effect of tripled

TABLE I. OPERATIONAL EQUATIONS*

Utilization Law	$U_i = X_i S_i$
Little's Law	$\bar{n}_i = X_i R_i$
Forced Flow Law	$X_i = V_i X_0$
Output Flow Law	$X_0 = \sum_{i=1}^{K} X_i q_{i0}$
General Response Time Law	$R = \sum_{i=1}^{K} V_i R_i$
Interactive Response Time Formula (Assumes flow balance)	$R = M/X_0 - Z$

* Operational derivations for most of these equations were first presented in [BUZE76b].

batch throughput on interactive response time. The third example illustrates a more complex prediction problem, estimating the effect of consolidating two separate time sharing systems; it illustrates the use of invariance assumptions in the modification analysis.

•

For the first example, we suppose that these data have been measured on a time sharing system:

Each job generates 20 disk requests;
The disk utilization is 50%;
The mean service time at the disk is 25 milliseconds;
There are 25 terminals; and
Think time is 18 seconds.

We can calculate the response time after first calculating the throughput. Let subscript i refer to the disk. The forced flow and utilization laws imply

$$X_0 = X_i/V_i = U_i/V_iS_i.$$

From the data,

$$X_0 = \frac{(.5)}{(20)(.025)} = 1 \text{ job/second.}$$

From the interactive response time formula,

$$R = 20/1 - 18 = 2 \text{ seconds.}$$

•

Our second example considers a mixed system of the type shown in Figure 7. These data are collected:

There are 40 terminals;
Think time is 15 seconds;
Interactive response time is 5 seconds;
Disk mean service time is 40 milliseconds;
Each interactive job generates 10 disk requests;
Each batch job generates 5 disk requests; and
Disk utilization is 90%.

We would like to calculate the throughput of the batch system and then estimate a lower bound on interactive response time assuming that batch throughput is tripled. The interactive response time formula gives the interactive throughput:

$$\begin{aligned} X_0' &= M/(Z + R') \\ &= 40/(15 + 5) \\ &= 2 \text{ jobs/second.} \end{aligned}$$

Let subscript i refer to the disk. The disk throughput is $X_i + X_i'$, where X_i is the batch component and X_i' is the interactive component. The utilization law implies

$$\begin{aligned} X_i + X_i' &= U_i/S_i \\ &= (.9)/(.04) \\ &= 22.5 \text{ requests/second.} \end{aligned}$$

The forced flow law implies that the interactive component is

$$X_i' = V_i'X_0' = (10)(2) = 20 \text{ requests/second,}$$

so that the batch component is

$$X_i = 22.5 - 20 = 2.5 \text{ requests/second.}$$

Using the forced flow law again, we find the batch throughput:

$$X_0 = X_i/V_i = 2.5/5 = 0.5 \text{ jobs/second.}$$

Now consider the effect of tripling the batch throughput. If X_0 were changed to 1.5 jobs/second without changing V_i, then X_i would become $V_iX_0 = 7.5$ requests/second. Assuming that the increased throughput does not change the disk service time, the maximum completion rate at the disk is $1/S_i = 25$ requests/second; this implies that the completion rate of the interactive workload, X_i', cannot exceed $25 - 7.5 = 17.5$ requests/second. Therefore

$$X_0' = X_i'/V_i' \leq 17.5/10 = 1.75 \text{ jobs/second}$$

and

$$\begin{aligned} R' &= M/X_0' - Z \geq 40/1.75 - 15 \\ &= 7.9 \text{ seconds.} \end{aligned}$$

Tripling batch throughput increases interactive response time by at least 2.9 seconds.

Notice that the validity of these estimates depends on the assumptions that the parameters M, Z, V_i, and S_i are invariant under the change of batch throughput. Although these are often reasonable assumptions, the careful analyst will check them by verifying that the internal policies of the operating system do not adjust to the new load, and that interactive users are independent of batch users.

•

For the third example, we consider a computer center which has two time shar-

ing systems; each is based on a swapping disk whose mean service time per request is 42 msec. The mean think time in both systems is $Z = 15$ seconds. These data have been collected:

System A	*System B*
16 terminals	10 terminals
25 disk requests/job	16 disk requests/job
80% disk utilization	40% disk utilization

In order to reduce disk rentals, management is proposing to consolidate the two systems into one with 26 terminals and using only one of the disks. We would like to estimate the effect on the response times for the two classes of users.

We let subscript i refer to the disk, and use primed symbols to refer to System B. The formula $X_0 = U_i/V_iS_i$ gives throughputs for the two systems:

$$X_0 = \frac{(.8)}{(25)(.042)}$$
$$= 0.77 \text{ jobs/second} \quad (\text{System } A)$$
$$X_0' = \frac{(.4)}{(16)(.042)}$$
$$= 0.60 \text{ jobs/second} \quad (\text{System } B)$$

The response times are

$$R = 16/(.77) - 15$$
$$= 5.8 \text{ seconds} \quad (\text{System } A)$$
$$R' = 10/(.6) - 15$$
$$= 1.1 \text{ seconds} \quad (\text{System } B)$$

Over an observation period of T seconds there would be X_iT disk requests serviced in System A, and $X_i'T$ in System B; the fraction of all disk requests which are A-requests would be

$$X_iT/(X_iT + X_i'T) = U_i/(U_i + U_i') = 2/3.$$

In order to estimate the effect of consolidation, we need to know the disk completion rates for each workload when both workloads share the one disk. Because the characteristics of the users and the disk are the same before and after the change, it is reasonable to make this invariance assumption: In the consolidated system, 2/3 of the disk requests will come from the A-users. It is also reasonable to assume that the disk utilization will be nearly 100% in the consolidated system. This implies that the total disk throughput will be $1/S_i = 1/(.042) =$ 24 requests/second. Of this total, the throughputs of the two types of users are

$$X_i = (2/3)(24)$$
$$= 16 \text{ requests/second} \quad (A\text{-users})$$
$$X_i' = (1/3)(24)$$
$$= 8 \text{ requests/second} \quad (B\text{-users})$$

This implies that the system throughputs are

$$X_0 = X_i/V_i$$
$$= 16/25$$
$$= 0.64 \text{ jobs/second} \quad (A\text{-users})$$
$$X_0' = X_i'/V_i'$$
$$= 8/16$$
$$= 0.5 \text{ jobs/second} \quad (B\text{-users})$$

and that the response times are

$$R = 16/(.64) - 15$$
$$= 10 \text{ seconds} \quad (A\text{-users})$$
$$R' = 10/(.5) - 15$$
$$= 5 \text{ seconds} \quad (B\text{-users})$$

Note that the two types of users experience different response times. This is because the B-users, who generate less work for the disk, are delayed less at the disk than the A-users.

Once again it is worth noting explicitly that the parameters V_i, V_i', S_i, and Z are assumed to be invariant under the proposed change. The careful analyst will check the validity of these assumptions. The assumption that the ratio of System A to System B throughputs is invariant under the change should be approached with caution; it is typical of the assumptions a skilled analyst will make when given insufficient data about the problem. We will present an example shortly in which a faster CPU affects two workloads differently: the ratio of interactive to batch throughput changes.

Bottleneck Analysis

This section deals with the asymptotic behavior of throughput and response time of closed systems as N, the number of jobs in the system, increases. We will assume that the visit ratios and mean service times are invariant under changes in N.

Note that the ratio of completion rates for any two devices is equal to the ratio of their visit ratios:

$$X_i/X_j = V_i/V_j.$$

Since $U_i = X_iS_i$, a similar property holds for utilizations:

$$U_i/U_j = V_iS_i/V_jS_j.$$

Our invariance assumptions imply that these ratios are the same for all N.

Device i is *saturated* if its utilization is approximately 100%. If $U_i = 1$, the utilization law implies that

$$X_i = 1/S_i;$$

this means that the saturated device is completing work at its capacity—an average of one request each S_i seconds. In general, $U_i \leq 1$ and $X_i \leq 1/S_i$.

Let the subscript b refer to any device capable of saturating as N becomes large. Such devices are called *bottlenecks* because they limit the system's overall performance. Every network has at least one bottleneck.

Since the ratios U_i/U_j are fixed, the device i with the largest value of V_iS_i will be the first to achieve 100% utilization as N increases. Thus we see that, whenever device b is a bottleneck,

$$V_bS_b = \max\{V_1S_1, \ldots, V_KS_K\}.$$

The bottleneck(s) is (are) determined by device and workload parameters.

Now: if N becomes large we will observe $U_b = 1$ and $X_b = 1/S_b$; since $X_0/X_b = 1/V_b$, this implies

$$X_0 = 1/V_bS_b$$

is the maximum possible value of system throughput. Since V_iS_i is the total of all service requests per job for device i, the sum

$$R_0 = V_1S_1 + \cdots + V_KS_K,$$

which ignores queueing delays, denotes the smallest possible value of mean response time. In fact, R_0 is the response time when $N = 1$. This implies that $X_0 = 1/R_0$ when $N = 1$.

The properties of X_0 are summarized in Figure 9. As a function of N, X_0 rises monotonically from $1/R_0$ at $N = 1$ to the asymptote $1/V_bS_b$. It stays below the line of slope $1/R_0$ emanating from the origin: Job interference via queueing when $N = k$ usually prevents throughput from achieving k/R_0.

Were we to hypothesize that k jobs always managed to avoid delaying each other in the network, so that $X_0 = k/R_0$, the saturation asymptote requires that $k/R_0 \leq 1/V_bS_b$, or

$$k \leq N^* = \frac{R_0}{V_bS_b} = \frac{V_1S_1 + \quad + V_KS_K}{V_bS_b} \leq K.$$

In words, $k > N^*$ would imply with certainty that jobs queue somewhere in the system. Since N^* thus represents the load beyond which queueing is certain to be observed, we call N^* the *saturation point* of the system.

These results extend to the response time of the terminal driven system (Figure 6). For M terminals and think time Z, the mean response time is $R = M/X_0 - Z$. When $M = 1$, R must be R_0. Since X_0 cannot exceed $1/V_bS_b$,

$$R \geq MV_bS_b - Z \geq MV_iS_i - Z, \qquad i = 1, \cdots, K.$$

As M becomes large, R approaches the asymptote $MV_bS_b - Z$. These facts are summarized in Figure 10.

Notice that the response time asymptote intersects the horizontal axis at

$$M_b = Z/V_bS_b.$$

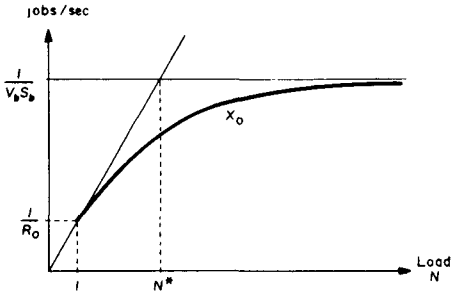

FIGURE 9. System throughput

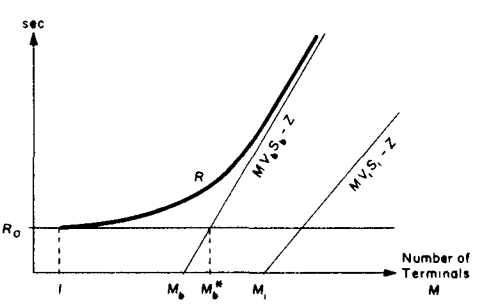

FIGURE 10. Response time

This is a product of a waiting time at the terminals (Z) and a saturation job flow through the terminals ($1/V_bS_b$); by Little's law, M_b denotes the mean number of thinking terminals when the system is saturated. The response time asymptote crosses the minimum response time R_0 at

$$M_b^* = (R_0 + Z)/V_bS_b = N^* + M_b.$$

when there are more than M_b^* terminals, queueing is certain to be observed in the central subsystem.

Notice that the response time asymptotes and intersections M_b and M_b^* depend only on M, Z, V_b, and S_b. The only assumptions needed to compute them are job flow balance and invariance of the visit ratios and mean service times under changes in load. Note also that when $Z = 0$ these results yield the response time asymptotes of a closed central system. These results may be extended to include the case where service times are not strictly invariant, but each S_i approaches some limit as the queue length at device i increases [MUNT74, DENN75a].

To summarize: the workload parameters or the visit ratio equations allows the analyst to determine the visit ratios, V_i. Device characteristics allow determination of the mean service time per visit, S_i. The largest of the products V_iS_i determines the bottleneck device, b. The sum of these products determines the smallest possible response time, R_0. The system throughput is $1/V_bS_b$ in saturation. The saturation point N^* of the central subsystem is R_0/V_bS_b; and $N^* + Z/V_bS_b$ terminals will begin to saturate the terminal driven system.

An analysis leading to sketches such as Figures 9 and 10 may give some gross guidance on effects of proposed changes. For example, reducing V_iS_i for a device which is not a bottleneck (e.g., by reducing the service time or the visit ratio) will not affect the bottleneck; it will make no change in the asymptote $1/V_bS_b$ and will generally produce only a minor change in minimal response time R_0. Reducing the product V_iS_i for all the bottleneck devices will remove the bottleneck(s); it will raise the asymptote $1/V_bS_b$ and reduce R_0. However, this effect will be noticed only as long as V_bS_b remains the largest of the V_iS_i: too much improvement at device b will move the bottleneck elsewhere. These points will be illustrated by the example of the next section.

The property that $1/V_bS_b$ limits system throughput was observed by Buzen for Markovian central server networks [BUZE71a]. It was shown to hold under very general conditions by Chang and Lavenberg [CHAN74]. Muntz and Wong used it in bottleneck analysis of general stochastic queueing networks [MUNT74, MUNT75]; Denning and Kahn derived the operational counterpart [DENN75a]. Response time asymptotes were observed by Scherr for his model of CTSS [SCHE67], and by Moore for his model of MTS [MOOR71]. The concept of saturation point was introduced by Kleinrock [KLEI68], who also studied all these results in detail in his book [KLEI76].

Examples

This section illustrates the applications of bottleneck analysis for the three cases of Figures 11 through 13. For each, we consider a series of questions as might be posed by a computing center's managers, who seek to understand the present system and to explore the consequences of proposed changes.

Figure 11(a) depicts a central server system driven by a set of interactive terminals. The visit ratio equations for this network are

$$V_0 = 1 = .05V_1$$
$$V_1 = V_0 + V_2 + V_3$$
$$V_2 = .55V_1$$
$$V_3 = .40V_1$$

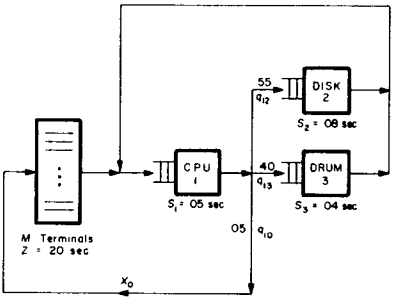

FIGURE 11(a). An example system.

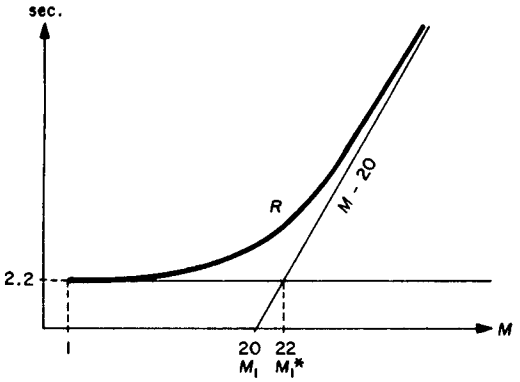

FIGURE 11(b). Response time curve.

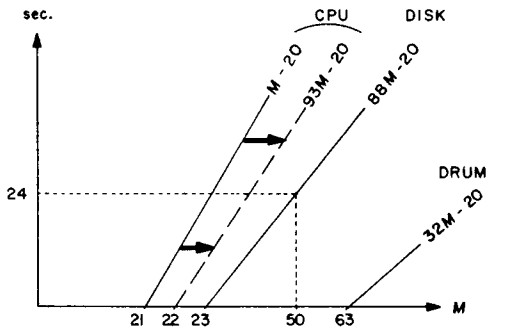

FIGURE 11(c). Response time asymptotes.

The solution is

$$V_1 = 20, \quad V_2 = 11, \quad V_3 = 8.$$

The V_iS_i products are

$V_1S_1 = (20)(.05)$
$= 1.00$ seconds (Total CPU time)
$V_2S_2 = (11)(.08)$
$= .88$ seconds (Total Disk time)
$V_3S_3 = (8)(.04)$
$= .32$ seconds (Total Drum time)

These products sum to the minimal response time

$$R_0 = 2.2 \text{ seconds.}$$

The largest product is V_1S_1; therefore $b = 1$ and the CPU is the bottleneck. (The system is "CPU bound.")

Figure 11(b) shows the asymptotes of the response time curve. The number of thinking terminals in saturation is

$$M_1 = Z/V_1S_1 = 20 \text{ terminals.}$$

The saturation point of the central subsystem is

$$N^* = R_0/V_1S_1 = 2.2 \text{ jobs.}$$

The number of terminals required to begin saturating the entire system is

$$M_1^* = 22.2.$$

Question: *Throughput is measured as 0.715 jobs/second and mean response time as 5.2 seconds. What is the mean number of users logged in during the observation period?* The interactive response time formula can be solved for the (mean) number of active terminals,

$M = (R + Z)/X_0$
$= (5.2 + 20)/(.715)$
$= 18$ terminals.

Question: *Is 8-second response time feasible when 30 users are logged in? If not, what minimum amount of CPU speedup is required?* Since the response time asymptote requires that, for $M = 30$,

$$R \geq (30)(1.00) - 20 = 10 \text{ seconds,}$$

the 8-second requirement cannot be met. If S_1' is the service time of a faster CPU, we need

$$MV_1S_1' - Z \leq 8 \text{ seconds,}$$

or

$$S_1' \leq .047 \text{ seconds.}$$

This gives a speedup factor of $S_1/S_1' \geq 1.07$; the new CPU must be at least 7% faster. Since $V_1S_1' = (20)(.047) = .93$, the system would still be CPU-bound with this faster processor (see Figure 11(c)); therefore the change is feasible.

Question: *Is 10-second response time feasible when 50 users are logged in? If not, what minimum amount of CPU speedup is required?* If the CPU were infinitely fast ($S_1 = 0$), the disk would be the bottleneck (see Figure 11(c)). In this case

$$R \geq MV_2S_2 - Z.$$

For $M = 50$,

$$R \geq (50)(.88) - 20 = 24 \text{ seconds.}$$

Thus, no amount of CPU speedup will make 10-second response feasible when $M = 50$.

•

Our second example concerns the 25-terminal time sharing system of Figure 12. A measurement has revealed that jobs require

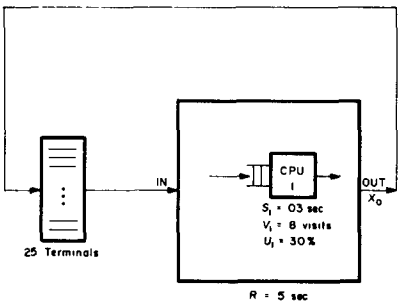

FIGURE 12. A time sharing system.

a mean total CPU time of 240 msec, that CPU utilization is 30%, and that response time is 5 seconds. The throughput and think time are

$$X_0 = U_1/V_1S_1$$
$$= (.30)/(.24) \quad = 1.25 \text{ jobs/second}$$
$$Z = M/X_0 - R$$
$$= 25/1.25 - 5 = 15 \text{ seconds.}$$

Question: *The CPU utilization seems low. What effect would a cheaper CPU of half speed have on response time?* Installing a CPU of half speed cannot increase system throughput, nor can it reduce throughput below half its original value. (If all service times, including Z, were doubled, throughput would be exactly half the original value.) Therefore,

$$0.625 \leq X_0 \leq 1.25 \text{ jobs/second}$$

after the change. (With $U_1 = X_0V_1S_1$ this implies $0.3 \leq U_1 \leq 0.6$ after the change.) Applying the response time formula,

$$5.0 \leq R \leq 25.0 \text{ seconds.}$$

The slower CPU will have no effect on response time if some other device is saturated (no change in X_0); otherwise, it could cause response time to increase by as much as a factor of five.

This example illustrates why system bottlenecks can confuse the unwary analyst. If some device (not the CPU) is saturated, lowering CPU speed will increase CPU utilization without observable effect on response time. CPU utilization can be a deceptive measure of a system's performance.

•

Our third example concerns the system of Figure 13, which has two workloads. It will illustrate how a faster device may affect performance adversely. Each batch job requires one disk-swap followed by an uninterrupted CPU execution burst averaging 1 second. Each interactive job requires an average of 10 page swaps from the disk, each followed by a short CPU burst averaging 10 msec.

Primed symbols refer to the interactive workload. It is easy to see from Figure 13 that the batch visit ratios are $V_1 = V_2 = 1$, and the interactive visit ratios are $V_1' = V_2' = 10$. The total of times required by jobs at the devices are:

	Disk	*CPU*
Batch	$V_1S_1 = .09$ sec.	$V_2S_2 = 1.0$ sec.
Interactive	$V_1'S_1' = .90$ sec.	$V_2'S_2' = .1$ sec.

Evidently the interactive workload is disk-bound and the batch workload CPU-bound. This is a good mixture of jobs in the system.

Question: *A measurement reveals that the CPU is saturated, and that interactive response time is 4 seconds. What is batch throughput? Disk utilization?* We can solve the interactive response time formula for the interactive throughput:

$$X_0' = M/(R' + Z)$$
$$= 25/(4 + 30) = .735 \text{ jobs/second.}$$

Since $X_0' = X_2'/V_2'$, the interactive component of CPU throughput is $X_2' = 7.35$ requests/second, and the utilization due to interactive jobs is

$$U_2' = X_2'S_2' = (7.35)(.01) = .074.$$

Since total utilization is 1.00, the component due to batch jobs must be $U_2 = .926$. Thus the batch throughput is

$$X_0 = X_1 = X_2 = U_2/S_2 = .926 \text{ jobs/second.}$$

The utilization of the disk is

$$X_1S_1 + X_1'S_1' = (.926)(.09) + (7.35)(.09) = .745.$$

Question: *An analysis of batch backlogs reveals that the computing center needs to support a batch throughput of at least 4.5 jobs/second. Is this feasible in the present system?* If there were no interactive jobs, the highest possible CPU batch throughput would be $X_2 = 1/S_2 = 1$ job/second. The required batch throughput cannot be achieved.

Question: *A CPU 5 times faster is available. What happens if batch throughput of*

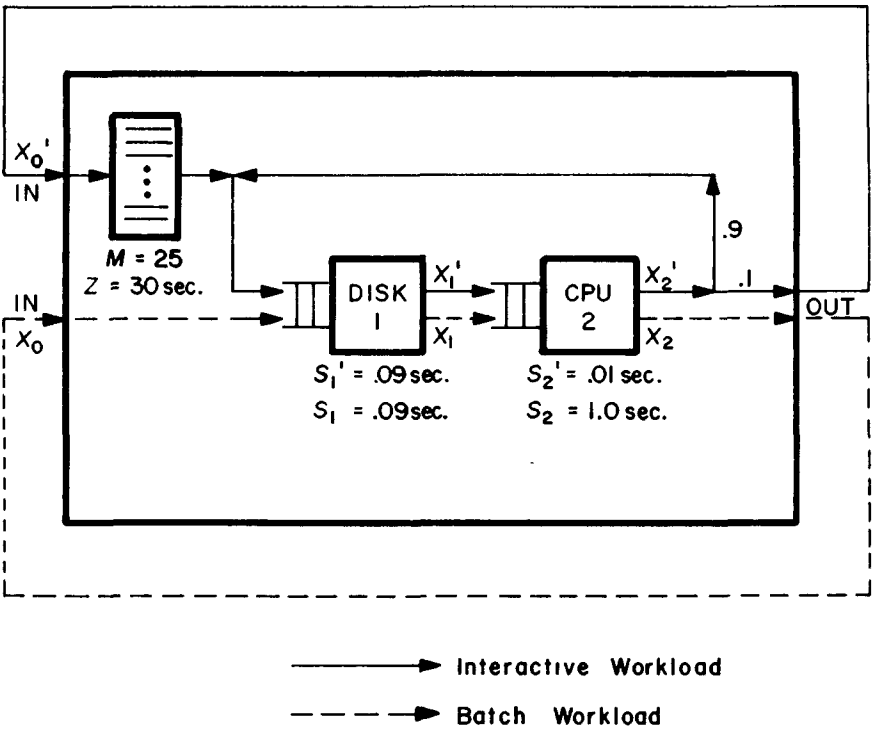

FIGURE 13. A system with two workloads.

4.5 jobs/second is achieved with this CPU? With the new CPU, the batch CPU burst becomes $S_2 = .2$ second, and the interactive CPU burst $S_2' = .002$ second. With a forced batch flow of $X_0 = X_1 = X_2 = 4.5$ jobs/second, the batch components of disk and CPU utilization would be

$$U_1 = X_1 S_1 = (4.5)(.09) = .41$$
$$U_2 = X_2 S_2 = (4.5)(.20) = .90$$

This gives bounds for the interactive components of throughput:

$$X_1' = U_1'/S_1' \leq (1 - .41)/(.09)$$
$$= 6.61 \text{ requests/second}$$
$$X_2' = U_2'/S_2' \leq (1 - .90)/(.002)$$
$$= 50 \text{ requests/second.}$$

Since $X_1' = X_2'$, the maximum possible interactive job flow at the CPU is 6.61 requests/second, and the maximum possible interactive throughput is 0.66 jobs/second. This implies a lower bound on the interactive response time:

$$R' = M/X_0' - Z \geq 25/.66 - 30 = 7.8 \text{ seconds.}$$

The interesting feature of this example is that the added capacity in the system actually hurts the performance of the interactive workload. The reason is that speeding up the CPU alleviates the batch bottleneck there, allowing more batch jobs to queue up at the disk. The additional disk queueing interferes with the already disk-bound interactive jobs. To achieve $X_0 = 4.5$ jobs/second in the batch stream without affecting interactive response time, the system needs a faster disk as well. The disk must support $X_1 + X_1' = 4.5 + 7.35 = 11.85$ requests/second total, which means that its service time must not exceed 1/11.85 second (85 msec). We will return to this example later.

Summary

By augmenting the basic operational definitions with the assumption that job flow is balanced in the system, the analyst can use visit ratios, via the forced flow law, to determine flows everywhere in the network. Response times of interactive systems can also be estimated. Table I summarized the principal equations.

When the available information is insufficient to determine flows in the network at a given load, the analyst can still approximate the behavior under light and heavy loads. For light loads the lack of queueing permits determining response time and throughput directly from the products $V_i S_i$. For heavy loads, a saturating device limits the flow at one point in the network, thereby limiting the flows everywhere; again, response time and throughput can be computed easily. For intermediate loads, further assumptions about the system are needed.

5. LOAD DEPENDENT BEHAVIOR

The examples of the preceding section were based on assumptions of invariance for service times, visit ratios, and routing frequencies. These assumptions are too rigid for many real systems. For example, if the moving-arm disk employs a scheduler that minimizes arm movement, a measurement of the mean seek time during a lightly loaded baseline period will differ significantly from the average seek time observed in a heavily loaded projection period. Similarly, the visit ratios for a swapping device will differ in baseline and projection periods having different average levels of multiprogramming.

These two examples illustrate *load dependent behavior*. To cope with it, the analyst replaces the simple invariance assumptions with *conditional invariance assumptions* that express the dependence of important parameters on the load. Instead of asserting that the disk's mean seek time is invariant in all observation periods, the analyst asserts that the mean seek time is the same in any two intervals in which the disk's queue length is the same. That is, the average seek time, whenever the disk's queue length is n (for any integer n), is assumed to be the same in both the baseline and the projection period, but the proportion of time that the queue length is n may differ in the two periods. Similarly, the swapping device's visit ratio whenever the multiprogramming level is N is assumed to be the same in both the baseline and the projection period, but the proportion of time that the multiprogramming level is N may differ in the two periods.

Tables II and III summarize the operational concepts needed to express conditional invariants and to work with load dependent behavior. Table II shows that each of the basic quantities (C_{ij}, B_i) is replaced with a function of the load. Thus $C_{ij}(n)$ counts the number of times t at which jobs request service at device j immediately on completing a service request at device i, given that $n_i = n$ just before each such time t. The function $T_i(n)$ specifies the total time during which $n_i = n$.

Table III shows the various operational measures which can be derived from the basic quantities of Table II. There are two new concepts here. The first is the *service function*, $S_i(n) = 1/X_i(n)$, which measures the mean time between completions when $n_i = n$; if device i can process several service requests at once, $S_i(n)$ can be less than the mean amount of service required by a request. The second concept is the *queue length distribution*, $p_i(n)$, which measures the proportion of time during which $n_i = n$. That the mean queue length $\bar{n}_i = W_i/T$ is equivalent to the usual definition $\mathrm{E}_{n>0}$ $np_i(n)$ can be seen from the definition of W_i in Table II.

The method of partitioning the data according to time intervals in which $n_i(t) = n$ is called *stratified sampling*. The sets of intervals in which $n_i(t) = n$ are called the "strata" of the sample. All data in the same stratum are aggregated to form the measures of Tables II and III.

Our analytic methods can deal with only two kinds of load dependent behavior: a device's service function may depend on the length of that device's queue; the visit ratios and routing frequencies may depend on the total number of jobs in the system. Thus quantities like $q_{ij}(n) = C_{ij}(n)/C_i(n)$ or

TABLE II. BASIC MEASURES

$i = 1, \ldots, K \qquad j = 0, \ldots, K$

Completion Counts		
Interdevice	$C_{ij}(n)$	= Number of times t at which a job requests service at device j next after completing a service request at device i, given $n_i = n$ just before t.
Device, conditional	$C_i(n)$	$= \sum_{j=0}^{K} C_{ij}(n)$
Device, unconditional	C_i	$= \sum_{n>0} C_i(n)$
Arrival Counts		
To a device	A_{0j}	= Number of times t at which an arriving job uses device j for its first service request.
To the system	A_0	$= \sum_{j=1}^{K} A_{0j}$
Busy Times		
Conditional	$T_i(n)$	= Total time during which $n_i = n$.
Unconditional	B_i	$= \sum_{n>0} T_i(n) = T - T_i(0)$
Routing Frequencies		
Originating in system	q_{ij}	$= \frac{1}{C_i} \sum_{n>0} C_{ij}(n)$ [Undefined if $C_i = 0$]
Originating outside system	q_{0j}	$= A_{0j}/A_0$ [Undefined if $A_0 = 0$]
Accumulated Waiting Time	W_i	$= \sum_{n>0} nT_i(n)$

TABLE III. OPERATIONAL PERFORMANCE MEASURES

$i = 1, \ldots, K \qquad j = 0, \ldots, K$

[Any quantity whose denominator would be zero is undefined]

Request Completion Rates		
Conditional	$X_i(n)$	$= C_i(n)/T_i(n)$
Unconditional, device*	X_i	$= C_i/T$
Unconditional, system	X_0	$= \sum_{i=1}^{K} X_i q_{i0} = C_0/T$
Mean Service Time Between Completions		
Conditional	$S_i(n)$	$= T_i(n)/C_i(n)$
Unconditional	S_i	$= B_i/C_i$
Queue Size Distribution	$p_i(n)$	$= T_i(n)/T$
Utilization	U_i	$= B_i/T = 1 - p_i(0)$
Mean Queue Length	$\bar{n}_i$	$= W_i/T$
Mean Response Time	R_i	$= W_i/C_i$

* Note that $X_i = \sum_{n>0} X_i(n)\, p_i(n)$ is an identity.

$V_i(n) = C_i(n)/C_0$ cannot be handled. Because routing frequencies and visit ratios ordinarily depend only on the intrinsic demands of jobs and not on local queue lengths, such quantities are of little interest. However, quantities like $q_{ij}(N)$ and $V_i(N)$ do arise frequently—e.g., when the demand for swapping depends on the multiprogramming level N [DENN75b, DENN76]—and these quantities can be handled in the models.

Example: An initially idle device i is observed for 16 seconds. At $t = 0$, 1, 2, and 3 jobs arrive. Each job requires exactly 4 seconds of service. (This implies that completions occur at times $t = 4$, 8, 12, and 16.) The resulting $n_i(t)$ is sketched in Figure 14. The load dependent quantities are:

n	$C_i(n)$	$T_i(n)$	$S_i(n)$	$X_i(n)$	$p_i(n)$
0	0	0	–	–	0
1	1	5	5	1/5	5/16
2	1	5	5	1/5	5/16
3	1	5	5	1/5	5/16
4	1	1	1	1	1/16
Totals:	$C_i = 4$	$T = 16$	–	–	16/16

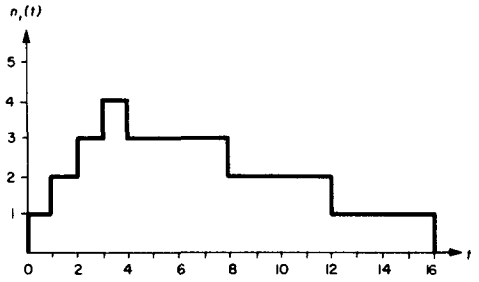

FIGURE 14. A queueing at a device.

Note that $C_i(0) = 0$ since departures from an idle device are impossible. The unconditional mean service time is, as expected,

$$S_i = B_i/C_i = 16/4 = 4 \text{ seconds.}$$

Notice, however, that $S_i(n)$ is *not* 4 for any value of n. The accumulated waiting time is:

$$W_i = \sum_{n>0} n\, T_i(n) = 34 \text{ job-seconds.}$$

Therefore the mean queue length and response time are:

$$\bar{n}_i = W_i/T = 34/16 = 2.125 \text{ jobs} \qquad R_i = W_i/C_i = 34/4 = 8.5 \text{ seconds.}$$

If the arrivals were synchronized with the departures—i.e., occurring at $t = 0, 4, 8$, and 12—then $n_i(t) = 1$ throughout the 16-second observation period. In this case $S_i(1) = 4$ seconds, $S_i(n) = 0$ for $n > 1$, $\bar{n}_i = 1$, and $R_i = 4$ seconds.

This example illustrates two important points. First, the amount of queueing depends on the nature of the arrivals and departures. Different patterns of arrival of the same jobs may produce different measures of mean queue length and response time, even while producing the same throughput and utilization. This is why an analyst who seeks to measure queueing (e.g., with $p_i(n)$, or $\bar{n}_i$, or R_i) needs to make more assumptions.

The example also illustrates that the observed service function $S_i(n)$ depends on the arrival pattern, even though all the jobs may have identical service requirements. There is, in general, no simple relationship between $S_i(n)$ and the service times required by jobs.

6. SOLVING FOR STATE OCCUPANCIES

The assumption of job flow balance is insufficient to find flows in a closed network, or to compute response times accurately. These quantities depend on how jobs distribute throughout the network; the job flow balance equations do not. To represent the job distribution, we define a "state" of the system: a vector

$$\mathbf{n} = (n_1, \ldots, n_K)$$

in which $n_i \geq 0$ is the number of jobs in queue at device i, and $N = n_1 + \ldots + n_K$ is the total number of jobs in the system.

The set of all states $\mathbf{n}$ is called the "system state space." The number of possible states is usually quite large. We observe that each state can be encoded by a binary string of N 1s and $K-1$ 0s,

$$\underbrace{11\ldots1}_{n_1}0\underbrace{11\ldots1}_{n_2}0\ldots0\underbrace{11\ldots1}_{n_K} \quad ;$$

the number of such strings is the number of permutations of N indistinguishable objects and $K-1$ indistinguishable objects, namely,

$$L = \frac{(N+K-1)!}{N!\,(K-1)!} = \binom{N+K-1}{K-1}.$$

The number of possible states, L, can be large even for relatively small systems; for example, when $N = K = 10$, L is approximately 92,000. For an open system, where N itself can change, the number of possible states can be considerably larger. We will be greatly concerned with the computational feasibility of solutions over this state space.

State Transition Balance

In the following discussion, $\mathbf{k}$, $\mathbf{n}$, and $\mathbf{m}$ denote distinct system states. If the system moves from state $\mathbf{n}$ to state $\mathbf{m}$ without passing through any observable intermediate state, a *one-step state transition* (from $\mathbf{n}$ to $\mathbf{m}$) has occurred. Let $C(\mathbf{n}, \mathbf{m})$ denote the number of one-step state transitions observed from $\mathbf{n}$ to $\mathbf{m}$; since no transition implies no state change, $C(\mathbf{n}, \mathbf{n}) = 0$.

Now, if job flow is balanced, the number of arrivals at every device is the same as the number of departures. This means that $n_i(0) = n_i(T)$ for each device i, or equivalently that $\mathbf{n}(0) = \mathbf{n}(T)$. In moving from its initial state to its final state, the system must leave every state once for every entry. Hence job flow balance is equivalent to the

Principle of State Transition Balance: The number of entries to every state is the same as the number of exits from that state during the observation period.

From now on we will use the term *flow balance* to mean that arrivals equal departures at every device or system state. With the flow balance principle we can write "conservation of transition" equations:

$$\sum_{\mathbf{k}} C(\mathbf{k}, \mathbf{n}) = \sum_{\mathbf{m}} C(\mathbf{n}, \mathbf{m}) \quad \text{all} \quad \mathbf{n}.$$

For given $\mathbf{n}$, both sides of this equation are 0 if and only if $T(\mathbf{n}) = 0$.

If we use these equations without flow balance, the only error would be a + 1 (or −1) term missing on the right side if $\mathbf{n}$ is the final (or initial) state of the system for the observation period. This error is not significant if the initial and final states are visited frequently during the observation period. (The error is zero if the initial and final states are the same—i.e., flow is balanced.) As we noted in discussing job flow balance, choosing the observation period so that flow is balanced is not a new notion.

The "transition rate" from $\mathbf{n}$ to $\mathbf{m}$ is the number of transitions per unit time while $\mathbf{n}$ is occupied:

$$r(\mathbf{n}, \mathbf{m}) = C(\mathbf{n}, \mathbf{m})/T(\mathbf{n});$$

it is not defined if $T(\mathbf{n}) = 0$. The transition conservation equations can be re-expressed as

$$\sum_{\mathbf{k}} T(\mathbf{k})\, r(\mathbf{k}, \mathbf{n}) = T(\mathbf{n}) \sum_{\mathbf{m}} r(\mathbf{n}, \mathbf{m}),$$

for all $\mathbf{n}$ from which exit rates $r(\mathbf{n}, \mathbf{m})$ are defined; note $T(\mathbf{n}) = 0$ if $r(\mathbf{n}, \mathbf{m})$ is not defined. Substituting $T(\mathbf{n}) = p(\mathbf{n})T$ and cancelling T, we obtain the

STATE SPACE BALANCE EQUATIONS

$$\sum_{\mathbf{k}} p(\mathbf{k})\, r(\mathbf{k}, \mathbf{n}) = p(\mathbf{n}) \sum_{\mathbf{m}} r(\mathbf{n}, \mathbf{m})$$

for all $\mathbf{n}$ in which $r(\mathbf{n},.)$ is defined.

Because the $T(\mathbf{n})$ sum to T, we can augment these equations with the normalizing condition

$$\sum_{\mathbf{n}} p(\mathbf{n}) = 1.$$

If the system can move from any $\mathbf{n}$ to any $\mathbf{m}$, then these are $L-1$ linearly independent balance equations; only one set of $p(\mathbf{n})$ can satisfy them and the normalizing condition simultaneously. (Our definitions imply $p(\mathbf{n}) = 0$ for states not included in these balance equations.)

Solving the Balance Equations

The state space balance equations are operational relationships expressing the values of $p(\mathbf{n})$ in terms of the $r(\mathbf{n}, \mathbf{m})$. This form of expression is generally not useful since the analyst does not have the values of $r(\mathbf{n}, \mathbf{m})$. Instead, the analyst wishes to express the $r(\mathbf{n}, \mathbf{m})$ in terms of available quantities such as visit ratios and the service functions, and then solve for the $p(\mathbf{n})$.

To avoid a lot of symbol manipulation, we will outline the steps of the solution; the details are found in [DENN77a]. The solution uses two additional assumptions about system's behavior. The first is:

One-Step Behavior: The only observable state changes result from single jobs either entering the system, or moving between pairs of devices in the system, or exiting from the system.

The hypothesis of one-step behavior asserts that simultaneous job-moves will not be observed; it reduces the number of nonzero rates $r(\mathbf{n}, \mathbf{m})$ that must be considered. For example, when a job moves from device i to device j, the system moves from state $\mathbf{n}$ to its "neighbor," $\mathbf{n}_{ij}$, where

$$\mathbf{n} = (n_1, \quad , n_i, \quad , n_j, \quad , n_K)$$
$$\mathbf{n}_{ij} = (n_1, \cdots, n_i - 1, \cdots, n_j + 1, \quad , n_K).$$

The nonzero transition rates correspond to (i, j) job moves under the one-step assumption. Thus there are about LK^2 rates to specify, rather than L^2. (L is the size of the state space.) With this assumption, $r(\mathbf{n}, \mathbf{n}_{ij})$ depends only on the rate of job flow from device i to device j. The one-step property is met in many real systems.

To specify the transition rates in terms of routing frequencies and service functions, we need to remove the conditioning on the total system state. The assumptions that do this are called "homogeneity" because they assert that, for given n_i, device i is influenced equally by all system states:

Device Homogeneity: The output rate of a device is determined completely by its queue length, and is otherwise independent of the system's state.

Routing Homogeneity: The routing frequencies for a given total load (N) are independent of the system's state.

Device homogeneity is a reasonable assumption for systems in which no device can block any other.[3] Routing homogeneity is a reasonable assumption for most systems because job transitions generally depend on the intrinsic demands of jobs but not on instantaneous queue lengths.

The stochastic counterpart of routing homogeneity is the assumption that job transitions among devices follow an ergodic Markov chain. The stochastic counterpart of device homogeneity is that interdeparture times of a device are exponentially distributed. Because they are operationally testable, homogeneity assumptions are fundamentally different from their stochastic counterparts. The example of the next subsection (based on Figure 15) illustrates a homogeneous system. It is impossible to determine whether or not this system satisfies any stochastic assumptions.

Device homogeneity asserts that the ratio $C(\mathbf{n}, \mathbf{n}_{ij})/T(\mathbf{n})$ is the same as $C_{ij}(n_i)/T_i(n_i)$. Routing homogeneity asserts that the count $C_{ij}(n_i)$ is the same as $q_{ij}C_i(n_i)$. Both assertions imply

$$r(\mathbf{n}, \mathbf{n}_{ij}) = q_{ij}/S_i(n_i).$$

With this substitution, the state space balance equations reduce to a set of "homogenized balance equations" [DENN77a]. The resulting solution for $p(\mathbf{n})$ is of the so-called *product form* because it separates into K factors, one for each device, as shown in the box below [BASK75, COFF73, GORD67, JACK63, KLEI75].

PRODUCT FORM SOLUTION

$$p(\mathbf{n}) = F_1(n_1)F_2(n_2) \cdots F_K(n_K)/G$$

where the factor for device i is

$$F_i(n) = \begin{cases} 1, & n = 0 \\ X_i^n S_i(n)S_i(n-1) \cdots S_i(1), & n > 0 \end{cases}$$

and G is a normalizing constant. The $S_i(n)$ are the service functions. The X_i are a solution of the job flow balance equations; for an open system $X_i = V_i X_0$, and for a closed system $X_i = V_i$ will do.

Jackson showed that, for open systems, this solution is separable further into the product of the queue-length distributions of the individual devices [JACK57, KLEI75]. The operational counterpart of Jackson's result is proved in [DENN77a]:

$$p(\mathbf{n}) = p_1(n_1)p_2(n_2) \cdots p_K(n_K)$$

where

$$p_i(n) = F_i(n)/G_i$$

$$G_i = \sum_{n=0}^{N} F_i(n)$$

and N is the maximum number of jobs observed in any queue of the system. The constant G of the product form solution is $G_1G_2 \cdots G_K$. (Because each n_i can range from 0 to N in an open system, we can interchange the sums and products in the definition of G, thereby manipulating G into the product of the G_i.) Note that $p_i(n)$ is exactly the queue length distribution obtained by considering device i as an isolated, single-device network with throughput X_i [BUZE76b]. Jackson's result shows that the performance quantities of device i in an open network are easy to calculate from these formulae.

For a closed system, $p(\mathbf{n})$ cannot be separated into the product of the individual queue length distributions. This is because the queue lengths are not independent and the products and sums in the definition of G cannot be interchanged. More complex computations are required for closed systems.

To simplify calculations analysts sometimes use the operational assumption called *homogeneous service times* (HST). It asserts that the conditional service times $S_i(n)$ all have the same value S_i, which is the (unconditional) mean time between completions. (That is, $S_i(n) = S_i$ for all n.) In this case the factor $F_i(n)$ becomes $(V_iS_i)^n$ for a closed system and $(X_iS_i)^n = U_i^n$ for an open system. In obtaining parameters for the HST solution, the analyst does not need to know each V_i and S_i; he needs only V_iS_i, the mean total time a job requires at device i. As illustrated in the next subsection,

[3] Examples of blocking are multiple CPUs that can lock one another out of the scheduling queues, or a store-and-forward communications processor that cannot transmit a message to the next node because no buffer space is available at that node. Device homogeneity can also be a poor approximation in a closed system if some device has a very high variance in the times between the completions of requests for its service.

the HST assumption may cause significant errors in the queue length distributions.

Open and closed networks with more than one class of jobs (workloads) exhibit similar product form solutions. The major difference is that there is a factor corresponding to each job class at each device [BASK75].

An Example

Figure 15 illustrates a simple system with $K = 2$ and $N = 2$. The timing diagram shows a possible behavior. The numbers inside the diagram show which job is using a device, and shaded portions show idleness. The observation period lasts 20 seconds. All three possible states—$(n_1 n_2) = 20$, 11, and 02—are observed; they are displayed along the bottom of the diagram.

We will compare the actual performance quantities with the model's estimates. The actual proportions of time of state occupancy are

$$p(20) = T(20)/T = 16/20 = .80$$
$$p(11) = T(11)/T = 3/20 = .15$$
$$p(02) = T(02)/T = 1/20 = .05$$

The transition rates are:

$$r(20,11) = T(20,11)/T(20) = 2/16 = .125$$
$$r(11,02) = T(11,02)/T(11) = 1/3 = .333$$
$$r(02,11) = T(02,11)/T(02) = 1/1 = 1.000$$
$$r(11,20) = T(11,20)/T(11) = 2/3 = .667$$

The state space balance equations are:

$$p(11)(2/3) = p(20)(2/16)$$
$$p(20)(2/16) + p(02)(1) = p(11)(1/3 + 2/3)$$
$$p(11)(1/3) = p(02)(1)$$
$$p(20) + p(11) + p(02) = 1$$

It is easily verified that the actual $p(\mathbf{n})$ satisfy these equations.

Because the initial and final states of the observation period are the same, the system is flow balanced. Because there are no routing choices ($q_{12} = q_{21} = 1$), and because the state is fully determined by either queue length ($n_1 = 2 - n_2$), the system is homogeneous.[4] Therefore, the product form solution is exact. We will verify this. The device service functions are:

n	$S_1(n)$	$S_2(n)$	
1	3/1 = 3.0	3/2 = 1.5	(seconds)
2	16/2 = 8.0	1/1 = 1.0	

Since $V_1 = V_2 = 1$, the device factors are:

n	$F_1(n)$	$F_2(n)$
0	1.0	1.0
1	3.0	1.5
2	24.0	1.5

The normalizing factor is

$$G = F_1(2)F_2(0) + F_1(1)F_2(1) + F_1(0)F_2(2)$$
$$= (24.0)(1.0) + (3.0)(1.5) + (1.0)(1.5)$$
$$= 30$$

[4] This illustrates that a system can be homogeneous without its devices having to satisfy assumptions of exponentially distributed interdeparture times.

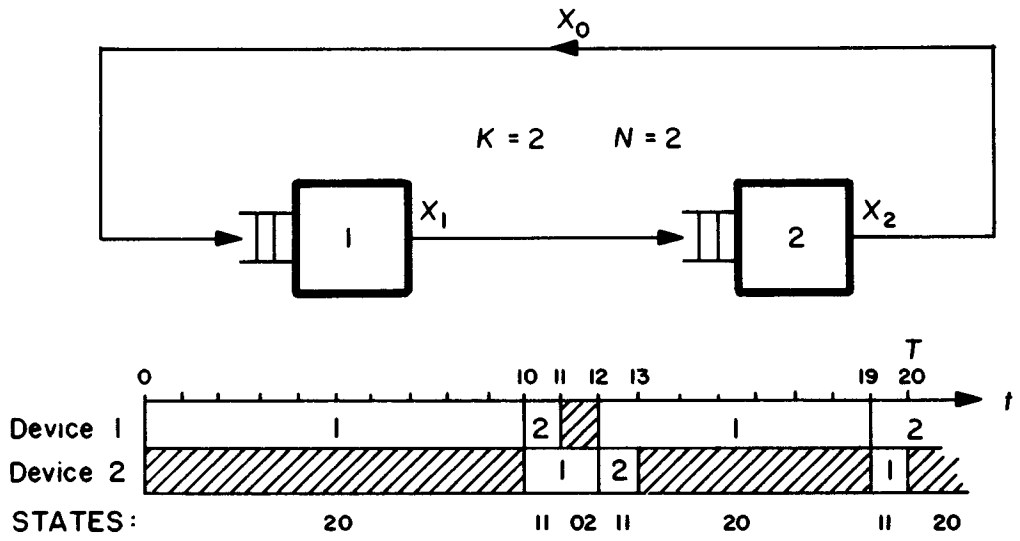

FIGURE 15. A two-device system and observed behavior.

The state occupancies are as observed:

$$p(20) = F_1(2)F_2(0)/G = (24.0)(1.0)/30 = .80$$
$$p(11) = F_1(1)F_2(1)/G = (3.0)(1.5)/30 = .15$$
$$p(02) = F_1(0)F_2(2)/G = (1.0)(1.5)/30 = .05$$

Next, we will compare with the solution based on homogeneous service times (HST). Because the service time functions are not constant, this solution is not exact for this system. The unconditional mean service times are

$$S_1 = B_1/C_1 = 19/3 = 6.333 \text{ seconds}$$
$$S_2 = B_2/C_2 = 4/3 = 1.333 \text{ seconds}$$

The HST transition rates are

$$r(20,11) = r(11,02) = 1/S_1$$
$$r(02,11) = r(11,20) = 1/S_2$$

There are significant errors between these and the actual rates:

$r(\mathbf{n}, \mathbf{m})$	*Actual*	*HST Model*	*Error*
$r(20,11)$	.125	.158	+26.4%
$r(11,02)$	.333	.158	−52.5%
$r(02,11)$	1.000	.750	−25.0%
$r(11,20)$	.667	.750	+12.5%

In the HST model, the device factors are of the form $F_i(n) = (V_iS_i)^n$, which works out to be:

n	$F_1(n)$	$F_2(n)$
0	1	1
1	19/3	4/3
2	361/9	16/9

The normalizing factor is

$$G = F_1(2)F_2(0) + F_1(1)F_2(1) + F_1(0)F_2(2)$$
$$= (361/9)(1) + (19/3)(4/3) + (1)(16/9)$$
$$= 453/9$$

With the formula $p(n_1n_2) = F_1(n_1)F_2(n_2)/G$ we can calculate the state occupancies according to the HST model:

$p(n_1n_2)$	*Actual*	*HST Model*	*Error*
$p(20)$	.800	.797	−0.4%
$p(11)$	.150	.168	+11.8%
$p(02)$	.050	.035	−29.3%

This shows that the HST model can make significant errors in the queue length distributions, e.g., $p_1(n) = p(n, 2-n)$. However these errors are less serious than the ones in transition rates, and they hardly affect the model's estimates of utilizations ($U_1 = 1 - p(02)$, $U_2 = 1 - p(20)$):

U_i	*Actual*	*HST Model*	*Error*
U_1	.950	.965	+1.5%
U_2	.200	.203	+1.5%

The model estimates that $X_0 = U_1/S_1 = U_2/S_2 = .152$ jobs/second, which is 1.5% higher than actual. The mean queue lengths are calculated as

$$\tilde{n}_1 = 2 \cdot p(20) + 1 \cdot p(11)$$
$$\tilde{n}_2 = 2 - \tilde{n}_1$$

They work out as follows:

$\tilde{n}_i$	*Actual*	*HST Model*	*Error*
$\tilde{n}_1$	1.750	1.762	+0.7%
$\tilde{n}_2$	.250	.238	−4.8%

The mean response time in the system is $R = 2/X_0$. The HST model estimates that $R = 13.2$ seconds, which is about 1.3% less than the actual of 13.3 seconds.

This example illustrates what is observed frequently in practice: the HST model gives excellent approximations of utilizations and system response times, fair to good approximations for mean queue lengths (and response times) at devices; and fair to poor approximations to the queue length distributions.

Accuracy of the Analysis

Flow balance, one-step behavior, and homogeneity are the weakest known assumptions leading to a product form solution for $p(\mathbf{n})$. The balance assumptions introduce no error if the observation period is chosen so that the initial state of the system is the same as the final. Otherwise, the error will be small if the observation period starts and ends on frequently visited states.

One-step behavior is a property of many real systems. In many others, the number of simultaneous job transitions are a small fraction of the total number of state changes. (There are, however, systems in which "bulk arrivals" allow groups of jobs to make transitions together, in violation of

the one-step assumption. Such cases can be treated by introducing new operational assumptions to characterize the bulk arrivals.)

Homogeneity is often a reasonable approximation. In systems where devices cannot block each other, a device's service function may not be influenced significantly by queueing at other devices. Routing frequencies seldom depend on local queue lengths. If used, the homogeneous service time (HST) approximation can introduce further errors; these errors affect queue length distributions the most, utilizations the least. HST models seldom estimate utilizations with errors exceeding 10%, but they may make larger errors in estimating mean queue lengths (as much as 30%).

As we will see in the section on decomposition, device homogeneity is equivalent to the assumption that a device's service function $S_i(n)$ is the same whether the device is observed online, or offline under a constant load of n requests. For single-server devices an offline experiment will report that $S_i(n)$ is the mean of request sizes regardless of the queue length—an HST assumption. In reality, the relation between the distribution of request sizes and the service function is more complex.

7. COMPUTATION OF PERFORMANCE QUANTITIES

The product form solution for $p(\mathbf{n})$ is mathematically neat but not obviously useful: computing a utilization U_i, for example, seems to require first computing the normalizing constant G, then summing the $p(\mathbf{n})$ for those $\mathbf{n}$ in which $n_i \geq 1$. For a closed system with homogeneous service times, a direct computation requires

$$L = \binom{N+K-1}{K-1}$$

additions, and $N - 1$ multiplications for each addition—a total of LN arithmetic operations. This computation would be prohibitively expensive for reasonable choices of N and K.

In 1971 Buzen developed a fast algorithm for computing G [BUZE71b, BUZE73]. For a system with homogeneous service times, it requires about $2KN$ arithmetic operations; a utilization (U_i) can be computed with 2 more operations, and a mean queue length ($\bar{n}_i$) with $2N$ more. For systems whose devices have load dependent service functions, the computation of G increases to about N^2K operations.

The next two subsections review the essentials of these computations for two kinds of systems with homogeneous service times: a closed system and a terminal-driven system. A third subsection surveys the general algorithms and returns to the example of Figure 13.

Closed System with Homogeneous Service Times

Figure 16 shows the essence of the result developed by Buzen [BUZE71b, BUZE73]. The algorithm fills in numbers in a two-dimensional matrix g. The columns of g correspond to devices, rows to loads. The computation starts with 1s in the first row and 0s in the first column below the first row. A typical interior element is computed from

$$g(n, k) = g(n, k-1) + Y_k g(n-1, k),$$

where $Y_k = V_k S_k$. The normalizing constant G is $g(N, K)$. It can be computed in $2KN$ arithmetic operations.

The algorithm actually requires much less storage than Figure 16 suggests. Because the matrix can be filled one column at a time, we need only store the column currently being computed. Let $G[0 \quad N]$, initially 0, denote a vector array representing a current column of g, and let $Y[1 \cdot K]$ denote another vector containing

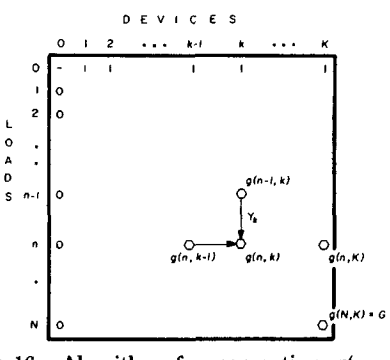

FIGURE 16. Algorithm for computing $g(n, k)$ of closed system with homogeneous service times.

$V_1S_1, \cdots, V_KS_K$. Then the algorithm is

```
{initialize:} G[0] := 1
for k:= 1 to K do {compute kth column}

    for n := 1 to N do
        {G[n − 1]  contains g(n − 1, k);
         G[n] contains g(n, k − 1)}
        G[n] := G[n] + Y[k]*G[n − 1]
    end
end
```

When this procedure terminates, $G[N]$ contains the normalizing constant.

The importance of this algorithm is not only that it computes $g(n,K) = G[n]$ quickly, but that the mean queue lengths and the utilizations can be expressed as simple functions of $g(n, K)$ [BUZE71b]. The results are shown below.

Proportion of time $n_i \geq n$	$Q_i(n)$	$= Y_i^n \dfrac{g(N-n, K)}{g(N, K)}$
Utilization	U_i	$= Q_i(1) = Y_i \dfrac{g(N-1, K)}{g(N, K)}$
System throughput	X_0	$= \dfrac{g(N-1, K)}{g(N, K)}$
Mean queue length	$\bar{n}_i$	$= \sum_{n=1}^{N} Y_i^n \dfrac{g(N-n, K)}{g(N, K)}$

The formula for $\bar{n}_i$ can be rewritten as a recursion,

$$\bar{n}_i(N) = U_i(N)(1 + \bar{n}_i(N-1)),$$

with initial condition $\bar{n}_i(0) = 0$. This shows that $\bar{n}_i(N)$ can be calculated iteratively with $2N$ arithmetic operations.

Example: For the example of Figure 15, we had:

$$Y_1 = V_1S_1 = 19/3 = 6.33 \text{ seconds}$$
$$Y_2 = V_2S_2 = 4/3 = 1.33 \text{ seconds.}$$

The table below shows the matrix g for loads $N = 1, \quad , 5$:

		0	1	2	$X_0(N)$
	0	-	1.00	1.00	-
	1	0	6.33	7.67	.130
	2	0	40.1	50.3	.152
N	3	0	254.	321.	.157
	4	0	1609.	2037.	.158
	5	0	10190.	12906.	.158

The numbers in the X_0 column are computed from the system throughput formula for the given N. For example, when $N = 2$,

$$X_0(2) = \frac{g(1,2)}{g(2,2)} = 7.67/50.3 = .152$$

which is the value obtained previously for the HST model of Figure 15. The mean queue length at device 1 when $N = 2$ is

$$\bar{n}_1 = \sum_{n=1}^{2} Y_1^n \frac{g(2-n, 2)}{g(2, 2)} = \frac{(6.33)(7.67) + (6.33)^2(1.00)}{50.3} = 1.762$$

which is the same as the value obtained previously. Observe that the model predicts that X_0 saturates at $1/V_1S_1 = 0.158$ jobs/second for $N \geq 4$. The actual system is on the verge of saturation when $N = 2$, for $U_1 = 0.95$.

Terminal Driven System with Homogeneous Service Times

Now we consider an interactive system of the form of Figure 17. Each of the M terminals has think time Z. The number of active jobs is denoted by N, and the number of thinking terminals by M-N. The central subsystem has K devices with homogeneous service times and visit ratios independent of N.

By treating the terminals as a "device" whose service function is Z/n when there are n thinkers, we can employ efficient computational procedures to compute a normalizing constant for this system [WILL76]. The algorithm fills in a matrix h as suggested in Figure 18. The rows correspond to numbers of terminals, columns to devices in the central subsystem. Initially row 0 and column 0 are all 1s. A typical interior element is computed from

$$h(m, k) = h(m, k-1) + \frac{mY_k}{Z} h(m-1, k),$$

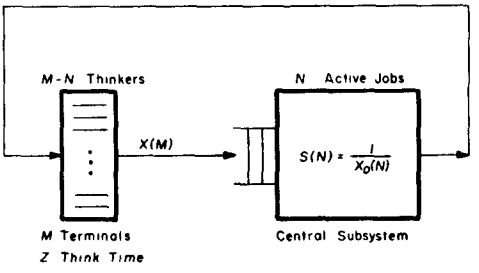

FIGURE 17. Terminal-driven system with central subsystem replaced by an equivalent device.

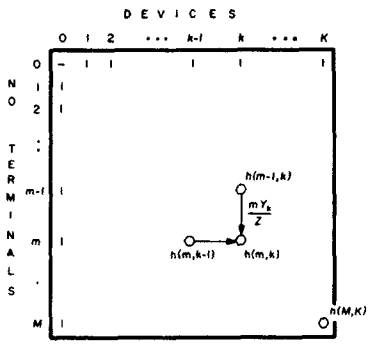

FIGURE 18. Algorithm for computing $h(m, k)$ of terminal driven system with homogeneous service times.

where $Y_k = V_k S_k$. When this computation terminates, the performance measures can be computed from the formulae below.

Proportion of time central subsystem is idle	$p(0)$	$= 1/h(M, K)$
Throughput	$X(M)$	$= \frac{M}{Z}\frac{h(M-1, K)}{h(M, K)}$
Response time	$R(M)$	$= M/X(M) - Z$
Mean active load	$\bar{N}$	$= M - ZX(M)$

Example: We used this algorithm to compute $h(M, K)$ for the system of Figure 11(a), calculating the response time $R(M)$ for $M = 1, 2, \cdots, 50$. The result is plotted in Figure 19. Note that the curve approaches the asymptote $M - 20$, is predicted in Figure 11(b). For $M = 18$, the formulae in the box yield these values:

$$X(18) = .715 \text{ jobs/second}$$
$$R(18) = 5.2 \text{ seconds}$$
$$p(0) = .062$$
$$\bar{N} = 3.7 \text{ jobs}$$

We used these throughput and response time values previously in our discussion of Figure 11. The model calculates that the central subsystem is idle for 6.2% of the time and that there are 3.7 active jobs on average.

Figure 17 suggests that the entire central subsystem can be replaced with an equivalent device whose service function is $S(N) = 1/X_0(N)$, $X_0(N)$ being the throughput of the central subsystem under a constant load N. Only if the actual system is homogeneous will this replacement be exact. (We will explain why in the next section.) With homogeneity, the distribution of active loads, $p(N)$, satisfies the state-space balance equation

$$p(N)X_0(N) = p(N-1)\frac{M-N+1}{Z}.$$

Using the value of $p(0)$ from the box, we

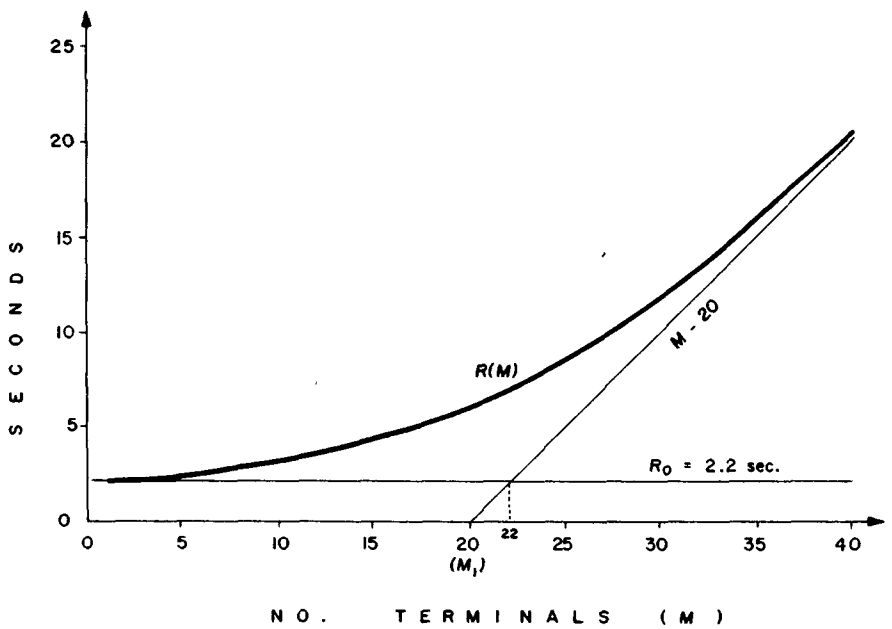

FIGURE 19. Response time for example network.

can calculate $p(N)$ iteratively from this balance equation. Note, however, that the measures shown in the box do not require calculating $X_0(N)$ or $p(N)$ first.

Example: Figure 20 shows the throughput function $X_0(N)$ for the central subsystem of the example of Figure 11(a), computed as if this subsystem were closed. The figure also displays $p(N)$, computed for $M = 18$ using the iterative formula. The straight line is the equation $(M - N)/Z$, which is the job submission rate of the thinking terminals; this line crosses $X_0(N)$ at $N = 3$, which is 19% less than the model's $\bar{N} = 3.7$. The crossing point represents the most favored value of N, the load that balances job submission rate with job completion rate. In many cases it is a good estimate of $\bar{N}$ (see [COUR75, COUR77]). The tick-marks indicate crossing points for other values of M. In the case shown, the model estimates that the load does not exceed 6 jobs for 97% of the time—nearly all the time at most 1/3 of the terminals are awaiting a response. (The tendency for $p(N)$ to be a normal distribution has been confirmed as long as the variance of execution times is less than 10 times the mean [BALB78].)

General Systems

The computational procedures have been studied and refined extensively. They now deal with open and closed networks, various queueing disciplines at the individual devices, and multiple classes of jobs (workloads) with class changes allowed. Some are available today as commercial queueing network evaluator packages [BUZE78b]. Comprehensive treatments of these algorithms have been given by Shum [SHUM76] and by Reiser and Sauer [REIS78]. One of the fastest algorithms has been reported by Balbo, et al. [BALB77]. (See also [CHAN75a, GELE76a, HERZ75, REIS75, WILL76].) The remarkable speed with which the performance quantities of complex networks can be calculated is an important reason that queueing network models have become so widely used.

Example: We have applied these algorithms to the problem of Figure 13 under the additional assumption that the average batch multiprogramming level is 10. The results are summarized in Table IV. The exact results confirm the approximate analysis given before: the faster CPU helps the batch workload, as planned, but hurts the interactive workload. The disk queue is longer after the change because the CPU is no longer the bottleneck for the batch workload; but the longer disk queue interferes with the I/O-bound interactive jobs, thereby increasing interactive response time from 4 to 10 seconds. Whereas the total

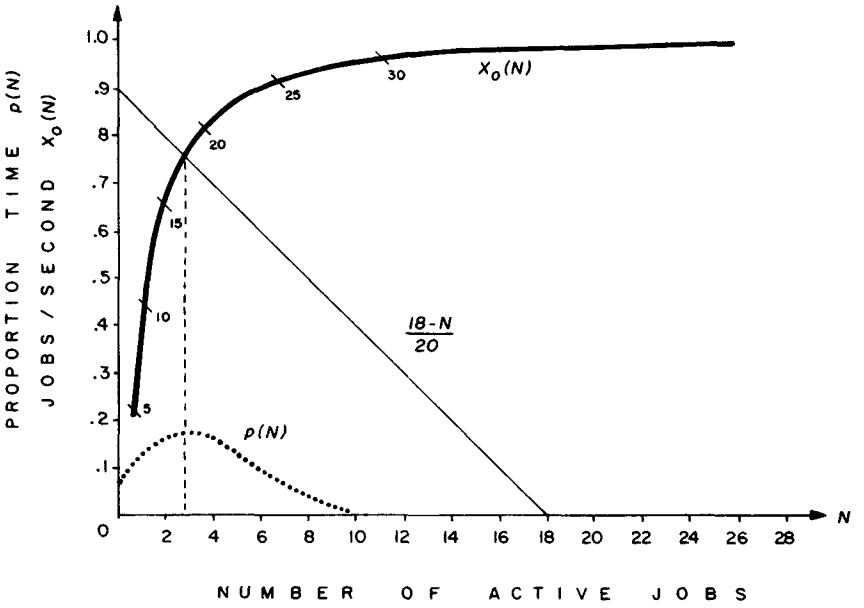

FIGURE 20. Load distribution in example network.

TABLE IV. EXACT RESULTS FOR EXAMPLE OF FIGURE 13

	Original System			CPU 5× Faster		
	Int.	*Batch.*	*Total*	*Int.*	*Batch*	*Total*
Throughput (job/sec)	.735	.926	1.66	.623	4.64	5.26
Response Time (sec)	4.0	10.8	—	10.1	2 2	—
CPU utilization (%)	7.4	92.6	100.0	1.2	92.8	94.0
CPU Queue Length (mean jobs)	.9	9.8	10.7	1	5.2	5 3
Disk Utilization (%)	66.2	8.3	74.5	56.1	41.8	97.9
Disk Queue Length (mean jobs)	2.1	.3	2.4	6.3	4.8	11 1

throughput increased by a factor of 3.2 (from 1.66 to 5.26 jobs/second), the batch throughput increased by a factor of 5.01 (from .926 to 4.64 jobs/second). The batch throughput was speeded up by more than the CPU speedup factor—at the expense of the interactive workload.

This example illustrates why it is safer to employ the analytic tool than to trust one's untrained intuition. Many analysts find this example surprising, until they realize that the ratios of throughputs for the different workloads are not invariant under the change of CPU.

8. DECOMPOSITION

The formulae derived from the product form solution will be more accurate when used with the *online service functions* of devices, obtained by stratified sampling while devices are in operation. However, for performance prediction, the analyst must estimate the actual service functions from the data on request sizes, a task complicated by the lack of a simple relationship between request sizes and intercompletion times. Decomposition is an important method of establishing such a relation.

Offline Experiments

Figure 21 shows that decomposition can be applied to a subsystem of one or more devices. The principle is to study the subsystem offline, that is, independently of any interactions with its environment. To do this, the analyst subjects the subsystem to a series of controlled experiments; each is based on measuring the subsystem's output rate when put under constant load. The "experiment" may be conceptual rather than physical, conducted with a model rather than a measurement.

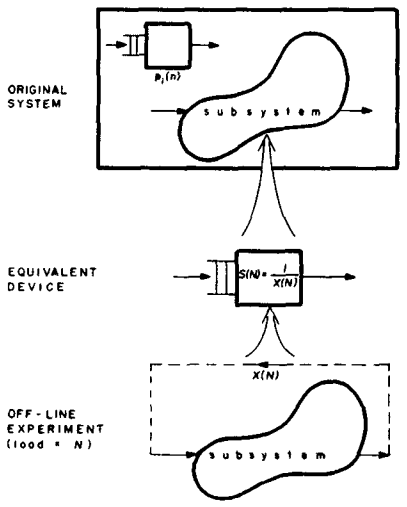

FIGURE 21. Principle of decomposition.

In an offline experiment, the subsystem is operated under a constant load of N jobs. Immediately after each job completion, the analyst adds another job to keep the load equal to N. If, during T seconds of such an experiment, the analyst counts C completions, he sets the conditional output rate to be $X(N) = C/T$. The subsystem is then replaced by an "equivalent device" whose load-dependent service function is $S(N) = 1/X(N)$. Note that arrivals and completions are synchronized in this kind of experiment.

A subsystem may be replaced exactly by an equivalent device only when the subsystem's output rate is completely determined by its given load (N) and is otherwise independent of the state of the whole system. In this case the distribution of jobs outside the subsystem cannot be influenced by the distribution of jobs inside, and the queue distribution $p_i(n)$ of any external device is the same whether the equivalent device or the real subsystem is online. It also means

that the subsystem responds the same to any environment that subjects it to the given N and, hence, the offline experiment must reveal the online service function. In other words, homogeneity asserts exact decomposability for a device.

It is clear that any subnetwork of a system whose devices and routing frequencies are homogeneous is perfectly decomposable from the system. This has also been proved by Chandy, Herzog, and Woo, who showed that, in a product form solution, the factors corresponding to devices in the subnetwork can be coalesced into a single factor whose service function is obtained from an offline experiment [CHAN75a].[5] This result explains why the decomposition used at Figure 17 (to replace the central subsystem of Figure 11) introduced no new error beyond what already resulted from the homogeneity assumptions. The converse of this result is not true: a perfectly decomposable subsystem need not comprise a network of homogeneous devices.

Decomposition gives a good approximation when the number of state changes within the subsystem between interactions with the environment is reasonably large [COUR77], for then the aggregated behavior dominates the possible influence of any individual subsystem state. In the example of Figure 11, each job causes an average of $V_0 + V_1 + V_2 + V_4 = 40$ state transitions in the central subsystem; hence we could expect reasonable results from the decomposition of Figure 17 even if the central subsystem were not internally homogeneous.

The online service function may depend on the variance of the distribution of request sizes: an occasional very long job will cause a long queue to build, whereupon longer interdeparture times may be observed for longer queue lengths. By introducing the concept of *stages of service*, the effect of variance can be represented in the equivalent device. (See [BASK75, CHAN75b, GELE76a,b, KLEI75, LAZO77a,b, SEVC77, SHUM76, SHUM77].) A detailed treatment of these topics has been given recently by Chandy and Sauer [CHAN78].

[5] In fact, if $\mathbf{n}$ denotes a state of a subnetwork containing N jobs, where the $p(\mathbf{n})$ sum to $p(N)$, the output rate is

$$X_0(N) = \sum_{\mathbf{n}} p(\mathbf{n})/p(N) \sum_i q_{i0}/S_i(n_i),$$

which is completely determined by N

Applications

The major application of decomposition is simplifying problems through modularization. In his definitive treatment, Courtois has shown that significant reductions in solution times can be obtained by employing decomposition; indeed, for systems with very large state spaces, decomposition may be the only computationally feasible approach to a solution whose accuracy can be guaranteed. [COUR77].

The most important applications of decomposition have been for virtual memory systems, blocking, and other behaviors which cannot be represented directly in the queueing network model.

The difficulty in virtual memory systems is that the fixed size of real memory causes the visit ratio at the swapping device to increase with the multiprogramming level. This effect was first treated in a queueing network model by Buzen [BUZE71b, BUZE71c]. Courtois made a significant contribution by using decomposition to treat systems whose multiprogramming level varied during the observation period; he also used decomposition to construct an elegant analysis of the dynamics of thrashing [COUR75, COUR77]. Others have extended the method to study optimal multiprogrammed memory management [BRAN74, BRAN77, DENN75b, DENN76]. (For a survey, see [DENN78].)

In systems where blocking occurs, the device homogeneity assumption may be seriously violated. Blocking may occur when a load controller stops admitting new jobs to active status because memory is fully committed [BRAN74, COUR75, DENN75b, DENN76]; or when an I/O channel may be temporarily blocked by some of the devices it controls [BROW75, BROW77]; or when the geometry of a rotating drum prevents it from serving its separate sector queues simultaneously [DENN72]. In such cases, an offline experiment may be used to replace, with an equivalent device, the subsystem in which blocking occurs.

Decomposition can be applied repeatedly: a system containing devices equivalent to subsystems may be replaced by an equivalent device. [BROW75, BROW77, COUR77]. Decomposition has been used to replace a subsystem of a simulation, thereby speeding up the simulator [SCHW78].

CONCLUSIONS

Operational queueing network theory is based on the premise of testability. All the basic performance quantities (Table II)—utilizations, completion rates, mean queue sizes, mean response times, load distributions—are defined as they would be in practice from data taken over a finite period. The analyst can test whether the basic assumptions—flow balance, one-step behavior, and homogeneity—hold in any observation period.

The operational laws (Tables I and III) are identities among operational quantities. They are a consistency check—a failure to satisfy an operational law reveals an error in the data. They simplify data collection by showing alternatives for computing performance quantities.

Job flow balance implies that the throughputs everywhere in a system are determined by the throughput at any one point in the system. Since an increasing load will drive some device into saturation, this assumption allows determining asymptotes on throughput and response time; the only data needed for such a "bottleneck analysis" are the visit ratios and saturation output rates at the devices.

Job flow analysis does not account for the effects of queueing in the system at intermediate loads, which must be studied in terms of the system's state space. Each state $\mathbf{n} = (n_1, \cdots, n_K)$ represents a possible distribution of jobs among the devices, and $p(\mathbf{n})$ represents the proportion of time state $\mathbf{n}$ is occupied. The objective is to express the $p(\mathbf{n})$ directly in terms of the operational parameters of the system.

Under the additional assumptions of one-step behavior and homogeneity, we can find balance equations relating the $p(\mathbf{n})$ to the operational visit ratios and service time functions. These appear to be the weakest assumptions leading to the product form solution for $p(\mathbf{n})$. By exploiting the product form of the solution, we can devise efficient methods for calculating performance quantities without having to compute the $p(\mathbf{n})$ explicitly. Indeed, the remarkable speed with which performance quantities can be computed using queueing network formulae is an important reason that this technology is so widely used.

Most errors with these results arise from the homogeneity assumptions. Homogeneity asserts that there is no interaction between a device and the rest of the system, except for dependence on the local queue length. In a real system the service function will depend on the pattern by which the rest of the system sends requests to a device, and that pattern may depend on the form of the request size distribution of that device.

In practice, errors from these assumptions are not serious. Even when the additional assumption of homogeneous service times is used to simplify the analysis further, these models estimate utilizations, throughputs, and system response times typically to within 10%, and mean queue lengths and device response times typically to within 30% [BUZE75, GIAM76, HUGH73, LIPS77]. Refining the model of devices to make explicit the effect of the request size distribution increases the accuracy, especially in predicting queue length distributions [BASK75, CHAN75b, LAZO77a, REIS76, SEVC77]. Very little is known about response time distributions for these systems. (However, see [CHOW77, LAZO77b, WONG77].)

To use these results for performance prediction, the analyst must estimate the parameter values for the projection period; then use these estimates in the equations to calculate the estimated performance measures in the projection period. We have offered no definitive treatment of the parameter estimation problem. Nor can we: it is in the realm of inductive mathematics, whereas operational analysis is a branch of deductive mathematics. (See [GARD76].) We have illustrated in the examples the kinds of invariance assumptions analysts use to estimate parameters.

Stochastic queueing theory makes some

analysts more comfortable when estimating parameters, since the theory tells how to deduce confidence intervals to bound the uncertainty in estimates derived from data taken in a finite baseline period. However, the stochastic model employs a hidden inductive assumption: that the values of the stochastic parameters in the projection period are known functions of the corresponding values in the baseline period. In fact, there is no way to know this for sure. Thus, the stochastic analyst faces exactly the same uncertainties as the operational analyst; both must estimate unknown values for the projection period from values observed in the baseline period. Dealing with uncertainties in estimation is a very important problem, but it is beyond the pale of the deductive mathematical system in which relationships among variables are derived. (For a complete discussion of these points, see [BUZE77, BUZE78a, GARD76].)

With its weaker basis, operational queueing network theory applies to a wider class of computer systems than Markovian queueing network theory. Conversely, Markovian theory includes assumptions not present in the operational framework of this paper. Markovian queueing network theory, for example, allows deriving differential equations relating time dependent probabilities $p(\mathbf{n}, t)$ to their derivatives; in principle, we can then solve for the transient behavior of the system. As presented in this paper, operational analysis contains no concept like $p(\mathbf{n}, t)$. It gives no information about a system's transient behavior.

These limitations, however, apply only to the formulation presented in this paper. Within the basic requirement of operational testability, it is possible to make further assumptions to deal with transient behavior. Transient behavior might be modeled as job flows "diffusing" in a system [KLEI75], or as sequences of homogeneous behaviors through successively higher levels of aggregation of system states [COUR77].

The path to further knowledge awaits exploration.

ACKNOWLEDGMENTS

We are grateful to the following individuals for their patience, constructive criticisms, wisdom, and insights: G. Balbo; S. C. Bruell; J. C. Browne; K. M. Chandy; P. J. Courtois; M. A. Franklin; W. D. Frazer; E. Gelenbe; R. P. Goldberg; G. S. Graham; D. L. Iglehart; K. C. Kahn; R. M. Keller; L. Kleinrock; E. D. Lazowska; J. Leroudier; I. Mitrani; R. R. Muntz; D. Potier; M. Reiser; D. B. Rubin; A. Schroeder; H. S. Schwenk; H. D. Schwetman; K. C. Sevcik; J. Shore; A. W. C. Shum; J. W. Wong; and L. S. Wright. Special thanks go to G. S. Graham and E. D. Lazowska for carefully reading earlier versions of this manuscript.

REFERENCES

BALB77 BALBO, G.; BRUELL, S. C.; AND SCHWETMAN, H. D. "Customer classes and closed network models—a solution technique," in *Proc. IFIP Congress 77*, North-Holland Publ. Co., Amsterdam, The Netherlands, pp. 559–564.

BALB78 BALBO, G.; AND DENNING, P. J. *Approximating load distributions in time sharing systems*, Tech. Rep. CSD-TR-259, Computer Science Dept., Purdue Univ., W. Lafayette, Ind., March 1978.

BASK75 BASKETT, F.; CHANDY, K. M.; MUNTZ, R. R.; AND PALACIOS, J. "Open, closed, and mixed networks with different classes of customers," *J. ACM* **22**, 2 (April 1975), 248–260.

BOUH78 BOUHANA, J. "Operational aspects of centralized queueing networks," PhD Thesis, Computer Science Dept., Univ. Wisconsin, Madison, Jan. 1978.

BRAN74 BRANDWAJN, A. "A model of a time sharing system solved using equivalence and decomposition methods," *Acta Inf.* **4**, 1 (1974), 11–47.

BRAN77 BRANDWAJN, A.; AND MOUNIEX, B. "A study of a page-on-demand system," *Inf. Process. Lett.* **6**, 4 (Aug. 1977), 125–132.

BROW77 BROWN, R. M.; BROWNE, J. C.; AND CHANDY, K. M. "Memory management and response time," *Commun. ACM* **20**, 3 (March 1977), 153–165.

BROW75 BROWNE, J. C.; CHANDY, K. M.; BROWN, R. M.; KELLER, T. W.; TOWSLEY, D. F.; AND DISSLY, C. W. "Hierarchical techniques for the development of realistic models of complex computer systems," *Proc. IEEE* **63**, 6 (June 1975), 966–976.

BUZE71a BUZEN, J. P. "Analysis of system bottlenecks using a queueing network model," in *Proc. ACM SIGOPS Workshop System Performance Evaluation*, 1971, ACM, New York, pp. 82–103.

BUZE71b BUZEN, J. P. "Queueing network models of multiprogramming," PhD Thesis, Div. Eng. and Applied Physics, Harvard Univ., Cambridge, Mass., May 1971. (NTIS #AD 731 575, Aug. 1971.)

BUZE71c BUZEN, J. P. "Optimizing the degree of multiprogramming in demand paging systems," in *Proc. IEEE COMPCON*, 1971, IEEE, New York, pp. 139–140.

BUZE73 BUZEN, J. P. "Computational algorithms for closed queueing networks with exponential servers," *Commun. ACM* **16**, 9 (Sept. 1973), 527–531.

BUZE75 BUZEN, J. P. "Cost effective analytic tools for computer performance evalua-

tion," in *Proc. IEEE COMPCON*, 1975, IEEE, New York, pp. 293-296.

BUZE76a BUZEN, J. P. "Operational analysis: the key to the new generation of performance prediction tools," in *Proc. IEEE COMPCON*, 1976, IEEE, New York.

BUZE76b BUZEN, J. P. "Fundamental operational laws of computer system performance," *Acta Inf.* **7**, 2 (1976), 167-182.

BUZE77 BUZEN, J. P. "Principles of computer performance modeling and prediction," in *Infotech state of the art report on performance modeling and prediction*, Infotech Int. Ltd., Maidenhead, UK, 1977, pp. 3-18.

BUZE78a BUZEN, J. P. "Operational analysis: an alternative to stochastic modeling," in *Proc. Int. Conf Performance Computer Installations*, 1978, North-Holland Publ. Co., Amsterdam, The Netherlands, pp. 175-194.

BUZE78b BUZEN, J. P., et al. "BEST/1—design of a tool for computer system capacity planning," in *Proc. 1978 AFIPS National Computer Conf.*, Vol. 47, AFIPS Press, Montvale, N.J., pp. 447-455.

CHAN75a CHANDY, K. M., HERZOG, U.; AND WOO, L. "Parametric analysis of queueing networks," *IBM J. Res. Dev.* **19**, 1 (Jan. 1975), 36-42.

CHAN75b CHANDY, K. M.; HERZOG, U.; AND WOO, L. "Approximate analysis of general queueing networks," *IBM J Res Dev.* **19**, 1 (Jan. 1975), 43-49.

CHAN78 CHANDY, K. M.; AND SAUER, C. H "Approximate methods for analyzing queueing network models of computer systems," *Comput. Surv.* **10**, 3 (Sept. 1978), 281-317

CHAN74 CHANG, A.; AND LAVENBERG, S. "Work rates in closed queueing networks with general independent servers," *Oper. Res.* **22**, 4 (1974), 838-847.

CHOW77 CHOW, W. *The cycle time distribution of exponential central server queues*, IBM Res. Rep. RC 6765, 1977.

COFF73 COFFMAN, E G., JR., AND DENNING, P. J. *Operating systems theory*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

COUR75 COURTOIS, P. J. "Decomposability, instabilities, and saturation in multiprogrammed systems," *Commun. ACM* **18**, 7 (July 1975), 371-377.

COUR77 COURTOIS, P. J. *Decomposability. queueing and computer system applications*, Academic Press, New York, 1977.

DENN72 DENNING, P. J. "A note on paging drum efficiency," *Comput. Surv.* **4**, 1 (March 1972), 1-3.

DENN75a DENNING, P. J.; AND KAHN, K. C. *Some distribution-free properties of throughput and response time*, Tech. Rep. CSD-TR-159, Computer Science Dept., Purdue Univ., W. Lafayette, Ind., May 1975.

DENN75b DENNING, P. J.; AND GRAHAM, G S. "Multiprogrammed memory management," *Proc IEEE* **63**, 6 (June 1975), 924-939

DENN76 DENNING, P. J.; KAHN, K. C.; LEROUDIER, J ; POTIER, D., AND SURI, R "Optimal multiprogramming," *Acta Inf.* **7**, 2 (1976), 197-216.

DENN77a DENNING, P. J.; AND BUZEN, J. P. "Operational analysis of queueing networks," in *Proc. Third Int. Symp. Computer Performance Modeling, Measurement, and Evaluation*, 1977, North-Holland Publ. Co., Amsterdam, The Netherlands.

DENN77b DENNING, P J.; AND BUZEN, J. P. "An operational overview of queueing networks," in *Infotech state of the art report on performance modeling and prediction*, Infotech Int. Ltd., Maidenhead, UK, 1977, pp. 75-108.

DENN78 DENNING, P. J. "Optimal multiprogrammed memory management," in *Current trends in programming methodology III*, K. M. Chandy and R. Yeh (Eds.), Prentice-Hall, Englewood Cliffs, N.J., 1978, pp. 298-322.

GARD76 GARDNER, M. "Mathematical games: On the fabric of inductive logic, and some probability paradoxes," *Sci. Am* **234**, 3 (March 1976), 119-122.

GELE76a GELENBE, E.; AND MUNTZ, R. R. "Probability models of computer systems I. exact results," *Acta Inf.* **7**, 1 (May 1976), 35-60.

GELE76b GELENBE, E., AND PUJOLLE, G. "The behavior of a single queue in a general queueing network," *Acta Inf.* **7**, 2 (1976), 123-136.

GIAM76 GIAMMO, T. "Validation of a computer performance model of the exponential queueing network family," *Acta Inf.* **7**, 2 (1976), 137-152.

GORD67 GORDON, W. J.; AND NEWELL, G. F. "Closed queueing systems with exponential servers," *Oper. Res.* **15** (1967), 254-265.

HERZ75 HERZOG, U.; WOO, L.; AND CHANDY, K. M. "Solution of queueing problems by a recursive technique," *IBM J. Res. Dev.* **19**, 3 (May 1975), 295-300.

HUGH73 HUGHES, P. H.; AND MOE, G. "A structural approach to computer performance analysis," in *Proc 1973 AFIPS National Computer Conf*, Vol. 42, AFIPS Press, Montvale, N.J., pp. 109-119.

IGLE78 IGLEHART, D. L. "The regenerative method for simulation analysis," in *Current trends in programming methodology III*, K. M. Chandy and R. Yeh (Eds.), Prentice-Hall Englewood, Cliffs, N.J., 1978, pp. 52-71.

JACK57 JACKSON, J. R. "Networks of waiting lines," *Oper. Res.* **5** (1957), 518-521.

JACK63 JACKSON, J. R. "Jobshop like queueing systems," *Manage. Sci.* **10** (1963), 131-142.

KLEI68 KLEINROCK, L. "Certain analytic results for time shared processors," in *Proc. IFIP Congress 1968*, North-Holland Publ. Co., Amsterdam, The Netherlands, pp. 838-845.

KLEI75 KLEINROCK, L. *Queueing systems I*, John Wiley, New York, 1975.

KLEI76 KLEINROCK, L. *Queueing systems II*, John Wiley, New York, 1976.

LAZO77a LAZOWSKA, E. D. "The use of percentiles in modeling CPU service time distributions," in *Proc. Int. Symp. Computer Performance Modeling, Measurement, and Evaluation*, 1977, North-Hol-

land Publ. Co., Amsterdam, The Netherlands, pp. 53–66.

LAZO77b LAZOWSKA, E. D. "Characterizing service time and response time distributions in queueing network models of computer systems," PhD Thesis, Univ. Toronto, Toronto, Ont., Canada. (Computer Systems Research Group, Tech. Rep. CSRG-85, Oct. 1977.)

LIPS77 LIPSKY, L.; AND CHURCH, J. D. "Applications of a queueing network model for a computer system," *Comput. Surv.* **9**, 3 (Sept. 1977), 205–222.

MOOR71 MOORE, C. G., III *Network models for large-scale time sharing systems*, Tech. Rep. 71-1, Dept. Industrial Eng., Univ. Michigan, Ann Arbor, April 1971, PhD Thesis.

MUNT74 MUNTZ, R. R.; AND WONG, J. W. "Asymptotic properties of closed queueing network models," in *Proc. 8th Princeton Conf. Information Sciences and Systems*, 1974, Dept. EECS, Princeton Univ., Princeton, N.J., pp. 348–352.

MUNT75 MUNTZ, R. R. "Analytic modeling of interactive systems," *Proc IEEE* **63**, 6 (June 1975), 946–953.

REIS75 REISER, M.; AND KOBAYSHI, H. "Queueing networks with multiple closed chains: theory and computation algorithms," *IBM J. Res. Dev.* **19** (May 1975), 283–294.

REIS78 REISER, M.; AND SAUER, C. H. "Queueing network models: methods of solution and their program implementations," in *Current trends in programming methodology III*, K. M. Chandy and R. Yeh (Eds.), Prentice-Hall, Englewood Cliffs, N.J., 1978, pp. 115–167.

ROSE78 ROSE, C. A. "Measurement procedure for queueing network models of computer systems," *Comput Surv.* **10**, 3 (Sept. 1978), 263–280.

SCHE67 SCHERR, A. L *An analysis of time shared computer systems*, MIT Press, Cambridge, Mass., 1967.

SCHW78 SCHWETMAN, H. D. "Hybrid simulation models of computer systems," *Commun. ACM* **21** (1978), to appear.

SEVC77 SEVCIK, K.; LEVY, A. I., TRIPATHI, S. K.; AND ZAHORJAN, J. L. "Improving approximations of aggregated queueing network subsystems," in *Proc. Int. Symp. Computer Performance Modeling, Measurement, and Evaluation*, 1977, North-Holland Publ. Co., Amsterdam, The Netherlands, pp. 1–22.

SHUM76 SHUM, A. W. C. "Queueing models for computer systems with general service time distributions," PhD Thesis, Div. Eng. and Applied Physics, Harvard Univ., Cambridge, Mass., Dec. 1976.

SHUM77 SHUM, A. W. C.; AND BUZEN, J. P. "The EPF technique: a method for obtaining approximate solutions to closed queueing networks with general service times," in *Proc. Int. Symp. Computer Performance Modeling, Measurement, and Evaluation*, 1977, North-Holland Publ. Co., Amsterdam, The Netherlands, pp. 201–222.

WILL76 WILLIAMS, A. C.; AND BHANDIWAD, R. A. "A generating function approach to queueing network analysis of multiprogrammed computers," *Networks* **6**, 1 (1976), 1–22.

WONG77 WONG, J. W. "Distribution of end-to-end delay in message-switched networks," *Comput. Networks* **2**, 1 (Feb. 1978), 44–49.

RECEIVED AUGUST 30, 1977; FINAL REVISION ACCEPTED MAY 16, 1978.