Raj Jain, "The Art of Computer Systems Performance, Analysis,"
John Wiley and Sons, 1991, pp. 547-569.

# CHAPTER 32

# QUEUEING NETWORKS

*A problem well stated is a problem half solved.*
—Charles F. Kettering

All the systems analyzed so far had only one queue. There are a number of systems that consist of several queues. A job may receive service at one or more queues before exiting from the system. Such systems are modeled by queueing networks. In general, a model in which jobs departing from one queue arrive at another queue (or possibly the same queue) is called a **queueing network**. This chapter presents several basic concepts about queueing networks. The techniques to solve the networks are described in Chapters 33 to 36.

## 32.1 OPEN AND CLOSED QUEUEING NETWORKS

Unlike single queues, there is no easy notation for specifying the type of queueing network. The simplest way to classify a queueing network is either open or closed. An **open queueing network** has external arrivals and departures, as shown in Figure 32.1. The jobs enter the system at "In" and exit at "Out." The number of jobs in the system varies with time. In analyzing an open system, we assume that the throughput is known (to be equal to the arrival rate), and the goal is to characterize the distribution of number of jobs in the system. A **closed queueing network** has no external arrivals or departures. As shown in Figure 32.2, the jobs in the system keep circulating from one queue to the next. The total number of jobs in the system is constant. It is possible to view a closed system as a system where the Out is connected back

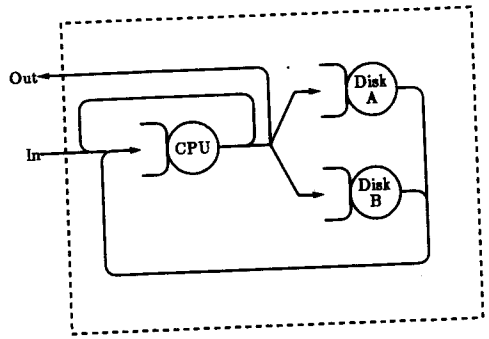


**FIGURE 32.1** An open queueing network has external arrivals and departures.

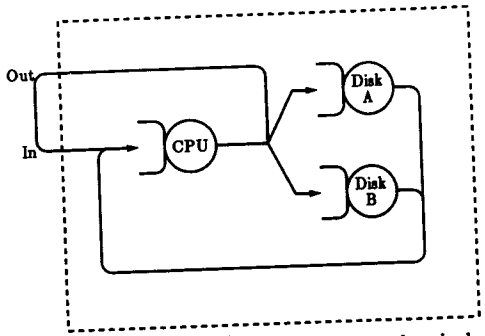


**FIGURE 32.2** A closed queueing network has no external arrivals and departures.

to the In. The jobs exiting the system immediately reenter the system. The flow of jobs in the Out-to-In link defines the throughput of the closed system. In analyzing a closed system, we assume that the number of jobs is given, and we attempt to determine the throughput (or the job completion rate).

It is also possible to have **mixed queueing networks** that are open for some workloads and closed for others. Figure 32.3 shows an example of a mixed system with two *classes* of jobs. The system is closed for interactive jobs and is open for batch jobs. The term *class* refers to types of jobs. All jobs of a single class have the same service demands and transition probabilities. Within each class, the jobs are indistinguishable. The discussion here is limited to one class of jobs. Multiclass models are beyond the scope of the introductory treatment in this book.

## 32.2 PRODUCT FORM NETWORKS

The simplest queueing network is a series of $M$ single-server queues with exponential service time and Poisson arrivals, as shown in Figure 32.4. The jobs



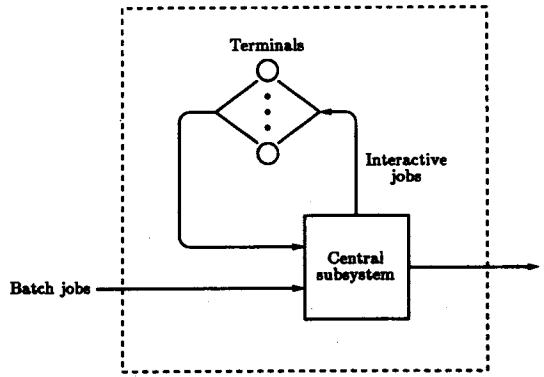


**FIGURE 32.3** A mixed queueing network is open for some classes and closed for others.

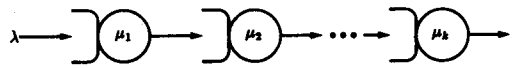


**FIGURE 32.4** A simple queueing network consisting of $k$ M/M/1 queues in series.

leaving a queue immediately join the next queue. It can be shown that each individual queue in this series can be analyzed independently of other queues. Each queue has an arrival as well as a departure rate of $\lambda$. If $\mu_i$ is the service rate for the $i$th server,

$$\text{Utilization of the } i\text{th server } \rho_i = \lambda/\mu_i$$

$$\text{Probability of } n_i \text{ jobs in the } i\text{th queue} = (1-\rho_i)\rho_i^{n_i}$$

The joint probability of queue lengths of $M$ queues can be computed simply by multiplying individual probabilities; for example,

$$\begin{aligned} P(n_1, n_2, n_3, \ldots, n_M) &= (1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2}(1-\rho_3)\rho_3^{n_3}\cdots(1-\rho_M)\rho_M^{n_M} \\ &= p_1(n_1)p_2(n_2)p_3(n_3)\cdots p_M(n_M) \end{aligned} \tag{32.1}$$

This queueing network is therefore a **product form network**. In general, the term applies to any queueing network in which the expression for the equilibrium probability has the following form:

$$P(n_1, n_2, \ldots, n_M) = \frac{1}{G(N)} \prod_{i=1}^{M} f_i(n_i)$$

When $f_i(n_i)$ is some function of the number of jobs at the $i$th facility, $G(N)$ is a normalizing constant and is a function of the total number of jobs in the system.

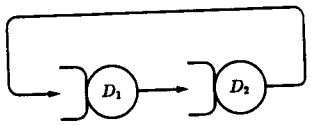


FIGURE 32.5 Closed network of two queues.

**Example 32.1** Consider a closed system with two queues and $N$ jobs circulating among the queues, as shown in Figure 32.5. Both servers have an exponentially distributed service time. The mean service times are 2 and 3, respectively.

Using the method discussed later in Chapter 35, the probability of having $n_1$ jobs in the first queue and $n_2 = N - n_1$ jobs in the second queue can be shown to be

$$P(n_1, n_2) = \frac{1}{3^{N+1} - 2^{N+1}} (2^{n_1} \times 3^{n_2})$$

In this case, the normalizing constant $G(N)$ is $3^{N+1} - 2^{N+1}$. The state probabilities are products of functions of the number of jobs in the queues. Thus, this is a product form network. □

Product form networks are easier to analyze than nonproduct form networks. The set of networks that have a product form solution is continuously being extended by the researchers. First among these was Jackson (1963), who showed that the above method of computing joint probability is valid for any arbitrary open network of $m$-server queues with exponentially distributed service times (see Figure 32.6). In particular, if all queues are single-server queues, the queue length distribution is given by Equation (32.1). However, it is not correct to assume that each queue becomes an independent M/M/1 queue with a Poisson arrival process. In general, the internal flow in such networks is not Poisson. Particularly, if there is any feedback in the network, so that jobs can return to previously visited service centers, the internal flows are not Poisson. It is surprising that even though the flows are not Poisson,

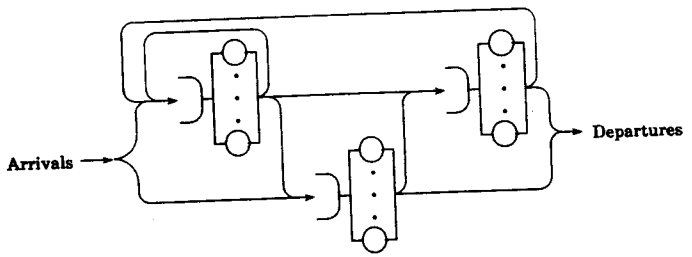


FIGURE 32.6 General open network of queues.



the queues are separable and can be analyzed as if they were independent M/M/$m$ queues.

Jackson's results were later extended to closed networks by Gordon and Newell (1967). They showed that any arbitrary closed networks of $m$-server queues with exponentially distributed service times also have a product form solution.

Baskett, Chandy, Muntz, and Palacios (1975) showed that product form solutions exist for an even broader class of networks. This class consists of networks satisfying the following criteria:

1. *Service Disciplines*: All service centers have one of the following four types of service disciplines: First Come, First Served (FCFS), Processor Sharing (PS), Infinite Servers (ISs or delay centers), and Last Come, First Served Preemptive Resume (LCFS-PR).
2. *Job Classes*: The jobs belong to a single class while awaiting or receiving service at a service center but may change classes and service centers according to fixed probabilities at the completion of a service request.
3. *Service Time Distributions*: At FCFS service centers, the service time distributions must be identical and exponential for all classes of jobs. At other service centers, where the service times should have probability distributions with rational Laplace transforms, different classes of jobs may have different distributions.
4. *State-dependent Service*: The service time at a FCFS service center can depend only on the total queue length of the center. The service time for a class at PS, LCFS-PR, and IS centers can also depend on the queue length for that class, but not on the queue length of other classes. Moreover, the overall service rate of a subnetwork can depend on the total number of jobs in the subnetwork.
5. *Arrival Processes*: In open networks, the time between successive arrivals of a class should be exponentially distributed. No bulk arrivals are permitted. The arrival rates may be state dependent. A network may be open with respect to some classes of jobs and closed with respect to other classes of jobs.

Networks satisfying these criteria are referred to as **BCMP networks** after the authors of the criteria.

Denning and Buzen (1978) further extended the class of product form networks to non-Markovian networks with the following conditions:

1. *Job Flow Balance*: For each class, the number of arrivals to a device must equal the number of departures from the device.
2. *One-Step Behavior*: A state change can result only from single jobs entering the system, moving between pairs of devices in the system, or exiting from the system. This assumption asserts that simultaneous job moves will not be observed.

3. *Device Homogeneity*: A device's service rate for a particular class does not depend on the state of the system in any way except for the total device queue length and the designated class's queue length. This assumption implies the following:

   (a) *Single-Resource Possession*: A job may not be present (waiting for service or receiving service) at two or more devices at the same time.

   (b) *No Blocking*: A device renders service whenever jobs are present; its ability to render service is not controlled by any other device.

   (c) *Independent Job Behavior*: Interaction among jobs is limited to queueing for physical devices; for example, there should not be any synchronization requirements.

   (d) *Local Information*: A device's service rate depends only on local queue length and not on the state of the rest of the system.

   (e) *Fair Service*: If service rates differ by class, the service rate for a class depends only on the queue length of that class at the device and not on the queue lengths of other classes. This means that the servers do not discriminate against jobs in a class depending on the queue lengths of other classes.

   (f) *Routing Homogeneity*: The job routing should be state independent.

   In the last condition, the term *routing* is used to denote a job's path in the network. The routing homogeneity condition implies that the probability of a job going from one device to another device does not depend upon the number of jobs at various devices.

The job flow balance assumption holds only in some observation periods. However, it is a good approximation for long observation intervals since the ratio of unfinished jobs to completed jobs is small.

## 32.3 QUEUEING NETWORK MODELS OF COMPUTER SYSTEMS

Two of the earliest queueing models of computer systems are the **machine repairman model** and the **central server model** shown in Figures 32.7 and 32.8, respectively. The machine repairman model, as the name implies, was originally developed for modeling machine repair shops. It has a number of working machines and a repair facility with one or more servers (repairmen). Whenever a machine breaks down, it is put in the queue for repair and serviced as soon as a repairman is available. Scherr (1967) used this model to represent a timesharing system with $n$ terminals. Users sitting at the terminals generate requests (jobs) that are serviced by the system, which serves as



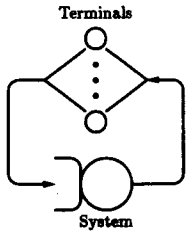


**FIGURE 32.7** Machine repairman model of a computer system.

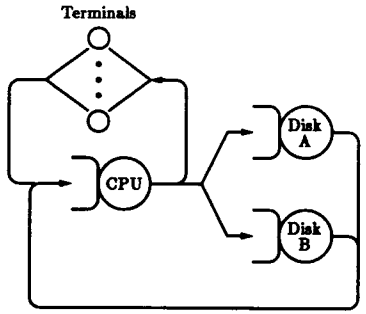


**FIGURE 32.8** Central server model of a timesharing system.

a repairman. After a job is done, it waits at the user terminal for a random "think-time" interval before cycling again.

The central server model shown in Figure 32.8 was introduced by Buzen (1973). The CPU in the model is the "central server" that schedules visits to other devices. After service at the I/O devices, the jobs return to the CPU for further processing and leave it when the next I/O is encountered or when the job is completed.

In computer systems modeling, we encounter three kinds of devices. Most devices have a single server whose service time does not depend upon the number of jobs in the device. Such devices are called **fixed-capacity service centers**. For example, the CPU in a system may be modeled as a fixed-capacity service center. Then there are devices that have no queueing, and jobs spend the same amount of time in the device regardless of the number of jobs in it. Such devices can be modeled as a center with infinite servers and are called **delay centers** or **IS** (infinite server). A group of dedicated terminals is usually modeled as a delay center. Finally, the remaining devices are called **load-dependent service centers** since their service rates may depend upon the load or the number of jobs in the device. A M/M/$m$ queue (with $m \geq 2$) is an example of a load-dependent service center. Its total service rate increases as more and more servers are used. A group of parallel links between two

nodes in a computer network is an example of a load-dependent service center. Only fixed-capacity centers and delay centers are considered in this chapter. Load-dependent centers are considered in Section 36.1. Unless specified otherwise, it is assumed that the service times for all servers are exponentially distributed.

In the next few chapters, we will describe a number of techniques to solve these queueing network models. The techniques are presented in the order of their complexity. The simplest technique is using operational analysis, and is presented in Chapter 33.

### EXERCISE

32.1 Which product form condition is violated, if any, by the following system behaviors?

a. Jobs are lost if there are no buffers.

b. One job class has a priority over the other class.

c. Processes wait for fork and join primitives to complete.

d. A job intelligently chooses its path depending upon the congestion at the devices.

e. The system uses a back-pressure mechanism to avoid job loss when there are no buffer.

f. A job may execute while its I/O is proceeding.

g. The jobs are serviced in batches.

# CHAPTER 33

# OPERATIONAL LAWS

*All professional men are handicapped by not being allowed to ignore things which are useless.*
—Johann Wolfgang von Goethe

A large number of day-to-day problems in computer systems performance analysis can be solved by using some simple relationships that do not require any assumptions about the distribution of service times or interarrival times. Several such relationships called **operational laws** were identified originally by Buzen (1976) and later extended by Denning and Buzen (1978).

The word *operational* means directly measured. Thus, *operationally testable* assumptions are the assumptions that can be verified by measurements. For example, it is easy to verify whether the assumption of number of arrivals being equal to the number of completions holds for a particular system. Therefore, this assumption, commonly called the **job flow balance** assumption, is operationally testable. On the other hand, one can never establish through measurement that a set of observed service times is or is not a sequence of independent random variables. Thus, the assumption of independence, although commonly used in stochastic modeling of queues, is not operationally testable.

The *operational quantities* are the quantities that can be directly measured during a finite observation period. For example, consider the black-box view of a device $i$ shown earlier in Figure 30.3. If we observe the device for a finite time $T$, we can measure the number of arrivals $A_i$, the number of completions $C_i$, and the busy time $B_i$ during this period. All of these are operational

quantities. From these we can further derive the following operational quantities:

$$\text{Arrival rate } \lambda_i = \frac{\text{number of arrivals}}{\text{time}} = \frac{A_i}{T}$$

$$\text{Throughput } X_i = \frac{\text{number of completions}}{\text{time}} = \frac{C_i}{T}$$

$$\text{Utilization } U_i = \frac{\text{busy time}}{\text{total time}} = \frac{B_i}{T}$$

$$\text{Mean service time } S_i = \frac{\text{total time served}}{\text{number served}} = \frac{B_i}{C_i}$$

Notice that these operational quantities are *variables* that can change from one observation period to the next, but there are certain relationships that hold in every observation period. Such relationships are called **operational laws.** Several such laws are presented later in this chapter.

## 33.1 UTILIZATION LAW

Given the number of completions $C_i$ and the busy time $B_i$ of a device $i$ during an observation period $T$, the following relationship holds among these variables:

$$U_i = \frac{B_i}{T} = \frac{C_i}{T} \times \frac{B_i}{C_i}$$

or

$$U_i = X_i S_i$$

This relationship is called the **utilization law.**

The utilization law, as well as other operational laws, is similar to the elementary laws of motion. For example, if a body starts at rest and accelerates at a constant acceleration $a$ for an interval of time $t$, the distance traveled $d$ is given by

$$d = \tfrac{1}{2}at^2$$

Notice that distance $d$, acceleration $a$, and time $t$ are operational quantities. There is no need to consider them as expected values of random variables or to assume a probability distribution for them.

**Example 33.1** Consider the network gateway problem of Example 31.1 again. The packets arrive at a rate of 125 packets per second (pps) and the gateway takes an average of two milliseconds to forward them.

$$\text{Throughput } X_i = \text{exit rate} = \text{arrival rate} = 125 \text{ pps}$$

$$\text{Service time } S_i = 0.002 \text{ second}$$

$$\text{Utilization } U_i = X_i S_i = 125 \times 0.002 = 0.25 = 25\%$$



Although this utilization is the same as that computed in Example 31.1, the key point to note is that while the result of Example 31.1 required assuming the interarrival times and service times to be IID random variables with exponential distribution, in this section we have made no such assumption, and the result is valid for any arrival or service process. □

## 33.2 FORCED FLOW LAW

The forced flow law relates the system throughput to individual device throughputs. In an open model, the number of jobs leaving the system per unit time defines the system throughput. In a closed model, no job actually leaves the system. However, traversing the outside link (the link connecting "Out" to "In" in Figure 32.2) is equivalent to leaving the system and immediately reentering it, and the system throughput is defined as the number of jobs traversing this link per unit time.

If our observation period $T$ is such that the number of job arrivals at each device is the same as the number of job completions, that is,

$$A_i = C_i$$

we can say that the device satisfies the assumption of job flow balance. If the observation period $T$ is long, the difference $A_i - C_i$ is usually small compared to $C_i$. It will be exact if the initial queue length at each device is the same as the final queue length.

Suppose each job makes $V_i$ requests for the $i$th device in the system, as shown in Figure 33.1. If the job flow is balanced, the number of jobs $C_0$ traversing the outside link and the number of jobs $C_i$ visiting the $i$th device are related by

$$C_i = C_0 V_i \quad \text{or} \quad V_i = \frac{C_i}{C_0}$$

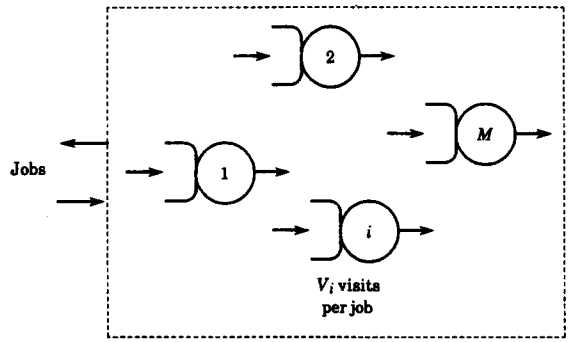


**FIGURE 33.1** Internal view of the system.

Thus, the variable $V_i$ is the ratio of visits to the $i$th device and the outside link. It is therefore called the **visit ratio.** The system throughput during the observation period is

$$\text{System throughput } X = \frac{\text{jobs completed}}{\text{total time}} = \frac{C_0}{T}$$

The throughput of the $i$th device and the system throughput are therefore related as follows:

$$\text{Device throughput } X_i = \frac{C_i}{T} = \frac{C_i}{C_0} \times \frac{C_0}{T}$$

In other words,

$$X_i = XV_i \tag{33.1}$$

This is the **forced flow law.** It applies whenever the job flow balance assumption is true.

It is clear that if the job flow balance is not true, then either there are some jobs that arrived during the observation period but did not leave or there were some jobs already in the system at the beginning of the observation period. In either case, it is possible that these jobs have not yet made all $V_i$ visits to the $i$th device and $C_i \neq C_0V_i$.

Combining the forced flow law and the utilization law, we get

$$\begin{aligned}\text{Utilization of the } i\text{th device } U_i &= X_iS_i \\ &= XV_iS_i\end{aligned}$$

or

$$U_i = XD_i \tag{33.2}$$

Here $D_i = V_iS_i$ is the total service demand on the device for all visits of a job. Equation (33.2) states that the utilizations of various devices in the system are proportional to total demand $D_i$ per job on the device. Hence, the device with the highest $D_i$ has the highest utilization and is the **bottleneck device.** This result will be used later in Section 33.6.

**Example 33.2** In a timesharing system, accounting log data produced the following profile for user programs. Each program requires 5 seconds of CPU time and makes 80 I/O requests to disk A and 100 I/O requests to disk B. The average think time of the users was 18 seconds. From the device specifications, it was determined that disk A takes 50 milliseconds to satisfy an I/O request and disk B takes 30 milliseconds per request. With 17 active terminals, disk A throughput was observed to be 15.70 I/O requests per second. We want to find the system throughput and device utilizations.

This system can be represented by the queueing network model shown earlier in Figure 32.8. Symbolically, we are given the following: $D_{CPU} = 5$ seconds, $V_A = 80$, $V_B = 100$, $Z = 18$ seconds, $S_A = 0.050$ second, $S_B = 0.030$ second, $N = 17$, and $X_A = 15.70$ jobs/second.



Since the jobs must visit the CPU before going to the disks or terminals, the CPU visit ratio is

$$V_{CPU} = V_A + V_B + 1 = 181$$

The first step in operational analysis generally is to determine total service demands $D_i$ for all devices. In this case,

$$\begin{aligned}D_{CPU} &= 5 \text{ seconds} \\ D_A &= S_AV_A = 0.050 \times 80 = 4 \text{ seconds} \\ D_B &= S_BV_B = 0.030 \times 100 = 3 \text{ seconds}\end{aligned}$$

Using the forced flow law, the throughputs are

$$\begin{aligned}X &= \frac{X_A}{V_A} = \frac{15.70}{80} = 0.1963 \text{ job/second} \\ X_{CPU} &= XV_{CPU} = 0.1963 \times 181 = 35.48 \text{ requests/second} \\ X_B &= XV_B = 0.1963 \times 100 = 19.6 \text{ requests/second}\end{aligned}$$

Using the utilization law, the device utilizations are

$$\begin{aligned}U_{CPU} &= XD_{CPU} = 0.1963 \times 5 = 98\% \\ U_A &= XD_A = 0.1963 \times 4 = 78.4\% \\ U_B &= XD_B = 0.1963 \times 3 = 58.8\%\end{aligned}$$ □

Visit ratios are one way of specifying the routing of jobs in a queueing network. Another way is to specify the **transition probabilities** $p_{ij}$ of a job moving to the $j$th queue after service completion at the $i$th queue. The visit ratios and transition probabilities are equivalent in the sense that given one we can always find the other. In a system with job flow balance,

$$C_j = \sum_{i=0}^{M} C_ip_{ij}$$

Here, subscript 0 is used to denote visits to the outside link. Thus, $p_{i0}$ is the probability of a job exiting from the system after completion of service at the $i$th device. Dividing both sides of the equation by $C_0$, we get

$$V_j = \sum_{i=0}^{M} V_ip_{ij} \tag{33.3}$$

Since each visit to the outside link is defined as the completion of the job, we have

$$V_0 = 1 \tag{33.4}$$

Equations (33.3) and (33.4) are called **visit ratio equations** and can be used to get visit ratios from transition probabilities. A unique solution is always possible provided the network is operationally connected, that is, each device in the system is visited at least once by each job.

In central server models, after completion of service at every queue, the jobs always move back to the CPU queue:

$$p_{i1} = 1 \qquad \forall i \neq 1$$
$$p_{ij} = 0 \qquad \forall i, j \neq 1$$

These probabilities apply to the exit and entrance from the system ($i = 0$) also. Therefore, the visit ratio equations become

$$1 = V_1 p_{10}$$
$$V_1 = 1 + V_2 + V_3 + \cdots + V_M$$
$$V_j = V_1 p_{1j} = \frac{p_{1j}}{p_{10}}, \qquad j = 2,3,\ldots,M$$

Thus, we can find the visit ratios by dividing the probability $p_{1j}$ of moving to the $j$th queue from the CPU by the exit probability $p_{10}$.

**Example 33.3** Consider the queueing network of Figure 32.8. The visit ratios given in Example 33.2 are $V_A = 80$, $V_B = 100$, and $V_{CPU} = 181$.

It is easy to see in this case that after completion of service at the CPU, the probabilities of the job moving to disk A, disk B, or the terminals are $\frac{80}{181}$, $\frac{100}{181}$, and $\frac{1}{181}$, respectively. Thus, the transition probabilities are 0.4420, 0.5525, and 0.005525.

Given the transition probabilities, we can find the visit ratios by dividing these probabilities by the exit probability (0.005525):

$$V_A = \frac{0.4420}{0.005525} = 80$$
$$V_B = \frac{0.5525}{0.005525} = 100$$
$$V_{CPU} = 1 + V_A + V_B = 1 + 80 + 100 = 181$$

□

## 33.3 LITTLE'S LAW

Little's law is also an operational law. In its derivation in Section 30.3, we used only operational (measurable) quantities. The only assumption required was that the number of arrivals is equal to the number of completions. This is the operationally testable assumption of job flow balance.



We can apply Little's law to relate queue length $Q_i$ and response time $R_i$ at the $i$th device:

Mean number in device = arrival rate × mean time in device

$$Q_i = \lambda_i R_i$$

If the job flow is balanced, the arrival rate is equal to the throughput, and we can write

$$Q_i = X_i R_i \tag{33.5}$$

It is this version of Little's law that we will use frequently hereafter. Notice that the queue length $Q_i$ is synonymous with the number of jobs at the $i$th device. It includes not only the jobs waiting in the $i$th queue but also those getting service at the device.

**Example 33.4** The average queue length in the computer system of Example 33.2 was observed to be 8.88, 3.19, and 1.40 jobs at the CPU, disk A, and disk B, respectively. What were the response times of these devices?

In Example 33.2, the device throughputs were determined to be

$$X_{CPU} = 35.48, \qquad X_A = 15.70, \qquad X_B = 19.6$$

The new information given in this example is

$$Q_{CPU} = 8.88, \qquad Q_A = 3.19, \qquad Q_B = 1.40$$

Using Little's law, the device response times are

$$R_{CPU} = Q_{CPU}/X_{CPU} = 8.88/35.48 = 0.250 \text{ second}$$
$$R_A = Q_A/X_A = 3.19/15.70 = 0.203 \text{ second}$$
$$R_B = Q_B/X_B = 1.40/19.6 = 0.071 \text{ second}$$

□

## 33.4 GENERAL RESPONSE TIME LAW

All timesharing systems can be divided into two subsystems: the terminal subsystem and the central subsystem consisting of the remaining devices including the CPU, as shown in Figure 33.2. There is one terminal per user and the rest of the system is shared by all users.

It is interesting to note that Little's law can be applied to any part of the system. The only requirement is that the job flow in that part be balanced. In particular, it can be applied to the central subsystem, which gives

$$Q = XR$$

Here, $Q$ is the total number of jobs in the system, $R$ is the system response time, and $X$ is the system throughput. Given individual queue lengths $Q_i$ at the devices, we can compute $Q$:

$$Q = Q_1 + Q_2 + \cdots + Q_M$$

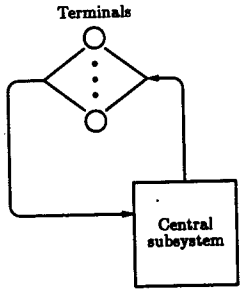


**FIGURE 33.2** Two components of a timesharing system: terminals and the central subsystem.

Substituting for $Q_i$ from Equation (33.5) yields

$$XR = X_1R_1 + X_2R_2 + \cdots + X_MR_M$$

Dividing both sides of this equation by $X$ and using the forced flow law, we get

$$R = V_1R_1 + V_2R_2 + \cdots + V_MR_M$$

or

$$R = \sum_{i=1}^{M} R_iV_i \tag{33.6}$$

This is called the **general response time law.** It is possible to show that this law holds even if the job flow is not balanced. Intuitively, the law states that the total time spent by a job at a server is the product of time per visit and the number of visits to the server; and the total time in the system is equal to the sum of total times at various servers.

**Example 33.5** Let us compute the response time for the timesharing system of Examples 33.2 and 33.4. For this system

$$V_{CPU} = 181, \quad V_A = 80, \quad V_B = 100$$

$$R_{CPU} = 0.250, \quad R_A = 0.203, \quad R_B = 0.071$$

The system response time is

$$R = R_{CPU}V_{CPU} + R_AV_A + R_BV_B$$

$$= 0.250 \times 181 + 0.203 \times 80 + 0.071 \times 100 = 68.6$$

The system response time is 68.6 seconds. □



## 33.5 INTERACTIVE RESPONSE TIME LAW

In an interactive system, the users generate requests that are serviced by the central subsystem and responses come back to the terminal. After a think time $Z$, the users submit the next request. If the system response time is $R$, the total cycle time of requests is $R + Z$. Each user generates about $T/(R + Z)$ requests in time period $T$. If there are $N$ users,

$$\text{System throughput } X = \frac{\text{total number of requests}}{\text{total time}}$$

$$= \frac{N[T/(R+Z)]}{T}$$

$$= \frac{N}{R+Z}$$

or

$$R = (N/X) - Z \tag{33.7}$$

This is the **interactive response time law.**

**Example 33.6** For the timesharing system of Example 33.2, we can compute the response time using the interactive response time law as follows:

$$X = 0.1963, \quad N = 17, \quad Z = 18$$

Therefore,

$$R = \frac{N}{X} - Z = \frac{17}{0.1963} - 18 = 86.6 - 18 = 68.6 \text{ seconds}$$

This is the same as that obtained earlier in Example 33.5. □

## 33.6 BOTTLENECK ANALYSIS

One consequence of the forced flow law is that the device utilizations are proportional to their respective total service demands:

$$U_i \propto D_i$$

The device with the highest total service demand $D_i$ has the highest utilization[1] and is called the **bottleneck device.** This device is the key limiting factor in achieving higher throughput. Improving this device will provide the highest payoff in terms of system throughput. Improving other devices will have little effect on the system performance. Therefore, *identifying the bottleneck device should be the first step in any performance improvement project.*

[1]Delay centers can have utilizations more than one without any stability problems. Therefore, delay centers cannot be a bottleneck device. Only queueing centers should be considered in finding the bottleneck or computing $D_{max}$.

Suppose we find that the device $b$ is the bottleneck. This implies that $D_b = D_{max}$ is the highest among $D_1, D_2, \ldots, D_M$. Then the throughput and response times of the system are bound as follows:

$$X(N) \le \min\left\{\frac{1}{D_{max}}, \frac{N}{D+Z}\right\} \tag{33.8}$$

and

$$R(N) \ge \max\{D, ND_{max} - Z\} \tag{33.9}$$

Here, $D = \sum D_i$ is the sum of total service demands on all devices except terminals. Equations (33.8) and (33.9) are known as **asymptotic bounds**. The proof follows.

**Proof 33.1** The asymptotic bounds are based on the following observations:

1. The utilization of any device cannot exceed 1. This puts a limit on the maximum obtainable throughput.
2. The response time of the system with $N$ users cannot be less than a system with just one user. This puts a limit on the minimum response time.
3. The interactive response time formula can be used to convert the bound on throughput to that on response time and vice versa.

We now derive the bounds using these observations. For the bottleneck device $b$ we have

$$U_b = XD_{max}$$

Since $U_b$ cannot be more than 1, we have

$$XD_{max} \le 1$$

or

$$X \le \frac{1}{D_{max}} \tag{33.10}$$

With just one job in the system, there is no queueing and the system response time is simply the sum of the service demands:

$$R(1) = D_1 + D_2 + \cdots + D_M = D$$

Here, $D$ is defined as the sum of all service demands. With more than one user there may be some queueing and so the response time will be higher. That is,

$$R(N) \ge D \tag{33.11}$$

Applying the interactive response time law to the bounds specified by Equations (33.10) and (33.11), we get

$$R(N) = \frac{N}{X(N)} - Z \ge ND_{max} - Z$$



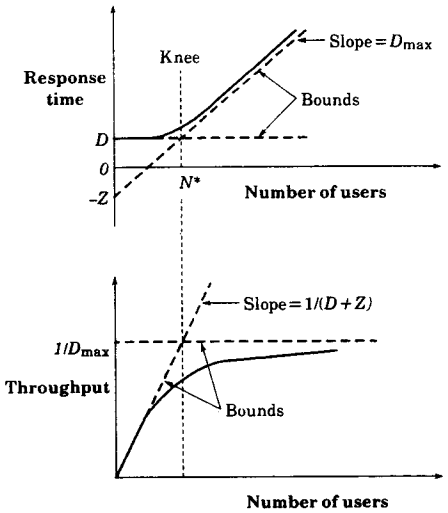


**FIGURE 33.3** Typical asymptotic bounds.

and

$$X(N) = \frac{N}{R(N)+Z} \le \frac{N}{D+Z}$$

Combining these bounds with Equations (33.10) and (33.11), we get the asymptotic bounds as specified in Equations (33.8) and (33.9). □

Figure 33.3 shows asymptotic bounds for a typical case. As shown in the figure, both throughput and response time bounds consist of two straight lines. The response time bounds consist of a horizontal straight line at $R = D$ and a line passing through the point $(-Z, 0)$ at a slope of $D_{max}$. The throughput bounds consist of a horizontal straight line at $X = 1/D_{max}$ and a line passing through the origin at a slope of $1/(D+Z)$. The point of intersection of the two lines is called the **knee**. For both response time and throughput, the knee occurs at the same value of number of users. The number of jobs $N^*$ at the knee is given by

$$D = N^*D_{max} - Z \quad \text{or} \quad N^* = \frac{D+Z}{D_{max}}$$

If the number of jobs is more than $N^*$, then we can say with certainty that there is queueing somewhere in the system.

The asymptotic bounds are useful in practice since they can be easily derived and explained to people who do not have any background in queueing theory or performance analysis.

The following examples illustrate the use of asymptotic bounds.

**Example 33.7** For the timesharing system considered in Example 33.2:

$$D_{CPU} = 5, \quad D_A = 4, \quad D_B = 3, \quad Z = 18$$

$$D = D_{CPU} + D_A + D_B = 5 + 4 + 3 = 12$$

$$D_{max} = D_{CPU} = 5$$

The asymptotic bounds are

$$X(N) \le \min\left\{\frac{N}{D+Z}, \frac{1}{D_{max}}\right\} = \min\left\{\frac{N}{30}, \frac{1}{5}\right\}$$

$$R(N) \ge \max\{D, ND_{max} - Z\} = \max\{12, 5N - 18\}$$

These bounds are shown by broken lines in Figure 33.4. The solid lines show the exact throughput and response times (obtained using the mean-value analysis to be discussed in Section 34.2). The knee occurs at

$$12 = 5N^* - 18$$

or

$$N^* = \frac{12+18}{5} = \frac{30}{5} = 6$$

Thus, if there are more than six users on the system, there will certainly be queueing in the system. □

**Example 33.8** How many terminals can be supported on the timesharing system of Example 33.2 if the response time has to be kept below 100 seconds?

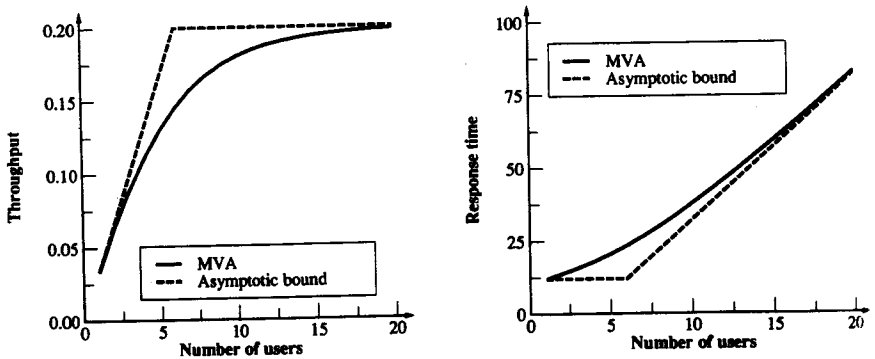


**FIGURE 33.4** Asymptotic bounds on the throughput and the response time.



Using the asymptotic bounds on the response time, we get

$$R(N) \ge \max\{12, 5N - 18\}$$

The response time will be more than 100 if

$$5N - 18 \ge 100$$

That is, if

$$N \ge 23.6$$

the response time is bound to be more than 100. Thus, the system cannot support more than 23 users if a response time of less than 100 is required. □

This completes the discussion on operational laws. A summary of all operational laws is presented in Box 33.1.

**Box 33.1 Operational Laws**

| | |
|---|---|
| Utilization law | $U_i = X_i S_i = X D_i$ |
| Forced flow law | $X_i = X V_i$ |
| Little's law | $Q_i = X_i R_i$ |
| General response time law | $R = \sum_{i=1}^{M} R_i V_i$ |
| Interactive response time law | $R = N/X - Z$ |
| Asymptotic bounds | $R \ge \max\{D, ND_{max} - Z\}$ |
| | $X \le \min\{1/D_{max}, N/(D+Z)\}$ |

Symbols:

| | |
|---|---|
| $D$ | Sum of service demands on all devices, $= \sum_i D_i$ |
| $D_i$ | Total service demand per job for the $i$th device, $= S_i V_i$ |
| $D_{max}$ | Service demand on the bottleneck device, $= \max_i\{D_i\}$ |
| $N$ | Number of jobs in the system |
| $Q_i$ | Number in the $i$th device |
| $R$ | System response time |
| $R_i$ | Response time per visit to the $i$th device |
| $S_i$ | Service time per visit to the $i$th device |
| $U_i$ | Utilization of the $i$th device |
| $V_i$ | Number of visits per job to the $i$th device |
| $X$ | System throughput |
| $X_i$ | Throughput of the $i$th device |
| $Z$ | Think time |

## EXERCISES

**33.1** During a 10-second observation period, 400 packets were serviced by a gateway whose CPU can service 200 pps. What was the utilization of the gateway CPU?

**33.2** The throughput of a timesharing system was observed to be five jobs per second over a 10-minute observation period. If the average number of jobs in the system was four during this period, what was the average response time?

**33.3** During a 10-second observation period, 40 requests were serviced by a file server. Each request requires two disk accesses. The average service time at the disk was 30 milliseconds. What was the average disk utilization during this period?

**33.4** A distributed system has a print server with a printing speed of 60 pages per minute. The server was observed to print 500 pages over a 10-minute observation period. If each job prints five pages on the average, what was the job completion rate of the system?

**33.5** For a timesharing system with two disks (user and system), the probabilities for jobs completing the service at the CPU were found to be 0.80 to disk A, 0.16 to disk B, and 0.04 to the terminals. The user think time was measured to be 5 seconds, the disk service times were 30 and 25 milliseconds, and the average service time per visit to the CPU was 40 milliseconds. Using the queueing network model shown in Figure 32.8, answer the following for this system:

- **a.** For each job, what are the visit ratios for CPU, disk A, and disk B?
- **b.** For each device, what is the total service demand?
- **c.** If disk A utilization is 60%, what is the utilization of the CPU and disk B?
- **d.** If the utilization of disk B is 10%, what is the average response time when there are 20 users on the system?

**33.6** For a transaction processing system, which can be represented by the open queueing network model of Figure 32.1, each transaction was found to make seven I/O's to disk A and eight I/O's to disk B and a total of 16 visits to the CPU. The service time per visit to the two disks and CPU were 20, 30, and 10 milliseconds, repectively. For this system, answer the following:

- **a.** What is the bottleneck device?
- **b.** What is the minimum average response time?
- **c.** What is the maximum possible disk A utilization for this configuration?
- **d.** What is the maximum possible throughput of this system?



- **e.** What changes in CPU speed would you recommend to achieve a response time of 10 seconds with 25 users? Would you also need a faster disk A or disk B?
- **f.** Write the expressions for asymptotic bounds on throughput and response time.

**33.7** For the system of Exercise 33.6, which device would be the bottleneck if

- **a.** the CPU is replaced by another unit that is twice as fast?
- **b.** disk A is replaced by another unit that is twice as slow?
- **c.** disk B is replaced by another unit that is twice as slow?
- **d.** the memory size is reduced so that the jobs make 20 times more visits to disk B due to increased page faults?