# Chapter 3

# Fundamental Laws

## 3.1. Introduction

This chapter provides the technical foundation for much of the remainder of the book. It has three objectives. The first is to define a number of quantities of interest and to introduce the notation that we will use in referring to these quantities. The second is to derive various algebraic relationships among these quantities, some of which, because of their importance, will be identified as *fundamental laws.* The third is to explore thoroughly the most important of these fundamental laws, *Little's law* (named for J.D.C. Little), which states that the average number of requests in a system must equal the product of the throughput of that system and the average time spent in that system by a request.

Because of the volume of notation introduced, this chapter may appear formidable. It is not. The material is summarized in three small tables in Section 3.6, which we suggest you copy for convenient reference.

## 3.2. Basic Quantities

If we were to observe the abstract system shown in Figure 3.1 we might imagine measuring the following quantities:

$T$, the length of *time* we observed the system

$A$, the number of request *arrivals* we observed

$C$, the number of request *completions* we observed

From these measurements we can define the following additional quantities:

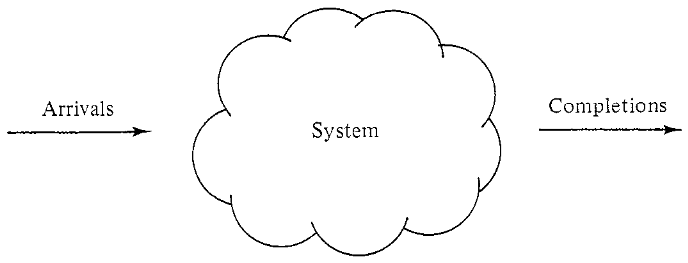

**Figure 3.1 – An Abstract System**

$\lambda$, the *arrival rate*: $\lambda \equiv \frac{A}{T}$

If we observe 8 arrivals during an observation interval of 4 minutes, then the arrival rate is 8/4 = 2 requests/minute.

$X$, the *throughput*: $X \equiv \frac{C}{T}$

If we observe 8 completions during an observation interval of 4 minutes, then the throughput is 8/4 = 2 requests/minute.

If the system consists of a single resource, we also can measure:

$B$, the length of time that the resource was observed to be *busy*

Two more defined quantities now are meaningful:

$U$, the *utilization*: $U \equiv \frac{B}{T}$

If the resource is busy for 2 minutes during a 4 minute observation interval, then the utilization of the resource is 2/4, or 50%.

$S$, the average *service requirement* per request: $S \equiv \frac{B}{C}$

If we observe 8 completions during an observation interval and the resource is busy for 2 minutes during that interval, then on the average each request requires 2/8 minutes of service.

We now can derive the first of our fundamental laws. Algebraically, $\frac{B}{T} = \frac{C}{T}\frac{B}{C}$. From the three preceding definitions, $\frac{B}{T} \equiv U$, $\frac{C}{T} \equiv X$, and $\frac{B}{C} \equiv S$. Hence:

$$\boxed{\textbf{The Utilization Law:}\quad U = XS}$$

That is, the utilization of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource. As an example, consider a disk that is serving 40 requests/second, each of which requires .0225 seconds of disk service. The utilization law tells us that the utilization of this disk must be $40 \times .0225 = 90\%$.

## 3.3. Little's Law

The utilization law in fact is a special case of Little's law, which we now will derive in a more general setting. Figure 3.2 is a graph of the total number of arrivals and completions occurring at a system over time. Each step in the higher step function signifies the occurrence of an arrival at that instant; each step in the lower signifies a completion. At any instant, the vertical distance between the arrival and completion functions represents the number of requests present in the system. Over any time interval, the area between the arrival and completion functions represents the accumulated time in system during that interval, measured in request-seconds (or request-minutes, etc.). For example, if there are three requests in the system during a two second period, then six request-seconds are accumulated. This area is shaded in Figure 3.2 for an observation interval of length $T = 4$ minutes. We temporarily denote accumulated time in system by $W$. We define:

$N$, the average *number of requests* in the system: $N \equiv \frac{W}{T}$

If a total of 2 request-minutes of residence time are accumulated during a 4 minute observation interval, then the average number of requests in the system is $2/4 = 0.5$.

$R$, the average system *residence time* per request: $R \equiv \frac{W}{C}$

If a total of 2 request-minutes of residence time are accumulated during an observation interval in which 8 requests complete, then the average contribution of each completing request (informally, the average system residence time per request) is $2/8 = 0.25$ minutes.

Algebraically, $\frac{W}{T} = \frac{C}{T}\frac{W}{C}$. But $\frac{W}{T} \equiv N$, $\frac{C}{T} \equiv X$, and $\frac{W}{C} \equiv R$.

Hence:

$$\boxed{\textbf{Little's Law: } N = XR}$$

That is, the average number of requests in a system is equal to the product of the throughput of that system and the average time spent in that system by a request.

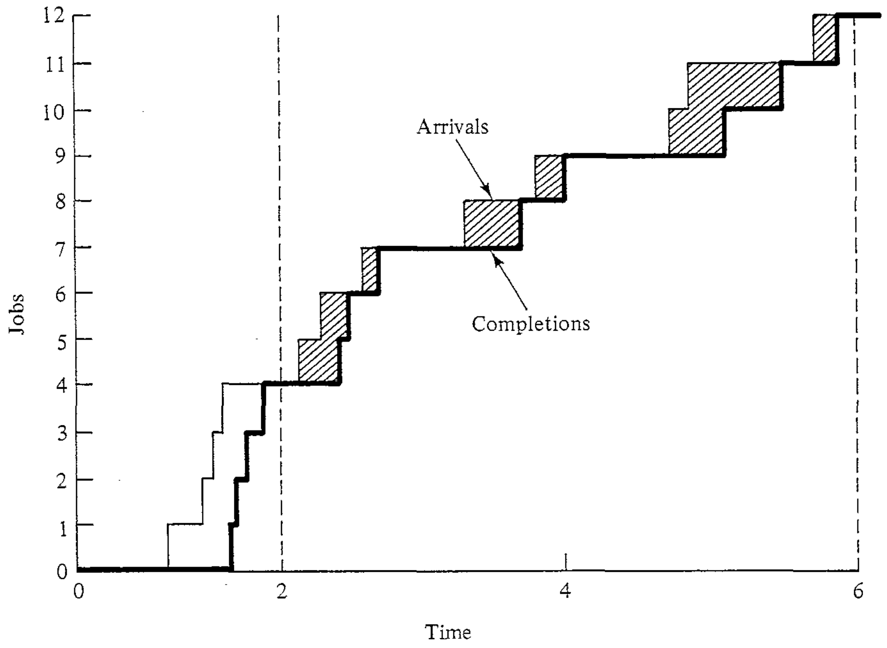

**Figure 3.2 — System Arrivals and Completions**

A subtle but important point in our derivation of Little's law is that the quantity $R$ does not necessarily correspond to our intuitive notion of average residence time or response time — the expected time from arrival to departure. This discrepancy is due to end effects: it is hard to know how to account for requests that are present just prior to the start or just after the end of an observation interval. For the time being, suffice it to say that if the number of requests passing through the system during the observation interval is significantly greater than the number present at the beginning or end, then $R$ corresponds closely to our intuition, and if the observation interval begins and ends at instants when system is empty, then this correspondence is exact. (End effects arise elsewhere; for example, considerations similar to those affecting $R$ also affect our earlier definition of $S$, the average service requirement per request.)

Little's law is important for three reasons. First, because it is so widely applicable (it requires only very weak assumptions), it will be valuable to us in checking the consistency of measurement data. Second, in studying computer systems we frequently will find that we know two of the quantities related by Little's law (say, the average number of requests in a system and the throughput of that system) and desire to know the third (the average system residence time, in this case). Third, Little's law is central to the algorithms for evaluating queueing network models, which we will introduce in Part II.

Given a computer system, Little's law can be applied at many different levels: to a single resource, to a subsystem, or to the system as a whole. The key to success is consistency: the definitions of population, throughput, and residence time must be compatible with one another. In Figure 3.3 we illustrate this by applying Little's law to a hypothetical timesharing system at four different levels, as indicated by the four boxes in the figure.

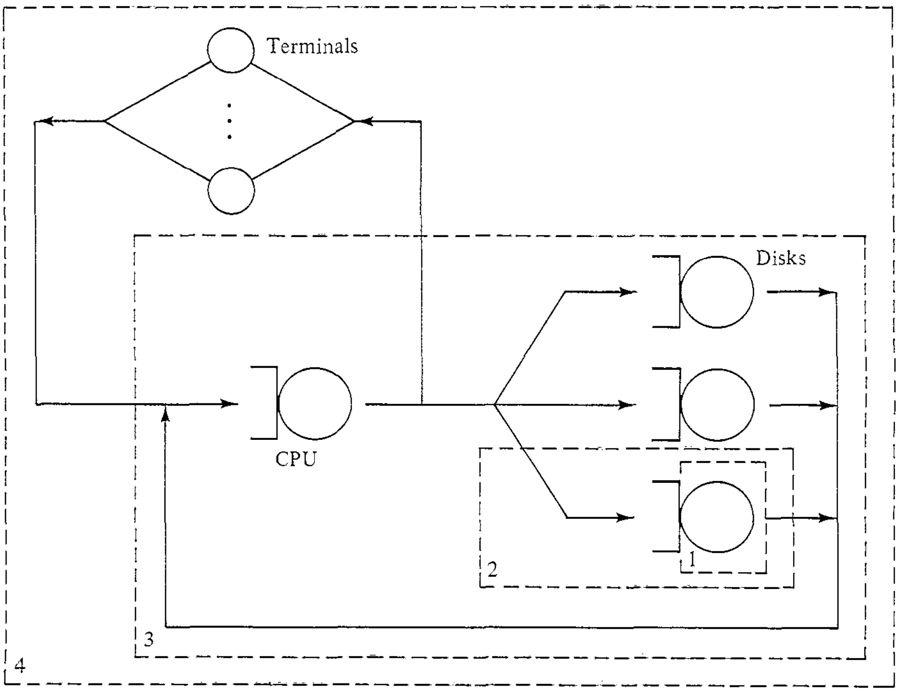

**Figure 3.3 – Little's Law Applied at Four Levels**

Box 1 is perhaps the most subtle; it illustrates the application of Little's law to a single resource, *not including* its queue. In this example, population corresponds to the utilization of the resource (there are either zero or one requests present at any instant in time; the resource is utilized whenever there is one request present; thus resource utilization is equal to the proportion of time there is one request present, which is also equal to the average number of requests present), throughput corresponds to the rate at which the resource is satisfying requests, and residence time corresponds to the average service requirement per request at the resource (remember, queueing delay is not included in this application of Little's law; once a request acquires the resource, it remains at that resource for its service time). This application of Little's law constitutes an alternative derivation of the utilization law. To repeat the example used previously, suppose that the resource is a disk, that the disk is serving 40 requests/second ($X = 40$), and that the average request requires .0225 seconds of disk service ($S = .0225$). Then Little's law ($U = XS$) tells us that the utilization of the disk must be $40 \times .0225 = 90\%$.

Box 2 illustrates the application of Little's law to the same resource, this time including its queue. Now, population corresponds to the total number of requests either in queue or in service, throughput remains the rate at which the resource is satisfying requests, and residence time corresponds to the average time that a request spends at the resource per visit, both queueing time and service time. Suppose that the average number of requests present is 4 ($N = 4$) and that the disk is serving 40 requests/second ($X = 40$). Then Little's law ($N = XR$) tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds. Note that we can now compute the average queueing time of a request (a total of 0.1 seconds are spent both queueing and receiving service, of which .0225 seconds are devoted to receiving service, so the average queueing time must be .0775 seconds) and also the average number of requests in the queue (an average total of 4 requests are either queueing or receiving service, and on the average there are 0.9 requests receiving service, so the average number awaiting service in the queue must be 3.1).

Box 3 illustrates the application of Little's law to the *central subsystem* — the system without its terminals. Our definition of "request" changes at this level: we are no longer interested in visits to a particular resource, but rather in system-level interactions. Population corresponds to the number of customers in the central subsystem, i.e., those users not thinking. Throughput corresponds to the rate at which interactions flow between the terminals and the central subsystem. Residence time corresponds to our conventional notion of response time: the period of time from when a user submits a request until that user's response is returned. Suppose that system throughput is 1/2 interaction per second ($X = 0.5$) and that, on the average, there are 7.5 "ready" users ($N = 7.5$). Then Little's law ($N = XR$) tells us that average response time must be $7.5/0.5 = 15$ seconds.

Finally, box 4 illustrates the application of Little's law to the entire system, including its terminals. Here, population corresponds to the total number of interactive users, throughput corresponds to the rate at which interactions flow between the terminals and the system, and residence time corresponds to the sum of system response time and user think time. Suppose that there are 10 users, average think time is 5 seconds, and average response time is 15 seconds. Then Little's law tells us that the system throughput must be $\frac{10}{15+5} = 0.5$ interactions/second. If we denote think time by $Z$ then we can write this incarnation of Little's law as $N = X(R+Z)$. As with the utilization law, this application is so ubiquitous that we give it its own name and notation, expressing $R$ in terms of the other quantities:

$$\boxed{\textbf{The Response Time Law: } R = \frac{N}{X} - Z}$$

As an example application of the response time law, suppose that a system has 64 interactive users, that the average think time is 30 seconds, and that system throughput is 2 interactions/second. Then the response time law tells us that response time must be $\frac{64}{2} - 30 = 2$ seconds.

In earlier chapters we have noted that throughputs and utilizations typically are projected with greater accuracy than residence times. We now are in a position to understand why this must be. Suppose we were to construct a queueing network model of the system in the previous example. The number of users (64) and the average think time (30 seconds) would be parameters of the model, along with the service demands at the various resources in the system. Throughput and response time would be outputs of the model. Suppose that the model projected a throughput of 1.9 interactions/second, an error of just 5%. Since the response time law must be satisfied by the queueing network model, a compensating error in projected response time must result:

$$R = \frac{64}{1.9} - 30$$

Thus the model must project a response time of 3.7 seconds, an error of 85%.

## 3.4. The Forced Flow Law

In discussing Little's law, we allowed our field of view to range from an individual resource to an entire system. At different levels of detail, different definitions of "request" are appropriate. For example, when considering a disk, it is natural to define a request to be a disk access, and to measure throughput and residence time on this basis. When considering an entire system, on the other hand, it is natural to define a request to be a user-level interaction, and to measure throughput and residence time on this basis.

The relationship between these two views of a system is expressed by the *forced flow law*, which states that the flows (throughputs) in all parts of a system must be proportional to one another. Suppose that during an observation interval we count not only system completions, but also the number of completions at each resource. We define the *visit count* of a resource to be the ratio of the number of completions at that resource to the number of system completions, or, more intuitively, to be the average number of visits that a system-level request makes to that resource. If we let a variable with the subscript $k$ refer to the $k$-th resource (a variable with no subscript continues to refer to the system as a whole), then we can write this definition as:

$V_k$, the *visit count* of resource $k$: $V_k \equiv \dfrac{C_k}{C}$

> If during an observation interval we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 150/10 = 15 disk operations.

If we rewrite this definition as $C_k = V_k C$ and recall that the completion count divided by the length of the observation interval is defined to be the throughput, then the throughput of resource $k$ is given by:

$$\boxed{\textbf{The Forced Flow Law:}\ X_k = V_k X}$$

An informal statement of the forced flow law is that the various components of a system must do comparable amounts of work (measured in "transaction's worth") in a given time interval. As an example, suppose we are told that each job in a batch processing system requires an average of 6 accesses to a specific disk, and that the disk is servicing 12 requests from batch jobs per second. Then we know that the system throughput of batch jobs must be 12/6 = 2 jobs/second. If, in addition, we are told that another disk is servicing 18 batch job requests per second, then we know that each batch job requires on average 18/2 = 9 accesses to this second disk.

Little's law becomes especially powerful when combined with the forced flow law. As an example, suppose that we are asked to determine average system response time for an interactive system with the following known characteristics:

25 terminals ($N = 25$)
18 seconds average think time ($Z = 18$)
20 visits to a specific disk per interaction ($V_{disk} = 20$)
30% utilization of that disk ($U_{disk} = .30$)
25 millisecond average service requirement per visit
to that disk ($S_{disk} = .025$ secs.)

We would like to apply the response time law: $R = \frac{N}{X} - Z$. We know the number of terminals and the average think time, but are missing the throughput. We do, however, know the visit count at one specific disk (that is, the average number of visits made to that disk by an interactive request), so if we knew the throughput at that disk we would be able to apply the forced flow law to obtain system-level throughput. To obtain disk throughput we can use the utilization law, since we know both utilization and service requirement at this device. We calculate the following quantities:

$$\text{disk throughput: } X_{disk} = \frac{U_{disk}}{S_{disk}} = \frac{.30}{.025} = 12 \text{ requests/sec.}$$

$$\text{system throughput: } X = \frac{X_{disk}}{V_{disk}} = \frac{12}{20} = 0.6 \text{ interactions/sec.}$$

$$\text{response time: } R = \frac{N}{X} - Z = \frac{25}{0.6} - 18 = 23.7 \text{ secs.}$$

Note that we can describe an interaction's disk service requirement in either of two ways: by saying that an interaction makes a certain number of visits to the disk and requires a certain amount of service on each visit, or by specifying the total amount of disk service required by an interaction. These two points of view are equivalent, and whichever is more convenient should be chosen. We define:

$D_k$, the *service demand* at resource $k$: $D_k \equiv V_k S_k$

If a job makes an average of 20 visits to a disk and requires an average of 25 milliseconds of service per visit, then that job requires a total of $20 \times 25 = 500$ milliseconds of disk service, so its service demand is 500 milliseconds at that disk.

From now on we will use $S_k$ to refer to the service requirement per visit at resource $k$, and $D_k$ to refer to the total service requirement at that resource. We define $D$, with no subscript, to be the sum of the $D_k$: the total service demanded by a job at all resources.

Again, consistency is crucial to success. Consider using the utilization law to calculate the utilization of a resource. We can express throughput in terms of visits to that resource ($X_k$), in which case service requirement must be expressed as service requirement per visit ($S_k$). Using the forced flow law, we can also express throughput in terms of system-level interactions ($X$), in which case service requirement must be expressed on a per-interaction basis ($D_k$). In other words, $U_k = X_k S_k = X D_k$.

In Chapter 1 we observed that service demands are one of the parameters required by queueing network models. If we observe a system for an interval of length $T$, we can easily obtain the utilizations of the various resources, $U_k$, and the system-level completion count, $C$. The service demands at the various resources then can be calculated as $D_k = \frac{B_k}{C} = \frac{U_k T}{C}$. It is fortunate that queueing network models can be parameterized in terms of the $D_k$ rather than the corresponding $V_k$ and $S_k$, since the former typically are much more easily obtained from measurement data than the latter.

As a final illustration of the versatility of Little's law in conjunction with the forced flow law, consider Figure 3.4, which represents a timesharing system with a memory constraint: swapping may occur between interactions, so a request may be forced to queue for a memory partition prior to competing for the resources of the central subsystem. As indicated by the boxes, we once again are going to apply Little's law at several different levels. The following actual measurement data was obtained by observing the timesharing workload on a system with several distinct workloads:

> average number of timesharing users: 23 ($N = 23$)
> average response time perceived by a user: 30 seconds ($R = 30$)
> timesharing throughput: 0.45 interactions/second ($X = .45$)
> average number of timesharing requests occupying memory: 1.9 ($N_{in\ mem} = 1.9$)
> average CPU service requirement per interaction: 0.63 seconds ($D_{CPU} = .63$)

Now, consider the following questions:

- What was the average think time of a timesharing user? Applying the response time law at the level of box 4 in the figure, $R = \frac{N}{X} - Z$, so $Z = \frac{23}{.45} - 30$, or 21 seconds.
- On the average, how many users were attempting to obtain service, i.e., how many users were not "thinking" at their terminals? Applying Little's law at the level of box 3, $N_{want\ mem} = XR = .45 \times 30$, or 13.5 users. Of the 23 users on this system, an average of 13.5 were

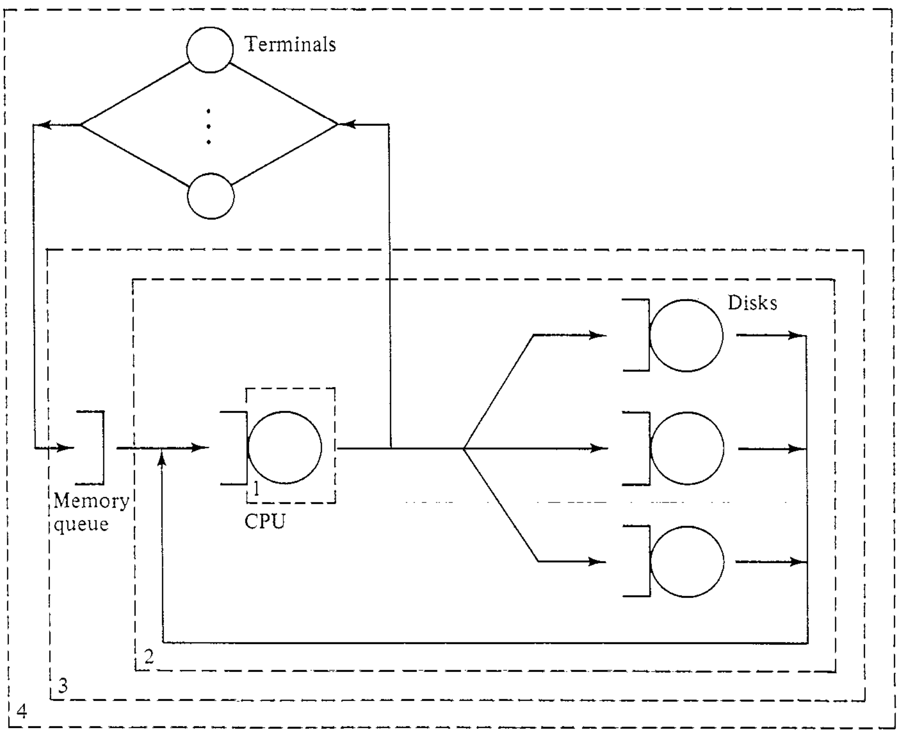

**Figure 3.4 – Little's Law Applied to a Memory Constrained System**

attempting to obtain service at any one time. We know from measurement data that only 1.9 were occupying memory on the average, so the remaining 11.6 must have been queued awaiting access to memory.

- On the average, how much time elapses between the acquisition of memory and the completion of an interaction? Applying Little's law at the level of box 2, $N_{in\ mem} = XR_{in\ mem}$, so $R_{in\ mem} = 1.9/0.45$, or 4.2 seconds. In other words, of the 30 second response time perceived by a user, nearly 26 seconds are spent queued awaiting access to memory.
- What is the contribution to CPU utilization of the timesharing workload? Applying the utilization law to the CPU (box 1), $U_{CPU} = XD_{CPU} = .45 \times .63$, or 28% of the capacity of the CPU. Notice that in this application of the utilization law, throughput was defined in terms of system-level interactions and service requirement was defined on a per-interaction basis.

## 3.5. The Flow Balance Assumption

Frequently it will be convenient to assume that systems satisfy the *flow balance* property, namely, that the number of arrivals equals the number of completions, and thus the arrival rate equals the throughput:

$$\boxed{\textbf{The Flow Balance Assumption: } A = C, \text{ therefore } \lambda = X}$$

The flow balance assumption can be tested over any measurement interval, and it can be strictly satisfied by careful choice of measurement interval.

When used in conjunction with the flow balance assumption, Little's law and the forced flow law allow us to calculate device utilizations for systems whose workload intensities are described in terms of an arrival rate. In Figure 3.5 we show a queueing network model similar to that used to represent the VAX-11/780 in the case study described in Section 2.4. There are three devices (a CPU and two disks) and three transaction classes with the following characteristics:

| | | **service demand, seconds/transaction** | | |
|---|---|---|---|---|
| **transaction class** | **arrival rate trans./hr.** | **CPU** | **disk 1** | **disk 2** |
| compilation | 480 | 2.0 | 0.75 | 0.25 |
| execution | 120 | 11.9 | 5.0 | 5.7 |
| editing session | 600 | 0.5 | 0.2 | 0.6 |

To calculate the utilization of a device in this system we apply the utilization law separately to each transaction class, then sum the results. As an example, consider the CPU. If compilation transactions are arriving to the system at a rate of 480/hour and each one brings 2.0 seconds of work to the CPU, then CPU utilization due to compilation transactions must equal $\frac{480}{3600} \times 2.0 = 27\%$. Similar arguments for execution and editing transactions yield CPU utilizations of 40% and 8%, respectively. Thus total CPU utilization must be 75%.

How is it possible to analyze the classes independently without accounting for their mutual interference? Assuming that the system is able to handle the offered load (i.e., assuming that the calculated utilization of no device is greater than 100%), the flow balance assumption is reasonable. Thus the throughput of the system will be the same as the arrival rate to the system. The forced flow law guarantees that the various devices in the system will do comparable amounts of work (measured

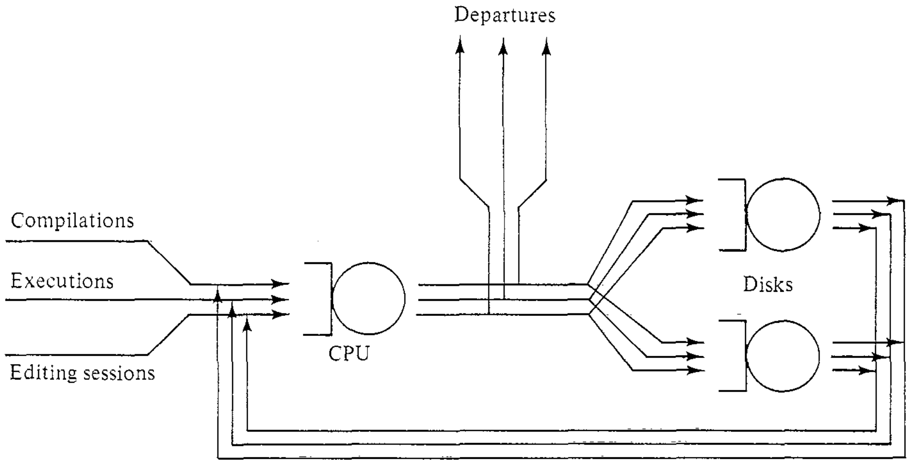

**Figure 3.5 — Calculating Utilizations Using Flow Balance**

in "transaction's worth") in a given time. Interference between transactions does not affect this. Rather, it causes an increase in the average number of transactions resident in the system, which causes a corresponding increase in response time (by Little's law). In Part II we will learn how to quantify the extent of this interference.

## 3.6. Summary

In this chapter we have defined a number of quantities of interest, introduced the notation that we will use in referring to these quantities, and derived various algebraic relationships among these quantities. These developments are reviewed in the following tables, which we suggest you copy for convenient reference.

Table 3.1 summarizes the notation that we have established. The table includes a subscript on those quantities that require one, either explicit or implicit. In some cases, the quantity must refer to a specific resource. In other cases, the quantity may refer either to a specific resource or to a specific subsystem. Table 3.2 summarizes the fundamental laws. Table 3.3 summarizes the additional algebraic relationships among the various quantities that we have defined. We also have introduced and used the flow balance assumption: $A = C$, therefore $\lambda = X$.

| | |
|---|---|
| $T$ | length of an observation interval |
| $A_k$ | number of arrivals observed |
| $C_k$ | number of completions observed |
| $\lambda_k$ | arrival rate |
| $X_k$ | throughput |
| $B_k$ | busy time |
| $U_k$ | utilization |
| $S_k$ | service requirement per visit |
| $N$ | customer population |
| $R_k$ | residence time |
| $Z$ | think time of a terminal user |
| $V_k$ | number of visits |
| $D_k$ | service demand |

**Table 3.1 — Notation**

| | |
|---|---|
| **The Utilization Law:** | $U_k = X_k S_k = X D_k$ |
| **Little's Law:** | $N = XR$ |
| **The Response Time Law:** | $R = \frac{N}{X} - Z$ |
| **The Forced Flow Law:** | $X_k = V_k X$ |

**Table 3.2 — Fundamental Laws**

## 3.7. References

Buzen and Denning's *operational analysis* has heavily influenced our philosophy in general, and this chapter in particular. Much of the notation and the identification of laws and assumptions is taken from their work. Of special note are [Buzen 1976] (from which we have even borrowed the title of this chapter) and [Denning & Buzen 1978].

Little's law is named for J.D.C. Little, who first proved it in 1961 [Little 1961].

$$\lambda_k \equiv \frac{A_k}{T}$$
$$X_k \equiv \frac{C_k}{T}$$
$$U_k \equiv \frac{B_k}{T}$$
$$S_k \equiv \frac{B_k}{C_k} = \frac{U_k T}{C_k}$$
$$V_k \equiv \frac{C_k}{C}$$
$$D_k \equiv V_k S_k = \frac{B_k}{C} = \frac{U_k T}{C}$$

**Table 3.3 — Additional Relationships**

[Buzen 1976]
Jeffrey P. Buzen. Fundamental Operational Laws of Computer System Performance. *Acta Informatica* 7,2 (1976), 167-182.

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

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

## 3.8. Exercises

1. Consider the specific computer system with which you are most familiar. How would you calculate the basic service demand $D_k$ at the CPU? At each disk device? How would you calculate the average number of jobs in memory?
2. Software monitor data for an interactive system shows a CPU utilization of 75%, a 3 second CPU service demand, a response time of 15 seconds, and 10 active users. What is the average think time of these users?

3. An interactive system with 80 active terminals shows an average think time of 12 seconds. On average, each interaction causes 15 paging disk accesses. If the service time per paging disk access is 30 ms. and this disk is 60% busy, what is the average system response time?

4. Suppose an interactive system is supporting 100 users with 15 second think times and a system throughput of 5 interactions/second.

   a. What is the response time of the system?

   b. Suppose that the service demands of the workload evolve over time so that system throughput drops to 50% of its former value (i.e., to 2.5 interactions/second). Assuming that there still are 100 users with 15 second think times, what would their response time be?

   c. How do you account for the fact that response time in (b) is more than twice as large as that in (a)?

5. Consider a system modelled as shown in Figure 3.6. A user request submitted to the system must queue for memory, and may begin processing (in the central subsystem) only when it has obtained a memory partition.

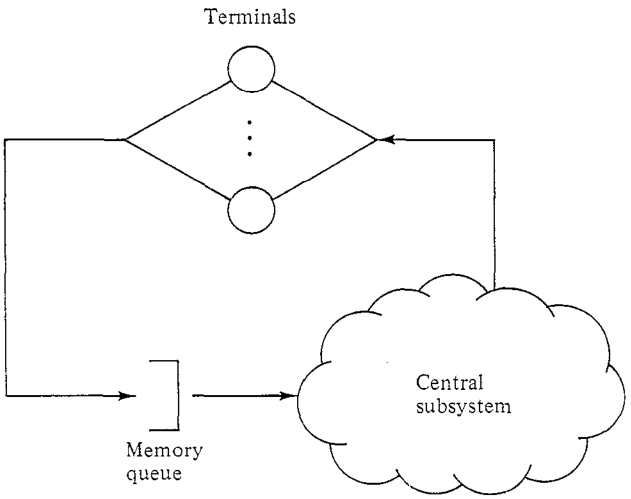

**Figure 3.6 – A Memory Constrained System**

   a. If there are 100 active users with 20 second think times, and system response time (the sum of memory queueing and central subsystem residence times) is 10 seconds, how many customers are competing for memory on average?

b. If memory queueing time is 8 seconds, what is the average number of customers loaded in memory?

6. In a 30 minute observation interval, a particular disk was found to be busy for 12 minutes. If it is known that jobs require 320 accesses to that disk on average, and that the average service time per access is 25 milliseconds, what is the system throughput (in jobs/second)?

7. Consider a very simple model of a computer system in which only the CPU is represented. Use Little's law to argue that the minimum average response time for this system is obtained by scheduling the CPU so that it always serves the job with the shortest expected remaining service time (i.e., the job that is expected to finish soonest if placed in service).

8. Consider the following measurement data for an interactive system with a memory constraint:

| | |
|---|---|
| length of measurement interval: | 1 hour |
| average number of users: | 80 |
| average response time: | 1 second |
| average number of memory-resident requests: | 6 |
| number of request completions: | 36,000 |
| utilizations of: | |
| CPU | 75% |
| Disk 1 | 50% |
| Disk 2 | 50% |
| Disk 3 | 25% |

a. What was throughput (in requests / second)?

b. What was the average "think time"?

c. On the average, how many users were attempting to obtain service (i.e., not "thinking")?

d. On the average, how much time does a user spend waiting for memory (i.e., not "thinking" but not memory-resident)?

e. What is the average service demand at Disk 1?