CHAPTER 20

QUEUEING ANALYSIS

20.1 HOW QUEUES BEHAVE—A SIMPLE EXAMPLE 3
20.2 WHY QUEUEING ANALYSIS? 8
20.3 QUEUEING MODELS 12
The Single-Server Queue 12
Queue Parameters 13
Illustration of Key Features 14
Model Characteristics 15
The Multiserver Queue 17
Basic Queueing Relationships 19
Assumptions 21
20.4 SINGLE-SERVER QUEUES 24
20.5 MULTISERVER QUEUES 28
20.6 EXAMPLES 30
Database Server 30
Calculating Percentiles 31
Tightly-Coupled Multiprocessor 33
Single-Server Approach 34
Multiserver Approach 34
A Multiserver Problem 35
Single-Server Model 35
Multiserver Model 36
20.7 QUEUES WITH PRIORITIES 38
20.8 NETWORKS OF QUEUES 40
Partitioning and Merging of Traffic Streams 41
Queues in Tandem 42
Jackson's Theorem 43
Application to a Packet-Switching Network 44
20.9 OTHER QUEUEING MODELS 46
20.10 ESTIMATING MODEL PARAMETERS 48
Sampling 48
Sampling Errors 52
20.11 RECOMMENDED READING 53
20.12 PROBLEMS 54

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Understand the characteristic behavior of queueing systems.
- Explain the value of queueing analysis.
- Explain the key features of single-server and multiserver queues.
- Analyze single-server queueing models.
- Analyze multiserver queueing models.
- Describe the effect of priorities on queueing performance
- Understand the key concept relating to queueing networks.
- Understand the issues involved in estimating queueing model parameters.

Queueing[1] analysis is one of the most important tools for those involved with computer and network analysis. It can be used to provide approximate answers to a host of questions, such as:

- What happens to file retrieval time when disk I/O utilization goes up?
- Does response time change if both processor speed and the number of users on the system are doubled?
- How will performance be affected if the process scheduling algorithm includes priorities?
- Which dick scheduling algorithm produces the best average performance?

[1] Two spellings are in use: queueing and queuing. The vast majority of queueing theory researchers use *queueing*. The premier journal in the fields is *Queueing systems: Theory and Applications*. On the other hand, most American dictionaries and spell checkers prefer the spelling *queuing*.

The number of questions that can be addressed with a queueing analysis is endless and touches on virtually every area in computer science. The ability to make such an analysis is an essential tool for those involved in this field.

Although the theory of queueing is mathematically complex, the application of queueing theory to the analysis of performance is, in many cases, remarkably straightforward. A knowledge of elementary statistical concepts (means and standard deviations) and a basic understanding of the applicability of queueing theory is all that is required. Armed with these, the analyst can often make a queueing analysis on the back of an envelope using readily available queueing tables, or with the use of simple computer programs that occupy only a few lines of code.

The purpose of this paper is to provide a practical guide to queueing analysis. A subset, although a very important subset, of the subject is addressed. In the final section, pointers to additional references are provided. An annex to this paper reviews some elementary concepts in probability and statistics.

This chapter provides a practical guide to queueing analysis. A subset, although a very important subset, of the subject is addressed.

20.1 HOW QUEUES BEHAVE—A SIMPLE EXAMPLE

Before getting into the details of queueing analysis, let us look at a crude example that will give some feel for the topic. Consider a Web server that is capable of handling an individual request in an average of 1 ms. In fact, to make things simple, assume that the server handles each request in exactly 1 ms. Now, if the rate of arriving requests is 1 per millisecond (1000/s), then it seems sensible to state that the server can keep up with the load.

Suppose that the requests arrive at a uniform rate of exactly one request each millisecond. When a request comes in, the server immediately handles the request. Just as the server completes the current request, a new request arrives and the server goes to work again.

Now let's take a more realistic approach and suppose that the average arrival rate for requests is 1 per millisecond but that there is some variability. During any given 1-ms period, there may be no requests, or one, or multiple requests, but the average is still 1 per millisecond. Again, common sense would seem to indicate that the server could keep up. During busy times, when lots of requests bunch up, the server can store outstanding requests in a buffer. Another way of putting this is to say that arriving requests enter a queue to await service. During quiet times, the server can catch up and clear the buffer. In this case, the interesting design issue would seem to be: How big should the buffer be?

Table 20.1 Queue Behavior with Normalized Arrival Rate of 0.5

Time	Input	Output	Queue
0	0	0	0
1	88	88	0
2	796	796	0
3	1627	1000	627
4	51	678	0
5	34	34	0
6	966	966	0
7	714	714	0
8	1276	1000	276
9	494	769	0
10	933	933	0
11	107	107	0
12	241	241	0
13	16	16	0
14	671	671	0
15	643	643	0
16	812	812	0
17	262	262	0
18	218	218	0
19	1378	1000	378
20	507	885	0
21	15	15	0
22	820	820	0
23	1253	1000	253
24	307	559	0
25	540	540	0

Time	Input	Output	Queue
26	190	190	0
27	500	500	0
28	96	96	0
29	943	943	0
30	105	105	0
31	183	183	0
32	447	447	0
33	542	542	0
34	166	166	0
35	165	165	0
36	490	490	0
37	510	510	0
38	877	877	0
39	37	37	0
40	163	163	0
41	104	104	0
42	42	42	0
43	291	291	0
44	645	645	0
45	363	363	0
46	134	134	0
47	920	920	0
48	1507	1000	507
49	598	1000	105
50	172	277	0
Average	499	499	43

Table 20.2 Queue Behavior with Normalized Arrival Rate of 0.95

Time	Input	Output	Queue
0	0	0	0
1	167	167	0
2	1512	1000	512
3	3091	1000	2604
4	97	1000	1701
5	65	1000	765
6	1835	1000	1601
7	1357	1000	1957
8	2424	1000	3382
9	939	1000	3320
10	1773	1000	4093
11	203	1000	3296
12	458	1000	2754
13	30	1000	1784
14	1275	1000	2059
15	1222	1000	2281
16	1543	1000	2824
17	498	1000	2322
18	414	1000	1736
19	2618	1000	3354
20	963	1000	3317
21	29	1000	2346
22	1558	1000	2904
23	2381	1000	4285
24	583	1000	3868
25	1026	1000	3894

Time	Input	Output	Queue
26	361	1000	3255
27	950	1000	3205
28	182	1000	2387
29	1792	1000	3179
30	200	1000	2378
31	348	1000	1726
32	849	1000	1575
33	1030	1000	1605
34	315	1000	921
35	314	1000	234
36	931	1000	165
37	969	1000	134
38	1666	1000	800
39	70	871	0
40	310	310	0
41	198	198	0
42	80	80	0
43	553	553	0
44	1226	1000	226
45	690	915	0
46	255	255	0
47	1748	1000	748
48	2863	1000	2611
49	1136	1000	2748
50	327	1000	2074
Average	948	907	1859

Table 20.3 Queue Behavior with Normalized Arrival Rate of 0.99

Time	Input	Output	Queue
0	0	0	0
1	174	174	0
2	1576	1000	576
3	3221	1000	2798
4	101	1000	1899
5	67	1000	966
6	1913	1000	1879
7	1414	1000	2292
8	2526	1000	3819
9	978	1000	3797
10	1847	1000	4644
11	212	1000	3856
12	477	1000	3333
13	32	1000	2365
14	1329	1000	2693
15	1273	1000	2967
16	1608	1000	3574
17	519	1000	3093
18	432	1000	2525
19	2728	1000	4253
20	1004	1000	4257
21	30	1000	3287
22	1624	1000	3910
23	2481	1000	5391
24	608	1000	4999
25	1069	1000	5068
26	376	1000	4445
27	990	1000	4435
28	190	1000	3625
29	1867	1000	4492
30	208	1000	3700
31	362	1000	3062
32	885	1000	2947
33	1073	1000	3020
34	329	1000	2349
35	327	1000	1676
36	970	1000	1646
37	1010	1000	1656
38	1736	1000	2392
39	73	1000	1465
40	323	1000	788
41	206	994	0
42	83	83	0
43	576	576	0
44	1277	1000	277
45	719	996	0
46	265	265	0
47	1822	1000	822
48	2984	1000	2805
49	1184	1000	2990
50	341	1000	2330
Average	988	942	2583

Tables 20.1 through 20.3 give a very rough idea of the behavior of this system. In Table 20.1, we assume an average arrival rate of 500 requests/s, which is half the capacity of the server. The entries in the table show the number of requests that arrive each second, the number of requests served during that second, and the number of outstanding requests waiting in the buffer at the end of the second. After 50 seconds, the table shows an average buffer contents of 43 requests, with a peak of over 600 requests. In Table 20.2, the average arrival rate is increased to 95% of the server's capacity, that is, 950 requests/s, and the average buffer contents rises to 1859. This seems a little surprising: the arrival rate has gone up by less than a factor of 2, but the average buffer contents has gone up by more than a factor of 40. In Table 20.3, the average arrival rate is increase slightly, to 99% of capacity, which yields an average buffer contents of 2583. Thus, a tiny increase in average arrival rate results in an increase of almost 40% in the average buffer contents.

This crude example suggests that the behavior of a system with a queue may not accord with our intuition.

20.2 WHY QUEUEING ANALYSIS?

There are many cases when it is important to be able to project the effect of some change in a design: Either the load on a system is expected to increase or a design change is contemplated. For example, an organization supports a number of terminals, personal computers, and workstations on a 100-Mbps local area network (LAN). An additional department in the building is to be cut over onto the network. Can the existing LAN handle the increased workload, or would it be better to provide a second LAN with a bridge between the two? There are other cases in which no facility exists

but, on the basis of expected demand, a system design needs to be created. For example, a department intends to equip all of its personnel with a personal computer and to configure these into a LAN with a file server. Based on experience elsewhere in the company, the load generated by each PC can be estimated.

The concern is system performance. In an interactive or real-time application, often the parameter of concern is response time. In other cases, throughput is the principal issue. In any case, projections of performance are to be made on the basis of existing load information or on the basis of estimated load for a new environment. A number of approaches are possible:

1. Do an after-the-fact analysis based on actual values.
2. Make a simple projection by scaling up from existing experience to the expected future environment.
3. Develop an analytic model based on queueing theory.
4. Program and run a simulation model.

Option 1 is no option at all: We will wait and see what happens. This leads to unhappy users and to unwise purchases. Option 2 sounds more promising. The analyst may take the position that it is impossible to project future demand with any degree of certainty. Therefore, it is pointless to attempt some exact modeling procedure. Rather, a rough-and-ready projection will provide ballpark estimates. The problem with this approach is that the behavior of most systems under a changing load is not what one would intuitively expect, as Section 20.1 suggests. If there is an environment in which there is a shared facility (e.g., a network, a transmission line, a time-sharing system), then the performance of that system typically responds in an exponential way to increases in demand.

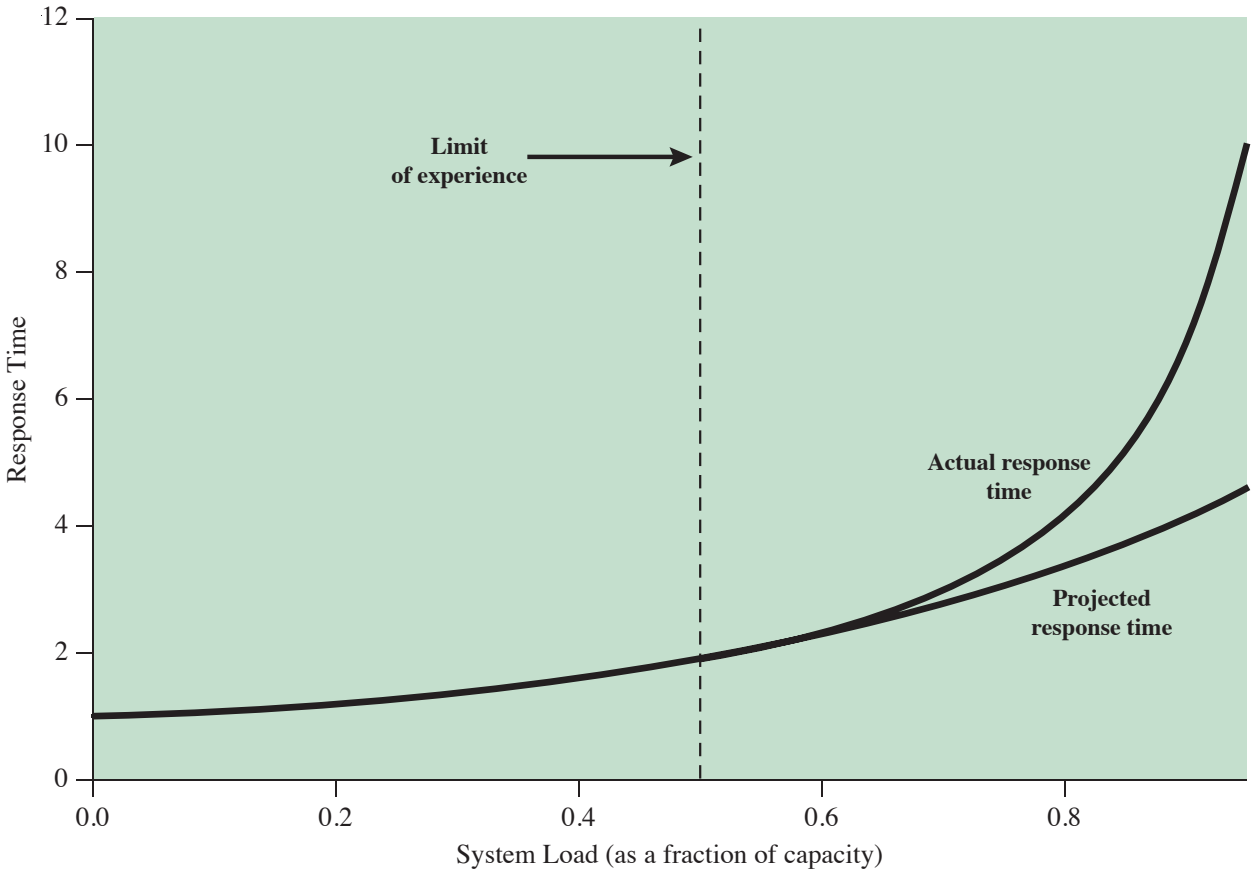


Figure 20.1 Projected Versus Actual Response Time

Figure 20.1 is a representative example. The upper line shows what typically happens to user response time on a shared facility as the load on that facility increases. The load is expressed as a fraction of capacity. Thus, if we are dealing with an input from a disk that is capable of transferring 1,000 blocks per second, then a load of 0.5 represents a transfer of 500 blocks per second, and the response time is the amount of time it takes to retransmit any incoming block. The lower line is a simple projection based on a knowledge of the behavior of the system up to a load of 0.5. Note that while things appear rosy when the simple projection is made, performance on the system will in fact collapse beyond a load of about 0.8—0.9.

Thus, a more exact prediction tool is needed. Option 3 is to make use of an analytic model, which is one that can be expressed as a set of equations

that can be solved to yield the desired parameters (response time, throughput, etc.). For computer, operating system, and networking problems, and indeed for many practical real-world problems, analytic models based on queueing theory provide a reasonably good fit to reality. The disadvantage of queueing theory is that a number of simplifying assumptions must be made to derive equations for the parameters of interest.

The final approach is a simulation model. Here, given a sufficiently powerful and flexible simulation programming language, the analyst can model reality in great detail and avoid making many of the assumptions required of queueing theory. However, in most cases, a simulation model is not needed or at least is not advisable as a first step in the analysis. For one thing, both existing measurements and projections of future load carry with them a certain margin of error. Thus, no matter how good the simulation model, the value of the results is limited by the quality of the input. For another, despite the many assumptions required of queueing theory, the results that are produced often come quite close to those that would be produced by a more careful simulation analysis. Furthermore, a queueing analysis can literally be accomplished in a matter of minutes for a well-defined problem, whereas simulation exercises can take days, weeks, or longer to program and run.

Accordingly, it behooves the analyst to master the basics of queueing theory.

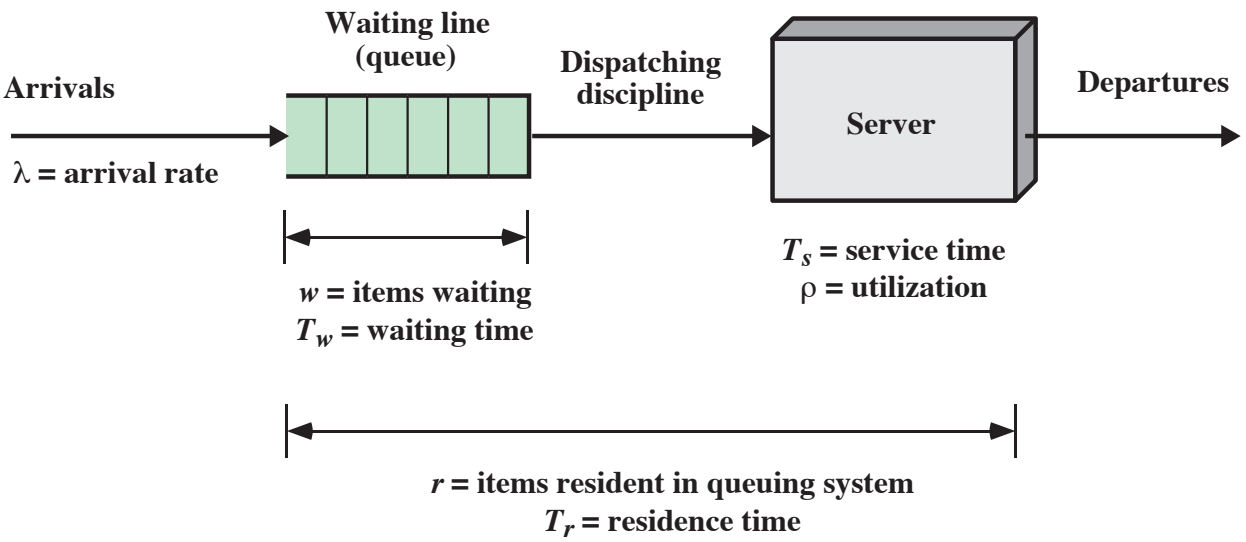


Figure 20.2 Queuing System Structure and Parameters for Single-Server Queue

20.3 QUEUEING MODELS

The Single-Server Queue

The simplest queueing system is depicted in Figure 20.2. The central element of the system is a server, which provides some service to items. Items from some population of items arrive at the system to be served. If the server is idle, an item is served immediately. Otherwise, an arriving item joins a waiting line.[2] When the server has completed serving an item, the item departs. If there are items waiting in the queue, one is immediately dispatched to the server. The server in this model can represent anything that performs some function or service for a collection of items. For, example, a processor provides service to processes; a transmission line provides a transmission service to packets or frames of data; and an I/O device provides a read or write service for I/O requests.

---

[2] The waiting line is referred to as a queue in some treatments in the literature; it is also common to refer to the entire system as a queue. Unless otherwise noted, we use the term *queue* to mean waiting line.

QUEUE PARAMETERS

Figure 20.2 also illustrates some important parameters associated with a queueing model. Items arrive at the facility at some average rate (items arriving per second) λ. Examples of items arriving include packets arriving at a router and calls arriving at a telephone exchange. At any given time, a certain number of items will be waiting in the waiting line (zero or more); the average number waiting is w, and the mean time that an item must wait is T_w. T_w is averaged over all incoming items, including those that do not wait at all. The server handles incoming items with an average service time T_s; this is the time interval between the dispatching of an item to the server and the departure of that item from the server. Utilization, ρ, is the fraction of time that the server is busy, measured over some interval of time. Finally, two parameters apply to the system as a whole. The average number of items resident in the system, including the item being served (if any) and the items waiting (if any), is r; and the average time that an item spends in the system, waiting and being served, is T_r; we refer to this as the *mean residence time*.[3]

If we assume that the capacity of the queue is infinite, then no items are ever lost from the system; they are just delayed until they can be served. Under these circumstances, the departure rate equals the arrival rate. As the arrival rate, which is the rate of traffic passing through the system, increases, the utilization increases and with it, congestion. The queue becomes longer, increasing waiting time. At $\rho = 1$, the server becomes saturated, working 100% of the time. So long as utilization is less than 100%, the server can keep up with arrivals, so that the average

---

[3] Again, in some of the literature, this is referred to as the mean queueing time, while other treatments use mean queueing time to mean the average time spent waiting in the waiting line (before being served).

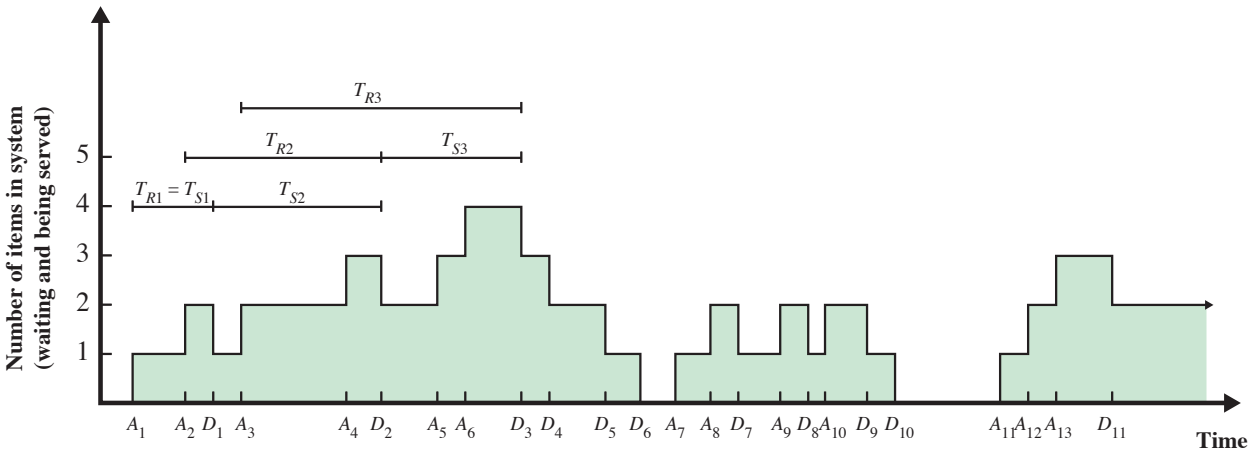
departure rate equals the average arrival rate. Once the server is saturated, working 100% of the time, the departure rate remains constant, no matter how great the arrival rate becomes. Thus, the theoretical maximum input rate that can be handled by the system is:

$$\lambda_{\max} = \frac{1}{T_s}$$

However, queues become very large near system saturation, growing without bound when $\rho = 1$. Practical considerations, such as response time requirements or buffer sizes, usually limit the input rate for a single server to 70–90% of the theoretical maximum.

ILLUSTRATION OF KEY FEATURES

It is helpful to have an illustration of the processes involved in queueing. Figure 20.3 shows an example realization of a queueing process, with the total number of items in the system plotted against time. The shaded areas represent time periods in which the server is busy. On the time axis are marked two types of events: the arrival of item j at time A_j and the completion of service of item j at time D_j, when the item departs the system. The time that item j spends in the system is $T_{Rj} = D_j - A_j$; the actual service time for item j is denoted by T_{Sj}.



For item i:

A_i = Arrival time
D_i = Departure time
T_{Ri} = Residence time
T_{Si} = Service time

Figure 20.3 Example of a Queuing Process

In this example, T_{R1} is composed entirely of the service time T_{S1} for the first item, because when item 1 arrives the system is empty and it can go straight into service. T_{R2} is composed of the time that item 2 waits for service ($D_1 - A_2$) plus its service time T_{S2}. Similarly, $T_{R3} = (D_3 - A_3) = (D_3 - D_2) + (D_2 - A_3) = T_{S3} + (D_2 - A_3)$. However, item n may depart before the arrival of item $n + 1$, (e.g., $D_6 < A_7$), so the general expression is $T_{Rn+1} = T_{Sn+1} + \text{MAX}[0, D_n - A_{n+1}]$.

MODEL CHARACTERISTICS

Before deriving any analytic equations for the queueing model, certain key characteristics of the model must be chosen. The following are the typical choices, usually reasonable in a data communications context:

- **Item population:** We assume that items arrive from a source population so large that it can be viewed as infinite. The effect of this assumption is that the arrival rate is not altered as items enter the

system. If the population is finite, then the population available for arrival is reduced by the number of items currently in the system; this would typically reduce the arrival rate proportionally. Networking and server problems can usually be handled with an infinite-population assumption.

- **Queue size:** We assume an infinite queue size. Thus, the queue can grow without bound. With a finite queue, items can be lost from the system; that is, if the queue is full and additional items arrive, some items must be discarded. In practice, any queue is finite, but in many cases, this makes no substantive difference to the analysis. We address this issue briefly later in this chapter.
- **Dispatching discipline:** When the server becomes free, and if there is more than one item waiting, a decision must be made as to which item to dispatch next. The simplest approach is first-in, first-out (FIFO), also known as first-come, first-served (FCFS); this discipline is what is normally implied when the term *queue* is used. Another possibility is last-in, first-out (LIFO). A common approach is a dispatching discipline based on relative priority. For example, a router may use QoS (quality of service) information to give preferential treatment to some packets. We discuss dispatching based on priority subsequently. One that you might encounter in practice is a dispatching discipline based on service time. For example, a process scheduler may choose to dispatch processes on the basis of shortest first (to allow the largest number of processes to be granted time in a short interval) or longest first (to minimize processing time relative to service time). Unfortunately, a discipline based on service time is very difficult to model analytically.

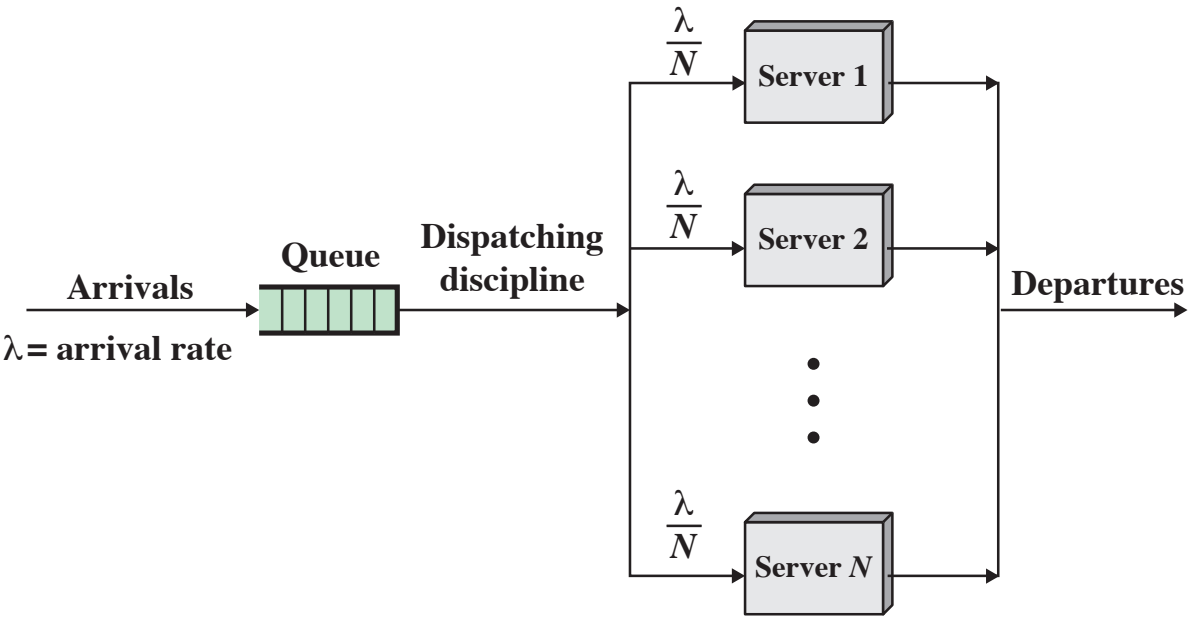
Table 20.4 Notation for Queueing Systems

λ	=	arrival rate; mean number of arrivals per second
T_s	=	mean service time for each arrival; amount of time being served, not counting time waiting in the queue
σ_{Ts}	=	standard deviation of service time
ρ	=	utilization; fraction of time facility (server or servers) is busy
u	=	traffic intensity
r	=	mean number of items in system, waiting and being served
R	=	number of items in system, waiting and being served
T_r	=	mean time an item spends in system (residence time)
T_R	=	time an item spends in system (residence time)
σ_r	=	standard deviation of r
σ_{Tr}	=	standard deviation of T_r
w	=	mean number of items waiting to be served
σ_w	=	standard deviation of w
T_w	=	mean waiting time (including items that have to wait and items with waiting time = 0)
T_d	=	mean waiting time for items that have to wait
N	=	number of servers
$m_X(y)$	=	the yth percentile; that value of y below which x occurs y percent of the time

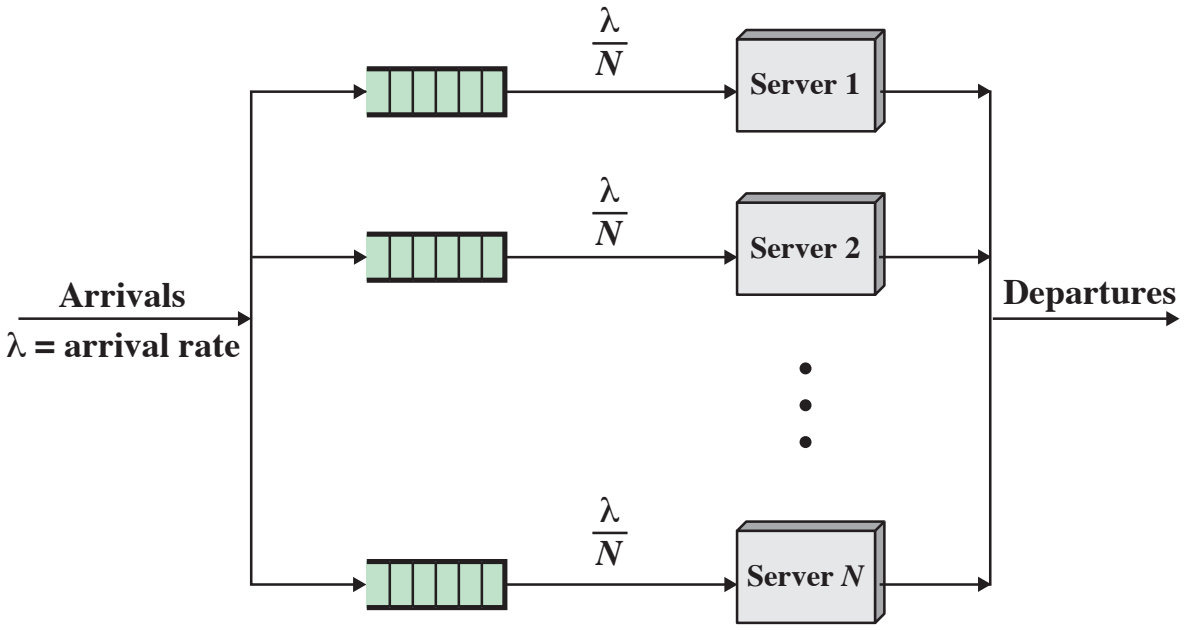
Table 20.4 summarizes the notation that is used in Figure 20.2 and introduces some other useful parameters. In particular, we are often interested in the variability of various parameters, and this is neatly captured in the standard deviation.

The Multiserver Queue

Figure 20.4a shows a generalization of the simple model we have been discussing for multiple servers, all sharing a common queue. If an item arrives and at least one server is available, then the item is immediately dispatched to that server. It is assumed that all servers are identical; thus, if more than one server is available, the selection of a particular server for a waiting item has no effect on service time. If all servers are busy, a queue



(a) Multiserver queue



(b) Multiple Single-server queues

Figure 20.4 Multiserver Versus Multiple Single-Server Queues

begins to form. As soon as one server becomes free, an item is dispatched from the queue using the dispatching discipline in force.

With the exception of utilization, all of the parameters illustrated in Figure 20.2 carry over to the multiserver case with the same interpretation. If we have N identical servers, then ρ is the utilization of each server, and we can consider $N\rho$ to be the utilization of the entire system; this latter term is often referred to as the *traffic intensity*, u. Thus, the theoretical maximum utilization is $N \times 100\%$, and the theoretical maximum input rate is:

$$\lambda_{\max} = \frac{N}{T_s}$$

The key characteristics typically chosen for the multiserver queue correspond to those for the single-server queue. That is, we assume an infinite population and an infinite queue size, with a single infinite queue shared among all servers. Unless otherwise stated in this discussion, the dispatching discipline is FIFO. For the multiserver case, if all servers are assumed identical, the selection of a particular server for a waiting item has no effect on service time.

By way of contrast, Figure 20.4b shows the structure of multiple single-server queues. As we shall see, this apparently minor change in structure has significant impact on performance.

Basic Queueing Relationships

To proceed much further, we are going to have to make some simplifying assumptions. These assumptions risk making the models less valid for various real-world situations. Fortunately, in most cases, the results will be sufficiently accurate for planning and design purposes.

Table 20.5 Some Basic Queueing Relationships

General	Single Server	Multiserver
$r = \lambda T_r$ Little's formula $w = \lambda T_w$ Little's formula $T_r = T_w + T_s$	$\rho = \lambda T_s$ $r = w + \rho$	$\rho = \frac{\lambda T_s}{N}$ $u = \lambda T_s = \rho N$ $r = w + N\rho$

There are, however, some relationships that are true in the general case, and these are illustrated in Table 20.5. By themselves, these relationships are not particularly helpful, although they can be used to answer a few basic questions. For example, consider a spy from Burger King trying to figure out how many people are inside the McDonald's across the way. He can't sit inside the McDonald's all day, so he has to determine an answer just based on observing the traffic in and out of the building. Over the course of the day, he observes that on average 32 customers per hour go into the restaurant. He notes certain people and finds that on average a customer stays inside 12 minutes. Using Little's formula, the spy deduces that there are on average 6.4 customers in McDonald's at any given time (6.4 = 32 customers/hour $\times$ 0.2 hours/customer).

It would be useful at this point to gain an intuitive grasp of the equations in Table 20.5. For the equation $\rho = \lambda T_s$, consider that for an arrival rate of λ the average time between arrivals is $1/\lambda = T$. If T is greater than T_s, then during a time interval T, the server is only busy for a time T_s for a utilization of $T_s/T = \lambda T_s$. Similar reasoning applies in the multiserver case to yield $\rho = (\lambda T_s)/N$.

To understand Little's formula, consider the following argument, which focuses on the experience of a single item. When the item arrives, it will find on average w items waiting ahead of it. When the item leaves the queue behind it to be serviced, it will leave behind on average the same number of items in the queue, namely w. To see this, note that while the item is

waiting, the line in front of it shrinks until the item is at the front of the line; meanwhile, additional items arrive and get in line behind this item. When the item leaves the queue to be serviced, the number of items behind it, on average, is w, because w is defined as the average number of items waiting. Further, the average time that the item was waiting for service is T_w. Since items arrive at a rate of λ, we can reason that in the time T_w, a total of λT_w items must have arrived. Thus $w = \lambda T_w$. Similar reasoning can be applied to the relationship $r = \lambda T_r$.

Turning to the last equation in the first column of Table 20.5, it is easy to observe that the time that an item spends in the system is the sum of the time waiting for service plus the time being served. Thus, on average, $T_r = T_w + T_s$. The last equations in the second and third columns are easily justified. At any time, the number of items in the system is the sum of the number of items waiting for service plus the number of items being served. For a single server, the average number of items being served is ρ. Therefore, $r = w + \rho$ for a single server. Similarly, $r = w + N\rho$ for N servers.

Assumptions

The fundamental task of a queueing analysis is as follows: Given the following information as input,

- Arrival rate
- Service time
- Number of servers

provide as output information concerning:

- Items waiting
- Waiting time
- Items in residence
- Residence time

What specifically would we like to know about these outputs? Certainly we would like to know their average values (w, T_w, r, T_r). In addition, it would be useful to know something about their variability. Thus, the standard deviation of each would be useful $\left(\sigma_r, \sigma_{T_r}, \sigma_w, \sigma_{T_w}\right)$. Other measures may also be useful. For example, to design a buffer associated with a router or multiplexer, it might be useful to know for what buffer size the probability of overflow is less than 0.001. That is, what is the value of N such that Pr[items waiting < N] = 0.999?

To answer such questions in general requires complete knowledge of the probability distribution of the interarrival times (time between successive arrivals) and service time. Furthermore, even with that knowledge, the resulting formulas are exceedingly complex. Thus, to make the problem tractable, we need to make some simplifying assumptions.

The most important of these assumptions concerns the arrival rate. We assume that the interarrival times are exponential, which is equivalent to saying that the number of arrivals in a period t obeys the Poisson distribution, which is equivalent to saying that the arrivals occur randomly and independent of one another. This assumption is almost invariably made. Without it, most queueing analysis is impractical, or at least quite difficult. With this assumption, it turns out that many useful results can be obtained if only the mean and standard deviation of the arrival rate and service time are known. Matters can be made even simpler and more detailed results can be obtained if it is assumed that the service time is exponential or constant.

A convenient notation, called **Kendall's notation**, has been developed for summarizing the principal assumptions that are made in developing a queueing model. The notation is X/Y/N, where X refers to the distribution of the interarrival times, Y refers to the distribution of service times, and N refers to the number of servers. The most common distributions are denoted as follows:

G = general distribution of interarrival times or service times
GI = general distribution of interarrival times with the restriction that interarrival times are independent
M = negative exponential distribution
D = deterministic arrivals or fixed-length service.

Thus, M/M/1 refers to a single-server queueing model with Poisson arrivals (exponential interarrival times) and exponential service times.

Table 20.6 Formulas for Single-Server Queues

Assumptions:
1. Poisson arrival rate.
2. Dispatching discipline does not give preference to items based on service times.
3. Formulas for standard deviation assume first-in, first-out dispatching.
4. No items are discarded from the queue.

(a) General Service Times (M/G/1)

$$A = \frac{1}{2}\left\lfloor 1 + \left(\frac{\sigma_{T_s}}{T_s}\right)^2 \right\rfloor$$

$$r = \rho + \frac{\rho^2 A}{1-\rho}$$

$$w = \frac{\rho^2 A}{1-\rho}$$

$$T_r = T_s + \frac{\rho T_s A}{1-\rho}$$

$$T_w = \frac{\rho T_s A}{1-\rho}$$

(b) Exponential Service Times (M/M/1)

$$r = \frac{\rho}{1-\rho} \qquad w = \frac{\rho^2}{1-\rho}$$

$$T_r = \frac{T_s}{1-\rho} \qquad T_w = \frac{\rho T_s}{1-\rho}$$

$$\sigma_r = \frac{\sqrt{\rho}}{1-\rho} \qquad \sigma_{T_r} = \frac{T_s}{1-\rho}$$

$$\Pr[R = N] = (1-\rho)\rho^N$$

$$\Pr[R \le N] = \sum_{i=0}^{N} (1-\rho)\rho^i$$

$$\Pr[T_R \le T] = 1 - e^{-(1-\rho)t/T_s}$$

$$m_{T_r}(y) = T_r \times \ln\left(\frac{100}{100-y}\right)$$

$$m_{T_w}(y) = \frac{T_w}{\rho} \times \ln\left(\frac{100\rho}{100-y}\right)$$

(c) Constant Service Times (M/D/1)

$$r = \frac{\rho^2}{2(1-\rho)} + \rho$$

$$w = \frac{\rho^2}{2(1-\rho)}$$

$$T_r = \frac{T_s(2-\rho)}{2(1-\rho)}$$

$$T_w = \frac{\rho T_s}{2(1-\rho)}$$

$$\sigma_r = \frac{1}{1-\rho}\sqrt{\rho - \frac{3\rho^2}{2} + \frac{5\rho^3}{6} - \frac{\rho^4}{12}}$$

$$\sigma_{T_r} = \frac{T_s}{1-\rho}\sqrt{\frac{\rho}{3} - \frac{\rho^2}{12}}$$

20.4 SINGLE-SERVER QUEUES

Table 20.6a provides some equations for single-server queues that follow the M/G/1 model. That is, the arrival rate is Poisson and the service time is general. Making use of a scaling factor, *A*, the equations for some of the key output variables are straightforward. Note that the key factor in the scaling

parameter is the ratio of the standard deviation of service time to the mean. No other information about the service time is needed. Two special cases are of some interest. When the standard deviation is equal to the mean, the service time distribution is exponential (M/M/1). This is the simplest case, and the easiest one for calculating results. Table 20.6b shows the simplified versions of equations for the standard deviation of r and T_r, plus some other parameters of interest. The other interesting case is a standard deviation of service time equal to zero, that is, a constant service time (M/D/1). The corresponding equations are shown in Table 20.6c.

Figures 20.5 and 20.6 plot values of average queue size and residence time versus utilization for three values of σ_{T_s}/T_s. This latter quantity is known as the **coefficient of variation** and gives a normalized measure of variability. Note that the poorest performance is exhibited by the exponential service time, and the best by a constant service time. In many cases, one can consider the exponential service time to be a worst case, so that an analysis based on this assumption will give conservative results. This is nice, because tables are available for the M/M/1 case and values can be looked up quickly.

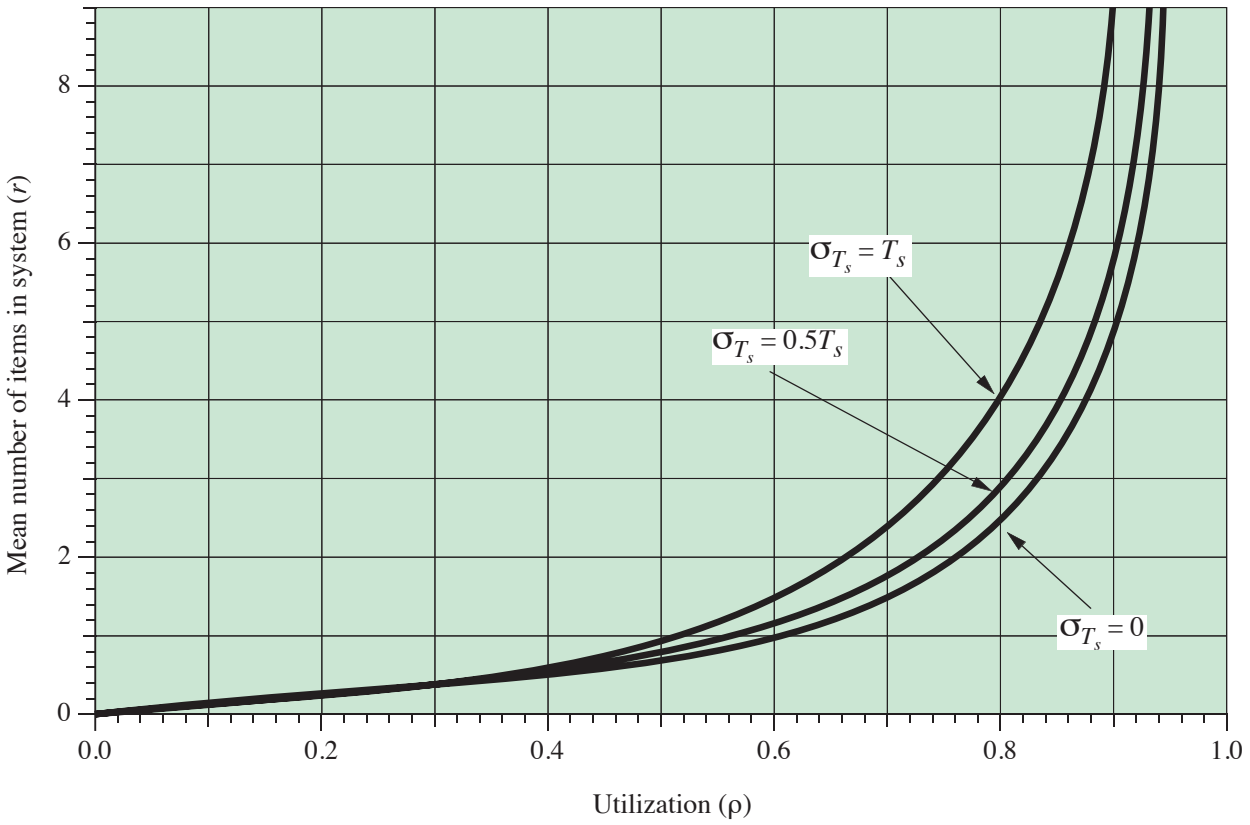


Figure 20.5 Mean Number of Items in System for Single-Server Queue

What value of σ_{T_s}/T_s is one likely to encounter? We can consider four regions:

- **Zero:** This is the rare case of constant service time. For example, if all transmitted packets are of the same length, they would fit this category.
- **Ratio less than 1:** Because this ratio is better than the exponential case, using M/M/1 tables will give queue sizes and times that are slightly larger than they should be. Using the M/M/1 model would give answers on the safe side. An example of this category might be a data entry application for a particular form.

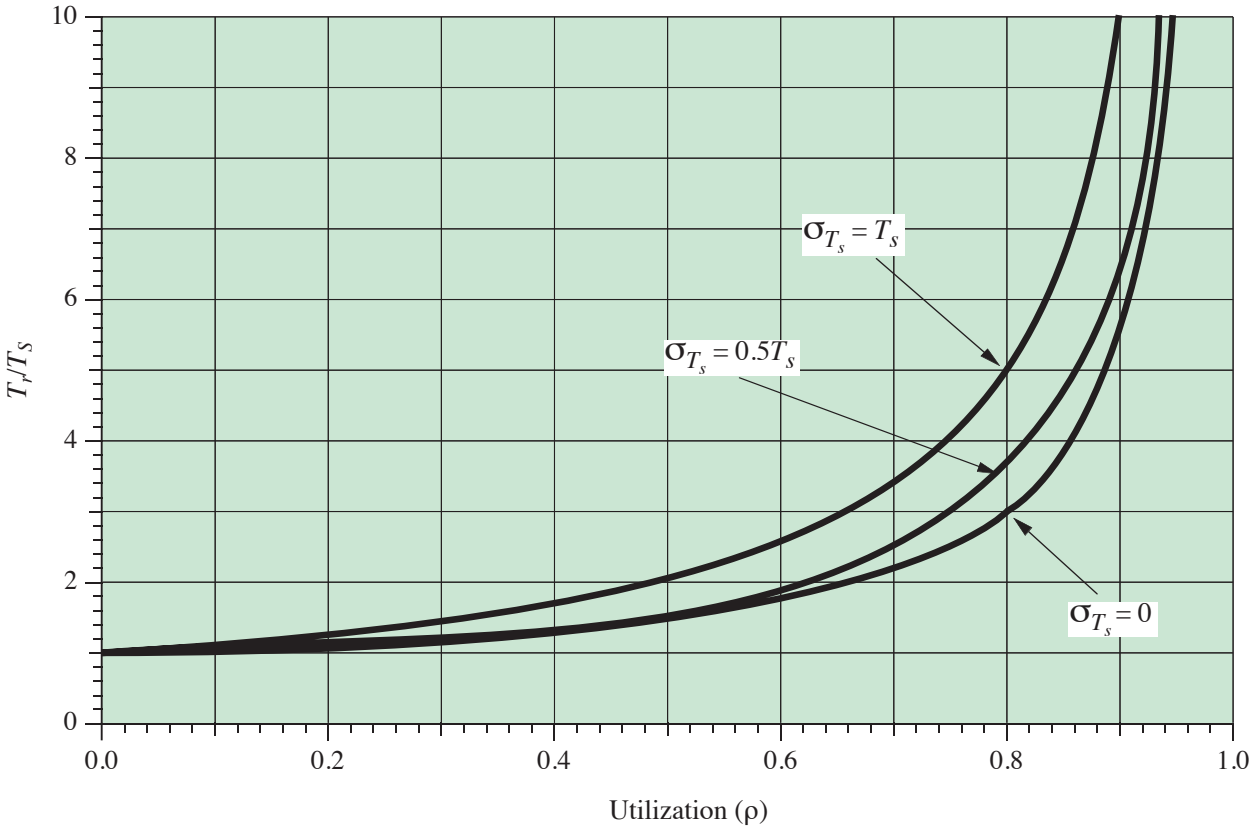


Figure 20.6 Mean Residence Time for Single-Server Queue

- **Ratio close to 1:** This is a common occurrence and corresponds to exponential service time. That is, service times are essentially random. Consider message lengths to a computer terminal: A full screen might be 1920 characters, with message sizes varying over the full range. Airline reservations, file lookups on inquires, shared LAN, and packet-switching networks are examples of systems that often fit this category.
- **Ratio greater then 1:** If you observe this, you need to use the M/G/1 model and not rely on the M/M/1 model. A common occurrence of this is a bimodal distribution, with a wide spread between the peaks. An example is a system that experiences many short messages, many long messages, and few in between.

The same consideration applies to the arrival rate. For a Poisson arrival rate, the interarrival times are exponential, and the ratio of standard deviation to mean is 1. If the observed ratio is much less than one, then arrivals tend to be evenly spaced (not much variability), and the Poisson assumption will overestimate queue sizes and delays. On the other hand, if the ratio is greater than 1, then arrivals tend to cluster and congestion becomes more acute.

20.5 MULTISERVER QUEUES

Table 20.7 lists formulas for some key parameters for the multiserver case. Note the restrictiveness of the assumptions. Useful congestion statistics for this model have been obtained only for the case of M/M/N, where the exponential service times are identical for the N servers.

Note the presence of the Erlang-C function in nearly all of the equations. This is the probability that all servers are busy at a given instant; equivalently, this is the probability that the number of items in the system (waiting and being served) is greater than or equal to the number of servers. The equation has the form

$$C(N,\rho)=\frac{1-K(N,\rho)}{1-\rho K(N,\rho)}$$

where *K* is known as the Poisson ratio function. Because *C* is a probability, its value is always between zero and one. As can be seen, this quantity is a function of the number of servers and the utilization. This expression turns up frequently in queueing calculations. Tables of values are readily found, or a computer program must be used. Note that for a single-server system, this equation simplifies to $C(1, \rho) = \rho$.

Table 20.7 Formulas for Multiserver Queues (M/M/N)

Assumptions:
1. Poisson arrival rate.
2. Exponential service times
3. All servers equally loaded
4. All servers have same mean service time
5. First-in, first-out dispatching
6. No items are discarded from the queue

$$K = \frac{\sum_{I=0}^{N-1} \frac{(N\rho)^I}{I!}}{\sum_{I=0}^{N} \frac{(N\rho)^I}{I!}} \qquad \text{Poisson ratio function}$$

$$\text{Erlang-C function} = \text{Probability that all servers are busy} = C = \frac{1-K}{1-\rho K}$$

$$r = C\frac{\rho}{1-\rho} + N\rho \qquad w = C\frac{\rho}{1-\rho}$$

$$T_r = \left(\frac{C}{N}\right)\frac{T_s}{1-\rho} + T_s \qquad T_w = \left(\frac{C}{N}\right)\frac{T_s}{1-\rho}$$

$$\sigma_{T_r} = \frac{T_s}{N(1-\rho)}\sqrt{C(2-C) + N^2(1-\rho)^2}$$

$$\sigma_w = \frac{1}{1-\rho}\sqrt{C\rho(1+\rho-C\rho)}$$

$$\Pr[T_W > t] = Ce^{-N(1-\rho)t/T_s}$$

$$m_{T_w}(y) = \frac{T_s}{N(1-\rho)}\ln\left(\frac{100C}{100-y}\right)$$

$$T_d = \frac{T_s}{N(1-\rho)}$$

20.6 EXAMPLES

Let us look at a few examples to get some feel for the use of these equations.

Database Server

Consider a LAN with 100 personal computers and a server that maintains a common database for a query application. The average time for the server to respond to a query is 0.6 seconds, and the standard deviation is estimated to equal the mean. At peak times, the query rate over the LAN reaches 20 queries per minute. We would like to answer the following questions:

- What is the average response time ignoring line overhead?
- If a 1.5-second response time is considered the maximum acceptable, what percent growth in message load can occur before the maximum is reached?
- If 20% more utilization is experienced, will response time increase by more or less than 20%?

Assume an M/M/1 model, with the database server being the server in the model. We ignore the effect of the LAN, assuming that its contribution to the delay is negligible. Facility utilization is calculated as:

$$\rho = \lambda T_s$$

= (20 arrivals per minute)(0.6 seconds per transmission)/(60 s/min)

= 0.2

The first value, average response time, is easily calculated:

$$T_r = T_s/(1 - \rho)$$
$$= 0.6/(1 - 0.2) = 0.75 \text{ seconds}$$

The second value is more difficult to obtain. Indeed, as worded, there is no answer because there is a nonzero probability that some instances of response time will exceed 1.5 seconds for any value of utilization. Instead, let us say that we would like 90% of all responses to be less than 1.5 seconds. Then, we can use the equation from Table 20.6b:

$$m_{T_r}(y) = T_r \times \ln\left(100/(100 - y)\right)$$
$$m_{T_r}(90) = T_r \times \ln(10) = \frac{T_s}{1-\rho} \times 2.3 = 1.5 \text{ seconds}$$

We have $T_s = 0.6$. Solving for ρ yields $\rho = 0.08$. In fact, utilization would have to decline from 20% to 8% to put 1.5 seconds at the 90th percentile.

The third part of the question is to find the relationship between increases in load versus response time. Because a facility utilization of 0.2 is down in the flat part of the curve, response time will increase more slowly than utilization. In this case, if facility utilization increases from 20% to 40%, which is a 100% increase, the value of T_r goes from 0.75 seconds to 1.0 second, which is an increase of only 33%.

Calculating Percentiles

Consider a configuration in which packets are sent from computers on a LAN to systems on other networks. All of these packets must pass through a router that connects the LAN to a wide area network and hence to the outside world. Let us look at the traffic from the LAN through the router. Packets arrive with a mean arrival rate of five per second. The average

packet length is 144 octets, and it is assumed that packet length is exponentially distributed. Line speed from the router to the wide area network is 9,600 bps. The following questions are asked:

1. What is the mean residence time for the router?
2. How many packets are in the router, including those waiting for transmission and the one currently being transmitted (if any), on the average?
3. Same question as (2), for the 90th percentile.
4. Same question as (2), for the 95th percentile.

λ = 5 packets/s

T_s = (144 octets $\times$ 8 bits/octet)/9600 bps = 0.12 s

ρ = $\lambda T_s = 5 \times 0.12 = 0.6$

T_r = $T_s/(1 - \rho) = 0.3$ s Mean residence time

r = $\rho/(1 - \rho) = 1.5$ packets Mean number of resident items

To obtain the percentiles, we use the equation from Table 20.6b:

$$\Pr[R = N] = (1 - \rho)\rho^N$$

To calculate the yth percentile of queue size, we write the preceding equation in cumulative form:

$$\frac{y}{100} = \sum_{k=0}^{m_r(y)} (1-\rho)\rho^k = 1 - \rho^{1+m_r(y)}$$

Here $m_r(y)$ represents the maximum number of packets in the queue expected y percent of the time. That is, $m_r(y)$ is that value below which R

occurs y percent of the time. In the form given, we can determine the percentile for any queue size. We wish to do the reverse: Given y, find $m_r(y)$. So, taking the logarithm of both sides:

$$m_r(y) = \frac{\ln\left(1 - \frac{y}{100}\right)}{\ln \rho} - 1$$

If $m_r(y)$ is fractional, take the next higher integer; if it is negative, set it to zero. For our example, $\rho = 0.6$ and we wish to find $m_r(90)$ and $m_r(95)$:

$$m_r(90) = \frac{\ln(1-0.90)}{\ln(0.6)} - 1 = 3.5$$
$$m_r(95) = \frac{\ln(1-0.95)}{\ln(0.6)} - 1 = 4.8$$

Thus, 90% of the time there are fewer than four packets in the queue, and 95% of the time there are fewer than five packets. If we were designing to a 95th percentile criterion, a buffer would have to be provided to store at least five packets.

Tightly-Coupled Multiprocessor

Let us consider the use of multiple tightly-coupled processors in a single computer system. One of the design decisions had to do with whether processes are dedicated to processors. If a process is permanently assigned to one processor from activation until its completion, then a separate short-term queue is kept for each processor. In this case, one processor can be idle, with an empty queue, while another processor has a backlog. To prevent this situation, a common queue can be used. All processes go into

one queue and are scheduled to any available processor. Thus, over the life of a process, the process may be executed on different processors at different times.

Let us try to get a feel for the performance speed-up to be achieved by using a common queue. Consider a system with five processors and that the average amount of processor time provided to a process while in the Running state is 0.1 sec. Assume that the standard deviation of service time is observed to be 0.094 sec. Because the standard deviation is close to the mean, we will assume exponential service time. Also assume that processes are arriving at the Ready state at the rate of 40 per second.

SINGLE-SERVER APPROACH

If processes are evenly distributed among the processors, then the load for each processor is 40/5 = 8 processes per second. Thus,

$$\begin{aligned}\rho &= \lambda T_S \\ &= 8 \times 0.1 = 0.8\end{aligned}$$

The residence time is then easily calculated:

$$t_r = \frac{T_s}{1-\rho} = \frac{0.1}{0.2} = 0.5 \text{ sec}$$

MULTISERVER APPROACH

Now assume that a single Ready queue is maintained for all processors. We now have an aggregate arrival rate of 40 processes per second. However, the facility utilization is still 0.8 ($\lambda T_S/M$). To calculate the residence time from the formula in Table 20.7, we need to first calculate the Erlang C function. If you have not programmed the parameter, it can be looked up in a table under a facility utilization of 0.8 for five servers to yield $C = 0.554$. Substituting,

$$t_r = (0.1) + \frac{(0.544)(0.1)}{5(1-0.8)} = 0.1544$$

So the use of multiserver queue has reduced average residence time from 0.5 s down to 0.1544 s, which is greater than a factor of 3. If we look at just the waiting time, the multiserver case is 0.0544 seconds compared to 0.4 seconds, which is a factor of 7.

Although you may not be an expert in queueing theory, you now know enough to be annoyed when you have to wait in a line at a multiple single-server queue facility.

A Multiserver Problem

An engineering firm provides each of its analysts with a personal computer, all of which are hooked up over a LAN to a database server. In addition, there is an expensive, stand-alone graphics workstation that is used for special-purpose design tasks. During the course of a typical eight-hour day, 10 engineers will make use of the workstation and spend an average of 30 minutes at a session.

Single-Server Model

The engineers complain to their manager that the wait for using the workstation is long, often an hour or more, and are asking for more workstations. This surprises the manager since the utilization of the workstation is only 5/8 ($10 \times 1/2 = 5$ hours out of 8). To convince the manager, one of the engineers performs a queueing analysis. The engineer makes the usual assumptions of an infinite population, random arrivals, and exponential service times, none of which seem unreasonable for rough calculations. Using the equations in Tables 20.5 and 20.6b, the engineer gets:

$$T_w = \frac{\rho T_s}{1-\rho} = 50 \text{ minutes}$$

Average time an engineer spends waiting for the

	workstation
$m_{T_w}(90) = \frac{T_w}{\rho} \times \ln(10\rho) = 146.6 \text{ minutes}$	90th percentile waiting time
$\lambda = \frac{10}{8 \times 60} = 0.021 \text{ engineers / minute}$	Arrival rate of engineers
$w = \lambda T_w = 1.0416$ engineers	Average number of engineers waiting

These figures show that indeed the engineers do have to wait an average of almost an hour to use the workstation and that in 10% of the cases, an engineer has to wait well over two hours. Even if there is a significant error in the estimate, say 20%, the waiting time is still far too long. Furthermore, if an engineer can do no useful work while waiting for the workstation, then a little over one engineer-day is being lost per day.

MULTISERVER MODEL

The engineers have convinced the manager of the need for more workstations. They would like the mean waiting time not to exceed 10 minutes, with the 90th percentile value not to exceed 15 minutes. This concerns the manager, who reasons that if one workstation results in a waiting time of 50 minutes, then five workstations will be required to get the average down to 10 minutes.

The engineers set to work to determine how many workstations are required. There are two possibilities: Put additional workstations in the same room as the original one (multiserver queue) or scatter the workstations to various rooms on various floors (multiple single-server queues). First, we look at the multiserver case and consider the addition of a second workstation in the same room. Let's assume that the addition of the new workstation, which reduces waiting time, does not affect the arrival rate (10 engineers per day). Then the available service time is 16 hours in an eight-

hour day with a demand of five hours (10 engineers $\times$ 0.5 hours), giving a utilization of 5/16 = 0.3125. Using the equations in Table 20.7:

$C(2, \rho) = C(2, 0.3125) = 0.1488$	Probability that both servers are busy
$T_w = \frac{CT_s}{N(1-\rho)} = 3.247 \text{ minutes}$	Average time an engineer spends waiting for a workstation
$m_{T_w}(90) = \frac{T_s}{2(1-\rho)}\ln(10C) = 8.67 \text{ minutes}$	90th percentile waiting time
$w = \lambda T_w = 0.07$ engineers	Average number of engineers waiting

With this arrangement, the probability that an engineer who wishes to use a workstation must wait is less than 0.15 and the average wait is just a little over three minutes, with the 90th-percentile wait of under nine minutes. Despite the manager's doubts, the multiserver arrangement with two workstations easily meets the design requirement.

All of the engineers are housed on two floors of the building, so the manager wonders whether it might be more convenient to place one workstation on each floor. If we assume that the traffic to the two workstations is about evenly split, then there are two M/M/1 queues, each with a λ of five engineers per eight-hour day. This yields:

$\rho = \lambda T_s = 0.3125$	Utilization of one server
$T_w = \frac{\rho T_s}{1-\rho} = 13.64 \text{ minutes}$	Average time an engineer spends waiting for the workstation
$m_{T_w}(90) = \frac{T_w}{\rho} \times \ln(10\rho) = 49.73 \text{ minutes}$	90th percentile waiting time
$w = \lambda T_w = 0.142$ engineers	Average number of engineers waiting

This performance is significantly worse than the multiserver model and does not meet the design criteria. Table 20.8 summarizes the results and also shows the results for four and five separate workstations. Note that to meet the design goal, five separate workstations are needed compared to only two multiserver workstations.

Table 20.8 Summary of Calculations for Multiserver Example

Workstations	System	ρ	T_w	$m_{Tw}(90)$
1	M/M/1	0.625	50	146.61
2	M/M/2	0.3125	3.25	8.67
2	M/M/1's	0.3125	13.64	49.73
4	M/M/1's	0.15625	5.56	15.87
5	M/M/1's	0.125	4.29	7.65

20.7 QUEUES WITH PRIORITIES

So far we have considered queues in which items are treated in a first-come, first-served basis. There are many cases in both networking and operating system design in which it is desirable to use priorities. Priorities may be assigned in a variety of ways. For example, priorities may be assigned on the basis of traffic type. If it turns out that the average service time for the various traffic types is identical, then the overall equations for the system are not changed, although the performance seen by the different traffic classes will differ.

Table 20.9 Formulas for Single-Server Queues with Two Priority Categories

Assumptions:
1. Poisson arrival rate.
2. Priority 1 items are serviced before priority 2 items.
3. First-in, first-out dispatching for items of equal priority.
4. No item is interrupted while being served.
5. No items leave the queue (lost calls delayed).

(a) General Formulas

$\lambda = \lambda_1 + \lambda_2$

$\rho_1 = \lambda_1 T_{s1}; \quad \rho_2 = \lambda_2 T_{s2}$

$\rho = \rho_1 + \rho_2$

$$T_s = \frac{\lambda_1}{\lambda} T_{s1} + \frac{\lambda_2}{\lambda} T_{s2}$$

$$T_r = \frac{\lambda_1}{\lambda} T_{r1} + \frac{\lambda_2}{\lambda} T_{r2}$$

(b) Exponential Service Times

$$w_1 = \frac{\rho_1(\rho_1 T_{s1} + \rho_2 T_{s2})}{T_{s1}(1-\rho_1)}$$

$$w_2 = w_1 \frac{\lambda_2}{\lambda_1(1-\rho)}$$

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1} + \rho_2 T_{s2}}{1-\rho_1}$$

$$T_{r2} = T_{s2} + \frac{T_{r1} - T_{s1}}{1-\rho}$$

An important case is one in which priority is assigned on the basis of average service time. Often, items with shorter expected service times are given priority over items with longer service times. For example, a router may assign a higher priority to a stream of voice packets than a stream of data packets, and typically, the voice packets would be much shorter than the data packets. With this kind of scheme, performance is improved for higher-priority traffic.

Table 20.9 shows the formulas that apply when we assume two priority classes with different service times for each class. These results are easily generalized to any number of priority classes.

To see the effects of the use of priority, let us consider a simple example of a data stream consisting of a mixture of long and short packets

being transmitted by a packet-switching node and that the rate of arrival of the two types of packets is equal. Suppose that both packets have lengths that are exponentially distributed and that the long packets have a mean packet length of 10 times the short packets. In particular, let us assume a 64-kbps transmission link and that the mean packet lengths are 80 and 800 octets. Then the two service times are 0.01 and 0.1 seconds. Also assume that the arrival rate for each type is 8 packets per second. So that the shorter packets are not held up by the longer packets, let us assign the shorter packets a higher priority. Then:

$$\rho_1 = 8 \times 0.01 = 0.08 \qquad \rho_2 = 8 \times 0.1 = 0.8 \qquad \rho = 0.88$$

$$T_{r1} = 0.01 + \frac{0.08 \times 0.01 + 0.8 \times 0.1}{1 - 0.08} = 0.098 \text{ seconds}$$

$$T_{r2} = 0.1 + \frac{0.098 - 0.01}{1 - 0.88} = 0.833 \text{ seconds}$$

$$T_r = 0.5 \times 0.098 + 0.5 \times 0.833 = 0.4655 \text{ seconds}$$

So we see that the higher-priority packets get considerably better service than the lower-priority packets.

20.8 NETWORKS OF QUEUES

In a distributed environment, isolated queues are unfortunately not the only problem presented to the analyst. Often, the problem to be analyzed consists of several interconnected queues. Figure 20.7 illustrates this situation, using nodes to represent queues and the interconnecting lines to represent traffic flow.

Two elements of such a network complicate the methods shown so far:

- The partitioning and merging of traffic, as illustrated by nodes 1 and 5, respectively, in the figure
- The existence of queues in tandem, or series, as illustrated by nodes 3 and 4

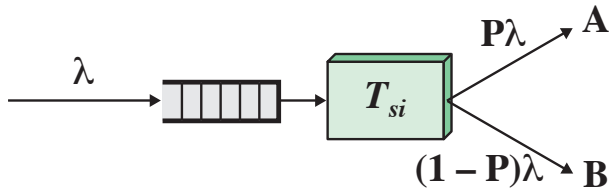
No exact method has been developed for analyzing general queueing problems that have the aforementioned elements. However, if the traffic flow is Poisson and the service times are exponential, an exact and simple solution exists. In this section, we first examine the two elements listed previously, and then present the approach to queueing analysis.

Partitioning and Merging of Traffic Streams

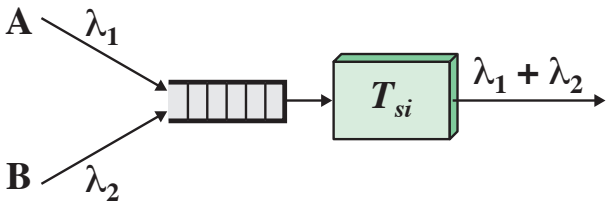
Suppose that traffic arrives at a queue with a mean arrival rate of λ, and that there are two paths, A and B, by which an item may depart (Figure 20.8a). When an item is serviced and departs the queue, it does so via path A with probability P and via path B with probability $(1 - P)$. In general, the traffic distribution of streams A and B will differ from the incoming distribution. However, if the incoming distribution is Poisson, then the two departing traffic flows also have Poisson distributions, with mean rates of $P\lambda$ and $(1 - P)\lambda$.

A similar situation exists for traffic merging (Figure 20.8b). If two Poisson streams with mean rates of λ_1 and λ_2 are merged, the resulting stream is Poisson with a mean rate of $\lambda_1 + \lambda_2$.

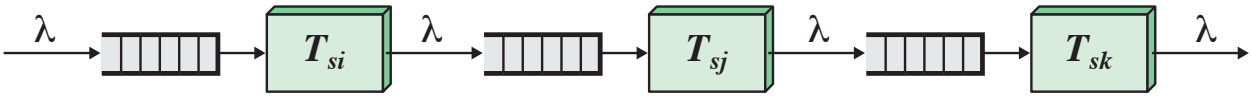
Both of these results generalize to more than two departing streams for partitioning and more than two arriving streams for merging.



(a) Traffic partitioning



(b) Traffic merging



(c) Simple tandem queue

Figure 20.8 Elements of Queuing Networks

Queues in Tandem

Figure 20.8c is an example of a set of single-server queues in tandem: The input for each queue except the first is the output of the previous queue. Assume that the input to the first queue is Poisson. Then, if the service time of each queue is exponential and the queues are of infinite capacity, the output of each queue is a Poisson stream statistically identical to the input. When this stream is fed into the next queue, the delays at the second queue are the same as if the original traffic had bypassed the first queue and fed directly into the second queue. Thus, the queues are independent and may be analyzed one at a time. Therefore, the mean total delay for the tandem system is equal to the sum of the mean delays at each stage.

This result can be extended to the case where some or all of the nodes in tandem are multiserver queues.

Jackson's Theorem

Jackson's theorem can be used to analyze a network of queues. The theorem is based on three assumptions:

1. The queueing network consists of *m* nodes, each of which provides an independent exponential service.
2. Items arriving from outside the system to any one of the nodes arrive with a Poisson rate.
3. Once served at a node, an item goes (immediately) to one of the other nodes with a fixed probability, or out of the system.

Jackson's theorem states that in such a network of queues, each node is an independent queueing system, with a Poisson input determined by the principles of partitioning, merging, and tandem queueing. Thus, each node may be analyzed separately from the others using the M/M/1 or M/M/N model, and the results may be combined by ordinary statistical methods. Mean delays at each node may be added to derive system delays, but nothing can be said about the higher moments of system delays (e.g., standard deviation).

Jackson's theorem appears attractive for application to packet-switching networks. One can model the packet-switching network as a network of queues. Each packet represents an individual item. We assume that each packet is transmitted separately and, at each packet-switching node in the path from source to destination, the packet is queued for transmission on the next length. The service at a queue is the actual transmission of the packet and is proportional to the length of the packet.

The flaw in this approach is that a condition of the theorem is violated: namely, it is not the case that the service distributions are independent. Because the length of a packet is the same at each transmission link, the arrival process to each queue is correlated to the service process. However, Kleinrock [KLEI76] has demonstrated that, because of the averaging effect of merging and partitioning, assuming independent service times provides a good approximation.

Application to a Packet-Switching Network[4]

Consider a packet-switching network, consisting of nodes interconnected by transmission links, with each node acting as the interface for zero or more attached systems, each of which functions as a source and destination of traffic. The external workload that is offered to the network can be characterized as:

$$\gamma = \sum_{j=1}^{N} \sum_{k=1}^{N} \gamma_{jk}$$

where

γ = total workload in packets per second

γ_{jk} = workload between source j and destination k

N = total number of sources and destinations

Because a packet may traverse more than one link between source and destination, the total internal workload will be higher than the offered load:

$$\lambda = \sum_{i=1}^{L} \lambda_i$$

where

---

[4] This discussion is based on the development in [KLEI76].

λ = total load on all of the links in the network

λ_i = load on link i

L = total number of links

The internal load will depend on the actual path taken by packets through the network. We will assume a routing algorithm is given such that the load on the individual links, λ_i, can be determined from the offered load, γ_{jk}. For any particular routing assignment, we can determine the average number of links that a packet will traverse from these workload parameters. Some thought should convince you that the average length for all paths is given by:

$$\mathrm{E}[\text{number of links in a path}] = \frac{\lambda}{\gamma}$$

Now, our objective is to determine the average delay, T, experienced by a packet through the network. For this purpose, it is useful to apply Little's formula (Table 20.5). For each link in the network, the average number of items waiting and being served for that link is given by:

$$r_i = \lambda_i T_{ri}$$

where T_{ri} is the yet-to-be-determined queueing delay at each queue. Suppose that we sum these quantities. That would give us the average total number of packets waiting in all of the queues of the network. It turns out that Little's formula works in the aggregate as well.[5] Thus, the number of

---

[5] In essence, this statement is based on the fact that the sum of the averages is the average of the sums.

packets waiting and being served in the network can be expressed as γT. Combining the two:

$$T = \frac{1}{\gamma} \sum_{i=1}^{L} \lambda_i T_{ri}$$

To determine the value of T, we need to determine the values of the individual delays, T_{ri}. Because we are assuming that each queue can be treated as an independent M/M/1 model, this is easily determined:

$$T_{ri} = \frac{T_{si}}{1 - \rho_i} = \frac{T_{si}}{1 - \lambda_i T_{si}}$$

The service time T_{si} for link i is just the ratio of the average packet length in bits (M) to the data rate on the link in bits per second (R_i). Then:

$$T_{ri} = \frac{\frac{M}{R_i}}{1 - \frac{M\lambda_i}{R_i}} = \frac{M}{R_i - M\lambda_i}$$

Putting all of the elements together, we can calculate the average delay of packets sent through the network:

$$T = \frac{1}{\gamma} \sum_{i=1}^{L} \frac{M\lambda_i}{R_i - M\lambda_i}$$

20.9 OTHER QUEUEING MODELS

In this chapter, we have concentrated on one type of queueing model. There are in fact a number of models, based on two key factors:

- The manner in which blocked items are handled
- The number of traffic sources

When an item arrives at a server and finds that server busy, or arrives at a multiple-server facility and finds all servers busy, that item is said to be blocked. Blocked items can be handled in a number of ways. First, the item can be placed in a queue awaiting a free server. This policy is referred to in the telephone traffic literature as *lost calls delayed*, although in fact the call is not lost. Alternatively, no queue is provided. This in turn leads to two assumptions about the action of the item. The item may wait some random amount of time and then try again; this is known as *lost calls cleared*. If the item repeatedly attempts to gain service, with no pause, it is referred to as *lost calls held*. The lost calls delayed model is the most appropriate for most computer and data communications problems. Lost calls cleared is usually the most appropriate in a telephone-switching environment.

The second key element of a traffic model is whether the number of sources is assumed infinite or finite. For an infinite source model, there is assumed to be a fixed arrival rate. For the finite source case, the arrival rate will depend on the number of sources already engaged. Thus, if each of L sources generates arrivals at a rate λ/L, then when the queueing facility is unoccupied, the arrival rate is λ. However, if K sources are in the queueing facility at a particular time, then the instantaneous arrival rate at that time is $\lambda(L - K)/L$. Infinite source models are easier to deal with. The infinite source assumption is reasonable when the number of sources is at least 5–10 times the capacity of the system.

20.10 ESTIMATING MODEL PARAMETERS

To perform a queueing analysis, we need to estimate the values of the input parameters, specifically the mean and standard deviation of the arrival rate and service time. If we are contemplating a new system, these estimates may have to be based on judgment and an assessment of the equipment and work patterns likely to prevail. However, it will often be the case that an existing system is available for examination. For example, a collection of terminals, personal computers, and host computers are interconnected in a building by direct connection and multiplexers, and it is desired to replace the interconnection facility with a LAN. To be able to size the network, it is possible to measure the load currently generated by each device.

Sampling

The measurements that are taken are in the form of samples. A particular parameter, for example, the rate of packets generated by a terminal or the size of packets, is estimated by observing the number of packets generated during a period of time.

The most important quantity to estimate is the mean. For many of the equations in Tables 20.6 and 20.7, this is the only quantity that need be estimated. The estimate is referred to as the sample mean $\overline{X}$ and is calculated as follows:

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{N} X_i$$

where

N = sample size

X_i = *i*th item in the sample

It is important to note that the sample mean is itself a random variable. For example, if you take a sample from some population and calculate the sample mean, and do this a number of times, the calculated values will differ. Thus, we can talk of the mean and standard deviation of the sample mean, or even of the entire probability distribution of the sample mean. To distinguish the concepts, it is common to refer to the probability distribution of the original random variable *X* as the *underlying distribution* and the probability distribution of the sample mean $\overline{X}$ as the *sampling distribution of the mean*.

The remarkable thing about the sample mean is that its probability distribution tends to the normal distribution as *N* increases for virtually all underlying distributions. The assumption of normality breaks down only if *N* is very small or if the underlying distribution is highly abnormal.

The mean and variance of $\overline{X}$ are as follows:

$$\mathrm{E}\left[\overline{X}\right] = \mathrm{E}\left[X\right] = \mu$$

$$\mathrm{Var}\left[\overline{X}\right] = \frac{\sigma_X^2}{N}$$

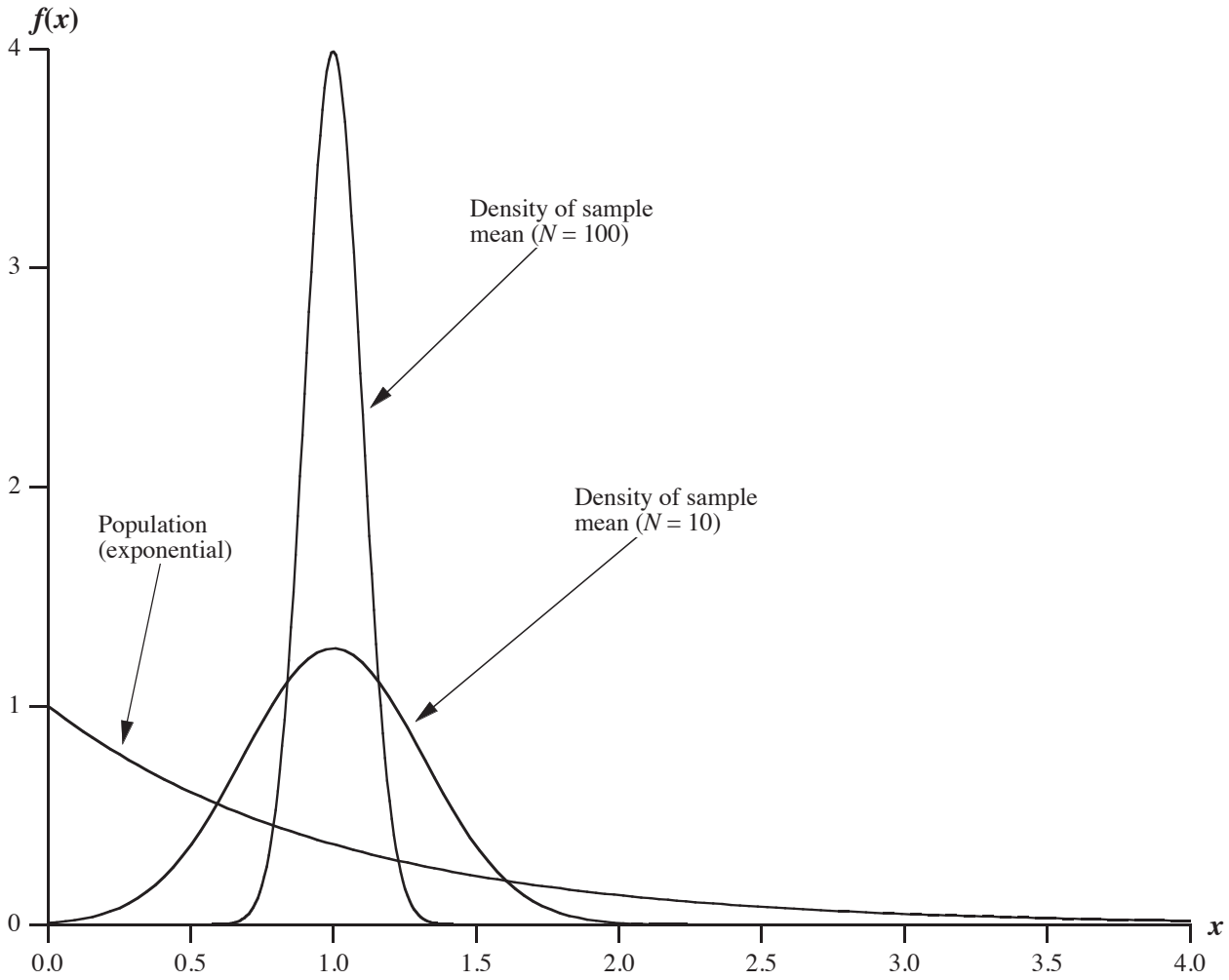


Figure 20.9 Sample Means for an Exponential Population

Thus, if a sample mean is calculated, its expected value is the same as that of the underlying random variable and the variability of the sample mean around this expected value decreases as *N* increases. These characteristics are illustrated in Figure 20.9. The figure shows an underlying exponential distribution with mean value $\mu = 1$. This could be the distribution of service times of a server or of the interarrival times of a Poisson arrival process. If a sample of size 10 is used to estimate the value of μ, then the expected value is indeed μ, but the actual value could easily be off by as much as 50%. If the sample size is 100, the spread among possible

calculated values is considerably tightened, so that we would expect the actual sample mean for any given sample to be much closer to μ.

The sample mean as defined previously can be used directly to estimate the service time of a server. For arrival rate, one can observe the interarrival times for a sequence of *N* arrivals, calculate the sample mean, and then calculate the estimated arrival rate. An equivalent and simpler approach is to use the following estimate:

$$\overline{\lambda} = \frac{N}{T}$$

where *N* is the number of items observed in a period of time of duration *T*.

For much of queueing analysis, it is only an estimate of the mean that is required. But for a few important equations, an estimate of the variance of the underlying random variable, σ_X^2, is also needed. The sample variance is calculated as follows:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2$$

The expected value of S^2 has the desired value:

$$\mathrm{E}\left[S^2\right] = \sigma_X^2$$

The variance of S^2 depends on the underlying distribution and is, in general, difficult to calculate. However, as you would expect, the variance of S^2 decreases as *N* increases.

Table 20.10 summarizes the concepts discussed in this section.

Table 20.10 Statistical Parameters

	Population	Sample Mean	Sample Variance
Random variable	X	$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$	$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2$
Expected value	$\mathrm{E}[X] = \mu$	$\mathrm{E}\left[\overline{X}\right] = \mu$	$\mathrm{E}\left[S^2\right] = \sigma_X^2$
Variance	$\mathrm{Var}[X] = \mathrm{E}\left[(X-\mu)^2\right] = \sigma_X^2$	$\mathrm{Var}\left[\overline{X}\right] = \frac{\sigma_X^2}{N}$	

Sampling Errors

When we estimate values such as the mean and standard deviation on the basis of a sample, we leave the realm of probability and enter that of statistics. This is a complex topic that will not be explored here, except to provide a few comments.

The probabilistic nature of our estimated values is a source of error, known as **sampling error**. In general, the greater the size of the sample taken, the smaller the standard deviation of the sample mean or other quantity, and therefore the closer that our estimate is likely to be to the actual value. By making certain reasonable assumptions about the nature of the random variable being tested and the randomness of the sampling procedure, one can in fact determine the probability that a sample mean or sample standard deviation is within a certain distance from the actual mean or standard deviation. This concept is often reported with the results of a sample. For example, it is common for the result of an opinion poll to include a comment such as, "The result is within 5% of the true value with a confidence (probability) of 99%."

There is, however, another source of error, which is less widely appreciated among nonstatisticians: **bias**. For example, if an opinion poll is conducted and only members of a certain socioeconomic group are interviewed, the results are not necessarily representative of the entire

population. In a communications context, sampling done during one time of day may not reflect the activity at another time of day. If we are concerned with designing a system that will handle the peak load that is likely to be experienced, then we should observe the traffic during the time of day that is most likely to produce the greatest load.

20.11 RECOMMENDED READING

A good practical reference on queueing analysis is [GUNT00].

For those who wish to delve more deeply into queueing theory, a host of books is available. A good text, and one that is quite readable, that covers queueing theory and its application to computers and communications is [GROS09]. The classic treatment of queueing theory for computer applications, with a detailed discussion of computer networks, is found in [KLEI75] and [KLEI76]. Perhaps the best book on performance modeling, covering queueing analysis and simulation, is [JAIN91].

A good elementary introduction to statistics is [PHIL99]. For a more detailed and rigorous treatment, there are numerous texts; one that is particularly well suited for self-study is [BULM79]. The U.S. government publishes two excellent guides to the practical application of statistics. [NIST10] contains tables, formulas, and examples that aid in determining the proper procedure for estimating values from samples and in evaluating the results. [LURI94] contains detailed step-by-step procedures for performing various statistical tests, as well as a certain amount of tutorial information on the procedures.

BULM79 Bulmer, M. *Principles of Statistics.* New York: Dover, 1979.

GROS09 Gross, D., and Harris, C. *Fundamentals of Queueing Theory.* New York: Wiley, 2009.

GUNT00 Gunther, N. *The Practical Performance Analyst.* New York: Authors Choice Press, 2000.

JAIN91 Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York: Wiley, 1991.

KLEI75 Kleinrock, L. *Queueing Systems, Volume I: Theory.* New York: Wiley, 1975.

KLEI76 Kleinrock, L. *Queueing Systems, Volume II: Computer Applications.* New York: Wiley, 1976.

LURI94 Lurie, D., and Moore, R. *Applying Statistics.* U.S. Nuclear Regulatory Commission Report NUREG-1475. (Available from the Government Printing Office, GPO Stock Number 052-020-00390-4.)

NIST10 National Institute of Standards and Technology. *NIST/SEMATECH e-Handbook of Statistical Methods.* 2010. http://www.itl.nist.gov/div898/handbook

PHIL99 Phillips, J. *How to Think about Statistics.* New York: Freeman, 1999.

20.12 PROBLEMS

20.1 Section 20.3 provided an intuitive argument to justify Little's formula. Develop a similar argument to justify the relationship $r = \lambda T_r$.

20.2 Figure 20.3 shows the number of items in a system as a function of time. This can be viewed as the difference between an arrival process and a departure process, of the form $n(t) = a(t) - d(t)$.

a. On one graph, show the functions $a(t)$ and $d(t)$ that produce the $n(t)$ shown in Figure 20.3.

b. Using the graph from (a), develop an intuitive argument to justify Little's formula. *Hint*: Consider the area between the two step functions, computed first by adding vertical rectangles and second by adding horizontal rectangles.

20.3 The owner of a shop observes that on average 18 customers per hour arrive and there are typically 8 customers in the shop. What is the average length of time each customer spends in the shop?

20.4 A simulation program of a multiprocessor system starts running with no jobs in the queue and ends with no jobs in the queue. The simulation program reports the average number of jobs in the system over the simulation run as 12.356, the average arrival rate as 25.6 jobs per minute, and the average delay for a job of 8.34 minutes. Was the simulation correct?

20.5 Section 20.3 provided an intuitive argument to justify the single-server relationship $\rho = \lambda T_s$. Develop a similar argument to justify the multiserver relationship $\rho = \lambda T_s/N$.

20.6 If an M/M/1 queue has arrivals at a rate of two per minute and serves at a rate of four per minute, how many customers are found in the system on average? How many customers are found in service on average?

20.7 What is the utilization of an M/M/1 queue that has four people waiting on average?

20.8 At an ATM machine in a supermarket, the average length of a transaction is two minutes, and on average, customers arrive to use the machine once every five minutes. How long is the average time that a person must spend waiting and using the machine? What is the 90th percentile of residence time? On average, how many people are waiting to use the machine? Assume M/M/1.

20.9 Messages arrive at random to be sent across a communications link with a data rate of 9,600 bps. The link is 70% utilized, and the average message length is 1,000 octets. Determine the average waiting time for constant-length messages and for exponentially distributed length messages.

20.10 Messages of three different sizes flow through a message switch. Seventy percent of the messages take 1 ms to serve, 20% take 3 ms, and 10% take 10 ms. Calculate the average time spent in the switch, and the average number of messages in the switch, when messages arrive at an average rate of

a. one per 3 ms
b. one per 4 ms
c. one per 5 ms.

20.11 Messages arrive at a switching center for a particular outgoing communications line in a Poisson manner with a mean arrival rate of

180 messages per hour. Message length is distributed exponentially with a mean length of 14,400 characters. Line speed is 9,600 bps.

a. What is the mean waiting time in the switching center?

b. How many messages will be waiting in the switching center for transmission on the average?

20.12 Often inputs to a queueing system are not independent and random but occur in clusters. Mean waiting delays are greater for this type of arrival pattern than for Poisson arrivals. This problem demonstrates the effect with a simple example. Assume items arrive at a queue in fixed-size batches of M items. The batches have a Poisson arrival distribution with mean rate λ/M, yielding a customer arrival rate of λ. For each item, the service time is T_s, and the standard deviation of service time of σ_{Ts}.

a. If we treat the batches as large-size items, what is the mean and variance of batch service time? What is the mean batch waiting time?

b. What is the mean waiting time for service for an item once its batch begins service? Assume that an item may be in any of the M positions in a batch with equal probability. What is the total mean waiting time for an item?

c. Verify the results of (b) by showing that for $M = 1$, the results reduce to the M/G/1 case. How do the results vary for values of $M > 1$?

20.13 Consider a single queue with a constant service time of four seconds and a Poisson input with mean rate of 0.20 items per second.

a. Find the mean and standard deviation of queue size.

b. Find the mean and standard deviation of residence time.

20.14 Consider a frame relay node that is handling a Poisson stream of incoming frames to be transmitted on a particular 1-Mbps outgoing link. The stream consists of two types of frames. Both types of frames have the same exponential distribution of frame length with a mean of 1000 bits.

a. Assume that priorities are not used. The combined arrival rate of frame of both types is 800 frames/second. What is the mean residence time (T_r) for all frames?

b. Now assume that the two types are assigned different priorities, with the arrival rate of type 1 of 200 frames/second and the arrival rate of type 2 of 600 frames/second. Calculate the mean residence time for type 1, type 2, and overall.

c. Repeat (b) for $\lambda_1 = \lambda_2 = 400$ frames/second.

d. Repeat (b) for λ_1 = 600 frames/second and λ_2 = 200 frames/second.

20.15 The Multilink Protocol (MLP) is part of X.25; a similar facility is used in IBM's System Network Architecture (SNA). With MLP, a set of data links exists between two nodes and is used as a pooled resource for transmitting packets, regardless of virtual circuit number. When a packet is presented to MLP for transmission, any available link can be chosen for the job. For example, if two LANs at different sites are connected by a pair of bridges, there may be multiple point-to-point links between the bridges to increase throughput and availability.

The MLP approach requires extra processing and frame overhead compared to a simple link protocol. A special MLP header is necessary for the protocol. An alternative is to assign each of the arriving packets to the queue for a single outgoing link in round-robin fashion. This would simplify processing, but what kind of effect would it have on performance?

Let us consider a concrete example. Suppose that there are five 9,600-bps links connecting two nodes, that the average packet size is 100 octets with an exponential distribution, and that packets arrive at a rate of 48 per second.

a. For a single-server design, calculate ρ and T_r.

b. For a multiserver design, it can be calculated that the Erlang C function has a value of 0.554. Determine T_r.

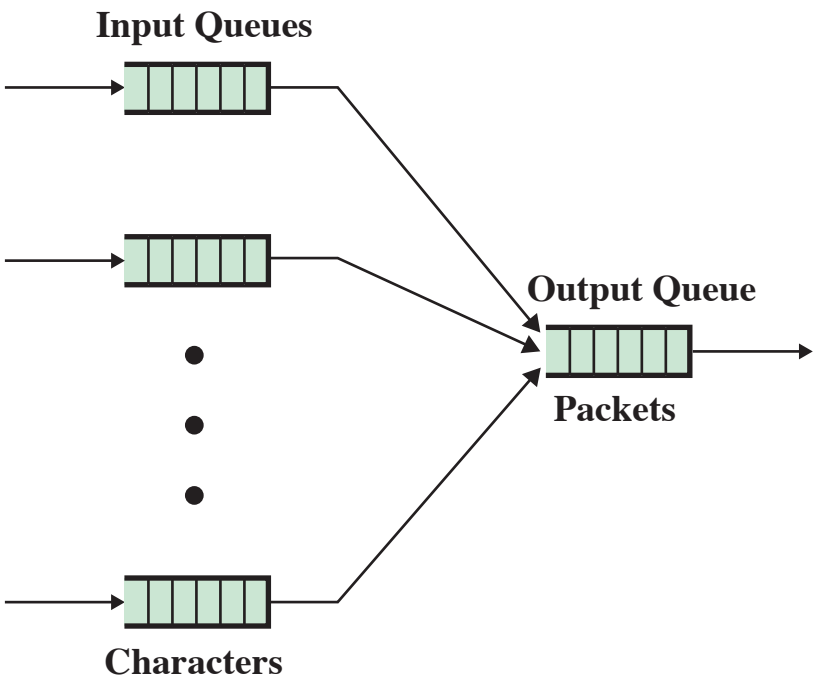


Figure 20.10 Queuing Model for a Packet Assembler/Disassembler (PAD)

20.16 A supplement to the X.25 packet-switching standard is a set of standards for a packet assembler-disassembler (PAD), defined in standards X.3, X.28, and X.29. A PAD is used to connect asynchronous terminals to a packet-switching network. Each terminal attached to a PAD sends characters one at a time. These are buffered in the PAD and then assembled into an X.25 packet that is transmitted to the packet-switching network. The buffer length is equal to the maximum data field size for an X.25 packet. A packet is formed from assembled characters and transmitted whenever the buffer is full, a special control character such as a carriage return is received, or when a timeout occurs. For this problem, we ignore the last two conditions. Figure 20.10 illustrates the queueing model for the PAD. The first queue models the delay for characters waiting to be put into a packet; this queue is completely emptied when it is filled. The second queue models the delay waiting to transmit packets. Use the following notation:

λ = Poisson input rate of characters from each terminal

C = Rate of transmission on the output channel in characters per second

M = Number of data characters in a packet

H = Number of overhead characters in a packet

K = Number of terminals

a. Determine the average waiting time for a character in the input queue.
b. Determine the average waiting time for a packet in the output queue.
c. Determine the average time spent by a character from when it leaves the terminal to when it leaves the PAD. Plot the result as a function of normalized load.

20.17 A fraction P of the traffic from a single exponential server is fed back into the input as shown in Figure 20.11. In the figure, Λ denotes the system throughput, which is the output rate from the server.
a. Determine the system throughput and the server utilization and the mean residence time for one pass through the server.
b. Determine the mean number of passes that an item makes through the system and the mean total time spent in the system.

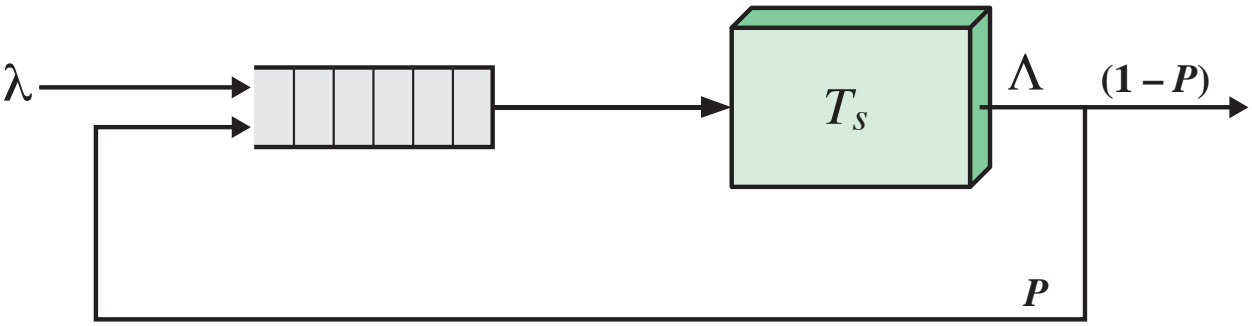


Figure 20.11 Feedback Queue