Appendix H
Queueing System Concepts

William Stallings

Supplement to
Operating Systems, Eighth Edition
Pearson 2014
http://williamstallings.com/OperatingSystems/

In a number of chapters in this book, results from queueing theory are used. Chapter 20 provides a detailed discussion of queueing analysis. For purposes of understanding the description of the results in the book, however, the brief overview in this appendix should suffice. In this appendix we present a brief definition of queueing systems and define key terms.

H.1 WHY QUEUEING ANALYSIS?

It is often necessary to make projections of performance 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. 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 H.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 a router that is capable of processing and forwarding 1000 packets per second, then a load of 0.5 represents an arrival rate of 500 packets per second, and the response time is the amount of time it takes to retransmit any incoming packet. The lower line is a simple projection[1] based on 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 to 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

[1] The lower line is based on fitting a third-order polynomial to the data available up to a load of 0.5.

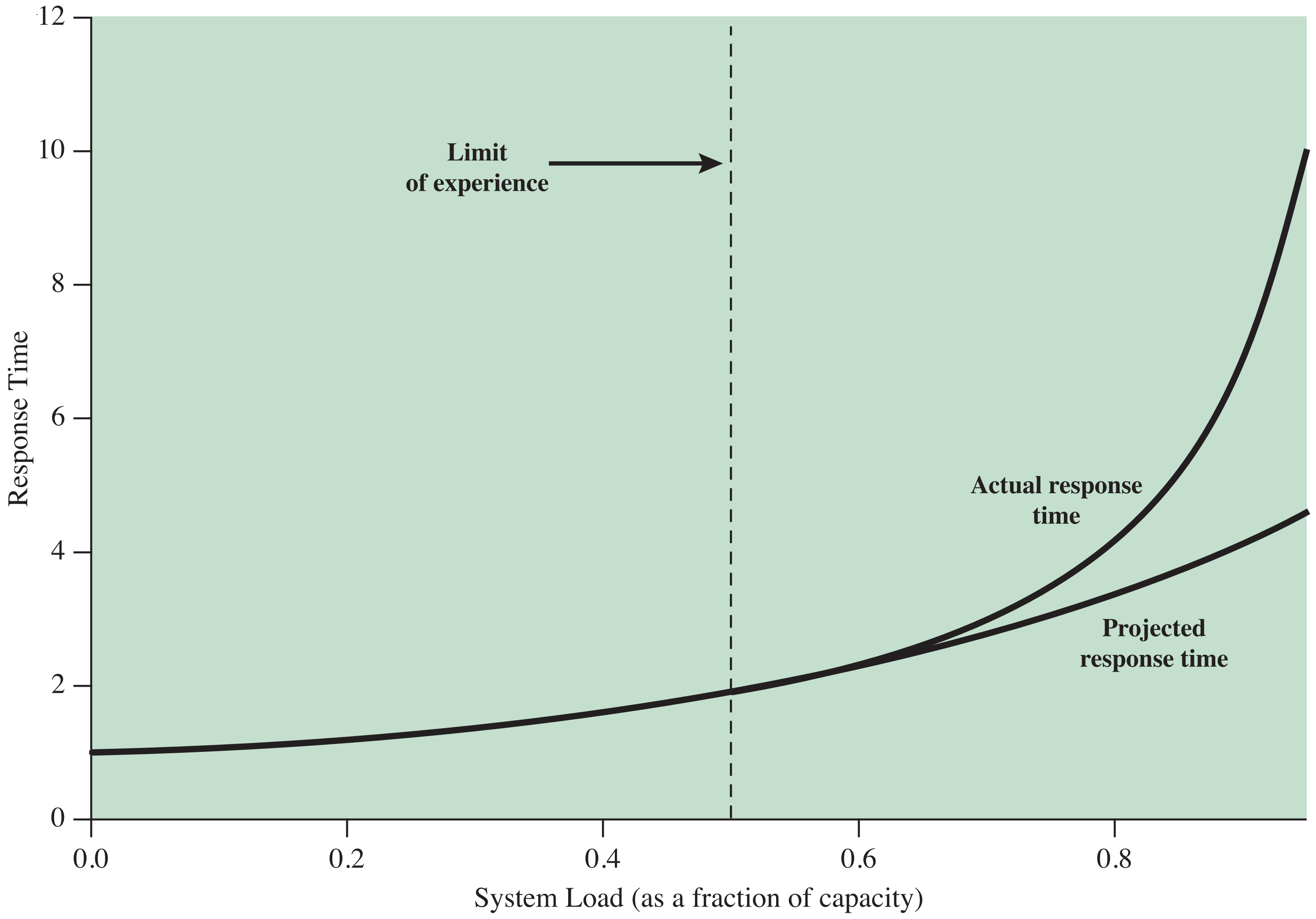


Figure H.1 Projected Versus Actual Response Time

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.

H.2 THE SINGLE-SERVER QUEUE

The simplest queueing system is depicted in Figure H.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. Examples: a processor provides service to processes; a transmission line provides a transmission service to packets or frames of data; an I/O device provides a read or write service for I/O requests.

Table H.1 summarizes some important parameters associated with a queueing model. Items arrive at the facility at some average rate (items

[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.

Table H.1 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
ρ	=	utilization; fraction of time facility (server or servers) is busy
w	=	mean number of items waiting to be served
T_w	=	mean waiting time (including items that have to wait and items with waiting time = 0)
r	=	mean number of items resident in system (waiting and being served)
T_r	=	mean residence time; time an item spends in system (waiting and being served)

arriving per second) λ. At any given time, a certain number of items will be waiting in the queue (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;

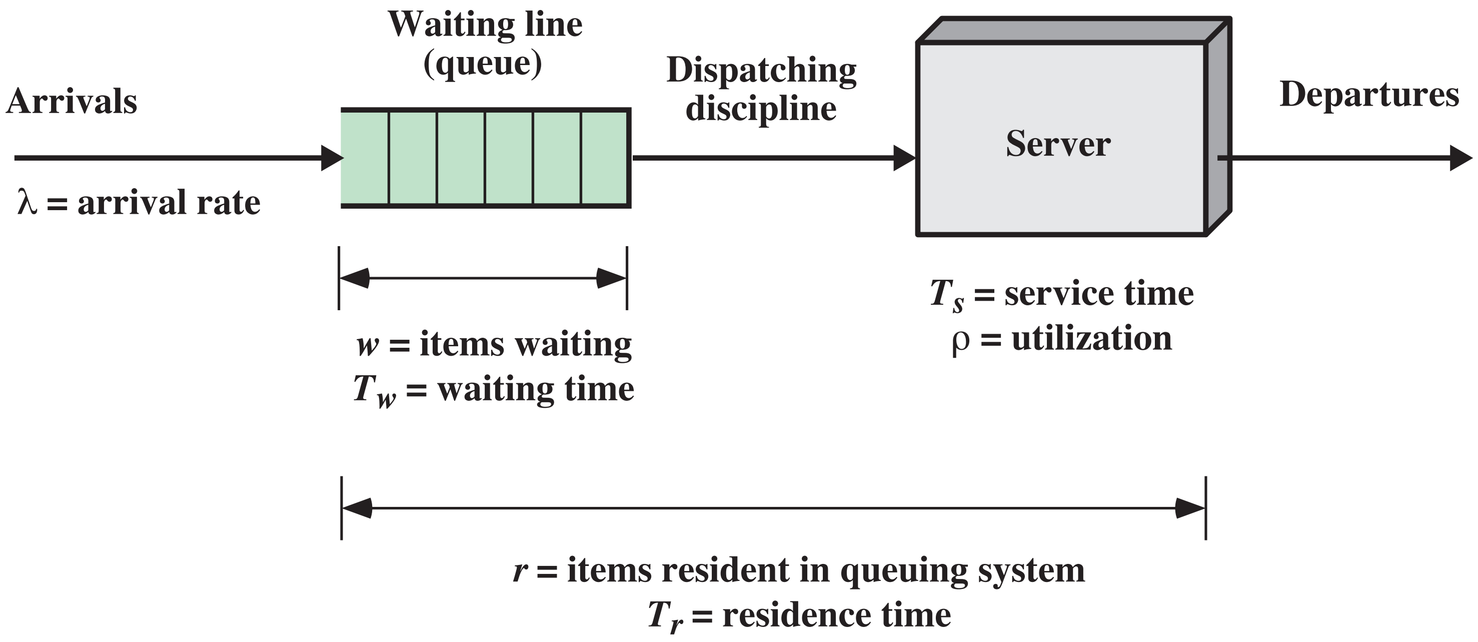


Figure H.2 Queueing System Structure and Parameters for Single-Server Queue

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 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. 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 between 70 and 90% of the theoretical maximum.

The following assumptions are typically made:

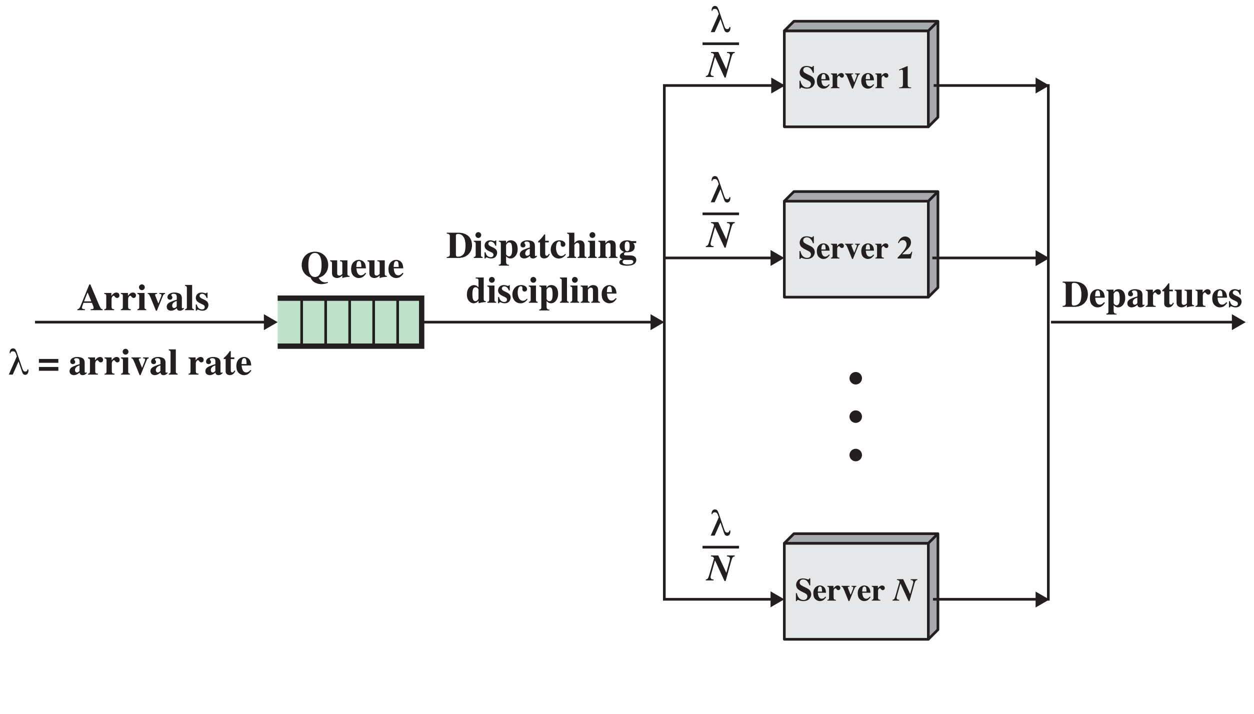
- **Item population:** Typically, we assume an infinite population. This means that the arrival rate is not altered by the loss of population. If the population is finite, then the population available for arrival is reduced by the number of items currently in the system; this would typically reduce the arrival rate proportionally.

[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 queue (before being served).

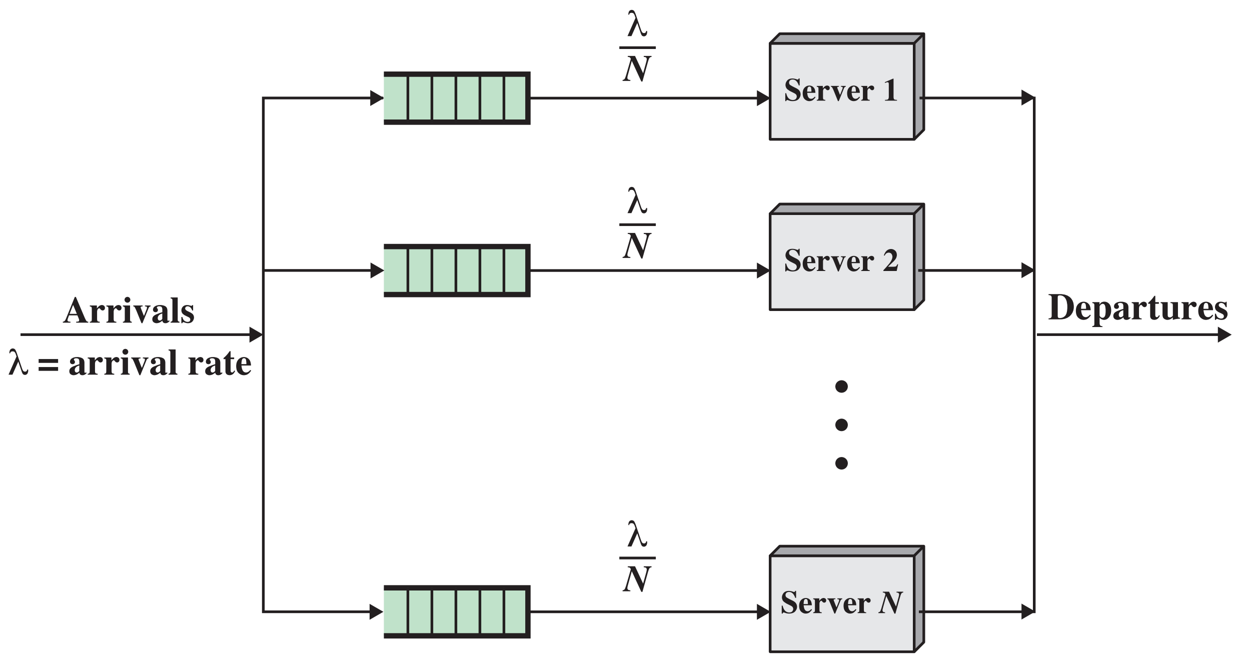
- **Queue size:** Typically, we assume an infinite queue size. Thus, the waiting line can grow without bound. With a finite queue, it is possible for items to be lost from the system. In practice, any queue is finite. In many cases, this will make no substantive difference to the analysis.
- **Dispatching 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; this discipline is what is normally implied when the term *queue* is used. Another possibility is last-in-first-out. One that you might encounter in practice is a dispatching discipline based on service time. For example, a packet-switching node may choose to dispatch packets on the basis of shortest first (to generate the most outgoing packets) or longest first (to minimize processing time relative to transmission time). Unfortunately, a discipline based on service time is very difficult to model analytically.

H.3 THE MULTISERVER QUEUE

Figure H.3 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, it makes no difference which server is chosen for the item. If all servers are busy, a queue begins to form. As soon



(a) Multiserver queue



(b) Multiple Single-server queues

Figure H.3 Multiserver Versus Multiple Single-Server Queues

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 H.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_ρ 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, 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 H.3b shows the structure of multiple single-server queues.

H.4 POISSON ARRIVAL RATE

Typically, analytic queueing models assume that the arrival rate obeys a Poisson distribution. This is what is assumed in the results of Table 9.6. We define this distribution as follows. If items arrive at a queue according to a Poisson distribution, this may be expressed as

$$\Pr[k \text{ items arrive in time interval } T] = \frac{(\lambda T)^k}{k!} e^{-\lambda T}$$

E[number of items to arrive in time interval T] = λT

Mean arrival rate, in items per second = λ

Arrivals occurring according to a Poisson process are often referred to as **random arrivals**. This is because the probability of arrival of an item in a small interval is proportional to the length of the interval and is independent of the amount of elapsed time since the arrival of the last item. That is, when items are arriving according to a Poisson process, an item is as likely to arrive at one instant as any other, regardless of the instants at which the other customers arrive.

Another interesting property of the Poisson process is its relationship to the exponential distribution. If we look at the times between arrivals of items T_a (called the interarrival times), then we find that this quantity obeys the exponential distribution:

$$\Pr[T_a < t] = 1 - e^{-\lambda t}$$

$$E[T_a] = \frac{1}{\lambda}$$

Thus, the mean interarrival time is the reciprocal of the arrival rate, as we would expect.