2.3.1. Introduction to Queueing Theory

Queueing systems can be used to model processes in which customers arrive, wait their turn for service, are serviced, and then depart. Supermarket checkout stands, World Series ticket booths, and doctor's waiting rooms are examples of queueing systems. Queueing systems can be characterized by five components:

1. The interarrival-time probability density function.
2. The service-time probability density function.
3. The number of servers.
4. The queueing discipline.
5. The amount of buffer space in the queues.

It is worth explicitly noting that we are considering only systems with an infinite number of customers (i.e., the existence of a long queue does not so deplete the population of customers that the input rate is materially reduced). (In contrast, in a time-sharing system model, there are only a finite number of customers. If half of them are waiting for a response, the input rate will be significantly reduced.)

The interarrival-time probability density describes the interval between consecutive arrivals. One could imagine hiring someone (e.g., a graduate student, since they do not cost much), to watch customers arrive. At each arrival the graduate student would record the elapsed time since the previous arrival. After a sufficiently long sampling time, the list of numbers could be sorted and grouped: so many interarrival times of 0.1 sec, so many of 0.2 sec, etc. This probability density characterizes the arrival process.

Each customer requires a certain amount of the server's time. The amount of service time varies from customer to customer (e.g., one has a full shopping cart of groceries and the next has only a small box of peanut butter cookies). To analyze a queueing system, the service-time probability density function, like the interarrival density function, must be known.

The number of servers speaks for itself. Many banks, for example, have one big queue for all customers. Whenever a teller is free, the customer at the front of the queue goes directly to that teller. Such a system is a multiserver queueing system. In other banks, each teller has his or her own private queue. In this case we have a collection of independent single-server queues, not a multiserver system.

The queueing discipline describes the order in which customers are taken from the queue. Supermarkets use first come, first served. Hospital emergency rooms often use sickest first rather than first come, first served. In friendly office environments, shortest job first prevails at the photocopy machine.

Not all queueing systems have an infinite amount of buffer space. When too many customers are queued up for a finite number of slots, some customers get lost or rejected.

We will concentrate exclusively on infinite-buffer, single-server systems using first come, first served. The notation A/B/m is widely used in the queueing literature for these systems, where A is the interarrival-time probability density, B the service-time probability density, and m the number of servers. The probability densities A and B are chosen from the set

M - exponential probability density (M stands for Markov)
D - all customers have the same value (D is for deterministic)
G - general (i.e., arbitrary probability density)

The state of the art ranges from the M/M/1 system, about which everything is known, to the G/G/m system, for which no exact analytic solution is yet known.

Throughout this book we use queuing theory as one of our basic tools to analyze network performance. In particular, we usually assume the M/M/1 model. The assumption of an exponential interarrival probability is completely reasonably for any system that has a large number of independent customers. Under such conditions, the probability of exactly n customers arriving during an interval of length t is given by the Poisson law:

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{2-1}$$

where λ is the mean arrival rate. (See Problem 2-6 for a derivation of the Poisson law.)

Now we will show that Poisson arrivals generate an exponential interarrival probability density. The probability, $a(t)\Delta t$, that an interarrival interval is between t and $t + \Delta t$ is just the probability of no arrivals for a time t times the probability of one arrival in the infinitesimal interval Δt:

$$a(t)\Delta t = P_0(t)P_1(\Delta t)$$

$P_0(t)$ is $e^{-\lambda t}$, and $P_1(\Delta t)$ is $\lambda \Delta t e^{-\lambda \Delta t}$. In the limit $\Delta t \to 0$, the exponential factor in P_1 approaches unity, so

$$a(t)\, dt = \lambda e^{-\lambda t}\, dt \tag{2-2}$$

Note that the integral of Eq. (2-2) from 0 to ∞ is 1, as it should be.

Although the assumption of an exponential interarrival probability density is usually reasonable, the assumption of exponential services times is harder to defend on general grounds. Nevertheless, for situations in which increasingly long service times are increasingly less likely, M/M/1 may be an adequate approximation. We leave it to the reader to show that if the probability of service finishing in some small time interval Δt is $\mu \Delta t$, then the service-time probability density function is $\mu e^{-\mu t}$, with a mean service time of $1/\mu$ sec/customer.

2.3.2. The M/M/1 Queue in Equilibrium

The state of an M/M/1 queueing system (see Fig. 2-14) is completely described by telling how many customers are currently in the system, including both queue and server. At first you might think that it would also be necessary to describe how far along the customer currently being served was, but the exponential density function has no memory: the probability of the remaining service time requiring t seconds is independent of how much service the customer has already received! The exponential function is the only one with this remarkable property, which is why queueing theorists love it so much.

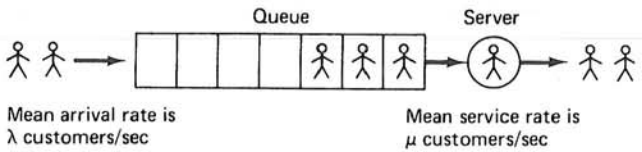


Fig. 2-14. A single-server queueing system with four customers, one in service and three in the queue.

Let p_k be the equilibrium probability that there are exactly k customers in the system (queue + server). Once we have derived these probabilities we can find the mean number of customers in the system, the expected waiting time, and other statistical properties of the system. Even when the system is in equilibrium, transitions between states take place. If the system is in state 4 (i.e., four customers in the system) and a new customer arrives, the system moves into state 5. Similarly, when a customer receives his desired service, the system moves down one state. Queueing systems in which the only transitions are to adjacent states are known in the trade as **birth-death systems**.

Figure 2-15 shows the states for a single-server queueing system, with the allowed transitions indicated by arrows. To analyze this system, we must know how many of each transition occur per second. If the mean arrival rate is λ customers/sec, the mean number of transitions/sec from state 0 to state 1 is λp_0. Suppose, for example, that 40 customers/sec arrive, there is a 20% chance of finding the system in state 0 (empty), and a 15% chance of finding the system in state 1 (one customer being served, queue empty). There will be eight transitions from 0 to 1 each second on the average, and six transitions from 1 to 2. In general, the transition rate from state k to state $k + 1$ is λp_k.

Similarly, if the server is capable of processing μ customers/sec the transition rate from state $k + 1$ to state k is μp_{k+1}. Note that the transition rate is the completion rate times the probability of the system being in the initial state,

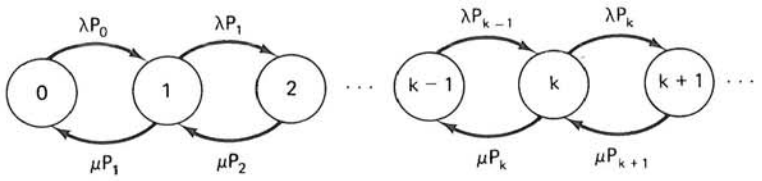


Fig. 2-15. State diagram for a single-server queueing system. The transition rates are shown as labels on the arrows.

not the final state. If you want to know how many transitions per second happen from state 4 to state 3, you need to know what the probability is of finding the system in state 4. You do not need to know what the state 3 probability is.

In equilibrium the probability of finding the system in a given state does not change with time. In particular, the probability of there being more than k customers in the system is constant. The transition from k to $k + 1$ increases this probability, and the transition from $k + 1$ to k decreases it. Therefore, these two transitions must occur at the same rate. If this were not so, the system would not be in equilibrium. If there were many transitions from, say, 4 to 5, but few transitions from 5 to 4, the mean number of customers in states above 4 would increase in time, violating our assumption about the system being in equilibrium. This principle, sometimes referred to as the principle of **detailed balancing**, is the key to solving for the state probabilities.

Looking at Fig. 2-15, we see that

$$\lambda p_0 = \mu p_1 \tag{2-3}$$

$$\lambda p_1 = \mu p_2 \tag{2-4}$$

and in general,

$$\lambda p_k = \mu p_{k+1} \tag{2-5}$$

Using Eq. (2-3), we can solve for p_1 in terms of p_0. Using Eq. (2-4), we can solve for p_2 in terms of p_1 and then use the previous result to get p_2 in terms of p_0. The solution for the general case can be found be repeating this process, yielding

$$p_k = \rho^k p_0 \tag{2-6}$$

where we have introduced $\rho = \lambda/\mu$. The variable ρ is known as the **traffic intensity**. It must be less than 1. Queueing systems that receive input faster than the server can process it are inherently unstable, and their queues grow without bound.

To eliminate p_0 from Eq. (2-6), we use the fact that the probabilities must sum to 1:

$$\sum_{k=0}^{\infty} \rho^k p_0 = 1$$

Using the well-known formula for the sum of a geometric series,

$$\sum_{k=0}^{\infty} \rho^k = \frac{1}{1-\rho} \quad , \tag{2-7}$$

we find that $p_0 = 1 - \rho$, and finally

$$p_k = (1-\rho)\rho^k \tag{2-8}$$

Notice that $\rho = 1 - p_0$ is just the probability that the system is not empty (i.e., the probability that the server is busy).

The mean number of customers in the system, N, can now be found directly from the state probabilities, Eq. (2-8):

$$N = \sum_{k=0}^{\infty} k p_k = (1-\rho) \sum_{k=0}^{\infty} k\rho^k$$

The value of the summation can be found by differentiating both sides of Eq. (2-7) with respect to ρ and then multiplying through by ρ. Using this result, we find the mean number of customers in the system to be

$$N = \frac{\rho}{1-\rho} \tag{2-9}$$

As ρ approaches 1, the queue length grows very quickly.

Having found the mean number of customers in the system, we are now ready to determine the total waiting time, T, the mean interval between customer arrival and customer departure, including service time. Imagine that our friend the graduate student stops recording interarrival times once in a while, and in a fit of pique paints one of the customers shocking pink. The pink customers progress along like the other customers and are eventually disgorged from the system. Since the pink customers are in the system for an average time T (the mean time for all customers), the mean number of new arrivals subsequent to the arrival of a pink customer and just prior to his own departure is λT. This result follows directly from the fact that the arrival rate is λ customers/sec. At the instant of the pink customer's departure, all λT of these customers, and no others, are in the system. Since the mean number of customers in the system at any time is N, we have the basic result: $N = \lambda T$. This equation was first proven by D. C. Little (1961) and is known as **Little's result**.

Using Little's result and Eq. (2-9), we can now find the total waiting time, including service time:

$$T = \frac{N}{\lambda} = \frac{\rho/\lambda}{1-\rho} = \frac{1/\mu}{1-\rho} = \frac{1}{\mu - \lambda} \tag{2-10}$$

This key result will be the basis of our network delay analysis.

As an example of Eq. (2-10), consider a public birdbath at which birds arrive according to a Poisson distribution. The mean arrival rate is 3 birds/min. The bathing time is exponentially distributed with a mean of 10 sec/bird. How long does a bird have to wait in the queue? The mean arrival rate is $\lambda = 0.05$ customer/sec. The mean service rate is $\mu = 0.10$ customer/sec. From Eq. (2-10) we find that $T = 20$ sec for waiting plus service. Since the mean service-time $(1/\mu)$ is 10 sec, the mean queueing time is then $20 - 10 = 10$ sec.

For the sake of generality, we will state, but not derive, the formula for the mean number of customers in the system for an M/G/1 queueing system:

$$N = \rho + \rho^2 \frac{1 + C_b^2}{2(1 - \rho)} \tag{2-11}$$

where C_b is the ratio of the standard deviation to the mean of the service time probability density function. This result, known as the **Pollaczek-Khinchine** equation, is valid for any service-time distribution. It shows that if two service-time distributions have equal means, the one with the larger standard deviation will produce a longer waiting time. For the Poisson distribution, upon which M/M/1 is based, $C_b = 1$.

2.3.3. Networks of M/M/1 Queues

The results derived above for the M/M/1 queue can be directly applied to the problem of finding the queueing delay for packets in an IMP. But first it is convenient to change the notation slightly to convert the units of service time from customers/sec to bits/sec. Let the probability density function for packet size in bits be $\mu e^{-\mu x}$ with a mean of $1/\mu$ bits/packet. Now introduce the capacity of communication channel i as C_i bits/sec. The product μC_i is then the service rate in packets/sec. The arrival rate for channel i is λ_i packets/sec. Equation (2-10) can now be rewritten for channel i as

$$T_i = \frac{1}{\mu C_i - \lambda_i} \tag{2-12}$$

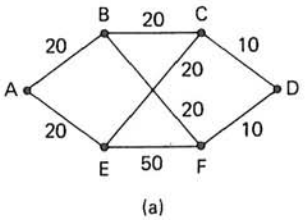
where T_i includes both queueing and transmission time, as can be seen by taking the limit $\lambda_i \to 0$. Notice that the mean packet size does not depend on the channel, as the capacity and input rate do. This application of queueing theory to a communication channel was first due to Kleinrock (1964).

One problem that we face in a network is that the communication channels are not isolated. The output of one channel becomes the input to another. Several lines may converge upon a single IMP dumping packets there. Thus the input to a certain line is no longer a Poisson process outside the network, but the sum of the outputs of several other network lines. Fortunately, it has been shown (Burke, 1956) that if the outputs of several M/M/1 servers feed into the input queue of another server, the resulting input process is also a Poisson process, with mean equal to the sum of the means of the feeding

processes. Even better, Jackson (1957) has shown that an open network of M/M/1 queues can be analyzed as though each one were isolated from all the others. All you need to know is the mean input rate.

However, there is still one obstacle in our way. When a packet moves around the network, it maintains its size, of course. This property introduces nonrandom correlations into the system. When a big monster packet comes along, it takes a long time to service, hence causing a noticeable gap in the arrival pattern of the queue being fed into. We can get around this problem by assuming that every time a packet arrives at an IMP, it loses its identity and a new length is chosen for it at random. This assumption, first made by Kleinrock (1964), is known as the **Independence Assumption**. Simulation and actual measurements show that it is quite reasonable to make it. Besides, if we do not make it, we cannot make any progress at all.

Let us illustrate the calculation of the mean queueing delay in a network by an example. Fig. 2-16(a) depicts a six node network using full-duplex lines. It may be helpful to think of each full-duplex line as a pair of oppositely directed simplex lines, one eastbound and one westbound. The capacities of the lines are given in kbps. The matrix of Fig. 2-16(b) has an entry for each source-destination pair. The entry for source i and destination j shows the route to be used for i-j traffic and also the number of packets/sec (γ_{ij}) to be sent from i to j. For example, there are 3 packets/sec from B to D, and they use the route BFD.



(a)

Destination

Source	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

(b)

Fig. 2-16. (a) A network, with line capacities shown in bps. (b) The traffic (in packets/sec) and the routing matrix.

Given these routing and traffic matrices, it is possible to calculate the total traffic in line i, λ_i. For example, the B-D traffic contributes 3 packets/sec to the BF line and 3 packets/sec to the FD line. Similarly, the A-D traffic

contributes 1 packet/sec to each of three lines. The total traffic in each eastbound line is shown in Fig. 2-17. In this example, the traffic matrix is symmetric (i.e., there is always as much traffic from X to Y as from Y to X). Furthermore, the Y-X traffic uses the exact reverse route of the X-Y traffic. We have done this to make $\lambda_{XY} = \lambda_{YX}$ for all X and Y (to simplify the example).

i	Line	λ_i (pkts/sec)	C_i (kbps)	μC_i (pkts/sec)	T_i (ms)
1	AB	14	20	25	91
2	BC	12	20	25	77
3	CD	6	10	12.5	154
4	AE	11	20	25	71
5	EF	13	50	62.5	20
6	FD	8	10	12.5	222
7	BF	10	20	25	67
8	EC	8	20	25	59

Fig. 2-17. Analysis of the network of Fig. 2-16 using a mean packet size of 800 bits. The reverse traffic (*BA*, *CB*, *DC*, etc.) is the same as the corresponding forward traffic, and has the same delay.

Figure 2-17 shows λ_i and C_i for each of the lines in Fig. 2-16. Let us now assume a mean packet size $1/\mu = 800$ bits/packet, leading to the fifth column of Fig. 2-17 With λ_i and μC_i now known, we can apply Eq. (2-12) and calculate the queueing + transmission time for each line. These times are given in the last column of Fig. 2-17.

To calculate the mean packet delay for a network with n IMPs and m lines, it is convenient to define

$$\gamma = \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij} \qquad \text{and} \qquad \lambda = \sum_{i=1}^{m} \lambda_i$$

Although it might appear at first that summing the traffic between each pair of IMPs (γ) should give the same result as summing the traffic on all the lines (λ) this is not so. The reason for this seeming anomaly is that some routes use two or more hops, such as the route from A to D in Fig. 2-16(a). The A-D traffic contributes to the traffic in three lines. In other words, the A-D traffic is counted three times in λ but only once in γ. If every route were three hops, then λ would be three times γ. The ratio $\bar{n} = \lambda/\gamma$ is the mean number of hops per packet. Notice that $\bar{n}$ depends only on the relative traffic pattern and the routing algorithm, but not on the absolute volume of traffic. Doubling each

element of γ_{ij} has no effect on $\bar{n}$.

As an aside, it is interesting to notice that λ can be related to the traffic matrix γ_{ij} by

$$\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij}\, \gamma_{ij}$$

where h_{ij} is the number of hops in the route from IMP i to IMP j. If shortest path routing is used, as it is in our example, the λ so obtained is the theoretical minimum achievable for the given topology and traffic matrix.

The mean delay per hop is simply the sum of the individual line delays, from Eq. (2-12), weighted by the amount of traffic in the line, λ_i. To normalize the result, we divide by the total traffic, λ, yielding

$$\text{mean delay per line} = \sum_{i=1}^{m} \frac{\lambda_i T_i}{\lambda}$$

However, the mean packet delay, T, is longer, because many packets must make several hops. Remembering that $\bar{n}$ is the mean number of hops per packet, we have

$$T = \bar{n} \sum_{i=1}^{m} \frac{\lambda_i T_i}{\lambda} = \bar{n} \sum_{i=1}^{m} \frac{\lambda_i/\lambda}{\mu C_i - \lambda_i} \tag{2-13}$$

Although we will use Eq. (2-13) repeatedly throughout this chapter, it is worth pointing out that it is only an approximation to reality. First, the assumption of exponentially distributed packet lengths may or may not be a good one in a given application. Also, we have neglected all other components to the delay, such as the time needed for the IMP to compute the software checksum (if any) and otherwise massage the packet, and the propagation delay. We have also neglected the effects of errors, retransmissions, and control packets. For a more realistic model of the delay, see Kleinrock (1970).

Using Eq. (2-13) we can now calculate the mean packet delay for our example of Fig. 2-17. For this example, $\gamma = 124$ and $\lambda = 164$ packets/sec (including the reverse traffic *BA*, *CB*, etc.). The value of $\bar{n}$ is 164/124 = 1.32 hops/packet. The mean packet delay turns out to be 114 msec.

An important question is how much traffic a network can support using a given static routing algorithm. To see what the maximum traffic is for the routing of Fig. 2-16(b), we scale up the traffic matrix by multiplying each element by the same constant, and then recompute T. This has been done in Fig. 2-18. A scale factor of 1.0 corresponds to the traffic of Fig. 2-16(b). Higher numbers represent heavier loads, and lower numbers represent lighter loads. The second curve is for a mean packet size of 1200 instead of 800 bits.

Another interesting question is: What is the mean packet delay when the network is almost empty? It is not 0, because the transmission (service) time is included in the delay, and that is constant, independent of the load. In an unloaded network, $\mu C_i >> \lambda_i$ so we can neglect the λ_i term in the

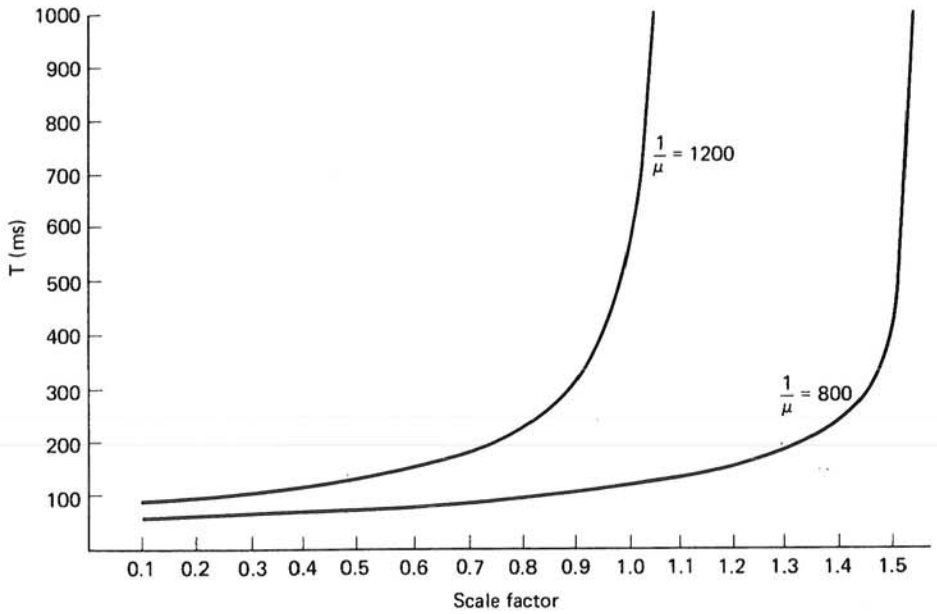


Fig. 2-18. Mean delay for Fig. 2-16(a) as a function of overall load.

denominator of Eq. (2-13) to get

$$T_0 = \bar{n} \sum_{i=1}^{n} \frac{\lambda_i/\lambda}{\mu C_i}$$

For the example of Fig. 2-16, T_0 is 57 msec. A crude approximation of the curves of Fig. 2-18 is a constant value of T_0 extending from zero load up to some critical value, which then suddenly jumps to infinity. To find the critical value, we need to know which line saturates first. The line that saturates first is the one with the lowest $\mu C_i/\lambda_i$ ratio. From Fig. 2-17 we see that line *FD* has the lowest ratio, 12.5/8 = 1.56. As we multiply all the entries in the traffic matrix by a common scale factor, line *FD* will be the first to reach saturation. Saturation will be achieved at a scale factor of 1.56. This conclusion is also clear from Fig. 2-18.

One last remark before leaving this example concerns the true maximum flow. The capacity of the smallest cut of Fig. 2-16 is 20 kbps or 25 packets/sec. As we scale up the traffic, the smallest cut will be the first to saturate. Using the max-flow min-cut theorem we see that it should be possible to pump twenty-five 800-bit packets/sec from *A* to *D*. In reality, only 1.56 × 14 = 21.8/sec will go from *A* to *D*.

This low value occurs because we are now dealing with multiple sources

and sinks simultaneously. Just looking only at the required *D*-*F* flow (4 packets/sec, or 3.2 kbps) and the capacity exiting from *D* (20 kbps), we can see immediately that the scale factor cannot be much greater than 6. In other words, the *D*-*F* flow requirement affects the *A*-*D* flow. No such effect is present in the single source-sink flow problem. Nonetheless, it is still interesting to see how much traffic flows across the minimum cut by adding up the entries for *CD* and *FD* from Fig. 2-17, and then multiplying by the scale factor that saturates the network (1.56). The result is 21.8 packets/sec. To squeeze the last 15% out of the network, we would need to improve the routing, in particular, we would need to allow load splitting. Since *FD* saturates before *CD*, we could try to route some of the *A*-*D* or *B*-*D* traffic over *C* instead of having all of it go via *F*.

By now it should be clear, however, that to achieve the maximum theoretical flow, we must tolerate an infinite delay. In fact, to even achieve a flow close (say within 20%) of the theoretical limit, we must tolerate a queueing delay several times the delay of the unloaded network. From Fig. 2-18 we see that at a scale factor of 1.25 (80% of the maximum achievable with the given routing algorithm), the delay is triple the delay of an unloaded network. We can now clearly see that there is an inherent conflict between high throughput and low delay. To get the maximum throughput, we need $\mu C_i = \lambda_i$ in Eq. (2-13) for those lines that are contained in the minimum cut. But the very act of allowing the flow to approach the capacity in any line drives the delay through the roof. Since short delay times and high throughput are difficult to achieve simultaneously, it is important that the network designers clearly understand what their goals are.