27 A Little Bit of Queueing Theory

We have alluded to the fact that probability is useful in the performance analysis and design of computer systems. Queueing theory is an area of applied probability which directly targets systems performance. Here the “system” might refer to a computer system, a call center, a healthcare system, a manufacturing system, a banking system, or one of many other examples. Markov chains (particularly continuous-time chains) are just one of many tools used in queueing theory. In this final part of the book, we provide a very brief introduction to queueing theory. For a much more in-depth coverage, see [35].

27.1 What Is Queueing Theory?

Queueing theory is the theory behind what happens when you have lots of jobs, scarce resources, and subsequently long queues and delays. It is literally the “theory of queues”: what makes queues appear, how to predict the queue lengths, and how to improve system design by making the queues get shorter or go away.

Imagine a computer system, say a web server, where there is only one job. The job arrives, it uses certain resources (say some CPU, some I/O, some bandwidth), and then it departs. If we know the job’s resource requirements, it is very easy to predict exactly when the job will depart. There is no delay because there are no queues. If every job indeed got to run in isolation on its own computer system, there would be no need for queueing theory. Unfortunately, that is rarely the case.

Queueing theory applies anywhere that queues come up (see Figure 27.1). We have all had the experience of waiting in line at the bank, wondering why there are not more tellers, or waiting in line at the supermarket, wondering why the express lane is for 8 items or less rather than 15 items or less, or whether it might be best to actually have *two* express lanes, one for 8 items or less and the other for 15 items or less. Queues are also at the heart of any computer system. Your CPU uses a time-sharing scheduler to serve a queue of jobs waiting for CPU time. A computer disk serves a queue of jobs waiting to read or write blocks. A router in a network serves a queue of packets waiting to be routed. The router queue is

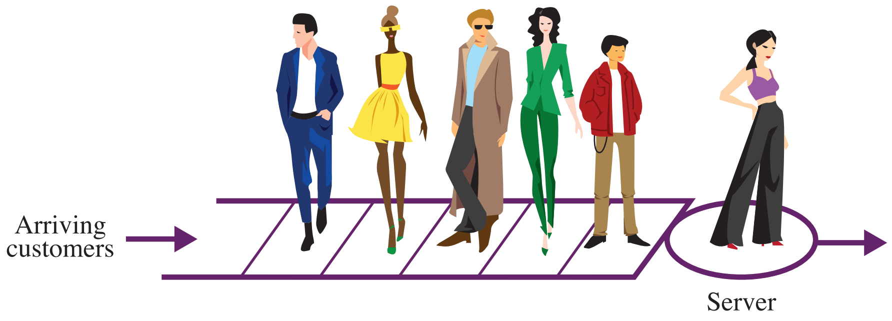


Figure 27.1 *Illustration of a queue, in which customers wait to be served, and a server. The picture shows one customer being served at the server and five others waiting in the queue.*

a finite-capacity queue, in which packets are dropped when demand exceeds the buffer space. Memory banks serve queues of threads requesting memory blocks. Databases sometimes have lock queues, where transactions wait to acquire the lock on a record. Server farms consist of many servers, each with its own queue of jobs. Data centers often have a single central queue, where each job requests some number of resources to run. The list of examples goes on and on.

The goals of a queueing theorist are three-fold. The first is *predicting* the system performance. Typically this means predicting mean delay or delay variability or the probability that delay exceeds some service level agreement (SLA). However, it can also mean predicting the number of jobs that will be queueing or the mean number of servers being utilized (e.g., total power needs), or any other such metric. The second goal is *capacity provisioning*, namely determining how many resources are needed to achieve certain performance goals. One might want to provision to ensure that the system is stable, meaning that the queue lengths don't grow unboundedly. Or one might provision to ensure that certain SLAs are met. The third goal is finding a superior system *design* to improve performance. This often takes the form of a smarter scheduling policy or routing policy to reduce delays.

27.2 A Single-Server Queue

Figure 27.2 illustrates a single-server queue. The circle represents the server. One job can be served (worked on) at a time. New jobs arrive over time. If the server is free, the arriving job is served immediately; otherwise, the job has to queue. Unless stated otherwise, we assume that the jobs queue in First-Come-First-Served (FCFS) order.

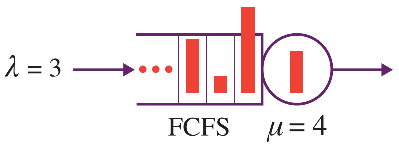


Figure 27.2 *Single-server queue. The height of the rectangle indicates the size of the job.*

We formally define these quantities associated with the single-server queue:

Service order This is the order in which jobs (packets) will be served by the server. Unless otherwise stated, assume FCFS.

Average arrival rate This is the average rate, λ, at which jobs arrive to the server. For example, $\lambda = 3$ jobs/s, indicates that on average 3 jobs are arriving every second. Of course, some seconds might see more than 3 jobs, and others might see fewer than 3 jobs.

Interarrival time, mean interarrival time It is common to imagine that the times between arrivals are independent and identically distributed (i.i.d.), where there is a random variable (r.v.), I, which represents the time between successive job arrivals. In this case, $\mathbf{E}[I]$ would represent the mean interarrival time.

Question: How can we think of $\mathbf{E}[I]$ in terms of what we've already seen?

Answer: By definition, $\mathbf{E}[I] = \frac{1}{\lambda}$. Thus, in our example, the mean interarrival time would be $\frac{1}{\lambda} = \frac{1}{3}$ seconds. Note that we have not said anything about what the distribution of the interarrival time looks like. It might be Deterministic, meaning that exactly one job arrives every $\frac{1}{3}$ seconds, or it might be Exponential with rate 3, meaning that on average the time between arrivals is $\frac{1}{3}$, but it might be anywhere from 0 to infinity.

Service requirement, size It is common to assume that the sizes of jobs are i.i.d. and are denoted by the r.v. S. The size of a job is also called its service requirement. S is expressed in units of time and denotes the time it would take the job to run on this server if there were no other jobs around (no queueing). Importantly, the size is typically associated with the server.

Mean service time This is $\mathbf{E}[S]$, namely the average time required to serve a job on this server, where again "service" does not include queueing time. For example, we might have $\mathbf{E}[S] = 0.25$ seconds.

Average service rate This is the average rate, μ, at which jobs are served.

Question: How can we think of μ in terms of what we've already seen?

Answer: By definition, $\mu = \frac{1}{\mathbf{E}[S]} = 4$ jobs/s. Again, we have not said anything about the job size distribution. S might follow a Deterministic job size distribution, where every job has exactly size 0.25 seconds, or, for example, a Pareto distribution, with mean 0.25 seconds.

Observe that this way of speaking is different from the way we normally talk about servers in conversation. For example, nowhere have we mentioned the absolute speed of the server; rather, we have only defined the server's speed in terms of the set of jobs that it is working on.

Imagine that the server is a CPU with a FCFS queue of jobs. In **normal conversation**, we might say:

- The average arrival rate of jobs is 3 jobs per second.
- Jobs have different service requirements, but the average number of cycles required by a job is 2,000 cycles per job.
- The CPU speed is 8,000 cycles per second.
- That is, an average of 6,000 cycles of work arrive at the CPU each second, and the CPU can process 8,000 cycles of work per second.

In the **queueing-theoretic** way of talking, we would never mention the word "cycle." Instead, we would simply say:

- The average arrival rate of jobs is 3 jobs per second.
- The average rate at which the CPU can service jobs is 4 jobs per second.

This second way of speaking suppresses some of the detail and thus makes the problem a little easier to think about. You should feel comfortable going back and forth between the two.

27.3 Kendall Notation

In queueing theory there is a shorthand notation, called Kendall notation, which is used to represent simple commonly seen queueing systems consisting of just a single queue. Kendall notation is in no way expressive enough to represent all queueing systems (in particular, it is not useful for representing systems consisting of multiple queues, like networks), but it is still worth learning this shorthand.

As we saw in the previous section, what's most relevant is the distribution of the interarrival times of jobs and the distribution of the job sizes (service times). In Kendall notation there are typically three slots. The first indicates the distribution of the interarrival times; the second indicates the job size distribution; and the third indicates the number of servers. So, for example, writing:

$$D/\text{Pareto}(\alpha)/1$$

indicates that we're talking about a single-server queue where the interarrival times follow a Deterministic distribution and the job sizes follow a Pareto(α) distribution. Likewise, the notation

$$M/M/1$$

indicates that we're talking about a single-server queue where both the interarrival times and the job sizes are Exponentially distributed; the letter M is reserved for the Exponential distribution, and denotes the Markovian (memoryless) property of that distribution.

One thing that we have not discussed is independence. Kendall notation typically assumes (unless otherwise indicated) that the interarrival times are all independent random variables drawn from some distribution, and likewise that the job sizes are independent random variables drawn from some distribution, and that there is no correlation between interarrival times and the job sizes.

We also have not discussed the scheduling policy. Kendall notation typically assumes FCFS scheduling. If the service order is something other than FCFS, such as Shortest-Job-First, that information is sometimes included in a fourth slot. However we'll see that the fourth slot can be used for other things as well, such as indicating that the buffer capacity (number of jobs that can be in the system) is limited.

Question: Recall that back in Section 26.2, we talked about a queue where time was discretized. At each discrete time step, with probability p a packet (job) arrived, and with probability q a packet (job) departed, if there was a packet there. How can we represent such a system via Kendall notation?

Answer: Geometric(p)/Geometric$(q)/1$.

27.4 Common Performance Metrics

We consider these common **performance metrics** for queueing systems:

- **Response time, time in system, T**: We define a job's response time by

$$T = t_{\text{depart}} - t_{\text{arrive}},$$

where t_{depart} is the time when the job leaves the system, and t_{arrive} is the time when the job arrived to the system. We are interested in $\mathbf{E}[T]$, the mean response time; $\mathbf{Var}(T)$, the variance in response time; and the tail behavior of T, $\mathbf{P}\{T > t\}$.

- **Waiting time, delay, T_Q:** This is the time that the job spends in the queue, not being served. It is also called the "time in queue" or the "wasted time."
- **Number of jobs in the system, N:** The r.v. N includes those jobs in the queues, plus any jobs in service.
- **Number of jobs in queue, N_Q:** The r.v. N_Q denotes only the number of jobs waiting (in queues).

Question: For a single-server queue, with FCFS service order, as in Figure 27.2, what is the relationship between T and T_Q?

Answer: In a single-server queue with FCFS service order, waiting time can be defined as the time from when a job arrives to the system until it first receives service. Here $T = T_Q + S$.

27.4.1 Immediate Observations about the Single-Server Queue

There are some immediate observations that we can make about the single-server queue. First, observe that as λ, the mean arrival rate, increases, all the performance metrics mentioned earlier increase (get worse). Also, as μ, the mean service rate, increases, all the performance metrics mentioned earlier decrease (improve).

We require that $\lambda \leq \mu$ (we always assume $\lambda < \mu$).

Question: If $\lambda > \mu$ what happens?

Answer: If $\lambda > \mu$ then it seems like the queue length grows over time. We refer to this as **instability**. If the queue were represented by a Markov chain, this would be a transient chain.

Question: Can you provide the intuition for why the number of jobs in the system grows over time?

Answer: Consider a large time t. Let $N(t)$ be the number of jobs in the system at time t. Let $A(t)$ (respectively, $C(t)$) denote the number of arrivals (respectively, completions) by time t. Then we have:

$$\begin{aligned}
N(t) &= A(t) - C(t) \\
\mathbf{E}[N(t)] &= \mathbf{E}[A(t)] - \mathbf{E}[C(t)] \\
&\geq \lambda t - \mu t \\
&= t(\lambda - \mu) \\
&\to \infty \text{ as } t \to \infty \qquad\qquad (\text{when } \lambda > \mu\,).
\end{aligned}$$

(The inequality comes from the fact that the expected number of departures by time t is actually smaller than μt, because the server is not always busy.)

Throughout this book, we assume $\lambda < \mu$, which is needed for **stability**, which is defined as keeping queue sizes from growing unboundedly with time. When we later deal with networks of queues, we will also assume stability.

Question: Given the previous stability condition ($\lambda < \mu$), for a $D/D/1$ queue, what is T_Q? What is T?

Answer: $T_Q = 0$, and $T = S$.

Therefore queueing (waiting) results from *variability* in service time and/or interarrival time distributions. Here is an example of how variability leads to queues: Let's discretize time. Suppose at each time step, an arrival occurs with probability $p = 1/6$. Suppose at each time step, a departure occurs with probability $q = 1/3$. Then there is a non-zero probability that the queue will build up (temporarily) if several arrivals occur without a departure.

27.5 Another Metric: Throughput

We have already seen four performance metrics: $\mathbf{E}[N]$, $\mathbf{E}[T]$, $\mathbf{E}\left[N_Q\right]$, and $\mathbf{E}\left[T_Q\right]$. Now we introduce two new performance metrics: throughput and utilization. Throughput is arguably the performance metric most used in conversation. Everyone wants higher throughput! Is higher throughput related to lower response time? Let's see.

Question: In Figure 27.3, which system has higher throughput?

Answer: You might be tempted to think that the top queue has higher throughput, since its server is faster and thus jobs complete more quickly. While the top queue does have lower mean response time, both queues have the *same* throughput.

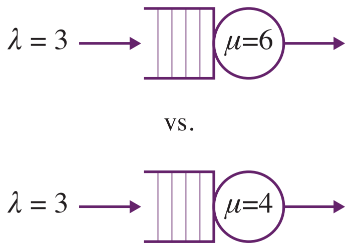


Figure 27.3 *Comparing throughput of two systems.*

Definition 27.1 Throughput, *denoted by X, is the long-run rate of job completions over time. We write:*

$$X = \lim_{t\to\infty} \frac{C(t)}{t},$$

where $C(t)$ denotes the number of completions by time t. Note that X is traditionally capitalized even though it's not a random variable.

Question: What is the throughput, X, of the queues in Figure 27.3?

Answer: Both systems have the same throughput of $X = \lambda = 3$ jobs/s. No matter how high we make μ, the completion rate is still bounded by the arrival rate: "Rate in = Rate out." Changing μ affects the *maximum possible* X, but not the *actual* X.

We now move on to understanding throughput for more complex systems.

27.5.1 Throughput for $M/G/k$

Figure 27.4 illustrates a k-server queueing system with a single shared queue. Whenever a server is free, it picks the job at the head of the queue to work on; if there is no job there, it sits idling until a job arrives. Because there is only one queue, we can describe this with Kendall notation. For example, this might be an $M/G/k$ queue with arrival rate λ jobs per second, where the M indicates that the interarrival times are distributed as $\text{Exp}(\lambda)$. The G here denotes that job sizes are i.i.d. following some general distribution, which we haven't specified. We use r.v. S to denote job size where the service rate at each server, μ, is defined to be $\mu = \frac{1}{\mathbf{E}[S]}$. Here there are k servers, and a single FCFS queue, where the servers all pick their jobs from the same queue.

Question: What condition is needed to keep the system in Figure 27.4 stable?

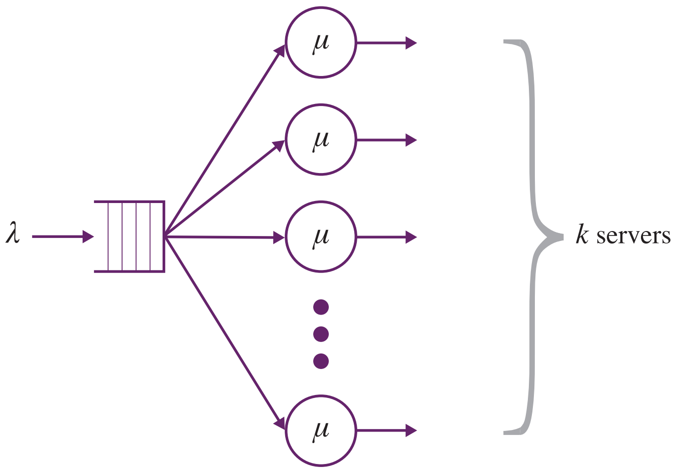


Figure 27.4 *A k-server queueing system.*

Answer: To keep the queue from growing unboundedly, we need to ensure that the total arrival rate of jobs into the system, λ, is less than the total rate at which jobs can leave the system, $k\mu$. So we want $\lambda < k\mu$.

Question: What is the throughput of the system in Figure 27.4?

Answer: Assuming a stable system, what comes in all goes out, so the completion rate is the arrival rate, namely $X = \lambda$.

In today's data centers, it is commonly the case that a *job occupies multiple servers simultaneously* rather than just occupying a single server. Exercise 27.7 examines how the above answers change in these multi-server job settings.

27.5.2 Throughput for Network of Queues with Probabilistic Routing

Figure 27.5 shows a network of queues. Here, server i receives external arrivals ("outside arrivals") with average rate r_i. However, server i also receives internal arrivals from some of the other servers. A job that leaves server i next goes to server j with probability P_{ij}. Server i processes jobs with average rate μ_i.

Note that we have not said anything about the distribution of the interarrival times or the service times, but that won't matter for questions of stability or throughput.

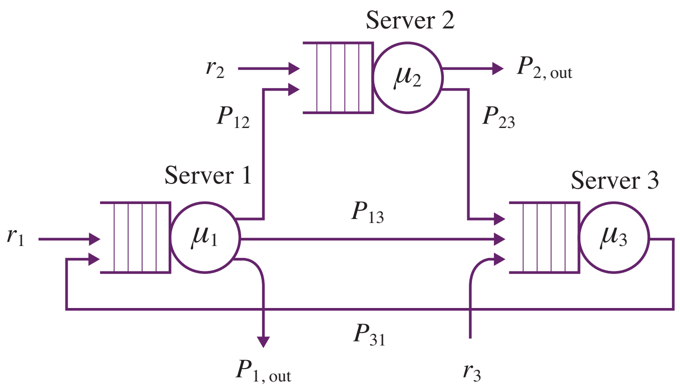


Figure 27.5 *Network of queues with probabilistic routing.*

Question: Assuming that the system is stable, what is the system throughput, X in Figure 27.5?

Answer: All jobs that arrive will also leave, so the rate of departures is the total rate of arrivals, namely: $X = \sum_i r_i$.

Question: What is the throughput at server i, X_i?

Answer: X_i is the rate of completions at server i. Let λ_i denote the *total* arrival rate into server i (including both external and internal arrivals). Then $X_i = \lambda_i$. But to get λ_i we need to solve these simultaneous equations:

$$\lambda_i = r_i + \sum_j \lambda_j P_{ji}. \tag{27.1}$$

Here, r_i denotes the rate of outside arrivals into server i, while $\lambda_j P_{ji}$ denotes the rate of internal arrivals into server i from server j. Note that λ_j here represents the total departure rate from server j (which is equal to the total arrival rate into server j).

Question: How are the r_i's constrained in these equations?

Answer: To maintain stability, we must have $\lambda_i < \mu_i$, $\forall i$, and this constrains the r_i's (see Exercise 27.8).

27.5.3 Throughput for Network of Queues with Deterministic Routing

In the queueing network in Figure 27.6, all jobs follow a predetermined route: CPU to disk 1 to disk 2 to disk 1 to disk 2 to disk 1 and then out.

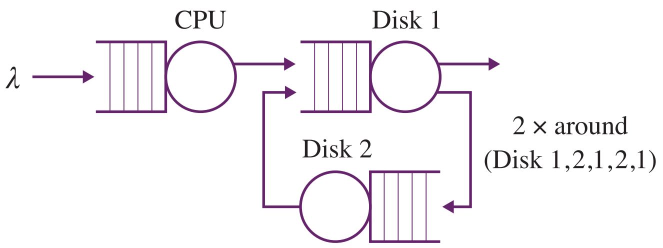


Figure 27.6 *Network of queues with non-probabilistic routing.*

Question: What is X in Figure 27.6?

Answer: $X = \lambda$.

Question: What are X_{Disk1} and X_{Disk2}?

Answer: $X_{\text{Disk1}} = 3\lambda$ and $X_{\text{Disk2}} = 2\lambda$.

27.5.4 Throughput for Finite Buffer

The queue in Figure 27.7 has finite capacity. The outside arrival rate is λ and the service rate is μ. Any arrival that finds no room is dropped.

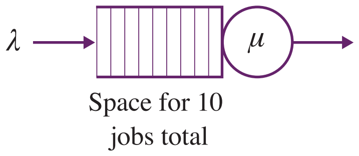


Figure 27.7 *Single-server network with finite buffer capacity.*

Question: What is X?

Answer: Here, $X < \lambda$ because not all arrivals are admitted. The exact formula is $X = \lambda \cdot \mathbf{P}\{\text{job is admitted}\}$.

27.6 Utilization

When we talk about "utilization," we're almost always talking about the utilization of a single device (think single server), which we call "device utilization."

Definition 27.2 *The* **device utilization**, *denoted by ρ, is the long-run fraction of time the device is busy. This is also called the* **device load**.

Think about watching the device for a long period of time, t. Let $B(t)$ denote the total time during the observation period that the device is non-idle (busy). Then

$$\rho = \lim_{t\to\infty} \frac{B(t)}{t}.$$

Question: Looking at the two queues in Figure 27.3, what do you think ρ is for each server?

Answer: Intuitively it seems that $\rho = \frac{3}{6} = \frac{1}{2}$ for the top server, while $\rho = \frac{3}{4}$ for the bottom one. For example, for the bottom queue we imagine that there are 3 jobs coming in per second, and each occupies the server for $\frac{1}{4}$ second on average, so the server is occupied for $\frac{3}{4}$ fraction of each second. This is NOT a proof! In Section 27.11, we will *formally prove* that

$$\rho = \frac{\lambda}{\mu} \tag{27.2}$$

in the case of a single-server queue with arrival rate λ and service rate μ.

Although utilization almost always refers to a single device, if all the devices in a system are homogeneous and receive stochastically the same arrivals, then we often define the **system load** to be the same as a single server load.

For example, in the $M/G/k$ of Figure 27.4, we would say:

$$\text{system load} = \frac{\lambda}{k\mu}. \tag{27.3}$$

To see this, observe that, by symmetry, an individual server receives $\frac{1}{k}$ of the arrivals, on average, so its arrival rate is $\frac{\lambda}{k}$ and its service rate is μ, leading to a server utilization of $\frac{\lambda/k}{\mu} = \frac{\lambda}{k\mu}$. The case where a job occupies multiple servers is not much more complicated, see Exercise 27.7.

27.7 Introduction to Little's Law

Thus far, we have not discussed how one can determine the response time in a queueing system. One way in which this is done is to first represent the queueing system via a Markov chain. Then we solve for the stationary distribution of the Markov chain, which gives us $\mathbf{E}[N]$, the mean number of jobs in the system. We then use a beautiful theorem, called Little's Law, which allows us to convert from $\mathbf{E}[N]$ to $\mathbf{E}[T]$, the mean response time. The purpose of this section is to present and prove Little's Law.

As a side note: Not all queueing systems can be represented easily as Markov chains. The

$$\text{Geometric}(p)/\text{Geometric}(q)/1$$

queue forms a nice discrete-time Markov chain (DTMC), because the Geometric distribution is memoryless. Likewise, the $M/M/1$ queue (for the same reason of memorylessness) can be represented by a Markov chain, but this time a continuous-time Markov chain (CTMC) is required. However, what do we do when the interarrival times or service times are not memoryless? It turns out that one can usually approximate general distributions by mixtures of memoryless distributions, see [35, chpt. 21]. This ends up being very convenient for modeling queueing systems via Markov chains. On the other hand, there are also many techniques for getting to $\mathbf{E}[T]$ without going through a Markov chain.

Little's Law does more than relate $\mathbf{E}[N]$ to $\mathbf{E}[T]$. It provides a formal law for obtaining an average by dividing two averages, a trick that has many applications! One important application of Little's Law is that it allows us to prove the formula for device utilization ($\rho = \frac{\lambda}{\mu}$) that we saw in (27.2).

One of the reasons that Little's Law is so powerful is that it holds for *any ergodic queueing system*, no matter how complex, no matter how many queues, no matter what routing between the queues, no matter what service order within each queue, etc.

Question: What do we mean when we talk about an "ergodic system"?

Answer: Recall that for a Markov chain, we said that the Markov chain is ergodic if it is (1) irreducible, (2) aperiodic, and (3) positive recurrent. These ergodicity properties were sufficient to ensure that the time-averages were equal to their ensemble-average counterparts with probability 1. Even if we're not explicitly talking about a Markov chain, the same points apply. Generally, any "well-behaved" system is ergodic. By "well-behaved" we mean that the system is stable, in that queue lengths do not grow to infinity, and that the mean time between the system emptying is finite (as in positive recurrent).

27.8 Intuitions for Little's Law

Before we state Little's Law, it is worth trying to guess what it might say on your own. It should seem intuitive that $\mathbf{E}[T]$ and $\mathbf{E}[N]$ are proportional. For example, a fast-food restaurant gets people out fast (low $\mathbf{E}[T]$) and also does not require much waiting room (low $\mathbf{E}[N]$). By contrast, a slow-service restaurant gets people out slowly (high $\mathbf{E}[T]$) and therefore needs a lot more seating room (high $\mathbf{E}[N]$). Thus it seems that $\mathbf{E}[T]$ should be directly proportional to $\mathbf{E}[N]$.

Let's see if you can "guess" what it might be, by just looking at a single-server queue. Figure 27.8 shows an illustration of a single-server queue with outside arrival rate λ jobs/s, and mean job size $\mathbf{E}[S] = \frac{1}{\mu}$ seconds/job.

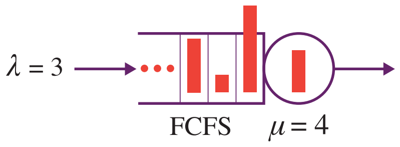


Figure 27.8 *Single-server queue. The height of the rectangle indicates the size of the job.*

Question: Suppose we know the mean number of jobs in this system, $\mathbf{E}[N]$. Is there a way to convert that to the mean response time, $\mathbf{E}[T]$?

Here's a (WRONG) attempt: Let's think of $\mathbf{E}[T]$ as adding up the work in the system as seen by an arrival, where S_i denotes the size of the ith job, maybe something like:

$$\mathbf{E}[T] = \mathbf{E}[N] \cdot \mathbf{E}[S].$$

Intuitively the above attempt seems right because an arrival sees $\mathbf{E}[N]$ jobs, and each of these requires $\mathbf{E}[S]$ service time. However, it is WRONG for several reasons. First of all, N and S are not independent. Second, we're not taking into account the remaining service time on the job in service; remember that the job in service is typically partially complete. Third, this logic in no way generalizes to larger systems with many queues and servers.

The correct answer is:

$$\mathbf{E}[T] = \frac{\mathbf{E}[N]}{\lambda}. \tag{27.4}$$

Question: Can you explain *intuitively* why (27.4) makes sense for a single-server queue?

Answer: Think about a single FCFS queue, as shown in Figure 27.8. From a time-average perspective suppose that there are $\mathbf{E}[N]$ jobs in the system. Now observe that, on average,

$$\mathbf{E}[\text{Time between completions}] = \frac{1}{\lambda},$$

not $1/\mu$, because the average rate of completions is $X = \lambda$ (note that $1/\lambda$ is larger than $1/\mu$). Hence, intuitively, the expected time until the customer leaves is $\mathbf{E}[T] \approx \frac{1}{\lambda} \cdot \mathbf{E}[N]$. This is NOT a proof, only intuition. Theorem 27.3 will give us a proof, and that proof will hold for any network of queues.

27.9 Statement of Little's Law

Theorem 27.3 (Little's Law) *For any ergodic system (including arbitrarily complex networks of queues) we have that:*

$$\mathbf{E}[N] = \lambda \mathbf{E}[T], \tag{27.5}$$

where $\mathbf{E}[N]$ *is the expected number of jobs in the system,* λ *is the average arrival rate into the system, and* $\mathbf{E}[T]$ *is the mean time jobs spend in the system.*

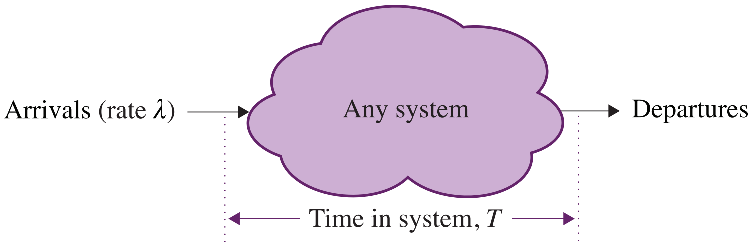


Figure 27.9 *Little's Law is extremely general.*

It is important to note that Little's Law makes no assumptions about the arrival process, the number of servers or queues, the service time distributions at the servers, the network topology, the service order, or anything! Also, since any *portion* of a queueing network is still a queueing network, Little's Law will apply to that portion as well.

Observe that, because we're considering ergodic systems, every job that arrives will complete in finite time (the mean time until the whole system empties is finite), so we know that

$$\lambda = X$$

in Figure 27.9. Here,

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t} \quad \text{and} \quad X = \lim_{t\to\infty} \frac{C(t)}{t},$$

where $A(t)$ is the number of arrivals by time t and $C(t)$ is the number of system completions (departures) by time t.

Restatement via Time Averages

Theorem 27.4 is a restatement of Little's Law in terms of time averages. This is the version that we'll actually be proving.

Theorem 27.4 (Little's Law restated) *Given any system, let $M(s)$ denote the number of jobs in the system at time s. Let T_i denote the response time of the ith arriving job.*

$$\overline{N}^{\textit{Time Avg}} = \lim_{t\to\infty} \frac{\int_0^t M(s)ds}{t} \qquad \textit{and} \qquad \overline{T}^{\textit{Time Avg}} = \lim_{t\to\infty} \frac{\sum_{i=1}^{A(t)} T_i}{A(t)}.$$

For any system where the above limits exist and where $\lambda = X$, then,

$$\overline{N}^{\textit{Time Avg}} = \lambda \cdot \overline{T}^{\textit{Time Avg}}. \tag{27.6}$$

Observe that Little's Law as stated in Theorem 27.4 is an equality between *time averages* on a single sample path, while Little's Law from Theorem 27.3 is an equality between *ensemble averages*.

Question: Does Theorem 27.4 imply Theorem 27.3?

Answer: Yes! Theorem 27.3 assumes ergodicity, which subsumes the assumption that $\lambda = X$, which is needed in Theorem 27.4.[1] As we've seen in Chapters 25 and 26, if we have an *ergodic* system then the time average equals the ensemble average with probability 1. So proving that the time-average equality (27.6) holds suffices to guarantee that the ensemble averages are equal too. The reason we need the stronger assumption of ergodicity in Theorem 27.3 is just to make sure that the ensemble averages exist. Thus, assuming ergodicity, we can apply Little's Law in an ensemble average sense, which is what we do.

27.10 Proof of Little's Law

We are now ready to prove Little's Law.

Proof: [Theorem 27.4] Let T_i denote the time that the ith arrival to the system spends in the system, as shown in Figure 27.10. Thus the rectangle T_i marks the time from when the first job arrives until it completes (this includes time that the job is being served and time that it spends waiting in various queues).

[1] Ergodicity says that the mean time between empties is finite, so clearly every job completes in finite time, so the long-run rate of arrivals and completions converge.

Now, for any time t, consider the area, $\mathcal{A}$, contained within all the rectangles in Figure 27.10, up to time t (this includes most of the rectangle labeled T_5).

The key idea in proving Little's Law is that this area $\mathcal{A}$ is the same, whether we view it by summing *horizontally* or by summing *vertically*. We will first view $\mathcal{A}$ by summing horizontally, and then, equivalently, view it again by summing vertically.

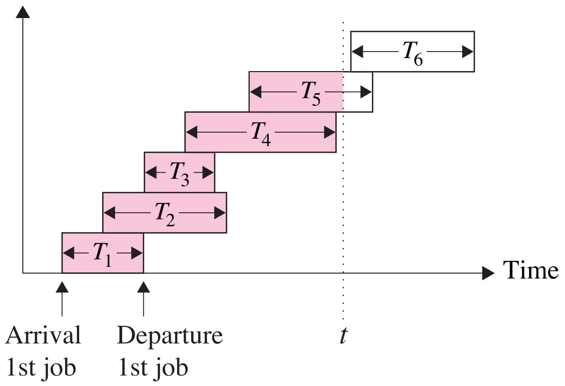


Figure 27.10 *Graph of arrivals.*

The horizontal view consists of summing up the T_i's as follows: We observe that

$$\sum_{i=1}^{C(t)} T_i \le \mathcal{A} \le \sum_{i=1}^{A(t)} T_i,$$

where $\sum_{i=1}^{C(t)} T_i$ denotes the sum of the time in system of those jobs that have completed by time t, and $\sum_{i=1}^{A(t)} T_i$ denotes the sum of the time in system of those jobs that have arrived by time t (a slight abuse of notation).

The vertical view of $\mathcal{A}$ adds up the number of jobs in the system at any moment in time, $M(s)$, where s ranges from 0 to t. Thus,

$$\mathcal{A} = \int_0^t M(s)ds.$$

Combining these two views, we have

$$\sum_{i=1}^{C(t)} T_i \le \int_0^t M(s)ds \le \sum_{i=1}^{A(t)} T_i.$$

Dividing by t throughout, we get:

$$\frac{\sum_{i=1}^{C(t)} T_i}{t} \le \frac{\int_0^t M(s)ds}{t} \le \frac{\sum_{i=1}^{A(t)} T_i}{t},$$

or, equivalently,

$$\frac{\sum_{i=1}^{C(t)} T_i}{C(t)} \cdot \frac{C(t)}{t} \leq \frac{\int_0^t M(s)ds}{t} \leq \frac{\sum_{i=1}^{A(t)} T_i}{A(t)} \cdot \frac{A(t)}{t}.$$

Taking limits as $t \to \infty$,

$$\lim_{t\to\infty} \frac{\sum_{i=1}^{C(t)} T_i}{C(t)} \cdot \lim_{t\to\infty} \frac{C(t)}{t} \leq \overline{N}^{\text{Time Avg}} \leq \lim_{t\to\infty} \frac{\sum_{i=1}^{A(t)} T_i}{A(t)} \cdot \lim_{t\to\infty} \frac{A(t)}{t}$$

$$\Rightarrow \overline{T}^{\text{Time Avg}} \cdot X \leq \overline{N}^{\text{Time Avg}} \leq \overline{T}^{\text{Time Avg}} \cdot \lambda.$$

Yet we are given that X and λ are equal. Therefore,

$$\overline{N}^{\text{Time Avg}} = \lambda \cdot \overline{T}^{\text{Time Avg}}.$$ ■

Question: Are we assuming FCFS service order in this argument?

Answer: No, this argument does not depend on service order. Observe that the second arrival departs *after* the third arrival departs.

Question: Are we assuming anywhere that this is a single-server system?

Answer: No, this argument holds for any system. In fact, Little's Law can also be applied to any part of a system, so long as that part is well behaved (ergodic). We'll see this in Example 27.8.

A final remark on the proof: The proof assumes $\lim_{t\to\infty} \frac{\sum_{i=1}^{C(t)}}{C(t)} = \lim_{t\to\infty} \frac{\sum_{i=1}^{A(t)}}{A(t)}$. To see why, observe that the difference in the numerators is just the total work in the system at time t, which is finite, whereas the denominators grow with time. Thus the difference disappears in the limit as $t \to \infty$.

27.11 Important Corollaries of Little's Law

Corollary 27.5 (Little's Law for time in queue) *Given any system where $\overline{N_Q}^{\text{Time Avg}}$, $\overline{T_Q}^{\text{Time Avg}}$, λ, and X exist and where $\lambda = X$, then*

$$\overline{N_Q}^{\text{Time Avg}} = \lambda \cdot \overline{T_Q}^{\text{Time Avg}},$$

where N_Q represents the number of jobs in the queues in the system and T_Q represents the time jobs spend in queues.

Question: How would you prove Corollary 27.5?

Answer: Same proof as for Theorem 27.4, except that now instead of drawing T_i, we draw $T_Q^{(i)}$, namely the time the ith arrival to the system spends in queues (wasted time). Note that $T_Q^{(i)}$ may not be a solid rectangle. It may be made up of several rectangles because the ith job might be in a queue for a while, then in service, then waiting in some other queue, then in service again, etc.

Corollary 27.6 (Utilization Law) *Consider a single device i with its own queue, possibly within a network of queues. Suppose that the average arrival rate into device i is λ_i jobs/s and the average service rate of device i is μ_i jobs/s, where $\lambda_i < \mu_i$. Let ρ_i denote the long-run fraction of time that device i is busy. Then,*

$$\rho_i = \frac{\lambda_i}{\mu_i}.$$

We refer to ρ_i as the "device utilization" or "device load."

Proof:

Question: Do you see where to apply Little's Law to queue i?

Hint: What should the "system" be for applying Little's Law?

Answer: Let the "system" consist of just the "service facility" (the server part *without* the associated queue), as shown in the shaded box of Figure 27.11. Now the number of jobs in the system is always just 0 or 1.

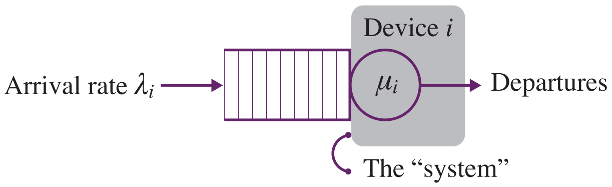


Figure 27.11 *Using Little's Law to prove the Utilization Law.*

Question: What is the expected number of jobs in the system as we have defined it?

Answer: The number of jobs in the system is 1 when the device is busy (this happens with probability ρ_i) and is 0 when the device is idle (this happens with probability $1 - \rho_i$). Hence the *expected* number of jobs in the system is ρ_i. So, applying Little's Law, we have:

$$\begin{aligned}\rho_i &= \text{Expected number jobs in service facility for device } i\\ &= (\text{Arrival rate into service facility}) \cdot (\text{Mean time in service facility})\\ &= \lambda_i \cdot \mathbf{E}\left[\text{Service time at device } i\right]\\ &= \lambda_i \cdot \frac{1}{\mu_i}.\end{aligned}$$ ■

We often express the Utilization Law as:

$$\boxed{\rho_i = \frac{\lambda_i}{\mu_i} = \lambda_i \mathbf{E}\left[S_i\right] = X_i \mathbf{E}\left[S_i\right],}$$

where ρ_i, λ_i, X_i, and $\mathbf{E}\left[S_i\right]$ are the load, average arrival rate, average throughput, and average service requirement at queue i, respectively.

Suppose we have some arbitrary network of queues. We'd like to again relate $\mathbf{E}\left[T\right]$ to $\mathbf{E}\left[N\right]$ for the system. However, we are only interested in "red" jobs, where "red" denotes some type of job. Specifically, we'd like to understand how $\mathbf{E}\left[N_{\text{red}}\right]$, the mean number of red jobs in the system relates to $\mathbf{E}\left[T_{\text{red}}\right]$, the mean response time of red jobs.

Question: Suppose we want to apply Little's Law to just a particular class of jobs, say the "red" jobs. Can we do this?

Answer: Yes.

Theorem 27.7 (Little's Law for red jobs) *For any ergodic system we have that:*

$$\mathbf{E}\left[N_{red}\right] = \lambda_{red}\mathbf{E}\left[T_{red}\right],$$

where $\mathbf{E}\left[N_{red}\right]$ is the expected number of red jobs in the system, λ_{red} is the average arrival rate of red jobs into the system, and $\mathbf{E}\left[T_{red}\right]$ is the mean time that red jobs spend in the system.

Proof: The proof is exactly the same as for Little's Law, but only the T_i's corresponding to the red jobs are included in Figure 27.10. ■

Example 27.8 (Repair center)

Repairs don't always work. In Jenny's repair center, shown in Figure 27.12, every arriving item undergoes a "repair attempt," but with probability 0.9 the item needs to go in for another round. We say that the total time for repair, T, is the time from when the item first arrives until it is fully repaired. Based on

Jenny's measurements, on average, $\lambda = 2$ items arrive to the repair center every hour, the average repair attempt takes $\mathbf{E}[S] = 2$ minutes, and $\mathbf{E}[T] = 10$ hours.

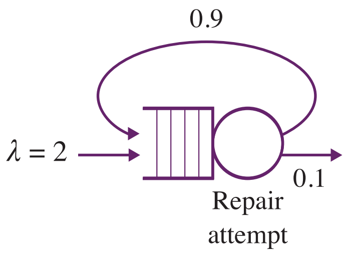


Figure 27.12 *Jenny's repair center.*

Question: What fraction of time is the repair center busy?

Answer: To answer this, we draw a gray box around just the server, as shown in Figure 27.13.

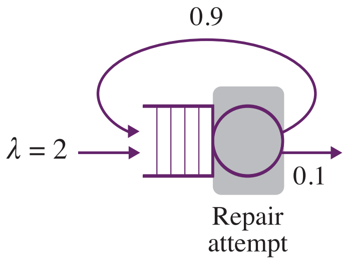


Figure 27.13 *The "system" here is the gray box.*

Applying Little's Law to this gray box system, we have:

$$\mathbf{E}[N_{\text{box}}] = \lambda_{\text{box}} \cdot \mathbf{E}[T_{\text{box}}].$$

Observe that $\mathbf{E}[N_{\text{box}}] = \rho$. Furthermore, $\mathbf{E}[T_{\text{box}}] = \mathbf{E}[S] = \frac{2}{60}$ hours. To get λ_{box}, we note that, on average, an item requires 10 repair attempts. Hence

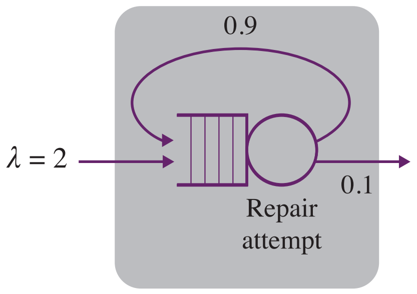
$$\lambda_{\text{box}} = 10 \cdot \lambda = 20 \text{ items/hour.}$$

Little's Law thus yields:

$$\begin{aligned} \mathbf{E}[N_{\text{box}}] &= \lambda_{\text{box}} \cdot \mathbf{E}[T_{\text{box}}] \\ \rho &= 20 \cdot \frac{2}{60} = \frac{2}{3}. \end{aligned}$$

Question: What is the expected number of items in the repair center, $\mathbf{E}[N]$?

Hint: This can be solved in two different ways, depending on how we define our system of interest in Figure 27.14.



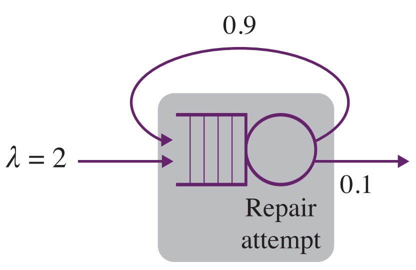


Figure 27.14 *Two different views of the "system," both with the same* $\mathbf{E}[N]$.

Answer: If we draw our gray box around the entire system, as shown in Figure 27.14(a), then $\lambda_{\text{box}} = \lambda = 2$. This yields:

$$\begin{aligned}\mathbf{E}[N_{\text{box}}] &= \lambda_{\text{box}} \cdot \mathbf{E}[T_{\text{box}}]\\ \mathbf{E}[N] &= \lambda \cdot \mathbf{E}[T]\\ &= 2 \cdot 10 = 20 \text{ items.}\end{aligned}$$

On the other hand, if we draw our gray box around a single attempt, as shown in Figure 27.14(b), then $\lambda_{\text{box}} = 10\lambda$. However, $\mathbf{E}[T_{\text{box}}]$ is only $\frac{\mathbf{E}[T]}{10}$ since there are 10 attempts on average. This yields:

$$\begin{aligned}\mathbf{E}[N_{\text{box}}] &= \lambda_{\text{box}} \cdot \mathbf{E}[T_{\text{box}}]\\ \mathbf{E}[N] &= (10\lambda) \cdot \mathbf{E}\left[T_{\text{attempt}}\right]\\ &= (10 \cdot 2) \cdot \frac{\mathbf{E}[T]}{10}\\ &= 20 \cdot \frac{10}{10} = 20 \text{ items.}\end{aligned}$$

Unsurprisingly, the answer is the same, since Little's Law applies to any system or portion of a system.

27.12 Exercises

27.1 **Professors and students**
A professor practices the following strategy with respect to taking on new PhD students. On the even-numbered years, she takes on two new PhD students. On the odd-numbered years, she takes on one new PhD student.

All students graduate and the average time to graduate is six years. How many PhD students on average will the professor have in her group? Prove your answer.

27.2 **Professors and students, revisited**
A professor practices the following strategy with respect to taking on new PhD students. On the even-numbered years, she takes on two new PhD students. On the odd-numbered years, she takes on one new PhD student. Not all students graduate. Of the students whom the professor takes on, $\frac{1}{3}$ end up staying for one year on average and then leave the program; $\frac{1}{6}$ graduate after four years; $\frac{1}{6}$ graduate after five years; $\frac{1}{6}$ graduate after six years; and $\frac{1}{6}$ graduate after seven years. How many PhD students on average will the professor have in her group? Prove your answer.

27.3 **Mean response time at router with infinite capacity**
Recall in Section 26.2 we derived the mean number of packets in a router with infinite capacity, where at every time step, with probability $p = \frac{1}{4}$ one packet arrives, and, independently, with probability $q = \frac{1}{3}$ one packet departs. What is the mean response time, $\mathbf{E}[T]$, for this particular system?

27.4 **Mean response time at router with finite capacity**
As in Exercise 27.3, we return to the router in Section 26.2, but this time, the router only has room for a total of 3 packets. Specifically, if a packet arrives when the state of the DTMC is 3, then the packet is dropped and the state of the system doesn't change. What is the mean response time, $\mathbf{E}[T]$, of those packets that are *not* dropped. [Hint: To create an ergodic system, you'll want to think about the "system" as consisting only of the arrivals that enter.]

27.5 **The single-server queue**
Kunhe's system consists of a single-server queue. Based on Kunhe's measurements, the average arrival rate is $\lambda = 5$ jobs/s; the average job size is $\mathbf{E}[S] = 0.1$ s; and the average number of jobs is $\mathbf{E}[N] = 10.5$ jobs.
(a) What is the fraction of time that Kunhe's server is busy?
(b) What is the average time that jobs spend queueing in Kunhe's system, $\mathbf{E}[T_Q]$?

27.6 **The Arthur Ravenel bridge**
The Arthur Ravenel bridge in Charleston allows walkers and joggers to get to downtown Charleston. During my visit to Charleston, I observed that:

- On average, 20 walkers arrive per hour and take an average of 1 hour to cross the bridge.

- On average, 10 joggers arrive per hour and take an average of 20 minutes to cross the bridge.

Based on this data, estimate the average number of people (walkers plus joggers) on the bridge at any time.

27.7 **Data center utilization**
The Clouds-R-Us company runs a data center with 10,000 servers shown in Figure 27.15. Jobs arrive to the data center with average rate $\lambda = 2$ jobs/s. Each job requires some number of servers K, where $K \sim \text{Binomial}(1000, 0.05)$. The job holds onto these K servers for some time S seconds, where $S \sim \text{Exp}(0.02)$, and then releases all its servers at once. Assume that K and S are independent. Jobs are served in FCFS order. If a job gets to the head of the queue, but the number of servers that it needs exceeds the number of idle servers, then the job simply waits (blocking those jobs behind it in the queue) until that number of servers becomes available. You may assume that the system is ergodic. On average, how many jobs are running at a time?

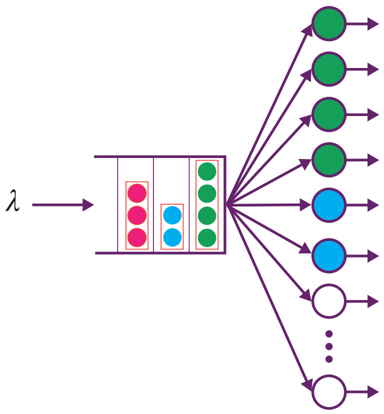


Figure 27.15 *Data center for Exercise 27.7.*

27.8 **Maximum outside arrival rate**
For the network-of-queues with probabilistic routing given in Figure 27.5, suppose that each server serves at an average rate of 10 jobs/s; that is, $\mu_i = 10, \forall i$. Suppose that $r_2 = r_3 = 1$. Suppose that $p_{12} = p_{2,out} = 0.8$, $p_{23} = p_{13} = 0.2$, $p_{1,out} = 0$, and $p_{31} = 1$. What is the maximum allowable value of r_1 to keep this system stable?

27.9 **Simplified power usage in data centers**
Given that power is expensive, it is common practice to leave servers on only when they are being used and to turn them off whenever they are

not in use. Assume that the following power-aware algorithm is used: When a job arrives, it instantly turns on a fresh server (assume zero setup cost). When the job completes service, it instantly turns off that server. Assume that there is always a server available for every job (i.e., there is no queueing). Your goal is to derive the time-average rate at which power is used in our system. Assume that when a server is on, it consumes power at a rate of $\mathcal{P} = 240$ watts. Assume $\lambda = 10$ jobs arrive per second and that the service requirement of jobs is Uniformly distributed, ranging from 1 second to 9 seconds.

27.10 **Going to the DMV**

When getting your driver's license at the DMV, you have to pass through two stations: the photo-taking station and the license-creation station. Unfortunately, at the end of license-creation, with probability 25% they find something wrong with the photo, and the whole process has to start all over again. Figure 27.16 shows the process. As shown in the figure, the average arrival rate of people to the DMV is $r = 15$ people per hour, the average number of people in the photo station is 10, and the average number in the license station is 20. Assume that the system is stable in that the total arrival rate into each station is less than the service rate at the station. Derive the mean time from when you walk into the DMV until you walk out with your driver's license.

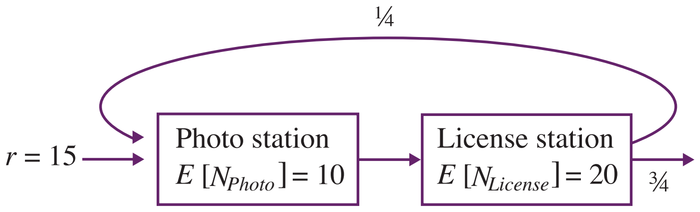


Figure 27.16 *The DMV process for Exercise 27.10.*

27.11 **Network that looks like a flip flop**

Tianxin's network, shown in Figure 27.17, looks like a flip flop.

Jobs arrive to Tianxin's network at a rate of $r = 1$ jobs per second. The routing probabilities are shown. The service rate at station A is $\mu_A = 3$ jobs per second, and that at station B is $\mu_B = 4$ jobs per second. An individual job might pass through Station A, then B, then A, then B, etc., before it eventually leaves. Tianxin has observed that the expected number of jobs at station A is $\mathbf{E}[N_A] = 2$ and the expected number of jobs at station B is $\mathbf{E}[N_B] = 1$.

(a) Let T denote the response time of a job, i.e., the time from when it arrives until it departs. What is $\mathbf{E}[T]$?

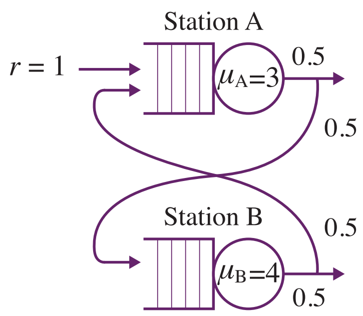


Figure 27.17 *Tianxin's network for Exercise 27.11.*

(b) Let λ_A denote the total arrival rate into station A. Let λ_B denote the total arrival rate into station B. What are λ_A and λ_B?

(c) What is the throughput of the system? What is the throughput of station A? Which is higher?

(d) Let T_A denote the time it takes for a job to make a *single* visit to station A (this includes queueing and then serving at station A). Likewise, let T_B denote the time it takes for a job to make a *single* visit to station B. What are $\mathbf{E}[T_A]$ and $\mathbf{E}[T_B]$?

(e) Let T_Q denote the total time that a job spends queueing while in the system. This includes the total time that the job is in queues from when it arrives until it leaves the system. What is $\mathbf{E}\left[T_Q\right]$?

27.12 **Finally a haircut!**

For over a year in lockdown, I haven't been able to get my hair cut, but finally I can return to the salon! At my salon there are two stations: the washing station and the cutting station, each with its own queue. The people who work at the washing station only wash hair. The people who work at the cutting station only cut hair. When a washing person frees up, they take the next person in the wash line, and similarly for the cutting line.

My salon is very quick. The average wash time is only $\mathbf{E}[S_{\text{wash}}] = \frac{1}{13}$ hours and the average cut time is only $\mathbf{E}[S_{\text{cut}}] = \frac{1}{7}$ hours. Unfortunately, they're so quick that they sometimes forget to rinse the shampoo, so, with probability $\frac{1}{4}$, I will need to rejoin the wash line after my wash is complete.

There are two types of customers at my salon: the "wash-and-cut" customers, who get their hair washed and then cut, and the "cut-only" customers who only get their hair cut (no wash).

Figure 27.18 shows the salon. Assume that 54 customers enter the salon per hour. Assume that $\frac{2}{3}$ are wash-and-cut customers and $\frac{1}{3}$ are cut-only customers.

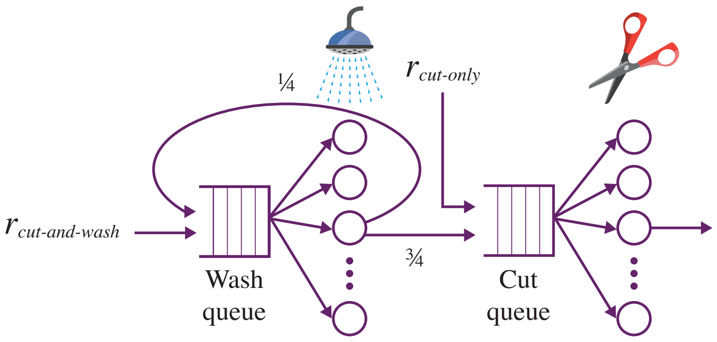


Figure 27.18 *Hair salon for Exercise 27.12.*

(a) What is the bare minimum number of staff (washers + cutters) that are needed to ensure that the hair salon is stable?

(b) On average, the number of customers at the wash station (either in the wash queue or having their hair washed) is 9. On average, the number of customers at the cutting station (either in the cut queue or having their hair cut) is 18.

  (i) What is the expected response time of a random customer (we're not told the type of customer)?

  (ii) What is the expected response time of a cut-only customer? [Hint: Think about the experience of a cut-only customer.]

27.13 **Little's Law and the vaccine center**

Our local COVID vaccine center is structured as a multi-server queue, where there are five nurses providing vaccines, and a queue of patients waiting to receive vaccines, as in Figure 27.19. When a patient is getting their vaccine, they sit in one of the five chairs. Due to social distancing rules, there is a limit of 25 on the total number of people allowed in the vaccine center (this includes the five nurses). There is an overflow queue of patients outside the vaccine center, waiting to get in. The total number of patients, N, grows and shrinks over time, but, on average there are $\mathbf{E}[N] = 80$ patients in total (including both the vaccine center and the overflow queue). The long-run average rate of patients joining the queue is $\lambda = 40$ patients per hour.

(a) What is the expected total response time (from arrival to departure)?

(b) Suppose we model $N \sim \text{Geometric}\left(\frac{1}{80}\right)$. Let N_{center} denote the number of people inside the vaccine center (gray area). Let T_{center} denote the time spent inside the vaccine center. What is $\mathbf{E}[T_{\text{center}}]$?
[Hint: Start by expressing N_{center} in terms of N. You will need a "min" term. Then derive $\mathbf{E}[N_{\text{center}}]$ and apply Little's Law.]

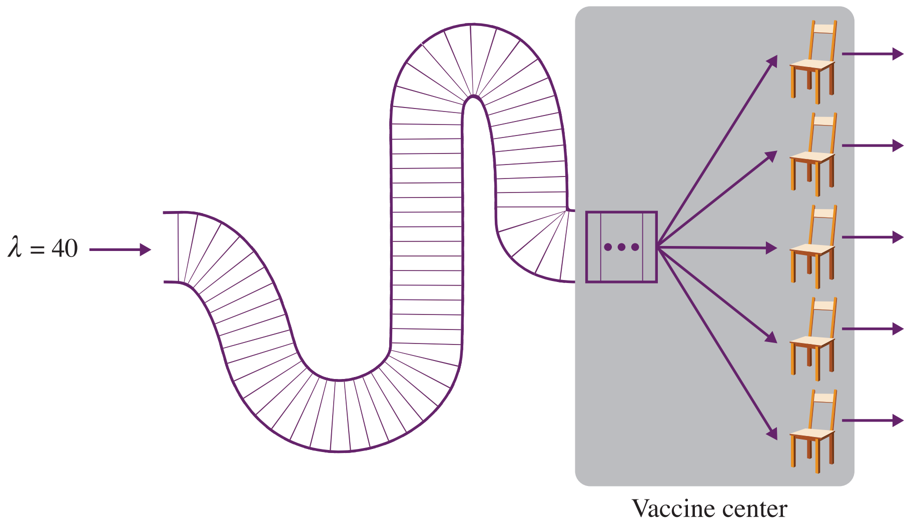


Figure 27.19 *Vaccine center.*

27.14 **Mean slowdown**

The slowdown metric is related to response time, but is somewhat more practical. A job's slowdown is defined as its response time divided by its size:

$$\text{Slowdown of job } j = \frac{\text{Response time of } j}{\text{Size of } j}. \tag{27.7}$$

The idea is that large jobs (like downloading a whole movie) can tolerate larger response times (while you go make popcorn), while small jobs (like downloading a web page) can tolerate only very small response times. The slowdown metric captures this tolerance better than response time.

(a) Jobs arrive at a server that services them in FCFS order (Figure 27.20).

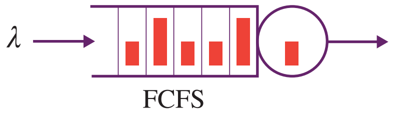


Figure 27.20 *Figure for Exercise 27.14.*

The average arrival rate is $\lambda = \frac{1}{2}$ job/s. The job sizes (service times) are i.i.d. and are represented by r.v. S, where

$$S = \begin{cases} 1 & \text{w/prob } \frac{3}{4} \\ 2 & \text{otherwise} \end{cases}.$$

You have measured the mean response time, $\mathbf{E}[T] = \frac{29}{12}$. Based on this information, compute the mean slowdown, $\mathbf{E}[\text{Slowdown}]$.

(b) If the service order in part (a) had been Shortest-Job-First, would the same technique have worked for computing mean slowdown?