CHAPTER 1

Introduction: Two Examples

To help illustrate the emphasis and tenor of this book, we begin with two examples. Although they are very different from each other, they show how quantitative models and analyses provide crucial insights into the behavior and performance of a packet network subsystem. In both cases, they helped to determine the direction of the research and development efforts on these subsystems.

1.1 Efficient Transport of Packet Voice Calls

A packet voice communication system is one in which the voice signal is digitized, the bits are grouped into packets and the packets transported from the source to the destination. A call made using such a system is called a packet voice call.

Figure 1.1 depicts a situation in which a communication link between two locations must carry several packet voice calls. The speed of the link is C (bits per second). Associated with each telephone instrument is a voice digitizer, coder, and packetizer. A random stream of voice packets from each call enters a communication switch (or *packet router*), at which point the packets are *multiplexed* into the link. At the other end of the link, a *router* extracts (or *demultiplexes*) the packets from the link and routes them to devices that *depacketize* and decode the voice. A mirror image of this path is followed by voice in the reverse direction. The objective of the system designer is that, given a link of speed C, the number of calls that can be simultaneously carried should be maximized subject to a constraint on voice quality.

There are two aspects to voice quality: distortion and delay. The voice coder is assumed to perform *embedded* (or *layered*) coding. Thus, if the peak coding rate is, say, 4 bits per sample, and if every packet is delivered intact, then essentially distortion-free (telephone-quality) speech is heard by the listener. On the other hand, embedded coding permits, for example, 1 bit per sample to be dropped (in an appropriate way) from each voice packet to obtain a coded voice stream with

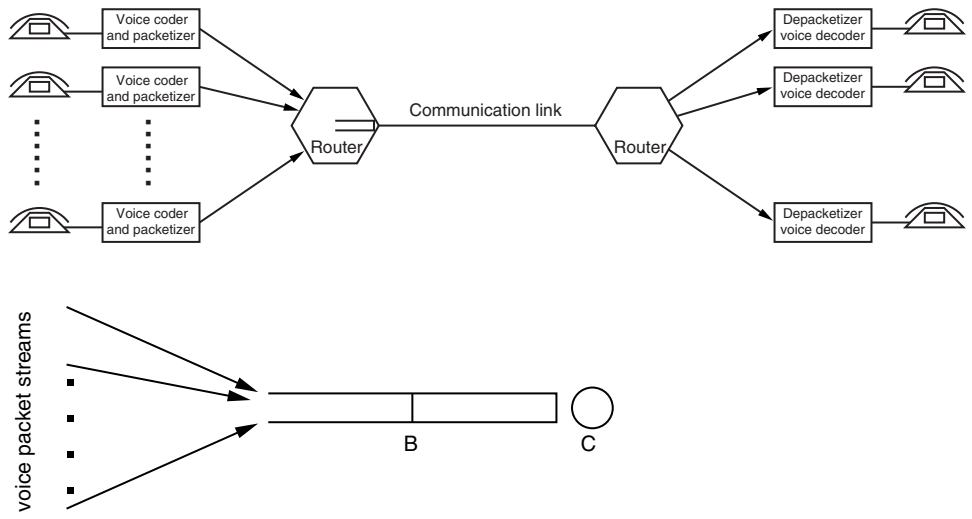

Figure 1.1 An interlocation link is used to carry packet voice calls (top). The LAN-WAN packet router on the left multiplexes several digitized, coded, and packet voice sources into the link, and the router on the right extracts packets from the link. On the bottom left is a queuing model for studying the performance of packet voice multiplexing.

3 bits per sample. This facility can be used to reduce the traffic, thus permitting more calls to be carried simultaneously. The quality of the received speech is acceptable, however, only if at least a fraction α of the coded bits are delivered (for example, with $\alpha = 0.95$ and 4 bits peak rate per sample, at least 3.8 bits per sample would need to be delivered). Because the arrivals of packets into the link are random, queuing of packets will necessarily occur, leading to delays. It is very difficult to carry out an interactive conversation when there is substantial delay. In practice, there are several components of the end-to-end delay (see Chapter 3), but here we focus only on the delay in the multiplexer. We can specify the multiplexer delay requirement in terms of a level B (bits) that should be exceeded only rarely in the multiplexer buffer (see Figure 1.1); this says that we desire the delay bound of $\frac{B}{C}$ to be rarely exceeded.

In this problem setting we have two design alternatives.

1. *Bit-dropping at the multiplexer:* The speech is coded at the full rate. When a packet arrives in the multiplexer buffer, if accepting this packet would cause the buffer level to exceed B, some bits are dropped from the arriving and

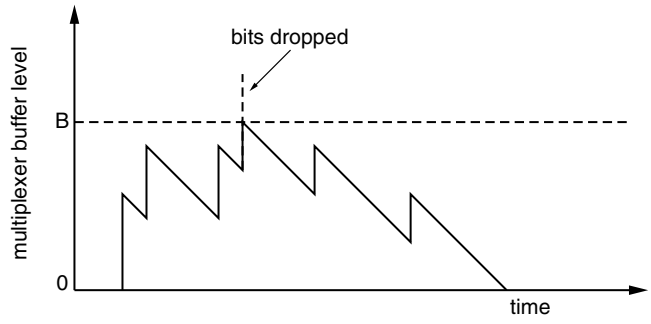

Figure 1.2 Evolution of the multiplexer buffer level with bit dropping control. At a packet arrival instant the buffer increases by the number of bits from the packet that are allowed into the buffer. The buffer decreases at the link rate *C* in between the packet arrival instants.

existing packets so that after accepting the new packet, the buffer level is exactly B (see Figure 1.2). To smooth out the effect of lower-rate coding, the bit-dropping can be evenly distributed over all the packets involved (to the extent possible, given the discrete nature of bits). We can then ask the question, what is the maximum number of voice calls that can be simultaneously handled by the link so that the system delivers at least a fraction α of the coded bits? Another way to implement such queue-length-dependent bit-dropping is to provide a signaling mechanism between the multiplexer and the coder; it is the coder that drops the bits based on the buffer level signaled by the multiplexer. If the signaling delay is zero and if only bits from the arriving packet are dropped, then this *buffer-adaptive coding* would be equivalent to our description of dropping bits at the multiplexer. Note that because the state of the system (namely, the queue length) is used to determine the instants and the amounts of bit dropping, this is *closed-loop* control.

2. *Lower-bit-rate coding at the source coder:* It can be argued that, because the application can tolerate a lower bit rate, the embedded coder can itself just send a fraction α of the peak number of bits per sample. The link must then carry all the bits of all the packets that it receives, and hence bit-dropping control is not possible in the multiplexer. Owing to randomness in the arrival process, however, the buffer can build up to beyond B. The question in this case is, what is the maximum number of voice calls that can

be simultaneously handled by the link so that the probability (fraction of time) of exceeding the buffer level B in the multiplexer is less than a small number ϵ—say, 0.001? We can view this as *open-loop* control because the coders reduce the bit rate without any consideration of the multiplexer queue length.

To analyze these two alternatives, we will make the simplifying assumption that the random process of arrival instants of packets into the multiplexer is Poisson (with rate λ) and that the packet lengths are continuous and exponentially distributed, with (*full-rate-coded*) mean $\frac{1}{\mu}$. Such a simplified model is sufficient for our present purpose of providing some insights into the problem. Analysis, albeit more complex, can also be carried out with more general assumptions (see Chapter 5).

We will assume that the mean packet size is fixed (a characteristic of the speech and the coder), whereas the packet arrival rate λ relates to the number of voice calls. The total, uncontrolled arrival rate of bits is $\frac{\lambda}{\mu}$. Define $\rho := \frac{\lambda}{C\mu}$, which is the total arrival rate of (full-rate) bits from the sources, normalized to the link speed. It is clear that it is sufficient to determine the maximum value of ρ that can be supported in each of the two alternatives.

The model can be analyzed to obtain the distribution of the steady state multiplexer buffer level for the bit-dropping multiplexer, and for lower-bit-rate coding at the source. In each case the probability distribution has a continuous part with a density and also has a point mass at 0. Let us denote by $f(\cdot)$ the density function, and by z the point mass at 0. The following results can be obtained quite easily by the technique of *level crossing analysis* (see Appendix D, Section D.1.9, where the following equations are derived).

The stationary distribution of the buffer level with bit-dropping is supported on the interval $[0, B]$. For $0 < x < B$, we have

$$f_{bit\text{-}dropping}(x) = \frac{\mu(1-\rho)\rho}{1-\rho e^{-\mu(1-\rho)B}}\, e^{-\mu(1-\rho)x} \tag{1.1}$$

and

$$z_{bit\text{-}dropping} = \frac{(1-\rho)}{1-\rho e^{-\mu(1-\rho)B}} \tag{1.2}$$

Because the link is sending bits whenever the buffer is not empty, the rate at which the multiplexer is putting bits through is $(1-z)C$. Hence the fraction of the

arriving bits that are carried is given by $\frac{(1-z)C}{(\lambda/\mu)} = \frac{1-z}{\rho}$. With bit-dropping at the multiplexer, the fraction of offered bits that are carried by the multiplexer should be at least α (e.g., 0.95). Using the formula for z, this requirement translates to the following requirement:

$$\frac{1 - e^{-\mu(1-\rho)B}}{1 - \rho e^{-\mu(1-\rho)B}} \geq \alpha \tag{1.3}$$

Note that, by definition, a bit-dropping multiplexer bounds the delay to the time taken to serve B bits.

Let us define the maximum load that can be offered to a bit-dropping multiplexer, so that the average bits per sample objective is met, by

$$\rho^{\max}_{bit\text{-}dropping} := \sup \left\{ \rho \colon 0 \leq \rho \leq 1, \frac{1 - e^{-\mu(1-\rho)B}}{1 - \rho e^{-\mu(1-\rho)B}} \geq \alpha \right\}$$

The stationary distribution of the buffer level with lower-rate-source coding is supported on $[0, \infty)$. For $x \geq 0$, we have

$$f_{low\text{-}rate\text{-}coding}(x) = \mu(1 - \alpha\rho)\rho e^{-\frac{\mu}{\alpha}(1-\alpha\rho)x} \tag{1.4}$$

and

$$z_{low\text{-}rate\text{-}coding} = 1 - \alpha\rho \tag{1.5}$$

This result requires that $\alpha\rho < 1$, and the distribution can be obtained from the one for bit-dropping as follows. Because a fraction α of the bits are retained by the source, the mean packet length is $\frac{\alpha}{\mu}$. Furthermore, we cannot drop any more bits in the multiplexer; this is equivalent to letting $B \to \infty$. Now consider a bit-dropping multiplexer with packet arrival rate λ and mean packet length $\frac{\alpha}{\mu}$. The preceding distribution will be obtained from Equations 1.1 and 1.2 by letting $B \to \infty$.

We, however, still need to meet the multiplexer delay objective. The requirement that the probability of the buffer level exceeding B be less than ϵ leads to the following inequality.

$$\alpha\rho e^{-\frac{\mu}{\alpha}(1-\alpha\rho)B} \leq \epsilon \tag{1.6}$$

Define the maximum voice load that can be offered to the multiplexer (after lowering the coding rate by a fraction α) by

$$\rho^{\max}_{low\text{-}rate\text{-}coding} := \sup\left\{\rho\colon 0 \le \rho \le 1, \alpha\rho e^{-\frac{\mu}{\alpha}(1-\alpha\rho)B} \le \epsilon\right\}$$

In Figure 1.3 we plot $\rho^{\max}_{bit\text{-}dropping}$ and $\rho^{\max}_{low\text{-}rate\text{-}coding}$ as a function of B normalized to the mean packet size $\frac{1}{\mu}$ (i.e., we plot the maximum load versus μB). Note that these plots depend on only two parameters—namely, α and ϵ—both specifying the performance objectives.

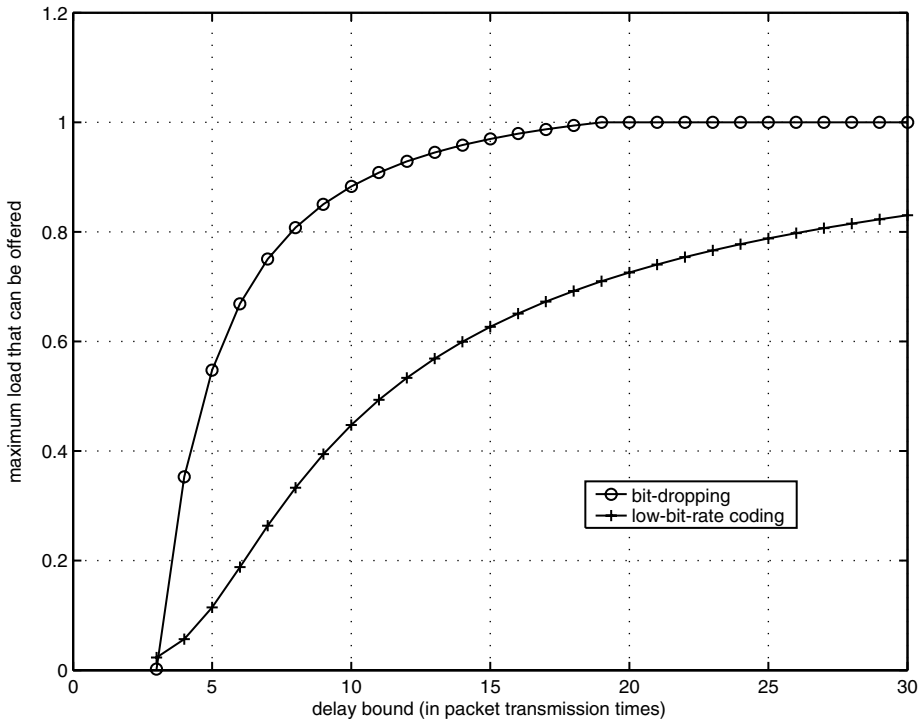

Figure 1.3 Maximum offered load that meets the performance objectives for bit-dropping and low-bit-rate coding, plotted versus the delay objective expressed as μB. The parameters are $\alpha = 0.95$, and $\epsilon = 0.001$.

The plots in Figure 1.3 show that bit-dropping (or closed-loop control) can permit the system to carry significantly more load than simply coding the voice at just enough bits per sample (open-loop control). For example, for $\mu B = 10$ (i.e., a delay bound of 10 packet service times), the bit-dropping approach will carry twice as many voice calls as the open-loop approach. The large difference is significant. The advantage of closed-loop control can be intuitively explained as follows. The closed-loop control intelligently drops bits only when there is need to, that is, when the delay threatens to exceed the target delay. On the other hand, the open-loop control naively drops all the bits it possibly can during coding itself. The variability in the buffer process remains, and there is no more leverage in the multiplexer to take any action to control congestion.

This simplified model has glossed over several practical issues (for example, the perceptual effect of variability in the coding rate as a result of bit-dropping), but the insights gained are remarkable. A simple model, along with quick analysis, has taught us a lot. In the engineering process, more detailed modeling and simulation experiments then take our understanding further, all the way to a complete design. (See [271] for an implementation example.) Such a complete design would yield a multiplexing and control strategy that would maximize the rate at which calls could be offered to the system, while meeting the quality of service objectives (bits per sample, packet delay, and call blocking).

1.2 Achievable Throughput in an Input-Queueing Packet Switch

As a second example, consider a packet switch that is at the heart of most high-speed routers, such as the Cisco 12000 series of Internet routers. Such a high-speed packet switch is typically designed to handle fixed-length packets, also called *cells*. Cell *switches* are usually operated in a time-slotted manner; time is divided into *slots*, and the activities at all the ports—such as cell arrivals from the input links, switching of the cells to the respective outputs, and departures on the output links—are synchronized to the slot boundaries. Each slot duration is equal to the cell transmission time. Consider such an $N \times N$ cell switch having N input ports and N output ports. Let the transmission rates on the input and output links be equal. Let d_i denote the destination of the cell in input i at the beginning of a slot. Assume that the switch is *nonblocking*, meaning that if $[d_1, d_2, \ldots d_N]$ is a permutation of $[1, 2, \ldots N]$, then the cell at input i can be sent to output d_i in that slot, for all i. The design of nonblocking switches is well understood and was originally considered in the context of telephone switches and in interconnection

networks used for processor and memory subsystem communication in parallel computing systems.

Now consider the operation of a cell switch. In each slot, up to one cell can be received on each of the N input links, and the cell can have any of the N output links as its destination. Thus it is possible that in a slot, more than one cell with the same destination can arrive at the inputs, causing *destination conflicts*. Because the output link rates are equal to the input link rates, in each slot at most one cell can be transmitted on each of the output links. So how do we handle the excess (more than one) cells destined for the same output that may have arrived in the slot? Obviously, dropping them is not a good idea; instead, they need to be buffered or queued. But where? We can see that the queue can be placed either at the output ports or at the input ports, and accordingly we will have an *output-queued* (OQ) or an *input-queued* (IQ) switch.

In the OQ switch, all the cells destined to a port are switched to the output in the same slot, and one of them is transmitted on the output link if no other packet from a previous slot is waiting. Clearly, the OQ switch can provide 100% throughput. In other words, if the average rate at which each output receives cells is less than the output link rate, then the queue will not grow in an unbounded manner. However, by definition, the OQ switch should be capable of switching up to N cells to an output in a slot. More importantly, the memory that is used for the output queue should be capable of supporting such a read-write speed. Improvements in memory access speed have not kept pace with that of processor technologies or even with the size of memory. Thus the OQ switch does not scale very well with N. This means that it might be feasible for small N, but as N becomes large, especially if the transmission rates are high, it can become technologically infeasible.

The IQ switch works as follows. There is a *first-in-first-out* (FIFO) queue maintained at each input and an arriving cell is placed at the tail of this queue. In each slot, at most one cell is transferred to each output which is immediately transmitted on the output link. Thus the switch should be capable of switching one cell to each output and one cell from each input. The speed of the memory used to maintain the cell queues should allow one read and one write in each slot. This is obviously a more scalable alternative to the OQ switch. However, as is shown in the example in Figure 1.4, it suffers from *head-of-line* (HOL) blocking, which severely affects its achievable throughput. The question then is, how low is this reduction in the throughput? To answer this question, we proceed as follows.

Consider an $N \times N$ IQ switch operating under the extreme condition of input saturation: Every time a cell is switched from the input to its output, it is immediately replaced by another cell. Any of the N output ports is equally

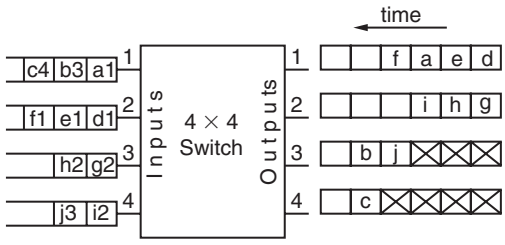

Figure 1.4 Example of a head-of-line (HOL) blocking situation in an IQ switch. At the input, the cells are shown with a label denoted by a letter and the destination port number. At the output, the cell departure sequence in five slots is shown. It is assumed that no more packets arrive at the inputs during these slots. Crosses indicate that there was no transmission from the output in that slot. Observe that although output 4 is free, packet c is blocked by its HOL packets, a and b, which are waiting for their outputs to become free.

likely to be the destination of this cell and is independent of everything else (e.g., destinations of other cells). Consider the case for $N = 2$. We will assume that in the case of destination conflict, one of the cells is chosen randomly to be switched and transmitted on the output link. Now consider the system at the beginning of every slot. Let $d_n^{(i)}$ be the destination of the cell in input i during slot n. $d_n := [d_n^{(1)}, d_n^{(2)}]$ can be called the state of the switch at the beginning of slot n. There are four possible states that the switch can be in: $[1,1]$, $[1,2]$, $[2,1]$, and $[2,2]$. Observe that to determine the probabilities for the state of the switch at the beginning of slot $n+1$, it is sufficient to know the state at the beginning of slot n. Thus the switch can be modeled as a four-state discrete time Markov chain, with transition probabilities as shown in Figure 1.5.

Solving the Markov chain of Figure 1.5, we get the stationary probability of the switch being in each of the four states to be 0.25. To obtain the throughput of the system we use the following observation. If the system is in state $[1,1]$ or $[2,2]$, only one of the cells can be switched to the output, and the throughput of the switch will be 0.5 cell per port. If the switch is in either $[1,2]$ or $[2,1]$, both cells can be switched to their output and the throughput is 1 cell per port. Thus the stationary saturation throughput of this switch will be $(0.25 \times 0.5 + 0.25 \times 0.5 + 0.25 \times 1.0 + 0.25 \times 1.0) = 0.75$ cells per port. Although this analysis can be extended to larger values of N, we immediately see a scalability issue in solving the model. The number of states in the Markov chain will be N^N, and even for small values of N it becomes impossible to enumerate the states, describe them, and solve

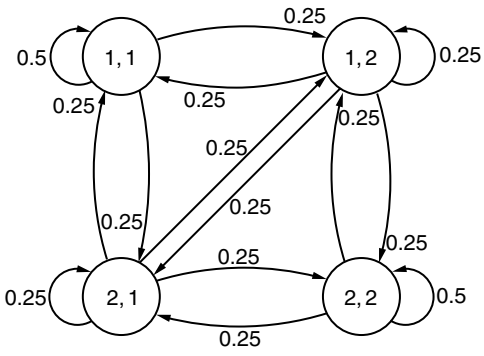

Figure 1.5 Markov chain representation of the state of the 2 × 2 input queued switch at the beginning of every slot. The numbers next to the arrows are the transition probabilities. The state of the Markov chain is $[d^{(1)}, d^{(2)}]$ where $d^{(i)}$ is the destination port of the packet at input *i*.

the Markov chain. The saturation throughputs for eight values of N are shown in Table 1.1 after obtaining them using this technique. As shown in the table, the throughput decreases as N increases. The question then is, does the saturation throughput converge to some value sufficiently greater than zero, or would it continue to decrease to zero with increasing N, thereby negating the technological advantages of the input-queued switch through a significant performance penalty for large N? The answer to this question is in the asymptotic analysis, discussed

N	Saturation throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184

Table 1.1 Saturation throughput of an $N \times N$ input-queued switch for $N = 1, 2, \ldots, 8$.

in Chapter 10, where we will show that the saturation throughput converges to $2 - \sqrt{2}$ (≈ 0.586) cells per port as N becomes large.

The saturation throughput of the switch is an indication of the *capacity* of the switch: the maximum packet arrival rate that can be applied to the inputs and yet keep the queue lengths bounded. In fact, it is widely believed, but not proved except for a special case, that the capacity is equal to the saturation throughput. From the foregoing discussion, the maximum per port packet arrival rate sustainable by an IQ switch is asymptotically 0.586, whereas it is 1.0 for the OQ switch.

Thus the capacity of the FIFO IQ switch—the input arrival rate that it can sustain at each input—is significantly greater than 0 even for large N but significantly worse than the ideal of 1.0 of the OQ switch. The natural interest then is in improving its capacity, possibly by doing some computing to smartly schedule the cells that will be offered to the switch fabric from each input, to make the capacity approach the ideal of 1.0. This will allow us to trade off switching speed for computing power. As has been shown in the example of Figure 1.4, the source of throughput degradation is the FIFO nature of the input queues and the consequent head-of-line blocking. To minimize this effect, many ad hoc schemes were tried, and it was not clear until 1996 whether IQ switches could ever achieve the ideal capacity. In that year, it was conclusively shown that an IQ switch can be made to yield a throughput of 1.0 per port by devising a scheme that takes an extreme step to break the HOL effect. In this scheme, called *virtual output queueing* (VOQ), each input, denoted by i, maintains a separate queue of cells for each output, denoted by j, called the virtual output queue at input i for output j and denoted by VOQ_{ij}. A weighted *bipartite* graph with a vertex for each input and each output is formed with edges defined only between an input and an output vertex. The weight of the edge (i, j), is Q_{ij}, the number of cells in VOQ_{ij}. A 2×2 VOQ switch and the weighted bipartite graph are shown in Figure 1.6. A *maximum-weight-matching* algorithm is executed on the weighted bipartite graph at the beginning of each slot, and the matching determines from which of the N virtual output queues at each input a cell will be switched to the respective outputs. We will study this and other algorithms in more detail in Chapter 10.

Note that an identical problem was solved in the context of multihop *packet radio networks* (PRNs). A multihop PRN is represented by a graph $G = (V, E)$, where V is the set of nodes in the network and E is a set of directed edges. A directed edge $e = (i, j) \in E$, where $i, j \in V$, indicates that node j is in the transmission range of node i—that is, node i can receive the transmissions from node j. Figure 1.7 shows an example. At any time, if link (i, j) is *active*, it means that node i is transmitting to node j. Radio networks differ from wired networks in the important aspect that not all links can be active at the same time. This is

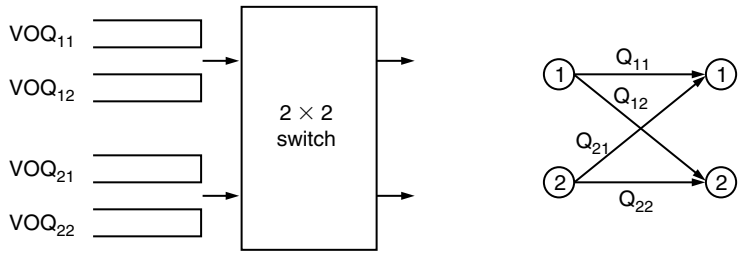

Figure 1.6 A 2 × 2 virtual output queue switch. The left panel shows the four queues—Q_{11}, Q_{12}, Q_{21}, and Q_{22}—at the inputs, and the right panel shows the corresponding bipartite graph.

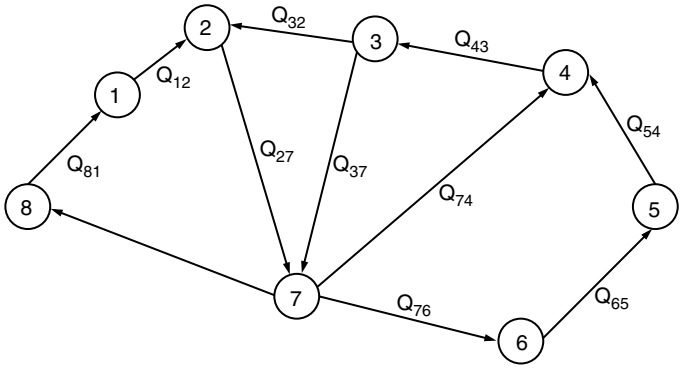

Figure 1.7 An example of a graph representing a packet radio network. The link weights are Q_{ij}, the number of packets in *i* with destination *j*.

for two reasons. First, in such networks, the radio transceiver can either send or receive at any point in time but cannot do both simultaneously. Second, a radio receiver can decode only one transmission at a time; multiple signals received at a node can destructively interfere so that neither is received correctly. On the other hand, if a radio receives one strong signal and another weak one, it may be able to detect the data in the strong signal. Thus, for example, in Figure 1.7, links $(1, 2)$ and $(3, 2)$ cannot be active at the same time because node 2 cannot

receive two simultaneous transmissions from such nearby nodes. On the other hand, we can specify that $(6, 5)$ and $(3, 2)$ can be active at the same time because the interfering signals are weak enough at this distance. The set of edges that can be simultaneously active is called a *feasible schedule*. Let S be the set of all feasible schedules.

Consider a time-slotted PRN with slot length equal to a packet transmission time. Let the packet arrivals form a random process, and, at the beginning of slot n, let $Q_n^{(ij)}$ be the number of packets that are at node i with destination to a neighbor j. Let us also assume that all transmissions need to go only one hop; when node i sends to node j, the packet exits the network at j. It is obvious that the sequence of schedules determines the capacity of the network: the maximum arrival rates that can be allowed between the source destination pairs without the queues becoming unbounded. Now consider the problem of scheduling the links to maximize the capacity of the packet radio network. The following algorithm is shown to provide the maximum capacity. At the beginning of every slot n, obtain a maximum-weight-matching on the weighted graph G with weight of edge (i, j) equal to $Q_n^{(ij)}$, and activate the links in the match. Observe that this problem, and the solution, is identical to the VOQ switch scheduling problem discussed earlier except that the graph in the PRN is not a bipartite graph. We discuss multihop PRNs in Chapter 8.

From this discussion, you can see that two important practical problems in networking, seemingly disparate, in the abstract have an almost identical solution technique. This underscores the importance of an analytical approach to networking, which is the focus of this book.

1.3 The Importance of Quantitative Modeling in the Engineering of Telecommunication Networks

In engineering, mathematical modeling is typically used for two purposes:

1. Simple "toy" models are developed and analyzed to obtain insights into phenomena. These models abstract away a lot of detail, and the results obtained from them are usually not directly useful in the design of actual systems. Such models are amenable to mathematical analysis, yielding provable mathematical relationships and sometimes even yielding closed-form expressions for the way performance varies with system parameters. These models can be used to derive control algorithms and scheduling policies and to motivate heuristics. The examples given earlier fall into this category and are only two among numerous instances in which mathematical

models and their analysis have provided important insights into phenomena in telecommunication networks.

2. More complicated models that capture more system detail are developed and analyzed to yield numerical results that can be used in actual system design. Often these models are too complex for obtaining closed-form expressions, and we must take recourse to numerical computation or even computer simulation. In such situations, simplified instances of the complete model, but amenable to mathematical analysis, are useful in validating the computer simulations. Algorithms and heuristics derived from the simpler models can be experimented with on these more detailed models.

Perhaps the earliest use of engineering models in telecommunication networks was in the *teletraffic engineering* of circuit-multiplexed telephone networks. In the early days of telephony, the deployment of long distance telephone circuits was expensive, and hence it was important to understand the call attempt rate that they could be offered for a given probability of call blocking. The most basic model used in the engineering of telephone circuits is the celebrated *Erlang-B* model. This model comprises several identical resources, along with a random arrival process of "customers," each of which requires exactly one (any one suffices) of the resources for a random duration of time, after which the customer departs. A customer that finds all resources occupied is sent away and is not heard from again. The mathematical assumptions are that the request arrival process is Poisson and that the resource holding times are independent and identically distributed; the model yields the probability of turning a customer away, or *call blocking*. Beginning from this model, there was hectic research activity in the area of trunk engineering with alternate-routed traffic (an available route is chosen from a set of routes using a predefined algorithm), time-varying traffic, and with different classes of calls requiring different performance objectives. The Erlang-B model, when extended to a circuit-multiplexed network with *fixed routes*, gave rise to the *network Erlang* model. A natural approximation was extensively studied for obtaining route-blocking probabilities from the network Erlang model. The area of *dynamic routing* was driven by the insights obtained from such engineering models. Although packet voice was explored as recently as the 1970s and 1980s, the idea of exploiting silences in speech to efficiently manage transoceanic telephone channels was studied via mathematical models as early as in the 1950s.

With the emergence of packet networking, attention was paid to the development of models of buffered-congestion systems. In the early days of this technology, packet networks were viewed as *computer networks*: networks

for interconnecting computers. Interestingly, the development of the theory of *queueing network* models, and of optimal scheduling in queueing systems, was given considerable impetus by the problems that arose in modeling interconnected computers that were executing distributed computing tasks. In the 1980s it was realized that there could be efficiencies if all kinds of telecommunications traffic (data, voice, video, etc.) could be integrated into packet networks. Because packet networks mix all traffic into the same links, it became necessary to develop local transmission scheduling at links and distributed traffic controls across the network in order for the different flows to obtain the desired performance. The need to understand the performance of *integrated packet networks* gave rise to the need to develop models of buffered-congestion systems with a variety of complex arrival processes (modeling, for example, superposed traffic from packet voice calls), complex service disciplines, and congestion-dependent traffic controls.

The use of analytical models for the end-to-end engineering of integrated packet networks is still in its infancy, but significant engineering insights have been obtained. In addition to the two examples given at the beginning of this chapter, the following are more such striking results:

- A "network calculus" for the end-to-end deterministic analysis of the performance of flows in networks, and for design with worst case performance guarantees.
- A stochastic calculus based on the notion of "effective bandwidths" for link sizing with stochastic performance guarantees.
- Distributed asynchronous algorithms for rate allocation to elastic flows sharing a packet network.
- The observation that packet flows in the Internet are not Poisson-like but instead have slowly decaying correlations (i.e., have long-range dependence), and an understanding of the origins of this phenomenon.
- The proof that the maximum-weight-matching algorithm achieves 100% throughput in an IQ packet switch, and the development of a large class of "maximal" scheduling algorithms that are significantly simpler to implement but perform as well as the maximum-weight-matching algorithm.
- The development of nonhierarchical routing algorithms for telephone networks. This led to the discovery of the problem of the network being in a "bad state" (high call-blocking probabilities) for extended periods

of time, and its solution by the method of trunk reservation and state protection.

- The insight that many optimal routing problems are equivalent to shortest path problems on the network topology when appropriate link weights are used.

One of the characteristic aspects of networking technologies (generally not seen in physical layer technologies) is that designs are typically not "derived" from engineering models but, in fact, precede any such modeling exercise. Simple, back-of-the-envelope calculations are taken to provide sufficient support for such designs. Simple, but not naive, technologies such as Aloha, Ethernet, and Transmission Control Protocol (TCP) are among the many important examples. Because such technologies adequately serve their initial purposes, they are quickly deployed. They work well to provide simple services, but to extend them beyond their original brief requires understanding their behavior. Analytical modeling is indispensable for this purpose because the generality of the conclusions is not possible with experimental investigations alone. In the context of TCP, for example, such analytical models have provided very important results on TCP performance problems and their improvement over wireless channels, and on TCP performance optimization and differentiation using Active Queue Management.

The approach of abstracting out the essentials of a problem and formulating it mathematically also permits the discovery of recurrent themes, as we saw in the packet switching and the packet radio network examples in Section 1.2. The problems of resource sharing in networks are usually very complex, and even to develop heuristics we often need a conceptual starting point. The analytical approach permits the development of optimal solutions from which better heuristics can be inferred.

As we have hinted at, the complete design of packet networks using stochastic models is a hard problem, and satisfactory solutions remain elusive. Into this void have stepped deterministic flow-based formulations and related constrained optimization problems. Practical problems—such as capacitated network topology design, setting of link weights for shortest path routing, and network overprovisioning to take care of failure situations—have been addressed using such approaches. These techniques at least provide a network operator with reasonable alternatives that can be starting points for manual optimizations.

This discussion has traced the role that mathematical models have played in the engineering of telecommunication networks. Quantitative modeling has been, and continues to be, indispensable for the proper understanding of and design of telecommunication networks. Such modeling can lead to major gains

even in the *qualitative* understanding of systems. For example, in the bit-dropping example of Section 1.1, we found that congestion-dependent bit-dropping could lead to a doubling of the load that could be carried for the same application-level performance. Ultimately, to the network operator the call-handling rate is what matters, because each accepted call provides revenue. Such insights are extremely valuable because they can help us to make judicious choices between various proposals and conjectures, and they can be obtained only via models and their analysis. In this book we cover most of the mathematical models and analysis techniques that have been mentioned.

1.4 Summary

This chapter aims to motivate our analytical approach to the topic of communication networking by presenting two examples that show how important insights are obtained by adopting the approach of mathematical modeling and analysis. One example is in the context of transporting voice over packet networks, and the other is on the problem of queueing and scheduling in high-speed switches. Using these examples as a springboard, we then recall the role of quantitative modeling in telecommunication networks, from the design of telephone networks to the analysis of myriad issues in integrated packet networks.

1.5 Notes on the Literature

The comparison of the performance of packet voice multiplexers with and without bit-dropping was performed by Sriram and Lucantoni [270], and the bit-dropping technique was implemented in a system described in [271]. We provide the results for a simple version of the authors' more detailed model. This analysis can be done using level crossing analysis, for which Brill and Posner [46] is the standard reference.

The saturation throughput of the input queued switch was derived by Karol, Hluchyj, and Morgan [158]. The maximum-weight-matching algorithm for the virtual output queued switch and the fact that this can lead to maximum capacity was shown by McKeown, Anantharam, and Walrand [209]. The packet radio network problem and its solution were provided by Tassiulas and Ephremides [283]; where they consider a more general system in which packets can traverse multiple wireless hops.

One of the first books to provide an analytical treatment of communication networks is the widely used textbook by Bertsekas and Gallager [35]. More modern treatments are provided by Schwartz [260] and by Walrand and Varaiya [297].

Problems

The following problems are in the spirit of this chapter, which is to illustrate the theme and tenor of the book. The solution of these problems is not critical to understanding this chapter or the rest of the book. However, at least thinking about them helps you to get into the mood.

1.1 In analyzing the buffer level in the multiplexer in the two schemes discussed in Section 1.1, we claimed that there will be a point mass at 0. Why will this be true?

1.2 If $\rho = 1.0$, show that $f(\cdot)$, as given in Equation 1.1, is independent of x.

1.3 With reference to Figure 1.3, would the difference in the number of calls carried by the two schemes approach a constant as the delay bound increases, or would it eventually become zero? Explain.

1.4 Consider an $N \times N$ cell switch that operates in a time-slotted manner. Assume that in a slot all the N inputs get a cell. Also assume that a cell has output j as its destination, with probability $\frac{1}{N}$ for $j = 1, \ldots, N$, and that the destination of a cell is independent of the other cells. What is the probability that there will be no destination conflict, that is, that each cell will choose a different destination?

1.5 Consider the same case as in 1.4 except that a cell arrives at an input with probability λ. Among the cells that arrived in the same slot, what is the probability that there is no destination conflict?