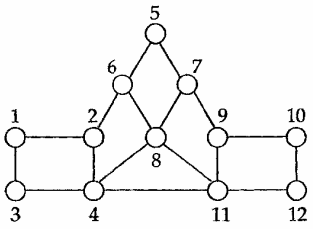
Chapter 11

Routing

11.1 Introduction

Routing is the process of finding a path from a source to every destination in the network. It allows users in the remotest part of the world to get to information and services provided by computers anywhere in the world. Routing is what makes networking magical: allowing telephone conversations between Botswana and Buenos Aires, and video clips from the space shuttle to be multicast to hundreds of receivers around the world! How does a network choose a path that spans the world? How does the routing system scale to describe paths to many millions of endpoints? How should the system adapt to a failed link? What if the user wants to choose a path that has the least delay, or the least cost, or the most available capacity? What if the users are themselves mobile, attaching to the network from different wired access points? These are the sorts of questions we will study in this chapter.

Routing is accomplished by means of *routing protocols* that establish mutually consistent *routing tables* in every router (or switch controller) in the network (Figure 11.1). A routing table contains at least two columns: the first is the address of a destination endpoint or a destination network, and the second is the address of the network element that is the next hop in the "best" path to this destination. When a packet arrives at a router (or when a call-setup packet arrives at a switch controller), the router or switch controller consults the routing table to decide the next hop for the packet.



ROUTING TABLE AT 1

Destination	Next hop	Destination	Next hop
1	—	7	2
2	2	8	2
3	3	9	2
4	3	10	2
5	2	11	3
6	2	12	3

Figure 11.1: Routing table. The figure shows the routing table at node 1. The table shows the next hop for each destination (which corresponds to a specific interface at host 1).

EXAMPLE 11.1

An example of a routing table for a toy network is shown in Figure 11.1. We see that the routing table for node 1 has one entry for every other node in the network. This allows it to choose the next hop for every possible destination. For example, packets that arrive at node 1 destined for node 10 are forwarded to node 2.

Notice that node 1 has only two choices: to forward a packet to 2 or to forward it to 3. This is a *local* routing choice. Yet this choice depends on the *global* topology, because the destination address by itself does not contain enough information to make a correct decision. For example, the shortest path from node 1 to node 6 is through 2, and the shortest path to 11 is through 3. Node 1, just by looking at the destination address "6," cannot decide that node 2 should be the next hop to that destination. *We conclude that any routing protocol must communicate global topological information to each routing element to allow it to make local routing decisions.* Yet global information, by its very nature, is hard to collect, subject to frequent change, and voluminous. How can we summarize this information to extract only the portions relevant to each node? This lies at the heart of routing protocols.

A routing protocol asynchronously updates routing tables at every router or switch controller. For ease of exposition, in the remainder of the chapter, we will refer to both routers and switch controllers as "routers." Recall that switch controllers are called upon

to route packets only at the time of call setup, so that they route connections, instead of packets (Section 8.1.1).

.2 Routing protocol requirements

A routing protocol must try to satisfy several mutually opposing requirements:

- *Minimizing routing table space:* We would like routing tables to be as small as possible, so that we can build cheaper routers with smaller memories that are more easily looked up. Moreover, routers must periodically exchange routing tables to ensure that they have a consistent view of the network's topology: the larger the routing table, the greater the overhead in exchanging routing tables. We usually require a routing table to grow more slowly than the number of destinations in the network.
- *Minimizing control messages:* Routing protocols require control message exchange. These represent an overhead on system operation and should be minimized.
- *Robustness:* The worst thing that a router can do is to misroute packets, so that they never reach their destination. (They are said to enter a *black hole.*) Routers in error may also cause *loops* and *oscillations* in the network.

EXAMPLE 11.2

If routing tables are inconsistent, loops can easily be formed. For example, router A may think that the shortest path to C is through B, and B may think that the shortest path to C is through A. Then, a packet to C loops back and forth between A and B until some other procedure (such as a *hop count* reaching zero, as described in Section 12.4.2) detects the loop and terminates the packet.

Oscillations can be caused if the routing protocol chooses paths based on the current load. Consider routers A and B connected by paths P1 and P2. Suppose P1 is heavily loaded and P2 is idle. The routing protocol may divert all traffic from P1 to P2, thus loading P2. This makes P1 more desirable, and traffic moves back to P1! If we are not careful, traffic oscillates from P1 to P2, and the network is always congested.

Black holes, loops, and oscillations are rare under normal conditions, but can show up if routing tables are corrupted, users specify incorrect information, links break or are restored, or routing control packets are corrupted. A robust routing protocol should protect itself from these types of problems by periodically running consistency tests, and by careful use of checksums and sequence numbers as described in Chapter 12.

- *Using optimal paths:* To the extent possible, a packet should follow the "best" path from a source to its destination. The "best" path may not necessarily be the shortest path: it may be a path that has the least delay, the most secure links, or the lowest monetary cost, or one that balances the load across the available paths. Routers along the entire path must collaborate to ensure that packets use the best possible route to maximize overall network performance.

As always, these requirements represent trade-offs in routing protocol design. example, a protocol may trade off robustness for a decrease in the number of cor messages and routing-table space. Many common protocols trade off a dramatic reduction in routing-table space for slightly longer paths.

11.3 Choices

Designers of routing protocols have many mechanisms available to them. In this section, we will describe some commonly available choices for routing [ME 90]. These choices also represent a rough taxonomy to categorize routing protocols.

- *Centralized versus distributed routing:* In centralized routing, a central processor collects information about the status of each link (up or down, utilization, and capacity) and processes this information to compute a routing table for every node. It then distributes these tables to all the routers. In distributed routing, routers cooperate using a distributed routing protocol to create mutually consistent routing tables. Centralized routing is reasonable when the network is centrally administered and the network is not too large, as in the core of the telephone network. However, it suffers from the same problems as a centralized name server: creating a single point of failure, and the concentration of routing traffic to a single point.

- *Source-based versus hop-by-hop:* A packet header can carry the entire route (that is, the addresses of every router on the path from the source to the destination), or the packet can carry just the destination address, and each router along the path can choose the next hop. These alternatives represent extremes in the degree to which a source can influence the path of a packet. A source route allows a sender to specify a packet's path precisely, but requires the source to be aware of the entire network topology. If a link or a router along the path goes down, a source-routed packet will not reach its destination. Moreover, if the path is long, the packet header can be fairly large. Thus, source routing trades off specificity in routing for packet-header size and extra overhead for control messages.

  An intermediate solution is to use a *loose source route.* With loose source routes, the sender chooses a subset of routers that the packet should pass

through, and the path may include routers not included in the source route. Loose source routes are supported in the IP version 4 and 6 headers.

- *Stochastic versus deterministic:* With a deterministic route, each router forwards packets toward a destination along exactly one path. In stochastic routing, each router maintains more than one next hop for each possible destination. It randomly picks one of these hops when forwarding a packet. The advantage of stochastic routing is that it spreads the load among many paths, so that the load oscillations characteristic of deterministic routing are eliminated. On the other hand, a destination may receive packets along the same connection out of order, and with varying delays. Consequently, modern networks usually use deterministic routing.

- *Single versus multiple path:* In single-path routing, a router maintains only one path to each destination. In multiple-path routing, a router maintains a *primary* path to a destination, along with *alternative* paths. If the primary path is unavailable for some reason, routers may send packets on the alternative path (with stochastic routing, routers may send packets on alternative paths even if the primary path is available). Single-path routing is used on the Internet, because maintaining alternative paths requires more routing table space. Telephone networks usually use multiple-path routing, because this reduces the call blocking probability, which is very important for customer satisfaction.

- *State-dependent versus state-independent:* With state-dependent or *dynamic* routing, the choice of a route depends on the current (measured) network state. For example, if some links are heavily loaded, routers may try to route packets around that link. With state-independent or *static* routing, the route ignores the network state. For example, a shortest-path route (where we measure the path length as the number of hops) is state independent. State-dependent routing usually finds better routes than state-independent routing, but can suffer from problems caused by network dynamics (such as the routing oscillations described earlier). It also requires more overhead for monitoring the network load. The Internet uses both state-dependent and state-independent routing. Telephone network routing used to be state independent, but state-dependent routing with multiple paths is now the norm.

Having broadly considered the choices in routing protocol design, the rest of the chapter deals with specific routing protocols that make a selection from the choices described earlier. The literature on routing (both in the telephone network and in the Internet) is vast. We can only touch upon the essential ideas in this book. We first study routing in the telephone network, then in the Internet. At the time of this writing, routing in ATM networks was still being discussed, but the outcome is likely to closely resemble routing on the Internet. We therefore do not discuss ATM routing in any detail.

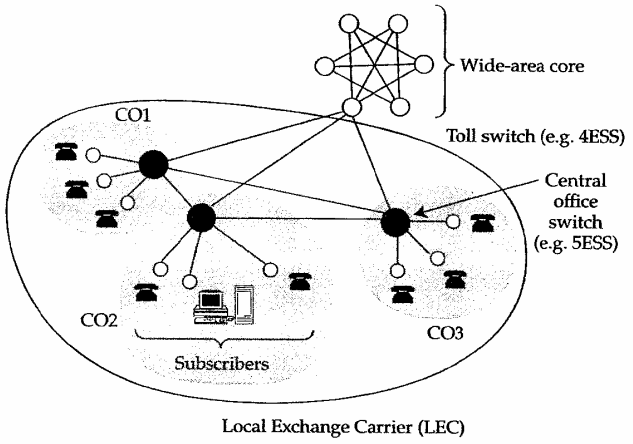


Figure 11.2: Simplified view of the telephone network. Subscribers are connected to a local exchange carrier (LEC). Central offices owned by LECs connect to a toll (wide-area) switch. Toll switches are connected to each other using single-hop paths.

11.4 Routing in the telephone network

In this section, we will focus on concepts and techniques for routing in the telephone network. A good reference that covers this material in greater depth is [Girard 90].

A national telephone network is structured as a fairly rigid three-level hierarchy, where the levels represent subscriber telephone instruments or modems, central offices, and long-distance or *toll* switching offices (Figure 11.2). We will refer to the set of toll switches as the network *core*. A *local exchange carrier*, such as Bell Atlantic in the eastern United States, provides local telephone service and manages many central offices interconnected with one-hop links. Every central office in each local exchange carrier's domain logically connects to one node (or, rarely, two nodes) in the core.[1] Each element in the core, such as the Lucent 4ESS switch we mentioned in Chapter 8, can switch up to 120,000 simultaneous calls and costs several million dollars.

The core is structured as a fully connected mesh or *clique*, with every toll switch connected to every other toll switch by a logical one-hop path. The number of interconnections is truly enormous. For example, the AT&T wide-area network core has around

[1]This is a simplification. Many local exchange carriers use *tandem* switches to provide an intermediate level in the hierarchy, and core switches connect to tandems instead of to central offices. Moreover, in some countries, such as the United States and Finland, several long-distance companies compete to provide network cores. Subscribers can choose one of several cores to complete their call, and local exchange carriers connect to multiple cores from their central offices.

135 switches interconnected by nearly 5 million *trunks*.[2] This dense connectivity simplifies routing. The routing algorithm is as follows:

(a) If a call's source and destination are within a central office, directly connect them.
(b) If the call is between central offices within a local exchange carrier, use a one-hop path between central offices.
(c) Otherwise send the call to (one of) the core(s).

The only major decision is at a toll switch, which chooses either a one-hop or a two-hop path to a destination switching system. It is not necessary to consider longer paths because the core is a logical clique. Thus, for an m-element core, each direct path corresponds to $m - 1$ alternative two-hop paths. The essence of telephone-network routing is in determining which one-hop path to choose in the core, and if this is fully used, the order in which to try two-hop paths.

Over the past hundred years, many different routing policies have been used in the telephone network. The computational limitations of electromechanical relays highly constrained the earliest policies. As computerized switching systems have replaced these, the routing policies have become increasingly sophisticated. However, all telephone routing policies have the same features, which we discuss next.

11.4.1 Features of telephone network routing and a comparison with Internet routing

All telephone routing policies draw upon the fact that aggregated telephone traffic is very predictable. Thus, it is possible to compute the approximate load between every pair of toll switches in advance, for every interval of every day. This allows routes to be chosen in advance. Second, telephone switches and links are extremely reliable. Switches go down no more than an average of a few minutes per year, and links are out of commission no more than a few hours every year. Therefore, unlike the Internet, where we can rarely count upon links to stay up, in the telephone network, the normal situation is that nearly all trunks and switches are up. This allows the routing protocol to be highly optimized: instead of just trying to maintain connectivity, network administrators can use sophisticated load-balancing strategies.

The third feature of a long-distance telephone network is that a single organization controls the entire network. Thus, traffic measurement and management policies can be universally implemented. Moreover, upgrades to routing policies can be carried out uniformly. In contrast, in the Internet, network administrators in different domains may choose differing policies, or worse, run out-of-date and inconsistent routing protocols.

Fourth, the network is very highly connected, with many equal-length alternative paths. In contrast, the Internet is rather sparse, so there are few choices for alternative

[2] A trunk corresponds to a pair of time slots carrying duplex 64-Kbps voice traffic. A single optical fiber can carry as many as 50,000 trunks. A collection of trunks between the same source destination pair form a *trunk group*.

paths in the core. (If this changes, some techniques used in the telephone network might become applicable to the Internet.)

Finally, routes in the telephone network are associated with a quality of service guarantee. Therefore, unlike the Internet, mere connectivity is not sufficient to complete a call: the path must also have sufficient resources available. Note, however, that all voice calls require the same, simple quality of service, so the admission control decision is trivial.

The cost of telephone network routing

Telephone network routing is simple because, historically, the electromechanical relays used for network control could not execute sophisticated programs. This simplicity, however, comes at a cost. For example, if a switch in the core crashes, the telephone network routing protocol cannot find an alternative path to an end-system reached through that switch. We conclude that to make the system reliable, *every* switch must be reliable. Similarly, the routing protocol requires every pair of switches to be connected by a one-hop logical trunk group. Although the physical connectivity is much sparser, creating and maintaining this logically fully connected clique is expensive. Would it not be cheaper to build a network where sophisticated routing algorithms lower the requirements for connectivity and component reliability? This is a hundred-billion dollar question!

If we could build a network that has all the features of the telephone network (low blocking probability, very high reliability, global coverage) using newer switching systems, the new network need not require a fully interconnected and reliable core, thus reducing costs. However, the switching, billing, signaling, and operations support systems that currently sit on top of this relatively simple core are so extensive, and so coupled to the existing architecture, that making a sudden change is too expensive.

Eventually, the telephone network core will run over ATM, allowing greater flexibility in routing and network topology. However, a complete cut over to ATM is likely to take at least a decade, if not longer. So, the short answer is, yes, current telephone network routing leads to increased cost, but, in the near term, the topology of the core is too ingrained to change.

In any case, note that a good rule of thumb is that about 80% of the cost of the entire telephone network is in the local loop, and about 90% of the local loop cost is in the labor of installation. The expense of nonoptimal routing in the core pales in comparison with these costs!

11.4.2 Dynamic nonhierarchical routing

The simplest possible routing algorithm in the network core is for it to accept a call if and only if the one-hop path is available. This algorithm is nonoptimal, in that a call

may be rejected though a two-hop path was available. A major advance in telephone routing was the introduction of *dynamic nonhierarchical routing* (*DNHR*). DNHR divides the day into approximately ten periods. In each period, each toll switch is assigned a primary (one-hop) path to another toll switch, and an ordered set of alternative (two-hop) paths. Incoming call-setup packets are first forwarded on the primary path. If sufficient resources are not available on the primary path, then the switch tries each of the allocated two-hop paths in turn (we call this *spilling* or *overflow*). The switch rejects the call if all the alternative paths are busy. The process in which a call rejected on a primary path is retried on an alternative path is called *crankback*. Crankback is necessary in any routing policy that supports quality-of-service constraints and wants to achieve a low call rejection rate.

DNHR draws upon the predictability of aggregated telephone traffic, and the fact that switches and links are usually available, to select optimal alternative two-hop paths. DNHR performance suffers when traffic changes unexpectedly, so that the list of alternative paths ought to change, but does not. This might increase load on trunk groups that are already heavily loaded while leaving other trunks underutilized. These problems are corrected with more sophisticated routing algorithms, which we discuss in Sections 11.4.3 and 11.4.4. We now make a small diversion to study the dynamics of routing with DNHR.

The Erlang map

An important idea associated with routing in general, and with telephone routing in particular, is the *Erlang map* [Girard 90, p. 159]. The *Erlang blocking formula*, described in Section 14.11.2, allows us to compute the blocking probability of a trunk group if we know the load on the group and its capacity. Given a fixed external call load on the network core, a routing strategy determines the load on each trunk group, and therefore the blocking probability for that trunk group. DNHR assigns a set of alternative paths to toll switches so that the expected blocking probability for an incoming call is minimized. Thus, the path of a new call depends on the expected load on each trunk group, which depends, in turn, on the routing! This circular dependency between routing and blocking probability leads to a system of equations called the *Erlang map*. We can show that the Erlang map has a *unique* fixed-point solution called the *Erlang fixed point*.

More precisely, we define $B(k)$ to be the blocking probability on trunk k, where $B(k) = E(L(k), C(k))$, $E(...)$ is the Erlang formula, $L(k)$ is the load on link k, and $C(k)$ is its capacity. Now, if a route r is denoted by a set of links, and $\nu(r)$ represents the external load on r, then we can approximate $L(k)$ by [Kelly 91]:

$$L(k) = \sum_{r:k\in r} \nu(r) \prod_{j\in r-\{k\}} (1 - B(j)) \tag{11.1}$$

To see this, note that $\nu(r)$ is the intrinsic load on route r. Each factor of $(1 - B(j))$ represents a "thinning" of the load, so that the load on trunk k is just the thinned sum of the loads on all the routes that share trunk k. $B(k)$ clearly depends on some number of other $B(j)$s, through $B(k) = E(L(k), C(k))$ and Equation 11.1. Thus, each of the $B(k)$s is implicitly defined by the others and forms the Erlang map.

Metastability in DNHR

Though the Erlang map has a unique solution, which represents a long-term mean blocking rate on each link, this solution is often the time-average of two distinct values, one high and the other low [Kelly 91]. In other words, given a blocking probability for a particular link, b, as the solution to Equation 11.1, we achieve b as the mean of two other values b_{high} and b_{low}. The underlying physical mechanism is that given a particular traffic load, the network periodically changes from a state where most blocking probabilities are low (b_{low}) to a state where most blocking probabilities are high (b_{high}). The *mean* blocking probability, b, is their average, and, in practice, the network may never achieve it. Let us see why.

Consider a network where a sudden burst of activity between toll switches A and B forces traffic to be off-loaded (spilled) onto paths ACB and ADB, adding extra load to links AC, CB, AD, and DB (Figure 11.3). If more traffic now appears on any one of these links, this traffic may, in turn, be diverted to other two-hop paths, making the situation worse. The key point is that every time a toll switch spills traffic to a two-hop alternative path, it increases the blocking probability for *two* other one-hop paths that use these links. We can show that the network, under a heavy load, can reach a *metastable* state where almost every call takes a two-hop path. Moreover, even with no change in mean offered load, the network can suddenly return to a "normal" state, where most paths are single hop. This is the physical mechanism underlying the multiple blocking probabili-

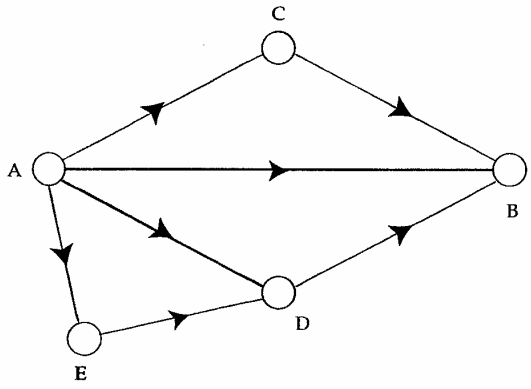


Figure 11.3: Metastability in telephone networks. Suppose the one-hop path from A to B is full (indicated by the heavy line), so that calls spill to two-hop paths A–C–B and A–D–B. This increases the load on trunks AC, AD, CB, and CB. The additional load may cause trunk AD to fill, causing calls from A to D to further spill to trunks AE and ED. If this continues, all calls take only two-hop paths through the core, although by rearranging calls, most or all of them can be diverted to one-hop paths. This is called metastability.

ties we mentioned in the previous paragraph. Metastability is undesirable because it can lead to a high call blocking rate even with moderate loads.

The existence of metastability in the telephone network is a cause of much concern. However, we can prevent it by simply reserving some part of the capacity on each trunk for one-hop calls [Akinpelu 84]. Continuing with our example, when the trunk AB is busy, AC and CB are allowed to carry no more than a given fraction of two-hop calls. Thus, new calls requiring, say, the CB trunk, would always have a good chance of being carried on a one-hop path. It can be shown that this technique (called *trunk reservation*) prevents the network from entering a metastable state [Girard 90, pp. 182–186]. The cost of trunk reservation is that the network may block a call even when it has sufficient capacity, thus increasing call blocking probability. However, because the network never reaches the metastable state, the *overall* blocking probability drops.

Shadow prices

How should the network allocate the set of alternative paths to a toll switch? The key theoretical result here is the computation of a *shadow price* to carry a call on a trunk [Kelly 88]. The idea is that if a call is carried on a trunk, it increases the blocking probability for future calls on that trunk. We quantify this increase in blocking probability as a shadow price. When a call comes in, we can compute the sum of the shadow prices for every alternate path. If the sum is larger than the revenue gained from a single call on every path, then the network should drop the call. Otherwise, the network should route it on the trunk that has the least cost. Given past traffic history, a network administrator can compute the expected least cost trunks for each period. These, then, form the set of alternative paths allocated to each toll switch by DNHR. If we can measure trunk loads dynamically, the shadow price approach allows a toll switch to compute optimal alternative paths for each call, instead of once every time period, as in DNHR.

11.4.3 TSMR

In DNHR, a central computer gives each toll switch a set of alternative paths based on past measurements, which are updated once a week. If a sudden surge of calls arrives on a trunk, the only adaptability in the network is to start trying the previously prescribed alternate paths. Although this was suitable for earlier toll switches that had very limited computational power, modern switch controllers can do much better. One step in this direction is *trunk status map routing*, or TSMR. In this scheme, each switch controller measures the load on each of its outgoing links and tells this to a central computer. The central computer periodically computes optimal alternative paths for each toll switch (based, for example, on the current load and the corresponding shadow prices) and loads these into all the toll switches. Thus, the central computer updates the choice of alternative paths more often than with DNHR. To dampen routing changes, a toll switch updates its load measurement only if this load changes "significantly," meaning, typically, 12.5%.

11.4.4 RTNR

The latest in the series of telephone routing algorithms is *real-time network routing* (telephone routing algorithms seem to require four-letter acronyms, and this is called RTNR).

RTNR, which typifies the current generation of telephone routing algorithms, replaced DNHR in AT&T's long-distance network in 1991. Unlike DNHR and TSMR, RTNR does not use centralized control. Instead, each toll switch monitors the loading of every outgoing trunk and computes a list of lightly loaded trunks. If the primary path for a call is busy, the originating toll switch asks the destination for the destination's list. Since calls are symmetric, the logical AND of the two lists is the set of lightly loaded alternative paths from the originating switch to the destination. For example, toll switch A may have light loads on links AB, AC, and AE. If the destination is D, D may report light loads on DC, DE, and DG. If DC is lightly loaded, from symmetry, so is CD. So, A knows that AC and CD are both lightly loaded, and discovers that A–C–D is a good alternative path.

RTNR allows a trunk to be partitioned among multiple traffic classes. Each class has its own blocking probability goal and trunk reservation level. When the network is lightly loaded, we give a class as much bandwidth as it can use, but under heavy loads, admission control ensures that the network meets the class's reservation level. Note that the traffic class here is used in a rather restricted sense to mean a voice call, a data call, or a multiplexed (N* 64 kbps) call.

RTNR is very effective in practice (see Exercise 11.1). For example, AT&T's long-distance network carries up to 260 million call attempts on busy days. Of these attempts, only about one or two are blocked in the core! Other telephone companies, with similarly engineered network cores, have similar call-blocking statistics.

11.5 Distance-vector routing

Telephone network routing is specialized to take advantage of the unique features of the telephone network, such as a predictable traffic flow, and a relatively small network core. Large packet networks, such as the Internet, present a very different environment. In the Internet, links and routers are unreliable, alternative paths are scarce, and traffic patterns can change unpredictably within minutes. It is not surprising that routing in the Internet, and in ATM networks, which are likely to have Internet-like characteristics, follows a different path. The two fundamental routing algorithms in packet-switched networks are *distance-vector* and *link-state*.

Both algorithms assume that a router knows (a) the address of each neighbor, and (b) the cost of reaching each neighbor (where the cost measures quantities like the link's capacity, the current queuing delay, or a per-packet charge). Both algorithms allow a router to find *global* routing information, that is, the next hop to reach every destination in the network by the shortest path, by exchanging routing information with only its neighbors. Roughly speaking, in a distance-vector algorithm, a node tells its *neighbors* its distance to *every* other node in the network, and in a link-state algorithm, a node tells *every* other node in the network its distance to its *neighbors*. Thus, both routing protocols are *distributed* and are suitable for large internetworks controlled by multiple administrative entities. In this section, we will focus on distance-vector algorithms. We will study link-state algorithms in Section 11.6.