widespread deployment of IPv6. Europe's Third Generation Partnership Program [3GPP 2012] has specified IPv6 as the standard addressing scheme for mobile multimedia.

One important lesson that we can learn from the IPv6 experience is that it is enormously difficult to change network-layer protocols. Since the early 1990s, numerous new network-layer protocols have been trumpeted as the next major revolution for the Internet, but most of these protocols have had limited penetration to date. These protocols include IPv6, multicast protocols (Section 4.7), and resource reservation protocols (Chapter 7). Indeed, introducing new protocols into the network layer is like replacing the foundation of a house—it is difficult to do without tearing the whole house down or at least temporarily relocating the house's residents. On the other hand, the Internet has witnessed rapid deployment of new protocols at the application layer. The classic examples, of course, are the Web, instant messaging, and P2P file sharing. Other examples include audio and video streaming and distributed games. Introducing new application-layer protocols is like adding a new layer of paint to a house—it is relatively easy to do, and if you choose an attractive color, others in the neighborhood will copy you. In summary, in the future we can expect to see changes in the Internet's network layer, but these changes will likely occur on a time scale that is much slower than the changes that will occur at the application layer.

4.4.5 A Brief Foray into IP Security

Section 4.4.3 covered IPv4 in some detail, including the services it provides and how those services are implemented. While reading through that section, you may have noticed that there was no mention of any security services. Indeed, IPv4 was designed in an era (the 1970s) when the Internet was primarily used among mutually-trusted networking researchers. Creating a computer network that integrated a multitude of link-layer technologies was already challenging enough, without having to worry about security.

But with security being a major concern today, Internet researchers have moved on to design new network-layer protocols that provide a variety of security services. One of these protocols is IPsec, one of the more popular secure network-layer protocols and also widely deployed in Virtual Private Networks (VPNs). Although IPsec and its cryptographic underpinnings are covered in some detail in Chapter 8, we provide a brief, high-level introduction into IPsec services in this section.

IPsec has been designed to be backward compatible with IPv4 and IPv6. In particular, in order to reap the benefits of IPsec, we don't need to replace the protocol stacks in *all* the routers and hosts in the Internet. For example, using the transport mode (one of two IPsec "modes"), if two hosts want to securely communicate, IPsec needs to be available only in those two hosts. All other routers and hosts can continue to run vanilla IPv4.

For concreteness, we'll focus on IPsec's transport mode here. In this mode, two hosts first establish an IPsec session between themselves. (Thus IPsec is connection-oriented!) With the session in place, all TCP and UDP segments sent between the

two hosts enjoy the security services provided by IPsec. On the sending side, the transport layer passes a segment to IPsec. IPsec then encrypts the segment, appends additional security fields to the segment, and encapsulates the resulting payload in an ordinary IP datagram. (It's actually a little more complicated than this, as we'll see in Chapter 8.) The sending host then sends the datagram into the Internet, which transports it to the destination host. There, IPsec decrypts the segment and passes the unencrypted segment to the transport layer.

The services provided by an IPsec session include:

- *Cryptographic agreement*. Mechanisms that allow the two communicating hosts to agree on cryptographic algorithms and keys.
- *Encryption of IP datagram payloads*. When the sending host receives a segment from the transport layer, IPsec encrypts the payload. The payload can only be decrypted by IPsec in the receiving host.
- *Data integrity*. IPsec allows the receiving host to verify that the datagram's header fields and encrypted payload were not modified while the datagram was en route from source to destination.
- *Origin authentication*. When a host receives an IPsec datagram from a trusted source (with a trusted key—see Chapter 8), the host is assured that the source IP address in the datagram is the actual source of the datagram.

When two hosts have an IPsec session established between them, all TCP and UDP segments sent between them will be encrypted and authenticated. IPsec therefore provides blanket coverage, securing all communication between the two hosts for all network applications.

A company can use IPsec to communicate securely in the nonsecure public Internet. For illustrative purposes, we'll just look at a simple example here. Consider a company that has a large number of traveling salespeople, each possessing a company laptop computer. Suppose the salespeople need to frequently consult sensitive company information (for example, pricing and product information) that is stored on a server in the company's headquarters. Further suppose that the salespeople also need to send sensitive documents to each other. How can this be done with IPsec? As you might guess, we install IPsec in the server and in all of the salespeople's laptops. With IPsec installed in these hosts, whenever a salesperson needs to communicate with the server or with another salesperson, the communication session will be secure.

4.5 Routing Algorithms

So far in this chapter, we've mostly explored the network layer's forwarding function. We learned that when a packet arrives to a router, the router indexes a forwarding table and determines the link interface to which the packet is to be directed. We also learned that routing algorithms, operating in network routers, exchange and

compute the information that is used to configure these forwarding tables. The interplay between routing algorithms and forwarding tables was shown in Figure 4.2. Having explored forwarding in some depth we now turn our attention to the other major topic of this chapter, namely, the network layer's critical routing function. Whether the network layer provides a datagram service (in which case different packets between a given source-destination pair may take different routes) or a VC service (in which case all packets between a given source and destination will take the same path), the network layer must nonetheless determine the path that packets take from senders to receivers. We'll see that the job of routing is to determine good paths (equivalently, routes), from senders to receivers, through the network of routers.

Typically a host is attached directly to one router, the **default router** for the host (also called the **first-hop router** for the host). Whenever a host sends a packet, the packet is transferred to its default router. We refer to the default router of the source host as the **source router** and the default router of the destination host as the **destination router**. The problem of routing a packet from source host to destination host clearly boils down to the problem of routing the packet from source router to destination router, which is the focus of this section.

The purpose of a routing algorithm is then simple: given a set of routers, with links connecting the routers, a routing algorithm finds a "good" path from source router to destination router. Typically, a good path is one that has the least cost. We'll see, however, that in practice, real-world concerns such as policy issues (for example, a rule such as "router *x,* belonging to organization *Y,* should not forward any packets originating from the network owned by organization *Z*") also come into play to complicate the conceptually simple and elegant algorithms whose theory underlies the practice of routing in today's networks.

A graph is used to formulate routing problems. Recall that a **graph** $G = (N,E)$ is a set N of nodes and a collection E of edges, where each edge is a pair of nodes from N. In the context of network-layer routing, the nodes in the graph represent routers—the points at which packet-forwarding decisions are made—and the edges connecting these nodes represent the physical links between these routers. Such a graph abstraction of a computer network is shown in Figure 4.27. To view some graphs representing real network maps, see [Dodge 2012, Cheswick 2000]; for a discussion of how well different graph-based models model the Internet, see [Zegura 1997, Faloutsos 1999, Li 2004].

As shown in Figure 4.27, an edge also has a value representing its cost. Typically, an edge's cost may reflect the physical length of the corresponding link (for example, a transoceanic link might have a higher cost than a short-haul terrestrial link), the link speed, or the monetary cost associated with a link. For our purposes, we'll simply take the edge costs as a given and won't worry about how they are determined. For any edge (x,y) in E, we denote $c(x,y)$ as the cost of the edge between nodes x and y. If the pair (x,y) does not belong to E, we set $c(x,y) = \infty$. Also, throughout we consider only undirected graphs (i.e., graphs whose edges do not have a direction), so that edge (x,y) is the same as edge (y,x) and that $c(x,y) = c(y,x)$. Also, a node y is said to be a **neighbor** of node x if (x,y) belongs to E.

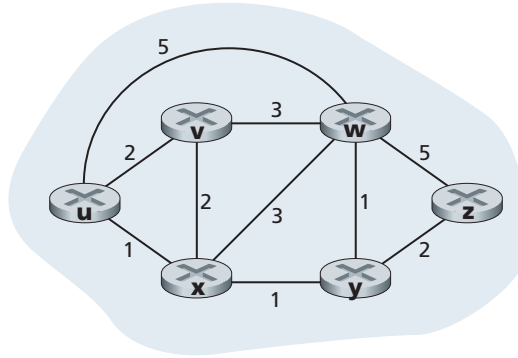

Figure 4.27 ♦ Abstract graph model of a computer network

Given that costs are assigned to the various edges in the graph abstraction, a natural goal of a routing algorithm is to identify the least costly paths between sources and destinations. To make this problem more precise, recall that a **path** in a graph $G = (N,E)$ is a sequence of nodes $(x_1, x_2,..., x_p)$ such that each of the pairs (x_1,x_2), $(x_2,x_3),...,(x_{p-1},x_p)$ are edges in E. The cost of a path $(x_1,x_2,..., x_p)$ is simply the sum of all the edge costs along the path, that is, $c(x_1,x_2) + c(x_2,x_3) + ...+ c(x_{p-1},x_p)$. Given any two nodes x and y, there are typically many paths between the two nodes, with each path having a cost. One or more of these paths is a **least-cost path**. The least-cost problem is therefore clear: Find a path between the source and destination that has least cost. In Figure 4.27, for example, the least-cost path between source node u and destination node w is (u, x, y, w) with a path cost of 3. Note that if all edges in the graph have the same cost, the least-cost path is also the **shortest path** (that is, the path with the smallest number of links between the source and the destination).

As a simple exercise, try finding the least-cost path from node u to z in Figure 4.27 and reflect for a moment on how you calculated that path. If you are like most people, you found the path from u to z by examining Figure 4.27, tracing a few routes from u to z, and somehow convincing yourself that the path you had chosen had the least cost among all possible paths. (Did you check all of the 17 possible paths between u and z? Probably not!) Such a calculation is an example of a centralized routing algorithm—the routing algorithm was run in one location, your brain, with complete information about the network. Broadly, one way in which we can classify routing algorithms is according to whether they are global or decentralized.

- A **global routing algorithm** computes the least-cost path between a source and destination using complete, global knowledge about the network. That is, the algorithm takes the connectivity between all nodes and all link costs as inputs. This then requires that the algorithm somehow obtain this information before actually performing the calculation. The calculation itself can be run at one site