1

HOW WI-FI WORKS

Up to a point, it's quite possible to treat your wireless network as a set of black boxes that you can turn on and use without knowing much about the way they work. That's the way most people relate to all of the technology that surrounds them—you shouldn't have to worry about the 802.11b specification to connect your laptop computer to a network. In an ideal world (ha!), it would work just as soon as you turn on the power switch.

But wireless Ethernet today is about where broadcast radio was in 1923. The technology was out there, but people spent a lot of time tweaking their equipment. And the people who understood what was happening behind that Bakelite-Dilecto panel were able to get better performance from their radios than the ones who expected to just turn on the power switch and listen.

In order to make the most effective use of wireless networking technology, it's still important to understand what's going on inside the box (or in this case, inside each of the boxes that makes up the network). This chapter describes the standards and specifications that control wireless networks, and it explains how data moves through the network from one computer to another.

Figure 1.1: Every new technology goes through the tweak-and-fiddle stage

When the network is working properly, you should be able to use it without thinking about all of this internal plumbing—just click a few icons on your computer's screen, and you're connected. But when you're designing and building a new network, or when you want to tweak the performance of an existing network, it can be essential to know how all that data is supposed to move from one place to another. And when the network does something you aren't expecting it to do, you will need a basic knowledge of the technology to do any kind of useful troubleshooting.

Moving data through a wireless network involves three separate elements: the radio signals, the data format, and the network structure. Each of these elements is independent of the other two, so it's necessary to define all three when you invent a new network. In terms of the familiar OSI (Open Systems Interconnection) reference model, the radio signal operates at the Physical layer, and the data format controls several of the higher layers. The network structure includes the interface adapters and base stations that send and receive the radio signals.

In a wireless network, the network adapters in each computer convert digital data to radio signals, which they transmit to other devices on the network, and they convert incoming radio signals from other network elements back to digital data. The IEEE (Institute of Electrical and Electronics Engineers) has produced a set of standards and specifications for wireless networks under the title "IEEE 802.11" that defines the format and structure of those signals.

The original 802.11 standard (without the "b" at the end) was released in 1997. It covers several different types of wireless media: two kinds of radio transmissions (which we'll explain later in this chapter) and networks that use infrared light. The more recent 802.11b standard provides additional specifications for wireless Ethernet networks. A related document, IEEE 802.11a, describes wireless networks that operate at higher speeds on different radio frequencies. Still other 802.11 radio networking standards with other letters are also moving toward public release.

The specification in widest use today is 802.11b. That's the *de facto* standard used by just about every wireless Ethernet LAN that you are likely to encounter in offices and public spaces and in most home networks. It's worth the trouble to keep an eye on the progress of those other standards, but for the moment, 802.11b is the one to use, especially if you're expecting to connect to networks where you don't control all the hardware yourself.

NOTE *The wireless networks described in this book follow the 802.11b standard, but much of the same information also applies to other kinds of 802.11 networks.*

You ought to know about two more names in the alphabet soup of wireless LAN standards: WECA and Wi-Fi. WECA (Wireless Ethernet Compatibility Alliance) is an industry group that includes all of the major manufacturers of 802.11b equipment. Their twin missions are to test and certify that wireless network devices from all of their member companies can operate together in the same network and to promote 802.11 networks as the worldwide standard for wireless LANs. WECA's marketing geniuses adopted the more "friendly" name of Wi-Fi (short for Wireless Fidelity) for the 802.11 specifications and changed their own name to the Wi-Fi Alliance.

Once or twice a year, the Alliance conducts an "interoperability bake-off" where engineers from many hardware manufacturers confirm that their hardware will communicate correctly with equipment from other suppliers. Network equipment that carries the Wi-Fi logo has been certified to meet the relevant standards, and to pass those interoperability tests. Figure 1.2 shows the Wi-Fi logos on network adapters from two different manufacturers.

Figure 1.2: Network adapters with the Wi-Fi logo

Radio Signals

802.11b networks operate in a special band of radio frequencies around 2.4 GHz that have been reserved in most parts of the world for unlicensed point-to-point spread-spectrum radio services.

The *unlicensed* part means that anybody using equipment that complies with the technical requirements can send and receive radio signals on these frequencies, without the need for a radio station license. Unlike most radio services, which require licenses that grant exclusive use of a frequency to a single user or group of users, and which restrict the use of that frequency to a specific type of service, an unlicensed service is a free-for-all, where everybody has an equal claim on the same piece of the spectrum. In theory, the technology of spread-spectrum radio makes it possible to coexist with other users (up to a point) without significant interference.

A *point-to-point* radio service operates a communication channel that carries information from a transmitter to a single receiver. The opposite of point-to-point is a *broadcast* service (such as a radio or television station) that sends the same signal to many receivers at the same time.

Spread spectrum is a family of methods for transmitting a single radio signal using a relatively wide segment of the radio spectrum. Wireless Ethernet networks use two different spread-spectrum radio transmission systems, called FHSS (frequency-hopping spread spectrum) and DSSS (direct-sequence spread spectrum). Some older 802.11 networks use the slower FHSS system, but the current generation of 802.11b and 802.11a wireless Ethernet networks use DSSS.

Spread-spectrum radio offers some important advantages over other types of radio signals that use a single narrow channel. Spread spectrum is extremely efficient, so the radio transmitters can operate with very low power. Because they operate on a relatively wide band of frequencies, they are less sensitive to interference from other radio signals and electrical noise, which means that the signals are often able to get through in environments where a conventional narrow-band signal would be impossible to receive and understand, and because a frequency-hopping spread-spectrum signal shifts among multiple channels, it can be extremely difficult for an unauthorized listener to intercept and decode the contents of a signal.

Spread-spectrum technology has an interesting history. It was invented by the actress Hedy Lamarr and the American avant-garde composer George Antheil as a "Secret Communication System" for directing radio-controlled torpedoes that would not be subject to enemy jamming. Before she came to Hollywood, Lamarr had been married to an arms merchant in Austria, where she learned about the problems of torpedo guidance at dinner parties with her husband's customers. Years later, during World War II, she came up with the concept of changing radio frequencies to cut through interference.

Antheil turned out to be the ideal person to make this idea work. His most famous composition was something called *Ballet Mechanique,* which was scored for 16 player pianos, two airplane propellers, four xylophones, four bass drums, and a siren. He used the same kind of mechanism that he had previously used to synchronize the player pianos to change radio frequencies in a spread-spectrum

transmission. The original slotted paper-tape system had 88 different radio channels — one for each of the 88 keys on a piano.

In theory, the same method could be used for voice and data communication as well as guiding torpedoes, but in the days of vacuum tubes, paper tape, and mechanical synchronization, the whole process was too complicated to actually build and use. By 1962, solid-state electronics had replaced the vacuum tubes and piano rolls, and the technology was used aboard U.S. Navy ships for secure communications during the Cuban Missile Crisis. Today, spread-spectrum radios are used in the U.S. Air Force Space Command's Milstar satellite communications system, in digital cellular telephones, and in wireless data networks.

Frequency-Hopping Spread Spectrum (FHSS)

Lamarr and Antheil's original design for spread-spectrum radio used a frequency-hopping system. As the name suggests, FHSS technology divides a radio signal into small segments and "hops" from one frequency to another many times per second as it transmits those segments. The transmitter and the receiver establish a synchronized hopping pattern that sets the sequence order in which they will use different subchannels.

FHSS systems overcome interference from other users by using a narrow carrier signal that changes frequency many times every second. Additional transmitter and receiver pairs can use different hopping patterns on the same set of subchannels at the same time. At any given point in time, each transmission is probably using a different subchannel, so there's no interference between signals. When a conflict does occur, the system resends the same packet until the receiver gets a clean copy and sends a confirmation back to the transmitting station.

For wireless data services, the unlicensed 2.4 GHz band is split into 75 subchannels, each of them 1 MHz wide. Because each frequency hop adds overhead to the data stream, FHSS transmissions are relatively slow.

Direct-Sequence Spread Spectrum (DSSS)

DSSS technology uses a method called an 11-chip Barker sequence to spread the radio signal through a single 22 MHz–wide channel without changing frequencies. Each DSSS link uses just one channel, without any hopping between frequencies. As Figure 1.3 shows, a DSSS transmission uses more bandwidth but less power than a conventional signal. The digital signal on the left is a conventional transmission, in which the power is concentrated within a tight bandwidth. The DSSS signal on the right uses the same amount of power, but it spreads that power across a wider band of radio frequencies. Obviously, the 22 MHz DSSS channel is a lot wider than the 1 MHz channels used in FHSS systems.

A DSSS transmitter breaks each bit in the original data stream into a series of redundant bit patterns called *chips* and transmits them to a receiver that reassembles the chips back into a data stream that is identical to the original. Because most interference is likely to occupy a narrower bandwidth than a DSSS signal, and because each bit is divided into several chips, the receiver can usually identify noise and reject it before it decodes the signal.

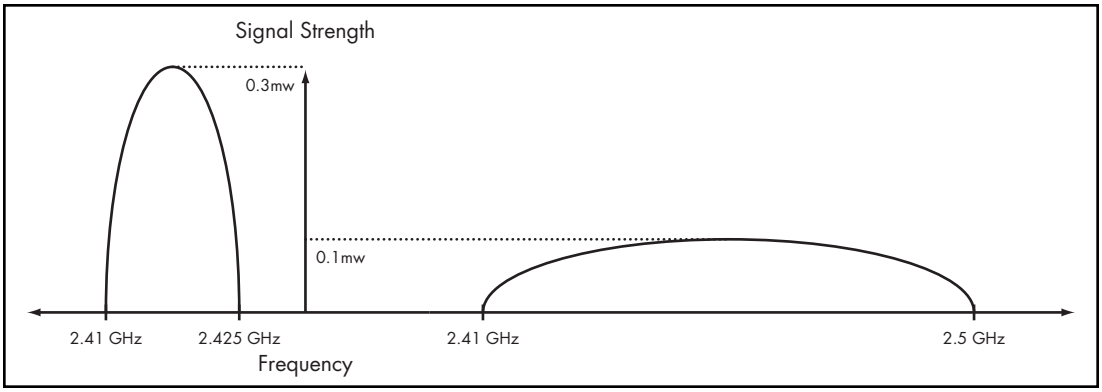

Figure 1.3: Conventional and DSSS radio signals

Like other networking protocols, a DSSS wireless link exchanges *handshaking* messages within each data packet to confirm that the receiver can understand each packet. The standard data transmission rate in an 802.11b DSSS network is 11 Mbps, but when the signal quality won't support that speed, the transmitter and receiver use a process called *dynamic rate shifting* to drop the speed down to 5.5 Mbps. The speed might drop because there's a source of electrical noise near the receiver or because the transmitter and receiver are too far apart to support full-speed operation. If 5.5 Mbps is still too fast for the link to handle, it drops again, down to 2 Mbps, or even 1 Mbps.

Frequency Allocations

By international agreement, a window of the radio spectrum near 2.4 GHz is supposed to be reserved for unlicensed industrial, scientific, and medical services, including spread-spectrum wireless data networks. However, the exact frequency allocations are slightly different from one part of the world to another; the authorities in different countries have assigned slightly different frequency bands. Table 1.1 lists the frequency assignments in several locations.

Table 1.1: Unlicensed 2.4 GHz Spread-Spectrum Frequency Assignments

Region	Frequency Band
North America	2.4000 to 2.4835 GHz
Europe	2.4000 to 2.4835 GHz
France	2.4465 to 2.4835 GHz
Spain	2.445 to 2.475 GHz
Japan	2.471 to 2.497 GHz

Just about every other country in the world also uses one of these bands. Those minor differences in frequency allocations are not particularly important (unless you plan to transmit across the border between France and Spain, or something equally unlikely), because most networks operate entirely within a sin-

gle country or region, and the normal signal range is usually just a few hundred feet. There's also enough overlap among the various national standards to allow the same equipment to operate legally anywhere in the world. You might have to set your network adapter to a different channel number when you take it abroad, but there's almost always a way to connect, assuming there's a network within range of your adapter.

In North America, Wi-Fi devices use 11 channels. Many other countries have authorized 13 channels, but Japan uses 14 channels, and only 4 are available in France. Fortunately, the entire world uses the same set of channel numbers, so Channel 9 in New York uses exactly the same frequency as Channel 9 in Tokyo or Paris. Table 1.2 lists the channels used in different countries and regions. Canada and some other countries use the same channel assignments as the United States.

Table 1.2: Wireless Ethernet Channel Assignments

Channel	Frequency (MHz) and Location
1	2412 (U.S., Europe, and Japan)
2	2417 (U.S., Europe, and Japan)
3	2422 (U.S., Europe, and Japan)
4	2427 (U.S., Europe, and Japan)
5	2432 (U.S., Europe, and Japan)
6	2437 (U.S., Europe, and Japan)
7	2442 (U.S., Europe, and Japan)
8	2447 (U.S., Europe, and Japan)
9	2452 (U.S., Europe, and Japan)
10	2457 (U.S., Europe, France, and Japan)
11	2462 (U.S., Europe, France, and Japan)
12	2467 (Europe, France, and Japan)
13	2472 (Europe, France, and Japan)
14	2484 (Japan only)

If you're not sure which channels to use in some other country, consult the local regulatory authority for specific information. Or use Channels 10 and 11, which are legal everywhere.

Note that the frequency specified for each of those channels is actually the center frequency of a 22 MHz channel. Therefore, each channel overlaps several other channels that are above and below it. The whole 2.4 GHz band has space for only three completely separate channels, so if your network runs on, say, Channel 4, and your neighbor is using Channel 5 or Channel 6, each network will detect the other network's signals as interference. Both networks will work, but the performance (as reflected in the data transfer speed) will not be as good as it would be when the channels are more widely separated from one another.

To minimize this kind of interference, you should try to coordinate channel use with nearby network managers. If possible, each network should use channels that are at least 25 MHz, or six channel numbers, apart. If you're trying to eliminate interference between two networks, use one high channel number and one low number. For three channels, your best choices are Channels 1, 6, and 11, as shown in Figure 1.4. For more than three networks, you'll have to put up with some amount of interference, but you can keep it to a minimum by assigning a new channel in the middle of an existing pair.

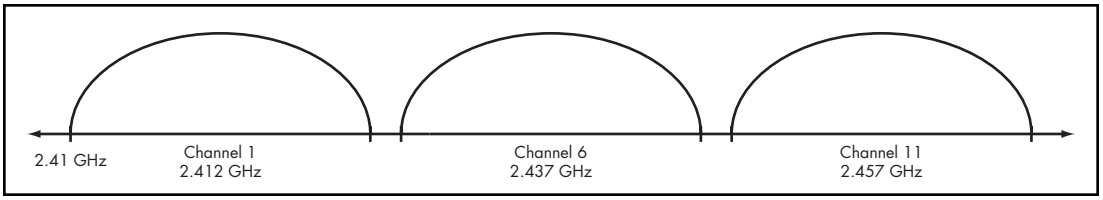

Figure 1.4: Channels 1, 6, and 11 do not interfere with one another

This all sounds like a more serious problem than it is likely to be on the ground. In practice, you can optimize the performance of your network by staying away from a channel that somebody else is using, but even if you and your neighbor are on adjacent channels, your networks will probably work just fine. You're more likely to have problems with interference from other devices using the 2.4 GHz band, such as cordless telephones and microwave ovens.

The 802.11 specifications and various national regulatory agencies (such as the Federal Communications Commission in the United States) have also set limits on the amount of transmitter power and antenna gain that a wireless Ethernet device can use. This restriction is intended to limit the distance that a network link can travel, and therefore to allow more networks to operate on the same channels without interference. We'll talk about methods for working around those power limits and extending the range of your wireless network without violating the law later in this book.

Moving Data Around

So now we have a bunch of radio transmitters and receivers that all operate on the same frequencies and all use the same kind of modulation (*modulation* is the method a radio uses to add some kind of content, such as voice or digital data, to a radio wave). The next step is to send some network data through those radios.

To begin, let's quickly recap the general structure of computer data and the methods networks use to move that data from one place to another. This is very basic stuff that might already be familiar to you, but bear with me for a couple of pages. This really will help you to understand how a wireless network operates.

Bits and Bytes

As you probably know, the processing unit of a computer can recognize only two information states: either a signal is present at the input to the processor, or there

is no signal there. These two conditions are usually described as 1 and 0, or on and off, or mark and space. Each instance of a 1 or a 0 is a *bit*.

Individual bits are not particularly useful, but when you string eight of them together (into a *byte*), you can have 256 different combinations. That's enough to assign different sequences to all the letters in the alphabet (both uppercase and lowercase), the ten digits from 0 to 9, spaces between words, and other symbols, such as punctuation marks and some letters used in foreign alphabets. A modern computer recognizes and processes several eight-bit bytes at the same time. When processing is complete, the computer uses the same bit code at its output. The output might be connected to a printer, a video display, or a data communications channel. Or it might be something else entirely, such as a series of flashing lights.

The inputs and outputs that we're concerned about here are the ones that form a communications circuit. Like the computer processor, a data channel can recognize only one bit at a time. Either there's a signal on the line or there isn't.

Over short distances, it's possible to send the data through a cable that carries eight (or some multiple of eight) signals *in parallel* through separate wires. Obviously, a parallel connection can be eight times faster than sending one bit through a single wire, but those eight wires cost eight times as much as a single wire. When you're trying to send the data over a long distance, that additional cost can be prohibitive. And when you're using existing circuits, such as telephone lines, you don't have any choice; you must find a way to send all eight bits through the same wire (or other media).

The solution is to transmit one bit at a time, with some additional bits and pauses that identify the beginning of each new byte. This is a *serial* data communications channel, because you're sending bits one after another. At this stage, it doesn't matter what medium you use to transmit those bits—it could be electrical impulses on a wire, or two different audio tones, or a series of flashing lights, or even a lot of notes attached to the legs of carrier pigeons—but you must have a method for converting the output of the computer to the signals used by the transmission medium, and converting it back again at the other end.

Error Checking

In a perfect transmission circuit, the signal that goes in at one end will be absolutely identical to the one that comes out at the other end. But in the real world, there's almost always some kind of noise that can interfere with the original pure signal. *Noise* is defined as anything that is added to the original signal; it can be caused by a lightning strike, interference from another communications channel, or dirt on an electrical contact someplace in the circuit (or in the case of those carrier pigeons, an attack by a marauding hawk). Whatever the source, noise in the channel can interrupt the flow of data. In a modern communications system, those bits are pouring through the circuit extremely quickly—millions of them every second—so even a noise hit for a fraction of a second can obliterate enough bits to turn your data into digital gibberish.

Therefore, you must include a process called *error checking* in your data stream. Error checking is accomplished by adding some kind of standard information called a *checksum* to each byte; if the receiving device discovers that the checksum is not what it expected, it instructs the transmitter to send the same byte again.

Handshaking

Of course, the computer that originates a message or a stream of data can't just jump online and start sending bytes. First it has to warn the device at the other end that it is ready to send and make sure the intended recipient is ready to accept data. To accomplish this, a series of *handshaking* requests and answers must surround the actual data.

The sequence of requests goes something like this:

Origin: "Hey destination! I have some data for you."

Destination: "Okay, origin, go ahead. I'm ready."

Origin: "Here comes the data."

Origin: Data data data data . . .

Origin: "That's the message. Did you get it?"

Destination: "I got something, but it appears to be damaged."

Origin: "Here it is again."

Origin: Data data data data . . .

Origin: "Did you get it that time?"

Destination: "Yup, I got it. I'm ready for more data."

Finding the Destination

A communication over a direct physical connection between the origin and destination doesn't need to include any kind of address or routing information as part of the message. You might have to set up the connection first (by placing a telephone call or plugging cables into a switchboard), but after you're connected, the link remains in place until you instruct the system to disconnect. This kind of connection is great for voice and for simple data links, but it's not particularly efficient for digital data on a complex network that serves many origins and destinations because it ties up the circuit all the time, even when no data is moving through the channel.

The alternative is to send your message to a switching center that will hold it until a link to the destination becomes available. This is known as a *store and forward* system. If the network has been properly designed for the type of data and the amount of traffic in the system, the waiting time will be insignificant. If the communications network covers a lot of territory, you can forward the message to one or more intermediate switching centers before it reaches the ultimate destination. The great advantage of this approach is that many messages can share the same circuits on an as-available basis.

To make the network even more efficient, you can divide messages that are longer than some arbitrary limit into separate pieces, called *packets*. Packets from

more than one message can travel together on the same circuit, combine with packets containing other messages as they travel between switching centers, and reassemble themselves into the original messages at the destination. Each data packet must contain yet another set of information: the address of the packet's destination, the sequence order of this packet relative to other packets in the original transmission, and so forth. Some of this information instructs the switching centers where to forward each packet, and other information tells the destination device how to reassemble the data in the packet back into the original message.

That same pattern is repeated every time you add another layer of activity to a communications system. Each layer may attach additional information to the original message and strip off that information after it has done whatever the added information instructed it to do. By the time a message travels from a laptop computer on a wireless network through an office LAN and an Internet gateway to a distant computer connected to another LAN, a dozen or more information attachments might be added and removed before the recipient reads the original text. A package of data that includes address and control information ahead of the bits that contain the content of the message, followed by an error-checking sequence, is called a *frame*. Both wired and wireless networks divide the data stream into frames that contain various forms of handshaking information along with the original data.

It might be helpful to think of these bits, bytes, packets, and frames as the digital version of a letter that you send through a complicated delivery system:

1. You write a letter and put it into an envelope. The address of the letter's recipient is on the outside of the envelope.
2. You take the letter to the mailroom, where a clerk puts your envelope into a bigger Express Mail envelope. The big envelope has the name and address of the office where the recipient works.
3. The mailroom clerk takes the big envelope to the post office, where another clerk puts it into a mail sack. The post office attaches a tag to the sack, marked with the location of the post office that serves the recipient's office.
4. The mail sack goes onto a truck to the airport, where it gets loaded into a shipping container along with other sacks going to the same destination city. The shipping container has a label that tells the freight handlers there's mail inside.
5. The freight handlers place the container inside an airplane.
6. At this point, your letter is inside your envelope, which is inside the Express Mail envelope, which is inside a mail sack, inside a container, inside an airplane. The airplane flies to another airport, near the destination city.
7. At the destination airport, the ground crew unloads the container from the airplane.
8. The freight handlers remove the sack from the shipping container and put it on another truck.
9. The truck takes the sack to a post office near the recipient's office.

10. At the post office, another mail clerk takes the big envelope out of the sack and gives it to a letter carrier.
11. The letter carrier delivers the big Express Mail envelope to the recipient's office.
12. The receptionist in the office takes your envelope out of the Express Mail envelope and gives it to the final recipient.
13. The recipient opens your envelope and reads the letter.

At each level, the information on the outside of the package tells somebody how to handle it, but that person doesn't much care what's inside. Neither you nor the person who ultimately reads your letter ever sees the big Express Mail envelope, the mail sack, the truck, the container, or the airplane, but every one of those packages plays an important part in moving your letter from here to there.

Instead of envelopes, sacks, and containers, an electronic message uses strings of data to tell the system how and where to handle your message, but the end result is just about the same. In the OSI network model, each mode of transportation would be a separate layer.

Fortunately, the network software automatically adds and removes all of the preambles, addresses, checksums, and other information, so you and the person receiving your message should never see them. However, each item added to the original data increases the size of the packet, frame, or other package, and therefore it increases the amount of time necessary to transmit the data through the network. Because the nominal data transfer speed includes all the overhead information along with the "real" data, the actual data transfer speed through a wireless network is a lot slower. In other words, even if your network connects at 11 Mbps, your actual file transfer speed might only be about 6 or 7 Mbps.

802.11b Wireless Network Controls

The 802.11b specification controls the way data moves through the Physical layer (the radio link), and it defines a Media Access Control (MAC) layer that handles the interface between the Physical layer and the rest of the network structure.

The Physical Layer

In an 802.11 network, the radio transmitter adds a 144-bit preamble to each packet, including 128 bits that the receiver uses to synchronize the receiver with the transmitter and a 16-bit start-of-frame field. This is followed by a 48-bit header that contains information about the data transfer speed, the length of the data contained in the packet, and an error-checking sequence. This header is called the *PHY preamble* because it controls the Physical layer of the communications link.

Because the header specifies the speed of the data that follows it, the preamble and the header are always transmitted at 1 Mbps. Therefore, even if a network link is operating at the full 11 Mbps, the effective data transfer speed is considerably slower. In practice, the best you can expect is about 85 percent of the nominal speed. And, of course, the other types of overhead in the data packets reduce the actual speed even more.

That 144-bit preamble is a holdover from the older and slower DSSS systems, and it has stayed in the specification to ensure that 802.11b devices will still be compatible with the older standards, but it really doesn't accomplish anything useful. So there's an optional alternative that uses a shorter, 72-bit preamble. In a short preamble, the synchronization field has 56 bits combined with the same 16-bit start-of-frame field used in long preambles. The 72-bit preamble is not compatible with old 802.11 hardware, but that doesn't matter as long as all the nodes in a network can recognize the short preamble format. In all other respects, a short preamble works just as well as a long one.

It takes the network a maximum of 192 milliseconds to handle a long preamble, compared to 96 milliseconds for a short preamble. In other words, the short preamble cuts the overhead on each packet in half. This makes a significant difference to the actual data throughput, especially for things like streaming audio and video and voice-over-Internet services.

Some manufacturers use the long preamble as the default, and others use the short preamble. It's usually possible to change the preamble length in the configuration software for network adapters and access points.

For most users, preamble length is one of those technical details that you don't have to understand, just as long as it's the same for all the devices in the network. Ten years ago, when telephone modems were the most common way to connect one computer to another, we all had to worry about setting the "data bits" and "stop bits" every time we placed a call through the modem. You probably never knew exactly what a stop bit was (it's the amount of time an old mechanical Teletype printer needed to return to the idle state after sending or receiving each byte), but you knew that it had to be the same at both ends. Preamble length is the same kind of obscure setting: it has to be the same on every node in a network, but most people neither know nor care what it means.

The MAC Layer

The MAC layer controls the traffic that moves through the radio network. It prevents data collisions and conflicts by using a set of rules called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), and it supports the security functions specified in the 802.11b standard. When the network includes more than one access point, the MAC layer associates each network client with the access point that provides the best signal quality.

When more than one node in the network tries to transmit data at the same time, CSMA/CA instructs all but one of the conflicting nodes to back off and try again later, and it allows the surviving node to send its packet. CSMA/CA works like this: when a network node is ready to send a packet, it listens for other signals first. If it doesn't hear anything, it waits for a random (but short) period of time and then listens again. If it still doesn't sense a signal, it transmits a packet. The device that receives the packet evaluates it, and if it's intact, the receiving mode returns an acknowledgement. But if the sending node does not receive the acknowledgement, it assumes that there has been a collision with another packet, so it waits for another random interval and then tries again.

CSMA/CA also has an optional feature that sets an access point (the bridge between the wireless LAN and the backbone network) as a point coordinator that

can grant priority to a network node that is trying to send time-critical data types, such as voice or streaming media.

The MAC layer can support two kinds of authentication to confirm that a network device is authorized to join the network: *open authentication* and *shared key authentication*. When you configure your network, all the nodes in the network must use the same kind of authentication.

The network supports all of these housekeeping functions in the MAC layer by exchanging (or trying to exchange) a series of control frames before it allows the higher layers to send data. It also sets several options on the network adapter:

- **Power mode** The network adapter supports two power modes: Continuous Aware Mode and Power Save Polling Mode. In Continuous Aware Mode, the radio receiver is always on and consuming power. In Power Save Polling Mode, the radio is idle much of the time, but it periodically polls the access point for new messages. As the name suggests, Power Save Polling Mode reduces the battery drain on portable devices such as laptop computers and PDAs.
- **Access control** The network adapter contains the access control that keeps unauthorized users out of the network. An 802.11b network can use two forms of access control: the SSID (the name of the network), and the MAC address (a unique string of characters that identifies each network node). Each network node must have the SSID programmed into it, or the access point will not associate with that node. An optional table of MAC addresses can restrict access to radios whose addresses are on the list.
- **WEP encryption** The network adapter controls the Wired Equivalent Privacy (WEP) encryption function. The network can use a 64-bit or a 128-bit encryption key to encode and decode data as it passes through the radio link.

Other Control Layers

All of the activity specified in the 802.11 standards takes place at the Physical and MAC layers. The higher layers control things like addressing and routing, data integrity, syntax, and the format of the data contained inside each packet. It doesn't make any difference to these higher layers whether they're moving packets through wires, fiber optic lines, or radio links. Therefore, you can use an 802.11b network with any kind of LAN or other network protocol. The same radios can handle TCP/IP, Novell NetWare, and all the other network protocols built into Windows, Unix, Mac OS, and other operating systems equally well.

Network Devices

Once we have defined the radio links and the data format, the next step is to set up a network structure. How do computers use the radios and the data format to actually exchange data?

802.11b networks include two categories of radios: *stations* and *access points*. A station is a computer, or some other device such as a printer, connected to a wireless network through an internal or external wireless network interface adapter.

An access point is the base station for a wireless network and a bridge between the wireless network and a traditional wired network.

Network Adapters

Network adapters for stations can take several physical forms:

- *Plug-in PC Cards that fit the PCMCIA sockets in most laptop computers*
  To bypass the internal shielding in most computers, the antennas and status lights in most wireless PC Card adapters extend about an inch beyond the opening of the card socket. Other PC Card adapters have sockets for external antennas.
- *Internal network adapters on PCI cards that fit inside a desktop computer*
  Most PCI adapters are actually PCMCIA sockets that allow a user to plug a PC Card into the back of the computer, but a few are built directly on the PCI expansion cards. As an alternative to a rear-panel socket, separate PCMCIA sockets that fit a computer's external front-panel drive bays are available from Actiontec and several other manufacturers.
- *External USB adapters*
  USB adapters are often a better choice than PC Cards, because it's almost always easier to move an adapter at the end of a cable to a position with the best possible signal path to the nearest access point.
- *Internal wireless adapters built into laptop computers*
  Internal adapters are modules that plug into the computer's motherboard. They present the same appearance to the operating system as an external PC Card. The antennas for built-in radios are usually hidden inside the computer's fold-over screen.
- *Plug-in adapters for PDAs and other handheld devices*
- *Internal network interfaces built into other devices, such as Internet-capable telephone sets and office or household appliances*

A network adapter should work with any operating system, as long as a driver for that adapter is available. In practice, that means you can find Windows drivers for just about everything, but you will have fewer choices if you're using a computer running Mac OS, Linux, or Unix. You can find pointers to sources for Linux and Unix drivers in the chapters devoted to those operating systems later in this book.

Access Points

Access points are often combined with other network functions. It's quite possible to find a stand-alone access point that just plugs into a wired LAN through a data cable, but there are plenty of other options. Common access-point configurations include the following:

- Simple base stations with a bridge to an Ethernet port for connection to a LAN
- Base stations that include a switch, hub, or router with one or more wired Ethernet ports along with the wireless access point

- Broadband routers that provide a bridge between a cable modem or DSL port and the wireless access point
- Software access points that use one of the wireless network interface adapters as the base station
- Residential gateways that support a limited number of operating channels

As Figure 1.5 shows, the physical design of access points varies from one manufacturer to another. Some look like industrial devices that were intended to be placed out of sight on the floor or mounted on an inconspicuous wall, but others have swooshy "aerodynamic" shapes that appear to have been designed for the top of a coffee table. Some have internal antennas, others have short vertical whips permanently attached, and still others have connectors for external antennas (which may or may not be supplied with the access point). Regardless of its size and shape, every access point includes a radio that sends and receives messages and data between network stations and an Ethernet port that connects to a wired network.

Figure 1.5: Access points from Zoom and D-Link

Operating Modes

802.11b networks operate in two modes: *ad hoc* networks and *infrastructure* networks. As the name suggests, an ad hoc network is usually temporary. An ad hoc network is a self-contained group of stations with no connection to a larger LAN or the Internet. It includes two or more wireless stations with no access point or connection to the rest of the world. Ad hoc networks are also called peer-to-peer networks and Independent Basic Service Sets (IBSS). Figure 1.6 shows a simple ad hoc network.

Infrastructure networks have one or more access points, almost always connected to a wired network. Each wireless station exchanges messages and data with the access point, which relays them to other nodes on the wireless network or the wired LAN. Any network that requires a wired connection through an access point to a printer, a file server or an Internet gateway is an infrastructure network. Figure 1.7 shows an infrastructure network.

An infrastructure network with just one base station is also called a Basic Service Set (BSS). When the wireless network uses two or more access points, the

network structure is an Extended Service Set (ESS). Remember from a few pages back that the technical name for a network ID is the SSID? You might also see it called a BSSID if the network has just one access point, or an ESSID when it has two or more access points.

A network with more than one access point (an Extended Service Set) creates several new complications. First, the network must include a way for only one base station to handle data from a particular station, even if the station is within range of more than one base station. And if the station is moving during a network session,

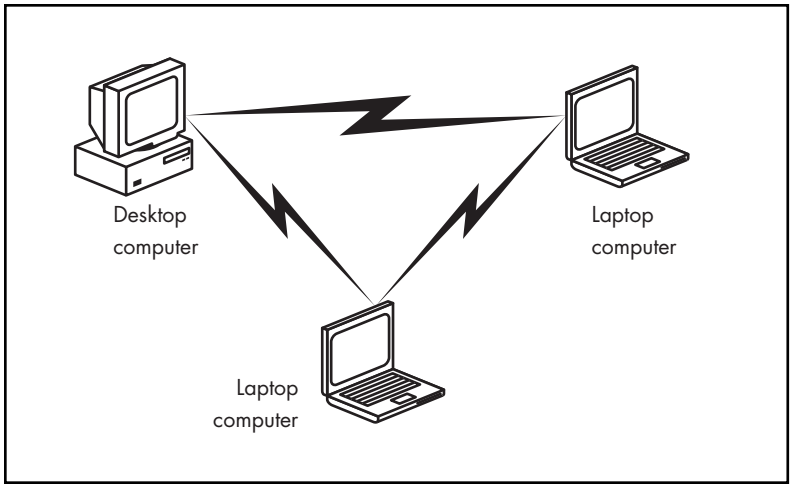

Figure 1.6: An ad hoc wireless network with three stations

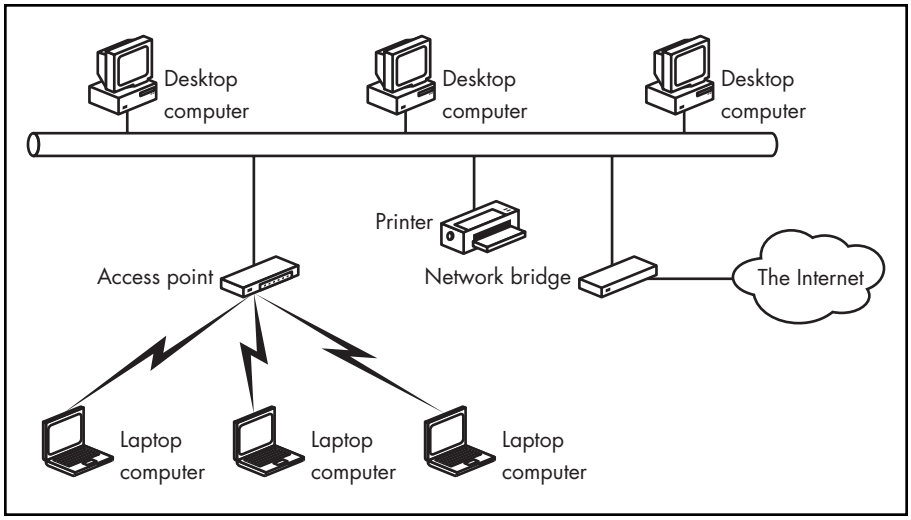

Figure 1.7: A simple infrastructure network

or if some kind of local interference crops up near the first access point, the network might have to hand off the connection from one access point to another. An 802.11b network handles this problem by associating a station with only one access point at a time and ignoring the signals from stations that are not associated. When the signal fades at one access point and improves at another, or the amount of traffic forces the network to rebalance the load, the network will reassociate the station with a new access point that can provide acceptable service. If you think this sounds a lot like the way a cellular telephone system handles roaming, you're absolutely correct; even the terminology is the same—it's called *roaming*.

Putting It All Together

The radio link, the data structure, and the network architecture are the three essential elements that form the internal plumbing of an 802.11b wireless Ethernet network. Like the components of most other networks (and most plumbing systems, for that matter), these elements should be entirely transparent to the people using the network—if users can send and receive messages, read files, and perform other activities on the network, they should never have to worry about the low-level details.

Of course, this assumes that it always works exactly as it's supposed to work, and no users ever have to call a network help desk to ask why they can't read their email. Now that you have read this chapter, you probably know more about the way your wireless LAN moves messages from here to there than 95 percent of the people who use Wi-Fi networks, and you have a good chance of understanding the support person who tells you to make sure you're using Channel 11 or you need to change your preamble length, or your adapter is operating in infrastructure mode.