4

THE MEDIUM ACCESS CONTROL SUBLAYER

Network links can be divided into two categories: those using point-to-point connections and those using broadcast channels. We studied point-to-point links in Chap. 2; this chapter deals with broadcast links and their protocols.

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. To make this point, consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once, leading to chaos. In a face-to-face meeting, chaos is avoided by external means. For example, at a meeting, people raise their hands to request permission to speak. When only a single channel is available, it is much harder to determine who should go next. Many protocols for solving the problem are known. They form the contents of this chapter. In the literature, broadcast channels are sometimes referred to as **multiaccess channels** or **random access channels**.

The protocols used to determine who goes next on a multiaccess channel belong to a sublayer of the data link layer called the **MAC** (**Medium Access Control**) sublayer. The MAC sublayer is especially important in LANs, particularly wireless ones because wireless is naturally a broadcast channel. WANs, in contrast, use point-to-point links, except for satellite networks. Because multiaccess channels and LANs are so closely related, in this chapter we will discuss LANs in

general, including a few issues that are not strictly part of the MAC sublayer, but the main subject here will be control of the channel.

Technically, the MAC sublayer is the bottom part of the data link layer, so logically we should have studied it before examining all the point-to-point protocols in Chap. 3. Nevertheless, for most people, it is easier to understand protocols involving multiple parties after two-party protocols are well understood. For that reason we have deviated slightly from a strict bottom-up order of presentation.

4.1 THE CHANNEL ALLOCATION PROBLEM

The central theme of this chapter is how to allocate a single broadcast channel among competing users. The channel might be a portion of the wireless spectrum in a geographic region, or a single wire or optical fiber to which multiple nodes are connected. It does not matter. In both cases, the channel connects each user to all other users and any user who makes full use of the channel interferes with other users who also wish to use the channel.

We will first look at the shortcomings of static allocation schemes for bursty traffic. Then, we will lay out the key assumptions used to model the dynamic schemes that we examine in the following sections.

4.1.1 Static Channel Allocation

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is to chop up its capacity by using one of the multiplexing schemes we described in Sec. 2.5, such as FDM (Frequency Division Multiplexing). If there are N users, the bandwidth is divided into N equal-sized portions, with each user being assigned one portion. Since each user has a private frequency band, there is now no interference among users. When there is only a small and constant number of users, each of which has a steady stream or a heavy load of traffic, this division is a simple and efficient allocation mechanism. A wireless example is FM radio stations. Each station gets a portion of the FM band and uses it most of the time to broadcast its signal.

However, when the number of senders is large and varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. And if more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

Even assuming that the number of users could somehow be held constant at N, dividing the single available channel into some number of static subchannels is

inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. A static allocation is a poor fit to most computer systems, in which data traffic is extremely bursty, often with peak traffic to mean traffic ratios of 1000:1. Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen with a simple queueing theory calculation. Let us start by finding the mean time delay, T, to send a frame onto a channel of capacity C bps. We assume that the frames arrive randomly with an average arrival rate of λ frames/sec, and that the frames vary in length with an average length of $1/\mu$ bits. With these parameters, the service rate of the channel is μC frames/sec. A standard queueing theory result is

$$T = \frac{1}{\mu C - \lambda}$$

(For the curious, this result is for an "M/M/1" queue. It requires that the randomness of the times between frame arrivals and the frame lengths follow an exponential distribution, or equivalently be the result of a Poisson process.)

In our example, if C is 100 Mbps, the mean frame length, $1/\mu$, is 10,000 bits, and the frame arrival rate, λ, is 5000 frames/sec, then $T = 200$ μsec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000-bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 μsec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/N bps. The mean input rate on each of the subchannels will now be λ/N. Recomputing T, we get

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \tag{4-1}$$

The mean delay for the divided channel is N times worse than if all the frames were somehow magically arranged orderly in a big central queue. This same result says that a bank lobby full of ATM machines is better off having a single queue feeding all the machines than a separate queue in front of each machine.

Precisely the same arguments that apply to FDM also apply to other ways of statically dividing the channel. If we were to use time division multiplexing (TDM) and allocate each user every Nth time slot, if a user does not use the allocated slot, it would just lie fallow. The same would hold if we split up the networks physically. Using our previous example again, if we were to replace the 100-Mbps network with 10 networks of 10 Mbps each and statically allocate each user to one of them, the mean delay would jump from 200 μsec to 2 msec.

Since none of the traditional static channel allocation methods work well at all with bursty traffic, we will now explore dynamic methods.

4.1.2 Assumptions for Dynamic Channel Allocation

Before we get to the first of the many channel allocation methods in this chapter, it is worthwhile to carefully formulate the allocation problem. Underlying all the work done in this area are the following five key assumptions:

1. **Independent Traffic**. The model consists of N independent **stations** (e.g., computers, telephones), each with a program or user that generates frames for transmission. The expected number of frames generated in an interval of length Δt is $\lambda \Delta t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

2. **Single Channel**. A single channel is available for all communication. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign them different roles (e.g., priorities).

3. **Observable Collisions**. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a **collision**. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No errors other than those generated by collisions occur.

4. **Continuous or Slotted Time**. Time may be assumed continuous, in which case frame transmission can begin at any instant. Alternatively, time may be slotted or divided into discrete intervals (called slots). Frame transmissions must then begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

5. **Carrier Sense or No Carrier Sense**. With the carrier sense assumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

Some discussion of these assumptions is in order. The first one says that frame arrivals are independent, both across stations and at a particular station, and that frames are generated unpredictably but at a constant rate. Actually, this assumption is not a particularly good model of network traffic, as it is well known that packets come in bursts over a range of time scales (Paxson and Floyd, 1995; and Leland et al., 1994). Nonetheless, **Poisson models**, as they are frequently called, are useful because they are mathematically tractable. They help us analyze

protocols to understand roughly how performance changes over an operating range and how it compares with other designs.

The single-channel assumption is the heart of the model. No external ways to communicate exist. Stations cannot raise their hands to request that the teacher call on them, so we will have to come up with better solutions.

The remaining three assumptions depend on the engineering of the system, and we will say which assumptions hold when we examine a particular protocol.

The collision assumption is basic. Stations need some way to detect collisions if they are to retransmit frames rather than let them be lost. For wired channels, node hardware can be designed to detect collisions when they occur. The stations can then terminate their transmissions prematurely to avoid wasting capacity. This detection is much harder for wireless channels, so collisions are usually inferred after the fact by the lack of an expected acknowledgement frame. It is also possible for some frames involved in a collision to be successfully received, depending on the details of the signals and the receiving hardware. However, this situation is not the common case, so we will assume that all frames involved in a collision are lost. We will also see protocols that are designed to prevent collisions from occurring in the first place.

The reason for the two alternative assumptions about time is that slotted time can be used to improve performance. However, it requires the stations to follow a master clock or synchronize their actions with each other to divide time into discrete intervals. Hence, it is not always available. We will discuss and analyze systems with both kinds of time. For a given system, only one of them holds.

Similarly, a network may have carrier sensing or not have it. Wired networks will generally have carrier sense. Wireless networks cannot always use it effectively because not every station may be within radio range of every other station. Similarly, carrier sense will not be available in other settings in which a station cannot communicate directly with other stations, for example a cable modem in which stations must communicate via the cable headend. Note that the word ''carrier'' in this sense refers to a signal on the channel and has nothing to do with the common carriers (e.g., telephone companies) that date back to the days of the Pony Express.

To avoid any misunderstanding, it is worth noting that no multiaccess protocol guarantees reliable delivery. Even in the absence of collisions, the receiver may have copied some of the frame incorrectly for various reasons. Other parts of the link layer or higher layers provide reliability.

4.2 MULTIPLE ACCESS PROTOCOLS

Many algorithms for allocating a multiple access channel are known. In the following sections, we will study a small sample of the more interesting ones and give some examples of how they are commonly used in practice.

4.2.1 ALOHA

The story of our first MAC starts out in pristine Hawaii in the early 1970s. In this case, "pristine" can be interpreted as "not having a working telephone system." This did not make life more pleasant for researcher Norman Abramson and his colleagues at the University of Hawaii who were trying to connect users on remote islands to the main computer in Honolulu. Stringing their own cables under the Pacific Ocean was not in the cards, so they looked for a different solution.

The one they found used short-range radios, with each user terminal sharing the same upstream frequency to send frames to the central computer. It included a simple and elegant method to solve the channel allocation problem. Their work has been extended by many researchers since then (Schwartz and Abramson, 2009). Although Abramson's work, called the ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel.

We will discuss two versions of ALOHA here: pure and slotted. They differ with respect to whether time is continuous, as in the pure version, or divided into discrete slots into which all frames must fit.

Pure ALOHA

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. Senders need some way to find out if this is the case. In the ALOHA system, after each station has sent its frame to the central computer, this computer rebroadcasts the frame to all of the stations. A sending station can thus listen for the broadcast from the hub to see if its frame has gotten through. In other systems, such as wired LANs, the sender might be able to listen for collisions while transmitting.

If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are known as **contention** systems.

A sketch of frame generation in an ALOHA system is given in Fig. 4-1. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable-length frames.

Whenever two frames try to occupy the channel at the same time, there will be a collision (as seen in Fig. 4-1) and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame that has almost finished, both frames will be totally destroyed (i.e., have incorrect checksums) and both will have to be retransmitted later. The checksum does not (and should not) distinguish between a total loss and a near miss. Bad is bad.

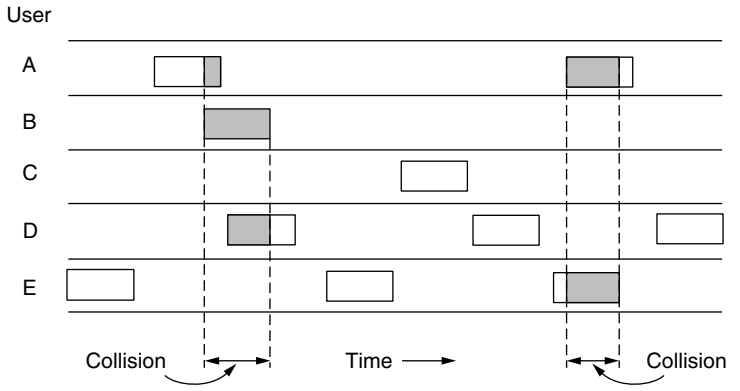

Figure 4-1. In pure ALOHA, frames are transmitted at completely arbitrary times.

An interesting question is: what is the efficiency of an ALOHA channel? In other words, what fraction of all transmitted frames escape collisions under these chaotic circumstances? Let us first consider an infinite collection of users typing at their terminals (stations). A user is always in one of two states: typing or waiting. Initially, all users are in the typing state. When a line is finished, the user stops typing, waiting for a response. The station then transmits a frame containing the line over the shared channel to the central computer and checks the channel to see if it was successful. If so, the user sees the reply and goes back to typing. If not, the user continues to wait while the station retransmits the frame over and over until it has been successfully sent.

Let the "frame time" denote the amount of time needed to transmit the standard, fixed-length frame (i.e., the frame length divided by the bit rate). At this point, we assume that the new frames generated by the stations are well modeled by a Poisson distribution with a mean of N frames per frame time. (The infinite-population assumption is needed to ensure that N does not decrease as users become blocked.) If $N > 1$, the user community is generating frames at a higher rate than the channel can handle, and nearly every frame will suffer a collision. For reasonable throughput, we would expect $0 < N < 1$.

In addition to the new frames, the stations also generate retransmissions of frames that previously suffered collisions. Let us further assume that the old and new frames combined are well modeled by a Poisson distribution, with mean of G frames per frame time. Clearly, $G \geq N$. At low load (i.e., $N \approx 0$), there will be few collisions, hence few retransmissions, so $G \approx N$. At high load, there will be many collisions, so $G > N$. Under all loads, the throughput, S, is just the offered load, G, times the probability, P_0, of a transmission succeeding—that is, $S = GP_0$, where P_0 is the probability that a frame does not suffer a collision.

A frame will not suffer a collision if no other frames are sent within one frame time of its start, as shown in Fig. 4-2. Under what conditions will the

shaded frame arrive undamaged? Let t be the time required to send one frame. If any other user has generated a frame between time t_0 and $t_0 + t$, the end of that frame will collide with the beginning of the shaded one. In fact, the shaded frame's fate was already sealed even before the first bit was sent, but since in pure ALOHA a station does not listen to the channel before transmitting, it has no way of knowing that another frame was already underway. Similarly, any other frame started between $t_0 + t$ and $t_0 + 2t$ will bump into the end of the shaded frame.

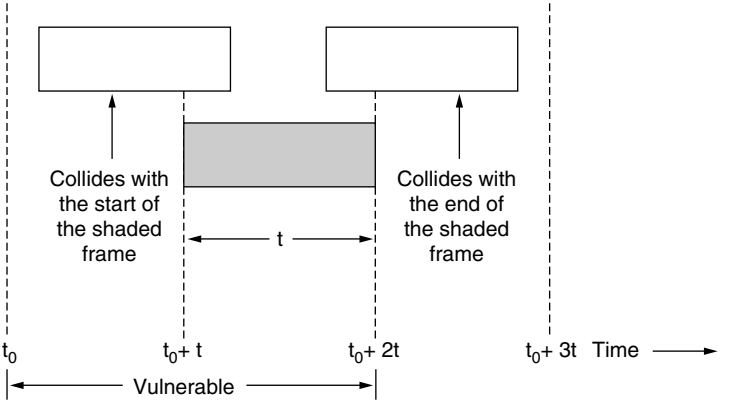

Figure 4-2. Vulnerable period for the shaded frame.

The probability that k frames are generated during a given frame time, in which G frames are expected, is given by the Poisson distribution

$$\Pr[k] = \frac{G^k \, e^{-G}}{k!} \tag{4-2}$$

so the probability of zero frames is just e^{-G}. In an interval two frame times long, the mean number of frames generated is $2G$. The probability of no frames being initiated during the entire vulnerable period is thus given by $P_0 = e^{-2G}$. Using $S = GP_0$, we get

$$S = Ge^{-2G}$$

The relation between the offered traffic and the throughput is shown in Fig. 4-3. The maximum throughput occurs at $G = 0.5$, with $S = 1/2e$, which is about 0.184. In other words, the best we can hope for is a channel utilization of 18%. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100% success rate.

Slotted ALOHA

Soon after ALOHA came onto the scene, Roberts (1972) published a method for doubling the capacity of an ALOHA system. His proposal was to divide time into discrete intervals called **slots**, each interval corresponding to one frame. This

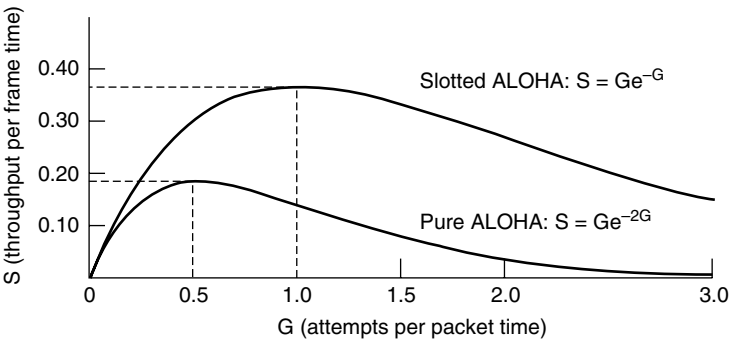

Figure 4-3. Throughput versus offered traffic for ALOHA systems.

approach requires the users to agree on slot boundaries. One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

In Roberts' method, which has come to be known as **slotted ALOHA**—in contrast to Abramson's **pure ALOHA**—a station is not permitted to send whenever the user types a line. Instead, it is required to wait for the beginning of the next slot. Thus, the continuous time ALOHA is turned into a discrete time one. This halves the vulnerable period. To see this, look at Fig. 4-3 and imagine the collisions that are now possible. The probability of no other traffic during the same slot as our test frame is then e^{-G}, which leads to

$$S = Ge^{-G} \tag{4-3}$$

As you can see from Fig. 4-3, slotted ALOHA peaks at $G = 1$, with a throughput of $S = 1/e$ or about 0.368, twice that of pure ALOHA. If the system is operating at $G = 1$, the probability of an empty slot is 0.368 (from Eq. 4-2). The best we can hope for using slotted ALOHA is 37% of the slots empty, 37% successes, and 26% collisions. Operating at higher values of G reduces the number of empties but increases the number of collisions exponentially. To see how this rapid growth of collisions with G comes about, consider the transmission of a test frame. The probability that it will avoid a collision is e^{-G}, which is the probability that all the other stations are silent in that slot. The probability of a collision is then just $1 - e^{-G}$. The probability of a transmission requiring exactly k attempts (i.e., $k - 1$ collisions followed by one success) is

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

The expected number of transmissions, E, per line typed at a terminal is then

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

As a result of the exponential dependence of E upon G, small increases in the channel load can drastically reduce its performance.

Slotted ALOHA is notable for a reason that may not be initially obvious. It was devised in the 1970s, used in a few early experimental systems, then almost forgotten. When Internet access over the cable was invented, all of a sudden there was a problem of how to allocate a shared channel among multiple competing users. Slotted ALOHA was pulled out of the garbage can to save the day. Later, having multiple RFID tags talk to the same RFID reader presented another variation on the same problem. Slotted ALOHA, with a dash of other ideas mixed in, again came to the rescue. It has often happened that protocols that are perfectly valid fall into disuse for political reasons (e.g., some big company wants everyone to do things its way) or due to ever-changing technology trends. Then, years later some clever person realizes that a long-discarded protocol solves his current problem. For this reason, in this chapter we will study a number of elegant protocols that are not currently in widespread use but might easily be used in future applications, provided that enough network designers are aware of them. Of course, we will also study many protocols that are in current use as well.

4.2.2 Carrier Sense Multiple Access Protocols

With slotted ALOHA, the best channel utilization that can be achieved is $1/e$. This low result is hardly surprising, since with stations transmitting at will, without knowing what the other stations are doing there are bound to be many collisions. In LANs, however, it is often possible for stations to detect what other stations are doing, and thus adapt their behavior accordingly. These networks can achieve a much better utilization than $1/e$. In this section, we will discuss some protocols for improving performance.

Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called **carrier sense protocols**. A number of them have been proposed, and they were long ago analyzed in detail. For example, see Kleinrock and Tobagi (1975). Below we will look at several versions of carrier sense protocols.

Persistent and Nonpersistent CSMA

The first carrier sense protocol that we will study here is called **1-persistent CSMA** (**Carrier Sense Multiple Access**). That is a bit of a mouthful for the simplest CSMA scheme. When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. If the channel is idle, the stations sends its data. Otherwise, if the channel is busy, the station just waits until it becomes idle. Then the station transmits a frame. If a collision occurs, the

station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmits with a probability of 1 when it finds the channel idle.

You might expect that this scheme avoids collisions except for the rare case of simultaneous sends, but it in fact it does not. If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends, and then both will begin transmitting exactly simultaneously, resulting in a collision. If they were not so impatient, there would be fewer collisions.

More subtly, the propagation delay has an important effect on collisions. There is a chance that just after a station begins sending, another station will become ready to send and sense the channel. If the first station's signal has not yet reached the second one, the latter will sense an idle channel and will also begin sending, resulting in a collision. This chance depends on the number of frames that fit on the channel, or the **bandwidth-delay product** of the channel. If only a tiny fraction of a frame fits on the channel, which is the case in most LANs since the propagation delay is small, the chance of a collision happening is small. The larger the bandwidth-delay product, the more important this effect becomes, and the worse the performance of the protocol.

Even so, this protocol has better performance than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame. Exactly the same holds for slotted ALOHA.

A second carrier sense protocol is **nonpersistent CSMA**. In this protocol, a conscious attempt is made to be less greedy than in the previous one. As before, a station senses the channel when it wants to send a frame, and if no one else is sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

The last protocol is **p-persistent CSMA**. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability $q = 1 - p$, it defers until the next slot. If that slot is also idle, it either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting. In the latter case, the unlucky station acts as if there had been a collision (i.e., it waits a random time and starts again). If the station initially senses that the channel is busy, it waits until the next slot and applies the above algorithm. IEEE 802.11 uses a refinement of p-persistent CSMA that we will discuss in Sec. 4.4.

Figure 4-4 shows the computed throughput versus offered traffic for all three protocols, as well as for pure and slotted ALOHA.

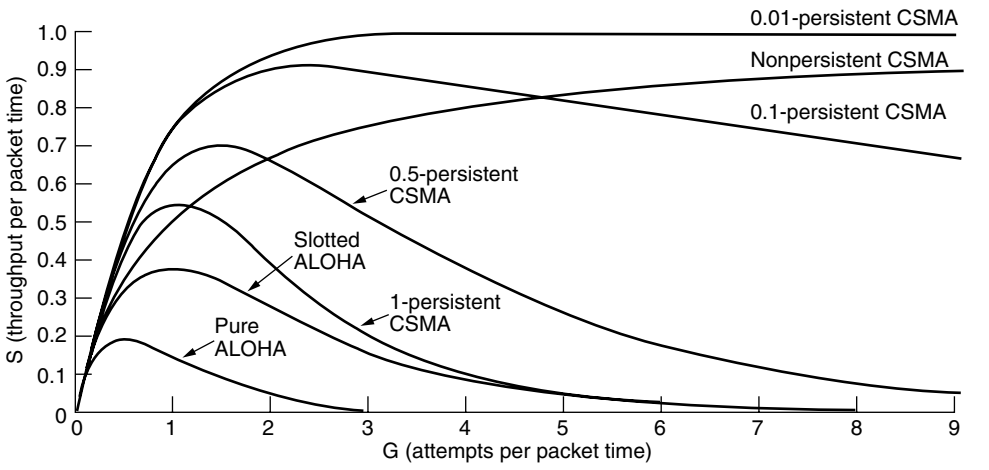

Figure 4-4. Comparison of the channel utilization versus load for various random access protocols.

CSMA with Collision Detection

Persistent and nonpersistent CSMA protocols are definitely an improvement over ALOHA because they ensure that no station begins to transmit while the channel is busy. However, if two stations sense the channel to be idle and begin transmitting simultaneously, their signals will still collide. Another improvement is for the stations to quickly detect the collision and abruptly stop transmitting, (rather than finishing them) since they are irretrievably garbled anyway. This strategy saves time and bandwidth.

This protocol, known as **CSMA/CD** (**CSMA with Collision Detection**), is the basis of the classic Ethernet LAN, so it is worth devoting some time to looking at it in detail. It is important to realize that collision detection is an analog process. The station's hardware must listen to the channel while it is transmitting. If the signal it reads back is different from the signal it is putting out, it knows that a collision is occurring. The implications are that a received signal must not be tiny compared to the transmitted signal (which is difficult for wireless, as received signals may be 1,000,000 times weaker than transmitted signals) and that the modulation must be chosen to allow collisions to be detected (e.g., a collision of two 0-volt signals may well be impossible to detect).

CSMA/CD, as well as many other LAN protocols, uses the conceptual model of Fig. 4-5. At the point marked t_0, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision. If a station detects a collision, it aborts its transmission, waits a random period of time, and then tries again (assuming that no other station has started transmitting in the

meantime). Therefore, our model for CSMA/CD will consist of alternating contention and transmission periods, with idle periods occurring when all stations are quiet (e.g., for lack of work).

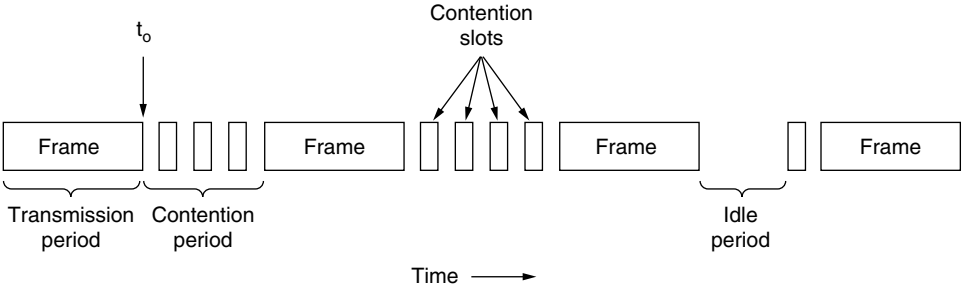

Figure 4-5. CSMA/CD can be in contention, transmission, or idle state.

Now let us look at the details of the contention algorithm. Suppose that two stations both begin transmitting at exactly time t_0. How long will it take them to realize that they have collided? The answer is vital to determining the length of the contention period and hence what the delay and throughput will be.

The minimum time to detect the collision is just the time it takes the signal to propagate from one station to the other. Based on this information, you might think that a station that has not heard a collision for a time equal to the full cable propagation time after starting its transmission can be sure it has seized the cable. By "seized," we mean that all other stations know it is transmitting and will not interfere. This conclusion is wrong.

Consider the following worst-case scenario. Let the time for a signal to propagate between the two farthest stations be τ. At t_0, one station begins transmitting. At $t_0 + \tau - \varepsilon$, an instant before the signal arrives at the most distant station, that station also begins transmitting. Of course, it detects the collision almost instantly and stops, but the little noise burst caused by the collision does not get back to the original station until time $2\tau - \varepsilon$. In other words, in the worst case a station cannot be sure that it has seized the channel until it has transmitted for 2τ without hearing a collision.

With this understanding, we can think of CSMA/CD contention as a slotted ALOHA system with a slot width of 2τ. On a 1-km long coaxial cable, $\tau \approx 5$ μsec. The difference for CSMA/CD compared to slotted ALOHA is that slots in which only one station transmits (i.e., in which the channel is seized) are followed by the rest of a frame. This difference will greatly improve performance if the frame time is much longer than the propagation time.

4.2.3 Collision-Free Protocols

Although collisions do not occur with CSMA/CD once a station has unambiguously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the

bandwidth-delay product is large, such as when the cable is long (i.e., large τ) and the frames are short. Not only do collisions reduce bandwidth, but they make the time to send a frame variable, which is not a good fit for real-time traffic such as voice over IP. CSMA/CD is also not universally applicable.

In this section, we will examine some protocols that resolve the contention for the channel without any collisions at all, not even during the contention period. Most of these protocols are not currently used in major systems, but in a rapidly changing field, having some protocols with excellent properties available for future systems is often a good thing.

In the protocols to be described, we assume that there are exactly N stations, each programmed with a unique address from 0 to $N - 1$. It does not matter that some stations may be inactive part of the time. We also assume that propagation delay is negligible. The basic question remains: which station gets the channel after a successful transmission? We continue using the model of Fig. 4-5 with its discrete contention slots.

A Bit-Map Protocol

In our first collision-free protocol, the **basic bit-map method**, each contention period consists of exactly N slots. If station 0 has a frame to send, it transmits a 1 bit during the slot 0. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 bit during slot 1, but only if it has a frame queued. In general, station j may announce that it has a frame to send by inserting a 1 bit into slot j. After all N slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting frames in numerical order (see Fig. 4-6).

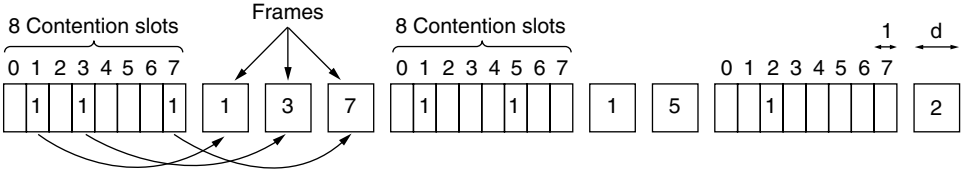

Figure 4-6. The basic bit-map protocol.

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another N-bit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again.

Protocols like this in which the desire to transmit is broadcast before the actual transmission are called **reservation protocols** because they reserve channel ownership in advance and prevent collisions. Let us briefly analyze the performance of this protocol. For convenience, we will measure time in units of the contention bit slot, with data frames consisting of d time units.

Under conditions of low load, the bit map will simply be repeated over and over, for lack of data frames. Consider the situation from the point of view of a low-numbered station, such as 0 or 1. Typically, when it becomes ready to send, the ''current'' slot will be somewhere in the middle of the bit map. On average, the station will have to wait $N/2$ slots for the current scan to finish and another full N slots for the following scan to run to completion before it may begin transmitting.

The prospects for high-numbered stations are brighter. Generally, these will only have to wait half a scan ($N/2$ bit slots) before starting to transmit. High-numbered stations rarely have to wait for the next scan. Since low-numbered stations must wait on average $1.5N$ slots and high-numbered stations must wait on average $0.5N$ slots, the mean for all stations is N slots.

The channel efficiency at low load is easy to compute. The overhead per frame is N bits and the amount of data is d bits, for an efficiency of $d/(d + N)$.

At high load, when all the stations have something to send all the time, the N-bit contention period is prorated over N frames, yielding an overhead of only 1 bit per frame, or an efficiency of $d/(d + 1)$. The mean delay for a frame is equal to the sum of the time it queues inside its station, plus an additional $(N - 1)d + N$ once it gets to the head of its internal queue. This interval is how long it takes to wait for all other stations to have their turn sending a frame and another bitmap.

Token Passing

The essence of the bit-map protocol is that it lets every station transmit a frame in turn in a predefined order. Another way to accomplish the same thing is to pass a small message called a **token** from one station to the next in the same predefined order. The token represents permission to send. If a station has a frame queued for transmission when it receives the token, it can send that frame before it passes the token to the next station. If it has no queued frame, it simply passes the token.

In a **token ring** protocol, the topology of the network is used to define the order in which stations send. The stations are connected one to the next in a single ring. Passing the token to the next station then simply consists of receiving the token in from one direction and transmitting it out in the other direction, as seen in Fig. 4-7. Frames are also transmitted in the direction of the token. This way they will circulate around the ring and reach whichever station is the destination. However, to stop the frame circulating indefinitely (like the token), some station needs

to remove it from the ring. This station may be either the one that originally sent the frame, after it has gone through a complete cycle, or the station that was the intended recipient of the frame.

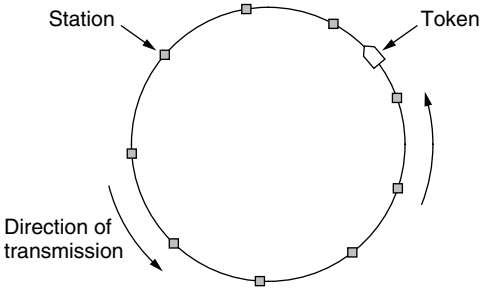

Figure 4-7. Token ring.

Note that we do not need a physical ring to implement token passing. The channel connecting the stations might instead be a single long bus. Each station then uses the bus to send the token to the next station in the predefined sequence. Possession of the token allows a station to use the bus to send one frame, as before. This protocol is called **token bus**.

The performance of token passing is similar to that of the bit-map protocol, though the contention slots and frames of one cycle are now intermingled. After sending a frame, each station must wait for all N stations (including itself) to send the token to their neighbors and the other $N - 1$ stations to send a frame, if they have one. A subtle difference is that, since all positions in the cycle are equivalent, there is no bias for low- or high-numbered stations. For token ring, each station is also sending the token only as far as its neighboring station before the protocol takes the next step. Each token does not need to propagate to all stations before the protocol advances to the next step.

Token rings have cropped up as MAC protocols with some consistency. An early token ring protocol (called "Token Ring" and standardized as IEEE 802.5) was popular in the 1980s as an alternative to classic Ethernet. In the 1990s, a much faster token ring called **FDDI** (**Fiber Distributed Data Interface**) was beaten out by switched Ethernet. In the 2000s, a token ring called **RPR** (**Resilient Packet Ring**) was defined as IEEE 802.17 to standardize the mix of metropolitan area rings in use by ISPs. We wonder what the 2010s will have to offer.

Binary Countdown

A problem with the basic bit-map protocol, and by extension token passing, is that the overhead is 1 bit per station, so it does not scale well to networks with thousands of stations. We can do better than that by using binary station addresses with a channel that combines transmissions. A station wanting to use the

channel now broadcasts its address as a binary bit string, starting with the high-order bit. All addresses are assumed to be the same length. The bits in each address position from different stations are BOOLEAN ORed together by the channel when they are sent at the same time. We will call this protocol **binary countdown**. It was used in Datakit (Fraser, 1987). It implicitly assumes that the transmission delays are negligible so that all stations see asserted bits essentially instantaneously.

To avoid conflicts, an arbitration rule must be applied: as soon as a station sees that a high-order bit position that is 0 in its address has been overwritten with a 1, it gives up. For example, if stations 0010, 0100, 1001, and 1010 are all trying to get the channel, in the first bit time the stations transmit 0, 0, 1, and 1, respectively. These are ORed together to form a 1. Stations 0010 and 0100 see the 1 and know that a higher-numbered station is competing for the channel, so they give up for the current round. Stations 1001 and 1010 continue.

The next bit is 0, and both stations continue. The next bit is 1, so station 1001 gives up. The winner is station 1010 because it has the highest address. After winning the bidding, it may now transmit a frame, after which another bidding cycle starts. The protocol is illustrated in Fig. 4-8. It has the property that higher-numbered stations have a higher priority than lower-numbered stations, which may be either good or bad, depending on the context.

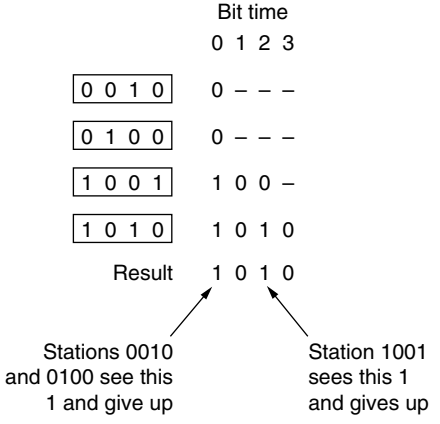

Figure 4-8. The binary countdown protocol. A dash indicates silence.

The channel efficiency of this method is $d/(d + \log_2 N)$. If, however, the frame format has been cleverly chosen so that the sender's address is the first field in the frame, even these $\log_2 N$ bits are not wasted, and the efficiency is 100%.

Binary countdown is an example of a simple, elegant, and efficient protocol that is waiting to be rediscovered. Hopefully, it will find a new home some day.

4.2.4 Limited-Contention Protocols

We have now considered two basic strategies for channel acquisition in a broadcast network: contention, as in CSMA, and collision-free protocols. Each strategy can be rated as to how well it does with respect to the two important performance measures, delay at low load and channel efficiency at high load. Under conditions of light load, contention (i.e., pure or slotted ALOHA) is preferable due to its low delay (since collisions are rare). As the load increases, contention becomes increasingly less attractive because the overhead associated with channel arbitration becomes greater. Just the reverse is true for the collision-free protocols. At low load, they have relatively high delay but as the load increases, the channel efficiency improves (since the overheads are fixed).

Obviously, it would be nice if we could combine the best properties of the contention and collision-free protocols, arriving at a new protocol that used contention at low load to provide low delay, but used a collision-free technique at high load to provide good channel efficiency. Such protocols, which we will call **limited-contention protocols**, do in fact exist, and will conclude our study of carrier sense networks.

Up to now, the only contention protocols we have studied have been symmetric. That is, each station attempts to acquire the channel with some probability, p, with all stations using the same p. Interestingly enough, the overall system performance can sometimes be improved by using a protocol that assigns different probabilities to different stations.

Before looking at the asymmetric protocols, let us quickly review the performance of the symmetric case. Suppose that k stations are contending for channel access. Each has a probability p of transmitting during each slot. The probability that some station successfully acquires the channel during a given slot is the probability that any one station transmits, with probability p, and all other $k-1$ stations defer, each with probability $1-p$. This value is $kp(1-p)^{k-1}$. To find the optimal value of p, we differentiate with respect to p, set the result to zero, and solve for p. Doing so, we find that the best value of p is $1/k$. Substituting $p = 1/k$, we get

$$\Pr[\text{success with optimal } p] = \left[\frac{k-1}{k}\right]^{k-1} \tag{4-4}$$

This probability is plotted in Fig. 4-9. For small numbers of stations, the chances of success are good, but as soon as the number of stations reaches even five, the probability has dropped close to its asymptotic value of $1/e$.

From Fig. 4-9, it is fairly obvious that the probability of some station acquiring the channel can be increased only by decreasing the amount of competition. The limited-contention protocols do precisely that. They first divide the stations into (not necessarily disjoint) groups. Only the members of group 0 are permitted

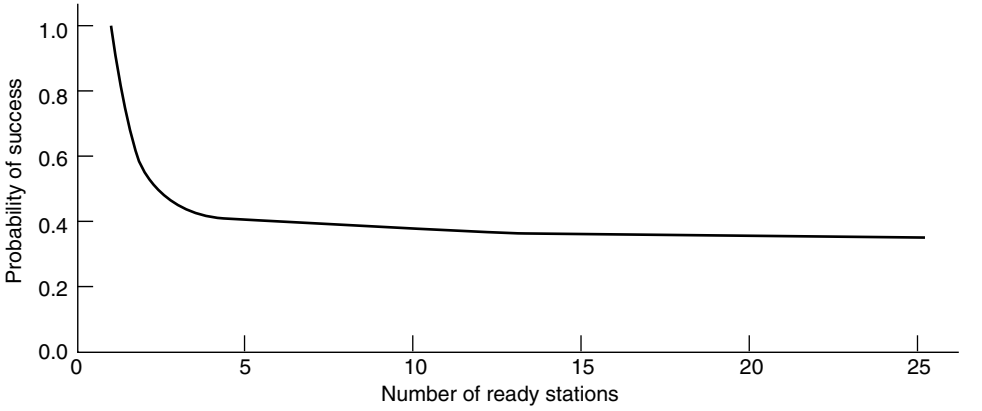

Figure 4-9. Acquisition probability for a symmetric contention channel.

to compete for slot 0. If one of them succeeds, it acquires the channel and transmits its frame. If the slot lies fallow or if there is a collision, the members of group 1 contend for slot 1, etc. By making an appropriate division of stations into groups, the amount of contention for each slot can be reduced, thus operating each slot near the left end of Fig. 4-9.

The trick is how to assign stations to slots. Before looking at the general case, let us consider some special cases. At one extreme, each group has but one member. Such an assignment guarantees that there will never be collisions because at most one station is contending for any given slot. We have seen such protocols before (e.g., binary countdown). The next special case is to assign two stations per group. The probability that both will try to transmit during a slot is p^2, which for a small p is negligible. As more and more stations are assigned to the same slot, the probability of a collision grows, but the length of the bit-map scan needed to give everyone a chance shrinks. The limiting case is a single group containing all stations (i.e., slotted ALOHA). What we need is a way to assign stations to slots dynamically, with many stations per slot when the load is low and few (or even just one) station per slot when the load is high.

The Adaptive Tree Walk Protocol

One particularly simple way of performing the necessary assignment is to use the algorithm devised by the U.S. Army for testing soldiers for syphilis during World War II (Dorfman, 1943). In short, the Army took a blood sample from N soldiers. A portion of each sample was poured into a single test tube. This mixed sample was then tested for antibodies. If none were found, all the soldiers in the group were declared healthy. If antibodies were present, two new mixed samples

were prepared, one from soldiers 1 through $N/2$ and one from the rest. The process was repeated recursively until the infected soldiers were determined.

For the computerized version of this algorithm (Capetanakis, 1979), it is convenient to think of the stations as the leaves of a binary tree, as illustrated in Fig. 4-10. In the first contention slot following a successful frame transmission, slot 0, all stations are permitted to try to acquire the channel. If one of them does so, fine. If there is a collision, then during slot 1 only those stations falling under node 2 in the tree may compete. If one of them acquires the channel, the slot following the frame is reserved for those stations under node 3. If, on the other hand, two or more stations under node 2 want to transmit, there will be a collision during slot 1, in which case it is node 4's turn during slot 2.

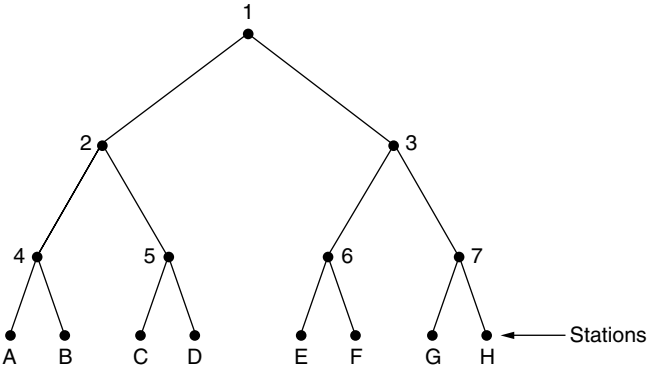

Figure 4-10. The tree for eight stations.

In essence, if a collision occurs during slot 0, the entire tree is searched, depth first, to locate all ready stations. Each bit slot is associated with some particular node in the tree. If a collision occurs, the search continues recursively with the node's left and right children. If a bit slot is idle or if only one station transmits in it, the searching of its node can stop because all ready stations have been located. (Were there more than one, there would have been a collision.)

When the load on the system is heavy, it is hardly worth the effort to dedicate slot 0 to node 1 because that makes sense only in the unlikely event that precisely one station has a frame to send. Similarly, one could argue that nodes 2 and 3 should be skipped as well for the same reason. Put in more general terms, at what level in the tree should the search begin? Clearly, the heavier the load, the farther down the tree the search should begin. We will assume that each station has a good estimate of the number of ready stations, q, for example, from monitoring recent traffic.

To proceed, let us number the levels of the tree from the top, with node 1 in Fig. 4-10 at level 0, nodes 2 and 3 at level 1, etc. Notice that each node at level i

has a fraction 2^{-i} of the stations below it. If the q ready stations are uniformly distributed, the expected number of them below a specific node at level i is just $2^{-i}q$. Intuitively, we would expect the optimal level to begin searching the tree to be the one at which the mean number of contending stations per slot is 1, that is, the level at which $2^{-i}q = 1$. Solving this equation, we find that $i = \log_2 q$.

Numerous improvements to the basic algorithm have been discovered and are discussed in some detail by Bertsekas and Gallager (1992). For example, consider the case of stations G and H being the only ones wanting to transmit. At node 1 a collision will occur, so 2 will be tried and discovered idle. It is pointless to probe node 3 since it is guaranteed to have a collision (we know that two or more stations under 1 are ready and none of them are under 2, so they must all be under 3). The probe of 3 can be skipped and 6 tried next. When this probe also turns up nothing, 7 can be skipped and node G tried next.

4.2.5 Wireless LAN Protocols

A system of laptop computers that communicate by radio can be regarded as a wireless LAN, as we discussed in Sec. 1.5.3. Such a LAN is an example of a broadcast channel. It also has somewhat different properties than a wired LAN, which leads to different MAC protocols. In this section, we will examine some of these protocols. In Sec. 4.4, we will look at 802.11 (WiFi) in detail.

A common configuration for a wireless LAN is an office building with access points (APs) strategically placed around the building. The APs are wired together using copper or fiber and provide connectivity to the stations that talk to them. If the transmission power of the APs and laptops is adjusted to have a range of tens of meters, nearby rooms become like a single cell and the entire building becomes like the cellular telephony systems we studied in Chap. 2, except that each cell only has one channel. This channel is shared by all the stations in the cell, including the AP. It typically provides megabit/sec bandwidths, up to 600 Mbps.

We have already remarked that wireless systems cannot normally detect a collision while it is occurring. The received signal at a station may be tiny, perhaps a million times fainter than the signal that is being transmitted. Finding it is like looking for a ripple on the ocean. Instead, acknowledgements are used to discover collisions and other errors after the fact.

There is an even more important difference between wireless LANs and wired LANs. A station on a wireless LAN may not be able to transmit frames to or receive frames from all other stations because of the limited radio range of the stations. In wired LANs, when one station sends a frame, all other stations receive it. The absence of this property in wireless LANs causes a variety of complications.

We will make the simplifying assumption that each radio transmitter has some fixed range, represented by a circular coverage region within which another station can sense and receive the station's transmission. It is important to realize that

in practice coverage regions are not nearly so regular because the propagation of radio signals depends on the environment. Walls and other obstacles that attenuate and reflect signals may cause the range to differ markedly in different directions. But a simple circular model will do for our purposes.

A naive approach to using a wireless LAN might be to try CSMA: just listen for other transmissions and only transmit if no one else is doing so. The trouble is, this protocol is not really a good way to think about wireless because what matters for reception is interference at the receiver, not at the sender. To see the nature of the problem, consider Fig. 4-11, where four wireless stations are illustrated. For our purposes, it does not matter which are APs and which are laptops. The radio range is such that *A* and *B* are within each other's range and can potentially interfere with one another. *C* can also potentially interfere with both *B* and *D*, but not with *A*.

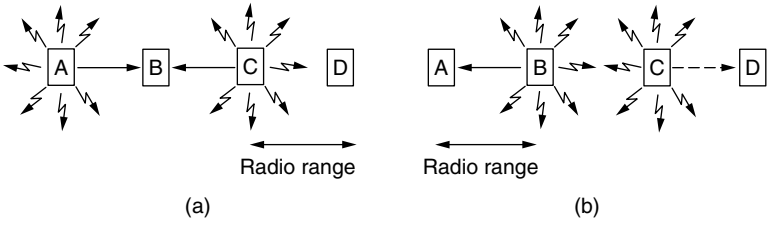

Figure 4-11. A wireless LAN. (a) *A* and *C* are hidden terminals when transmitting to *B*. (b) *B* and *C* are exposed terminals when transmitting to *A* and *D*.

First consider what happens when *A* and *C* transmit to *B*, as depicted in Fig. 4-11(a). If *A* sends and then *C* immediately senses the medium, it will not hear *A* because *A* is out of range. Thus *C* will falsely conclude that it can transmit to *B*. If *C* does start transmitting, it will interfere at *B*, wiping out the frame from *A*. (We assume here that no CDMA-type scheme is used to provide multiple channels, so collisions garble the signal and destroy both frames.) We want a MAC protocol that will prevent this kind of collision from happening because it wastes bandwidth. The problem of a station not being able to detect a potential competitor for the medium because the competitor is too far away is called the **hidden terminal problem**.

Now let us look at a different situation: *B* transmitting to *A* at the same time that *C* wants to transmit to *D,* as shown in Fig. 4-11(b). If *C* senses the medium, it will hear a transmission and falsely conclude that it may not send to *D* (shown as a dashed line). In fact, such a transmission would cause bad reception only in the zone between *B* and *C*, where neither of the intended receivers is located. We want a MAC protocol that prevents this kind of deferral from happening because it wastes bandwidth. The problem is called the **exposed terminal problem**.

The difficulty is that, before starting a transmission, a station really wants to know whether there is radio activity around the receiver. CSMA merely tells it

whether there is activity near the transmitter by sensing the carrier. With a wire, all signals propagate to all stations, so this distinction does not exist. However, only one transmission can then take place at once anywhere in the system. In a system based on short-range radio waves, multiple transmissions can occur simultaneously if they all have different destinations and these destinations are out of range of one another. We want this concurrency to happen as the cell gets larger and larger, in the same way that people at a party should not wait for everyone in the room to go silent before they talk; multiple conversations can take place at once in a large room as long as they are not directed to the same location.

An early and influential protocol that tackles these problems for wireless LANs is **MACA** (**Multiple Access with Collision Avoidance**) (Karn, 1990). The basic idea behind it is for the sender to stimulate the receiver into outputting a short frame, so stations nearby can detect this transmission and avoid transmitting for the duration of the upcoming (large) data frame. This technique is used instead of carrier sense.

MACA is illustrated in Fig. 4-12. Let us see how *A* sends a frame to *B*. *A* starts by sending an **RTS** (**Request To Send**) frame to *B*, as shown in Fig. 4-12(a). This short frame (30 bytes) contains the length of the data frame that will eventually follow. Then *B* replies with a **CTS** (**Clear To Send**) frame, as shown in Fig. 4-12(b). The CTS frame contains the data length (copied from the RTS frame). Upon receipt of the CTS frame, *A* begins transmission.

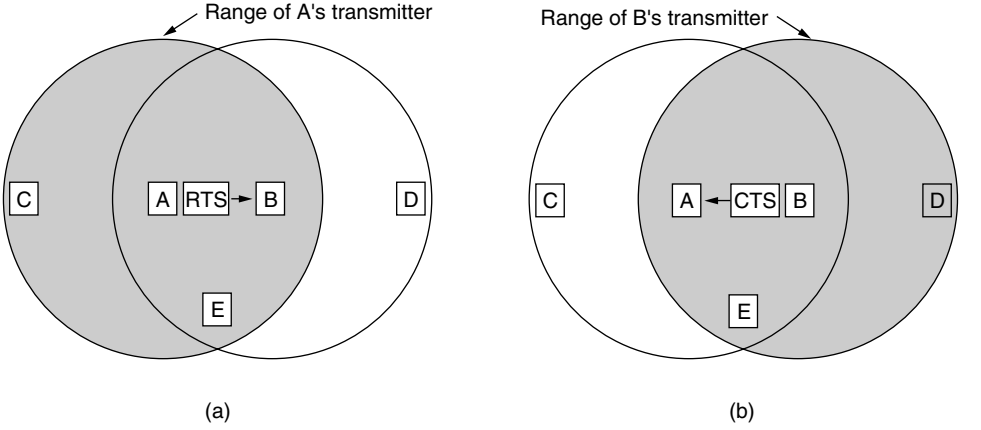

Figure 4-12. The MACA protocol. (a) *A* sending an RTS to *B*. (b) *B* responding with a CTS to *A*.

Now let us see how stations overhearing either of these frames react. Any station hearing the RTS is clearly close to *A* and must remain silent long enough for the CTS to be transmitted back to *A* without conflict. Any station hearing the CTS is clearly close to *B* and must remain silent during the upcoming data transmission, whose length it can tell by examining the CTS frame.

In Fig. 4-12, *C* is within range of *A* but not within range of *B*. Therefore, it hears the RTS from *A* but not the CTS from *B*. As long as it does not interfere with the CTS, it is free to transmit while the data frame is being sent. In contrast, *D* is within range of *B* but not *A*. It does not hear the RTS but does hear the CTS. Hearing the CTS tips it off that it is close to a station that is about to receive a frame, so it defers sending anything until that frame is expected to be finished. Station *E* hears both control messages and, like *D*, must be silent until the data frame is complete.

Despite these precautions, collisions can still occur. For example, *B* and *C* could both send RTS frames to *A* at the same time. These will collide and be lost. In the event of a collision, an unsuccessful transmitter (i.e., one that does not hear a CTS within the expected time interval) waits a random amount of time and tries again later.

4.3 ETHERNET

We have now finished our discussion of channel allocation protocols in the abstract, so it is time to see how these principles apply to real systems. Many of the designs for personal, local, and metropolitan area networks have been standardized under the name of IEEE 802. A few have survived but many have not, as we saw in Fig. 1-38. Some people who believe in reincarnation think that Charles Darwin came back as a member of the IEEE Standards Association to weed out the unfit. The most important of the survivors are 802.3 (Ethernet) and 802.11 (wireless LAN). Bluetooth (wireless PAN) is widely deployed but has now been standardized outside of 802.15. With 802.16 (wireless MAN), it is too early to tell. Please consult the 6th edition of this book to find out.

We will begin our study of real systems with Ethernet, probably the most ubiquitous kind of computer network in the world. Two kinds of Ethernet exist: **classic Ethernet**, which solves the multiple access problem using the techniques we have studied in this chapter; and **switched Ethernet**, in which devices called switches are used to connect different computers. It is important to note that, while they are both referred to as Ethernet, they are quite different. Classic Ethernet is the original form and ran at rates from 3 to 10 Mbps. Switched Ethernet is what Ethernet has become and runs at 100, 1000, and 10,000 Mbps, in forms called fast Ethernet, gigabit Ethernet, and 10 gigabit Ethernet. In practice, only switched Ethernet is used nowadays.

We will discuss these historical forms of Ethernet in chronological order showing how they developed. Since Ethernet and IEEE 802.3 are identical except for a minor difference (which we will discuss shortly), many people use the terms "Ethernet" and "IEEE 802.3" interchangeably. We will do so, too. For more information about Ethernet, see Spurgeon (2000).

4.3.1 Classic Ethernet Physical Layer

The story of Ethernet starts about the same time as that of ALOHA, when a student named Bob Metcalfe got his bachelor's degree at M.I.T. and then moved up the river to get his Ph.D. at Harvard. During his studies, he was exposed to Abramson's work. He became so interested in it that after graduating from Harvard, he decided to spend the summer in Hawaii working with Abramson before starting work at Xerox PARC (Palo Alto Research Center). When he got to PARC, he saw that the researchers there had designed and built what would later be called personal computers. But the machines were isolated. Using his knowledge of Abramson's work, he, together with his colleague David Boggs, designed and implemented the first local area network (Metcalfe and Boggs, 1976). It used a single long, thick coaxial cable and ran at 3 Mbps.

They called the system **Ethernet** after the *luminiferous ether*, through which electromagnetic radiation was once thought to propagate. (When the 19th-century British physicist James Clerk Maxwell discovered that electromagnetic radiation could be described by a wave equation, scientists assumed that space must be filled with some ethereal medium in which the radiation was propagating. Only after the famous Michelson-Morley experiment in 1887 did physicists discover that electromagnetic radiation could propagate in a vacuum.)

The Xerox Ethernet was so successful that DEC, Intel, and Xerox drew up a standard in 1978 for a 10-Mbps Ethernet, called the **DIX standard**. With a minor change, the DIX standard became the IEEE 802.3 standard in 1983. Unfortunately for Xerox, it already had a history of making seminal inventions (such as the personal computer) and then failing to commercialize on them, a story told in *Fumbling the Future* (Smith and Alexander, 1988). When Xerox showed little interest in doing anything with Ethernet other than helping standardize it, Metcalfe formed his own company, 3Com, to sell Ethernet adapters for PCs. It sold many millions of them.

Classic Ethernet snaked around the building as a single long cable to which all the computers were attached. This architecture is shown in Fig. 4-13. The first variety, popularly called **thick Ethernet,** resembled a yellow garden hose, with markings every 2.5 meters to show where to attach computers. (The 802.3 standard did not actually *require* the cable to be yellow, but it did *suggest* it.) It was succeeded by **thin Ethernet,** which bent more easily and made connections using industry-standard BNC connectors. Thin Ethernet was much cheaper and easier to install, but it could run for only 185 meters per segment (instead of 500 m with thick Ethernet), each of which could handle only 30 machines (instead of 100).

Each version of Ethernet has a maximum cable length per segment (i.e., unamplified length) over which the signal will propagate. To allow larger networks, multiple cables can be connected by **repeaters**. A repeater is a physical layer device that receives, amplifies (i.e., regenerates), and retransmits signals in both directions. As far as the software is concerned, a series of cable segments

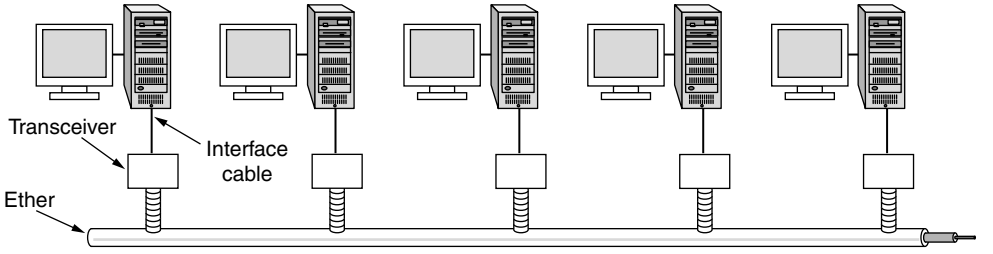

Figure 4-13. Architecture of classic Ethernet.

connected by repeaters is no different from a single cable (except for a small amount of delay introduced by the repeaters).

Over each of these cables, information was sent using the Manchester encoding we studied in Sec. 2.5. An Ethernet could contain multiple cable segments and multiple repeaters, but no two transceivers could be more than 2.5 km apart and no path between any two transceivers could traverse more than four repeaters. The reason for this restriction was so that the MAC protocol, which we will look at next, would work correctly.

4.3.2 Classic Ethernet MAC Sublayer Protocol

The format used to send frames is shown in Fig. 4-14. First comes a *Preamble* of 8 bytes, each containing the bit pattern 10101010 (with the exception of the last byte, in which the last 2 bits are set to 11). This last byte is called the *Start of Frame* delimiter for 802.3. The Manchester encoding of this pattern produces a 10-MHz square wave for 6.4 μsec to allow the receiver's clock to synchronize with the sender's. The last two 1 bits tell the receiver that the rest of the frame is about to start.

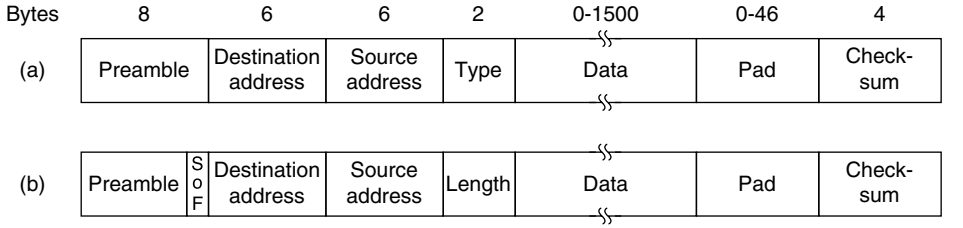

Figure 4-14. Frame formats. (a) Ethernet (DIX). (b) IEEE 802.3.

Next come two addresses, one for the destination and one for the source. They are each 6 bytes long. The first transmitted bit of the destination address is a 0 for

ordinary addresses and a 1 for group addresses. Group addresses allow multiple stations to listen to a single address. When a frame is sent to a group address, all the stations in the group receive it. Sending to a group of stations is called **multicasting**. The special address consisting of all 1 bits is reserved for **broadcasting**. A frame containing all 1s in the destination field is accepted by all stations on the network. Multicasting is more selective, but it involves group management to define which stations are in the group. Conversely, broadcasting does not differentiate between stations at all, so it does not require any group management.

An interesting feature of station source addresses is that they are globally unique, assigned centrally by IEEE to ensure that no two stations anywhere in the world have the same address. The idea is that any station can uniquely address any other station by just giving the right 48-bit number. To do this, the first 3 bytes of the address field are used for an **OUI** (**Organizationally Unique Identifier**). Values for this field are assigned by IEEE and indicate a manufacturer. Manufacturers are assigned blocks of 2^{24} addresses. The manufacturer assigns the last 3 bytes of the address and programs the complete address into the NIC before it is sold.

Next comes the *Type* or *Length* field, depending on whether the frame is Ethernet or IEEE 802.3. Ethernet uses a *Type* field to tell the receiver what to do with the frame. Multiple network-layer protocols may be in use at the same time on the same machine, so when an Ethernet frame arrives, the operating system has to know which one to hand the frame to. The *Type* field specifies which process to give the frame to. For example, a type code of 0x0800 means that the data contains an IPv4 packet.

IEEE 802.3, in its wisdom, decided that this field would carry the length of the frame, since the Ethernet length was determined by looking inside the data—a layering violation if ever there was one. Of course, this meant there was no way for the receiver to figure out what to do with an incoming frame. That problem was handled by the addition of another header for the **LLC** (**Logical Link Control**) protocol within the data. It uses 8 bytes to convey the 2 bytes of protocol type information.

Unfortunately, by the time 802.3 was published, so much hardware and software for DIX Ethernet was already in use that few manufacturers and users were enthusiastic about repackaging the *Type* and *Length* fields. In 1997, IEEE threw in the towel and said that both ways were fine with it. Fortunately, all the *Type* fields in use before 1997 had values greater than 1500, then well established as the maximum data size. Now the rule is that any number there less than or equal to 0x600 (1536) can be interpreted as *Length*, and any number greater than 0x600 can be interpreted as *Type*. Now IEEE can maintain that everyone is using its standard and everybody else can keep on doing what they were already doing (not bothering with LLC) without feeling guilty about it.

Next come the data, up to 1500 bytes. This limit was chosen somewhat arbitrarily at the time the Ethernet standard was cast in stone, mostly based on the fact

that a transceiver needs enough RAM to hold an entire frame and RAM was expensive in 1978. A larger upper limit would have meant more RAM, and hence a more expensive transceiver.

In addition to there being a maximum frame length, there is also a minimum frame length. While a data field of 0 bytes is sometimes useful, it causes a problem. When a transceiver detects a collision, it truncates the current frame, which means that stray bits and pieces of frames appear on the cable all the time. To make it easier to distinguish valid frames from garbage, Ethernet requires that valid frames must be at least 64 bytes long, from destination address to checksum, including both. If the data portion of a frame is less than 46 bytes, the *Pad* field is used to fill out the frame to the minimum size.

Another (and more important) reason for having a minimum length frame is to prevent a station from completing the transmission of a short frame before the first bit has even reached the far end of the cable, where it may collide with another frame. This problem is illustrated in Fig. 4-15. At time 0, station *A*, at one end of the network, sends off a frame. Let us call the propagation time for this frame to reach the other end τ. Just before the frame gets to the other end (i.e., at time $\tau - \varepsilon$), the most distant station, *B*, starts transmitting. When *B* detects that it is receiving more power than it is putting out, it knows that a collision has occurred, so it aborts its transmission and generates a 48-bit noise burst to warn all other stations. In other words, it jams the ether to make sure the sender does not miss the collision. At about time 2τ, the sender sees the noise burst and aborts its transmission, too. It then waits a random time before trying again.

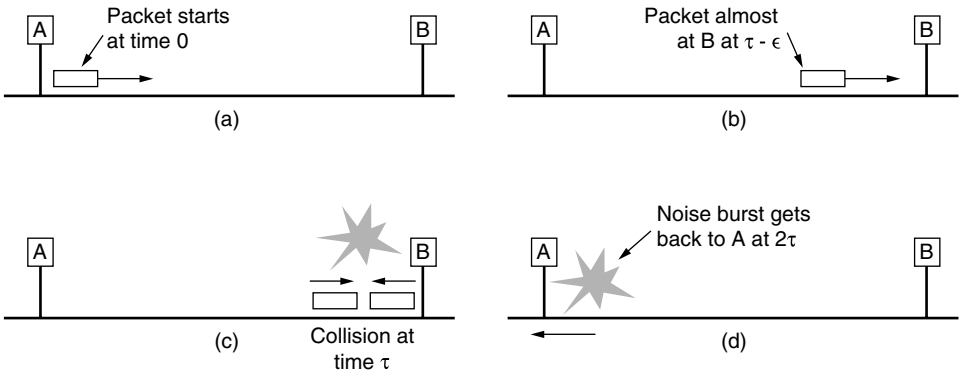

Figure 4-15. Collision detection can take as long as 2τ.

If a station tries to transmit a very short frame, it is conceivable that a collision will occur, but the transmission will have completed before the noise burst gets back to the station at 2τ. The sender will then incorrectly conclude that the frame was successfully sent. To prevent this situation from occurring, all frames must take more than 2τ to send so that the transmission is still taking place when

the noise burst gets back to the sender. For a 10-Mbps LAN with a maximum length of 2500 meters and four repeaters (from the 802.3 specification), the round-trip time (including time to propagate through the four repeaters) has been determined to be nearly 50 μsec in the worst case. Therefore, the shortest allowed frame must take at least this long to transmit. At 10 Mbps, a bit takes 100 nsec, so 500 bits is the smallest frame that is guaranteed to work. To add some margin of safety, this number was rounded up to 512 bits or 64 bytes.

The final field is the *Checksum*. It is a 32-bit CRC of the kind we studied in Sec. 3.2. In fact, it is defined exactly by the generator polynomial we gave there, which popped up for PPP, ADSL, and other links too. This CRC is an error-detecting code that is used to determine if the bits of the frame have been received correctly. It just does error detection, with the frame dropped if an error is detected.

CSMA/CD with Binary Exponential Backoff

Classic Ethernet uses the 1-persistent CSMA/CD algorithm that we studied in Sec. 4.2. This descriptor just means that stations sense the medium when they have a frame to send and send the frame as soon as the medium becomes idle. They monitor the channel for collisions as they send. If there is a collision, they abort the transmission with a short jam signal and retransmit after a random interval.

Let us now see how the random interval is determined when a collision occurs, as it is a new method. The model is still that of Fig. 4-5. After a collision, time is divided into discrete slots whose length is equal to the worst-case round-trip propagation time on the ether (2τ). To accommodate the longest path allowed by Ethernet, the slot time has been set to 512 bit times, or 51.2 μsec.

After the first collision, each station waits either 0 or 1 slot times at random before trying again. If two stations collide and each one picks the same random number, they will collide again. After the second collision, each one picks either 0, 1, 2, or 3 at random and waits that number of slot times. If a third collision occurs (the probability of this happening is 0.25), the next time the number of slots to wait is chosen at random from the interval 0 to $2^3 - 1$.

In general, after i collisions, a random number between 0 and $2^i - 1$ is chosen, and that number of slots is skipped. However, after 10 collisions have been reached, the randomization interval is frozen at a maximum of 1023 slots. After 16 collisions, the controller throws in the towel and reports failure back to the computer. Further recovery is up to higher layers.

This algorithm, called **binary exponential backoff**, was chosen to dynamically adapt to the number of stations trying to send. If the randomization interval for all collisions were 1023, the chance of two stations colliding for a second time would be negligible, but the average wait after a collision would be hundreds of slot times, introducing significant delay. On the other hand, if each station always

delayed for either 0 or 1 slots, then if 100 stations ever tried to send at once they would collide over and over until 99 of them picked 1 and the remaining station picked 0. This might take years. By having the randomization interval grow exponentially as more and more consecutive collisions occur, the algorithm ensures a low delay when only a few stations collide but also ensures that the collisions are resolved in a reasonable interval when many stations collide. Truncating the backoff at 1023 keeps the bound from growing too large.

If there is no collision, the sender assumes that the frame was probably successfully delivered. That is, neither CSMA/CD nor Ethernet provides acknowledgements. This choice is appropriate for wired and optical fiber channels that have low error rates. Any errors that do occur must then be detected by the CRC and recovered by higher layers. For wireless channels that have more errors, we will see that acknowledgements are used.

4.3.3 Ethernet Performance

Now let us briefly examine the performance of classic Ethernet under conditions of heavy and constant load, that is, with k stations always ready to transmit. A rigorous analysis of the binary exponential backoff algorithm is complicated. Instead, we will follow Metcalfe and Boggs (1976) and assume a constant retransmission probability in each slot. If each station transmits during a contention slot with probability p, the probability A that some station acquires the channel in that slot is

$$A = kp(1-p)^{k-1} \tag{4-5}$$

A is maximized when $p = 1/k$, with $A \to 1/e$ as $k \to \infty$. The probability that the contention interval has exactly j slots in it is $A(1-A)^{j-1}$, so the mean number of slots per contention is given by

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Since each slot has a duration 2τ, the mean contention interval, w, is $2\tau/A$. Assuming optimal p, the mean number of contention slots is never more than e, so w is at most $2\tau e \approx 5.4\tau$.

If the mean frame takes P sec to transmit, when many stations have frames to send,

$$\text{Channel efficiency} = \frac{P}{P + 2\tau/A} \tag{4-6}$$

Here we see where the maximum cable distance between any two stations enters into the performance figures. The longer the cable, the longer the contention interval, which is why the Ethernet standard specifies a maximum cable length.

It is instructive to formulate Eq. (4-6) in terms of the frame length, F, the network bandwidth, B, the cable length, L, and the speed of signal propagation, c, for the optimal case of e contention slots per frame. With $P = F/B$, Eq. (4-6) becomes

$$\text{Channel efficiency} = \frac{1}{1 + 2BLe/cF} \tag{4-7}$$

When the second term in the denominator is large, network efficiency will be low. More specifically, increasing network bandwidth or distance (the BL product) reduces efficiency for a given frame size. Unfortunately, much research on network hardware is aimed precisely at increasing this product. People want high bandwidth over long distances (fiber optic MANs, for example), yet classic Ethernet implemented in this manner is not the best system for these applications. We will see other ways of implementing Ethernet in the next section.

In Fig. 4-16, the channel efficiency is plotted versus the number of ready stations for $2\tau = 51.2$ μsec and a data rate of 10 Mbps, using Eq. (4-7). With a 64-byte slot time, it is not surprising that 64-byte frames are not efficient. On the other hand, with 1024-byte frames and an asymptotic value of e 64-byte slots per contention interval, the contention period is 174 bytes long and the efficiency is 85%. This result is much better than the 37% efficiency of slotted ALOHA.

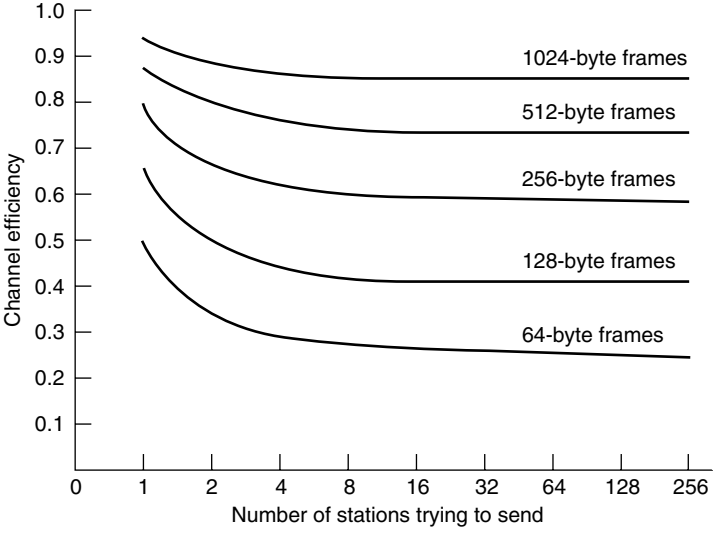

Figure 4-16. Efficiency of Ethernet at 10 Mbps with 512-bit slot times.

It is probably worth mentioning that there has been a large amount of theoretical performance analysis of Ethernet (and other networks). Most of the results should be taken with a grain (or better yet, a metric ton) of salt, for two reasons.

First, virtually all of the theoretical work assumes Poisson traffic. As researchers have begun looking at real data, it now appears that network traffic is rarely Poisson. Instead, it is self-similar or bursty over a range of time scales (Paxson and Floyd, 1995; and Leland et al., 1994). What this means is that averaging over long periods of time does not smooth out the traffic. As well as using questionable models, many of the analyses focus on the ''interesting'' performance cases of abnormally high load. Boggs et al. (1988) showed by experimentation that Ethernet works well in reality, even at moderately high load.

4.3.4 Switched Ethernet

Ethernet soon began to evolve away from the single long cable architecture of classic Ethernet. The problems associated with finding breaks or loose connections drove it toward a different kind of wiring pattern, in which each station has a dedicated cable running to a central **hub**. A hub simply connects all the attached wires electrically, as if they were soldered together. This configuration is shown in Fig. 4-17(a).

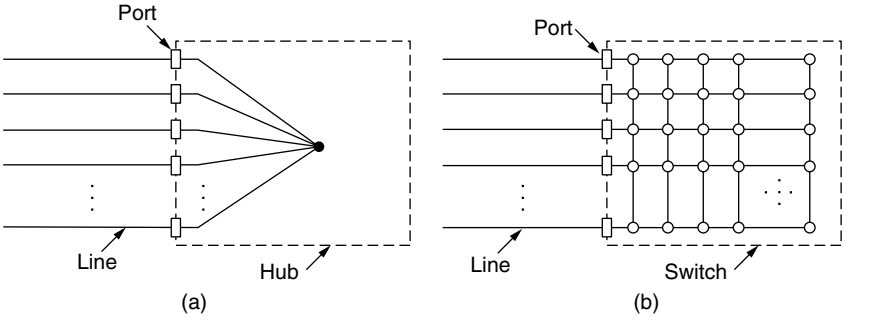

Figure 4-17. (a) Hub. (b) Switch.

The wires were telephone company twisted pairs, since most office buildings were already wired this way and normally plenty of spares were available. This reuse was a win, but it did reduce the maximum cable run from the hub to 100 meters (200 meters if high quality Category 5 twisted pairs were used). Adding or removing a station is simpler in this configuration, and cable breaks can be detected easily. With the advantages of being able to use existing wiring and ease of maintenance, twisted-pair hubs quickly became the dominant form of Ethernet.

However, hubs do not increase capacity because they are logically equivalent to the single long cable of classic Ethernet. As more and more stations are added, each station gets a decreasing share of the fixed capacity. Eventually, the LAN will saturate. One way out is to go to a higher speed, say, from 10 Mbps to 100 Mbps, 1 Gbps, or even higher speeds. But with the growth of multimedia and powerful servers, even a 1-Gbps Ethernet can become saturated.

Fortunately, there is an another way to deal with increased load: switched Ethernet. The heart of this system is a **switch** containing a high-speed backplane that connects all of the ports, as shown in Fig. 4-17(b). From the outside, a switch looks just like a hub. They are both boxes, typically with 4 to 48 ports, each with a standard RJ-45 connector for a twisted-pair cable. Each cable connects the switch or hub to a single computer, as shown in Fig. 4-18. A switch has the same advantages as a hub, too. It is easy to add or remove a new station by plugging or unplugging a wire, and it is easy to find most faults since a flaky cable or port will usually affect just one station. There is still a shared component that can fail—the switch itself—but if all stations lose connectivity the IT folks know what to do to fix the problem: replace the whole switch.

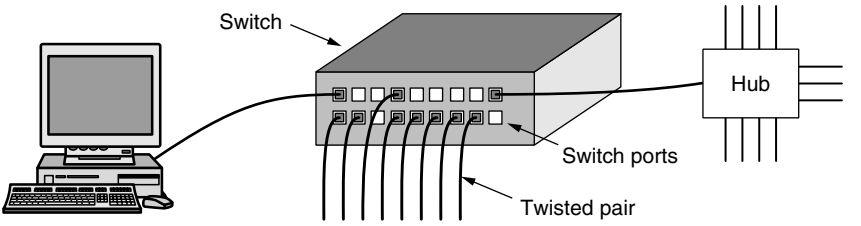

Figure 4-18. An Ethernet switch.

Inside the switch, however, something very different is happening. Switches only output frames to the ports for which those frames are destined. When a switch port receives an Ethernet frame from a station, the switch checks the Ethernet addresses to see which port the frame is destined for. This step requires the switch to be able to work out which ports correspond to which addresses, a process that we will describe in Sec. 4.8 when we get to the general case of switches connected to other switches. For now, just assume that the switch knows the frame's destination port. The switch then forwards the frame over its high-speed backplane to the destination port. The backplane typically runs at many Gbps, using a proprietary protocol that does not need to be standardized because it is entirely hidden inside the switch. The destination port then transmits the frame on the wire so that it reaches the intended station. None of the other ports even knows the frame exists.

What happens if more than one of the stations or ports wants to send a frame at the same time? Again, switches differ from hubs. In a hub, all stations are in the same **collision domain**. They must use the CSMA/CD algorithm to schedule their transmissions. In a switch, each port is its own independent collision domain. In the common case that the cable is full duplex, both the station and the port can send a frame on the cable at the same time, without worrying about other ports and stations. Collisions are now impossible and CSMA/CD is not needed. However, if the cable is half duplex, the station and the port must contend for transmission with CSMA/CD in the usual way.

A switch improves performance over a hub in two ways. First, since there are no collisions, the capacity is used more efficiently. Second, and more importantly, with a switch multiple frames can be sent simultaneously (by different stations). These frames will reach the switch ports and travel over the switch's backplane to be output on the proper ports. However, since two frames might be sent to the same output port at the same time, the switch must have buffering so that it can temporarily queue an input frame until it can be transmitted to the output port. Overall, these improvements give a large performance win that is not possible with a hub. The total system throughput can often be increased by an order of magnitude, depending on the number of ports and traffic patterns.

The change in the ports on which frames are output also has security benefits. Most LAN interfaces have a **promiscuous mode**, in which *all* frames are given to each computer, not just those addressed to it. With a hub, every computer that is attached can see the traffic sent between all of the other computers. Spies and busybodies love this feature. With a switch, traffic is forwarded only to the ports where it is destined. This restriction provides better isolation so that traffic will not easily escape and fall into the wrong hands. However, it is better to encrypt traffic if security is really needed.

Because the switch just expects standard Ethernet frames on each input port, it is possible to use some of the ports as concentrators. In Fig. 4-18, the port in the upper-right corner is connected not to a single station, but to a 12-port hub instead. As frames arrive at the hub, they contend for the ether in the usual way, including collisions and binary backoff. Successful frames make it through the hub to the switch and are treated there like any other incoming frames. The switch does not know they had to fight their way in. Once in the switch, they are sent to the correct output line over the high-speed backplane. It is also possible that the correct destination was one on the lines attached to the hub, in which case the frame has already been delivered so the switch just drops it. Hubs are simpler and cheaper than switches, but due to falling switch prices they have become an endangered species. Modern networks largely use switched Ethernet. Nevertheless, legacy hubs still exist.

4.3.5 Fast Ethernet

At the same time that switches were becoming popular, the speed of 10-Mbps Ethernet was coming under pressure. At first, 10 Mbps seemed like heaven, just as cable modems seemed like heaven to the users of telephone modems. But the novelty wore off quickly. As a kind of corollary to Parkinson's Law (''Work expands to fill the time available for its completion''), it seemed that data expanded to fill the bandwidth available for their transmission.

Many installations needed more bandwidth and thus had numerous 10-Mbps LANs connected by a maze of repeaters, hubs, and switches, although to the network managers it sometimes felt that they were being held together by bubble

gum and chicken wire. But even with Ethernet switches, the maximum bandwidth of a single computer was limited by the cable that connected it to the switch port.

It was in this environment that IEEE reconvened the 802.3 committee in 1992 with instructions to come up with a faster LAN. One proposal was to keep 802.3 exactly as it was, but just make it go faster. Another proposal was to redo it totally and give it lots of new features, such as real-time traffic and digitized voice, but just keep the old name (for marketing reasons). After some wrangling, the committee decided to keep 802.3 the way it was, and just make it go faster. This strategy would get the job done before the technology changed and avoid unforeseen problems with a brand new design. The new design would also be backward-compatible with existing Ethernet LANs. The people behind the losing proposal did what any self-respecting computer-industry people would have done under these circumstances: they stomped off and formed their own committee and standardized their LAN anyway (eventually as 802.12). It flopped miserably.

The work was done quickly (by standards committees' norms), and the result, 802.3u, was approved by IEEE in June 1995. Technically, 802.3u is not a new standard, but an addendum to the existing 802.3 standard (to emphasize its backward compatibility). This strategy is used a lot. Since practically everyone calls it **fast Ethernet**, rather than 802.3u, we will do that, too.

The basic idea behind fast Ethernet was simple: keep all the old frame formats, interfaces, and procedural rules, but reduce the bit time from 100 nsec to 10 nsec. Technically, it would have been possible to copy 10-Mbps classic Ethernet and still detect collisions on time by just reducing the maximum cable length by a factor of 10. However, the advantages of twisted-pair wiring were so overwhelming that fast Ethernet is based entirely on this design. Thus, all fast Ethernet systems use hubs and switches; multidrop cables with vampire taps or BNC connectors are not permitted.

Nevertheless, some choices still had to be made, the most important being which wire types to support. One contender was Category 3 twisted pair. The argument for it was that practically every office in the Western world had at least four Category 3 (or better) twisted pairs running from it to a telephone wiring closet within 100 meters. Sometimes two such cables existed. Thus, using Category 3 twisted pair would make it possible to wire up desktop computers using fast Ethernet without having to rewire the building, an enormous advantage for many organizations.

The main disadvantage of a Category 3 twisted pair is its inability to carry 100 Mbps over 100 meters, the maximum computer-to-hub distance specified for 10-Mbps hubs. In contrast, Category 5 twisted pair wiring can handle 100 m easily, and fiber can go much farther. The compromise chosen was to allow all three possibilities, as shown in Fig. 4-19, but to pep up the Category 3 solution to give it the additional carrying capacity needed.

The Category 3 UTP scheme, called **100Base-T4**, used a signaling speed of 25 MHz, only 25% faster than standard Ethernet's 20 MHz. (Remember that

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps (Cat 5 UTP)
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

Figure 4-19. The original fast Ethernet cabling.

Manchester encoding, discussed in Sec. 2.5, requires two clock periods for each of the 10 million bits sent each second.) However, to achieve the necessary bit rate, 100Base-T4 requires four twisted pairs. Of the four pairs, one is always to the hub, one is always from the hub, and the other two are switchable to the current transmission direction. To get 100 Mbps out of the three twisted pairs in the transmission direction, a fairly involved scheme is used on each twisted pair. It involves sending ternary digits with three different voltage levels. This scheme is not likely to win any prizes for elegance, and we will skip the details. However, since standard telephone wiring for decades has had four twisted pairs per cable, most offices are able to use the existing wiring plant. Of course, it means giving up your office telephone, but that is surely a small price to pay for faster email.

100Base-T4 fell by the wayside as many office buildings were rewired with Category 5 UTP for **100Base-TX** Ethernet, which came to dominate the market. This design is simpler because the wires can handle clock rates of 125 MHz. Only two twisted pairs per station are used, one to the hub and one from it. Neither straight binary coding (i.e., NRZ) nor Manchester coding is used. Instead, the **4B/5B** encoding we described in Sec 2.5 is used. 4 data bits are encoded as 5 signal bits and sent at 125 MHz to provide 100 Mbps. This scheme is simple but has sufficient transitions for synchronization and uses the bandwidth of the wire relatively well. The 100Base-TX system is full duplex; stations can transmit at 100 Mbps on one twisted pair and receive at 100 Mbps on another twisted pair at the same time.

The last option, **100Base-FX**, uses two strands of multimode fiber, one for each direction, so it, too, can run full duplex with 100 Mbps in each direction. In this setup, the distance between a station and the switch can be up to 2 km.

Fast Ethernet allows interconnection by either hubs or switches. To ensure that the CSMA/CD algorithm continues to work, the relationship between the minimum frame size and maximum cable length must be maintained as the network speed goes up from 10 Mbps to 100 Mbps. So, either the minimum frame size of 64 bytes must go up or the maximum cable length of 2500 m must come down, proportionally. The easy choice was for the maximum distance between any two stations to come down by a factor of 10, since a hub with 100-m cables falls within this new maximum already. However, 2-km 100Base-FX cables are

too long to permit a 100-Mbps hub with the normal Ethernet collision algorithm. These cables must instead be connected to a switch and operate in a full-duplex mode so that there are no collisions.

Users quickly started to deploy fast Ethernet, but they were not about to throw away 10-Mbps Ethernet cards on older computers. As a consequence, virtually all fast Ethernet switches can handle a mix of 10-Mbps and 100-Mbps stations. To make upgrading easy, the standard itself provides a mechanism called **auto-negotiation** that lets two stations automatically negotiate the optimum speed (10 or 100 Mbps) and duplexity (half or full). It works well most of the time but is known to lead to duplex mismatch problems when one end of the link autonegotiates but the other end does not and is set to full-duplex mode (Shalunov and Carlson, 2005). Most Ethernet products use this feature to configure themselves.

4.3.6 Gigabit Ethernet

The ink was barely dry on the fast Ethernet standard when the 802 committee began working on a yet faster Ethernet, quickly dubbed **gigabit Ethernet**. IEEE ratified the most popular form as 802.3ab in 1999. Below we will discuss some of the key features of gigabit Ethernet. More information is given by Spurgeon (2000).

The committee's goals for gigabit Ethernet were essentially the same as the committee's goals for fast Ethernet: increase performance tenfold while maintaining compatibility with all existing Ethernet standards. In particular, gigabit Ethernet had to offer unacknowledged datagram service with both unicast and broadcast, use the same 48-bit addressing scheme already in use, and maintain the same frame format, including the minimum and maximum frame sizes. The final standard met all these goals.

Like fast Ethernet, all configurations of gigabit Ethernet use point-to-point links. In the simplest configuration, illustrated in Fig. 4-20(a), two computers are directly connected to each other. The more common case, however, uses a switch or a hub connected to multiple computers and possibly additional switches or hubs, as shown in Fig. 4-20(b). In both configurations, each individual Ethernet cable has exactly two devices on it, no more and no fewer.

Also like fast Ethernet, gigabit Ethernet supports two different modes of operation: full-duplex mode and half-duplex mode. The "normal" mode is full-duplex mode, which allows traffic in both directions at the same time. This mode is used when there is a central switch connected to computers (or other switches) on the periphery. In this configuration, all lines are buffered so each computer and switch is free to send frames whenever it wants to. The sender does not have to sense the channel to see if anybody else is using it because contention is impossible. On the line between a computer and a switch, the computer is the only possible sender to the switch, and the transmission will succeed even if the switch is currently sending a frame to the computer (because the line is full duplex). Since

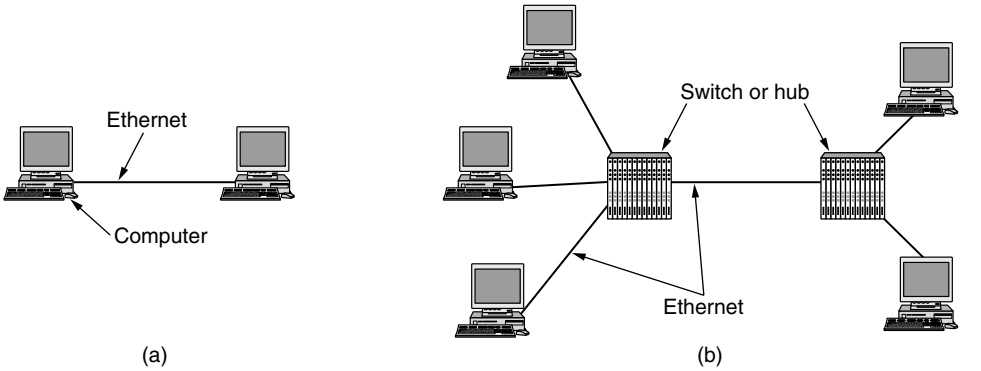

Figure 4-20. (a) A two-station Ethernet. (b) A multistation Ethernet.

no contention is possible, the CSMA/CD protocol is not used, so the maximum length of the cable is determined by signal strength issues rather than by how long it takes for a noise burst to propagate back to the sender in the worst case. Switch\%es are free to mix and match speeds. Autonegotiation is supported just as in fast Ethernet, only now the choice is among 10, 100, and 1000 Mbps.

The other mode of operation, half-duplex, is used when the computers are connected to a hub rather than a switch. A hub does not buffer incoming frames. Instead, it electrically connects all the lines internally, simulating the multidrop cable used in classic Ethernet. In this mode, collisions are possible, so the standard CSMA/CD protocol is required. Because a 64-byte frame (the shortest allowed) can now be transmitted 100 times faster than in classic Ethernet, the maximum cable length must be 100 times less, or 25 meters, to maintain the essential property that the sender is still transmitting when the noise burst gets back to it, even in the worst case. With a 2500-meter-long cable, the sender of a 64-byte frame at 1 Gbps would be long finished before the frame got even a tenth of the way to the other end, let alone to the end and back.

This length restriction was painful enough that two features were added to the standard to increase the maximum cable length to 200 meters, which is probably enough for most offices. The first feature, called **carrier extension**, essentially tells the hardware to add its own padding after the normal frame to extend the frame to 512 bytes. Since this padding is added by the sending hardware and removed by the receiving hardware, the software is unaware of it, meaning that no changes are needed to existing software. The downside is that using 512 bytes worth of bandwidth to transmit 46 bytes of user data (the payload of a 64-byte frame) has a line efficiency of only 9%.

The second feature, called **frame bursting**, allows a sender to transmit a concatenated sequence of multiple frames in a single transmission. If the total burst is less than 512 bytes, the hardware pads it again. If enough frames are waiting for transmission, this scheme is very efficient and preferred over carrier extension.

In all fairness, it is hard to imagine an organization buying modern computers with gigabit Ethernet cards and then connecting them with an old-fashioned hub to simulate classic Ethernet with all its collisions. Gigabit Ethernet interfaces and switches used to be expensive, but their prices fell rapidly as sales volumes picked up. Still, backward compatibility is sacred in the computer industry, so the committee was required to put it in. Today, most computers ship with an Ethernet interface that is capable of 10-, 100-, and 1000-Mbps operation and compatible with all of them.

Gigabit Ethernet supports both copper and fiber cabling, as listed in Fig. 4-21. Signaling at or near 1 Gbps requires encoding and sending a bit every nanosecond. This trick was initially accomplished with short, shielded copper cables (the 1000Base-CX version) and optical fibers. For the optical fibers, two wavelengths are permitted and result in two different versions: 0.85 microns (short, for 1000Base-SX) and 1.3 microns (long, for 1000Base-LX).

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Figure 4-21. Gigabit Ethernet cabling.

Signaling at the short wavelength can be achieved with cheaper LEDs. It is used with multimode fiber and is useful for connections within a building, as it can run up to 500 m for 50-micron fiber. Signaling at the long wavelength requires more expensive lasers. On the other hand, when combined with single-mode (10-micron) fiber, the cable length can be up to 5 km. This limit allows long distance connections between buildings, such as for a campus backbone, as a dedicated point-to-point link. Later variations of the standard allowed even longer links over single-mode fiber.

To send bits over these versions of gigabit Ethernet, the **8B/10B** encoding we described in Sec. 2.5 was borrowed from another networking technology called Fibre Channel. That scheme encodes 8 bits of data into 10-bit codewords that are sent over the wire or fiber, hence the name 8B/10B. The codewords were chosen so that they could be balanced (i.e., have the same number of 0s and 1s) with sufficient transitions for clock recovery. Sending the coded bits with NRZ requires a signaling bandwidth of 25% more than that required for the uncoded bits, a big improvement over the 100% expansion of Manchester coding.

However, all of these options required new copper or fiber cables to support the faster signaling. None of them made use of the large amount of Category 5 UTP that had been installed along with fast Ethernet. Within a year, 1000Base-T

came along to fill this gap, and it has been the most popular form of gigabit Ethernet ever since. People apparently dislike rewiring their buildings.

More complicated signaling is needed to make Ethernet run at 1000 Mbps over Category 5 wires. To start, all four twisted pairs in the cable are used, and each pair is used in both directions at the same time by using digital signal processing to separate signals. Over each wire, five voltage levels that carry 2 bits are used for signaling at 125 Msymbols/sec. The mapping to produce the symbols from the bits is not straightforward. It involves scrambling, for transitions, followed by an error correcting code in which four values are embedded into five signal levels.

A speed of 1 Gbps is quite fast. For example, if a receiver is busy with some other task for even 1 msec and does not empty the input buffer on some line, up to 1953 frames may have accumulated in that gap. Also, when a computer on a gigabit Ethernet is shipping data down the line to a computer on a classic Ethernet, buffer overruns are very likely. As a consequence of these two observations, gigabit Ethernet supports flow control. The mechanism consists of one end sending a special control frame to the other end telling it to pause for some period of time. These PAUSE control frames are normal Ethernet frames containing a type of 0x8808. Pauses are given in units of the minimum frame time. For gigabit Ethernet, the time unit is 512 nsec, allowing for pauses as long as 33.6 msec.

There is one more extension that was introduced along with gigabit Ethernet. **Jumbo frames** allow for frames to be longer than 1500 bytes, usually up to 9 KB. This extension is proprietary. It is not recognized by the standard because if it is used then Ethernet is no longer compatible with earlier versions, but most vendors support it anyway. The rationale is that 1500 bytes is a short unit at gigabit speeds. By manipulating larger blocks of information, the frame rate can be decreased, along with the processing associated with it, such as interrupting the processor to say that a frame has arrived, or splitting up and recombining messages that were too long to fit in one Ethernet frame.

4.3.7 10-Gigabit Ethernet

As soon as gigabit Ethernet was standardized, the 802 committee got bored and wanted to get back to work. IEEE told them to start on 10-gigabit Ethernet. This work followed much the same pattern as the previous Ethernet standards, with standards for fiber and shielded copper cable appearing first in 2002 and 2004, followed by the standard for copper twisted pair in 2006.

10 Gbps is a truly prodigious speed, 1000x faster than the original Ethernet. Where could it be needed? The answer is inside data centers and exchanges to connect high-end routers, switches, and servers, as well as in long-distance, high bandwidth trunks between offices that are enabling entire metropolitan area networks based on Ethernet and fiber. The long distance connections use optical fiber, while the short connections may use copper or fiber.

All versions of 10-gigabit Ethernet support only full-duplex operation. CSMA/CD is no longer part of the design, and the standards concentrate on the details of physical layers that can run at very high speed. Compatibility still matters, though, so 10-gigabit Ethernet interfaces autonegotiate and fall back to the highest speed supported by both ends of the line.

The main kinds of 10-gigabit Ethernet are listed in Fig. 4-22. Multimode fiber with the 0.85μ (short) wavelength is used for medium distances, and single-mode fiber at 1.3μ (long) and 1.5μ (extended) is used for long distances. 10GBase-ER can run for distances of 40 km, making it suitable for wide area applications. All of these versions send a serial stream of information that is produced by scrambling the data bits, then encoding them with a **64B/66B** code. This encoding has less overhead than an 8B/10B code.

Name	Cable	Max. segment	Advantages
10GBase-SR	Fiber optics	Up to 300 m	Multimode fiber (0.85μ)
10GBase-LR	Fiber optics	10 km	Single-mode fiber (1.3μ)
10GBase-ER	Fiber optics	40 km	Single-mode fiber (1.5μ)
10GBase-CX4	4 Pairs of twinax	15 m	Twinaxial copper
10GBase-T	4 Pairs of UTP	100 m	Category 6a UTP

Figure 4-22. 10-Gigabit Ethernet cabling.

The first copper version defined, 10GBase-CX4, uses a cable with four pairs of twinaxial copper wiring. Each pair uses 8B/10B coding and runs at 3.125 Gsymbols/second to reach 10 Gbps. This version is cheaper than fiber and was early to market, but it remains to be seen whether it will be beat out in the long run by 10-gigabit Ethernet over more garden variety twisted pair wiring.

10GBase-T is the version that uses UTP cables. While it calls for Category 6a wiring, for shorter runs, it can use lower categories (including Category 5) to allow some reuse of installed cabling. Not surprisingly, the physical layer is quite involved to reach 10 Gbps over twisted pair. We will only sketch some of the high-level details. Each of the four twisted pairs is used to send 2500 Mbps in both directions. This speed is reached using a signaling rate of 800 Msymbols/sec with symbols that use 16 voltage levels. The symbols are produced by scrambling the data, protecting it with a LDPC (Low Density Parity Check) code, and further coding for error correction.

10-gigabit Ethernet is still shaking out in the market, but the 802.3 committee has already moved on. At the end of 2007, IEEE created a group to standardize Ethernet operating at 40 Gbps and 100 Gbps. This upgrade will let Ethernet compete in very high-performance settings, including long-distance connections in backbone networks and short connections over the equipment backplanes. The standard is not yet complete, but proprietary products are already available.

4.3.8 Retrospective on Ethernet

Ethernet has been around for over 30 years and has no serious competitors in sight, so it is likely to be around for many years to come. Few CPU architectures, operating systems, or programming languages have been king of the mountain for three decades going on strong. Clearly, Ethernet did something right. What?

Probably the main reason for its longevity is that Ethernet is simple and flexible. In practice, simple translates into reliable, cheap, and easy to maintain. Once the hub and switch architecture was adopted, failures became extremely rare. People hesitate to replace something that works perfectly all the time, especially when they know that an awful lot of things in the computer industry work very poorly, so that many so-called "upgrades" are worse than what they replaced.

Simple also translates into cheap. Twisted-pair wiring is relatively inexpensive as are the hardware components. They may start out expensive when there is a transition, for example, new gigabit Ethernet NICs or switches, but they are merely additions to a well established network (not a replacement of it) and the prices fall quickly as the sales volume picks up.

Ethernet is easy to maintain. There is no software to install (other than the drivers) and not much in the way of configuration tables to manage (and get wrong). Also, adding new hosts is as simple as just plugging them in.

Another point is that Ethernet interworks easily with TCP/IP, which has become dominant. IP is a connectionless protocol, so it fits perfectly with Ethernet, which is also connectionless. IP fits much less well with connection-oriented alternatives such as ATM. This mismatch definitely hurt ATM's chances.

Lastly, and perhaps most importantly, Ethernet has been able to evolve in certain crucial ways. Speeds have gone up by several orders of magnitude and hubs and switches have been introduced, but these changes have not required changing the software and have often allowed the existing cabling to be reused for a time. When a network salesman shows up at a large installation and says "I have this fantastic new network for you. All you have to do is throw out all your hardware and rewrite all your software," he has a problem.

Many alternative technologies that you have probably not even heard of were faster than Ethernet when they were introduced. As well as ATM, this list includes FDDI (Fiber Distributed Data Interface) and Fibre Channel,† two ring-based optical LANs. Both were incompatible with Ethernet. Neither one made it. They were too complicated, which led to complex chips and high prices. The lesson that should have been learned here was KISS (Keep It Simple, Stupid). Eventually, Ethernet caught up with them in terms of speed, often by borrowing some of their technology, for example, the 4B/5B coding from FDDI and the 8B/10B coding from Fibre Channel. Then they had no advantages left and quietly died off or fell into specialized roles.

† It is called "Fibre Channel" and not "Fiber Channel" because the document editor was British.

It looks like Ethernet will continue to expand in its applications for some time. 10-gigabit Ethernet has freed it from the distance constraints of CSMA/CD. Much effort is being put into **carrier-grade Ethernet** to let network providers offer Ethernet-based services to their customers for metropolitan and wide area networks (Fouli and Maler, 2009). This application carries Ethernet frames long distances over fiber and calls for better management features to help operators offer reliable, high-quality services. Very high speed networks are also finding uses in backplanes connecting components in large routers or servers. Both of these uses are in addition to that of sending frames between computers in offices.

4.4 WIRELESS LANS

Wireless LANs are increasingly popular, and homes, offices, cafes, libraries, airports, zoos, and other public places are being outfitted with them to connect computers, PDAs, and smart phones to the Internet. Wireless LANs can also be used to let two or more nearby computers communicate without using the Internet.

The main wireless LAN standard is 802.11. We gave some background information on it in Sec. 1.5.3. Now it is time to take a closer look at the technology. In the following sections, we will look at the protocol stack, physical-layer radio transmission techniques, the MAC sublayer protocol, the frame structure, and the services provided. For more information about 802.11, see Gast (2005). To get the truth from the mouth of the horse, consult the published standard, IEEE 802.11-2007 itself.

4.4.1 The 802.11 Architecture and Protocol Stack

802.11 networks can be used in two modes. The most popular mode is to connect clients, such as laptops and smart phones, to another network, such as a company intranet or the Internet. This mode is shown in Fig. 4-23(a). In infrastructure mode, each client is associated with an **AP** (**Access Point**) that is in turn connected to the other network. The client sends and receives its packets via the AP. Several access points may be connected together, typically by a wired network called a **distribution system**, to form an extended 802.11 network. In this case, clients can send frames to other clients via their APs.

The other mode, shown in Fig. 4-23(b), is an **ad hoc network**. This mode is a collection of computers that are associated so that they can directly send frames to each other. There is no access point. Since Internet access is the killer application for wireless, ad hoc networks are not very popular.

Now we will look at the protocols. All the 802 protocols, including 802.11 and Ethernet, have a certain commonality of structure. A partial view of the 802.11 protocol stack is given in Fig. 4-24. The stack is the same for clients and

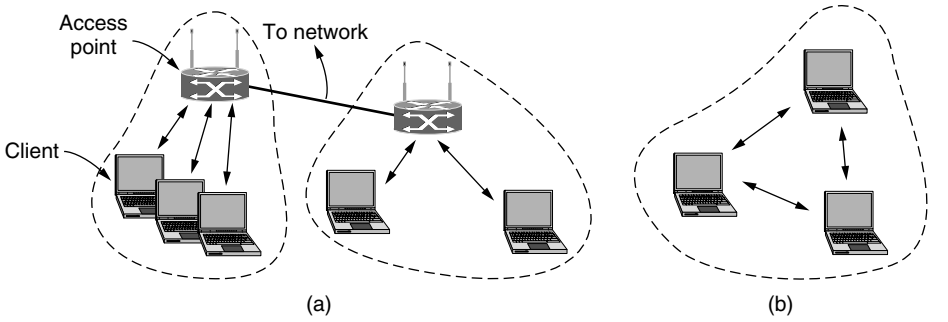

Figure 4-23. 802.11 architecture. (a) Infrastructure mode. (b) Ad-hoc mode.

APs. The physical layer corresponds fairly well to the OSI physical layer, but the data link layer in all the 802 protocols is split into two or more sublayers. In 802.11, the MAC (Medium Access Control) sublayer determines how the channel is allocated, that is, who gets to transmit next. Above it is the LLC (Logical Link Control) sublayer, whose job it is to hide the differences between the different 802 variants and make them indistinguishable as far as the network layer is concerned. This could have been a significant responsibility, but these days the LLC is a glue layer that identifies the protocol (e.g., IP) that is carried within an 802.11 frame.

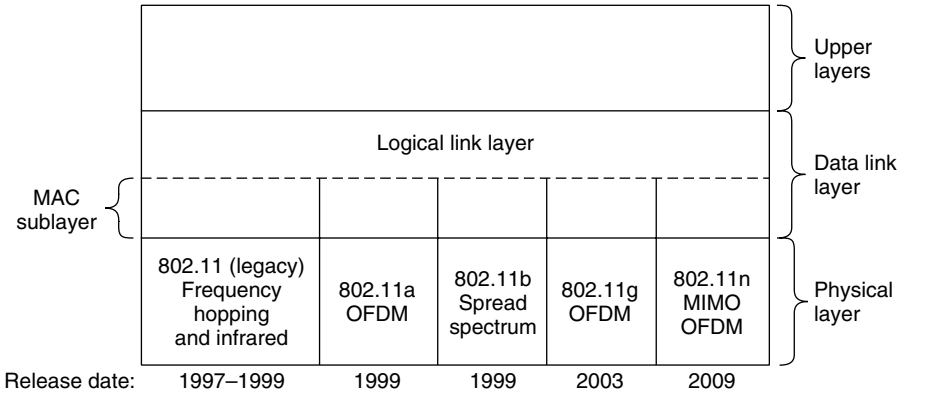

Figure 4-24. Part of the 802.11 protocol stack.

Several transmission techniques have been added to the physical layer as 802.11 has evolved since it first appeared in 1997. Two of the initial techniques, infrared in the manner of television remote controls and frequency hopping in the 2.4-GHz band, are now defunct. The third initial technique, direct sequence spread spectrum at 1 or 2 Mbps in the 2.4-GHz band, was extended to run at rates up to 11 Mbps and quickly became a hit. It is now known as 802.11b.

To give wireless junkies a much-wanted speed boost, new transmission techniques based on the OFDM (Orthogonal Frequency Division Multiplexing) scheme we described in Sec. 2.5.3 were introduced in 1999 and 2003. The first is called 802.11a and uses a different frequency band, 5 GHz. The second stuck with 2.4 GHz and compatibility. It is called 802.11g. Both give rates up to 54 Mbps.

Most recently, transmission techniques that simultaneously use multiple antennas at the transmitter and receiver for a speed boost were finalized as 802.11n in Oct. 2009. With four antennas and wider channels, the 802.11 standard now defines rates up to a startling 600 Mbps.

We will now examine each of these transmission techniques briefly. We will only cover those that are in use, however, skipping the legacy 802.11 transmission methods. Technically, these belong to the physical layer and should have been examined in Chap. 2, but since they are so closely tied to LANs in general and the 802.11 LAN in particular, we treat them here instead.

4.4.2 The 802.11 Physical Layer

Each of the transmission techniques makes it possible to send a MAC frame over the air from one station to another. They differ, however, in the technology used and speeds achievable. A detailed discussion of these technologies is far beyond the scope of this book, but a few words on each one will relate the techniques to the material we covered in Sec. 2.5 and will provide interested readers with the key terms to search for elsewhere for more information.

All of the 802.11 techniques use short-range radios to transmit signals in either the 2.4-GHz or the 5-GHz ISM frequency bands, both described in Sec. 2.3.3. These bands have the advantage of being unlicensed and hence freely available to any transmitter willing to meet some restrictions, such as radiated power of at most 1 W (though 50 mW is more typical for wireless LAN radios). Unfortunately, this fact is also known to the manufacturers of garage door openers, cordless phones, microwave ovens, and countless other devices, all of which compete with laptops for the same spectrum. The 2.4-GHz band tends to be more crowded than the 5-GHz band, so 5 GHz can be better for some applications even though it has shorter range due to the higher frequency.

All of the transmission methods also define multiple rates. The idea is that different rates can be used depending on the current conditions. If the wireless signal is weak, a low rate can be used. If the signal is clear, the highest rate can be used. This adjustment is called **rate adaptation**. Since the rates vary by a factor of 10 or more, good rate adaptation is important for good performance. Of course, since it is not needed for interoperability, the standards do not say how rate adaptation should be done.

The first transmission method we shall look at is **802.11b**. It is a spread-spectrum method that supports rates of 1, 2, 5.5, and 11 Mbps, though in practice the operating rate is nearly always 11 Mbps. It is similar to the CDMA system we

examined in Sec. 2.5, except that there is only one spreading code that is shared by all users. Spreading is used to satisfy the FCC requirement that power be spread over the ISM band. The spreading sequence used by 201.11b is a **Barker sequence**. It has the property that its autocorrelation is low except when the sequences are aligned. This property allows a receiver to lock onto the start of a transmission. To send at a rate of 1 Mbps, the Barker sequence is used with BPSK modulation to send 1 bit per 11 chips. The chips are transmitted at a rate of 11 Mchips/sec. To send at 2 Mbps, it is used with QPSK modulation to send 2 bits per 11 chips. The higher rates are different. These rates use a technique called **CCK** (**Complementary Code Keying**) to construct codes instead of the Barker sequence. The 5.5-Mbps rate sends 4 bits in every 8-chip code, and the 11-Mbps rate sends 8 bits in every 8-chip code.

Next we come to **802.11a**, which supports rates up to 54 Mbps in the 5-GHz ISM band. You might have expected that 802.11a to come before 802.11b, but that was not the case. Although the 802.11a group was set up first, the 802.11b standard was approved first and its product got to market well ahead of the 802.11a products, partly because of the difficulty of operating in the higher 5-GHz band.

The 802.11a method is based on **OFDM** (**Orthogonal Frequency Division Multiplexing**) because OFDM uses the spectrum efficiently and resists wireless signal degradations such as multipath. Bits are sent over 52 subcarriers in parallel, 48 carrying data and 4 used for synchronization. Each symbol lasts 4μs and sends 1, 2, 4, or 6 bits. The bits are coded for error correction with a binary convolutional code first, so only 1/2, 2/3, or 3/4 of the bits are not redundant. With different combinations, 802.11a can run at eight different rates, ranging from 6 to 54 Mbps. These rates are significantly faster than 802.11b rates, and there is less interference in the 5-GHz band. However, 802.11b has a range that is about seven times greater than that of 802.11a, which is more important in many situations.

Even with the greater range, the 802.11b people had no intention of letting this upstart win the speed championship. Fortunately, in May 2002, the FCC dropped its long-standing rule requiring all wireless communications equipment operating in the ISM bands in the U.S. to use spread spectrum, so it got to work on **802.11g**, which was approved by IEEE in 2003. It copies the OFDM modulation methods of 802.11a but operates in the narrow 2.4-GHz ISM band along with 802.11b. It offers the same rates as 802.11a (6 to 54 Mbps) plus of course compatibility with any 802.11b devices that happen to be nearby. All of these different choices can be confusing for customers, so it is common for products to support 802.11a/b/g in a single NIC.

Not content to stop there, the IEEE committee began work on a high-throughput physical layer called **802.11n**. It was ratified in 2009. The goal for 802.11n was throughput of at least 100 Mbps after all the wireless overheads were removed. This goal called for a raw speed increase of at least a factor of four. To make it happen, the committee doubled the channels from 20 MHz to 40 MHz and

reduced framing overheads by allowing a group of frames to be sent together. More significantly, however, 802.11n uses up to four antennas to transmit up to four streams of information at the same time. The signals of the streams interfere at the receiver, but they can be separated using **MIMO** (**Multiple Input Multiple Output**) communications techniques. The use of multiple antennas gives a large speed boost, or better range and reliability instead. MIMO, like OFDM, is one of those clever communications ideas that is changing wireless designs and which we are all likely to hear a lot about in the future. For a brief introduction to multiple antennas in 802.11 see Halperin et al. (2010).

4.4.3 The 802.11 MAC Sublayer Protocol

Let us now return from the land of electrical engineering to the land of computer science. The 802.11 MAC sublayer protocol is quite different from that of Ethernet, due to two factors that are fundamental to wireless communication.

First, radios are nearly always half duplex, meaning that they cannot transmit and listen for noise bursts at the same time on a single frequency. The received signal can easily be a million times weaker than the transmitted signal, so it cannot be heard at the same time. With Ethernet, a station just waits until the ether goes silent and then starts transmitting. If it does not receive a noise burst back while transmitting the first 64 bytes, the frame has almost assuredly been delivered correctly. With wireless, this collision detection mechanism does not work.

Instead, 802.11 tries to avoid collisions with a protocol called **CSMA/CA** (**CSMA with Collision Avoidance**). This protocol is conceptually similar to Ethernet's CSMA/CD, with channel sensing before sending and exponential back off after collisions. However, a station that has a frame to send starts with a random backoff (except in the case that it has not used the channel recently and the channel is idle). It does not wait for a collision. The number of slots to backoff is chosen in the range 0 to, say, 15 in the case of the OFDM physical layer. The station waits until the channel is idle, by sensing that there is no signal for a short period of time (called the DIFS, as we explain below), and counts down idle slots, pausing when frames are sent. It sends its frame when the counter reaches 0. If the frame gets through, the destination immediately sends a short acknowledgement. Lack of an acknowledgement is inferred to indicate an error, whether a collision or otherwise. In this case, the sender doubles the backoff period and tries again, continuing with exponential backoff as in Ethernet until the frame has been successfully transmitted or the maximum number of retransmissions has been reached.

An example timeline is shown in Fig. 4-25. Station *A* is the first to send a frame. While *A* is sending, stations *B* and *C* become ready to send. They see that the channel is busy and wait for it to become idle. Shortly after *A* receives an acknowledgement, the channel goes idle. However, rather than sending a frame right away and colliding, *B* and *C* both perform a backoff. *C* picks a short backoff,

and thus sends first. *B* pauses its countdown while it senses that *C* is using the channel, and resumes after *C* has received an acknowledgement. *B* soon completes its backoff and sends its frame.

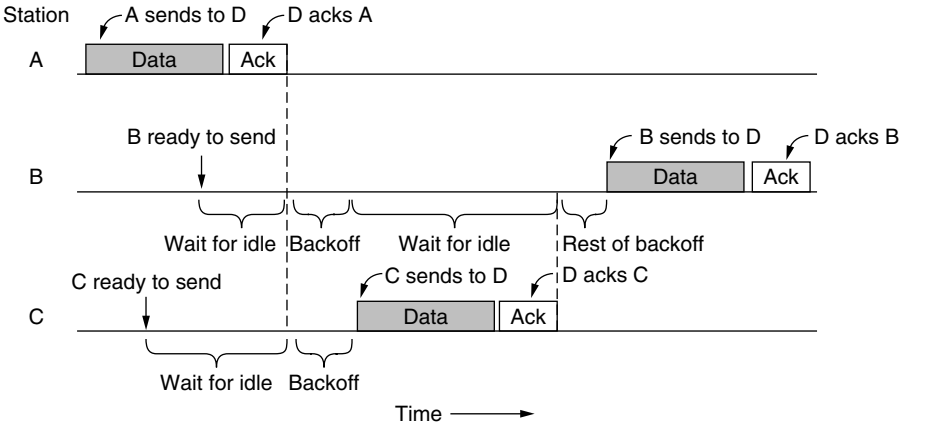

Figure 4-25. Sending a frame with CSMA/CA.

Compared to Ethernet, there are two main differences. First, starting backoffs early helps to avoid collisions. This avoidance is worthwhile because collisions are expensive, as the entire frame is transmitted even if one occurs. Second, acknowledgements are used to infer collisions because collisions cannot be detected.

This mode of operation is called **DCF** (**Distributed Coordination Function**) because each station acts independently, without any kind of central control. The standard also includes an optional mode of operation called **PCF** (**Point Coordination Function**) in which the access point controls all activity in its cell, just like a cellular base station. However, PCF is not used in practice because there is normally no way to prevent stations in another nearby network from transmitting competing traffic.

The second problem is that the transmission ranges of different stations may be different. With a wire, the system is engineered so that all stations can hear each other. With the complexities of RF propagation this situation does not hold for wireless stations. Consequently, situations such as the hidden terminal problem mentioned earlier and illustrated again in Fig. 4-26(a) can arise. Since not all stations are within radio range of each other, transmissions going on in one part of a cell may not be received elsewhere in the same cell. In this example, station *C* is transmitting to station *B*. If *A* senses the channel, it will not hear anything and will falsely conclude that it may now start transmitting to *B*. This decision leads to a collision.

The inverse situation is the exposed terminal problem, illustrated in Fig. 4-26(b). Here, *B* wants to send to *C*, so it listens to the channel. When it hears a

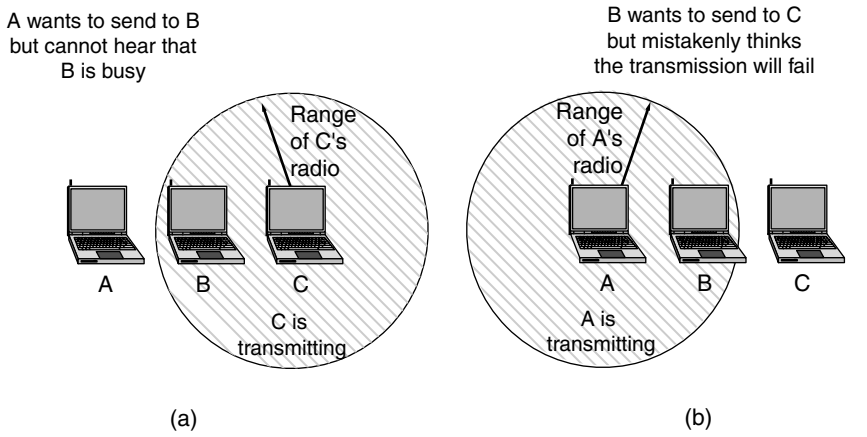

Figure 4-26. (a) The hidden terminal problem. (b) The exposed terminal problem.

transmission, it falsely concludes that it may not send to *C*, even though *A* may in fact be transmitting to *D* (not shown). This decision wastes a transmission opportunity.

To reduce ambiguities about which station is sending, 802.11 defines channel sensing to consist of both physical sensing and virtual sensing. Physical sensing simply checks the medium to see if there is a valid signal. With virtual sensing, each station keeps a logical record of when the channel is in use by tracking the **NAV** (**Network Allocation Vector**). Each frame carries a NAV field that says how long the sequence of which this frame is part will take to complete. Stations that overhear this frame know that the channel will be busy for the period indicated by the NAV, regardless of whether they can sense a physical signal. For example, the NAV of a data frame includes the time needed to send an acknowledgement. All stations that hear the data frame will defer during the acknowledgement period, whether or not they can hear the acknowledgement.

An optional RTS/CTS mechanism uses the NAV to prevent terminals from sending frames at the same time as hidden terminals. It is shown in Fig. 4-27. In this example, *A* wants to send to *B*. *C* is a station within range of *A* (and possibly within range of *B*, but that does not matter). *D* is a station within range of *B* but not within range of *A*.

The protocol starts when *A* decides it wants to send data to *B*. *A* begins by sending an RTS frame to *B* to request permission to send it a frame. If *B* receives this request, it answers with a CTS frame to indicate that the channel is clear to send. Upon receipt of the CTS, *A* sends its frame and starts an ACK timer. Upon correct receipt of the data frame, *B* responds with an ACK frame, completing the exchange. If *A*'s ACK timer expires before the ACK gets back to it, it is treated as a collision and the whole protocol is run again after a backoff.

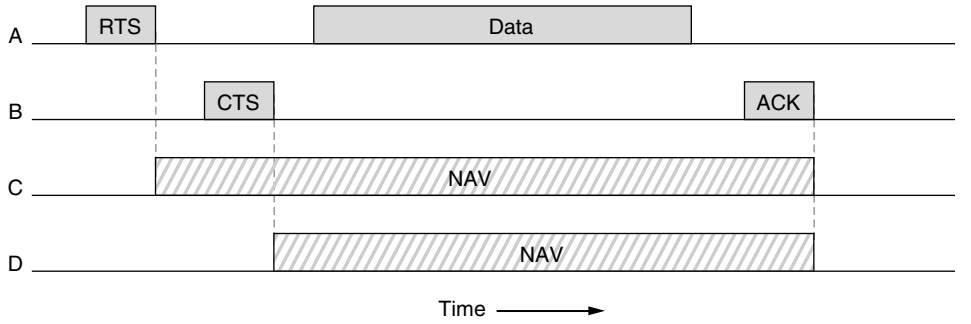

Figure 4-27. Virtual channel sensing using CSMA/CA.

Now let us consider this exchange from the viewpoints of *C* and *D*. *C* is within range of *A*, so it may receive the RTS frame. If it does, it realizes that someone is going to send data soon. From the information provided in the RTS request, it can estimate how long the sequence will take, including the final ACK. So, for the good of all, it desists from transmitting anything until the exchange is completed. It does so by updating its record of the NAV to indicate that the channel is busy, as shown in Fig. 4-27. *D* does not hear the RTS, but it does hear the CTS, so it also updates its NAV. Note that the NAV signals are not transmitted; they are just internal reminders to keep quiet for a certain period of time.

However, while RTS/CTS sounds good in theory, it is one of those designs that has proved to be of little value in practice. Several reasons why it is seldom used are known. It does not help for short frames (which are sent in place of the RTS) or for the AP (which everyone can hear, by definition). For other situations, it only slows down operation. RTS/CTS in 802.11 is a little different than in the MACA protocol we saw in Sec 4.2 because everyone hearing the RTS or CTS remains quiet for the duration to allow the ACK to get through without collision. Because of this, it does not help with exposed terminals as MACA did, only with hidden terminals. Most often there are few hidden terminals, and CSMA/CA already helps them by slowing down stations that transmit unsuccessfully, whatever the cause, to make it more likely that transmissions will succeed.

CSMA/CA with physical and virtual sensing is the core of the 802.11 protocol. However, there are several other mechanisms that have been developed to go with it. Each of these mechanisms was driven by the needs of real operation, so we will look at them briefly.

The first need we will look at is reliability. In contrast to wired networks, wireless networks are noisy and unreliable, in no small part due to interference from other kinds of devices, such as microwave ovens, which also use the unlicensed ISM bands. The use of acknowledgements and retransmissions is of little help if the probability of getting a frame through is small in the first place.

The main strategy that is used to increase successful transmissions is to lower the transmission rate. Slower rates use more robust modulations that are more likely to be received correctly for a given signal-to-noise ratio. If too many frames are lost, a station can lower the rate. If frames are delivered with little loss, a station can occasionally test a higher rate to see if it should be used.

Another strategy to improve the chance of the frame getting through undamaged is to send shorter frames. If the probability of any bit being in error is p, the probability of an n-bit frame being received entirely correctly is $(1 - p)^n$. For example, for $p = 10^{-4}$, the probability of receiving a full Ethernet frame (12,144 bits) correctly is less than 30%. Most frames will be lost. But if the frames are only a third as long (4048 bits) two thirds of them will be received correctly. Now most frames will get through and fewer retransmissions will be needed.

Shorter frames can be implemented by reducing the maximum size of the message that is accepted from the network layer. Alternatively, 802.11 allows frames to be split into smaller pieces, called **fragments**, each with its own checksum. The fragment size is not fixed by the standard, but is a parameter that can be adjusted by the AP. The fragments are individually numbered and acknowledged using a stop-and-wait protocol (i.e., the sender may not transmit fragment $k + 1$ until it has received the acknowledgement for fragment k). Once the channel has been acquired, multiple fragments are sent as a burst. They go one after the other with an acknowledgement (and possibly retransmissions) in between, until either the whole frame has been successfully sent or the transmission time reaches the maximum allowed. The NAV mechanism keeps other stations quiet only until the next acknowledgement, but another mechanism (see below) is used to allow a burst of fragments to be sent without other stations sending a frame in the middle.

The second need we will discuss is saving power. Battery life is always an issue with mobile wireless devices. The 802.11 standard pays attention to the issue of power management so that clients need not waste power when they have neither information to send nor to receive.

The basic mechanism for saving power builds on **beacon frames**. Beacons are periodic broadcasts by the AP (e.g., every 100 msec). The frames advertise the presence of the AP to clients and carry system parameters, such as the identifier of the AP, the time, how long until the next beacon, and security settings.

Clients can set a power-management bit in frames that they send to the AP to tell it that they are entering **power-save mode**. In this mode, the client can doze and the AP will buffer traffic intended for it. To check for incoming traffic, the client wakes up for every beacon, and checks a traffic map that is sent as part of the beacon. This map tells the client if there is buffered traffic. If so, the client sends a poll message to the AP, which then sends the buffered traffic. The client can then go back to sleep until the next beacon is sent.

Another power-saving mechanism, called **APSD** (**Automatic Power Save Delivery**), was also added to 802.11 in 2005. With this new mechanism, the AP buffers frames and sends them to a client just after the client sends frames to the

AP. The client can then go to sleep until it has more traffic to send (and receive). This mechanism works well for applications such as VoIP that have frequent traffic in both directions. For example, a VoIP wireless phone might use it to send and receive frames every 20 msec, much more frequently than the beacon interval of 100 msec, while dozing in between.

The third and last need we will examine is quality of service. When the VoIP traffic in the preceding example competes with peer-to-peer traffic, the VoIP traffic will suffer. It will be delayed due to contention with the high-bandwidth peer-to-peer traffic, even though the VoIP bandwidth is low. These delays are likely to degrade the voice calls. To prevent this degradation, we would like to let the VoIP traffic go ahead of the peer-to-peer traffic, as it is of higher priority.

IEEE 802.11 has a clever mechanism to provide this kind of quality of service that was introduced as set of extensions under the name 802.11e in 2005. It works by extending CSMA/CA with carefully defined intervals between frames. After a frame has been sent, a certain amount of idle time is required before any station may send a frame to check that the channel is no longer in use. The trick is to define different time intervals for different kinds of frames.

Five intervals are depicted in Fig. 4-28. The interval between regular data frames is called the **DIFS** (**DCF InterFrame Spacing**). Any station may attempt to acquire the channel to send a new frame after the medium has been idle for DIFS. The usual contention rules apply, and binary exponential backoff may be needed if a collision occurs. The shortest interval is **SIFS** (**Short InterFrame Spacing**). It is used to allow the parties in a single dialog the chance to go first. Examples include letting the receiver send an ACK, other control frame sequences like RTS and CTS, or letting a sender transmit a burst of fragments. Sending the next fragment after waiting only SIFS is what prevents another station from jumping in with a frame in the middle of the exchange.

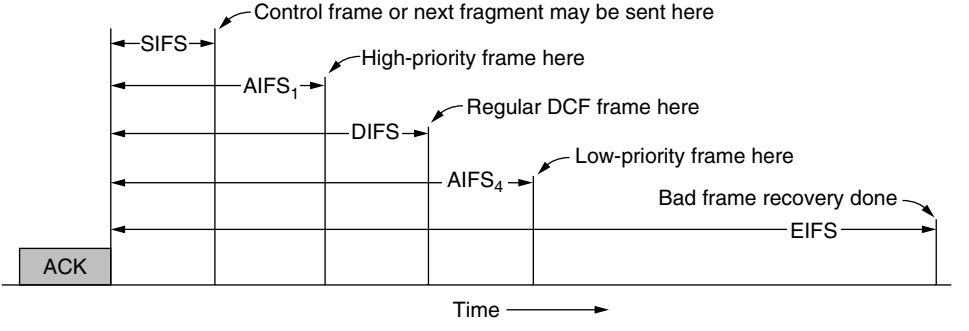

Figure 4-28. Interframe spacing in 802.11.

The two **AIFS** (**Arbitration InterFrame Space**) intervals show examples of two different priority levels. The short interval, $AIFS_1$, is smaller than DIFS but longer than SIFS. It can be used by the AP to move voice or other high-priority

traffic to the head of the line. The AP will wait for a shorter interval before it sends the voice traffic, and thus send it before regular traffic. The long interval, $AIFS_4$, is larger than DIFS. It is used for background traffic that can be deferred until after regular traffic. The AP will wait for a longer interval before it sends this traffic, giving regular traffic the opportunity to transmit first. The complete quality of service mechanism defines four different priority levels that have different backoff parameters as well as different idle parameters.

The last time interval, **EIFS** (**Extended InterFrame Spacing**), is used only by a station that has just received a bad or unknown frame, to report the problem. The idea is that since the receiver may have no idea of what is going on, it should wait a while to avoid interfering with an ongoing dialog between two stations.

A further part of the quality of service extensions is the notion of a **TXOP** or **transmission opportunity**. The original CSMA/CA mechanism let stations send one frame at a time. This design was fine until the range of rates increased. With 802.11a/g, one station might be sending at 6 Mbps and another station be sending at 54 Mbps. They each get to send one frame, but the 6-Mbps station takes nine times as long (ignoring fixed overheads) as the 54-Mbps station to send its frame. This disparity has the unfortunate side effect of slowing down a fast sender who is competing with a slow sender to roughly the rate of the slow sender. For example, again ignoring fixed overheads, when sending alone the 6-Mbps and 54-Mbps senders will get their own rates, but when sending together they will both get 5.4 Mbps on average. It is a stiff penalty for the fast sender. This issue is known as the **rate anomaly** (Heusse et al., 2003).

With transmission opportunities, each station gets an equal amount of airtime, not an equal number of frames. Stations that send at a higher rate for their airtime will get higher throughput. In our example, when sending together the 6-Mbps and 54-Mbps senders will now get 3 Mbps and 27 Mbps, respectively.

4.4.4 The 802.11 Frame Structure

The 802.11 standard defines three different classes of frames in the air: data, control, and management. Each of these has a header with a variety of fields used within the MAC sublayer. In addition, there are some headers used by the physical layer, but these mostly deal with the modulation techniques used, so we will not discuss them here.

We will look at the format of the data frame as an example. It is shown in Fig. 4-29. First comes the *Frame control* field, which is made up of 11 subfields. The first of these is the *Protocol version*, set to 00. It is there to allow future versions of 802.11 to operate at the same time in the same cell. Then come the *Type* (data, control, or management) and *Subtype* fields (e.g., RTS or CTS). For a regular data frame (without quality of service), they are set to 10 and 0000 in binary. The *To DS* and *From DS* bits are set to indicate whether the frame is going to or coming from the network connected to the APs, which is called the distribution

system. The *More fragments* bit means that more fragments will follow. The *Retry* bit marks a retransmission of a frame sent earlier. The *Power management* bit indicates that the sender is going into power-save mode. The *More data* bit indicates that the sender has additional frames for the receiver. The *Protected Frame* bit indicates that the frame body has been encrypted for security. We will discuss security briefly in the next section. Finally, the *Order* bit tells the receiver that the higher layer expects the sequence of frames to arrive strictly in order.

Figure 4-29. Format of the 802.11 data frame.

The second field of the data frame, the *Duration* field, tells how long the frame and its acknowledgement will occupy the channel, measured in microseconds. It is present in all types of frames, including control frames, and is what stations use to manage the NAV mechanism.

Next come addresses. Data frames sent to or from an AP have three addresses, all in standard IEEE 802 format. The first address is the receiver, and the second address is the transmitter. They are obviously needed, but what is the third address for? Remember that the AP is simply a relay point for frames as they travel between a client and another point on the network, perhaps a distant client or a portal to the Internet. The third address gives this distant endpoint.

The *Sequence* field numbers frames so that duplicates can be detected. Of the 16 bits available, 4 identify the fragment and 12 carry a number that is advanced with each new transmission. The *Data* field contains the payload, up to 2312 bytes. The first bytes of this payload are in a format known as **LLC** (**Logical Link Control**). This layer is the glue that identifies the higher-layer protocol (e.g., IP) to which the payloads should be passed. Last comes the *Frame check sequence*, which is the same 32-bit CRC we saw in Sec. 3.2.2 and elsewhere.

Management frames have the same format as data frames, plus a format for the data portion that varies with the subtype (e.g., parameters in beacon frames). Control frames are short. Like all frames, they have the *Frame control*, *Duration*, and *Frame check sequence* fields. However, they may have only one address and no data portion. Most of the key information is conveyed with the *Subtype* field (e.g., ACK, RTS and CTS).

4.4.5 Services

The 802.11 standard defines the services that the clients, the access points, and the network connecting them must be a conformant wireless LAN. These services cluster into several groups.

The **association** service is used by mobile stations to connect themselves to APs. Typically, it is used just after a station moves within radio range of the AP. Upon arrival, the station learns the identity and capabilities of the AP, either from beacon frames or by directly asking the AP. The capabilities include the data rates supported, security arrangements, power-saving capabilities, quality of service support, and more. The station sends a request to associate with the AP. The AP may accept or reject the request.

Reassociation lets a station change its preferred AP. This facility is useful for mobile stations moving from one AP to another AP in the same extended 802.11 LAN, like a handover in the cellular network. If it is used correctly, no data will be lost as a consequence of the handover. (But 802.11, like Ethernet, is just a best-effort service.) Either the station or the AP may also **disassociate**, breaking their relationship. A station should use this service before shutting down or leaving the network. The AP may use it before going down for maintenance.

Stations must also **authenticate** before they can send frames via the AP, but authentication is handled in different ways depending on the choice of security scheme. If the 802.11 network is "open," anyone is allowed to use it. Otherwise, credentials are needed to authenticate. The recommended scheme, called **WPA2** (**WiFi Protected Access 2**), implements security as defined in the 802.11i standard. (Plain WPA is an interim scheme that implements a subset of 802.11i. We will skip it and go straight to the complete scheme.) With WPA2, the AP can talk to an authentication server that has a username and password database to determine if the station is allowed to access the network. Alternatively a pre-shared key, which is a fancy name for a network password, may be configured. Several frames are exchanged between the station and the AP with a challenge and response that lets the station prove it has the right credentials. This exchange happens after association.

The scheme that was used before WPA is called **WEP** (**Wired Equivalent Privacy**). For this scheme, authentication with a preshared key happens before association. However, its use is discouraged because of design flaws that make WEP easy to compromise. The first practical demonstration that WEP was broken came when Adam Stubblefield was a summer intern at AT&T (Stubblefield et al., 2002). He was able to code up and test an attack in one week, much of which was spent getting permission from management to buy the WiFi cards needed for experiments. Software to crack WEP passwords is now freely available.

Once frames reach the AP, the **distribution** service determines how to route them. If the destination is local to the AP, the frames can be sent out directly over the air. Otherwise, they will have to be forwarded over the wired network. The

integration service handles any translation that is needed for a frame to be sent outside the 802.11 LAN, or to arrive from outside the 802.11 LAN. The common case here is connecting the wireless LAN to the Internet.

Data transmission is what it is all about, so 802.11 naturally provides a **data delivery** service. This service lets stations transmit and receive data using the protocols we described earlier in this chapter. Since 802.11 is modeled on Ethernet and transmission over Ethernet is not guaranteed to be 100% reliable, transmission over 802.11 is not guaranteed to be reliable either. Higher layers must deal with detecting and correcting errors.

Wireless is a broadcast signal. For information sent over a wireless LAN to be kept confidential, it must be encrypted. This goal is accomplished with a **privacy** service that manages the details of encryption and decryption. The encryption algorithm for WPA2 is based on **AES** (**Advanced Encryption Standard**), a U.S. government standard approved in 2002. The keys that are used for encryption are determined during the authentication procedure.

To handle traffic with different priorities, there is a **QOS traffic scheduling** service. It uses the protocols we described to give voice and video traffic preferential treatment compared to best-effort and background traffic. A companion service also provides higher-layer timer synchronization. This lets stations coordinate their actions, which may be useful for media processing.

Finally, there are two services that help stations manage their use of the spectrum. The **transmit power control** service gives stations the information they need to meet regulatory limits on transmit power that vary from region to region. The **dynamic frequency selection** service give stations the information they need to avoid transmitting on frequencies in the 5-GHz band that are being used for radar in the proximity.

With these services, 802.11 provides a rich set of functionality for connecting nearby mobile clients to the Internet. It has been a huge success, and the standard has repeatedly been amended to add more functionality. For a perspective on where the standard has been and where it is heading, see Hiertz et al. (2010).

4.5 BROADBAND WIRELESS

We have been indoors too long. Let us go outdoors, where there is quite a bit of interesting networking over the so-called "last mile." With the deregulation of the telephone systems in many countries, competitors to the entrenched telephone companies are now often allowed to offer local voice and high-speed Internet service. There is certainly plenty of demand. The problem is that running fiber or coax to millions of homes and businesses is prohibitively expensive. What is a competitor to do?

The answer is broadband wireless. Erecting a big antenna on a hill just outside of town is much easier and cheaper than digging many trenches and stringing

cables. Thus, companies have begun to experiment with providing multimegabit wireless communication services for voice, Internet, movies on demand, etc.

To stimulate the market, IEEE formed a group to standardize a broadband wireless metropolitan area network. The next number available in the 802 numbering space was **802.16**, so the standard got this number. Informally the technology is called **WiMAX** (**Worldwide Interoperability for Microwave Access**). We will use the terms 802.16 and WiMAX interchangeably.

The first 802.16 standard was approved in December 2001. Early versions provided a wireless local loop between fixed points with a line of sight to each other. This design soon changed to make WiMAX a more competitive alternative to cable and DSL for Internet access. By January 2003, 802.16 had been revised to support non-line-of-sight links by using OFDM technology at frequencies between 2 GHz and 10 GHz. This change made deployment much easier, though stations were still fixed locations. The rise of 3G cellular networks posed a threat by promising high data rates *and* mobility. In response, 802.16 was enhanced again to allow mobility at vehicular speeds by December 2005. Mobile broadband Internet access is the target of the current standard, IEEE 802.16-2009.

Like the other 802 standards, 802.16 was heavily influenced by the OSI model, including the (sub)layers, terminology, service primitives, and more. Unfortunately, also like OSI, it is fairly complicated. In fact, the **WiMAX Forum** was created to define interoperable subsets of the standard for commercial offerings. In the following sections, we will give a brief description of some of the highlights of the common forms of 802.16 air interface, but this treatment is far from complete and leaves out many details. For additional information about WiMAX and broadband wireless in general, see Andrews et al. (2007).

4.5.1 Comparison of 802.16 with 802.11 and 3G

At this point you may be thinking: why devise a new standard? Why not just use 802.11 or 3G? In fact, WiMAX combines aspects of both 802.11 and 3G, making it more like a 4G technology.

Like 802.11, WiMAX is all about wirelessly connecting devices to the Internet at megabit/sec speeds, instead of using cable or DSL. The devices may be mobile, or at least portable. WiMAX did not start by adding low-rate data on the side of voice-like cellular networks; 802.16 was designed to carry IP packets over the air and to connect to an IP-based wired network with a minimum of fuss. The packets may carry peer-to-peer traffic, VoIP calls, or streaming media to support a range of applications. Also like 802.11, it is based on OFDM technology to ensure good performance in spite of wireless signal degradations such as multipath fading, and on MIMO technology to achieve high levels of throughput.

However, WiMAX is more like 3G (and thus unlike 802.11) in several key respects. The key technical problem is to achieve high capacity by the efficient use of spectrum, so that a large number of subscribers in a coverage area can all get

high throughput. The typical distances are at least 10 times larger than for an 802.11 network. Consequently, WiMAX base stations are more powerful than 802.11 Access Points (APs). To handle weaker signals over larger distances, the base station uses more power and better antennas, and it performs more processing to handle errors. To maximize throughput, transmissions are carefully scheduled by the base station for each particular subscriber; spectrum use is not left to chance with CSMA/CA, which may waste capacity with collisions.

Licensed spectrum is the expected case for WiMAX, typically around 2.5 GHz in the U.S. The whole system is substantially more optimized than 802.11. This complexity is worth it, considering the large amount of money involved for licensed spectrum. Unlike 802.11, the result is a managed and reliable service with good support for quality of service.

With all of these features, 802.16 most closely resembles the 4G cellular networks that are now being standardized under the name **LTE** (**Long Term Evolution**). While 3G cellular networks are based on CDMA and support voice and data, 4G cellular networks will be based on OFDM with MIMO, and they will target data, with voice as just one application. It looks like WiMAX and 4G are on a collision course in terms of technology and applications. Perhaps this convergence is unsurprising, given that the Internet is the killer application and OFDM and MIMO are the best-known technologies for efficiently using the spectrum.

4.5.2 The 802.16 Architecture and Protocol Stack

The 802.16 architecture is shown in Fig. 4-30. Base stations connect directly to the provider's backbone network, which is in turn connected to the Internet. The base stations communicate with stations over the wireless air interface. Two kinds of stations exist. Subscriber stations remain in a fixed location, for example, broadband Internet access for homes. Mobile stations can receive service while they are moving, for example, a car equipped with WiMAX.

The 802.16 protocol stack that is used across the air interface is shown in Fig. 4-31. The general structure is similar to that of the other 802 networks, but with more sublayers. The bottom layer deals with transmission, and here we have shown only the popular offerings of 802.16, fixed and mobile WiMAX. There is a different physical layer for each offering. Both layers operate in licensed spectrum below 11 GHz and use OFDM, but in different ways.

Above the physical layer, the data link layer consists of three sublayers. The bottom one deals with privacy and security, which is far more crucial for public outdoor networks than for private indoor networks. It manages encryption, decryption, and key management.

Next comes the MAC common sublayer part. This part is where the main protocols, such as channel management, are located. The model here is that the base station completely controls the system. It can schedule the downlink (i.e., base to subscriber) channels very efficiently and plays a major role in managing

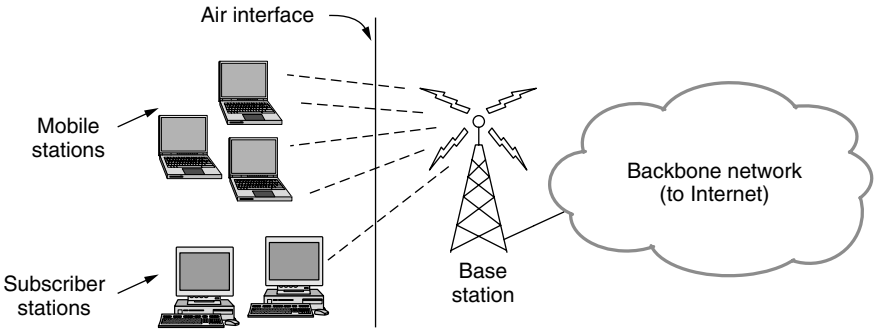

Figure 4-30. The 802.16 architecture.

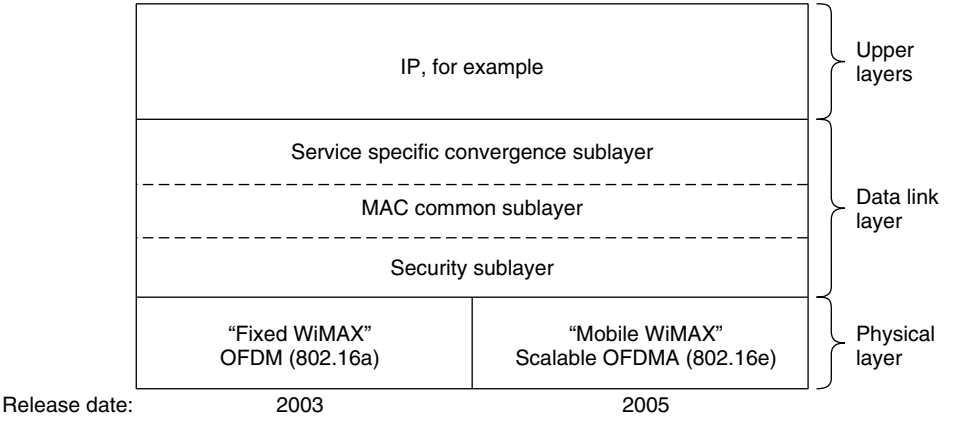

Figure 4-31. The 802.16 protocol stack.

the uplink (i.e., subscriber to base) channels as well. An unusual feature of this MAC sublayer is that, unlike those of the other 802 protocols, it is completely connection oriented, in order to provide quality of service guarantees for telephony and multimedia communication.

The service-specific convergence sublayer takes the place of the logical link sublayer in the other 802 protocols. Its function is to provide an interface to the network layer. Different convergence layers are defined to integrate seamlessly with different upper layers. The important choice is IP, though the standard defines mappings for protocols such as Ethernet and ATM too. Since IP is connectionless and the 802.16 MAC sublayer is connection-oriented, this layer must map between addresses and connections.

4.5.3 The 802.16 Physical Layer

Most WiMAX deployments use licensed spectrum around either 3.5 GHz or 2.5 GHz. As with 3G, finding available spectrum is a key problem. To help, the 802.16 standard is designed for flexibility. It allows operation from 2 GHz to 11 GHz. Channels of different sizes are supported, for example, 3.5 MHz for fixed WiMAX and from 1.25 MHz to 20 MHz for mobile WiMAX.

Transmissions are sent over these channels with OFDM, the technique we described in Sec. 2.5.3. Compared to 802.11, the 802.16 OFDM design is optimized to make the most out of licensed spectrum and wide area transmissions. The channel is divided into more subcarriers with a longer symbol duration to tolerate larger wireless signal degradations; WiMAX parameters are around 20 times larger than comparable 802.11 parameters. For example, in mobile WiMAX there are 512 subcarriers for a 5-MHz channel and the time to send a symbol on each subcarrier is roughly 100 μsec.

Symbols on each subcarrier are sent with QPSK, QAM-16, or QAM-64, modulation schemes we described in Sec. 2.5.3. When the mobile or subscriber station is near the base station and the received signal has a high signal-to-noise ratio (SNR), QAM-64 can be used to send 6 bits per symbol. To reach distant stations with a low SNR, QPSK can be used to deliver 2 bits per symbol. The data is first coded for error correction with the convolutional coding (or better schemes) that we described in Sec. 3.2.1. This coding is common on noisy channels to tolerate some bit errors without needing to send retransmissions. In fact, the modulation and coding methods should sound familiar by now as they are used for many networks we have studied, including 802.11 cable, and DSL. The net result is that a base station can support up to 12.6 Mbps of downlink traffic and 6.2 Mbps of uplink traffic per 5-MHz channel and pair of antennas.

One thing the designers of 802.16 did not like was a certain aspect of the way GSM and DAMPS work. Both of those systems use equal frequency bands for upstream and downstream traffic. That is, they implicitly assume there is as much upstream traffic as downstream traffic. For voice, traffic is symmetric for the most part, but for Internet access (and certainly Web surfing) there is often more downstream traffic than upstream traffic. The ratio is often 2:1, 3:1, or more:1.

So, the designers chose a flexible scheme for dividing the channel between stations, called **OFDMA** (**Orthogonal Frequency Division Multiple Access**). With OFDMA, different sets of subcarriers can be assigned to different stations, so that more than one station can send or receive at once. If this were 802.11, all subcarriers would be used by one station to send at any given moment. The added flexibility in how bandwidth is assigned can increase performance because a given subcarrier might be faded at one receiver due to multipath effects but clear at another. Subcarriers can be assigned to the stations that can use them best.

As well as having asymmetric traffic, stations usually alternate between sending and receiving. This method is called **TDD** (**Time Division Duplex**). The

alternative method, in which a station sends and receives at the same time (on different subcarrier frequencies), is called **FDD** (**Frequency Division Duplex**). WiMAX allows both methods, but TDD is preferred because it is easier to implement and more flexible.

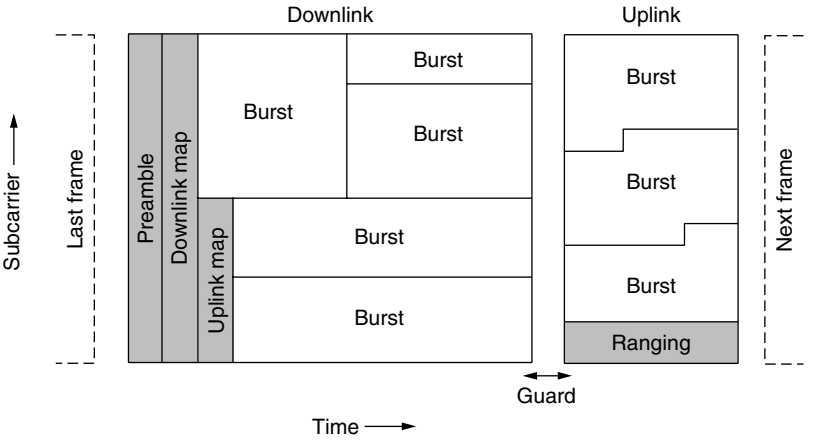

Figure 4-32. Frame structure for OFDMA with time division duplexing.

Fig. 4-32 shows an example of the frame structure that is repeated over time. It starts with a preamble to synchronize all stations, followed by downlink transmissions from the base station. First, the base station sends maps that tell all stations how the downlink and uplink subcarriers are assigned over the frame. The base station controls the maps, so it can allocate different amounts of bandwidth to stations from frame to frame depending on the needs of each station.

Next, the base station sends bursts of traffic to different subscriber and mobile stations on the subcarriers at the times given in the map. The downlink transmissions end with a guard time for stations to switch from receiving to transmitting. Finally, the subscriber and mobile stations send their bursts of traffic to the base station in the uplink positions that were reserved for them in the map. One of these uplink bursts is reserved for **ranging**, which is the process by which new stations adjust their timing and request initial bandwidth to connect to the base station. Since no connection is set up at this stage, new stations just transmit and hope there is no collision.

4.5.4 The 802.16 MAC Sublayer Protocol

The data link layer is divided into three sublayers, as we saw in Fig. 4-31. Since we will not study cryptography until Chap. 8, it is difficult to explain now how the security sublayer works. Suffice it to say that encryption is used to keep secret all data transmitted. Only the frame payloads are encrypted; the headers

are not. This property means that a snooper can see who is talking to whom but cannot tell what they are saying to each other.

If you already know something about cryptography, what follows is a one-paragraph explanation of the security sublayer. If you know nothing about cryptography, you are not likely to find the next paragraph terribly enlightening (but you might consider rereading it after finishing Chap. 8).

When a subscriber connects to a base station, they perform mutual authentication with RSA public-key cryptography using X.509 certificates. The payloads themselves are encrypted using a symmetric-key system, either AES (Rijndael) or DES with cipher block chaining. Integrity checking uses SHA-1. Now that was not so bad, was it?

Let us now look at the MAC common sublayer part. The MAC sublayer is connection-oriented and point-to-multipoint, which means that one base station communicates with multiple subscriber stations. Much of this design is borrowed from cable modems, in which one cable headend controls the transmissions of multiple cable modems at the customer premises.

The downlink direction is fairly straightforward. The base station controls the physical-layer bursts that are used to send information to the different subscriber stations. The MAC sublayer simply packs its frames into this structure. To reduce overhead, there are several different options. For example, MAC frames may be sent individually, or packed back-to-back into a group.

The uplink channel is more complicated since there are competing subscribers that need access to it. Its allocation is tied closely to the quality of service issue. Four classes of service are defined, as follows:

1. Constant bit rate service.
2. Real-time variable bit rate service.
3. Non-real-time variable bit rate service.
4. Best-effort service.

All service in 802.16 is connection-oriented. Each connection gets one of these service classes, determined when the connection is set up. This design is different from that of 802.11 or Ethernet, which are connectionless in the MAC sublayer.

Constant bit rate service is intended for transmitting uncompressed voice. This service needs to send a predetermined amount of data at predetermined time intervals. It is accommodated by dedicating certain bursts to each connection of this type. Once the bandwidth has been allocated, the bursts are available automatically, without the need to ask for each one.

Real-time variable bit rate service is for compressed multimedia and other soft real-time applications in which the amount of bandwidth needed at each instant may vary. It is accommodated by the base station polling the subscriber at a fixed interval to ask how much bandwidth is needed this time.

Non-real-time variable bit rate service is for heavy transmissions that are not real time, such as large file transfers. For this service, the base station polls the subscriber often, but not at rigidly prescribed time intervals. Connections with this service can also use best-effort service, described next, to request bandwidth.

Best-effort service is for everything else. No polling is done and the subscriber must contend for bandwidth with other best-effort subscribers. Requests for bandwidth are sent in bursts marked in the uplink map as available for contention. If a request is successful, its success will be noted in the next downlink map. If it is not successful, the unsuccessful subscriber have to try again later. To minimize collisions, the Ethernet binary exponential backoff algorithm is used.

4.5.5 The 802.16 Frame Structure

All MAC frames begin with a generic header. The header is followed by an optional payload and an optional checksum (CRC), as illustrated in Fig. 4-33. The payload is not needed in control frames, for example, those requesting channel slots. The checksum is (surprisingly) also optional, due to the error correction in the physical layer and the fact that no attempt is ever made to retransmit real-time frames. If no retransmissions will be attempted, why even bother with a checksum? But if there is a checksum, it is the standard IEEE 802 CRC, and acknowledgements and retransmissions are used for reliability.

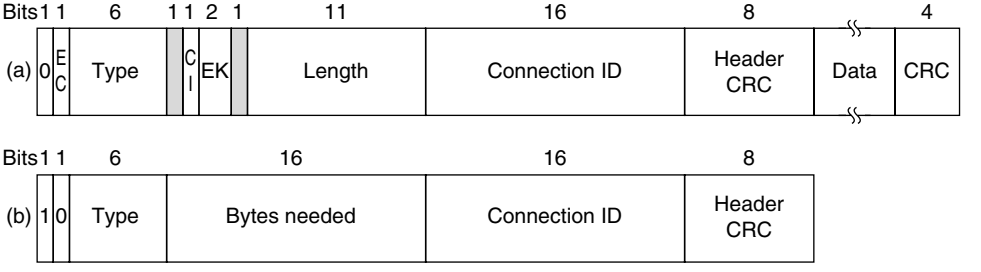

Figure 4-33. (a) A generic frame. (b) A bandwidth request frame.

A quick rundown of the header fields of Fig. 4-33(a) follows. The *EC* bit tells whether the payload is encrypted. The *Type* field identifies the frame type, mostly telling whether packing and fragmentation are present. The *CI* field indicates the presence or absence of the final checksum. The *EK* field tells which of the encryption keys is being used (if any). The *Length* field gives the complete length of the frame, including the header. The *Connection identifier* tells which connection this frame belongs to. Finally, the *Header CRC* field is a checksum over the header only, using the polynomial $x^8 + x^2 + x + 1$.

The 802.16 protocol has many kinds of frames. An example of a different type of frame, one that is used to request bandwidth, is shown in Fig. 4-33(b). It

starts with a 1 bit instead of a 0 bit and is otherwise similar to the generic header except that the second and third bytes form a 16-bit number telling how much bandwidth is needed to carry the specified number of bytes. Bandwidth request frames do not carry a payload or full-frame CRC.

A great deal more could be said about 802.16, but this is not the place to say it. For more information, please consult the IEEE 802.16-2009 standard itself.

4.6 BLUETOOTH

In 1994, the L. M. Ericsson company became interested in connecting its mobile phones to other devices (e.g., laptops) without cables. Together with four other companies (IBM, Intel, Nokia, and Toshiba), it formed a SIG (Special Interest Group, i.e., consortium) in 1998 to develop a wireless standard for interconnecting computing and communication devices and accessories using short-range, low-power, inexpensive wireless radios. The project was named **Bluetooth**, after Harald Blaatand (Bluetooth) II (940–981), a Viking king who unified (i.e., conquered) Denmark and Norway, also without cables.

Bluetooth 1.0 was released in July 1999, and since then the SIG has never looked back. All manner of consumer electronic devices now use Bluetooth, from mobile phones and laptops to headsets, printers, keyboards, mice, gameboxes, watches, music players, navigation units, and more. The Bluetooth protocols let these devices find and connect to each other, an act called **pairing**, and securely transfer data.

The protocols have evolved over the past decade, too. After the initial protocols stabilized, higher data rates were added to Bluetooth 2.0 in 2004. With the 3.0 release in 2009, Bluetooth can be used for device pairing in combination with 802.11 for high-throughput data transfer. The 4.0 release in December 2009 specified low-power operation. That will be handy for people who do not want to change the batteries regularly in all of those devices around the house. We will cover the main aspects of Bluetooth below.

4.6.1 Bluetooth Architecture

Let us start our study of the Bluetooth system with a quick overview of what it contains and what it is intended to do. The basic unit of a Bluetooth system is a **piconet**, which consists of a master node and up to seven active slave nodes within a distance of 10 meters. Multiple piconets can exist in the same (large) room and can even be connected via a bridge node that takes part in multiple piconets, as in Fig. 4-34. An interconnected collection of piconets is called a **scatternet**.

In addition to the seven active slave nodes in a piconet, there can be up to 255 parked nodes in the net. These are devices that the master has switched to a low-power state to reduce the drain on their batteries. In parked state, a device cannot

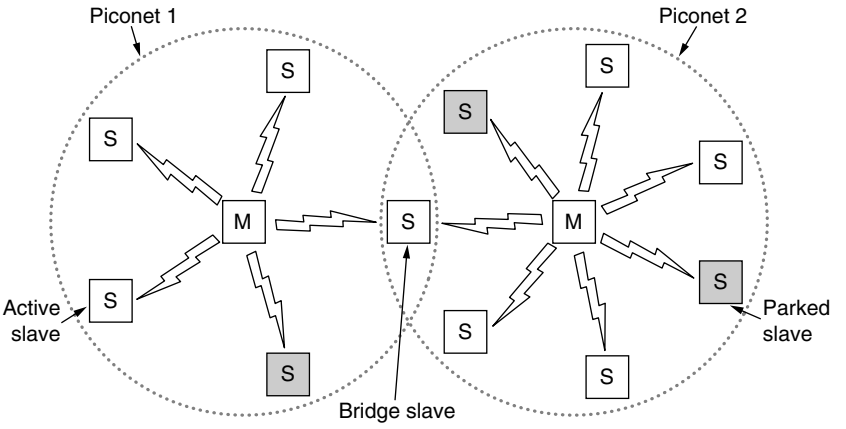

Figure 4-34. Two piconets can be connected to form a scatternet.

do anything except respond to an activation or beacon signal from the master. Two intermediate power states, hold and sniff, also exist, but these will not concern us here.

The reason for the master/slave design is that the designers intended to facilitate the implementation of complete Bluetooth chips for under $5. The consequence of this decision is that the slaves are fairly dumb, basically just doing whatever the master tells them to do. At its heart, a piconet is a centralized TDM system, with the master controlling the clock and determining which device gets to communicate in which time slot. All communication is between the master and a slave; direct slave-slave communication is not possible.

4.6.2 Bluetooth Applications

Most network protocols just provide channels between communicating entities and let application designers figure out what they want to use them for. For example, 802.11 does not specify whether users should use their notebook computers for reading email, surfing the Web, or something else. In contrast, the Bluetooth SIG specifies particular applications to be supported and provides different protocol stacks for each one. At the time of writing, there are 25 applications, which are called **profiles**. Unfortunately, this approach leads to a very large amount of complexity. We will omit the complexity here but will briefly look at the profiles to see more clearly what the Bluetooth SIG is trying to accomplish.

Six of the profiles are for different uses of audio and video. For example, the intercom profile allows two telephones to connect as walkie-talkies. The headset and hands-free profiles both provide voice communication between a headset and its base station, as might be used for hands-free telephony while driving a car.

Other profiles are for streaming stereo-quality audio and video, say, from a portable music player to headphones, or from a digital camera to a TV.

The human interface device profile is for connecting keyboards and mice to computers. Other profiles let a mobile phone or other computer receive images from a camera or send images to a printer. Perhaps of more interest is a profile to use a mobile phone as a remote control for a (Bluetooth-enabled) TV.

Still other profiles enable networking. The personal area network profile lets Bluetooth devices form an ad hoc network or remotely access another network, such as an 802.11 LAN, via an access point. The dial-up networking profile was actually the original motivation for the whole project. It allows a notebook computer to connect to a mobile phone containing a built-in modem without using wires.

Profiles for higher-layer information exchange have also been defined. The synchronization profile is intended for loading data into a mobile phone when it leaves home and collecting data from it when it returns.

We will skip the rest of the profiles, except to mention that some profiles serve as building blocks on which the above profiles are built. The generic access profile, on which all of the other profiles are built, provides a way to establish and maintain secure links (channels) between the master and the slaves. The other generic profiles define the basics of object exchange and audio and video transport. Utility profiles are used widely for functions such as emulating a serial line, which is especially useful for many legacy applications.

Was it really necessary to spell out all these applications in detail and provide different protocol stacks for each one? Probably not, but there were a number of different working groups that devised different parts of the standard, and each one just focused on its specific problem and generated its own profile. Think of this as Conway's Law in action. (In the April 1968 issue of *Datamation* magazine, Melvin Conway observed that if you assign *n* people to write a compiler, you will get an *n*-pass compiler, or more generally, the software structure mirrors the structure of the group that produced it.) It would probably have been possible to get away with two protocol stacks instead of 25, one for file transfer and one for streaming real-time communication.

4.6.3 The Bluetooth Protocol Stack

The Bluetooth standard has many protocols grouped loosely into the layers shown in Fig. 4-35. The first observation to make is that the structure does not follow the OSI model, the TCP/IP model, the 802 model, or any other model.

The bottom layer is the physical radio layer, which corresponds fairly well to the physical layer in the OSI and 802 models. It deals with radio transmission and modulation. Many of the concerns here have to do with the goal of making the system inexpensive so that it can become a mass-market item.

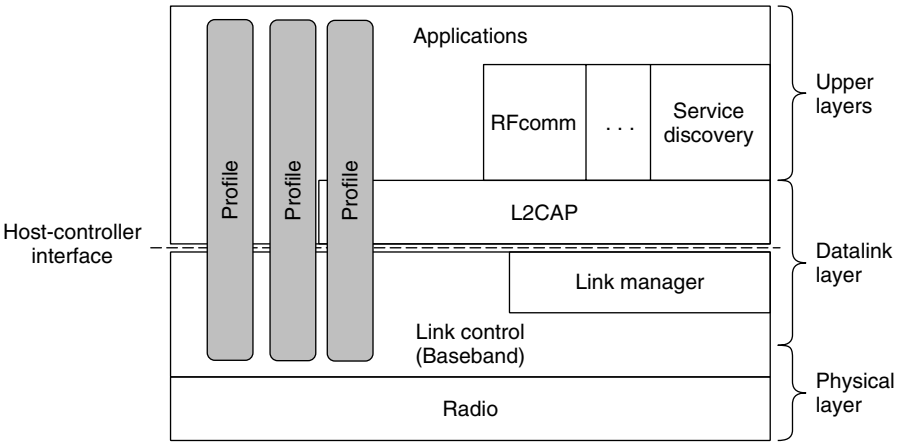

Figure 4-35. The Bluetooth protocol architecture.

The link control (or baseband) layer is somewhat analogous to the MAC sublayer but also includes elements of the physical layer. It deals with how the master controls time slots and how these slots are grouped into frames.

Next come two protocols that use the link control protocol. The link manager handles the establishment of logical channels between devices, including power management, pairing and encryption, and quality of service. It lies below the host controller interface line. This interface is a convenience for implementation: typically, the protocols below the line will be implemented on a Bluetooth chip, and the protocols above the line will be implemented on the Bluetooth device that hosts the chip.

The link protocol above the line is **L2CAP** (**Logical Link Control Adaptation Protocol**). It frames variable-length messages and provides reliability if needed. Many protocols use L2CAP, such as the two utility protocols that are shown. The service discovery protocol is used to locate services within the network. The RFcomm (Radio Frequency communication) protocol emulates the standard serial port found on PCs for connecting the keyboard, mouse, and modem, among other devices.

The top layer is where the applications are located. The profiles are represented by vertical boxes because they each define a slice of the protocol stack for a particular purpose. Specific profiles, such as the headset profile, usually contain only those protocols needed by that application and no others. For example, profiles may include L2CAP if they have packets to send but skip L2CAP if they have only a steady flow of audio samples.

In the following sections, we will examine the Bluetooth radio layer and various link protocols, since these roughly correspond to the physical and MAC sublayers in the other procotol stacks we have studied.

4.6.4 The Bluetooth Radio Layer

The radio layer moves the bits from master to slave, or vice versa. It is a low-power system with a range of 10 meters operating in the same 2.4-GHz ISM band as 802.11. The band is divided into 79 channels of 1 MHz each. To coexist with other networks using the ISM band, frequency hopping spread spectrum is used. There can be up to 1600 hops/sec over slots with a dwell time of 625 μsec. All the nodes in a piconet hop frequencies simultaneously, following the slot timing and pseudorandom hop sequence dictated by the master.

Unfortunately, it turned out that early versions of Bluetooth and 802.11 interfered enough to ruin each other's transmissions. Some companies responded by banning Bluetooth altogether, but eventually a technical solution was devised. The solution is for Bluetooth to adapt its hop sequence to exclude channels on which there are other RF signals. This process reduces the harmful interference. It is called **adaptive frequency hopping**.

Three forms of modulation are used to send bits on a channel. The basic scheme is to use frequency shift keying to send a 1-bit symbol every microsecond, giving a gross data rate of 1 Mbps. Enhanced rates were introduced with the 2.0 version of Bluetooth. These rates use phase shift keying to send either 2 or 3 bits per symbol, for gross data rates of 2 or 3 Mbps. The enhanced rates are only used in the data portion of frames.

4.6.5 The Bluetooth Link Layers

The link control (or baseband) layer is the closest thing Bluetooth has to a MAC sublayer. It turns the raw bit stream into frames and defines some key formats. In the simplest form, the master in each piconet defines a series of 625-μsec time slots, with the master's transmissions starting in the even slots and the slaves' transmissions starting in the odd ones. This scheme is traditional time division multiplexing, with the master getting half the slots and the slaves sharing the other half. Frames can be 1, 3, or 5 slots long. Each frame has an overhead of 126 bits for an access code and header, plus a settling time of 250–260 μsec per hop to allow the inexpensive radio circuits to become stable. The payload of the frame can be encrypted for confidentiality with a key that is chosen when the master and slave connect. Hops only happen between frames, not during a frame. The result is that a 5-slot frame is much more efficient than a 1-slot frame because the overhead is constant but more data is sent.

The link manager protocol sets up logical channels, called **links**, to carry frames between the master and a slave device that have discovered each other. A pairing procedure is followed to make sure that the two devices are allowed to communicate before the link is used. The old pairing method is that both devices must be configured with the same four-digit PIN (Personal Identification Number). The matching PIN is how each device would know that it was connecting to

the right remote device. However, unimaginative users and devices default to PINs such as "0000" and "1234" meant that this method provided very little security in practice.

The new **secure simple pairing** method enables users to confirm that both devices are displaying the same passkey, or to observe the passkey on one device and enter it into the second device. This method is more secure because users do not have to choose or set a PIN. They merely confirm a longer, device-generated passkey. Of course, it cannot be used on some devices with limited input/output, such as a hands-free headset.

Once pairing is complete, the link manager protocol sets up the links. Two main kinds of links exist to carry user data. The first is the **SCO** (**Synchronous Connection Oriented**) link. It is used for real-time data, such as telephone connections. This type of link is allocated a fixed slot in each direction. A slave may have up to three SCO links with its master. Each SCO link can transmit one 64,000-bps PCM audio channel. Due to the time-critical nature of SCO links, frames sent over them are never retransmitted. Instead, forward error correction can be used to increase reliability.

The other kind is the **ACL** (**Asynchronous ConnectionLess**) link. This type of link is used for packet-switched data that is available at irregular intervals. ACL traffic is delivered on a best-effort basis. No guarantees are given. Frames can be lost and may have to be retransmitted. A slave may have only one ACL link to its master.

The data sent over ACL links come from the L2CAP layer. This layer has four major functions. First, it accepts packets of up to 64 KB from the upper layers and breaks them into frames for transmission. At the far end, the frames are reassembled into packets. Second, it handles the multiplexing and demultiplexing of multiple packet sources. When a packet has been reassembled, the L2CAP layer determines which upper-layer protocol to hand it to, for example, RFcomm or service discovery. Third, L2CAP handles error control and retransmission. It detects errors and resends packets that were not acknowledged. Finally, L2CAP enforces quality of service requirements between multiple links.

4.6.6 The Bluetooth Frame Structure

Bluetooth defines several frame formats, the most important of which is shown in two forms in Fig. 4-36. It begins with an access code that usually identifies the master so that slaves within radio range of two masters can tell which traffic is for them. Next comes a 54-bit header containing typical MAC sublayer fields. If the frame is sent at the basic rate, the data field comes next. It has up to 2744 bits for a five-slot transmission. For a single time slot, the format is the same except that the data field is 240 bits.

If the frame is sent at the enhanced rate, the data portion may have up to two or three times as many bits because each symbol carries 2 or 3 bits instead of 1

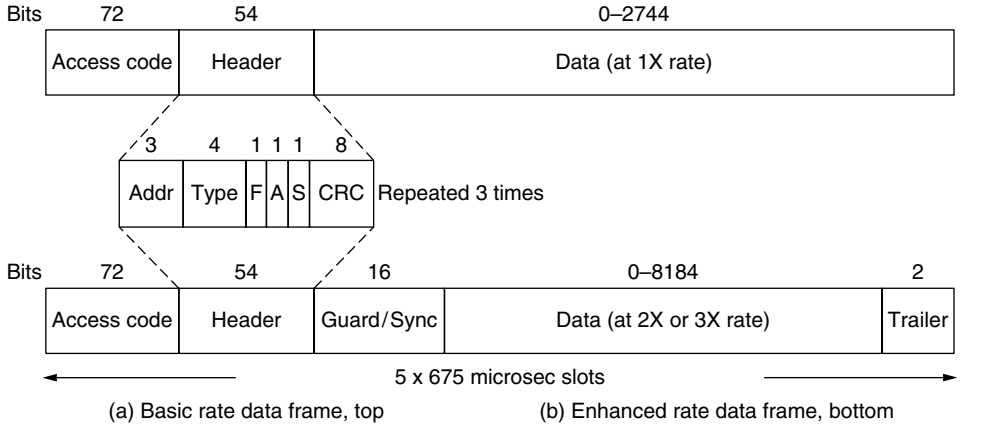

Figure 4-36. Typical Bluetooth data frame at (a) basic and (b) enhanced, data rates.

bit. These data are preceded by a guard field and a synchronization pattern that is used to switch to the faster data rate. That is, the access code and header are carried at the basic rate and only the data portion is carried at the faster rate. Enhanced-rate frames end with a short trailer.

Let us take a quick look at the common header. The *Address* field identifies which of the eight active devices the frame is intended for. The *Type* field identifies the frame type (ACL, SCO, poll, or null), the type of error correction used in the data field, and how many slots long the frame is. The *Flow* bit is asserted by a slave when its buffer is full and cannot receive any more data. This bit enables a primitive form of flow control. The *Acknowledgement* bit is used to piggyback an ACK onto a frame. The *Sequence* bit is used to number the frames to detect retransmissions. The protocol is stop-and-wait, so 1 bit is enough. Then comes the 8-bit header *Checksum*. The entire 18-bit header is repeated three times to form the 54-bit header shown in Fig. 4-36. On the receiving side, a simple circuit examines all three copies of each bit. If all three are the same, the bit is accepted. If not, the majority opinion wins. Thus, 54 bits of transmission capacity are used to send 10 bits of header. The reason is that to reliably send data in a noisy environment using cheap, low-powered (2.5 mW) devices with little computing capacity, a great deal of redundancy is needed.

Various formats are used for the data field for ACL and SCO frames. The basic-rate SCO frames are a simple example to study: the data field is always 240 bits. Three variants are defined, permitting 80, 160, or 240 bits of actual payload, with the rest being used for error correction. In the most reliable version (80-bit payload), the contents are just repeated three times, the same as the header.

We can work out the capacity with this frame as follows. Since the slave may use only the odd slots, it gets 800 slots/sec, just as the master does. With an 80-bit

payload, the channel capacity from the slave is 64,000 bps as is the channel capacity from the master. This capacity is exactly enough for a single full-duplex PCM voice channel (which is why a hop rate of 1600 hops/sec was chosen). That is, despite a raw bandwidth of 1 Mbps, a single full-duplex uncompressed voice channel can completely saturate the piconet. The efficiency of 13% is the result of spending 41% of the capacity on settling time, 20% on headers, and 26% on repetition coding. This shortcoming highlights the value of the enhanced rates and frames of more than a single slot.

There is much more to be said about Bluetooth, but no more space to say it here. For the curious, the Bluetooth 4.0 specification contains all the details.

4.7 RFID

We have looked at MAC designs from LANs up to MANs and down to PANs. As a last example, we will study a category of low-end wireless devices that people may not recognize as forming a computer network: the **RFID** (**Radio Frequency IDentification**) tags and readers that we described in Sec. 1.5.4.

RFID technology takes many forms, used in smartcards, implants for pets, passports, library books, and more. The form that we will look at was developed in the quest for an **EPC** (**Electronic Product Code**) that started with the Auto-ID Center at the Massachusetts Institute of Technology in 1999. An EPC is a replacement for a barcode that can carry a larger amount of information and is electronically readable over distances up to 10 m, even when it is not visible. It is different technology than, for example, the RFID used in passports,which must be placed quite close to a reader to perform a transaction. The ability to communicate over a distance makes EPCs more relevant to our studies.

EPCglobal was formed in 2003 to commercialize the RFID technology developed by the Auto-ID Center. The effort got a boost in 2005 when Walmart required its top 100 suppliers to label all shipments with RFID tags. Widespread deployment has been hampered by the difficulty of competing with cheap printed barcodes, but new uses, such as in drivers licenses, are now growing. We will describe the second generation of this technology, which is informally called **EPC Gen 2** (EPCglobal, 2008).

4.7.1 EPC Gen 2 Architecture

The architecture of an EPC Gen 2 RFID network is shown in Fig. 4-37. It has two key components: tags and readers. RFID tags are small, inexpensive devices that have a unique 96-bit EPC identifier and a small amount of memory that can be read and written by the RFID reader. The memory might be used to record the location history of an item, for example, as it moves through the supply chain.

Often, the tags look like stickers that can be placed on, for example, pairs of jeans on the shelves in a store. Most of the sticker is taken up by an antenna that is printed onto it. A tiny dot in the middle is the RFID integrated circuit. Alternatively, the RFID tags can be integrated into an object, such as a driver's license. In both cases, the tags have no battery and they must gather power from the radio transmissions of a nearby RFID reader to run. This kind of tag is called a "Class 1" tag to distinguish it from more capable tags that have batteries.

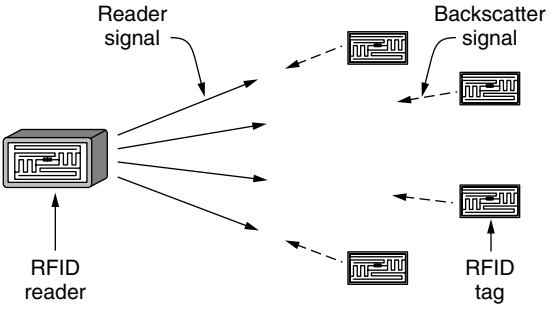

Figure 4-37. RFID architecture.

The readers are the intelligence in the system, analogous to base stations and access points in cellular and WiFi networks. Readers are much more powerful than tags. They have their own power sources, often have multiple antennas, and are in charge of when tags send and receive messages. As there will commonly be multiple tags within the reading range, the readers must solve the multiple access problem. There may be multiple readers that can contend with each other in the same area, too.

The main job of the reader is to inventory the tags in the neighborhood, that is, to discover the identifiers of the nearby tags. The inventory is accomplished with the physical layer protocol and the tag-identification protocol that are outlined in the following sections.

4.7.2 EPC Gen 2 Physical Layer

The physical layer defines how bits are sent between the RFID reader and tags. Much of it uses methods for sending wireless signals that we have seen previously. In the U.S., transmissions are sent in the unlicensed 902–928 MHz ISM band. This band falls in the UHF (Ultra High Frequency) range, so the tags are referred to as UHF RFID tags. The reader performs frequency hopping at least every 400 msec to spread its signal across the channel, to limit interference and satisfy regulatory requirements. The reader and tags use forms of ASK (Amplitude Shift Keying) modulation that we described in Sec. 2.5.2 to encode bits. They take turns to send bits, so the link is half duplex.

There are two main differences from other physical layers that we have studied. The first is that the reader is always transmitting a signal, regardless of whether it is the reader or tag that is communicating. Naturally, the reader transmits a signal to send bits to tags. For the tags to send bits to the reader, the reader transmits a fixed carrier signal that carries no bits. The tags harvest this signal to get the power they need to run; otherwise, a tag would not be able to transmit in the first place. To send data, a tag changes whether it is reflecting the signal from the reader, like a radar signal bouncing off a target, or absorbing it.

This method is called **backscatter**. It differs from all the other wireless situations we have seen so far, in which the sender and receiver never both transmit at the same time. Backscatter is a low-energy way for the tag to create a weak signal of its own that shows up at the reader. For the reader to decode the incoming signal, it must filter out the outgoing signal that it is transmitting. Because the tag signal is weak, tags can only send bits to the reader at a low rate, and tags cannot receive or even sense transmissions from other tags.

The second difference is that very simple forms of modulation are used so that they can be implemented on a tag that runs on very little power and costs only a few cents to make. To send data to the tags, the reader uses two amplitude levels. Bits are determined to be either a 0 or a 1, depending on how long the reader waits before a low-power period. The tag measures the time between low-power periods and compares this time to a reference measured during a preamble. As shown in Fig. 4-38, 1s are longer than 0s.

Tag responses consist of the tag alternating its backscatter state at fixed intervals to create a series of pulses in the signal. Anywhere from one to eight pulse periods can be used to encode each 0 or 1, depending on the need for reliability. 1s have fewer transitions than 0s, as is shown with an example of two-pulse period coding in Fig. 4-38.

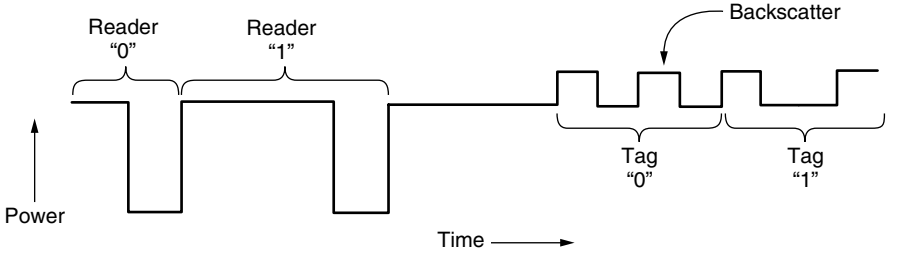

Figure 4-38. Reader and tag backscatter signals.

4.7.3 EPC Gen 2 Tag Identification Layer

To inventory the nearby tags, the reader needs to receive a message from each tag that gives the identifier for the tag. This situation is a multiple access problem for which the number of tags is unknown in the general case. The reader might

broadcast a query to ask all tags to send their identifiers. However, tags that replied right away would then collide in much the same way as stations on a classic Ethernet.

We have seen many ways of tackling the multiple access problem in this chapter. The closest protocol for the current situation, in which the tags cannot hear each others' transmissions, is slotted ALOHA, one of the earliest protocols we studied. This protocol is adapted for use in Gen 2 RFID.

The sequence of messages used to identify a tag is shown in Fig. 4-39. In the first slot (slot 0), the reader sends a *Query* message to start the process. Each *QRepeat* message advances to the next slot. The reader also tells the tags the range of slots over which to randomize transmissions. Using a range is necessary because the reader synchronizes tags when it starts the process; unlike stations on an Ethernet, tags do not wake up with a message at a time of their choosing.

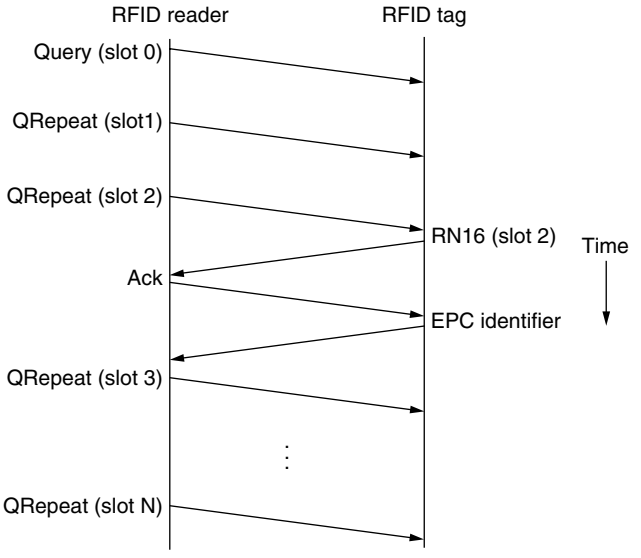

Figure 4-39. Example message exchange to identify a tag.

Tags pick a random slot in which to reply. In Fig. 4-39, the tag replies in slot 2. However, tags do not send their identifiers when they first reply. Instead, a tag sends a short 16-bit random number in an *RN16* message. If there is no collision, the reader receives this message and sends an *ACK* message of its own. At this stage, the tag has acquired the slot and sends its EPC identifier.

The reason for this exchange is that EPC identifiers are long, so collisions on these messages would be expensive. Instead, a short exchange is used to test whether the tag can safely use the slot to send its identifier. Once its identifier has been successfully transmitted, the tag temporarily stops responding to new *Query* messages so that all the remaining tags can be identified.

A key problem is for the reader to adjust the number of slots to avoid collisions, but without using so many slots that performance suffers. This adjustment is analogous to binary exponential backoff in Ethernet. If the reader sees too many slots with no responses or too many slots with collisions, it can send a *QAdjust* message to decrease or increase the range of slots over which the tags are responding.

The RFID reader can perform other operations on the tags. For example, it can select a subset of tags before running an inventory, allowing it to collect responses from, say, tagged jeans but not tagged shirts. The reader can also write data to tags as they are identified. This feature could be used to record the point of sale or other relevant information.

4.7.4 Tag Identification Message Formats

The format of the *Query* message is shown in Fig. 4-40 as an example of a reader-to-tag message. The message is compact because the downlink rates are limited, from 27 kbps up to 128 kbps. The *Command* field carries the code 1000 to identify the message as a *Query*.

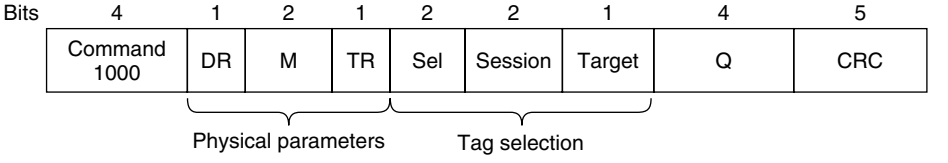

Figure 4-40. Format of the Query message.

The next flags, *DR*, *M*, and *TR*, determine the physical layer parameters for reader transmissions and tag responses. For example, the response rate may be set to between 5 kbps and 640 kbps. We will skip over the details of these flags.

Then come three fields, *Sel*, *Session*, and *Target*, that select the tags to respond. As well as the readers being able to select a subset of identifiers, the tags keep track of up to four concurrent sessions and whether they have been identified in those sessions. In this way, multiple readers can operate in overlapping coverage areas by using different sessions.

Next is the most important parameter for this command, Q. This field defines the range of slots over which tags will respond, from 0 to 2^Q-1. Finally, there is a CRC to protect the message fields. At 5 bits, it is shorter than most CRCs we have seen, but the *Query* message is much shorter than most packets too.

Tag-to-reader messages are simpler. Since the reader is in control, it knows what message to expect in response to each of its transmissions. The tag responses simply carry data, such as the EPC identifier.

Originally the tags were just for identification purposes. However, they have grown over time to resemble very small computers. Some research tags have sensors and are able to run small programs to gather and process data (Sample et al., 2008). One vision for this technology is the "Internet of things" that connects objects in the physical world to the Internet (Welbourne et al., 2009; and Gershenfeld et al., 2004).

4.8 DATA LINK LAYER SWITCHING

Many organizations have multiple LANs and wish to connect them. Would it not be convenient if we could just join the LANs together to make a larger LAN? In fact, we can do this when the connections are made with devices called **bridges**. The Ethernet switches we described in Sec. 4.3.4 are a modern name for bridges; they provide functionality that goes beyond classic Ethernet and Ethernet hubs to make it easy to join multiple LANs into a larger and faster network. We shall use the terms "bridge" and "switch" interchangeably.

Bridges operate in the data link layer, so they examine the data link layer addresses to forward frames. Since they are not supposed to examine the payload field of the frames they forward, they can handle IP packets as well as other kinds of packets, such as AppleTalk packets. In contrast, *routers* examine the addresses in packets and route based on them, so they only work with the protocols that they were designed to handle.

In this section, we will look at how bridges work and are used to join multiple physical LANs into a single logical LAN. We will also look at how to do the reverse and treat one physical LAN as multiple logical LANs, called **VLANs** (**Virtual LANs**). Both technologies provide useful flexibility for managing networks. For a comprehensive treatment of bridges, switches, and related topics, see Seifert and Edwards (2008) and Perlman (2000).

4.8.1 Uses of Bridges

Before getting into the technology of bridges, let us take a look at some common situations in which bridges are used. We will mention three reasons why a single organization may end up with multiple LANs.

First, many university and corporate departments have their own LANs to connect their own personal computers, servers, and devices such as printers. Since the goals of the various departments differ, different departments may set up different LANs, without regard to what other departments are doing. Sooner or later, though, there is a need for interaction, so bridges are needed. In this example, multiple LANs come into existence due to the autonomy of their owners.

Second, the organization may be geographically spread over several buildings separated by considerable distances. It may be cheaper to have separate LANs in each building and connect them with bridges and a few long-distance fiber optic links than to run all the cables to a single central switch. Even if laying the cables is easy to do, there are limits on their lengths (e.g., 200 m for twisted-pair gigabit Ethernet). The network would not work for longer cables due to the excessive signal attenuation or round-trip delay. The only solution is to partition the LAN and install bridges to join the pieces to increase the total physical distance that can be covered.

Third, it may be necessary to split what is logically a single LAN into separate LANs (connected by bridges) to accommodate the load. At many large universities, for example, thousands of workstations are available for student and faculty computing. Companies may also have thousands of employees. The scale of this system precludes putting all the workstations on a single LAN—there are more computers than ports on any Ethernet hub and more stations than allowed on a single classic Ethernet.

Even if it were possible to wire all the workstations together, putting more stations on an Ethernet hub or classic Ethernet would not add capacity. All of the stations share the same, fixed amount of bandwidth. The more stations there are, the less average bandwidth per station.

However, two separate LANs have twice the capacity of a single LAN. Bridges let the LANs be joined together while keeping this capacity. The key is not to send traffic onto ports where it is not needed, so that each LAN can run at full speed. This behavior also increases reliability, since on a single LAN a defective node that keeps outputting a continuous stream of garbage can clog up the entire LAN. By deciding what to forward and what not to forward, bridges act like fire doors in a building, preventing a single node that has gone berserk from bringing down the entire system.

To make these benefits easily available, ideally bridges should be completely transparent. It should be possible to go out and buy bridges, plug the LAN cables into the bridges, and have everything work perfectly, instantly. There should be no hardware changes required, no software changes required, no setting of address switches, no downloading of routing tables or parameters, nothing at all. Just plug in the cables and walk away. Furthermore, the operation of the existing LANs should not be affected by the bridges at all. As far as the stations are concerned, there should be no observable difference whether or not they are part of a bridged LAN. It should be as easy to move stations around the bridged LAN as it is to move them around a single LAN.

Surprisingly enough, it is actually possible to create bridges that are transparent. Two algorithms are used: a backward learning algorithm to stop traffic being sent where it is not needed; and a spanning tree algorithm to break loops that may be formed when switches are cabled together willy-nilly. Let us now take a look at these algorithms in turn to learn how this magic is accomplished.

4.8.2 Learning Bridges

The topology of two LANs bridged together is shown in Fig. 4-41 for two cases. On the left-hand side, two multidrop LANs, such as classic Ethernets, are joined by a special station—the bridge—that sits on both LANs. On the right-hand side, LANs with point-to-point cables, including one hub, are joined together. The bridges are the devices to which the stations and hub are attached. If the LAN technology is Ethernet, the bridges are better known as Ethernet switches.

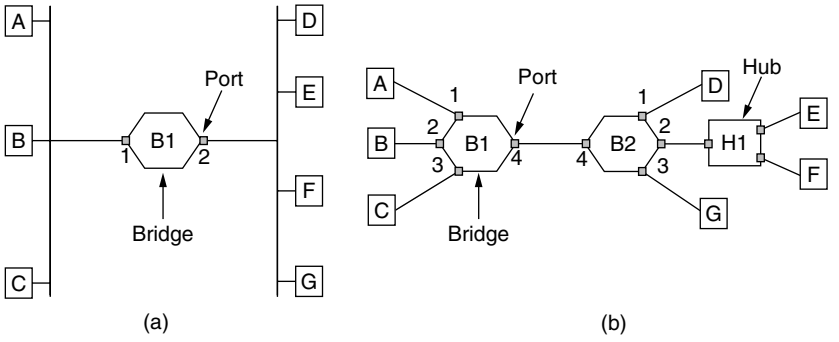

Figure 4-41. (a) Bridge connecting two multidrop LANs. (b) Bridges (and a hub) connecting seven point-to-point stations.

Bridges were developed when classic Ethernets were in use, so they are often shown in topologies with multidrop cables, as in Fig. 4-41(a). However, all the topologies that are encountered today are comprised of point-to-point cables and switches. The bridges work the same way in both settings. All of the stations attached to the same port on a bridge belong to the same collision domain, and this is different than the collision domain for other ports. If there is more than one station, as in a classic Ethernet, a hub, or a half-duplex link, the CSMA/CD protocol is used to send frames.

There is a difference, however, in how the bridged LANs are built. To bridge multidrop LANs, a bridge is added as a new station on each of the multidrop LANs, as in Fig. 4-41(a). To bridge point-to-point LANs, the hubs are either connected to a bridge or, preferably, replaced with a bridge to increase performance. In Fig. 4-41(b), bridges have replaced all but one hub.

Different kinds of cables can also be attached to one bridge. For example, the cable connecting bridge *B1* to bridge *B2* in Fig. 4-41(b) might be a long-distance fiber optic link, while the cable connecting the bridges to stations might be a short-haul twisted-pair line. This arrangement is useful for bridging LANs in different buildings.

Now let us consider what happens inside the bridges. Each bridge operates in promiscuous mode, that is, it accepts every frame transmitted by the stations

attached to each of its ports. The bridge must decide whether to forward or discard each frame, and, if the former, on which port to output the frame. This decision is made by using the destination address. As an example, consider the topology of Fig. 4-41(a). If station *A* sends a frame to station *B*, bridge *B1* will receive the frame on port 1. This frame can be immediately discarded without further ado because it is already on the correct port. However, in the topology of Fig. 4-41(b) suppose that *A* sends a frame to *D*. Bridge *B1* will receive the frame on port 1 and output it on port 4. Bridge *B2* will then receive the frame on its port 4 and output it on its port 1.

A simple way to implement this scheme is to have a big (hash) table inside the bridge. The table can list each possible destination and which output port it belongs on. For example, in Fig. 4-41(b), the table at *B1* would list *D* as belonging to port 4, since all *B1* has to know is which port to put frames on to reach *D*. That, in fact, more forwarding will happen later when the frame hits *B2* is not of interest to *B1*.

When the bridges are first plugged in, all the hash tables are empty. None of the bridges know where any of the destinations are, so they use a flooding algorithm: every incoming frame for an unknown destination is output on all the ports to which the bridge is connected except the one it arrived on. As time goes on, the bridges learn where destinations are. Once a destination is known, frames destined for it are put only on the proper port; they are not flooded.

The algorithm used by the bridges is **backward learning**. As mentioned above, the bridges operate in promiscuous mode, so they see every frame sent on any of their ports. By looking at the source addresses, they can tell which machines are accessible on which ports. For example, if bridge *B1* in Fig. 4-41(b) sees a frame on port 3 coming from *C*, it knows that *C* must be reachable via port 3, so it makes an entry in its hash table. Any subsequent frame addressed to *C* coming in to *B1* on any other port will be forwarded to port 3.

The topology can change as machines and bridges are powered up and down and moved around. To handle dynamic topologies, whenever a hash table entry is made, the arrival time of the frame is noted in the entry. Whenever a frame whose source is already in the table arrives, its entry is updated with the current time. Thus, the time associated with every entry tells the last time a frame from that machine was seen.

Periodically, a process in the bridge scans the hash table and purges all entries more than a few minutes old. In this way, if a computer is unplugged from its LAN, moved around the building, and plugged in again somewhere else, within a few minutes it will be back in normal operation, without any manual intervention. This algorithm also means that if a machine is quiet for a few minutes, any traffic sent to it will have to be flooded until it next sends a frame itself.

The routing procedure for an incoming frame depends on the port it arrives on (the source port) and the address to which it is destined (the destination address). The procedure is as follows.

1. If the port for the destination address is the same as the source port, discard the frame.
2. If the port for the destination address and the source port are different, forward the frame on to the destination port.
3. If the destination port is unknown, use flooding and send the frame on all ports except the source port.

You might wonder whether the first case can occur with point-to-point links. The answer is that it can occur if hubs are used to connect a group of computers to a bridge. An example is shown in Fig. 4-41(b) where stations *E* and *F* are connected to hub *H* 1, which is in turn connected to bridge *B2*. If *E* sends a frame to *F*, the hub will relay it to *B2* as well as to *F*. That is what hubs do—they wire all ports together so that a frame input on one port is simply output on all other ports. The frame will arrive at *B2* on port 4, which is already the right output port to reach the destination. Bridge *B2* need only discard the frame.

As each frame arrives, this algorithm must be applied, so it is usually implemented with special-purpose VLSI chips. The chips do the lookup and update the table entry, all in a few microseconds. Because bridges only look at the MAC addresses to decide how to forward frames, it is possible to start forwarding as soon as the destination header field has come in, before the rest of the frame has arrived (provided the output line is available, of course). This design reduces the latency of passing through the bridge, as well as the number of frames that the bridge must be able to buffer. It is referred to as **cut-through switching** or **wormhole routing** and is usually handled in hardware.

We can look at the operation of a bridge in terms of protocol stacks to understand what it means to be a link layer device. Consider a frame sent from station *A* to station *D* in the configuration of Fig. 4-41(a), in which the LANs are Ethernet. The frame will pass through one bridge. The protocol stack view of processing is shown in Fig. 4-42.

The packet comes from a higher layer and descends into the Ethernet MAC layer. It acquires an Ethernet header (and also a trailer, not shown in the figure). This unit is passed to the physical layer, goes out over the cable, and is picked up by the bridge.

In the bridge, the frame is passed up from the physical layer to the Ethernet MAC layer. This layer has extended processing compared to the Ethernet MAC layer at a station. It passes the frame to a relay, still within the MAC layer. The bridge relay function uses only the Ethernet MAC header to determine how to handle the frame. In this case, it passes the frame to the Ethernet MAC layer of the port used to reach station *D*, and the frame continues on its way.

In the general case, relays at a given layer can rewrite the headers for that layer. VLANs will provide an example shortly. In no case should the bridge look inside the frame and learn that it is carrying an IP packet; that is irrelevant to the

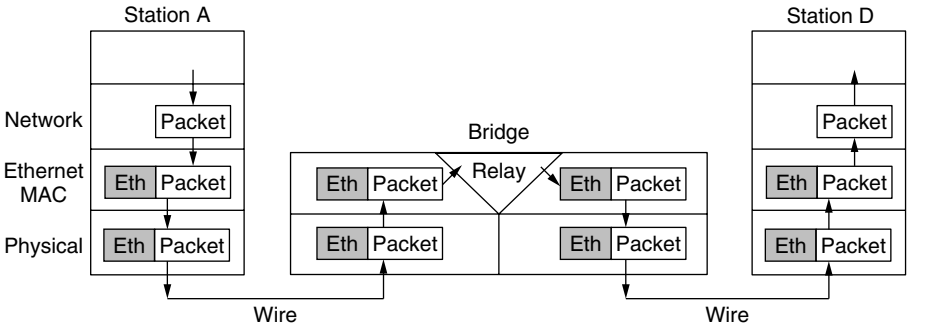

Figure 4-42. Protocol processing at a bridge.

bridge processing and would violate protocol layering. Also note that a bridge with k ports will have k instances of MAC and physical layers. The value of k is 2 for our simple example.

4.8.3 Spanning Tree Bridges

To increase reliability, redundant links can be used between bridges. In the example of Fig. 4-43, there are two links in parallel between a pair of bridges. This design ensures that if one link is cut, the network will not be partitioned into two sets of computers that cannot talk to each other.

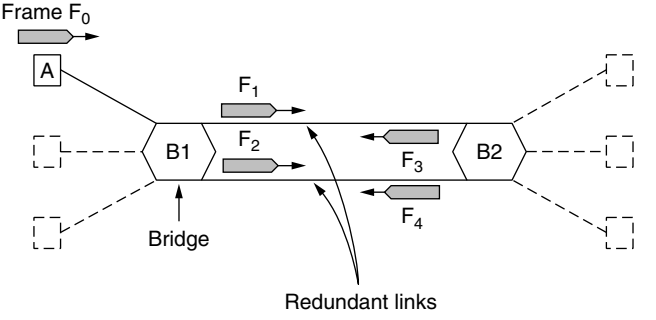

Figure 4-43. Bridges with two parallel links.

However, this redundancy introduces some additional problems because it creates loops in the topology. An example of these problems can be seen by looking at how a frame sent by A to a previously unobserved destination is handled in Fig. 4-43. Each bridge follows the normal rule for handling unknown destinations, which is to flood the frame. Call the frame from A that reaches bridge $B1$ frame F_0. The bridge sends copies of this frame out all of its other ports. We

will only consider the bridge ports that connect *B1* to *B2* (though the frame will be sent out the other ports, too). Since there are two links from *B1* to *B2*, two copies of the frame will reach *B2*. They are shown in Fig. 4-43 as F_1 and F_2.

Shortly thereafter, bridge *B2* receives these frames. However, it does not (and cannot) know that they are copies of the same frame, rather than two different frames sent one after the other. So bridge *B2* takes F_1 and sends copies of it out all the other ports, and it also takes F_2 and sends copies of it out all the other ports. This produces frames F_3 and F_4 that are sent along the two links back to *B1*. Bridge *B1* then sees two new frames with unknown destinations and copies them again. This cycle goes on forever.

The solution to this difficulty is for the bridges to communicate with each other and overlay the actual topology with a spanning tree that reaches every bridge. In effect, some potential connections between bridges are ignored in the interest of constructing a fictitious loop-free topology that is a subset of the actual topology.

For example, in Fig. 4-44 we see five bridges that are interconnected and also have stations connected to them. Each station connects to only one bridge. There are some redundant connections between the bridges so that frames will be forwarded in loops if all of the links are used. This topology can be thought of as a graph in which the bridges are the nodes and the point-to-point links are the edges. The graph can be reduced to a spanning tree, which has no cycles by definition, by dropping the links shown as dashed lines in Fig. 4-44. Using this spanning tree, there is exactly one path from every station to every other station. Once the bridges have agreed on the spanning tree, all forwarding between stations follows the spanning tree. Since there is a unique path from each source to each destination, loops are impossible.

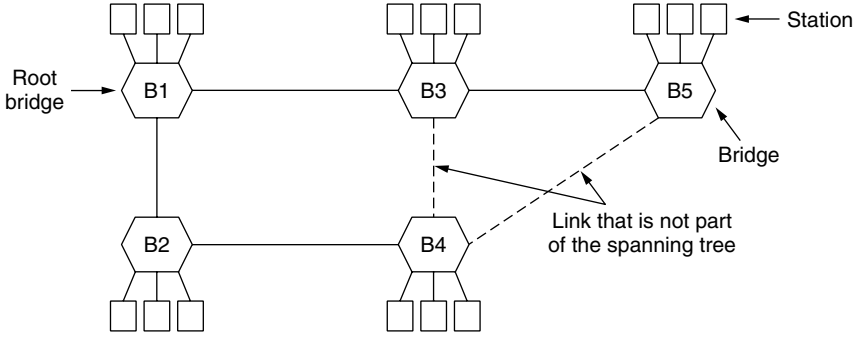

Figure 4-44. A spanning tree connecting five bridges. The dashed lines are links that are not part of the spanning tree.

To build the spanning tree, the bridges run a distributed algorithm. Each bridge periodically broadcasts a configuration message out all of its ports to its

neighbors and processes the messages it receives from other bridges, as described next. These messages are not forwarded, since their purpose is to build the tree, which can then be used for forwarding.

The bridges must first choose one bridge to be the root of the spanning tree. To make this choice, they each include an identifier based on their MAC address in the configuration message, as well as the identifier of the bridge they believe to be the root. MAC addresses are installed by the manufacturer and guaranteed to be unique worldwide, which makes these identifiers convenient and unique. The bridges choose the bridge with the lowest identifier to be the root. After enough messages have been exchanged to spread the news, all bridges will agree on which bridge is the root. In Fig. 4-44, bridge *B1* has the lowest identifier and becomes the root.

Next, a tree of shortest paths from the root to every bridge is constructed. In Fig. 4-44, bridges *B2* and *B3* can each be reached from bridge *B1* directly, in one hop that is a shortest path. Bridge *B4* can be reached in two hops, via either *B2* or *B3*. To break this tie, the path via the bridge with the lowest identifier is chosen, so *B4* is reached via *B2*. Bridge *B5* can be reached in two hops via *B3*.

To find these shortest paths, bridges include the distance from the root in their configuration messages. Each bridge remembers the shortest path it finds to the root. The bridges then turn off ports that are not part of the shortest path.

Although the tree spans all the bridges, not all the links (or even bridges) are necessarily present in the tree. This happens because turning off the ports prunes some links from the network to prevent loops. Even after the spanning tree has been established, the algorithm continues to run during normal operation to automatically detect topology changes and update the tree.

The algorithm for constructing the spanning tree was invented by Radia Perlman. Her job was to solve the problem of joining LANs without loops. She was given a week to do it, but she came up with the idea for the spanning tree algorithm in a day. Fortunately, this left her enough time to write it as a poem (Perlman, 1985):

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree.

The spanning tree algorithm was then standardized as IEEE 802.1D and used for many years. In 2001, it was revised to more rapidly find a new spanning tree after a topology change. For a detailed treatment of bridges, see Perlman (2000).

4.8.4 Repeaters, Hubs, Bridges, Switches, Routers, and Gateways

So far in this book, we have looked at a variety of ways to get frames and packets from one computer to another. We have mentioned repeaters, hubs, bridges, switches, routers, and gateways. All of these devices are in common use, but they all differ in subtle and not-so-subtle ways. Since there are so many of them, it is probably worth taking a look at them together to see what the similarities and differences are.

The key to understanding these devices is to realize that they operate in different layers, as illustrated in Fig. 4-45(a). The layer matters because different devices use different pieces of information to decide how to switch. In a typical scenario, the user generates some data to be sent to a remote machine. Those data are passed to the transport layer, which then adds a header (for example, a TCP header) and passes the resulting unit down to the network layer. The network layer adds its own header to form a network layer packet (e.g., an IP packet). In Fig. 4-45(b), we see the IP packet shaded in gray. Then the packet goes to the data link layer, which adds its own header and checksum (CRC) and gives the resulting frame to the physical layer for transmission, for example, over a LAN.

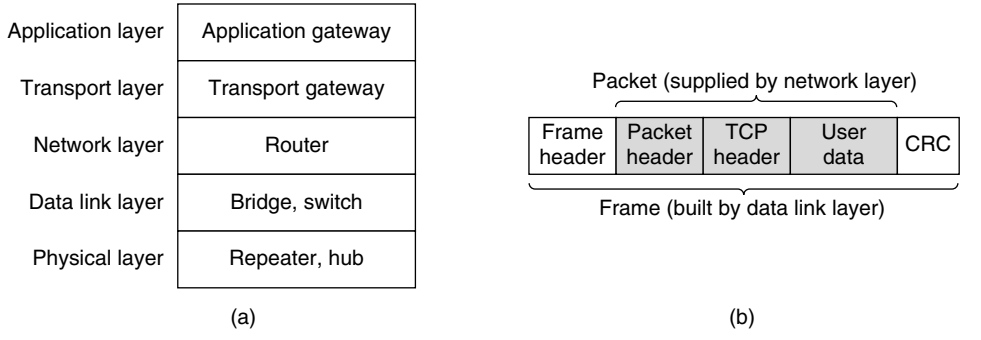

Figure 4-45. (a) Which device is in which layer. (b) Frames, packets, and headers.

Now let us look at the switching devices and see how they relate to the packets and frames. At the bottom, in the physical layer, we find the repeaters. These are analog devices that work with signals on the cables to which they are connected. A signal appearing on one cable is cleaned up, amplified, and put out on another cable. Repeaters do not understand frames, packets, or headers. They understand the symbols that encode bits as volts. Classic Ethernet, for example, was

designed to allow four repeaters that would boost the signal to extend the maximum cable length from 500 meters to 2500 meters.

Next we come to the hubs. A hub has a number of input lines that it joins electrically. Frames arriving on any of the lines are sent out on all the others. If two frames arrive at the same time, they will collide, just as on a coaxial cable. All the lines coming into a hub must operate at the same speed. Hubs differ from repeaters in that they do not (usually) amplify the incoming signals and are designed for multiple input lines, but the differences are slight. Like repeaters, hubs are physical layer devices that do not examine the link layer addresses or use them in any way.

Now let us move up to the data link layer, where we find bridges and switches. We just studied bridges at some length. A bridge connects two or more LANs. Like a hub, a modern bridge has multiple ports, usually enough for 4 to 48 input lines of a certain type. Unlike in a hub, each port is isolated to be its own collision domain; if the port has a full-duplex point-to-point line, the CSMA/CD algorithm is not needed. When a frame arrives, the bridge extracts the destination address from the frame header and looks it up in a table to see where to send the frame. For Ethernet, this address is the 48-bit destination address shown in Fig. 4-14. The bridge only outputs the frame on the port where it is needed and can forward multiple frames at the same time.

Bridges offer much better performance than hubs, and the isolation between bridge ports also means that the input lines may run at different speeds, possibly even with different network types. A common example is a bridge with ports that connect to 10-, 100-, and 1000-Mbps Ethernet. Buffering within the bridge is needed to accept a frame on one port and transmit the frame out on a different port. If frames come in faster than they can be retransmitted, the bridge may run out of buffer space and have to start discarding frames. For example, if a gigabit Ethernet is pouring bits into a 10-Mbps Ethernet at top speed, the bridge will have to buffer them, hoping not to run out of memory. This problem still exists even if all the ports run at the same speed because more than one port may be sending frames to a given destination port.

Bridges were originally intended to be able to join different kinds of LANs, for example, an Ethernet and a Token Ring LAN. However, this never worked well because of differences between the LANs. Different frame formats require copying and reformatting, which takes CPU time, requires a new checksum calculation, and introduces the possibility of undetected errors due to bad bits in the bridge's memory. Different maximum frame lengths are also a serious problem with no good solution. Basically, frames that are too large to be forwarded must be discarded. So much for transparency.

Two other areas where LANs can differ are security and quality of service. Some LANs have link-layer encryption, for example 802.11, and some do not, for example Ethernet. Some LANs have quality of service features such as priorities, for example 802.11, and some do not, for example Ethernet. Consequently, when

a frame must travel between these LANs, the security or quality of service expected by the sender may not be able to be provided. For all of these reasons, modern bridges usually work for one network type, and routers, which we will come to soon, are used instead to join networks of different types.

Switches are modern bridges by another name. The differences are more to do with marketing than technical issues, but there are a few points worth knowing. Bridges were developed when classic Ethernet was in use, so they tend to join relatively few LANs and thus have relatively few ports. The term ''switch'' is more popular nowadays. Also, modern installations all use point-to-point links, such as twisted-pair cables, so individual computers plug directly into a switch and thus the switch will tend to have many ports. Finally, ''switch'' is also used as a general term. With a bridge, the functionality is clear. On the other hand, a switch may refer to an Ethernet switch or a completely different kind of device that makes forwarding decisions, such as a telephone switch.

So far, we have seen repeaters and hubs, which are actually quite similar, as well as bridges and switches, which are even more similar to each other. Now we move up to routers, which are different from all of the above. When a packet comes into a router, the frame header and trailer are stripped off and the packet located in the frame's payload field (shaded in Fig. 4-45) is passed to the routing software. This software uses the packet header to choose an output line. For an IP packet, the packet header will contain a 32-bit (IPv4) or 128-bit (IPv6) address, but not a 48-bit IEEE 802 address. The routing software does not see the frame addresses and does not even know whether the packet came in on a LAN or a point-to-point line. We will study routers and routing in Chap. 5.

Up another layer, we find transport gateways. These connect two computers that use different connection-oriented transport protocols. For example, suppose a computer using the connection-oriented TCP/IP protocol needs to talk to a computer using a different connection-oriented transport protocol called SCTP. The transport gateway can copy the packets from one connection to the other, reformatting them as need be.

Finally, application gateways understand the format and contents of the data and can translate messages from one format to another. An email gateway could translate Internet messages into SMS messages for mobile phones, for example. Like ''switch,'' ''gateway'' is somewhat of a general term. It refers to a forwarding process that runs at a high layer.

4.8.5 Virtual LANs

In the early days of local area networking, thick yellow cables snaked through the cable ducts of many office buildings. Every computer they passed was plugged in. No thought was given to which computer belonged on which LAN. All the people in adjacent offices were put on the same LAN, whether they belonged together or not. Geography trumped corporate organization charts.

With the advent of twisted pair and hubs in the 1990s, all that changed. Buildings were rewired (at considerable expense) to rip out all the yellow garden hoses and install twisted pairs from every office to central wiring closets at the end of each corridor or in a central machine room, as illustrated in Fig. 4-46. If the Vice President in Charge of Wiring was a visionary, Category 5 twisted pairs were installed; if he was a bean counter, the existing (Category 3) telephone wiring was used (only to be replaced a few years later, when fast Ethernet emerged).

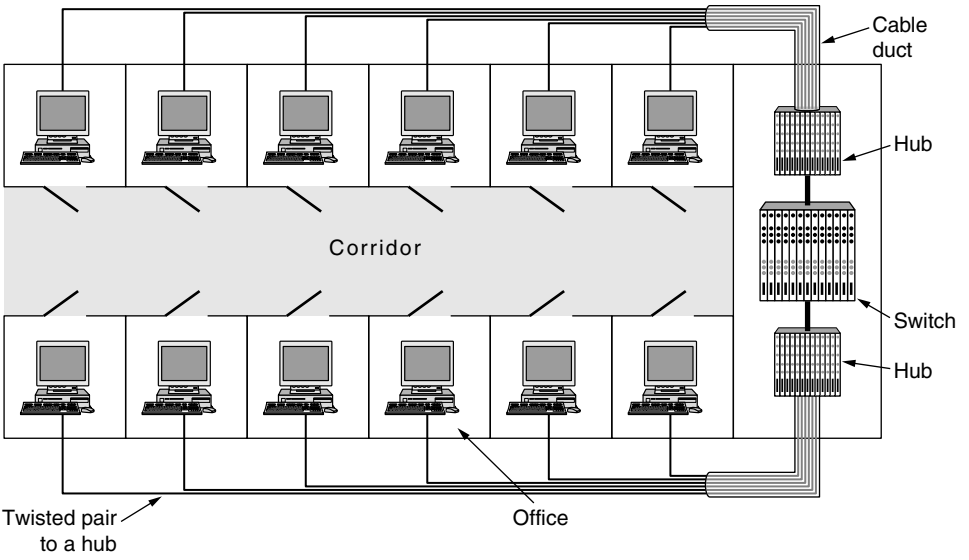

Figure 4-46. A building with centralized wiring using hubs and a switch.

Today, the cables have changed and hubs have become switches, but the wiring pattern is still the same. This pattern makes it possible to configure LANs logically rather than physically. For example, if a company wants k LANs, it could buy k switches. By carefully choosing which connectors to plug into which switches, the occupants of a LAN can be chosen in a way that makes organizational sense, without too much regard to geography.

Does it matter who is on which LAN? After all, in nearly all organizations, all the LANs are interconnected. In short, yes, it often matters. Network administrators like to group users on LANs to reflect the organizational structure rather than the physical layout of the building, for a variety of reasons. One issue is security. One LAN might host Web servers and other computers intended for public use. Another LAN might host computers containing the records of the Human Resources department that are not to be passed outside of the department. In such a situation, putting all the computers on a single LAN and not letting any of the servers be accessed from off the LAN makes sense. Management tends to frown when hearing that such an arrangement is impossible.

A second issue is load. Some LANs are more heavily used than others and it may be desirable to separate them. For example, if the folks in research are running all kinds of nifty experiments that sometimes get out of hand and saturate their LAN, the folks in management may not be enthusiastic about donating some of the capacity they were using for videoconferencing to help out. Then again, this might impress on management the need to install a faster network.

A third issue is broadcast traffic. Bridges broadcast traffic when the location of the destination is unknown, and upper-layer protocols use broadcasting as well. For example, when a user wants to send a packet to an IP address x, how does it know which MAC address to put in the frame? We will study this question in Chap. 5, but briefly summarized, the answer is that it broadcasts a frame containing the question "who owns IP address x?" Then it waits for an answer. As the number of computers in a LAN grows, so does the number of broadcasts. Each broadcast consumes more of the LAN capacity than a regular frame because it is delivered to every computer on the LAN. By keeping LANs no larger than they need to be, the impact of broadcast traffic is reduced.

Related to broadcasts is the problem that once in a while a network interface will break down or be misconfigured and begin generating an endless stream of broadcast frames. If the network is really unlucky, some of these frames will elicit responses that lead to ever more traffic. The result of this **broadcast storm** is that (1) the entire LAN capacity is occupied by these frames, and (2) all the machines on all the interconnected LANs are crippled just processing and discarding all the frames being broadcast.

At first it might appear that broadcast storms could be limited in scope by separating the LANs with bridges or switches, but if the goal is to achieve transparency (i.e., a machine can be moved to a different LAN across the bridge without anyone noticing it), then bridges have to forward broadcast frames.

Having seen why companies might want multiple LANs with restricted scopes, let us get back to the problem of decoupling the logical topology from the physical topology. Building a physical topology to reflect the organizational structure can add work and cost, even with centralized wiring and switches. For example, if two people in the same department work in different buildings, it may be easier to wire them to different switches that are part of different LANs. Even if this is not the case, a user might be shifted within the company from one department to another without changing offices, or might change offices without changing departments. This might result in the user being on the wrong LAN until an administrator changes the user's connector from one switch to another. Furthermore, the number of computers that belong to different departments may not be a good match for the number of ports on switches; some departments may be too small and others so big that they require multiple switches. This results in wasted switch ports that are not used.

In many companies, organizational changes occur all the time, meaning that system administrators spend a lot of time pulling out plugs and pushing them back

in somewhere else. Also, in some cases, the change cannot be made at all because the twisted pair from the user's machine is too far from the correct switch (e.g., in the wrong building), or the available switch ports are on the wrong LAN.

In response to customer requests for more flexibility, network vendors began working on a way to rewire buildings entirely in software. The resulting concept is called a **VLAN** (**Virtual LAN**). It has been standardized by the IEEE 802 committee and is now widely deployed in many organizations. Let us now take a look at it. For additional information about VLANs, see Seifert and Edwards (2008).

VLANs are based on VLAN-aware switches. To set up a VLAN-based network, the network administrator decides how many VLANs there will be, which computers will be on which VLAN, and what the VLANs will be called. Often the VLANs are (informally) named by colors, since it is then possible to print color diagrams showing the physical layout of the machines, with the members of the red LAN in red, members of the green LAN in green, and so on. In this way, both the physical and logical layouts are visible in a single view.

As an example, consider the bridged LAN of Fig. 4-47, in which nine of the machines belong to the G (gray) VLAN and five belong to the W (white) VLAN. Machines from the gray VLAN are spread across two switches, including two machines that connect to a switch via a hub.

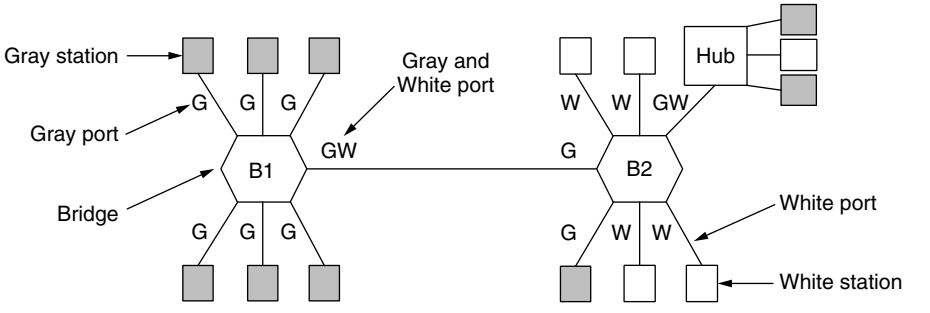

Figure 4-47. Two VLANs, gray and white, on a bridged LAN.

To make the VLANs function correctly, configuration tables have to be set up in the bridges. These tables tell which VLANs are accessible via which ports. When a frame comes in from, say, the gray VLAN, it must be forwarded on all the ports marked with a G. This holds for ordinary (i.e., unicast) traffic for which the bridges have not learned the location of the destination, as well as for multicast and broadcast traffic. Note that a port may be labeled with multiple VLAN colors.

As an example, suppose that one of the gray stations plugged into bridge *B1* in Fig. 4-47 sends a frame to a destination that has not been observed beforehand. Bridge *B1* will receive the frame and see that it came from a machine on the gray

VLAN, so it will flood the frame on all ports labeled G (except the incoming port). The frame will be sent to the five other gray stations attached to *B1* as well as over the link from *B1* to bridge *B2*. At bridge *B2*, the frame is similarly forwarded on all ports labeled G. This sends the frame to one further station and the hub (which will transmit the frame to all of its stations). The hub has both labels because it connects to machines from both VLANs. The frame is not sent on other ports without G in the label because the bridge knows that there are no machines on the gray VLAN that can be reached via these ports.

In our example, the frame is only sent from bridge *B1* to bridge *B2* because there are machines on the gray VLAN that are connected to *B2*. Looking at the white VLAN, we can see that the bridge *B2* port that connects to bridge *B1* is *not* labeled W. This means that a frame on the white VLAN will not be forwarded from bridge *B2* to bridge *B1*. This behavior is correct because no stations on the white VLAN are connected to *B1*.

The IEEE 802.1Q Standard

To implement this scheme, bridges need to know to which VLAN an incoming frame belongs. Without this information, for example, when bridge *B2* gets a frame from bridge *B1* in Fig. 4-47, it cannot know whether to forward the frame on the gray or white VLAN. If we were designing a new type of LAN, it would be easy enough to just add a VLAN field in the header. But what to do about Ethernet, which is the dominant LAN, and did not have any spare fields lying around for the VLAN identifier?

The IEEE 802 committee had this problem thrown into its lap in 1995. After much discussion, it did the unthinkable and changed the Ethernet header. The new format was published in IEEE standard **802.1Q**, issued in 1998. The new format contains a VLAN tag; we will examine it shortly. Not surprisingly, changing something as well established as the Ethernet header was not entirely trivial. A few questions that come to mind are:

1. Need we throw out several hundred million existing Ethernet cards?
2. If not, who generates the new fields?
3. What happens to frames that are already the maximum size?

Of course, the 802 committee was (only too painfully) aware of these problems and had to come up with solutions, which it did.

The key to the solution is to realize that the VLAN fields are only actually used by the bridges and switches and *not* by the user machines. Thus, in Fig. 4-47, it is not really essential that they are present on the lines going out to the end stations as long as they are on the line between the bridges. Also, to use VLANs, the bridges have to be VLAN aware. This fact makes the design feasible.

As to throwing out all existing Ethernet cards, the answer is no. Remember that the 802.3 committee could not even get people to change the *Type* field into a *Length* field. You can imagine the reaction to an announcement that all existing Ethernet cards had to be thrown out. However, new Ethernet cards are 802.1Q compliant and can correctly fill in the VLAN fields.

Because there can be computers (and switches) that are not VLAN aware, the first VLAN-aware bridge to touch a frame adds VLAN fields and the last one down the road removes them. An example of a mixed topology is shown in Fig. 4-48. In this figure, VLAN-aware computers generate tagged (i.e., 802.1Q) frames directly, and further switching uses these tags. The shaded symbols are VLAN-aware and the empty ones are not.

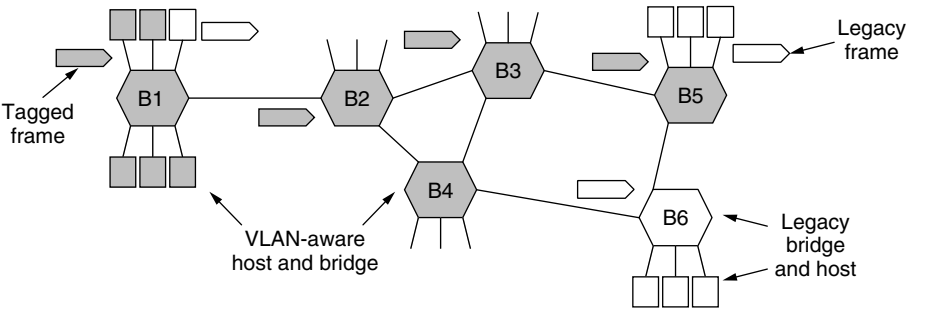

Figure 4-48. Bridged LAN that is only partly VLAN aware. The shaded symbols are VLAN aware. The empty ones are not.

With 802.1Q, frames are colored depending on the port on which they are received. For this method to work, all machines on a port must belong to the same VLAN, which reduces flexibility. For example, in Fig. 4-48, this property holds for all ports where an individual computer connects to a bridge, but not for the port where the hub connects to bridge *B2*.

Additionally, the bridge can use the higher-layer protocol to select the color. In this way, frames arriving on a port might be placed in different VLANs depending on whether they carry IP packets or PPP frames.

Other methods are possible, but they are not supported by 802.1Q. As one example, the MAC address can be used to select the VLAN color. This might be useful for frames coming in from a nearby 802.11 LAN in which laptops send frames via different ports as they move. One MAC address would then be mapped to a fixed VLAN regardless of which port it entered the LAN on.

As to the problem of frames longer than 1518 bytes, 802.1Q just raised the limit to 1522 bytes. Luckily, only VLAN-aware computers and switches must support these longer frames.

Now let us take a look at the 802.1Q frame format. It is shown in Fig. 4-49. The only change is the addition of a pair of 2-byte fields. The first one is the

VLAN protocol ID. It always has the value 0x8100. Since this number is greater than 1500, all Ethernet cards interpret it as a type rather than a length. What a legacy card does with such a frame is moot since such frames are not supposed to be sent to legacy cards.

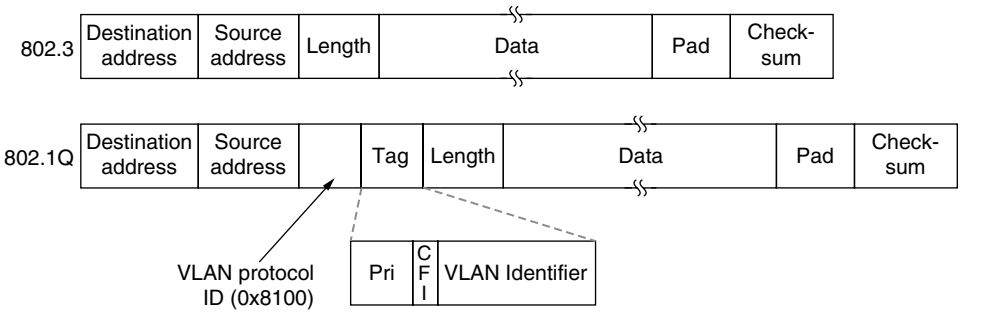

Figure 4-49. The 802.3 (legacy) and 802.1Q Ethernet frame formats.

The second 2-byte field contains three subfields. The main one is the *VLAN identifier*, occupying the low-order 12 bits. This is what the whole thing is about—the color of the VLAN to which the frame belongs. The 3-bit *Priority* field has nothing to do with VLANs at all, but since changing the Ethernet header is a once-in-a-decade event taking three years and featuring a hundred people, why not put in some other good things while you are at it? This field makes it possible to distinguish hard real-time traffic from soft real-time traffic from time-insensitive traffic in order to provide better quality of service over Ethernet. It is needed for voice over Ethernet (although in all fairness, IP has had a similar field for a quarter of a century and nobody ever used it).

The last field, *CFI* (*Canonical format indicator*), should have been called the *CEI* (*Corporate ego indicator*). It was originally intended to indicate the order of the bits in the MAC addresses (little-endian versus big-endian), but that use got lost in other controversies. Its presence now indicates that the payload contains a freeze-dried 802.5 frame that is hoping to find another 802.5 LAN at the destination while being carried by Ethernet in between. This whole arrangement, of course, has nothing whatsoever to do with VLANs. But standards' committee politics are not unlike regular politics: if you vote for my bit, I will vote for your bit.

As we mentioned above, when a tagged frame arrives at a VLAN-aware switch, the switch uses the VLAN identifier as an index into a table to find out which ports to send it on. But where does the table come from? If it is manually constructed, we are back to square zero: manual configuration of bridges. The beauty of the transparent bridge is that it is plug-and-play and does not require any manual configuration. It would be a terrible shame to lose that property. Fortunately, VLAN-aware bridges can also autoconfigure themselves based on observing the tags that come by. If a frame tagged as VLAN 4 comes in on port

3, apparently some machine on port 3 is on VLAN 4. The 802.1Q standard explains how to build the tables dynamically, mostly by referencing appropriate portions of the 802.1D standard.

Before leaving the subject of VLAN routing, it is worth making one last observation. Many people in the Internet and Ethernet worlds are fanatically in favor of connectionless networking and violently opposed to anything smacking of connections in the data link or network layers. Yet VLANs introduce something that is surprisingly similar to a connection. To use VLANs properly, each frame carries a new special identifier that is used as an index into a table inside the switch to look up where the frame is supposed to be sent. That is precisely what happens in connection-oriented networks. In connectionless networks, it is the destination address that is used for routing, not some kind of connection identifier. We will see more of this creeping connectionism in Chap. 5.

4.9 SUMMARY

Some networks have a single channel that is used for all communication. In these networks, the key design issue is the allocation of this channel among the competing stations wishing to use it. FDM and TDM are simple, efficient allocation schemes when the number of stations is small and fixed and the traffic is continuous. Both are widely used under these circumstances, for example, for dividing up the bandwidth on telephone trunks. However, when the number of stations is large and variable or the traffic is fairly bursty—the common case in computer networks—FDM and TDM are poor choices.

Numerous dynamic channel allocation algorithms have been devised. The ALOHA protocol, with and without slotting, is used in many derivatives in real systems, for example, cable modems and RFID. As an improvement when the state of the channel can be sensed, stations can avoid starting a transmission while another station is transmitting. This technique, carrier sensing, has led to a variety of CSMA protocols for LANs and MANs. It is the basis for classic Ethernet and 802.11 networks.

A class of protocols that eliminates contention altogether, or at least reduces it considerably, is well known. The bitmap protocol, topologies such as rings, and the binary countdown protocol completely eliminate contention. The tree walk protocol reduces it by dynamically dividing the stations into two disjoint groups of different sizes and allowing contention only within one group; ideally that group is chosen so that only one station is ready to send when it is permitted to do so.

Wireless LANs have the added problems that it is difficult to sense colliding transmissions, and that the coverage regions of stations may differ. In the dominant wireless LAN, IEEE 802.11, stations use CSMA/CA to mitigate the first problem by leaving small gaps to avoid collisions. The stations can also use the RTS/CTS protocol to combat hidden terminals that arise because of the second

problem. IEEE 802.11 is commonly used to connect laptops and other devices to wireless access points, but it can also be used between devices. Any of several physical layers can be used, including multichannel FDM with and without multiple antennas, and spread spectrum.

Like 802.11, RFID readers and tags use a random access protocol to communicate identifiers. Other wireless PANs and MANs have different designs. The Bluetooth system connects headsets and many kinds of peripherals to computers without wires. IEEE 802.16 provides a wide area wireless Internet data service for stationary and mobile computers. Both of these networks use a centralized, connection-oriented design in which the Bluetooth master and the WiMAX base station decide when each station may send or receive data. For 802.16, this design supports different quality of service for real-time traffic like telephone calls and interactive traffic like Web browsing. For Bluetooth, placing the complexity in the master leads to inexpensive slave devices.

Ethernet is the dominant form of wired LAN. Classic Ethernet used CSMA/CD for channel allocation on a yellow cable the size of a garden hose that snaked from machine to machine. The architecture has changed as speeds have risen from 10 Mbps to 10 Gbps and continue to climb. Now, point-to-point links such as twisted pair are attached to hubs and switches. With modern switches and full-duplex links, there is no contention on the links and the switch can forward frames between different ports in parallel.

With buildings full of LANs, a way is needed to interconnect them all. Plug-and-play bridges are used for this purpose. The bridges are built with a backward learning algorithm and a spanning tree algorithm. Since this functionality is built into modern switches, the terms ''bridge'' and ''switch'' are used interchangeably. To help with the management of bridged LANs, VLANs let the physical topology be divided into different logical topologies. The VLAN standard, IEEE 802.1Q, introduces a new format for Ethernet frames.

PROBLEMS

1. For this problem, use a formula from this chapter, but first state the formula. Frames arrive randomly at a 100-Mbps channel for transmission. If the channel is busy when a frame arrives, it waits its turn in a queue. Frame length is exponentially distributed with a mean of 10,000 bits/frame. For each of the following frame arrival rates, give the delay experienced by the average frame, including both queueing time and transmission time.

(a) 90 frames/sec.
(b) 900 frames/sec.
(c) 9000 frames/sec.

2. A group of N stations share a 56-kbps pure ALOHA channel. Each station outputs a 1000-bit frame on average once every 100 sec, even if the previous one has not yet been sent (e.g., the stations can buffer outgoing frames). What is the maximum value of N?

3. Consider the delay of pure ALOHA versus slotted ALOHA at low load. Which one is less? Explain your answer.

4. A large population of ALOHA users manages to generate 50 requests/sec, including both originals and retransmissions. Time is slotted in units of 40 msec.
(a) What is the chance of success on the first attempt?
(b) What is the probability of exactly k collisions and then a success?
(c) What is the expected number of transmission attempts needed?

5. In an infinite-population slotted ALOHA system, the mean number of slots a station waits between a collision and a retransmission is 4. Plot the delay versus throughput curve for this system.

6. What is the length of a contention slot in CSMA/CD for (a) a 2-km twin-lead cable (signal propagation speed is 82% of the signal propagation speed in vacuum)?, and (b) a 40-km multimode fiber optic cable (signal propagation speed is 65% of the signal propagation speed in vacuum)?

7. How long does a station, s, have to wait in the worst case before it can start transmitting its frame over a LAN that uses the basic bit-map protocol?

8. In the binary countdown protocol, explain how a lower-numbered station may be starved from sending a packet.

9. Sixteen stations, numbered 1 through 16, are contending for the use of a shared channel by using the adaptive tree walk protocol. If all the stations whose addresses are prime numbers suddenly become ready at once, how many bit slots are needed to resolve the contention?

10. Consider five wireless stations, A, B, C, D, and E. Station A can communicate with all other stations. B can communicate with A, C and E. C can communicate with A, B and D. D can communicate with A, C and E. E can communicate A, D and B.

(a) When A is sending to B, what other communications are possible?
(b) When B is sending to A, what other communications are possible?
(c) When B is sending to C, what other communications are possible?

11. Six stations, A through F, communicate using the MACA protocol. Is it possible for two transmissions to take place simultaneously? Explain your answer.

12. A seven-story office building has 15 adjacent offices per floor. Each office contains a wall socket for a terminal in the front wall, so the sockets form a rectangular grid in the vertical plane, with a separation of 4 m between sockets, both horizontally and vertically. Assuming that it is feasible to run a straight cable between any pair of sockets, horizontally, vertically, or diagonally, how many meters of cable are needed to connect all sockets using

(a) A star configuration with a single router in the middle?
(b) A classic 802.3 LAN?

13. What is the baud rate of classic 10-Mbps Ethernet?

14. Sketch the Manchester encoding on a classic Ethernet for the bit stream 0001110101.

15. A 1-km-long, 10-Mbps CSMA/CD LAN (not 802.3) has a propagation speed of 200 m/μsec. Repeaters are not allowed in this system. Data frames are 256 bits long, including 32 bits of header, checksum, and other overhead. The first bit slot after a successful transmission is reserved for the receiver to capture the channel in order to send a 32-bit acknowledgement frame. What is the effective data rate, excluding overhead, assuming that there are no collisions?

16. Two CSMA/CD stations are each trying to transmit long (multiframe) files. After each frame is sent, they contend for the channel, using the binary exponential backoff algorithm. What is the probability that the contention ends on round k, and what is the mean number of rounds per contention period?

17. An IP packet to be transmitted by Ethernet is 60 bytes long, including all its headers. If LLC is not in use, is padding needed in the Ethernet frame, and if so, how many bytes?

18. Ethernet frames must be at least 64 bytes long to ensure that the transmitter is still going in the event of a collision at the far end of the cable. Fast Ethernet has the same 64-byte minimum frame size but can get the bits out ten times faster. How is it possible to maintain the same minimum frame size?

19. Some books quote the maximum size of an Ethernet frame as 1522 bytes instead of 1500 bytes. Are they wrong? Explain your answer.

20. How many frames per second can gigabit Ethernet handle? Think carefully and take into account all the relevant cases. *Hint*: the fact that it is *gigabit* Ethernet matters.

21. Name two networks that allow frames to be packed back-to-back. Why is this feature worth having?

22. In Fig. 4-27, four stations, *A*, *B*, *C*, and *D*, are shown. Which of the last two stations do you think is closest to *A* and why?

23. Give an example to show that the RTS/CTS in the 802.11 protocol is a little different than in the MACA protocol.

24. A wireless LAN with one AP has 10 client stations. Four stations have data rates of 6 Mbps, four stations have data rates of 18 Mbps, and the last two stations have data rates of 54 Mbps. What is the data rate experienced by each station when all ten stations are sending data together, and

(a) TXOP is not used?
(b) TXOP is used?

25. Suppose that an 11-Mbps 802.11b LAN is transmitting 64-byte frames back-to-back over a radio channel with a bit error rate of 10^{-7}. How many frames per second will be damaged on average?

26. An 802.16 network has a channel width of 20 MHz. How many bits/sec can be sent to a subscriber station?

27. Give two reasons why networks might use an error-correcting code instead of error detection and retransmission.

28. List two ways in which WiMAX is similar to 802.11, and two ways in which it is different from 802.11.

29. From Fig. 4-34, we see that a Bluetooth device can be in two piconets at the same time. Is there any reason why one device cannot be the master in both of them at the same time?

30. What is the maximum size of the data field for a 3-slot Bluetooth frame at basic rate? Explain your answer.

31. Figure 4-24 shows several physical layer protocols. Which of these is closest to the Bluetooth physical layer protocol? What is the biggest difference between the two?

32. It is mentioned in Section 4.6.6 that the efficiency of a 1-slot frame with repetition encoding is about 13% at basic data rate. What will the efficiency be if a 5-slot frame with repetition encoding is used at basic data rate instead?

33. Beacon frames in the frequency hopping spread spectrum variant of 802.11 contain the dwell time. Do you think the analogous beacon frames in Bluetooth also contain the dwell time? Discuss your answer.

34. Suppose that there are 10 RFID tags around an RFID reader. What is the best value of Q? How likely is it that one tag responds with no collision in a given slot?

35. List some of the security concerns of an RFID system.

36. A switch designed for use with fast Ethernet has a backplane that can move 10 Gbps. How many frames/sec can it handle in the worst case?

37. Briefly describe the difference between store-and-forward and cut-through switches.

38. Consider the extended LAN connected using bridges *B1* and *B2* in Fig. 4-41(b). Suppose the hash tables in the two bridges are empty. List all ports on which a packet will be forwarded for the following sequence of data transmissions:

(a) *A* sends a packet to *C*.
(b) *E* sends a packet to *F*.
(c) *F* sends a packet to *E*.
(d) *G* sends a packet to *E*.
(e) *D* sends a packet to *A*.
(f) *B* sends a packet to *F*.

39. Store-and-forward switches have an advantage over cut-through switches with respect to damaged frames. Explain what it is.

40. It is mentioned in Section 4.8.3 that some bridges may not even be present in the spanning tree. Outline a scenario where a bridge may not be present in the spanning tree.

41. To make VLANs work, configuration tables are needed in the bridges. What if the VLANs of Fig. 4-47 used hubs rather than switches? Do the hubs need configuration tables, too? Why or why not?

42. In Fig. 4-48, the switch in the legacy end domain on the right is a VLAN-aware switch. Would it be possible to use a legacy switch there? If so, how would that work? If not, why not?

43. Write a program to simulate the behavior of the CSMA/CD protocol over Ethernet when there are *N* stations ready to transmit while a frame is being transmitted. Your program should report the times when each station successfully starts sending its frame. Assume that a clock tick occurs once every slot time (51.2 μsec) and a collision detection and sending of a jamming sequence takes one slot time. All frames are the maximum length allowed.