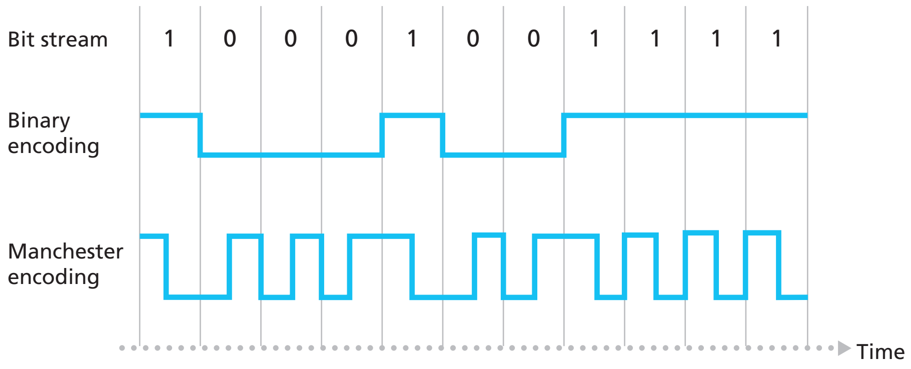

Figure 5.24 ♦ Manchester encoding

5.5.2 CSMA/CD: Ethernet's Multiple Access Protocol

Nodes in an Ethernet LAN are interconnected by a broadcast channel, so that when an adapter transmits a frame, all the adapters on the LAN receive the frame. As we mentioned in Section 5.3, Ethernet uses a CSMA/CD multiple access algorithm. Summarizing our discussion from Section 5.3, recall that CSMA/CD employs the following mechanisms:

1. An adapter may begin to transmit at any time; that is, no slots are used.
2. An adapter never transmits a frame when it senses that some other adapter is transmitting; that is, it uses carrier sensing.
3. A transmitting adapter aborts its transmission as soon as it detects that another adapter is also transmitting; that is, it uses collision detection.
4. Before attempting a retransmission, an adapter waits a random time that is typically small compared with the time to transmit a frame.

These mechanisms give CSMA/CD much better performance than slotted ALOHA in a LAN environment. In fact, if the maximum propagation delay between stations is very small, the efficiency of CSMA/CD can approach 100 percent. But note that the second and third mechanisms listed above require each Ethernet adapter to be able to (1) sense when some other adapter is transmitting, and (2) detect a collision while it is transmitting. Ethernet adapters perform these two tasks by measuring voltage levels before and during transmission.

Each adapter runs the CSMA/CD protocol without explicit coordination with the other adapters on the Ethernet. Within a specific adapter, the CSMA/CD protocol works as follows:

1. The adapter obtains a network-layer PDU from its parent node, prepares an Ethernet frame, and puts the frame in an adapter buffer.

2. If instead the adapter senses that the channel is idle (that is, there is no signal energy from the channel entering the adapter), it starts to transmit the frame. If the adapter senses that the channel is busy, it waits until it senses no signal energy (plus 96 bit times) and then starts to transmit the frame.
3. While transmitting, the adapter monitors for the presence of signal energy coming from other adapters. If the adapter transmits the entire frame without detecting signal energy from other adapters, the adapter is finished with the frame.
4. If the adapter detects signal energy from other adapters while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.
5. After aborting (that is, transmitting the jam signal), the adapter enters an **exponential backoff** phase. Specifically, when transmitting a given frame, after experiencing the nth collision in a row for this frame, the adapter chooses a value for K at random from $\{0,1,2, \ldots, 2^m - 1\}$ where $m := \min(n,10)$. The adapter then waits $K \cdot 512$ bit times and then returns to Step 2.

A few comments about the CSMA/CD protocol are certainly in order. The purpose of the jam signal is to make sure that all other transmitting adapters become aware of the collision. Let's look at an example. Suppose adapter A begins to transmit a frame, and just before A's signal reaches adapter B, adapter B begins to transmit. So B will have transmitted only a few bits when it aborts its transmission. These few bits will indeed propagate to A, but they may not constitute enough energy for A to detect the collision. To make sure that A detects the collision (so that it too can also abort), B transmits the 48-bit jam signal.

Next consider the exponential backoff algorithm. The first thing to notice here is that a bit time (that is, the time to transmit a single bit) is very short; for a 10 Mbps Ethernet, a bit time is 0.1 microsecond. Now let's look at an example. Suppose that an adapter attempts for the first time to transmit a frame, and while transmitting it detects a collision. The adapter then chooses $K = 0$ with probability 0.5 and chooses $K = 1$ with probability 0.5. If the adapter chooses $K = 0$, then it immediately jumps to Step 2 after transmitting the jam signal. If the adapter chooses $K = 1$, it waits 51.2 microseconds before returning to Step 2. After a second collision, K is chosen with equal probability from $\{0,1,2,3\}$. After three collisions, K is chosen with equal probability from $\{0,1,2,3,4,5,6,7\}$. After ten or more collisions, K is chosen with equal probability from $\{0,1,2, \ldots, 1023\}$. Thus the size of the sets from which K is chosen grows exponentially with the number of collisions (until $n = 10$); it is for this reason that Ethernet's backoff algorithm is referred to as "exponential backoff."

The Ethernet standard imposes limits on the distance between any two nodes. These limits ensure that if adapter A chooses a lower value of K than all the other adapters involved in a collision, then adapter A will be able to transmit its frame without experiencing a new collision. We will explore this property in more detail in the homework problems.

Why use exponential backoff? Why not, for example, select K from {0,1,2,3,4,5, 6,7} after every collision? The reason is that when an adapter experiences its first collision, it has no idea how many adapters are involved in the collision. If there are only a small number of colliding adapters, it makes sense to choose K from a small set of small values. On the other hand, if many adapters are involved in the collision, it makes sense to choose K from a larger, more dispersed set of values (why?). By increasing the size of the set after each collision, the adapter appropriately adapts to these different scenarios.

We also note here that each time an adapter prepares a new frame for transmission, it runs the CSMA/CD algorithm presented above. In particular, the adapter does not take into account any collisions that may have occurred in the recent past. So it is possible that an adapter with a new frame will immediately be able to sneak in a successful transmission while several other adapters are in the exponential backoff state.

Ethernet Efficiency

When only one node has a frame to send, the node can transmit at the full rate of the Ethernet technology (either 10 Mbps, 100 Mbps, or 1 Gbps). However, if many nodes have frames to transmit, the effective transmission rate of the channel can be much less. We define the **efficiency of Ethernet** to be the long-run fraction of time during which frames are being transmitted on the channel without collisions when there is a large number of active nodes, with each node having a large number of frames to send. In order to present a closed-form approximation of the efficiency of Ethernet, let t_{prop} denote the maximum time it takes signal energy to propagate between any two adapters. Let t_{trans} be the time to transmit a maximum-size Ethernet frame (approximately 1.2 msecs for a 10 Mbps Ethernet). A derivation of the efficiency of Ethernet is beyond the scope of this book (see [Lam 1980] and [Bertsekas 1991]). Here we simply state the following approximation:

$$\text{Efficiency} = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

We see from this formula that as t_{prop} approaches 0, the efficiency approaches 1. This matches our intuition that if the propagation delay is zero, colliding nodes will abort immediately without wasting the channel. Also, as t_{trans} becomes very large, efficiency approaches 1. This is also intuitive because when a frame grabs the channel, it will hold on to the channel for a very long time; thus the channel will be doing productive work most of the time.

5.5.3 Ethernet Technologies

The most common Ethernet technologies today are 10Base2, which uses thin coaxial cable in a bus topology and has a transmission rate of 10 Mbps; 10BaseT, which