

Chapter 12

Modems

Most personal computers today contain a modem — a device which lets it communicate with other computers through a telephone line.

In the previous chapter, we mentioned that the telephone set connects to the subscriber loop, a twisted-pair balanced line which connects to the telephone network, through a hybrid, which interfaces the two one-way or *simplex* connections (to the earphone, and from the microphone) to the two-way or duplex telephone line which leads to the telephone company's central office or CO. This line carries voice simultaneously in both directions.

At the CO, another hybrid splits the two-wire two-way subscriber loop back into two one-directional connections. The outgoing signal is converted from analog to digital, and stays digital all the way until it gets to the central office at the far end, at which point it is again converted into analog and sent to the called telephone set. We showed a block diagram of this system in Fig. 11-1.

Partially because of the characteristics of the subscriber loop, and partially because of the analog-to-digital-to-analog conversion process, the frequency response of the overall telephone network goes from about 300 to about 3400 Hz. This is fine for audio, but doom on pure digital signals — it means that digital pulses, which tend to consist of square edges with plenty of harmonics, have no chance of getting through the system without some help.

The Modem

With the above background, you can now understand the function of the modem. The modem (whose two main components are a *modulator* and a *demodulator* — that is where it gets its name) takes the digital data and disguises it to look like sound. In simple terms, it takes the digital data and modulates it onto an audio carrier at one end of the connection, and then demodulates it at the other end back into digital data. Temporarily converting it into audio keeps the telephone circuits happy.

It's useful to trace the development of modems to understand the techniques they use.

The Bell 103 modem

Up until the mid 1970's, only telephone companies ("telcos") were allowed to connect equipment to phone lines. Early modems were therefore identified by the

model numbers assigned to them by the telcos, usually part of the Bell System. The model 103 was the first commonly-available modem, useful for carrying data up to 300 bits per second (bps). It modulated the digital data onto an FM carrier, with a different carrier frequency used in each direction.

Because the digital data only has two values — a 0 or a 1 — the FM carrier, rather than continuously deviating back and forth over some range, jumps back and forth rapidly between two frequencies — a high frequency (which represents the digit 1) and a low frequency (which represents a 0). Since the carrier jumps (or "shifts") back and forth between them, the modulation method is called FSK or *frequency shift keying* rather than just FM. You can consider it as plain FM with a square-wave modulation; since the square-wave modulation has a fundamental and many harmonics, there are multiple sidebands on both sides of the carrier. But the higher harmonics of a square wave are progressively smaller, so the sidebands farther from the carrier frequency drop off rapidly.

Fig. 12-1 illustrates what FSK looks like — a few cycles of a high frequency for a 1, and a few cycles of a low frequency for a 0. (The figure also shows a problem that must be avoided in normal use. Note how the signal has sharp edges between adjacent bits; these edges cause clicks and interference, and also increase the bandwidth. In an actual FSK signal, the signal has to maintain *phase continuity*; that is, it has to smoothly and continuously blend from one frequency into the other.)

The 103 modem was *full duplex*, meaning that it allowed signals to go in both directions at the same time. (A mini-detour at this point: *Half-duplex* transmission also goes in both directions, but only one direction at any one time — the two modems in the connection must take turns sending to each other. *Simplex*, on the other hand, allows transmission in only one direction.)

To avoid interference, full duplex in the 103 modem required two separate carrier frequencies, one in each direction. The modem which placed the call (called the *originate* modem) sent out a carrier frequency of 1170

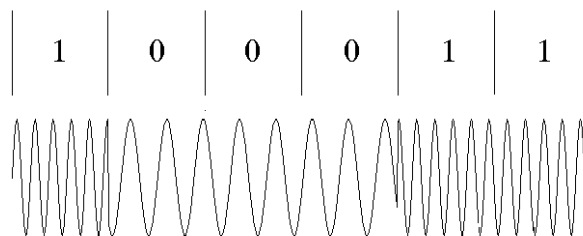


Fig. 12-1. Frequency-shift keying (FSK)

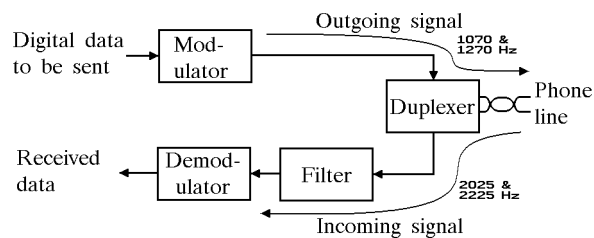


Fig. 12-2. Bell 103 modem block diagram

Hz with 100 Hz deviation up and down; the carrier frequency thus shifted back and forth between 1270 Hz (1170 plus 100) for a 1 and 1070 (1170 minus 100) for a 0. (Note that, since only the digits 0 and 1 are possible in the digital data, only the frequencies of 1070 and 1270 Hz are allowed for the carrier. In other words, the carrier frequency never actually became the 1170 Hz center frequency.) The *answer* modem at the other end used the same 100 Hz deviation, but sent out a center frequency of 2125 Hz; its output was therefore 2225 Hz for a 1 and 2025 Hz for a zero. Fig. 12-2 thus shows an originate modem — 1070 and 1270 Hz are sent out, while 2025 and 2225 Hz are received.

The sidebands depend on the actual digital data being sent, but typically extend about 600-800 Hz on each side of the carrier. The originate modem's signal therefore occupies the range from about 300 to about 1800 Hz, while the answer modem's sidebands range between about 1500 to about 3000 Hz. You'll note that the sidebands interfere with each other in the middle range of 1500-1800 Hz between the two carriers. Fortunately, the sidebands farther from the carrier drop off rapidly, which helps to reduce the interference somewhat. Still, the signal leaving each modem does interfere somewhat with the incoming signal.

Unfortunately, there is another effect which also comes into play. Since each carrier may have to travel through a lot of circuitry on its way from one modem to the other, the outgoing signal from the modem is much stronger than the incoming signal, typically by 20 db or more. The strong outgoing signal therefore interferes with the weak incoming signal, which the modem's demodulator is trying to receive. The effect is somewhat like trying to hear someone far away while someone else is yelling into your ear. The fact that the incoming and outgoing sidebands overlap to some extent makes things even worse.

As shown in Fig. 12-2, the modem has two circuits which try to solve the problem — a filter, and a *duplexer* (which is just another name for the hybrid.) The filter passes mainly the frequencies to be received, and

tries to eliminate the outgoing signal (as well as other noise picked up on the phone line) from getting into the demodulator. But because the outgoing and incoming sidebands interfere in the middle frequencies, the filter cannot completely eliminate the outgoing signal without also removing some of the incoming signal.

That's where the duplexer/hybrid comes in. We have already mentioned that a hybrid acts as an interface between one two-way circuit (which typically requires two wires) to two one-way circuits (which typically have two wires each, so together they need four wires.) The name duplexer comes from its ability to combine two simplex signals into one duplex signal.

But it does more than that. It also acts as a one-way valve, letting the outgoing signal go from the modulator to the phone line, letting the incoming signal go from the phone line to the filter and demodulator, but preventing the outgoing modulated signal from taking the shortcut down to the filter.

Telephone company people call their duplexer a hybrid, primarily because the circuit inside a telephone set uses a combination of a transformer, capacitor, and resistor (and a few other components). Fig. 12-3a shows the telephone hybrid used in many older telephones. Although Fig. 12-3a shows two transformers, actually there is only one transformer because all five of the windings are wound on one common core. An incoming signal from the telephone line flows through the four upper windings (which all act as one big primary), and then through the transformer to the earphone. You thus hear the person at the other end of the line.

When you speak, however, the outgoing signal from the microphone comes in at the center of the top four windings, and splits in half — roughly half of the microphone audio signal goes right to the phone line, while the other half goes left to resistor R and capacitor C. Because the currents go in opposite directions, they cancel in the transformer, and very little of the signal gets sent to the earphone. You hear your own voice, but not very loud.

In order for the microphone current to split in half, with equal parts going right and left, the impedance on the right and the impedance on the left must be equal. Resistor R and capacitor C are thus chosen so that their series impedance is approximately equal to the impedance of the telephone line; they are sometimes called a *balancing network*. If R and C were chosen exactly right, the mike currents going right and left would exactly balance each other, and you would hear none of your voice in the earphone at all. This is hard to achieve, because every telephone line is slightly different and requires slightly different values of R and C. In any case, complete cancelling of your voice from your

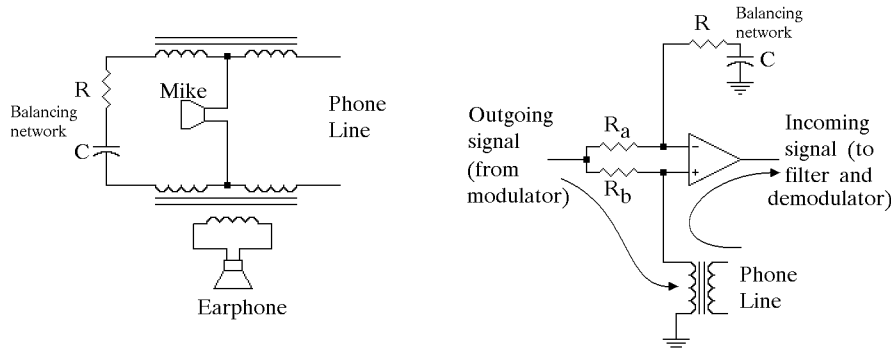


Fig. 12-3. The hybrid / duplexer circuit

earphone is not desired — most people like to hear a bit of their own voice to convince themselves that the telephone is working! So, in an actual telephone, R and C are intentionally slightly off to produce a slight amount of what is called *sidetone* — the feedthrough of the mike signal into the earphone.

Fig. 12-3b shows a more solid-state version of the hybrid (and there are other versions too.) As before, resistor R and capacitor C are a balancing network whose impedance should be equal to the phone line. The outgoing signal from the modulator, coming in at the left, is split in two. Part of it goes through resistor R_a to the balancing network, while the other part goes through resistor R_b (and a transformer) to the phone line. Resistors R_a and R_b are equal; if the impedance of the balancing network is the same as the impedance of the telephone line (as seen through the transformer), the voltages at the + and - inputs to the op amp will be equal. An op amp amplifies the *difference* between its two inputs, but there is no difference between them and so the amplifier amplifies nothing. So none (or, at least, very little) of the outgoing signal gets from the modulator into the demodulator.

The incoming signal from the phone line, on the other hand, goes directly through the transformer to the + input of the op amp, where it is amplified and sent to the filter and then the demodulator.

Because you want sidetone in a normal phone, the hybrid circuit inside the telephone set is intentionally not completely balanced; that is, the RC balancing circuit is intentionally slightly misadjusted. In a modem, on the other hand, the situation is reversed — you must eliminate the sidetone as much as possible to avoid confusing the demodulator. Still, it is not possible to manufacture a modem with the precisely exact values of R and C for every possible situation.

Returning to the 103 modem, it is now clear that some of the outgoing signal from the modulator will get back into the demodulator and cause interference. Com-

bined with any external noise and interference coming in from the telephone line, this makes it harder for the demodulator to correctly identify the incoming frequency.

If you look at Fig. 12-1, you will note that the difference between the high frequency and the low frequency in that FSK signal is fairly obvious. But this figure exaggerates the difference because it uses a 2-to-1 difference in frequencies. The actual differ-

ence between the two frequencies in a 103 modem was less than 20% for the low tone, and less than 10% for the high tone. If Fig. 12-1 showed such a small difference, you would have a hard time telling the difference by eye. The demodulator has the same problem; it therefore needs several cycles of a tone before it can reliably tell whether it has a 1 or a 0, and this effect limits the bit-per-second rate that the modem can handle.

As a rough rule, these modems needed about two or three milliseconds of a signal before they could correctly identify the signal; this therefore set the shortest bit-time at 2 or 3 ms. In this way, we can see that the fastest modem speed was somewhere in the range of 333 to 500 bps. Since the two nearest common bps rates used at that time were 300 or 600 bps, these modems could clearly work at 300 bps or slower, but not at 600 bps.

One way to speed up operation is to make the difference between the low and high frequencies greater, so that fewer cycles of a signal would be needed to tell them apart. This led to the Bell 202 modem, which used 500 Hz deviation on a 1700 Hz carrier; the FSK therefore shifted the frequency to 1200 Hz (for a 1) and 2200 Hz (for a 0). This speeded up operation to 1200 bps, but used up the full bandwidth of the phone line. As a result, the modem could only be used in one direction at a time; it was therefore a half-duplex device.

DPSK Modems

Once it became legal to connect your own modem to a telephone line, modem design changed from a Bell monopoly to a free-for-all, with many companies using incompatible methods to increase speeds. Gradually, a set of standards was developed by the CCITT (The Consultative Committee for International Telephony and Telegraphy, which is now called the International Telecommunications Union – Telecommunications Sector or ITU-T.) The progression of modem speeds is

shown in the listing of standards (the letters *bis* and *ter* refer to the second and third revision of a standard):

- V.21: 300 bps
- V.22: 1200 bps
- V.22bis: 2400 bps
- V.32: 9600 bps
- V.32bis: 14,400 bps
- V.32ter: 19,200 bps
- V.34: 28,800 bps
- V.34Q: 33,600 bps
- V.90: "56K", actually about 52K bps

Modems up to 1200 bps used the older, simpler techniques we discussed in the previous section. Significant speed increases came later, with the development of *differential phase-shift keying* or DPSK.

Instead of changing the frequency, DPSK modems change the phase of the carrier. Fig. 12-4 shows a simple example. Suppose we agree that

- 0 is a 0° phase shift, and
- 1 is a 180° phase shift.

We start off with a carrier, and for each bit change the phase (from the previous bit) by 0° for a zero, and 180° for a one. This will give us the waveform in Fig. 12-4. Since a 0° phase change is no change at all, we only see the phase change (by 180°) at the beginning of each 1. It's clear what PSK means in this context; the D in DPSK means that the phase changes not from some fixed reference, but from the previous bit; it is the *difference* between consecutive bits.

Since this system has two possible phase changes, it is called a DPSK-2 system.

Let's now consider a slightly more complex example, called DPSK-4 because there are four possible angles, defined as follows:

- 00 = 270°
- 01 = 180°
- 10 = 90°
- 11 = 0°

To use this, we will divide our data (100011 in the prior examples) into groups of two bits called *dibits* — 10, 00, and 11 — and then send each dibit on its own portion of the signal; that portion will be called a *symbol*. We encode each group of two bits onto its symbol by changing the phase angle of that symbol according to the above table. For these six bits we have:

- 10 = 90° , which means go forward 90° , or skip forward $1/4$ of a cycle
- 00 = 270° means jump ahead 270° . This is equivalent to -90° , which tells us to go back 90°
- 11 = 0° , means no change in phase.

The result is the signal in Fig. 12-5, where we have shown the -90° phase change at the beginning of the 00

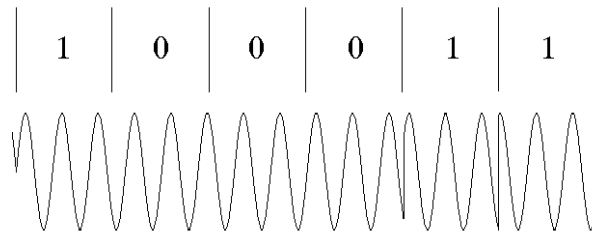


Fig. 12-4. DPSK-2 with 2 phase angles

dibit, and also noted that the phase does not change for the 11 dibit. (If you look carefully, you can also see the $+90^\circ$ phase change at the beginning of the 10 symbol, as well as another change at the far right, after the 11 dibit's symbol.)

Figs. 12-4 and 12-5 bring up an important concept. Suppose the bandwidth of a circuit is such that we can only change the signal's phase 600 times per second; i.e., we can only send 600 symbols per second. If we used the simple scheme of Fig. 12-4, then each symbol would only carry 1 bit, and therefore we would also be limited to 600 bits per second. But using the scheme of Fig 12-5 lets us pack *two* bits into each symbol; we can therefore send 1200 bits per second, instead of 600. We have thus doubled the amount of data that can be sent through the phone line in each second.

Let us now define a very misunderstood term: *Baud Rate*. The baud rate is the number of symbols (or signal changes) per second. It is misunderstood because many people think that "baud rate" means "bit-per-second rate", which is true only in very simple cases.

Look at either Fig. 12-1 or Fig. 12-4. In both of these cases, each symbol carries at one bit. In this very simple case, the number of symbols per second is also equal to the number of bits per second. So the baud rate is equal to the bps rate. But look at Fig. 12-5 — here

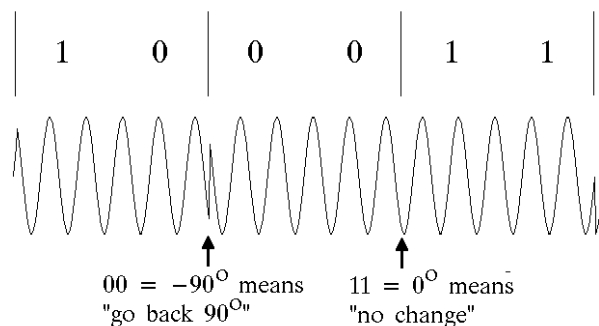


Fig. 12-5. DPSK-4 with four phase angles

each symbol carries two bits. Hence the bit-per-second or bps rate is actually *twice* the baud rate!

Because of the limited bandwidth of the common telephone line, modems are limited to about 3000 symbols per second or 3000 baud. What is commonly advertised as, for example, a “9600 baud modem” actually generates a 2400-baud signal which crams 4 bits on each symbol to give an effective rate of 9600 bps. The same idea holds for faster modems.

The term baud rate is misused so often that it has become common to refer to the bps rate as the baud rate. But you should remember the difference because, every now and then, some stickler for accuracy (like a college professor) will try trip you up on the difference.

The information we have tabulated so far as

$$\begin{aligned} 00 &= 270^\circ \\ 01 &= 180^\circ \\ 10 &= 90^\circ \\ 11 &= 0^\circ \end{aligned}$$

can be shown in a slightly different form called a *constellation diagram*, Fig. 12-6. Imagine that each of the four different angles is a vector whose angle indicates the phase angle, and whose length represents the amplitude of the corresponding symbol (all the vectors in this example would be the same length, because the amplitude of the DPSK signal is constant — so far). To reduce the clutter, though, we simply put a dot where the end of the vector should be, rather than drawing a full arrow. For example, the dot at the top represents a vector at 90° (angles are measured from the right axis, just as in normal math.) Each dot is labelled with its corresponding bit pattern, so the one at 90° is labelled 10. The constellation diagram is merely a more compact way of listing the same information as the previous table.

The reason for using constellation diagrams is that they let us show how faster modems really work. For example, a fairly simple 9600 “baud” modem (which is

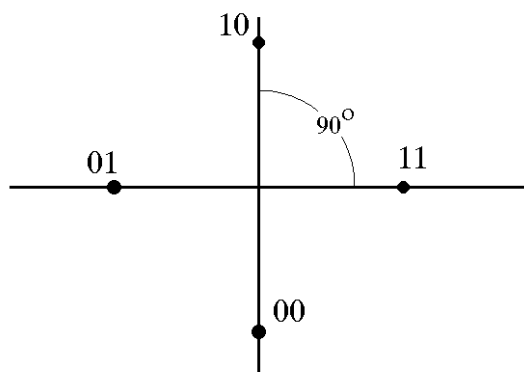


Fig. 12-6. A DPSK-4 constellation diagram

really a 2400-baud modem, working at 9600 bps because each symbol carries four bits) has the constellation diagram shown in Fig. 12-7.

QAM

Quadrature Amplitude Modulation or QAM is the basis of all modern high-speed modems. Fig. 12-7 is a very simple example of QAM, which is a combination of differential phase shift keying (DPSK) and amplitude shift keying (ASK).

As we see in the figure, the constellation chart of this 9600-baud modem has sixteen dots. Each dot carries four bits (since 2^4 is 16), so 2400 symbols per second (2400 baud) will carry 4×2400 , or 9600 bps.

Altogether, the sixteen dots appear at 12 different phase angles. But note also that the dots are not all equal distances from the center of the chart. Remember that each dot simply represents the tip of a vector; the direction of the vector shows the phase angle of the symbol, while its length shows its amplitude. For example, the dots that carry 0011 and 0001 are both at an angle of 45° , but 0011 has a small amplitude, while 0001 has a large amplitude.

This method therefore combines phase shift keying with amplitude shift keying to provide the 16 different symbols needed to permit a fairly high bit-per-second rate, while still keeping the baud rate — the actual number of different symbols in each second — within limits.

But there is a price to pay — with two different phase shifts as in Fig. 12-4, the difference between any two successive symbols is either 0 or 180 degrees, and it is fairly easy to tell the difference between them. With the 12 different phases of Fig. 12-7, and three different amplitudes, there is only about a 30-degree difference between the various phases. Add some noise, and possibly even phase differences introduced by the telephone line, and you can see that a modem’s demodulator can easily make mistakes.

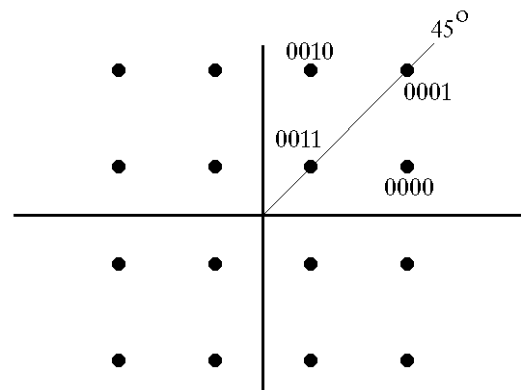


Fig. 12-7. 9600-bps DPSK-16 modem

Now imagine that we want to extend this idea to faster modems. For example, suppose that we just double the speed to 19,200 bps. Still being limited to 2400 baud, we must pack 8 bits on each symbol. Now there must be 2^8 , or 256 different dots in the constellation chart. If we use four different amplitudes, there will be 64 different phase angles. You can see that this would greatly increase the modem's error rate, probably making it unusable. So something else has to be done.



Before continuing, just a quick detour to explain the word *quadrature* in quadrature amplitude modulation.

In math, quadrature means "at right angles". But the angles in DPSK are not 90° ; so why is it called quadrature?

In math, a sine wave and a cosine wave look the same, but they are shifted 90° apart; they really are in quadrature. If you add a sine wave to a cosine wave of the same frequency, you get a new signal which looks just like a sine wave, but is offset by some new phase angle. The modem's modulator produces the 12 different phase angles in Fig. 12-7 by adding various amounts of the quadrature sine and cosine signals together.



Modern Modems

So how do modern 28K or 33K modems achieve their remarkable speed and performance? By using a number of tricks. Some of these involve fairly complex math, so we will only give a brief qualitative description, without any of the mathematical detail.

Negotiation and line probing

When one modem calls another, the answering modem sends back a tone to tell the caller that it is there. The two modems now start to negotiate with each other to decide what speed they will use, what kind of modulation to use, what carrier frequency and amplitude to use, and so on. If the connection is very bad, the modems will agree on a *fallback speed*; a slower speed which allows error-free communications even with a bad circuit. In fact, 28.8K bps and faster modems can even fall back to different speeds in each direction.

Echo Suppression

Remember that the old FSK modems used different frequencies in each direction; this meant that only half the bandwidth was usable in each direction. To achieve high speeds, modern modems need the full 300–3500 Hz bandwidth of the line each way. Each modem therefore has to make sure that the outgoing and incoming

signals are kept completely apart — none of the locally-generated signal should enter the local demodulator and cause errors.

The modulator therefore sends out a short test pulse, and the demodulator listens for it. Ideally, none of the test pulse should get through the duplexer, and none should be echoed back from the phone line, so the demodulator should hear nothing. In reality, though, the demodulator will hear a small signal, probably with some delay caused by travel through the phone system. The modem then computes an "equal, but opposite" signal which, when added to the signal heard by the demodulator, will cancel it out. In all further communication, the modem will automatically use this to compute an echo cancellation signal, sent to the demodulator to cancel out any echo or signal slipping the wrong way through the duplexer.

Scrambling

Even though the digital data being sent by the two modems may be asynchronous, the connection between the two modems is actually synchronous. We will describe the differences in a future chapter; for now let us simply say that this improves the speed somewhat, because synchronous connections do not require constant start and stop bits, and they also have better error detection.

But synchronous communications requires that the sending and receiving modems remain synchronized with each other. That is, each modem must have a clock oscillator, and the two clocks must run at exactly the same frequency. Rather than send a clock signal through the phone line (which would take up bandwidth and slow down the data), the receiving modem has to synchronize its clock from the data itself.

But some combinations of data may prevent that. For instance, look at the DPSK-4 scheme in Fig. 12-6. If the data consisted of a long string of ones, the signal would consist of a long carrier with a constant 0° phase shift. The receiving modem would have a difficult time telling where one symbol ends and another begins.

To avoid this problem, modern modems scramble the incoming data in a known way so as to avoid these troublesome combinations. (Needless to say, the receiving modem must then also unscramble the bits.)

Trellis coding

Even after all the previous tricks, the receiving modem is still likely to make errors when the constellation contains many points. Hence a form of error correction called *trellis coding* is used. To keep things as simple as possible (a difficult task!) let's confine ourselves to a 9600-bps modem.

Fig. 12-7 showed the 16-dot constellation of a simple 9600-bps modem. Actually, most modern 9600-bps modems use Trellis coding with the 32-dot constellation of Fig. 12-8. When you look at this, you note that the dots are even closer together than those of Fig. 12-7; hence you probably suspect that there must be even more errors than before. This would be true — except that Trellis modulation at the sending modem, and Viterbi detection at the receiving modem, work together to greatly reduce errors.

First, we note that the telephone line can carry 2400 baud (symbols per second), and this is a 9600-bps modem. Hence we only need to put four bits on any one symbol, and therefore only need 2^4 or 16 different symbols; having 32 is overkill.

What the sending modem does, however, is to add a fifth error-correction bit to every group of four bits (in a sort of scrambling operation that mixes in some of the previous data, and actually changes two of the four desired bits as well); hence each symbol actually encodes five bits, although only four of these are actual data; this explains the 2^5 dots. The resulting five bits now depend not just on the current bits, but also on the data that was sent earlier.

So let's assume that you are sitting inside the sending modem, monitoring what is going on. You know that at some particular instant, the modem just output the particular symbol that corresponds to dot A in Fig. 12-8. The modem now gets the next group of four bits, and its Trellis coding circuit computes a new five-bit code from it. But the coding circuit must follow very specific rules; given a particular set of four bits (and a particular history of past data) it must generate a very specific five-bit output code. In other words, since there are only 16 possible combinations of new data, there can only be 16 possible numbers it generates. In still other words, although five bits can make up 32 different numbers, only 16 of those 32 can actually come out of

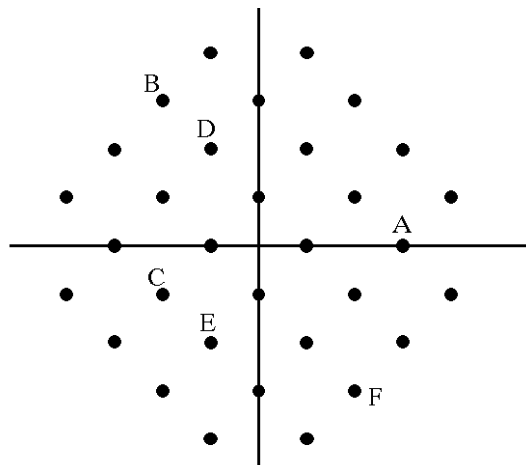


Fig. 12-8. Trellis constellation for a 9600-bps modem

the coder; the other 16 are illegal at this instant (they could be generated other times, however.)

What this means is this: there are 32 possible symbols in Fig. 12-8, but if you have just output symbol A, 16 of those 32 symbols are illegal for the *next* symbol. For instance, symbols B and C might be legal, but D and E might not.

So what does all this mean? It means that, although the dots in Fig. 12-8 are much closer than those of Fig. 12-7, if you cross out all those that are illegal at any particular instant, you find that the remaining ones are roughly the same distance apart as those in Fig. 12-7. In other words, the presence of 32 dots is no worse than the 16 dots of Fig. 12-7.

So far, it looks like Fig. 12-8 is no worse than 12-7, but no better either. But that changes with the next trick:

Viterbi detection

Let's suppose that the sending modem sent out a batch of data and ended up on symbol A (all of which the receiving modem received correctly). It next sends out symbols B and E in that order, but suppose the receiving modem makes an error and thinks the sending modem sent out D and E (see Fig. 12-8.)

Since all the previous data was received correctly, the receiving modem knows which 16 out of the 32 symbols would have been legal after A. It doesn't know which one was actually sent, but it does know that it couldn't have been D, because D is not legal after A (at this particular time.)

Think of the correct sequence A-B-E as being a road through a maze — the maze is the constellation diagram — where only certain dots in the diagram have roads connecting them at any particular instant. The receiving modem now says to itself: I've been told that the sender route was A-D-E, but there is no road there. *What is the closest actual road that starts at A, passes close to D, and ends close to E?*

The receiving modem knows all the rules of the game; it can determine which roads go where and when. So it makes a list of all legal roads that start at A, pass close to D, and end close to E. That list might include roads A-B-E, as well as A-F-E and A-D-C. It then goes through a fairly simple computation to determine which of these is closest to the A-D-E route that it thinks the other modem sent, and comes up with A-B-E as being the most likely.

This process sounds pretty chancy (and it is — since it is based on the rules of probability), but actually works quite well. Think of probability this way: Suppose you take a quick measurement of, say, the length of a room. If you're fairly sloppy about it, your measurement may be wrong by several inches. But if you

measure the room several times and average your readings, the average will generally be fairly close to the actual value because your various errors will tend to cancel themselves out.

The Trellis code / Viterbi decoding scheme relies on the same principle. Modern modems generally keep track of up to four or five symbols in a row, and use the closest legal path that matches the last four or five received symbols. Since a path involving four or five consecutive symbols is fairly complex, the number of legal paths that will lie close is fairly small; hence the Viterbi decoder can pick out the correct path with a fairly good chance of success. Just in case of error, however, most modern modems apply error correction (and possibly data compression) to the received data before they pass it on to the communications program.

28,800 bps Modems

28,800 bps modems use similar schemes, but with much greater complexity. For example, a 28,800 bps V.34 modem uses an actual line baud rate of 3200 symbols per second (just about the absolute maximum that the phone line can handle). To send 28,800 bps, it needs to pack $28,800/3200$ or 9 bits into every symbol. This would give it a minimum of 2^9 or 512 dots on its constellation chart. If we drew this chart, the dots would be so close together that they would be almost impossible to tell apart. In reality, the V.34 modem uses Trellis coding and Viterbi detection with 960 dots.

56K modems

Claude Shannon, a famous Bell Laboratories mathematician who invented the entire field of Information Theory, calculated in 1948 the absolute maximum theoretical number of bits that can be sent through a communications circuit in one second as

$$\text{max bps} = BW \log_2 (S/N + 1)$$

where BW is the bandwidth, and S/N is the signal-to-noise ratio — the ratio between the signal power and the noise power.

Mathematicians call the expression \log_2 in this equation the “logarithm base 2.” Calculating it is somewhat difficult, so for a good approximation let’s think of it simply as the number of bits needed to represent the value of $(S/N + 1)$ in binary. For example, today’s telephone lines have a bandwidth of about 3000 Hz, and a signal-to-noise ratio of about 1000 to 1. We need about 10 bits to represent the number 1001 (which is $1000 + 1$), so the theoretical maximum number of bits per second in a telephone connection would be about 3000 times 10, or roughly 30,000 bps.

Modem speeds of 28K or 33K bps therefore seem to be about the fastest that the normal voice-oriented POTS telephone network can handle with today’s technology. So how are 56K modems possible?

Faster speeds require a little help from the telephone company. Fig. 11-1 in Chapter 11 showed how the POTS network handles a voice call with a microphone and earphone at each end, connected through hybrids to the subscriber loop, and then using a pair of analog-to-digital and digital-to-analog converters at each end of the phone network itself. When a pair of 33,600 bps (or slower) modems talk to each other, the configuration is almost the same, except that a modulator replaces the microphone, and a demodulator replaces the earphone. As a result, digital data is first converted to audio; then an a-to-d converter converts it to digital for transmission through the network (which itself uses 64K bps data transmission); then a d-to-a converter converts it back into audio, and finally a demodulator converts it back to digital. A rather roundabout process, which results in less than optimum speed.

56K modems are slightly different. They are not designed to work in pairs; in fact, if two 56K modems call each other, they will only work at 28,800 bps, just like any other V.34 modem. Instead, they are designed to allow an Internet customer to connect to his *Internet Service Provider* (ISP). The 56K modem forms an asymmetric system — fast one way (“downstream”, from the ISP to you), slower the other (“upstream”, from you to the ISP.) In the upstream direction, the 56K modem works at normal 28,800 bps speed, using Trellis/Viterbi coding, just as we described earlier. The difference is in the downstream direction.

The idea is that the initial a-to-d conversion at the sender’s central office creates the most problems; so the ISP bypasses that step by sending digital data directly into the network. In order to provide 56K service, the ISP must install a digital line from their computer back to the telephone company’s central office. Rather than convert the digital downstream data into audio, and then have the telephone company convert it back into digital form, the ISP sends it down as pure digital data. It stays digital all the way back to *your* central office, at which point it is converted to analog for the last portion of its trip, down to your modem.

By bypassing the most error-prone portion of a typical telephone connection, this process allows speeds greater than the 30K bps or so that Shannon’s equation would otherwise predict. Because of some technical limitations, the maximum speed is only about 52–53K bps, not the 56K speed that the name “56K modem” would imply, but that is still a worthwhile increase over 28.8 or 33.4K bps.