

Chapter 15

Multiplexing and TDM

When Alexander Graham Bell invented the telephone in the 1870's, it was an accident — he was searching for a way to multiplex telegraph signals, but discovered the telephone instead.

Multiplexing is a technique for sending several signals simultaneously through one channel. Bell was trying to find a way to increase the capacity of telegraph systems by sending several messages at the same time through the same cable, but the technology of the time was not quite up to the job.

Today, we have a number of multiplexing techniques which we shall explore in this chapter. The most useful technique for digital systems is TDM or *Time Division Multiplexing*, but we shall examine a few related topics first.

FDM: Frequency Division Multiplexing

Without calling it that, we have already discussed FDM in previous chapters. The whole idea of radio broadcasting — using carriers of different frequencies to carry different signals, all at the same time — is nothing but frequency division multiplexing. Each signal occupies a different range of frequencies; all of these signals are then mixed up and carried through the air to your antenna; the receiver then uses tuned filters to separate the signals and recover the one we want. *Frequency division multiplexing* merely takes a large range of frequencies, divides that range into smaller sections, assigns each signal that smaller range of frequencies, and then combines (“multiplexes”) them together.

FDM can also be used through wires, of course. Alexander Graham Bell was experimenting with FDM when he invented the telephone; his idea was to send the telegraph dots and dashes through the wire as musical tones of different frequencies, which could then be separated at the receiver with filters.

Eventually, the telephone company itself became a major user of frequency division multiplexing over wires. It soon became obvious that it was uneconomical to devote a separate wire to carry each conversation, especially between cities. So they developed their “carrier” system, where each voice signal was modulated onto a different carrier; the carriers were then combined together onto one cable. Their first system used plain AM, and the carrier frequencies were multiples of 8 kHz. For example, one voice signal would be modulated onto an 8000 Hz carrier, whose sidebands would extend from just above 4 kHz to just below 12 kHz (since the

audio went to just under 4 kHz). The next voice signal, modulated onto a 16 kHz carrier, would occupy the frequency range from about 12 kHz to about 20 kHz, and so on.

This telephone company FDM method has now been superseded with newer methods, but cable TV still uses FDM. TV signals, each occupying 6 MHz of spectrum, are combined onto one coax cable and carried into your home. (This method too is about to be superseded with digital technology, though.)

WDM: Wavelength Division Multiplexing

Remember that frequency and wavelength are related by the equation

$$\text{wavelength} = \frac{\text{velocity}}{\text{frequency}}$$

Thus for every carrier's frequency there is a corresponding wavelength (assuming you know the velocity at which the signal travels in whatever medium you are considering.) WDM or *wavelength division multiplexing* — that is, using a different wavelength for each signal — is therefore the same as frequency division multiplexing, where a different frequency is used for each signal.

In optical fibers, though, the color of the light is generally described in terms of its wavelength, not its frequency. So using different colors for different signals — that is, sending several different light beams through a fiber at the same time — is referred to by the name WDM rather than FDM. (Using the term “color” is a bit misleading, since “color” implies something that the eye can see, whereas most practical fiber-optic communication systems use invisible infrared light.)

WDM is an extremely useful technique with fiber optics, because it greatly increases the amount of information that can be sent through a single fiber. The bandwidth over a single light beam is primarily limited by the dispersion in the fiber; using two or more different color light beams allows use of the full bandwidth for each different beam. As of 1996, for example, several systems have been demonstrated that use 50 or so light beams, each carrying 20 billion bits per second, to carry a total of 1 trillion (10^{12}) bits per second of data through one fiber.

Spread Spectrum

Spread spectrum, already discussed in Chapter 10, is a bit different from FDM. Instead of assigning a different frequency range to each station, as is done with commercial broadcasting, spread spectrum transmitters can coexist on the same frequencies as long as they use dif-



Fig. 15-1. Left/Right sampling and TDM in FM stereo

ferent spreading codes — different chips for their ONES and ZEROS.

Under the name CDMA or *Code Division Multiple Access*, spread spectrum is used by several companies in cellular and PCS (*Personal Communication System*) phones.

PAM: Pulse Amplitude Modulation

Aside from FDM and WDM, most other multiplexing methods involve some sort of *sampling*. Sampling involves providing small portions of a wave, called *samples*, with enough detail to allow the receiver to fill in the missing parts.

We have seen sampling as it is used with pulse code modulation (PCM) in Chapter 14, and we looked at how often and how accurately we must take those samples.

Let's look at a somewhat simpler example. Most FM broadcast stations transmit in stereo. That is, they send out two distinct channels of sound — a left channel and a right channel, intended for a left speaker and a right speaker. These two channels carry different signals, but they must be combined into one transmitted signal. They do it by sampling the two channels at a 38 kHz rate, and sending the samples alternately. Fig. 15-1 shows a simplified picture. At the transmitter, a switch rapidly switches back and forth between the left and right channels; at exactly the same time, a switch in the receiver switches rapidly back and forth between the two speakers. The two switches are exactly synchronized, so they are up at the same time, and down at the same time. The signal from the left mike therefore goes only to the left speaker, and the signal from the right mike only to the right speaker. Each of the two signals is broken up — it continuously goes on-off-on-off... — but this happens so fast that our brain and ears think it is continuously on. It is similar to a light which flickers so fast that it looks as though it is continuously lit.

The audio bandwidth of FM stations typically extends to 15 kHz. The Nyquist Sampling Theorem would then require a minimum sampling rate of 30 kHz, but the stations use 38 kHz because they must carry an additional signal to synchronize the switches, and this increases the bandwidth.

Fig. 15-2 shows some typical waveforms. Suppose the top two waveforms show typical left and right channel signals. Each of them is then sampled at a 38 kHz

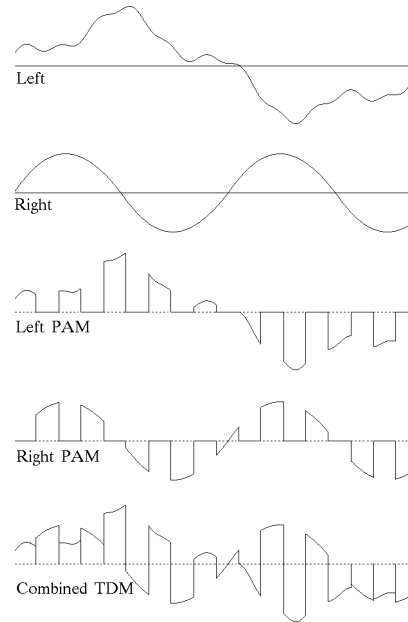


Fig. 15-2. TDM as used for FM stereo

rate, resulting in series of pulses, as shown in the third and fourth waveforms. You can think of these as thin slices of the original waveform, but taken at alternate times. Since each of these signals consists of pulses whose amplitude matches the corresponding signal, we call this *Pulse Amplitude Modulation* or PAM.

Finally, the bottom waveform in Fig. 15-2 shows how the two PAM signals are sandwiched together into a single signal called TDM.



In Fig. 15-2, the tops of the pulses match the shape of the signal. This is not really necessary — in most PAM systems, the tops of the pulses are flat and horizontal, because they come from a sample-and-hold circuit which grabs the voltage level at one particular instant and keeps it constant during the pulse.

The pulses are also relatively wide — as wide as the space between them. This is also not necessary — the pulses could be much narrower than the space between them. This might allow us to slip narrow pulses from additional channels into the space between them, thereby combining more than just two channels into one signal.

There is one more item to note. The pulses in Fig. 15-2 have a problem: some are positive and some are negative, which makes them difficult to send through some circuits. That's no problem in FM stereo, but

sometimes it is better to make all the pulses the same polarity by adding a fixed dc voltage to the original analog signal; this changes it from pure ac to pulsating dc.



We should point out one other fact — although we are dealing with pulses here, the height of the pulses is being expressed as an analog voltage. There is no digital or binary coding here. Hence PAM is an analog technique, even though the pulses make it look digital — a sort of analog-digital mongrel.

TDM: Time Division Multiplexing

Now look at Fig. 15-2, where we have both a left and a right channel, being combined to give a stereo signal. A good description of this system would be *Time Division Multiplexing* (TDM) of two PAM signals.

Note the similarity of Time Division Multiplexing (TDM) to Frequency Division Multiplexing or FDM.

- In FDM, a frequency range is divided into sections, each of which then carries a separate signal.
- In TDM, it is *time* that is divided into sections called *time slots*, and each time slot carries a separate signal.

In this particular case, we have two such signals being multiplexed, so that their PAM pulses alternate. But if the PAM pulses were narrower, there would be room between them to insert more pulses; it would then be possible to multiplex more than just two signals.

PWM: Pulse Width Modulation

We will return to TDM shortly, but let's just mention two other "modulations" that are somewhat related to PAM.

Rather than modulate the amplitude of each pulse, we can change the *width* of the pulses; this results in *Pulse Width Modulation* or PWM, which is shown in

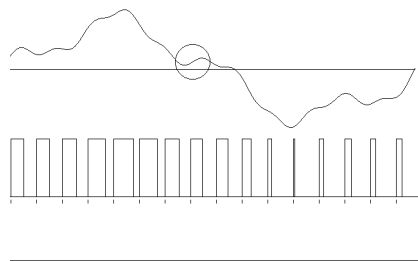


Fig. 15-3. Pulse Width Modulation

Fig. 15-3. A pulse is wide when the amplitude of the original signal is high, and narrow when the amplitude is low. (Note, by the way, that the sampling rate in Fig. 15-3 is much too low. The Nyquist sampling theorem states that there should be more than two samples taken during the fastest cycle of the waveform. We have circled what looks like a fairly small/fast cycle, and there aren't two complete samples in that small interval.)

As you can see from the small "tick" marks under the pulses, the leading edges of the pulses are evenly spaced; it is the trailing edges that move back and forth to vary the width of each pulse in our example. There are variations of PWM which move only the leading edge, or which move both edges to vary the width.

PWM has some advantages over PAM. Since all the pulses are the same height, PWM can be sent through digital circuits which would otherwise ignore or alter pulse heights. A PWM signal can be fairly easily converted back to an analog signal by passing it through an "integrator" circuit; that is, a circuit which measures the area under each pulse and converts that area to a fixed voltage. It can also be easily analyzed by digital circuitry, which can measure the width of a pulse fairly accurately by timing it.

Nevertheless, PWM is still only an analog-digital hybrid. Any kind of distortion which slightly changes the width of a pulse will distort the signal. PWM therefore has fairly limited uses in communications. One fairly common use is in high-power amplifiers or power supplies. Since the voltage in a PWM signal is either at zero or at full height, there are no in-between states. When such a signal is amplified by a transistor, the transistor is either fully off (when the current through it is zero) or fully on (when the voltage across it is very close to zero.) The power lost in a transistor is the product of current times voltage; if one or the other of those two is zero or close to zero at all times, their product is small, and the power dissipated in the transistor will be small. This allows the amplifier to provide a high power output signal without itself losing much power in the process.

PFM: Pulse Frequency Modulation

Pulse Frequency Modulation is another analog-digital hybrid; here the frequency of the pulses depends on the analog signal's voltage, as shown in Fig. 15-4.

PFM is very similar to FM, where the frequency of a carrier also depends on the analog signal being transmitted, but instead of a sinusoidal carrier, we have rectangular pulses. PFM is sometimes used in instrumentation and measurements, but its use in communications is usually associated with FM. For example, there are some FM detector circuits which convert FM to PFM,

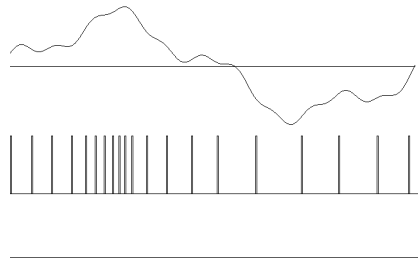


Fig. 15-4. Pulse Frequency Modulation

and then use digital circuits to measure the frequency and convert it to an analog signal.

Purely digital TDM

PAM, PWM, and PFM have been mixtures of analog and digital techniques. The value of an analog signal affected either the amplitude, width, or frequency of pulses, in a continuous or analog fashion. By this we mean that the amplitude, width, or frequency could take on a continuous set of values; there could be an infinite variety of amplitudes, widths, or frequencies. The problem with these approaches is that small circuit errors, which might produce slight changes in the amplitude or width or frequency of a pulse, cause errors in the signal.

On the other hand, PCM or Pulse Code Modulation, discussed in Chapter 14, is a purely digital method. Like the combination analog-digital methods discussed in this chapter, PCM takes samples of the analog signal, but these samples are then encoded as binary numbers. The ZEROS and ONES of these binary numbers are, in most computer systems, represented by two voltage levels: one that we call a *low*, and another one we call a *high*. The low is often close to zero volts, while the high is often somewhere between +2 and +5 volts. The beauty of this scheme is that the precise voltage of a high or low is not important, so that slight imperfections in a circuit which might change, for instance, +5 volts into +3 volts, make no difference because a high is a high. But changing a high into a low, or vice versa, is catastrophic, so if the change is large enough to change +5 to 0 volts, then all Hades breaks loose. Fortunately, with careful design, we can avoid most such huge errors.

PCM is thus ideal for long-distance transmission. And it can easily be combined with TDM to make a purely digital system, and there are two flavors of such TDM — synchronous TDM and statistical TDM.

Synchronous TDM

With synchronous TDM, data from multiple sources are multiplexed in order — synchronized to a clock of some sort.

Let's consider a simple example. Suppose we have three PCM sources, each of which outputs its data in four-bit chunks called *nybbles*.

- Source S1 outputs 0000 0001 0010
- Source S2 outputs 1111 1110 1100
- Source S3 outputs 0101 0110 0101

A synchronous TDM *multiplexer* (the device which would take the three separate bit streams and combine them into one, and often abbreviated as *mux*) would then take one nybble from each source in turn, in the order

S1 S2 S3 S1 S2 S3 S1 S2 S3 ...

and output this:

0000 1111 0101 0001 1110 0110 0010 1100 0101 ...

Actually, the bits would not be separated, as in the above line — they would appear as

000011110101000111100110001011000101...

At the other end of the connection, a *demultiplexer* would “demux” the data — separate the bits into the three separate sets of data.

We would need to take care of some details to make this work:

- Since the multiplexer takes three sets of inputs and combines them into one, its output must operate faster than each of the inputs. In this example, if each of the three inputs operates at 1000 bits per second (bps), the output must be able to run at 3000 bps to keep up.
- Since the three sets of bits are all intermixed, the demultiplexer will need just a bit of help to separate the bits and steer them to the right output. Because of the repetitive pattern of S1 S2 S3 and so on, it can count the bits to figure out where they go, but it needs to know what is at the very beginning.

Think of this as a freight train which carries the output from three factories that produce sugar, salt, and cereal. The freight cars are intermixed, so that freight cars 1, 4, 7, 10, and so on always carry sugar; freight cars 2, 5, 8, 11 and so on always carry salt, and freight cars 3, 6, 9, 12, and so on always carry cereal. The freight cars could be marked to show what is inside, but that would be inefficient and not necessary. Because the freight cars are always in the same order, the demultiplexer (the freight

yard which must separate the freight cars at the end) can separate them.

Now here is the distinguishing characteristic of synchronous TDM:

- If one of the sources has nothing to send, it still gets its time slot, but that time slot will be either empty or carry garbage.

In terms of the freight train, if one of the factories goes on strike, the railroad still has a signed and paid-up contract to supply the right number of freight cars, and so every third freight car of the freight train belongs to that factory, whether it's full or empty.

From the point of view of the factory managers, this system has some advantages and disadvantages. Each factory knows ahead of time exactly how many freight cars will be available. As they produce their product, they know that there will always be a freight car available to carry it. On the other hand, if there is a problem with production, they still have to pay the transportation costs even if the freight cars travel empty. A further problem is that they can't temporarily increase factory output, because there is no additional capacity for putting more freight cars on the train — all the slots are taken.

From the point of view of the railroad management, this system also has some advantages and disadvantages. They have a guaranteed source of income; if a factory has trouble producing, that's not the railroad's problem since they get paid anyway for the empty freight cars.

On the other hand, they can't sell what they don't have. Suppose the railroad can handle 300 freight cars per day, and they contract for 100 cars to each factory. Even if production in a factory drops, they can't sell the empty freight cars to someone else. In essence, they would like to do what airlines do — sell the 300 seats in a plane to 350 passengers, in the hope that 50 of them won't show up.

That's where statistical TDM comes into the picture.

Statistical TDM

In synchronous TDM, it often happens that time slots are empty. Still, the carrier has to provide the capacity anyway, just in case it is needed. This results in a certain amount of waste.

In statistical TDM, on the other hand, the carrier estimates the average amount of incoming data, and provides only enough outgoing capacity for that amount (plus perhaps just a bit more.) In terms of our freight train analogy, suppose each of the three factories has a maximum output of 100 freight cars a day, but typically produces only 75 per day. This adds up to an average

day of 225 freight cars, so the railroad might install only enough capacity for perhaps 250 freight cars.

This introduces two new problems:

- Since the number of freight cars varies, each car now needs to be identified so the receiving demultiplexer knows which car contains what product or data. In other words, each shipment needs to have an address to indicate where it is to go. This adds some overhead to the process.
- Although an average day might only produce 225 freight cars, there will be times when each factory produces its maximum, for a total of 300 cars. Since the railroad can only handle 250 cars a day, what to do with the excess?

Data communications carriers handle this excess load problem in one of two ways:

- Sometimes they will provide buffers to temporarily store the excess data until it can be transmitted later.
- Other times, they may simply throw the extra data away. This happens particularly in extremely high-speed systems, where even a short delay might require the storage of huge amounts of data.

Throwing away a few freight cars of merchandise sounds terrible. But in data communications, it's not always such a bad idea. Because of the possibility of errors, senders usually keep a copy of their outgoing data anyway until the receiver has acknowledged receiving it correctly, so it's not too much of a problem to simply send it again later if it was discarded by the carrier.

Buffer performance

A *buffer* is a temporary storage area which holds data. Statistical TDM multiplexers often contain a buffer to hold incoming data that is coming in faster than it can be sent out.

The buffer size is important — too big a buffer wastes space, too small a buffer and you lose data.

Let's look at a simple case. Suppose data is coming into a multiplexer from ten sources. Each source can transmit at a rate of 100 bits per second, but on the average only five sources are active at any one time. The maximum incoming data rate would then be 1000 bps, but the average is only 500 bps. Let's also assume that the multiplexer output is limited to 500 bps as well, and prepare the following table:

Time (sec)	Active sources	Incoming bps	Bits in buffer from prev. sec.	Output bps	Bits stored into buffer
1	4	400	0	400	0
2	6	600	0	500	100
3	8	800	100	500	400
4	9	900	400	500	800
5	3	300	800	500	600
6	8	800	600	500	900
7	2	200	900	500	600
8	6	600	600	500	700
9	2	200	700	500	400
10	2	200	400	500	100

This table points out two facts:

- If the buffer were too small, data might have to be discarded.
- If the multiplexer output rate is exactly equal to the average input rate, then if you ever fall behind, you might never catch up. We see that right at the beginning of the table in the first second: the output rate there was only 400 bps, so the multiplexer fell behind the average by 100 bits, and it never caught up.

If we were to increase the multiplexer output rate to 600 bps to give it a slight safety factor, the table would change to

Time (sec)	Active sources	Incoming bps	Bits in buffer from prev. sec.	Output bps	Bits stored into buffer
1	4	400	0	400	0
2	6	600	0	600	0
3	8	800	0	600	200
4	9	900	200	600	500
5	3	300	500	600	200
6	8	800	200	600	400
7	2	200	400	600	0
8	6	600	0	600	0
9	2	200	0	300	0
10	2	200	0	200	0

Not only does the multiplexer have a chance to catch up now, but we also do not need as large a buffer to avoid data loss.

Synchronous vs. Statistical TDM

Both synchronous and statistical TDM are in use today. Each method has its advantages for certain kinds of data.

When voice or video are digitized, they generally produce a fairly constant stream of data. Moreover, there is a kind of rush to the data — if you are speaking to someone, you want your words or pictures to arrive quickly and in the right order. You need a fairly constant data path, without the delays that might be introduced by lack of capacity or buffering. Synchronous TDM is the ideal transmission method for this kind of data.

Computer data, on the other hand, tends to be *bursty*. That is, it arrives in bursts of activity, separated by periods of inactivity. Providing a synchronous TDM path for such data is inefficient — you have to provide for the high speed data during periods of intense activity, but then wind up with large numbers of unused time slots between those bursts. Statistical TDM, on the other hand, allows you to use the unused time to send someone else's data.

But it is inefficient to have to maintain two kinds of networks — a synchronous TDM system for voice and video, and a statistical TDM system for everything else. There has therefore recently been a move to send voice and video data over statistical TDM along with the traditional computer data, helped by three techniques:

- Compressing the voice or video data to use fewer bits. It is then less likely that it would need to be delayed or buffered.
- If the data carrier needs to discard data because of overflow problems, marking voice or video data as being more important than computer data lets them choose what to discard.
- Sending the data in smaller batches so it can easily be handled by hardware, rather than needing extensive processing by computers along the way. These “batches” of data are called frames.

Frames

A *frame* is essentially a group of related bits which travel together through the physical medium (cable, fiber, or wireless). For example, the eight bits in an asynchronous ASCII character, plus its start and stop bits, constitute a ten-bit frame. We would say that the eight data bits are *framed* by the start and stop bits. Some would also say that the start and stop bits are *framing bits*.

The precise structure of a frame depends on the system. Frames can be anywhere from 10 bits long to thousands of bits long, and we will see some specific frame

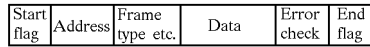


Fig. 15-5. Structure of a typical frame

formats as we continue to future chapters. But the basic frame structure is like Fig. 15-5 (although a frame need not contain all of these elements). It consists of

- A start flag. This is some special bit or bit pattern which lets us recognize the beginning of the frame.
- One or more addresses, which identify the source and destination of the frame.
- A frame type and some other indicators which define the purpose of the frame.
- The actual data carried by the frame.
- An error check portion which detects errors in the preceding part of the frame. This is often a CRC (see Appendix C), and is usually called FCS or *Frame Check Sequence*.
- An end flag so we can recognize the end of the frame.

Every frame has some sort of a start flag, but some of the other parts might be missing. For example, the asynchronous ASCII character sent on a RS-232 circuit contains a start flag (the start bit), data bits (the ASCII character), possibly an error check bit (the parity bit, if used), and an end flag (the stop bit.) There is no address, and no frame type.

Little frames sometimes become the building blocks of bigger frames. For example, a typical message in BISYNC, a system developed by IBM in 1964, might look like Fig. 15-6. Each of the elements in this big frame is in turn one or more ASCII characters — i.e., little frames. If you look at Table 3-3 in Chapter 3, you will see that some of the elements of the BISYNC frame are just plain ASCII characters:

- SYN is ASCII character 0010110, a synchronization character that lets the receiver synchronize its clock to the signal.
- SOH is ASCII character 0000001, and means Start of Header
- The Header is usually an address
- STX is ASCII character 0000010, and means Start of Text
- The text data is just some text, expressed as a bunch of ASCII characters
- ETX is ASCII character 0000011, and means End of Text
- The Error check is one or two bytes which provide a way of checking the accuracy of the previous part of the frame.

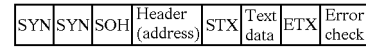


Fig. 15-6. Typical BISYNC message frame format

Other frames might use the ASCII characters ACK for Acknowledge, NAK for Not Acknowledge, etc.

Packets

In the last section, we defined frames as “a group of related bits which travel together through the physical medium (cable, fiber, or wireless).”

Like frames, packets are also groups of related bits, but they may not necessarily travel together through the physical medium. A full explanation will have to wait for later, so here is a brief explanation:

When a program or device prepares a message to be transmitted from one place to another, it may not have any idea how that will be done. Much like a boss who says to his secretary, “Miss Jones, take a letter,” the program or device doesn’t want to be bothered with the details of how that message will get to its destination.

Like the secretary, the subservient program may then have the task of addressing the message to its final destination. It knows where the message is to go, and probably even wants to make sure it gets there without error. But it has no idea how the physical circuitry (the company’s mailroom) plans to actually deliver the message. Depending on the size of the message, this program may package the message into one or more packets, and send them “downstairs” to be sent out.

But the physical circuits — the mail room — may still find the packet inconvenient for the particular delivery method, and may repackage it (and perhaps even subdivide it) into frames, which eventually travel through the physical medium which may be a wire, an optical fiber, or some wireless system such as a satellite or microwave link.

So the message trickles down through the “chain of command”: boss, secretary, mail room, truck. At the receiving side, it percolates back up the chain: truck, mail room, secretary, boss.

We will stop at this point; much more detail will appear in Chapter 20, when we discuss the OSI Model and other data communications topics.