DS1/DS3 and E1/E3 Framing

P. Michael Henderson
mike@michael-henderson.us

April 15, 2002

As communications becomes ever more important to both individuals and corporations, the speed and reliability of the lines that provide access to the Internet becomes critical. There's a great deal of talk about new forms of access, especially access based on 100 Mbps and 1 Gbps Ethernet but, for now, most corporate access is based on T1 and T3 in North America (NA) and E1 and E3 in Europe. This paper describes how data is carried in these T1/T3 and E1/E3 lines, specifically the framing and multiplexing formats used in these circuits.

Note that I use the terms T1 and T3 but use DS1/DS3 in the title of the paper. I don't know if there's a real, official definition of the difference between Tx and DSx (where x is an integer) but I distinguish between the two as follows: a Tx signal (as T1 or T3) is the actual electrical signal on the transmission media. The DSx signal (as DS1 or DS3) is the digital signal to be transmitted over the T-carrier. The difference between the two comes about because there must be a certain ones-density on the physical transmission media to maintain synchronization, which leads to various types of encoding to do this. I've included a discussion of ones-density and the various line codes in Appendix A – I didn't include it in the main part of the paper because I felt it would distract from my primary purpose, which is to discuss framing and multiplexing.

The European standard doesn't have this nomenclature problem – it's E1 and E3 all the time.

Since my main interest is the framing and multiplexing of the digital data, I'll primarily use the terms DS1 and DS3 for the North American technology, except when I refer to the actual signal on the transmission media.

I wrote this paper because I couldn't find a good single source of information about DS1/DS3 and E1/E3 framing and multiplexing. There are a few books with the words "T1/T3 Networking" in their title but most seem to have been written by network technicians whose primary interest in how to install and service T1/T3 lines and equipment. Discussion of the actual frame formats is generally superficial, ignoring how things are actually handled in the frame or multiplex structure. There's almost nothing with the terms E1/E3 in the title.

Alternately, there are the ANSI and ITU standards which contain the detailed data, albeit in abbreviated form and without any explanation of why things were done the way they were. Additionally, the reader often has to refer to a number of standards documents to understand how the whole system works.

What was needed, I felt, was a document which explained things from an engineer's point of view, in sufficient detail that you could gain a relatively full understanding of the system.

Writing a paper of this nature consumes quite a bit of time and energy, both for the research and the actual writing. As feedback to me, to let me know that people are actually reading this paper, I'd appreciate if you would send me an e-mail (at the address above) with nothing but a subject line of "Framing." If you'd like to make comments, corrections, or offer suggestions, that would be appreciated of course, but is not required.

In writing the paper, I have to assume that you have a certain background in communications. For example, I do not discuss how speech is digitized into a 64 Kbps data stream, known as a DS0 in North America. I assume that you know about protocols and understand the need for framing, and probably a number of other things that I can't think of right now. If you find that I assume too much knowledge, please let me know and I'll add additional explanation in future revisions.

I begin by examining the North American digital hierarchy, starting with the framing of the DS1 signal, including the usage of the framing bits and the so-called "robbed bits." Next, I move to the framing of the DS2 signal because, although the DS2 signal is no longer used in the network, it is an intermediate multiplexing level required for multiplexing into a DS3 frame. An M13 multiplexer chip creates multiple DS2 frames within the chip before creating the DS3 frame. This is essentially the only place in the network where a DS2 frame is still used. I finish the North American section with a discussion of the DS3 frame itself, describing the framing and overhead bits.

Next, I examine the European digital hierarchy, starting with the E1 signal, then moving to the E2 intermediate level, and finally to the E3 level. The European E1, E2, and E3 framing is more regular than the North American framing and is much easier to understand. This simplicity shows up in this paper, with the European section being much shorter than the North American section.

To be fair, I must note that the North American techniques were developed earlier than the European techniques, and the Europeans learned from the missteps of the Americans. Additionally, the European techniques were developed in standards bodies, since the telephone systems of the various European countries had to interoperate. In these open forums the relative merits of various techniques were argued and agreed. In the US, the AT&T network architects were able to implement without much concern for interoperability since they were the first to develop digital networks and were only concerned with interworking with themselves.

But enough preface – let's go examine the framing and multiplexing techniques.

North American Digital Hierarchy

DS1 Framing

The DS1 framing is the first level in the digital hierarchy above the actual voice channels. Speech is coded according to the ITU G.711 specification, using μ-law encoding in North America, to give a 64Kbps bit stream per channel. Specifically, the speech is coded eight bits per sample, with 8,000 samples taken per second.

The DS1 signal takes twenty-four of these DS0 channels and multiplexes them into a single bit stream, giving a rate of 1.536 Mbps. This multiplexing is octet (8 bits) oriented. That is, eight bits for each channel are contiguous. See Figure 1.

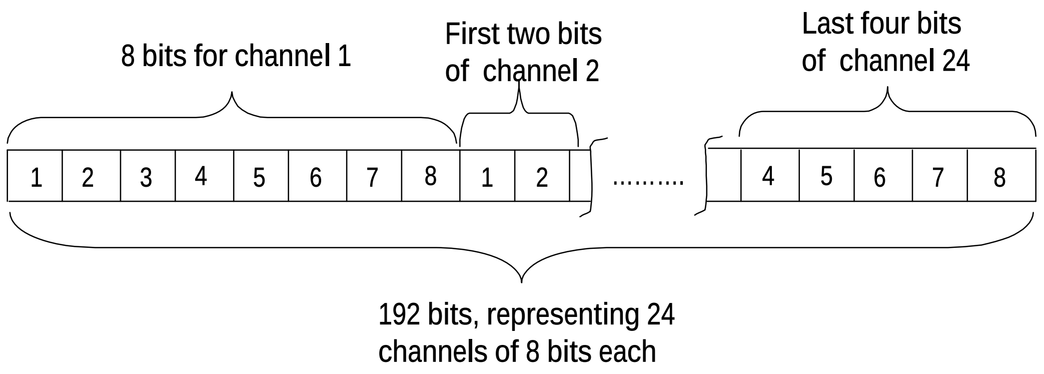
8 bits for channel 1
First two bits of channel 2
Last four bits of channel 24
1 2 3 4 5 6 7 8 1 2
....
4 5 6 7 8
192 bits, representing 24 channels of 8 bits each

Figure 1: The multiplexing of 24 voice channels into a DS1 channel.

But if the DS1 consisted of just a continuous stream of bits from the 24 voice channels, we wouldn't know how to identify which byte belonged to which channel, or for that matter, where the byte boundaries are. Put yourself into the position of the receiver. If you were receiving the string of bits, how would you determine what's what? To find the start of this "frame" of 24 voice channels, we need some type of framing.

In HDLC, we use a special octet, the string of bits 0111 1110, as a framing character. Inside the frame, we suppress occurrences of this string with a technique known as "zero bit insertion." And since HDLC frames can be variable length, we use another of these special characters as an ending flag to show where the frame ends. See Figure 2.

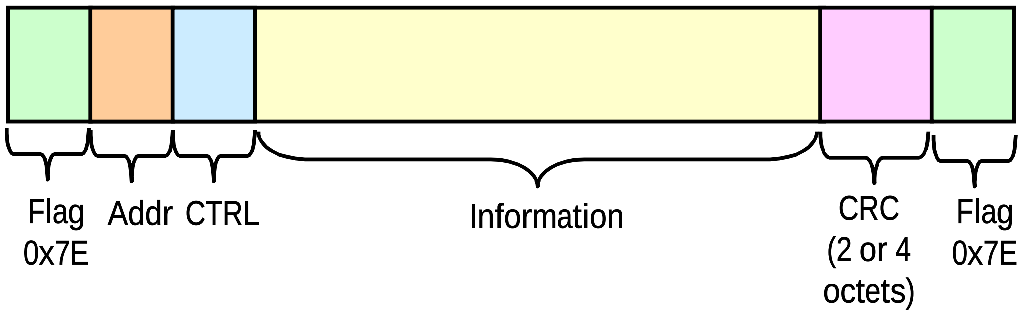

Figure 2: HDLC framing, showing the framing character, also known as a flag.

Today, bandwidth is cheap and we use bits freely. However, back in the late '50's and early '60's when digital telephony was first being designed, bandwidth was expensive and scarce. Because of this, the system architects of the DS1 frame did not use a byte as a framing character – they used a bit. See Figure 3 which shows the basic 193-bit DS1 frame. Since this 193-bit frame occurs 8,000 times per second, the nominal rate of a T1 line is 1.544 Mbps[1]. You might wonder how a bit can be used as a framing character, and that's what we'll explore next.

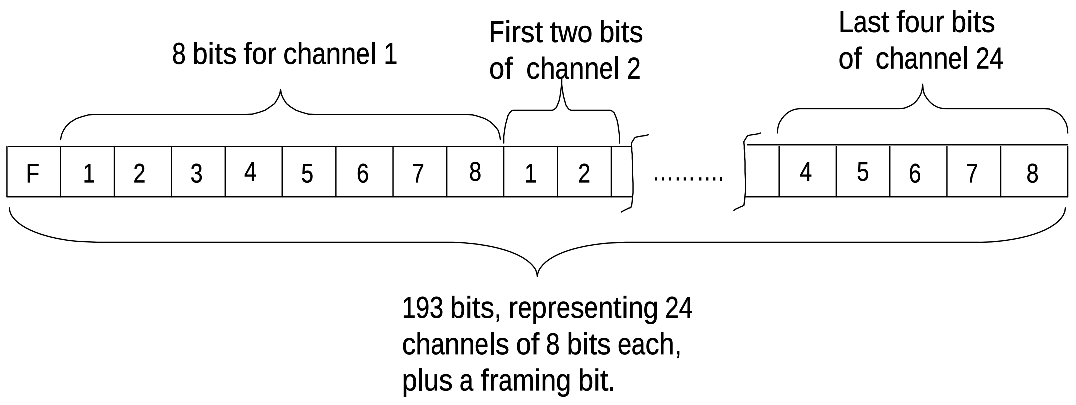

Figure 3: DS1 framing showing the framing bit added to the 192 bits carrying the 24 voice channels.

If I'm sitting at the receiver and I receive 193 bits, I know that one of those bits is the framing bit but I can't tell which one it is. To find the framing bit, I must receive many frames and the framing bits must create a bit sequence which has a very low probability of occurring in any other bit position. When the DS1 framing was first defined, the sequence of the framing bits was an alternating pattern of ones and zeros.

Since the sampling rate of speech is 8,000 samples per second, by Nyquist the maximum bandwidth which can be sampled is 4 Khz. To prevent aliasing, the G.711 codec has an analog filter which attenuates signals above 3400 Hz, requiring at least 14 dB of attenuation at 4KHz. Because of this, it is extremely unlikely that the digitized speech contains a significant 4KHz component. And if you work it out, a voice channel can only have alternating bits in the information portion of the frame if the digital

[1] ANSI T1.102 of 1993 calls for a tolerance of +/-32 parts per million on the 1.544 Mbps signal.

data represents a 4KHz signal[2]. So a bit alternating between zero and one has a low probability of occurring in any bit position except the framing bit.

So how do we find the framing bit? There are a number of more efficient techniques, but the brute force technique is to receive a sequence of bits, and select one bit position as the candidate position for the framing bit. Then, look ahead 193 bits and see if that bit is opposite to the first bit chosen. If not, select the next bit and try again. But if the bit is opposite, look ahead another 193 bits and see if that bit changes polarity back to the first bit. If not, select the next bit and try again. Eventually, you'll find the framing bit, and with that, you'll be able to decode each of the speech channels correctly. Once you find the framing bit, continue to monitor it to make sure that it continues to alternate. If it stops alternating, it means that you no longer have frame synchronization and you need to go back and find the framing bit again.

So how long does it take to establish framing? For this type of bit framing, the *average* (or expected) time to achieve framing is given by the equation[3]:

$$\text{Frame time} = N^2 + \frac{N}{2} \text{ bit times}$$

Where N is the number of bits in the frame, including the framing bit. For a DS1 frame of 193 bits, the expected framing time is 37,346 bits, or 24.188 ms. The maximum is twice this, or 48.25 ms., representing the time it would take to examine each of the 193 bits in sequence (you start at the bit just after the framing bit and have to run through all the bits in the frame). It's important that the framing time be fast enough that people do not hear a loss of speech if synchronization is lost on a T1 line. That is, the line should be able to detect loss of synchronization and resynchronize fast enough that a person does not hear anything unusual. Special techniques have been developed to quickly recover from loss of synchronization based on the fact that loss of sync is usually a slip of a small number of bits.

With this framing, we can quickly find frames and obtain the payload (speech) information. But there's some additional information that needs to be communicated about a telephone call, and that's the "call progress" information. For example, you're sitting at the T1 receiver in a telephone central office, all framed up, taking bytes and putting them in the right DS0 channels. But telephone calls are not permanent – they are set up and then torn down. How can you know when a call is coming in on a DS0 or when the call is finished?

Today, this is accomplished by communications in a separate channel using a technology known as Signaling System 7. This technique is known as "common channel signaling" (CCS). But back in the early days of the telephone network, signaling was done in-band, known as "channel associated signaling" (CAS)[4].

Signaling does not require a great many bits, and if you're going to preempt some of the information bits for signaling, you want to take as few as possible. The way the designers obtained bits for signaling is that they stole (usually called "robbed") the low order bit from a speech byte, every six speech samples.

[2] This, of course, is also the reason we use a 1004Hz test tone instead of a 1000Hz tone. If we used a 1000Hz tone and it was synchronized with the sample clock (exactly 1/8 of the 8,000Hz sampling rate), the same eight samples would be sent over and over. This would not do a good job of testing the codec or the channel.
[3] The derivation of this equation is given in Bellamy. See the Bibliography.
[4] It was this in-band signaling which allowed the "phone phreaks" to steal long distance. They would put certain tones on the line which kept the system from billing for a call. Common channel signaling stopped this. But who would want to risk stealing long distance now, when it's essentially free?

What impact might this have on the speech? The byte with the robbed bit is sent to the DS0 channel, where it's used to generate the speech sound. Fifty percent of the time, even though the network uses the low order bit, it is not changed. The other fifty percent of the time, the bit is changed but the change in sound is minimal – the sound is changed to the next quantization point (up or down) but the actual difference in sound is not great enough to cause noticeable distortion for the listener.

This robbed bit could be used in a variety of ways. Most logically, it could create a low speed channel within which signaling messages could be sent. But this is not the way signaling was done in the early days of the digital network. What the designers did was to create a two bit message (later extended to four bits), called the A and B bits. With the A/B bits, four messages could be sent – {00, 01, 10, 11}. Additionally, they could pulse the bits (called "wink") to send additional signals[5].

In light of today's technology, this signaling technique seems extremely limited while also adding complex framing conditions (to be explained shortly). But remember how this developed. T1 lines were initially used between analog central offices, simply to replace the multiple analog circuits which used to be used. The signaling had to be something which would interface with the analog switches already installed and operational in the central offices. Thus this "kluge" of the A/B bits.

So how did the designers set up the framing so that they could find these A/B bits? They created a concept called a "superframe" which consists of twelve of the 193 bit frames. So now, in addition to finding the start of a frame, the receiver has to find the start of the superframe. How is this done?

We still need a pattern of alternating zeros and ones because this pattern has a low probability of occurring in the speech samples. Suppose that we put this alternating pattern on every *other* frame instead of each frame. So the framing bit on frame one is a 1, the framing bit on frame three is a 0, the framing bit on frame five is a 1, etc. We can certainly find this pattern because it's just as if we increased the frame size to 386 bits instead of 193 bits. It will take longer to obtain synchronization (worse case of 96.5 ms with the brute force technique) but superior techniques have been developed which obtained synchronization faster than the brute force technique described earlier.

And once we find synchronization on the 386 bit frames, we need to find the start of the 12 frame (193 bit frame) superframe. This is done by putting a unique sequence of bits in the other six framing bits within the superframe. This sequence is {0, 0, 1, 1, 1, 0}. Now we can obtain frame synchronization *and* superframe synchronization. See Table 1 which shows how the framing bits are allocated.

[5] Later, tones were used in conjunction with the A/B bits. For example, MF tones are used to send dialed digits after the receiver signals through the A/B bits that it is ready to receive.

Frame number	Bit number	Frame alignment bit value	Superframe alignment bit value	Signaling bit value in low order data bits
1	0	1	-	
2	193	-	0	
3	386	0	-	
4	579	-	0	
5	772	1	-	
6	965	-	1	A
7	1158	0	-	
8	1351	-	1	
9	1544	1	-	
10	1737	-	1	
11	1930	0	-	
12	2123	-	0	B

Table 1: Superframe framing and signaling bits.

And within this superframe, we're interested in the sixth frame and the twelfth frame. Each of these two 193 bit frames contains 24 bytes of digital speech, one byte for each channel. Each of these bytes has the last (lowest order) bit stolen to be used for signaling. The low order bits in the sixth frame are the A bits and the low order bits in the twelfth frame are the B bits. Note that this affects *every* channel on the T1 circuit. Both the sixth and twelfth frames contain bytes for all 24 channels and all have the lower order bit stolen. So every channel on any T1 which uses robbed bit signaling will have "corruption" in every sixth speech sample.

More information on how the A/B bits are used for signaling can be found in ANSI T1.403.02-1999.

The superframe technique was a good solution for use between central offices. The telephone companies could monitor the lines on each end and determine the quality of the lines. But when T1s started being used for provisioning service to customers, problems were encountered, primarily in monitoring the quality of the line. To solve this problem, another framing technique, known as "Extended Superframe" (ESF) was developed.

ESF groups 24 frames (193 bit frames) together to form the extended superframe, so now we have 24 "framing bits" that we can use for a variety of purposes. Remember that in the superframe, we used six of the twelve framing bits for individual frame alignment, and six for superframe alignment. A more complex allocation of bits is used for ESF. Of the 24 bits, six are allocated for the "frame alignment signal" (FAS), six bits are allocated to CRC, and twelve bits are allocated to a data link channel.

The frame alignment bits occur every fourth frame, starting with the fourth frame and consist of the bits {0, 0, 1, 0, 1, 1}. Note that the framing pattern is no longer an alternating {0, 1} pattern. While this pattern could be duplicated in the data portion of the frame, it is very unlikely that both the FAS and the CRC could be duplicated, making the frame synchronization even more robust than previous techniques[6].

[6] Since there are six CRC bits, the probability of a match is only one in 64 making the probability of a match of both the FAS and the CRC of 1 out of 64^2 for random data.

Clever synchronization techniques are required now because the framing bits occur every 772 bits, leading to a long synchronization acquisition time if the brute force technique were used (almost 200 ms worse case).

The CRC is calculated on the extended superframe block with the 24 framing bits assumed to be set to 1. The result of the CRC calculation is then put into the CRC locations in the *next* ESF block[7]. Thus, the framing bits are not checked for errors by the CRC – only the information bits.

The data channel is open for any use but is usually used for system monitoring traffic. That is, HDLC frames are communicated between the central office and the T1 terminating equipment for control, testing and status monitoring purposes. See Table 2 which shows how these bits are allocated.

Frame number	Bit number	Frame Alignment Signal bit value	Data Channel bits	CRC bits	Signaling bit value in low order data bits
1	0	-	M1		
2	193	-		C1	
3	386	-	M2		
4	579	0			
5	772	-	M3		
6	965	-		C2	A
7	1158	-	M4		
8	1351	0			
9	1544	-	M5		
10	1737	-		C3	
11	1930	-	M6		
12	2123	1			B
13	2316	-	M7		
14	2509	-		C4	
15	2702	-	M8		
16	2895	0			
17	3088	-	M9		
18	3281	-		C5	C
19	3474	-	M10		
20	3667	1			
21	3860	-	M11		
22	4053	-		C6	
23	4246	-	M12		
24	4439	1			D

Table 2: Extended Superframe framing, data link and CRC bits.

Note that we now have a set of four "robbed bits" designated as {A, B, C, D}. This allows us to have sixteen signaling states instead of the four possible with just {A, B} bits. The four state robbed bits can be used in a degenerate form as {A, B, A, B} which is the same as the superframe situation.

[7] Details of how the CRC is calculated can be found in ANSI T1.107-1995.

DS2 Framing

When digital links were first introduced into the network, it was as a replacement for analog lines between analog central offices. Because of this, it was not important to maintain any kind of synchronization between the different T1 links. The voice traffic was converted to analog in the central office so there was no passage of "clock" between T1 circuits. Each T1 was supposed to run at 1.544 Mbps but in reality different T1 lines had different average bit rates.

This caused problems when the designers wanted to multiplex a number of T1 lines together. How can you bit multiplex several sources with different clocks? The easiest way to do it is to have the output bit stream operate at a rate slightly higher than the combined rate of the input bit streams, and use bit stuffing to make up the extra bits.

Let's take a made up example. Suppose you wanted to multiplex two bit streams which are each supposed to be 1 Mbps, but could be slightly more or less than 1 Mbps. We could multiplex these two streams into a 2.02 Mbps stream (excluding any overhead in the multiplexed stream), providing 10 Kbps of overage for each of the bit streams. Figure 4 shows an example where the two input bit streams are running fast enough that stuffing bits are not required (at least not for the short period shown in the example).

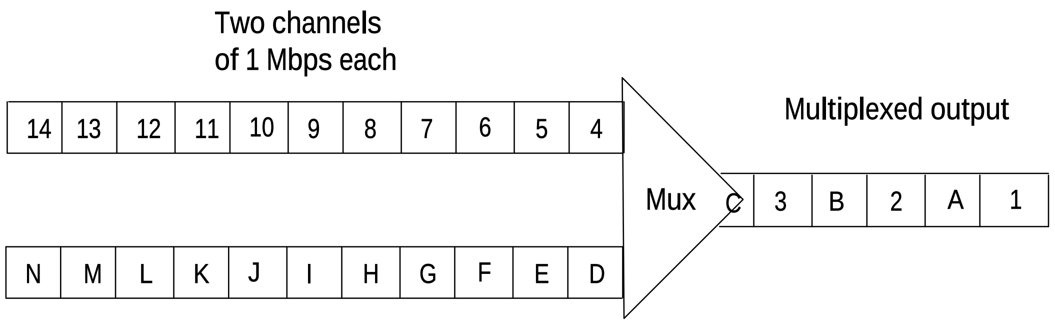

Figure 4: Two bit streams multiplexed without stuffing.

The way it works is that the two input data streams flow into the multiplexed stream, bit interleaved. When there's not enough input bits in one of the data streams, a "stuff" bit is inserted into the output bit stream in a bit location for that channel. On the receiving end, this stuff bit is removed from the bit stream. Figure 5 shows an example where the first input bit stream is operating at a slower rate than the second bit stream and requires a stuffing bit.

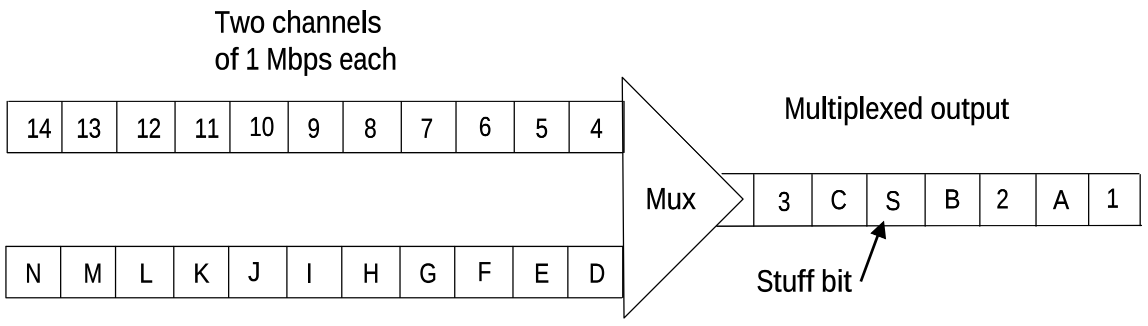

Figure 5: Two bit streams multiplexed with stuffing in the first bit stream.

In a real implementation, additional overhead bits are required in the output stream to signal the receiver when a stuff bit is inserted. As long as none of the input bit streams exceeded 1.01 Mbps, this system will work fine. Depending upon how often a stuff opportunity occurs, there's a lower limit on the bit rate of each of the input data streams. For example, if stuff opportunities occur 10,000 times per second, the input data stream could not be less than 0.990 Mbps or errors would occur. At 0.990 Mbps, each stuff opportunity would be stuffed, rather than carry an information bit.

One important thing to note is that the multiplexing does not change the lower speed bit streams – they are just taken bit-by-bit and multiplexed into the higher rate bit stream. The multiplexing process does not look for framing or overhead bits in the lower speed input bit stream. Any examination of the lower speed bit stream must wait until the bit streams are demultiplexed. This means that it is difficult to drop or insert a voice channel at an intermediate node of a high-speed link. To drop or insert, the higher speed link must be completely demultiplexed, the channel dropped or inserted and then everything multiplexed back up again.

Stuffing also requires that the individual demultiplexed bit streams be "clock smoothed" at the receiver so that the clock interval does not change abruptly when a stuff bit arrives. Buffers and phase locked loop circuits are used for this.

The technique described above is essentially the technique used to multiplex DS1 bit streams into a DS2 stream and DS2 streams into a DS3 stream. Now, let's examine the specifics of how this is done, starting with the DS2 framing.

A DS2 bit stream has a nominal rate of 6.312 Mbps and multiplexes four DS1 signals. The DS2 frame is as shown in Figure 6.

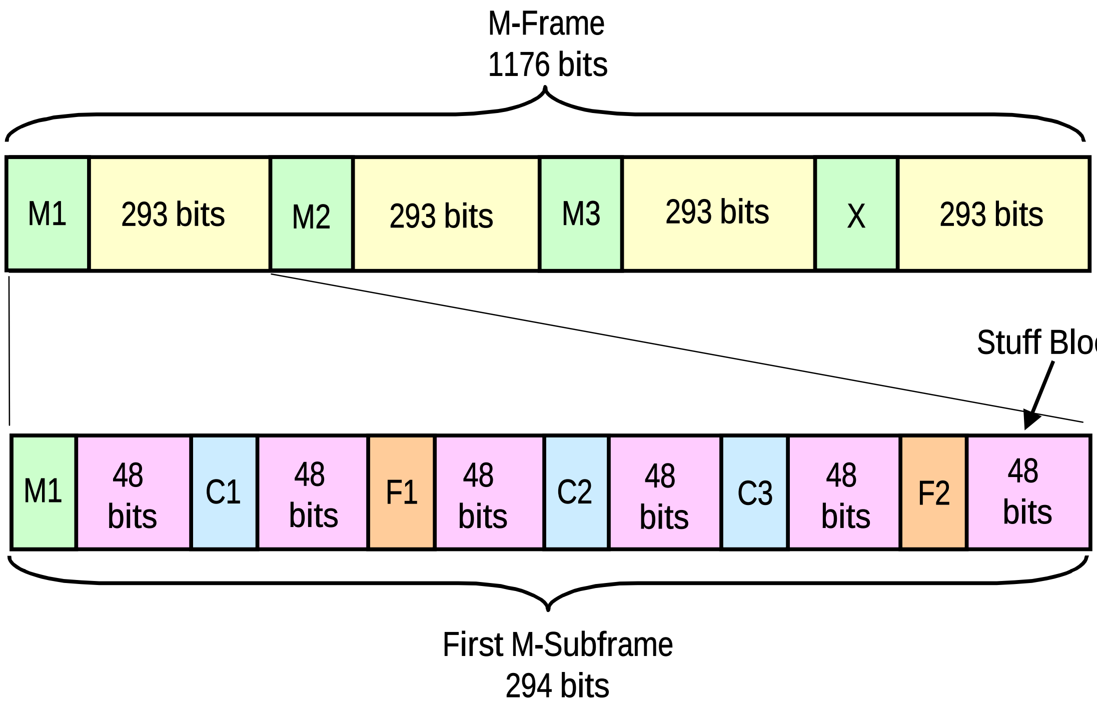

Figure 6: The DS2 Frame.

Frame synchronization is found as follows. First, the receiver looks for the repeating F1, F2 pattern which is {0, 1}. Each of these bits defines a 147-bit "frame" and there are eight of these 147-bit frames in an M-Frame of 1176 bits. Average F bit synchronization time is 21,683 bits, very quick with the nominal rate of 6.312 Mbps. Once the receiver has found the repeating F bit pattern, it has found the basic frame structure but not the M-Frame structure. The three M bits are then used to obtain M-frame synchronization and have the pattern {0, 1, 1}. The M-Frame synchronization can be found very rapidly by buffering 1176 bits and examining the candidate M bit positions. The X bit is used for an alarm channel and is normally set to 1 unless synchronization is lost on the Rx side of the link.

The DS1 data is bit multiplexed into the 48 bit information blocks of the M-Subframe in order. That is, if we look at any one 48 bit information block, the first bit will be used by the first DS1, the second bit will be used by the second DS1, the third bit will be used by the third DS1, and the fourth bit will be used by the fourth DS1. The fifth bit will then be used by the first DS1 and the cycle repeats. Thus, each 48-bit information block carries 12 bits for each DS1, except the last 48-bit block in each subframe, which will be described next.

Stuffing is controlled by the C bits in a Subframe. The first M-Subframe has a stuff location for the first DS1 located in the first bit position of the last 48-bit information block. Thus, the first M-Subframe carries either 71 or 72 bits for the first DS1 and 72 bits for the rest of the DS1s. The C bits indicate whether the stuff location carries an information bit or a stuff bit. If the C bits are set to 1, the location is interpreted as a stuff bit and the bit is ignored at the receiver. If the C bits are set to 0, the location is interpreted as an information bit and the receiver passes the bit into the output bit stream. Determination of whether to interpret the C bits as a 0 or 1 is done by majority voting to handle situations where bit errors may have occurred. Thus two C bits set to 1 and one C bit to 0 will cause the stuff location to be handled as a stuff bit.

The second M-Subframe has a stuff location in the second bit of the last 48-bit information block, and that location is controlled by the C bits in that Subframe. The third and fourth subframes have stuff locations in the third and fourth bit positions of their last 48-bit information block, respectively, and are controlled by the C bits in their subframe.

This means that each DS1 can send 287 or 288 bits per M-Frame. Let's look at the DS1 bit rate tolerance.

$$\frac{6.312 * 288}{1176} = 1.5458 \text{ Mbps}$$

$$\frac{6.312 * 287}{1176} = 1.5404 \text{ Mbps}$$

So as long as the DS1 signals stay between 1.5404 and 1.5458 Mbps, the DS2 will be able to accommodate the rates without introducing errors. This is a very wide range requiring clock accuracy of only about 1,000 parts per million. These rates are not absolute because the DS2 rate of 6.312 Mbps is a nominal rate and the actual rate may be higher or lower, but these are reasonable guidelines, and extremely easy to meet.

Next, I'm going to describe how the DS2 frames are multiplexed into a DS3 frame. This is done in an M13 multiplexer chip when 28 DS1 circuits are multiplexed into a DS3. What really happens inside the chip is that the DS1 circuits are taken four at a time and multiplexed into DS2 frames. Then, the seven DS2 frames are multiplexed into the DS3 frame. When this is done, the nominal bit rate of the DS2 is set to 6.306272 Mbps rather than 6.312 Mbps. This places a somewhat tighter restriction on the DS1 rates, allowing the DS1 to operate between 1.539 and 1.5444 Mbps. The DS1 clocks must be better than 260 parts per million in an M13 system, a specification which is easy to meet.

DS3 Framing

If you followed my explanation of the DS2 framing, the DS3 framing will be easy to understand – it's essentially the same thing but with seven inputs instead of four.

The nominal bit rate of a DS3 is 44.736 Mbps. It multiplexes seven DS2 signals, each of which contains four DS1 signals so there are twenty-eight DS1 signals in a DS3, and 672 DS0 channels. Figure 7 shows the DS3 framing structure.

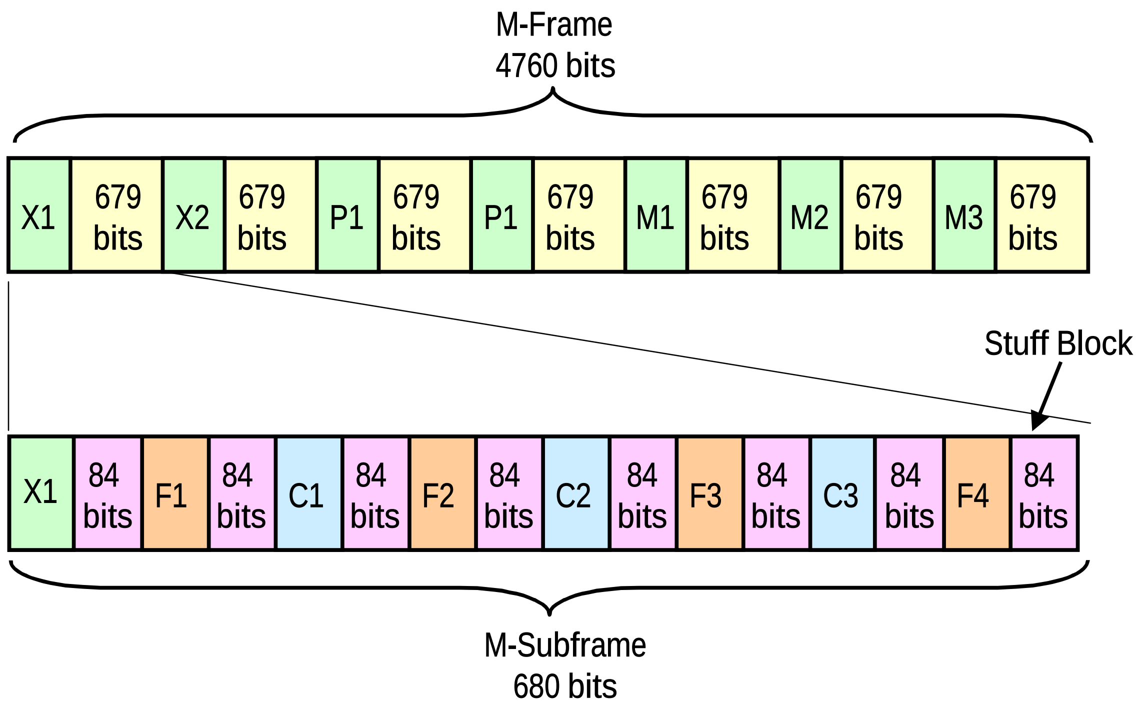

Figure 7: The DS3 Frame.

Frame synchronization is found in a manner similar to the DS2 frame. First, framing is found on the F bits which alternate {1, 0, 1, 0} (F1, F2, F3, F4). Each F bit defines a 170 bit frame, meaning that the average F bit synchronization time is 28,985 bit times, very quick at the 44.736 Mbps line rate. Once F bit frame synchronization is found, M-Frame synchronization is found by searching for the M bits, which are set {0, 1, 0} (M1, M2, M3). The X and P bits will either be {0, 0} or {1, 1}. This allows M-Frame synchronization to be found because the only {0, 1, 0} transition will occur in the {M1, M2, M3) bits and cannot occur in any other combination of the {X, P, M} bits.

The X bits can be used to send backward status, to indicate the receipt of a severely errored frame or certain other types of failure indications. The P bits are the parity bits, based on the previous frame. If the parity of the data bits (only) of the previous frame is 1, the two P bits will be set to {1, 1}. If the parity is zero, the P bits are set to {0, 0}.

The DS2 bit streams are bit multiplexed into the 84 bit data blocks of the DS3 subframe in order. So the first DS2 uses the first bit of an 84-bit block, etc., just like the DS1s are laid into the DS2 frame.

Stuffing is controlled by the C bits, with all ones indicating a stuff bit and all zeros indicating a data bit. Majority voting is applied to the C bits. The stuff locations are the first seven bits of the final 84 bit block in the M-Subframes. The first M-Subframe has its first bit as a stuff bit for the first DS2. All other bits in that block are standard data bits. The second M-Subframe has its second bit as a stuff bit for the second DS2, etc.

This means that an M-Subframe carries either 95 or 96 bits of a DS2 and an M-Frame carries either 671 or 672 bits for each DS2. Calculating the bit rate tolerance give us:

$$\frac{44.736 * 672}{4760} = 6.31567 \text{ Mbps}$$

$$\frac{44.736 * 671}{4760} = 6.306272 \text{ Mbps}$$

In the M13 multiplexer, all of the accommodation of the clock differences is accomplished in the DS2 frame, and the DS3 frame always contains stuff bits (the C bits will always be {1, 1, 1}) when operating in M13 mode.

This presented an “opportunity” to the standards people. If the C bits were not being used in an M13 multiplexer, they decided to use them for something else. That something else is called “C-bit parity” and will not be described here – just be aware that the C bits in a DS3 frame can have a different meaning.

European Digital Hierarchy

E1 Framing

The E1 framing is the first level in the digital hierarchy above the actual voice channels. Speech is coded according to the ITU G.711 specification, using A-law encoding, to give a 64Kbps bit stream per channel. Specifically, the speech is coded eight bits per sample, with 8,000 samples taken per second.

The E1 frame carries 30 voice channels in a 256-bit frame. Since 30 channels only requires 240 bits, 16 bits are available for framing, signaling, error checking and supervisory communications. These extra 16 bits are divided into two groups of 8 bits each. The first octet is located in the first 8 bit positions of the 256 bit frame, while the second is located in bit positions 129 through 136 (with the first bit numbered as 1). If we take the 256 bits 8 bits at a time, and call each 8 bits a channel, we have 32 channels in the frame. In the standard, these channels are numbered from 0 (zero) to 31. The first overhead octet represents channel zero, therefore, and is known as the framing channel. The next overhead octet represents channel 16 and is known as the signaling channel. The remaining channels are known as message channels. See Figure 8 which shows the basic E1 frame. Note that each numbered block represents 8 bits.

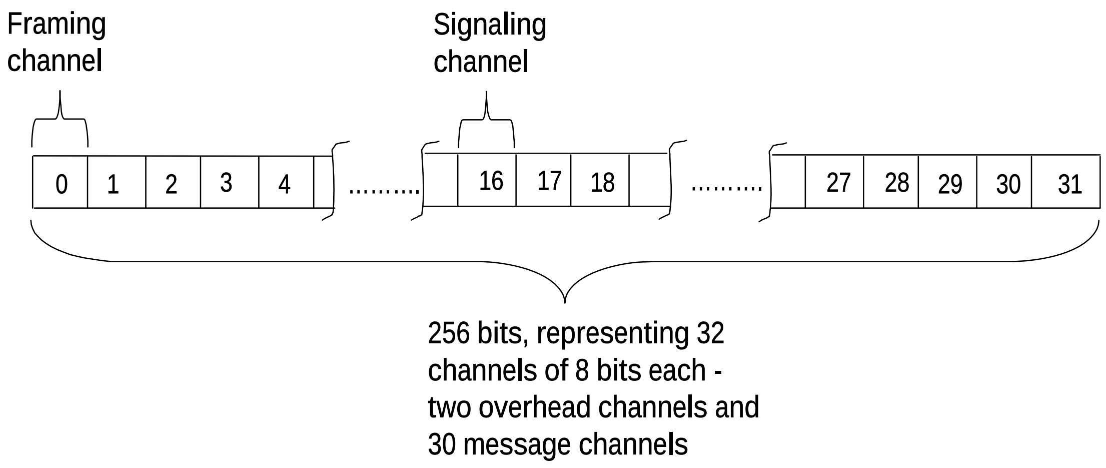
Framing channel
Signaling channel
0
1
2
3
4
....
16
17
18
....
27
28
29
30
31
256 bits, representing 32 channels of 8 bits each - two overhead channels and 30 message channels

Figure 8: The basic E1 frame of 256 bits, showing the framing channel and the signaling channel.

Sixteen of these 256 bit frames are organized into a multiframe, as shown in Figure 9. Now, let's examine how the framing is done so that we can find the start of the frame and the multiframe.

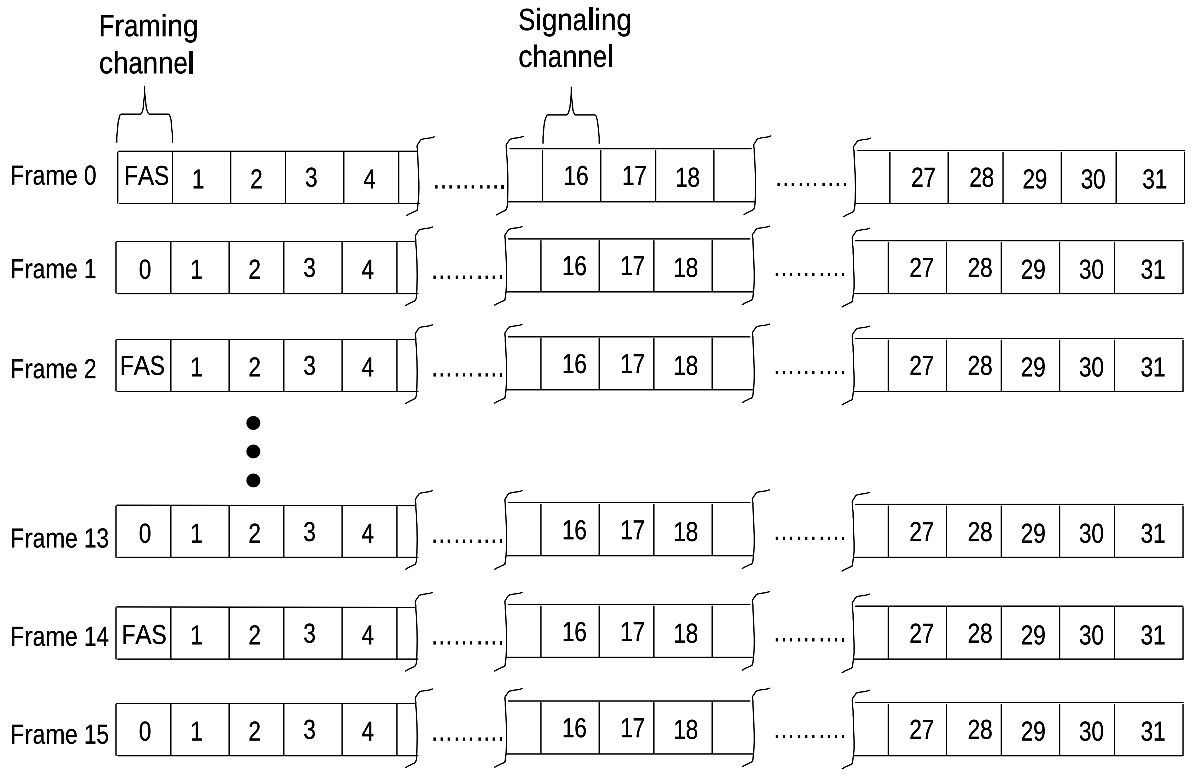

Figure 9: The E1 multiframe of 16 frames. Note that the frame alignment signal occurs every other frame.

The bits of the first octet of each frame are allocated either to the frame alignment signal (FAS) or to a message link for operations, maintenance, or performance monitoring. The use of the first octet alternates. The first octet on one frame will be the FAS, while the first octet on the next frame will be for the administrative channel. This means that, for framing purposes, the frame length is 512 bits.

The bits of the octets are set as follows (Table 3):

Frame description	**Bit number**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Frame containing the frame alignment signal	S_i	0	0	1	1	0	1	1
Frame not containing the frame alignment signal	S_i	1	A	S_{a4}	S_{a5}	S_{a6}	S_{a7}	S_{a8}

Table 3: Bit assignments in the framing channel of an E1 multiframe.

The first bit of each octet of the framing channel may be used for a CRC-4 to further verify multiframe alignment. This is described later. The frame alignment signal is {0, 0, 1, 1, 0, 1, 1}. In the octets not containing the FAS, the second bit is fixed to 1 to guarantee that the pattern in this octet can never be the same as the FAS. I won't describe the rest of the bits in the non-FAS octet except to say that they are used for remote alarm indication (A), administrative purposes, and synchronization status. See G.704 for more details.

Here we have seven bits of framing for each "frame", and for framing purposes, a frame is 512 bits. The equation[8] for the average time to achieve framing is

$$\text{Frame time } = \frac{N^2}{2(2^L - 1)} + \frac{N}{2} \text{ bit times}$$

Here, N = 512, and L = 7. The average time to achieve framing is 1288 bit times, or about 0.63 ms. Note that this is the average time. The maximum time will be twice the average. The reason this is so much faster than DS1 framing is that there's a higher percentage of framing bits. Note also, that if you substitute the DS1 values of N=193 and L=1 into this equation, you will not get the same answer as given in the DS1 section. This is because the DS1 framing consists of alternating bits while the E1 framing bits are fixed. With alternating bits the framing time is longer, here about twice as long.

When the first bit of each frame is not used for CRC-4 framing, it is normally set to 1. When the first bit of each frame is used for CRC-4, the multiframe is separated into two sub multiframes (SMF) of eight frames each. The first bit of the FAS in each SMF is used for the CRC-4 while the first bits of all eight non-FAS octets are used to carry a framing pattern plus a backward error indicator. This is shown in Table 4.

	Sub-multiframe number	Frame number	Bit 1 of each frame
Multiframe	I	0	C_1
		1	0
		2	C_2
		3	0
		4	C_3
		5	1
		6	C_4
		7	0
	II	8	C_1
		9	1
		10	C_2
		11	1
		12	C_3
		13	E
		14	C_4
		15	E

Table 4: Meaning of the first bit of each frame, when the bit is used for CRC-4.

The CRC-4 is calculated over the submultiframe (2048 bits), with the CRC bits set to 0 for the calculation. The result of the CRC-4 calculation is then placed in the C_1-C_4 bit locations of the *next* submultiframe. The CRC-4 alignment signal is {0, 0, 1, 0, 1, 1}. The E bits are used to indicate whether submultiframes were received with errors. Normally, the E bits will be 1, but are set to 0 to indicate a

[8] This equation is also derived in Bellamy.

submultiframe with a CRC error (the first E bit applies to the first submultiframe of a multiframe, while the second E bit applies to the second submultiframe).

Signaling in E1 is accomplished through the signaling channel (channel 16). This channel can carry signaling similar to the robbed bit signaling for DS1, or can carry HDLC (LAP-D) frames for higher-level type signaling. If ABCD bit type signaling is used, the first octet in channel 16 of the multiframe will contain {0, 0, 0, 0, X, Y, X, X}. The remaining octets of channel 16 (fifteen octets since there are total of 16 frames in a multiframe) will contain {A, B, C, D, A, B, C, D}. That is, each octet will contain the signaling for two channels. The first four bits will contain the signaling for the channel with the same number as the frame number. So the first four bits in the channel 16 octet of frame 1 will contain the signaling for channel 1.

The second four bits in that octet will contain the signaling for the channel 15 channels higher. So the second four bits in the channel 16 octet in frame 1 will contain the signaling for channel 16. Thus, when we get to frame 15 (the last frame of the multiframe), the signaling octet will contain signaling for channels 15 and 30. Thus, each multiframe contains signaling for each channel. The difference from DS1 is that no bits are taken from the message channels so each message channel is a clear channel 64Kbps (no robbed bits). Signaling in this channel is described in the Q.400 series of ITU recommendations.

E2 Framing

The E2 frame is created by bit multiplexing four E1 bit streams. Of course, it's a bit more complex than that because framing must be added along with a technique for adapting to varying clocks. Let's explore.

The E2 rate is 8.448 Mbps with a tolerance of +/- 30 ppm. Note the specification of a tolerance. There is no tolerance specified for E1.

The basic E2 frame is 848 bits, with a 10-bit frame alignment signal of {1, 1, 1, 1, 0, 1, 0, 0, 0, 0} at the beginning. The 848-bit frame is broken into four 212 bit sub frames, as shown in Figure 10.

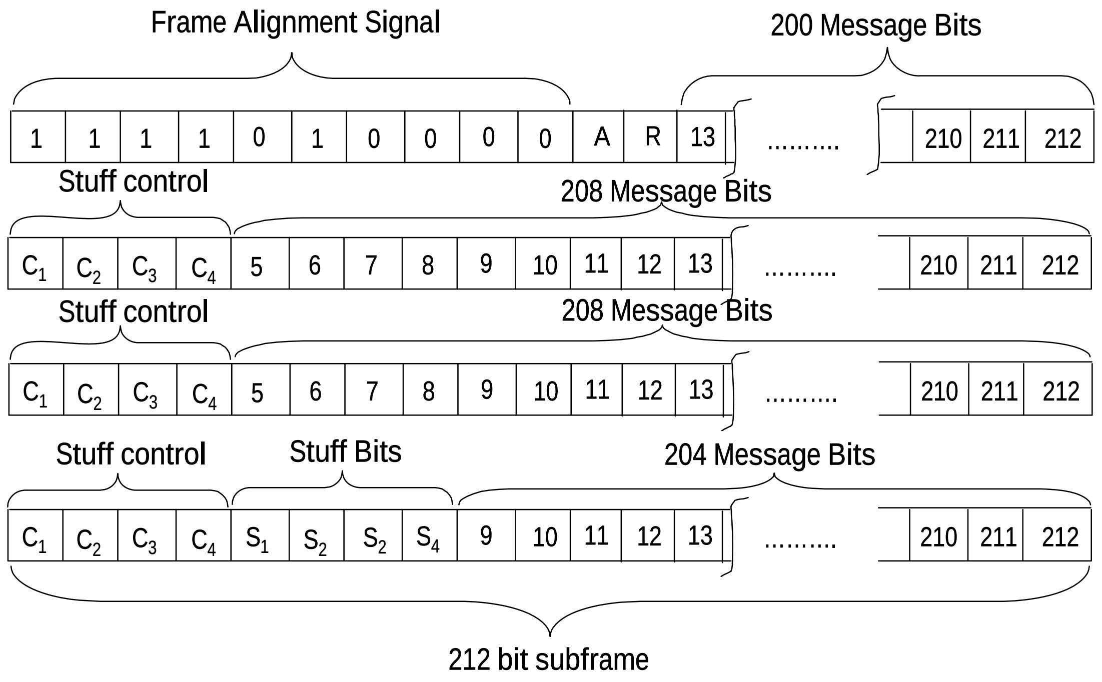

Figure 10: The E2 frame of 848 bits. The frame is broken into four subframes of 212 bits each.

Note the C_1, C_2, C_3, and C_4 stuff control bits. The three C_1 bits control stuffing for the first E1 signal, while the C_2, C_3, and C_4 bits control their respective E1 signals. Ones in the stuff control bits indicate justification, while zeros indicate no justification. Majority voting applies.

The maximum and minimum E1 rates which this justification can handle is

$$\frac{8.448 * 206}{848} = 2.052226 \text{ Mbps}$$

$$\frac{8.448 * 205}{848} = 2.042264 \text{ Mbps}$$

The difference between the nominal E1 rate is about –5.74Kbps and +4.23Kbps, leading to an E1 tolerance of better than 2000 parts per million.

The time to gain synchronization is given by the same equation as for the E1 frame

$$\text{Frame time } = \frac{N^2}{2(2^L - 1)} + \frac{N}{2} \text{ bit times}$$

The number of framing bits is ten and the number of bits in the frame is 848, leading to an average framing time of 775 bit times, or less than 0.1 ms. The maximum framing time is twice the average, or 1550 bit times (0.18ms).

E3 Framing

The E3 frame is created by bit multiplexing four E2 frames. The E3 rate is 34.368 Mbps, with a tolerance of +/- 20 parts per million.

The basic E3 frame is 1536 bits, with a 10-bit frame alignment signal of {1, 1, 1, 1, 0, 1, 0, 0, 0, 0} at the beginning. The 1536-bit frame is broken into four 384 bit sub frames, as shown in Figure 11.

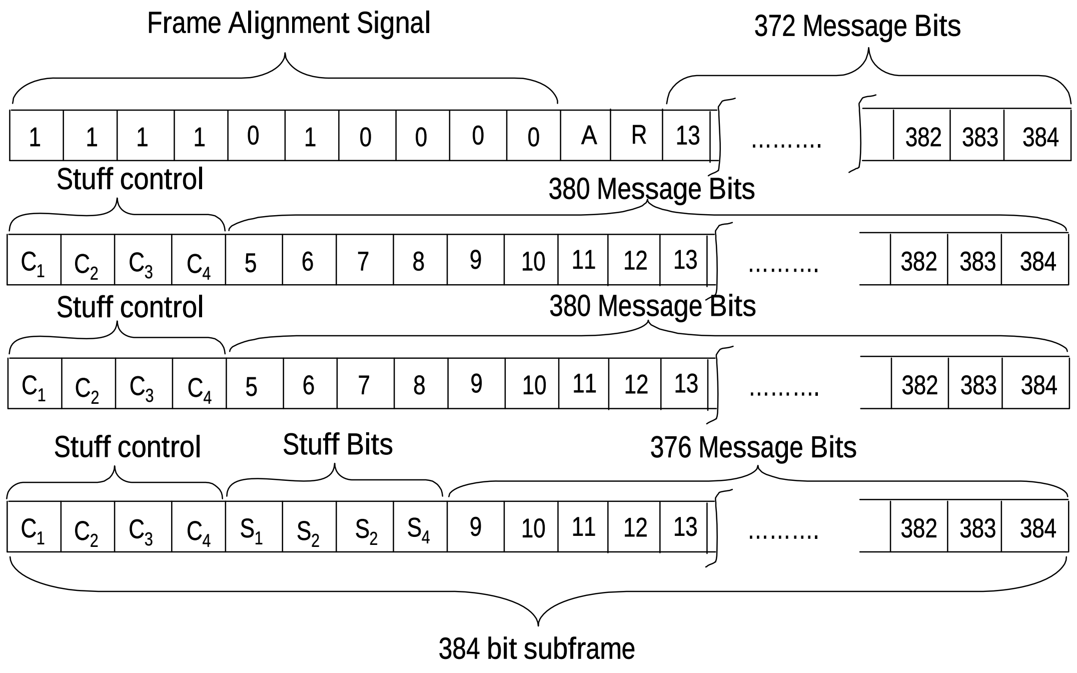
Frame Alignment Signal
372 Message Bits
1 1 1 1 0 1 0 0 0 0 A R 13 .... 382 383 384
Stuff control
380 Message Bits
C1 C2 C3 C4 5 6 7 8 9 10 11 12 13 .... 382 383 384
Stuff control
380 Message Bits
C1 C2 C3 C4 5 6 7 8 9 10 11 12 13 .... 382 383 384
Stuff control
Stuff Bits
376 Message Bits
C1 C2 C3 C4 S1 S2 S2 S4 9 10 11 12 13 .... 382 383 384
384 bit subframe

Figure 11: The E3 frame of 1536 bits. The frame is broken into four subframes of 384 bits each.

Note the C_1, C_2, C_3, and C_4 stuff control bits. The three C_1 bits control stuffing for the first E2 signal, while the C_2, C_3, and C_4 bits control their respective E2 signals. Ones in the stuff control bits indicate justification, while zeros indicate no justification. Majority voting applies.

The maximum and minimum E2 rates which this justification can handle is

$$\frac{34.368*377}{1536} = 8.435375 \text{ Mbps}$$

$$\frac{34.368*378}{1536} = 8.45775 \text{ Mbps}$$

The difference between the nominal E2 rate is about –12.625Kbps and +9.75Kbps, leading to an E2 tolerance of better than 1000 parts per million.

The time to gain synchronization is given by the same equation as for the E1 and E2 frames

$$\text{Frame time } = \frac{N^2}{2(2^L - 1)} + \frac{N}{2} \text{ bit times}$$

The number of framing bits is ten and the number of bits in the frame is 1536, leading to an average framing time of 1921 bit times, or about 0.05 ms. The maximum framing time is twice the average, or about 3442 bit times (0.11ms).

Summary

In this paper, I have attempted to explain the North American and European framing and multiplexing structures (DS1 to DS3 and E1 to E3), which are the digital side of the transmission problem. Appendix A includes a discussion of the line codes used to communicate this digital information over the copper wires.

I hope that this paper provided some assistance to you in understanding the North American and European framing and multiplexing structures. And if you're looking for information on SONET/SDH framing, see my paper on that subject available at www.michael-henderson.us.

Appendix A – Line codes

The basic technique for sending information over a copper wire is to apply a voltage in a certain pattern which can be detected and interpreted at the other end. Of course, like most things in life, the devil is in the details. I'm going to "selectively ignore" some of the details in this discussion and just talk about the voltage patterns which are used on the wire to communicate T1, T2, T3, E1, E2 and E3 signals.

The following discussion is "high level" and only addresses why certain things are done. I won't discuss the specific voltage levels, line impedances, etc. Those details can be found in the appropriate standards documents.

The basic technique for communicating over copper wire is to use voltage pulses. For example, a positive going voltage pulse could be used to represent a logic 1 and a negative going voltage pulse could be used to represent a logic 0. Such a signal is called a non-return to zero (NRZ) signal. If a non-zero voltage is used for a logic 1 (for example) and zero voltage is used for logic 0, the signal is called a return to zero (RZ) signal. If the voltage level is one sided (only a positive voltage or only a negative voltage), a problem known as DC balance is encountered. Because of this, one-sided RZ signaling is not commonly used in communications.

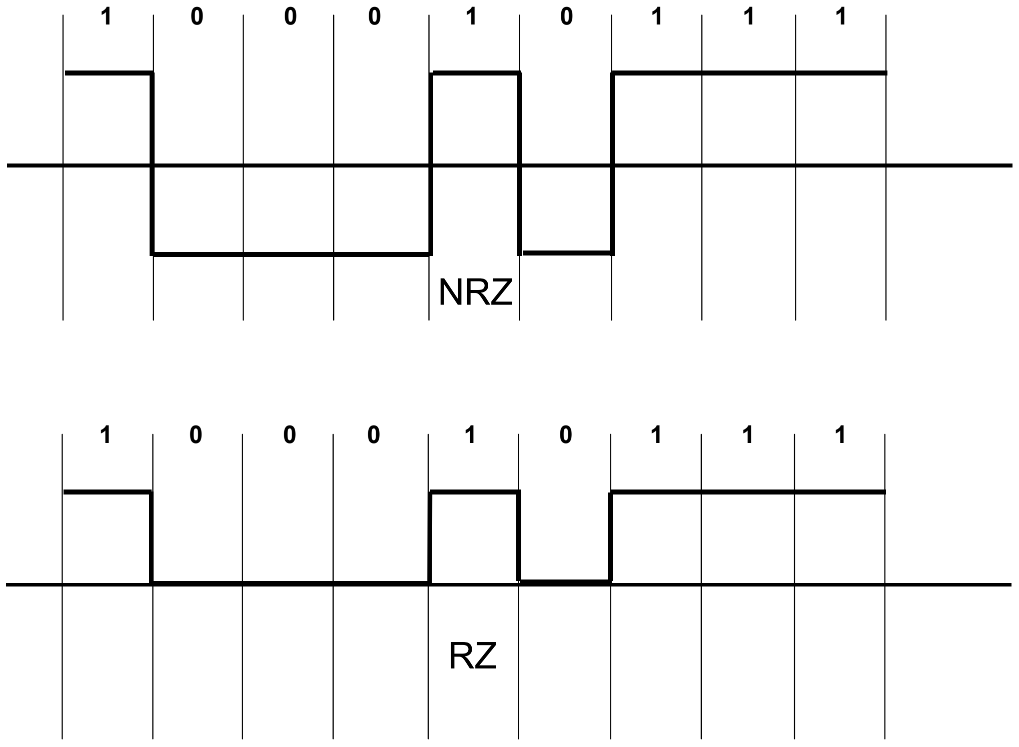

Figure 12: An example of a non-return to zero (NRZ) and a return to zero (RZ) signal.

In either case, if the pulse is a square wave with a 100% duty cycle, the frequency spectrum on the wire will be the familiar (sin x)/x function, shown below, the plot of which is shown in Figure 13.

$$F(\omega) = T\frac{\sin(\omega T/2)}{\omega T/2}$$

where: $\omega = 2\pi f$
T = pulse time

This equation is obtained by taking the Fourier integral[9] for a single square wave pulse centered at t=0 and extending from -T/2 to T/2. The power spectrum is obtained by taking the square of the modulus and multiplying it by a scaling factor. See Figure 13.

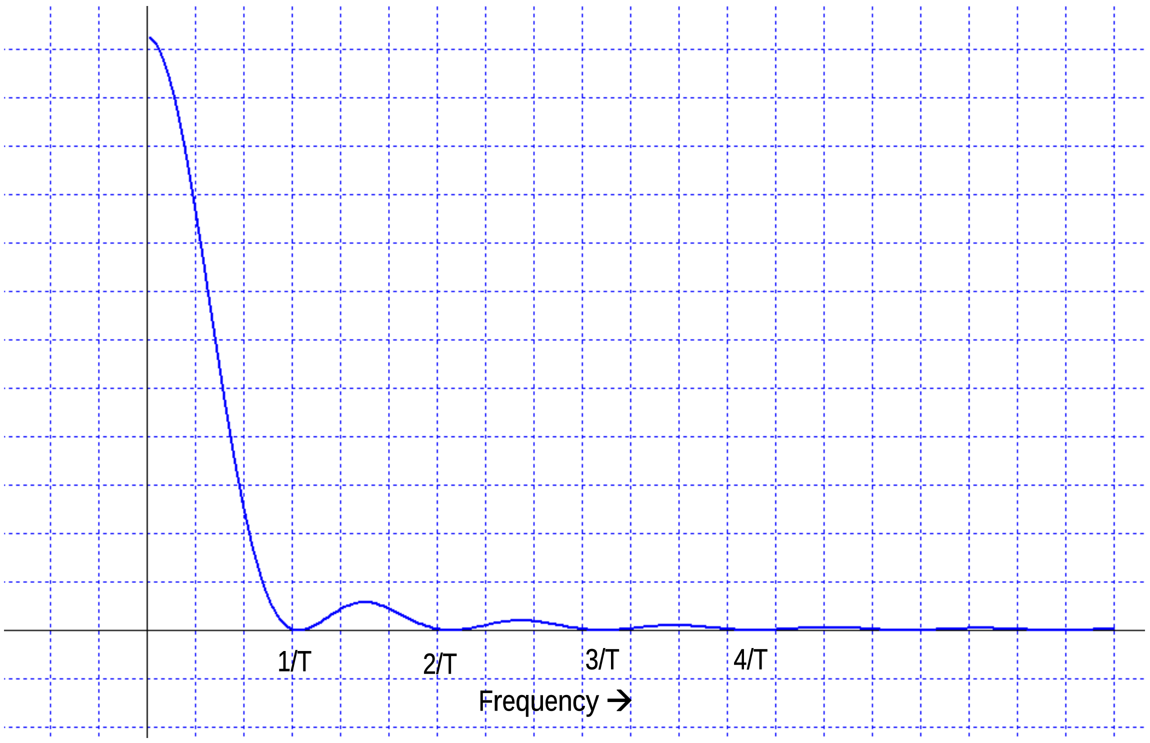

Figure 13: The plot of $F(\omega)^2(1/T)$, the power spectral density of the pulses represented by $F(\omega) = T\frac{\sin(\omega T/2)}{\omega T/2}$.

There are several things to note in this figure. First, note that the energy goes to zero periodically, with the period related to the duration of the pulse. Second, most of the energy is in the frequency band below 1/T frequency. Finally, since the pulse is a square wave, the frequencies will extend to infinity.

Now, let's talk about the real world. First, the frequencies can't extend to infinity because the attenuation on copper wire increases significantly as the frequency increases. So the pulse will not be square – it will be rounded off, approaching the shape shown below.

[9] The Fourier integral is $F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$. The Fourier integral converts a signal in the time domain to the frequency domain.

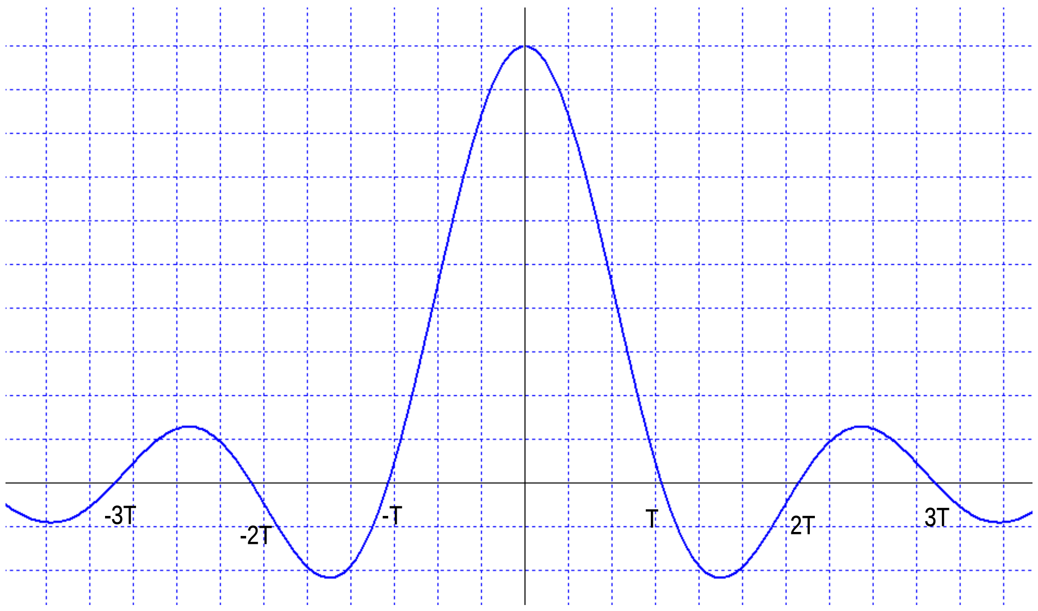

Figure 14: The typical pulse response of a band limited channel.

Note the little "wiggles" in the curve beyond time T. Now, look at Figure 15 which shows the frequency response of multiple pulses, at time zero, T, and 2T. Unless each signal is sampled at exactly the right time (at exactly 0, T, 2T, 3T, etc.), intersymbol interference will occur. For example, the pulses at T and 2T will be affected by the tail of the pulse at time 0. The pulse at time 2T will be affected by the tail of the pulse at time zero and time T.

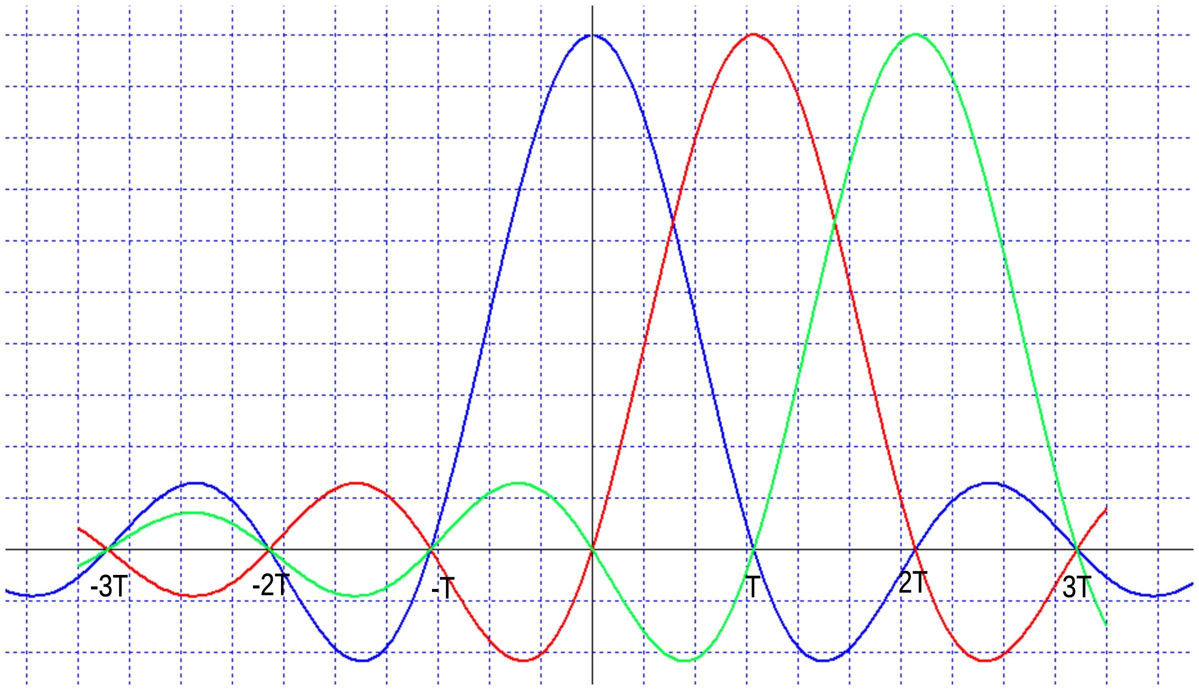

Figure 15: The frequency response of multiple pulses – at time zero, T, and 2T. The actual signal on the line will be the sum of the pulses, which are shown here individually for clarity.

The next figure, Figure 16, adds the summation of the individual pulses shown in Figure 15. Note how the amplitude is "wrong" everywhere except at exactly 0, T, 2T, 3T, etc. This is intersymbol interference.

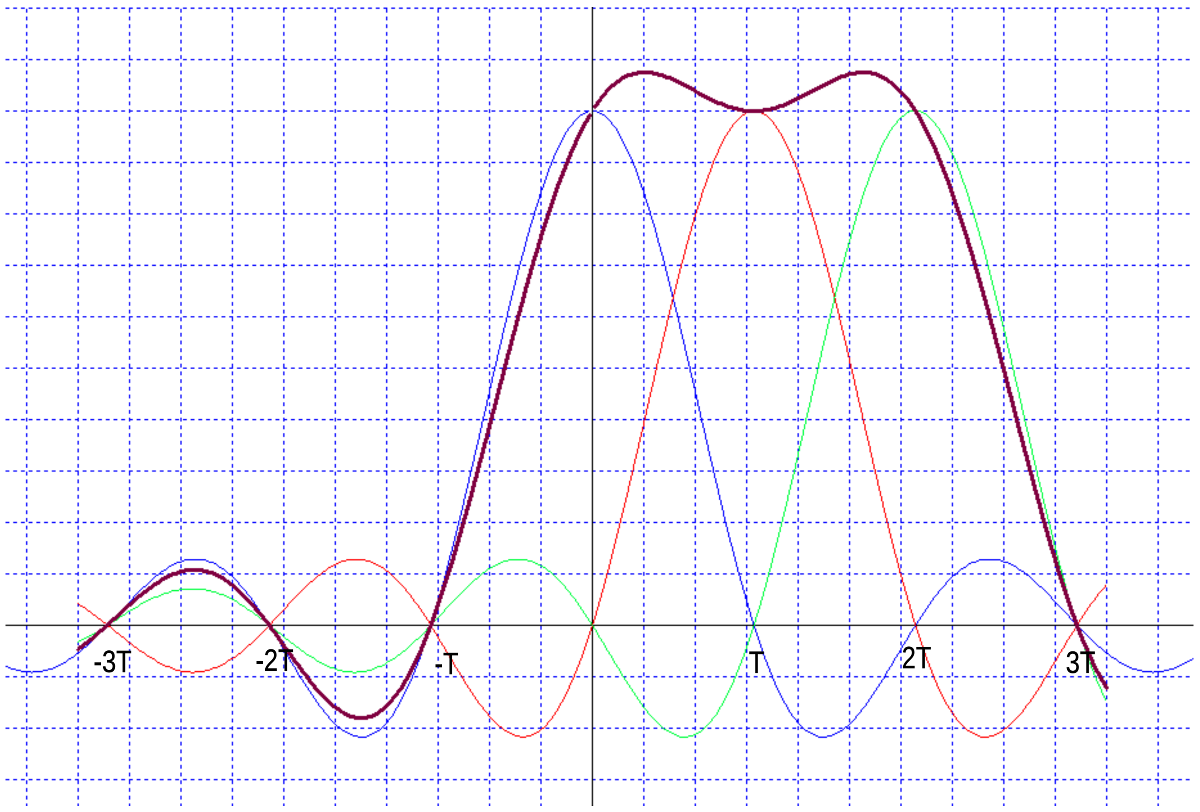

Figure 16: The actual signal on the wire is the sum of the individual pulses. Here, the heavy trace indicates the actual signal.

The second "real world" problem relates to DC in the signal. Figure 13 indicates that there's a substantial DC component in the signal. However, the signals are always AC coupled to the copper wires to avoid ground loops in the circuit, usually via a transformer. So if a positive voltage is used for logic 1, and a series of 1s are sent, the voltage on the line will decay towards zero with a time constant which depends upon the actual circuit design. If you sent a long enough sequence of 1s, the voltage would wind up very close to zero. The copper wire looks like an RC circuit with a certain time constant.

Both of these problems drive the need for transitions in the signal. Lots of signal transitions provide a way for us to maintain the accuracy of our sample clock and, if used properly, can avoid the problem of DC balance.

Suppose we use a positive pulse for a logic 1 and a negative pulse for a logic zero. If we send a lot more 1s than 0s, we will be putting more positive charging pulses on the wire than negative charging pulses. If we do this in a very short time period, in the order of a few RC time constants, the line will accumulate a net positive voltage. In an extreme case, the line could accumulate a positive voltage close to the value used for the logic 1. Now, when a negative pulse comes along, it may only drive the line to zero, making it difficult for the detector to know what was actually sent. The same problem will occur if there are a lot more negative pulses than positive pulses, but the polarity will be reversed.

The easiest way to avoid this problem is to have an equal number of positive and negative pulses, so that the average voltage on the line is zero (no DC).

Okay, now that we've discussed all this theory, let's see exactly what the designers actually did to communicate over the copper wire.

The actual line code used is known as a "bipolar" line code, where a plus *or* a minus voltage represents a logic 1, while a zero voltage represents a logic 0. Additionally, the positive and negative pulses have a 50% duty cycle instead of a 100% duty cycle as we discussed earlier[10]. Since in the "old days" a logic 1 was known as a "mark"[11], this code is commonly known as "alternate mark inversion (AMI)."

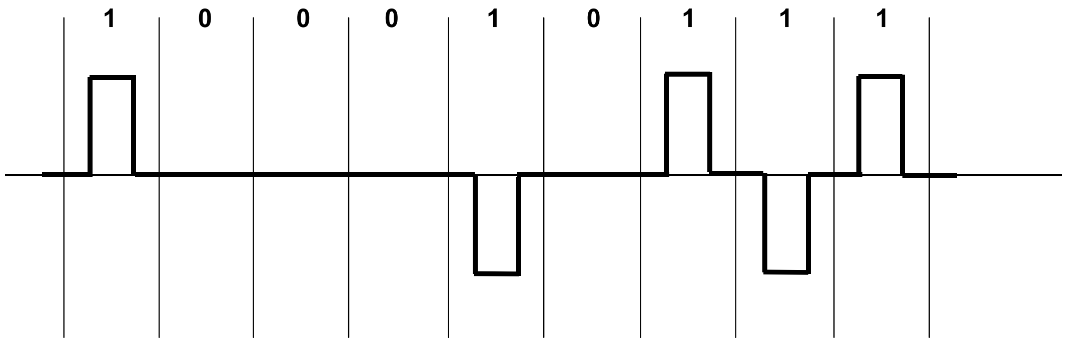

Figure 17: 50% duty cycle alternate mark inversion (AMI) code.

AMI provides the ability to check for single bit errors. If two consecutive pulses are received with the same polarity, then a single bit error has occurred. This is known as a "bipolar violation" and is used to monitor the line for errors.

The equation for the frequency spectrum of a 50% duty cycle pulse is

$$F(\omega) = \frac{T}{2} \frac{\sin(\omega T/4)}{\omega T/4}$$

This equation is derived by taking the Fourier integral of a single square wave pulse centered at *t*=0 and extending from -T/4 to T/4. The power spectral density for this equation has "nodes" which are twice as wide as the nodes of the 100% duty cycle pulse spectrum shown in Figure 13, but half the amplitude – the energy of the pulse is "spread" in the frequency domain.

A string of logic 1s will have a strong timing component since the signal will alternate in polarity each symbol time. However, a string of logic zeros will be a constant zero voltage. The AMI technique avoids the DC balance problem but a long string of logic zeros could allow the clock at the receiver to drift sufficiently such that it is not sampling at the correct point. This introduces intersymbol interference, or with really severe drift, could allow the sampling to miss a symbol completely.

[10] The 50% duty cycle is specified for T1, T2, T3, E1, E2, and E3 lines.
[11] A logic 0 was known as a "space."

This means that something must be done to avoid long strings of logic 0s, and that's what we'll discuss next. The term generally used for this is "maintaining ones-density."

Initially T1 lines only carried voice information. To maintain one-density, the all zeros speech sample was corrupted to have a 1 in the second low order bit (bit 7 where bit 8 is the low order bit). This caused an "error" in the speech, but an all zeros speech sample didn't occur very often and the human ear tolerated the error well. But when computer data is carried over a T1 line, you can't corrupt each all-zero byte. The error would cause a retransmission, which would encounter the same error (unless the data was scrambled). And for higher-level signals, such as the DS2 or DS3 signals, you can't corrupt bits because these higher-level signals have no idea what each of the lower level signal bits mean. Some other solution had to be found.

The solution chosen is a form of "binary N-zero substitution" (BNZS) and involves introducing bipolar violations in a special way to signal the receiver that a string of zeros is being communicated. Let's take for an example the code used on a T3 line, the replacement of a string of three zeros, called a binary three-zero substitution (B3ZS). In the B3ZS line code, every string of three consecutive zeros is coded as 00V or B0V, where V represents a bipolar violation and B represents a standard bipolar alternation (not a violation). The choice between substituting with 00V and B0V is made based on the number of "normal" bipolar alternations since the last violation. The choice is made so that the number of normal alternations is odd between violations. Thus if the number of alternations prior to the 000 is odd, the 00V substitution is chosen. Otherwise, the choice is B0V. If you write out a sequence of pulses according to this rule, including the previous violation and its preceding positive or negative pulse, you'll see that this rule maintains DC balance.

This rule leads to the following substitution table.

Polarity of preceding pulse	**Number of bipolar pulses since the last violation**	
	Odd	**Even**
Minus	00–	+0+
Plus	00+	–0–

Table 5: Substitution table for the B3ZS code

T1 circuits use a B8ZS code and the obsolete T2 line used a B6ZS. Since the substitution contains an even number of positive and negative pulses (each is DC balanced) it is not necessary to track whether there have been an even or odd number of pulses since the last violation.

Polarity of preceding pulse	**Substitution**
Minus	0 – + 0 + –
Plus	0 + – 0 – +

Table 6: Substitution table for the B6ZS code.

Polarity of preceding pulse	Substitution
Minus	000 – + 0 + –
Plus	000 + – 0 – +

Table 7: Substitution table for the B8ZS code. Two zeros are added to the beginning of the B6ZS code.

The ITU chose a slightly different code which they called "high-density bipolar" (HDB). The specific code they chose allows up to three consecutive zeros so the code is called an HDB3 code. In reality, it's a B4ZS code with the substitution rules as given in Table 8.

Polarity of preceding pulse	Number of bipolar pulses since the last violation	
	Odd	Even
Minus	000–	+00+
Plus	000+	–00–

Table 8: Substitution table for the HDB3 code.

The HDB3 code is used for E1, E2 and E3 lines.

HDSL

Most T1 lines are not provisioned with AMI line coding today. Because of the wide spectrum requirements, the AMI signal is rapidly attenuated – the attenuation on copper twisted pair increases significantly with frequency. Standard AMI requires amplification every 6,000 feet, making provisioning very time consuming. Additionally, cross talk in the cable bundle is directly proportional to frequency, making AMI T1 signals very "noisy." In fact, most AMI T1 lines have the transmit and receive pairs provisioned in different cable bundles.

To get around this problem, a technique known as high-bit rate digital subscriber line (HDSL) was developed. HDSL is nothing more than the 2B1Q line coding used on ISDN BRI lines, but pushed to higher speeds. HDSL uses echo cancellation so that a single pair can be used for both transmit and receive. However, when HDSL was developed, it was not possible to put the full 1.544 Mbps signal over a single pair so two pair were used, each carrying half the T1 data rate.

If the 1.536 Mbps *payload* signal is divided between two lines, each will carry 768 Kbps. The 8 Kbps of DS1 framing is duplicated and transmitted on each line and 8 Kbps of overhead is added per line, giving an actual line rate is 784Kbps per pair. Since 2B1Q line coding is used, the signaling rate will be 392 K symbols (or pulses) per second. If we assume that the pulses have a 100% duty cycle, the equation for the frequency spectrum is as given earlier:

$$F(\omega) = T\frac{\sin(\omega T/2)}{\omega T/2}$$

The duration of the pulse, T, is 1/392,000 sec. The first zero in the power spectrum plot occurs at 1/T Hz or at 392KHz, which is significantly lower than the first zero for AMI which is at 3.088MHz.

HDSL revolutionized the provisioning of T1 lines. No longer did craftspeople have to go down in manholes to install repeaters every 6,000 feet. A T1 line could be provisioned (in many cases) simply by installing an HDSL modem at the central office and one at the customer premises.

Even though HDSL is used to provision a T1 line, the interface to the customer is still B8ZS AMI. The HDSL modem simply converts between AMI and the 2B1Q actually used on the copper wire. Many T1/E1 line interface units (LIUs) take advantage of this fact and only implement what is called "short haul" functionality, or the ability to drive the AMI signal about 650 feet or so, rather than 6,000 feet.

Bibliography

ANSI T1.102-1993. Digital Hierarchy – Electrical Interfaces. Alliance for Telecommunications Industry Solutions (ATIS), 1993. (Available at www.atis.org).

ANSI T1.107-1995. Digital Hierarchy – Formats Specifications. Alliance for Telecommunications Industry Solutions (ATIS), 1995. (Available at www.atis.org).

ANSI T1.403.02-1999. Network and Customer Installation Interfaces – DS1 Robbed-Bit Signaling State Definitions. Alliance for Telecommunications Industry Solutions (ATIS), 1999. (Available at www.atis.org).

Bellamy, John C., *Digital Telephony, Third Edition*. Wiley, 2000. (Excellent book, recommended to anyone with an interest in the digital telephone network.)

ITU G.704. *General Aspects of Digital Transmission Systems – Synchronous Frame Structures used at 1544, 6312, 2048, 8488 and 44 736 kbit/sec Hierarchical Levels*. International Telecommunications Union (ITU), 1995. (available through the electronic bookshop at www.itu.int).

ITU G.711. *General Aspects of Digital Transmission Systems, Terminal Equipments – Pulse Code Modulation (PCM) of Voice Frequencies.* International Telecommunications Union (ITU), 1988. (available through the electronic bookshop at www.itu.int).

ITU G.742. *General Aspects of Digital Transmission Systems, Terminal Equipments –Second Order Digital Multiplex Equipment Operating at 8448 kbits/s and Using Positive Justification.* International Telecommunications Union (ITU), 1972. (available through the electronic bookshop at www.itu.int).

ITU G.751. *General Aspects of Digital Transmission Systems, Terminal Equipments – Digital Multiplex Equipments Operating at the Third Order Bit Rate of 34 368 kbit/s and the Fourth Order Bit Rate of 139 264 kbits/s and Using Positive Justification.* International Telecommunications Union (ITU), 1976. (available through the electronic bookshop at www.itu.int).