Channelization of a Communications Medium

Norman Matloff

University of California at Davis
©2001-2005, N. Matloff

September 27, 2005

Contents

1 Overview

The term **channelization** refers to the sharing of a point-to-point communications medium.[1] For example, many telephone conversations (or in our context, computer-to-computer network transactions) can be submitted simultaneously on a single wire, with each conversation being on a separate **channel**.

The notion of a channel is very closely related to the household concept of radio and TV channels. The frequency spectrum for television, for instance, is divided into subranges called channels, and these correspond to our everyday concept of TV channels. Each channel is used to transmit different information, all simultaneously.

There are three main ways of doing this:

- In **time-division** multiplexing (TDMA),[2]different sources transmit on the line at different times, each taking (very short) turns. This is used in long-distance phone lines, for example.[3]
- In **frequency-division** multiplexing (FDMA), the different sources attached to the line send on different frequencies (e.g. different radio frequencies, or different light frequencies, i.e. different colors). This is used for radio and television transmission, and increasingly for computer-to-computer network transactions.
- In **code-division multiplexing** (CDMA), all nodes on the network send at the same time, on the same frequency, but using different codes. (Think of one node using a 4B/5B code, another using a second kind of code, and so on.) This is used in some cellular telephone systems.

2 TDMA

2.1 Synchronous TDMA

At one end of the point-to-point line we have n source nodes. Messages from these sources are **multiplexed** onto the line: The n sources continually send in **Round Robin** fashion, meaning that they take turns: Source node 0 sends, then source node 1, then source node 2,..., then source node n-1, then source node 0, then source node 1, and so on. Each cycle of n transmissions (one from each source, starting with source node 0) is called a **frame**. The hardware that does the multiplexing is called, not surprisingly, a **multiplexer**.

At the other end of the line there may be a host computer, in which case the n sources are typically terminals. Or there may be a **demultiplexer** at the destination end, distributing the messages from the n sources to n destination entities.

Each source sends the same amount of data—typically set at one byte or one bit—per turn. This way the destination end of the line knows whose data is arriving at any given time. If the unit of data is bytes, for instance, then you know that any byte which arrives at time k, where k mod n = 5, is from source node 5.

[1]That medium could be a physical wire, an optical fiber, a radio link, and so on.

[2] The full formal title is Time-Division Multiple Access.

[3] Technically, this would seem to violate our definition at the outset of this unit, in which said that the various channels operate simultaneously. However, since the turns are so short, each channel's turn is followed very quickly by its next turn, so we have the illusion that it is simultaneous.

A specific example is the T1 lines which are common for long-distance telephone communication in North America. Here n = 24 and the unit of data is bytes. Thus there should be $24 \times 8 = 192$ bits sent on the line per frame.[4]

We say that a T1 line consists of 24 **channels** of one byte each. In TDMA, we are dividing time, instead of frequency range as in the radio/TV case. For instance, the totality of all the third bytes from all frames—that is, all the bytes sent by the third source—would be called channel 2 (the first one being numbered 0).

Suppose a channel is used for voice transmission, sampled at regular intervals and digitized. The sample taken at any given time consists of the numerical value of the loudness of the sound at that time, which we represent as an 8-bit number, i.e. one byte. Recall that the bandwidth of human voice is about 4 KHz. One can prove that this means we need to sample at $2 \times 4,000 = 8,000$ times per second to accurately capture the voice signal. So, on a T1 line we must send 8,000 frames per second, thus $8,000 \times 192$ or 1.54 million bits per second.

What if instead we use some or all of the channels to transmit data? T1 lines are set up to send 7-bit sets instead of bytes, with the eighth bit used for encoding control information. Thus we are sending 56,000 bits per second on each channel.

However, since the clock at the source-end multiplexer will not exactly match that of the destination end, some drift will occur between them. Eventually the destination end will drift ahead of or behind the sources by one entire bit's duration of time, resulting in reception of wrong data. Thus each frame has an extra bit for **synchronization** purposes, so that the two ends can synchronize their clocks at regular intervals. In T1 lines, then, this will be the 193rd bit in each frame. The totality of these bits is then called the **control channel**. In it the transmitter will send the repeating pattern 101010101... If the receiver receives a 0 on that channel when it is expecting a 1, or vice versa, the receiver will know that it has, literally, "gone out of sync"; it will then have to look at several subsequent frames for the 1010101... sequence before getting back on track.

2.2 Statistical TDMA

The problem with synchronous TDMA is that some source nodes may sometimes have nothing to send; they simply are idle during turns when they have nothing to send. This of course wastes the line, which may be leased at a very expensive monthly fee.

Statistical TDMA remedies this by allowing a source node to skip its turn in such a situation. Each time a node has something to send, it places the item on a first-in, first-out (FIFO) queue into the multiplexer. This approach makes better use of the line's bandwidth. Of course we do pay a price, in the sense that each source now must sent its source number in addition to the data, so that the receiver on the other end of the line can determine who sent the data. Thus part of the line's capacity will be used for sending these source numbers, which we did not have in the synchronous TDMA case.

3 FDMA

If we send data in their original form, this is called **baseband** transmission. A much more flexible alternative is **broadband** transmission, in which we **modulate**, i.e. transform the data.

The "AM" in "AM radio" stands for **amplitude modulation**. Here analog data, say voice, is still sent in an

[4]Plus one more bit, to be described below.

analog signal, but in the form of a product of the orginal data and a sine wave (or a cosine wave, which is after all just a sine wave which has been "moved over"). Because the sine wave varies in height, the height of the original signal is thus changed over time, i.e. its amplitude is modified.

Say x(t) is the voice waveform, and consider a radio station at a **carrier frequency** g_c (this is the frequency at which you tune your radio in order to receive that station's broadcast). Then the radio station will transmit the product

$$p(t) = x(t)cos(2\pi g_c t) \tag{1}$$

What is the effect of this? Well, x(t) has a Fourier series, as we saw in our unit on the Physical Layer:

$$x(t) = \sum_{n=0}^{\infty} a_n \cos(2\pi n f_0 t) + \sum_{n=1}^{\infty} b_n \sin(2\pi n f_0 t) \tag{2}$$

Consider one cosine term in that series,

$$a_n \cos(2\pi n f_0 t) \tag{3}$$

and its associated term(s) in the Fourier series for p(t). In other words, what we are asking is, what will happen when this $a_n \cos(2\pi n f_0 t)$ term gets multiplied by $cos(2\pi g_c t)$? Using high-school trig identities, we can show that

$$cos(a)cos(b) = \frac{1}{2}[cos(a+b) + cos(a-b)] \tag{4}$$

so

$$a_n \cos(2\pi n f_0 t) \cdot cos(2\pi g_c t) = \frac{a_n}{2}[cos(2\pi[g_c + nf_0]t) + cos(2\pi[g_c - nf_0]t)] \tag{5}$$

In other words, the component which was of frequency nf_0 in the original voice waveform x(t) has now been transformed into two frequencies in p(t), centered around the radio station's carrier frequency g_c: $g_c + nf_0$ and $g_c - nf_0$.

Now the Federal Communications Commission, which regulates radio stations, will make sure that the g_c values for various radio stations are spaced well apart. So the bandwidth interval for a given station will be a **channel**

$$[g_c - d, g_c + d] \tag{6}$$

for some value of d. You can see that since the bandwidth for human voice is about 4000, setting

$$d \geq 4000 \tag{7}$$

will ensure that radio stations at a given geographical location and having adjacent carrier frequencies will not interfere with each other. Yes, the human voice does have higher-frequency components and these will cause some overlap between stations, but these components are so weak that the listeners of one station will not hear those of the neighboring station.

There are many refinements to this which are used in actual practice,[5] but the basic principles are clear: If we use broadband trasmission (on any medium, not just radio or microwave), we can send many signals at different frequencies, with the spacings between adjacent frequencies being determined by the the highest frequency (in the effective bandwidth) in the original signal, such as 4×10^6 in the "101010..." example in the unit on the Physical Layer. This is called **frequency division multiplexing**.

Note that there is no fundamental difference in efficiency between TDMA and FDMA for data transmission. In TDMA with G channels, a given node sends only 1/G th of the time; in G-channel FDMA, a given node sends full-time, but only at 1/G th the speed (since it has only 1/G th of the medium's bandwidth).

When done in an optical-fiber medium, FDMA is called Wave-Division Multiple Access (WDMA). Each channel uses a different color of light.

4 CDMA

4.1 Basic Concepts

Here each data bit will be sent as G transmission bits. (Recall that in Manchester coding each data bit is sent as two transmission bits, and in 4B/5B coding each group of four data bits is sent as five transmission bits.) All stations which have data to send will send simultaneously, using the entire bandwidth of the line. (CDMA is called a **spread spectrum** method, alluding to the fact that the bandwidth of a given signal, i.e. its "spectrum," will be much broader in its transmitted form than in its data form.)

The stations are sending to each other. The best example of this is CDMA-based cell phone systems, where different nodes have people talking to each other.

Suppose we have N nodes. Each node j will have its own code, a set of G numbers $c_{j0}, ..., c_{jg}$ where g = G-1 and each c_{ji} is +1 or -1. Denote the data bit sent by node j as d_j, treated as +1 for a 1 bit and -1 for a 0 bit. Then node j will send G transmission bits $d_j c_{j0}, ..., d_j c_{jg}$ in order to send the one data bit d_j. (For simplicity, we will assume that all nodes are now sending, though it may be the case that only some of them are currently sending.)

For example, suppose have N = 4 nodes, with G = 6. Say for instance that node 2's code is (1,-1,-1,1,1,-1), i.e. $c_{20} = 1$, $c_{21} = -1$, $c_{22} = -1$, and so on. If node 2 wishes to send a 0 data bit, i.e. d_2 =-1, then it would send the G transmission bits -1, 1, 1, -1, -1, 1.

The code used by a node is set to be **orthogonal** to those of the other nodes. This is in the sense of linear algebra: The "dot product" between node i's code and node j's code must be 0,

$$\sum_{m=0}^{g} c_{im} c_{jm} = 0 \tag{8}$$

for all cases of $i \neq j$. In our example in the preceding paragraph, for instance, one possibility for our choice for the code for node 1 might be (1,1,1,1,1), which would satisfy orthogonality with node 2, since the dot product of this with (1,-1,-1,1,1,-1) is 0. Note again, we must choose the codes so that there is orthogonality between all pairs of nodes. (We will discuss how to do this later.)

[5]For example, we can use a **single sideband** technique; this means that the radio station would only transmit the frequency range $[g_c, g_c + d]$ (which is no loss of information, since the Fourier components in this interval mirror those in $[g_c - d, g_c]$). In this way, we save half the bandwidth of the radio spectrum, and can fit twice as many stations in.

We saw above how a sending node encodes a bit. But how does the intended recipient decode it?

Consider a node, say node r, which is currently receiving from node s. The receiving node r knows node s's code (which it has been notified of previously, say on a special channel). Keep in mind that node r will receive G signals, each of which is a sum of the signals sent by the various nodes, not just the signals sent by node s. Node r's job, then, is to somehow extract the bit node s sent, out of these G sums.

Specifically node r (and, for that matter, each of the N nodes) samples the medium at G transmission bit times, thus receiving the G quantities $w_0, ..., w_g$, where

$$w_k = \sum_{i=0}^{n} d_i c_{ik}$$

where n = N-1. In order to extract d_s, the bit sent by node s, node r will now take the dot product of $w_0, ..., w_g$ with node s's code, $c_{s0}, ..., c_{sg}$. Let's see what we get:

$$v_s = \sum_{k=0}^{g} w_k c_{sk} = \sum_{k=0}^{g} \left(\sum_{i=0}^{n} d_i c_{ik} \right) c_{sk} = \sum_{i=0}^{n} d_i \sum_{k=0}^{g} c_{ik} c_{sk} \tag{9}$$

The inner sum is 0 for all i except i = s; in that latter case, the sum is g. So, $v_s = Gd_s$. So in order to achieve its goal of finding d_s, node r simply computes v_s and divides by G.

4.2 Choosing the Codes

4.2.1 Welsh Matrices

Welsh matrices provide an easy way to generate sets of orthogonal codes for G equal to powers of 2. They are defined recursively as follows:

- W_1 consists of the 1x1 matrix (-1).
- Given W_m, the 2mx2m matrix $W_{2m}$is defined by

$$W_{2m} = \begin{pmatrix} W_m & W_m \\ W_m & \overline{W_m} \end{pmatrix}$$

where the overbar here means logical negation, in this case changing 1 to -1 and vice versa. The G rows of W_G then form an orthogonal set of codes.

For example,

$$\begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 \\ -1 & +1 & +1 & -1 \end{pmatrix}$$

The four rows of this matrix do indeed form an orthogonal set.

4.2.2 Randomly Generated Codes

In this approach, the G code values for a node are generated at random (typically by circuitry in the hardware), like flipping a coin G times. In doing so, we give up the condition of exact orthogonality; Equation (8) will usually not hold.

However, for $i \neq j$ we will have that $c_{ik}c_{jk}$ will be equal to +1 or -1 with probability 0.5 each, and because of the "law of averages," most of these +1 and -1 values will cancel each other out. Thus in Equation (9) we will have $v_s \approx Gd_s$. In other words, if d_s is 1, v_s will come out near G, while if d_s is -1, v_s will come out near -G. The probability that v_s will be negative when d_s is 1, or positive when d_s is -1, will be low, so we can simply take d_s to be 1 or -1, depending on whether is negative or positive; the error probability here will be low (and, as always, we use error-detection methods to minimize the chance of undetected errors).

4.2.3 Comparison of Exact and Random Methods

One might at first think that exact methods, such as the one using Walsh matrices, are superior. But exact methods require exact timing synchronization among nodes, which may be difficult to achieve. Thus random methods have their place too, and both approaches are used in practice.

4.3 Comparison of CDMA to TDMA and FDMA

The reader might at first think that CDMA is inefficient, since it takes G transmission bits just to send out one data bit. But this is no different from having G channels in TDMA; in either case, a given node is sending only one data bit every G time slots. And we have previously noted the basic equivalence of TDMA and FDMA.

4.4 Security

One advantage of CDMA is security. Since each +1 is changed to +G and each -1 is changed to -G, we can set the +1 and -1 to low signal levels, i.e. low power. That would make it harder for eavesdroppers to pick up node s's data.