UCLA CS 118: Sept 2024

Computer Networks: Physical Layer Lectures

In the first lecture, we finished our overview of networks. Now we move bottom up through the layers, starting with the physical layer. Recall that the physical layer is the actual transmission medium for bits, usually through some form of wire. In the Hazel's hats analogy, it corresponds to the type of plane or the form of balloon used to transport mail.

We will roughly use 3-4 lectures for most layers except the physical layer as its of less interest and use to CS students. We will aim to finish the Physical Layer in 2 lectures.

1.0 What is the Physical Layer?

The physical layer is a bit pipe that is used to connect one sender with possibly multiple receivers as shown in Figure 1. The bits may be lost or corrupted with some probability; thus we need mechanisms to recover from such errors at higher layers. The bits are sent in the form of energy (e.g., electricity, light) that can *travel from the sender to a distant receiver*.

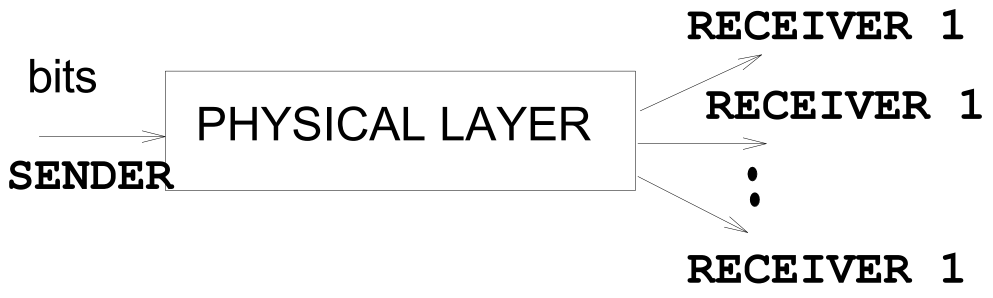


Figure 1: The physical layer as a bit pipe between a sender and possibly multiple receivers.

A familiar example of a bit pipe is sending Morse Code to possibly multiple receivers (for example if the sender is signaling with a flashlight that can be seen by multiple receivers). We can divide up the issues associated with using Morse code to send bits into the following categories:

- **Fundamental Limits to Transmission**: It is easy to see that if the sender tries to signal beyond a certain rate, the receiver will get confused between symbols and be unable to decode. Essentially your brain-eye system can process signals only at a certain rate. If the sender signals faster than this rate, the receiver can be processing a signal when it receives the next one. The result is that the receiver gets confused between symbols in a phenomenon known as Inter Symbol Interference (ISI). Similarly, noise (e.g., from other reflections) can affect the decoding.
- **Technical Aspects of the Medium:** For example, the sender could use a flashlight to send bits by turning it on and off, or use a semaphore (in which case it may send multiple bits). The technical aspects of the particular medium do matter (e.g., flashlight battery lifetime, ability to see semaphores at night etc.)

- **Encoding and Decoding Information**: Encoding in Morse code consists of using a code to send characters. Unlike ASCII, it is a variable length code. However, decoding is trickier. A long series of dashes could be interpreted as twice the number of dashes if the receiver thinks the sender is going twice as fast. In Morse code, the receiver has to know or learn how fast the sender is going in order to recover the sender bits. This is the problem of "clock recovery"; after knowing at what rate the sender is sending, the receiver can interpret the received signal and decode it. Morse code can have a simple clock recovery scheme if the receiver and sender have agreed beforehand on the rate and they both have accurate stopwatches. However, there is still the problem of starting correctly (getting in phase). The problem gets worse in high speed communications when the stopwatches of sender and receiver drift over time.

2.0 Physical Layer Sublayers

As with the Morse code example above, we divide up the problem of the physical layer into three sublayers as shown in Figure 2. Each sublayer deals with an independent problem and has independent concerns

- **Transmission Sublayer**: The bottom sublayer describes the essential properties of the media (e.g., frequency response, bit error rate) that influence and limit transmission rates. This can be studied almost independently of the particular kind of media used.
- **Media Dependent Sublayer**: The middle sublayer describes properties of particular media — e.g. satellites, coaxial cable, fiber. The particular properties do influence protocols and are worth knowing. For example, satellites have a large propagation delay and this necessitates different protocols.
- **Coding and Decoding Layer:** The top sublayer is about how information from higher layers is encoded and decoded. Decoding once again requires the receiver to understand how fast the sender is sending and figure out when the sender starts ("getting in phase"). These problems are more severe than in the Morse Code example because of the large bit rate (say 100 Mbps) which means that a small error in timing at the receiver can cause the receiver to go out of synch. To help the receiver, the sender adds some extra bits to its input stream to help the receiver get into and stay in synch. Thus the encoded bit stream in Figure 2 may be different from the input bit stream.

  Notice an important idea here. We are using layering to understand and separate concerns within the physical layer. The physical layer need not be implemented this way but layering is a tool for understanding. Layers can be studied and combined independently.

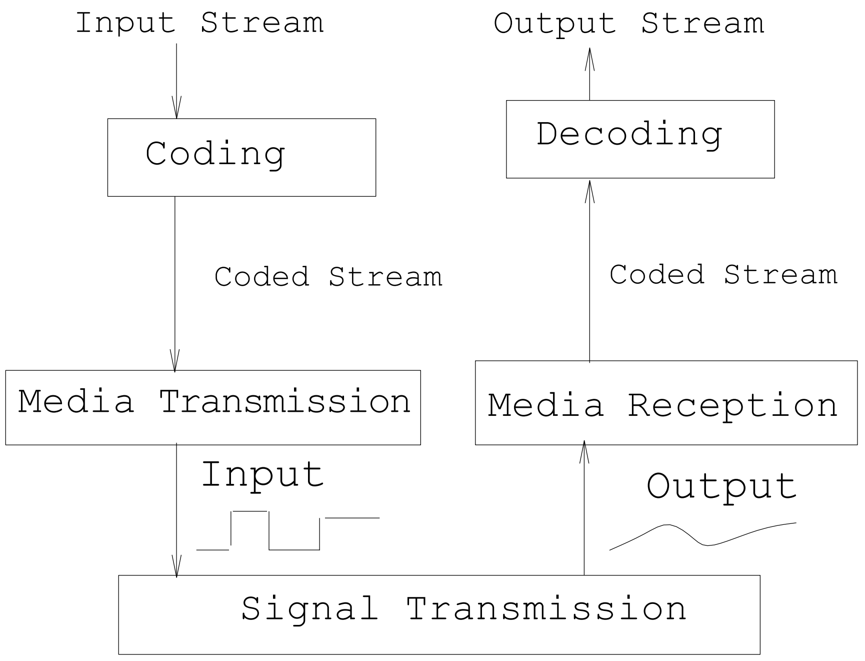


Figure 2: Physical Layer Sublayers

We will use sublayering again to understand the Data Link, Routing and other layers in later parts of the course.

We will study the sublayers of the physical layer roughly bottom up. We will start with the fundamental transmission sublayer which talks about the basic issues of sending signals and bits over physical channels and the limits to the maximum bit rate we can achieve. Next, we will talk about the topmost sublayer: the clock recovery sublayer and show how we encode bits to force *transitions* that help the receiver do *clock recovery*. Finally, we talk about the specific types of media (i.e., fiber, satellites, etc) and how each of them have their advantages and disadvantages.

3.0 Transmission Sublayer

The goal of the physical layer is to send a sequence of symbols (for now 0's and 1's) from a sender to a receiver. Since the receiver is far away from the sender and they do not have a common memory, the most common way is by sending energy (e.g., light, electricity) over a channel (e.g., fiber, cable). There are many ways to code bits using energy (e.g., we can sing high to encode a 1, sing low to encode a 0).

For the next few sections we will consider the simplest coding scheme called On-off coding. Thus we code a 0 by sending no energy down the channel during a "bit time"; we code a 1 by sending energy down the channel. Our flashlight is an example where we send a 1 by turning light on. However, we have the following

Problem: Real channels distort input energy signals. The output signal is not the same as the input signal. This leads to the immediate question. How does distortion affect maximum bit rate? We will show that the inherent inertia (i.e., sluggishness) of the channel affects the maximum signaling rate (Nyquist limit) and the inertia plus noise affects the maximum bit rate (Shannon limit).

These are the issues we cover in this section. Thus the rest of this section will have 3 subsections. First, we talk about the physical layer concept of bandwidth, then about the Nyquist limit and finally we finish with the Shannon limit.

3.1 What we mean Bandwidth at the Physical Layer

A signal is something that carries a measurable amount of energy that varies with time. This could be light, sound or electricity. The easiest way to draw a signal is to draw a graph of how the energy varies with time. The magnitude of the energy is called the amplitude and we plot it on the y-axis while time is shown on the x-axis. For electricity, we might measure amplitude in volts; for sound in decibels, for light in terms of light intensity.

A channel is a physical medium that conveys energy from a sender to a receiver (e.g., a fiber link for light, a copper wire for electricity). Of course the output signal typically distorts the input signal. So we need to understand what a given channel will to do to a particular input signal using Fourier Analysis which we will not study. However, the big picture is that if we forget about noise, most channels are "nice" to sine waves. A sine wave of frequency *f* is always scaled by a fixed factor *s(f)* and phase shifted by a fixed amount *p(f)* regardless of amplitude or time. Thus we can completely describe a channel by plotting the values of *s(f)* (frequency response) and *p(f)* (phase response) for all values of frequency f .

To find what happens to arbitrary signal S, we use Fourier Analysis to rewrite S as a sum of sine waves of different frequencies. Then we use the frequency and phase response *of the channel* to see effect of each sine wave. Finally, we add the scaled sine waves to find the output signal at the other of the channel using inverse Fourier analysis.

Thus we can completely characterize a channel by doing an experiment where send in a large number of sine waves of different frequencies and observe the resulting scaling and shift. This can be used to plot a frequency and phase response for any channel whether it's a wire or a radio channel. An example is given in Figure 3. The figure shows that the channel basically scales all frequencies between 200 and 1200 Hz by a factor of 2, and does not pass through (i.e., scales down to 0) all other frequencies.

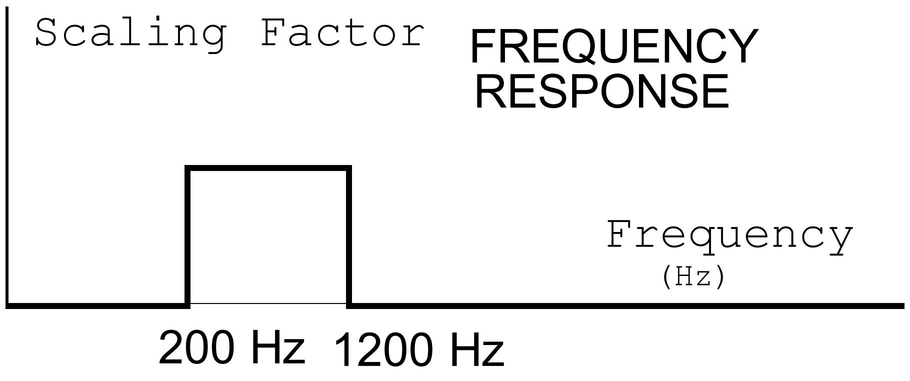


Figure 3: The frequency response of a highly simplified and ideal channel

This figure shows that this simplified channel is a *band pass* filter — i.e., it only passes through a narrow band of frequencies. The *bandwidth* of the channel is the width of its pass band, in this case 1000 Hz. Most real channels have a much less clearly defined band region (instead of vertical lines delineating a band you have a curved fall off) and one has to define the band in some arbitrary way (e.g., the range of frequencies for which the scale factor is within 80 percent of the maximum scale factor). Many speakers and other sound equipment will show similar curves to show their fidelity over a range of frequencies; bad speakers have a poor range and will tend to distort sounds of high pitch

Most real channels will not pass through frequencies beyond a certain cutoff point. So what happens if the input signal has a frequency greater than this cutoff frequency? For example, suppose the channel does not pass through frequencies greater than 1200 Hz and we send the sequence of bits 0101010… at the rate of say 3000 bits per second. The bit sequence creates a periodic signal of 1500 Hz (Hz just means how many reps per second) which the channel cannot pass through. The result is that the output will be significantly distorted.

In effect, most channels are sluggish (they take time to respond to sudden changes) because they turn a deaf ear to higher frequencies in the input signal. Thus lower bandwidth channels are more sluggish. This can seen by Fourier Analysis as show in the following picture (we don't show the details of how the Fourier Analysis is done, just the result).

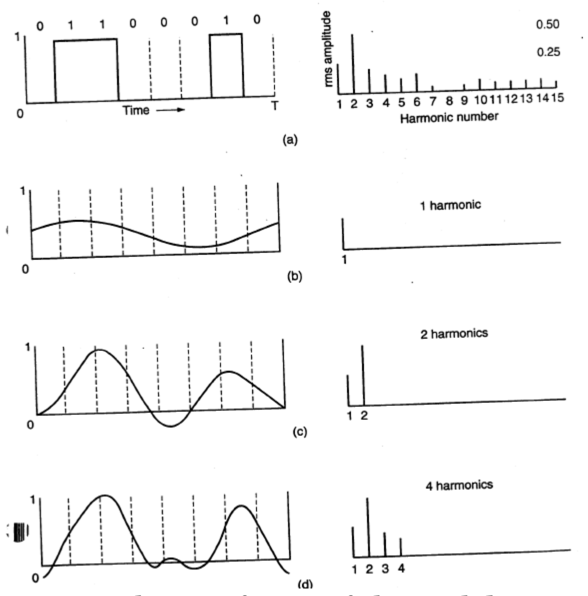
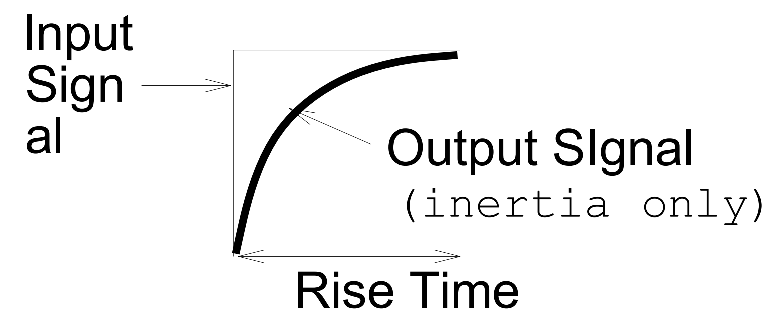


Figure 4: The two facets of channel distortion: *inertia* and *noise*

In the picture above we show what happens to a bit pattern sent by a sender as we gradually increased the channel bandwidth. The Fourier Analysis is shown on the right. When we start at the base frequency, the output is only a sine wave of frequency that "approximates" the square sender bit pattern. Instead of a rapid rise and fall, the sine wave gradually rises and falls. If we increase the bandwidth by a factor of 2x this allows the channel to pass through another sine wave of twice the frequency. This allows the edges of the output signal to be sharper. Essentially, we are like an artist adding more and more detail with finer and finer brushes. Finally, when we increase the bandwidth by around 3x, we get something that looks close to the pattern. But we don't really need to go that far as long as we can recover the bit pattern of the sender which we probably can with a bandwidth of 2x.

Finally, what about noise? So far we have ignored noise. There are different models of noise for different channels depending on the source of noise (e.g., thermal noise, interference etc.). One simple and common model is white or thermal noise. We assume the noise is uniformly distributed at all frequencies and its amplitude is normally distributed within a frequency. This is shown in Figure 4. The lack of channel bandwidth leads to sluggishness as shown in the top figure. If the noise has a maximum amplitude of *N*, we can imagine drawing a band of *N* around the noise-free channel output. Now at each point in the old curve, randomly select (according to a normal distribution) a new point. Connect the resulting points to form a much more jagged curve than the original curve. This description is a gross simplification but it should give you some intuition without doing a whole course worth of communication theory.


Input
Sign
al
Output SIgnal
(inertia only)
Rise Time

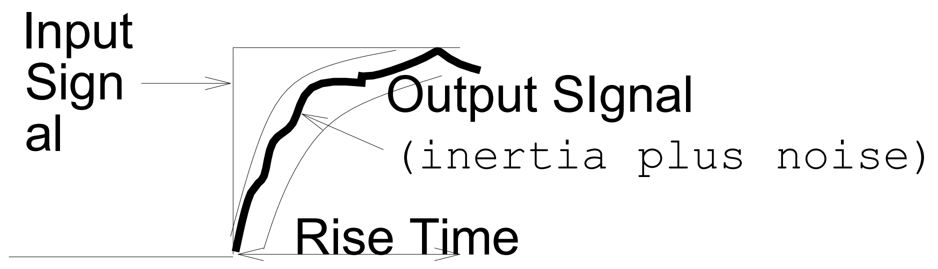

Input
Sign
al
Output SIgnal
(inertia plus noise)
Rise Time

Figure 4: The two facets of channel distortion: *inertia* and *noise*

3.2 Nyquist Limit: How fast can we send symbols because of bandwidth limits

We saw in the last section that if we sent a signal which contains higher frequencies than the channel can pass, then the output will be distorted. Similarly, if we transmit bits at a rate that exceeds the channel bandwidth, then the output will be distorted and the receiver will not be able to recover the bits. In the next two sections, we will make this intuition more precise.

To do so, we need to start with a model of how we transmit information. Assume that we use the on-off method of coding information with a 1 represented by energy, and a 0 by no energy. Then, as we show in Figure 5 (the figure assumes that the channel has adequate bandwidth), the output signal is a sluggish form of the input signal. Now, receivers cannot physically monitor the signal at all times. Instead they sample the output signal periodically. If the receiver samples the output signal at roughly the middle of a bit period, then the receiver can decide whether the bit is a 1 or a 0 by just checking if the received energy at the sample point is above a threshold. (How the receiver figures out the points to sample is the topic of the next subsection, on clock recovery).

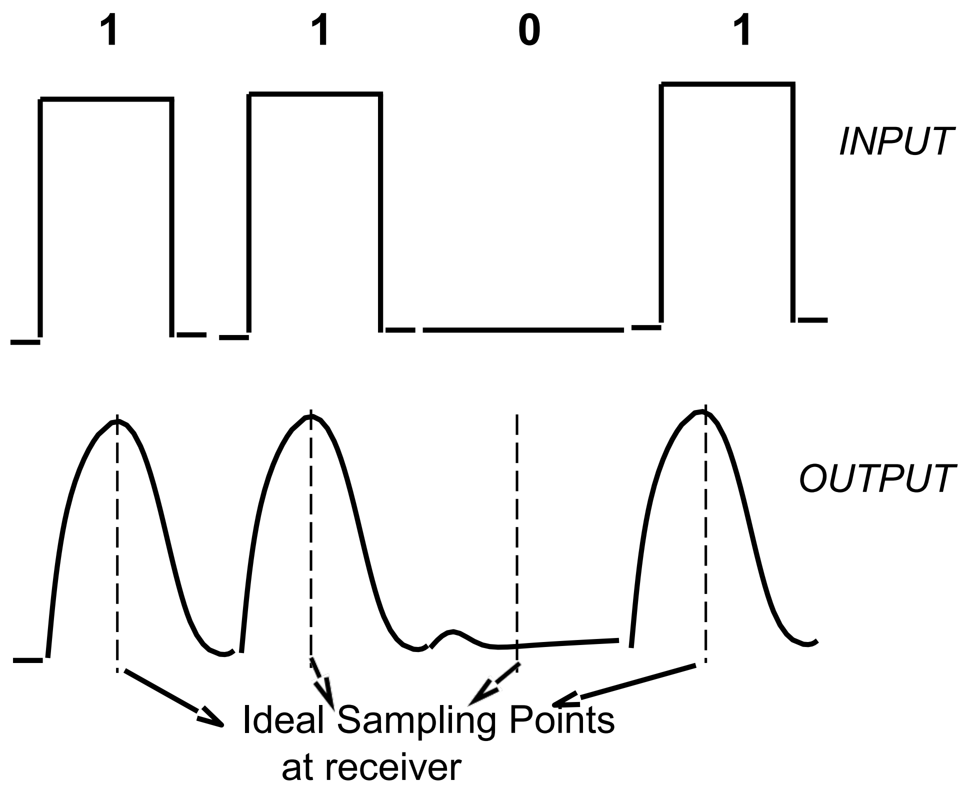


Figure 5: Receivers recover the bits in the input signal by *sampling* the output signal as close to the middle of the bit period as possible.

Suppose the channel has bandwidth $f = 1/T$ and the channel receives an input signal pulse (that represents a 1) of width T. This is shown in Figure 6. If the input is a 0, then the input is sent as 0 Volts, and the output is always 0 Volts for all time. Then the channel response is a damped sinusoid. It begins as a sine wave that approximates the input pulse, however after the input pulse stops, the output continues to "ring" for a while as a sine wave of the same frequency except with a maximum amplitude that gradually dies away to zero. Figure 6 only shows the overshoot from T to 2T but in reality it keeps dying out. This is called the *sinc ()* function and comes from Fourier analysis.

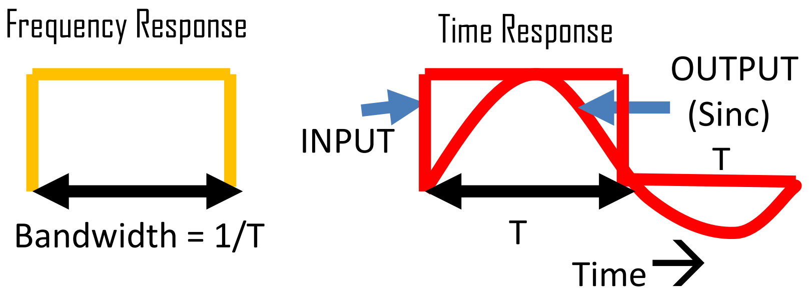


Figure 6: The output of a square wave representing a 1 is a *sinc ()* function that rises and falls in time T if the channel has a bandwidth of 1/T

Since the energy of the output response lasts at least T (after that there is some energy but it dies away exponentially), one might think that it is safe to send the next bit T seconds later. Indeed, that is true. However, Nyquist noticed that we can even shorten the first bit to $T/2$ and send the next bit $T/2$ seconds later with *zero* inter-symbol interference even though the second bit is sent before the first has settled because the zeroes of earlier bits line up with the peak of the current bit (see Figure 7) . Thus we get the Nyquist limit that the theoretical maximum bit rate is 2/T which is twice the bandwidth B.

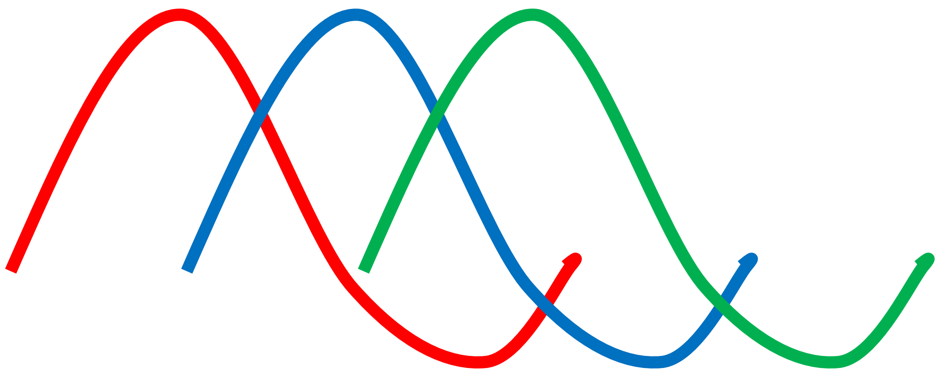


Figure 7: The output of a square wave representing a 1 is a *sinc ()* function that rises and falls in time T if the channel has a bandwidth of 1/T

If we try a faster rate, we soon will find intersymbol interference. Thus the Nyquist limit says that we cannot send symbols faster than a rate of $2B$ per second (if the channel bandwidth is B) without causing intersymbol interference.

3.3 Shannon Limit: How fast can we send bits due to bandwidth *and* noise

In the previous subsection (see Figure 6) we sent 1 bit per pulse. If we were dealing with electricity, we might have a 1 be $5V$ and a 0 be 0 V. However, we might be able to send pulses of various levels. For example: 0, 2, 4 and $5V$. In that case each pulse may actually have 4 possible values. Thus we can send multiple bits per pulse. With 4 levels, we can send two bits per pulse: 00 could be $0V$, 01 could be $2V$, 10 could be $4V$, and 11 could be $5V$. In general, if we allow L levels, we can have $\log L$ bits per pulse, where the log is taken to base 2.

Thus in the future, we use the word *symbol* to denote an individual pulse of energy. We call the rate of sending pulses the *signalling rate* or the *baud rate*. The bit rate is the baud rate multiplied by the number of bits per symbol.

That immediately raises the question: can't we transmit at as high a bit rate as we like by dividing each pulse into as many levels as we want? We can then transmit at mega-tera-gigabits. Right?

Not really. Essentially all real channels have noise that prevent a signal from being divided into too many levels. If they are, then noise could cause one level to be confused with another. In the example coding with 4 levels, if we have noise of 0.5 volt, the receiver would be in trouble if it receives a 4.5 volt signal. Is this a 10 (which has been corrupted by noise from 4 to 4,5 V) or a 11 (which has been corrupted by noise from 5 to 4,5 V)?

Figure 8 shows that if we have a maximum signal amplitude of S and a maximum noise amplitude of N, then we cannot be safe in the worst case unless the levels are separated by at least $2N$. (This prevents confusion between a level that has moved down by N due to noise and an adjacent level that has moved up by N due to noise.) While the figure shows 5 levels, in general, if we have a maximum signal strength of S and we have a gap of at least $2N$ between levels, we can have at most $S/2N$ levels. Thus we can transmit at most $\log(S/2N)$ bits per symbol.

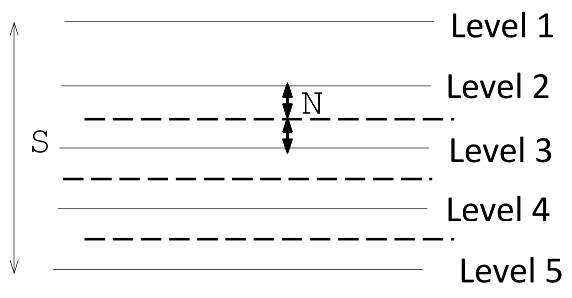


Figure 8: If the signal strength is S and the noise is N, we cannot transmit more than S/2N distinct values per symbol or noise can corrupt one symbol to another

If we combine this with the Nyquist limit, which says we cannot send symbols at a rate faster than $2B$ per second, where B is the channel bandwidth, we can guess that we cannot send faster than $2B \log(S/2N)$ bis per second. The actual Shannon theorem is a bit different and a lot deeper than this simple analysis. This is because it applies to all forms of encoding bits (not just on-off encoding), it applies to cases where the noise is probabilistic, and it applies to cases where the Nyquist limit is violated. Shannon also showed that there was *some* coding scheme that could achieve this maximum channel bandwidth and today's turbo codes are getting close.

4.0: Clock Recovery Sublayer

We showed how the receiver obtains bits from the sender in Figure 5. In order to do so the receiver needs to sample the output signal as close as possible to the mid bit periods (the dotted lines) because the signal reaches its highest value at these times and is less likely to be corrupted by noise. Then the receiver applies some threshold function (e.g., if a 1 is encoded by $2V$ and a 0 by $0V$, then the receiver could decide that anything above $1V$ is a 1, anything below $1V$ is a 0.) Effectively the sequence of sampling instants is called the receiver clock.

The problem of *clock recovery* is to have the receiver dynamically adjust its receiver clock in order to track (as closely as possible) the middle bit instants of the sender. Clock recovery consists of two basic tasks: getting in phase (or figuring out when to sample the first bit) and staying in phase (keep the successive sampling instants synchronized).

The second problem would be very hard if the receiver did not have any information about the clock period of the source. For example, the same signal shown in Figure 8 could be decoded in two completely different ways depending on the length of the bit/clock period. Thus we assume that the receiver roughly knows the clock period of the sender by some *a priori* arrangement (such as using similar quartz crystals in both sender and receiver).[2]

Thus the first problem (getting in phase) is always a issue. The second problem would not be an issue if the receiver and sender had exactly identical clocks. However, all clocks are slightly inaccurate and, over a period of time, the receiver and sender clocks will drift. Even if the sender and receiver are both installed with a 100 MHz clock, the receivers clock may tick slightly faster (say by 1 percent) than the senders, and so drift out of synchronization over long periods. Thus, over a long period, with no further correction, the receiver clock will drift arbitrarily far away from the sender clock. The result is that the receiver will sample the signal at the wrong instants. In particular, if the receiver samples the signal during the start of a bit period during a transition from a 0 to 1 or vice versa, the receiver can get an arbitrary value, which can result in a bit error.

To solve the first problem, the sender generally sends a well-defined sequence of "training" bits before it sends a stream of data bits. These bits get the receiver in phase. To solve the second problem, we can either rely on the accuracy of the receiver clock (and limit the number of bits sent to a small value) or we can ensure that the data will always have *transitions*. Transitions are changes in signal amplitudes (for example, a change from 0 to a 1). Transitions provide cues that help the receiver stay in synch after getting in phase, even over a long stream of consecutive bits. It may help to think of the receiver clock as a stopwatch that can be started and restarted (and even sped up and slowed down!) based on transitions.

However, real data bits do not either provide either training bits (sometimes called *start bits* or a *preamble*) or guarantee transitions. To solve these problems, the top sublayer in the physical sublayer (Figure 2) is a coding sublayer that encodes the raw data bits fed to it by the Data Link Layer and codes it to add training bits and transitions.

Thus we can define the following terminology. The physical layer will be handed some sequence of bits to transmit. These bits will be coded by adding a preamble (or start bits) at the start and possibly coding the data bits to force transitions. We will call the result a *frame*.

We now study two generic kinds of coding techniques: first asynchronous coding, and then several examples of so-called synchronous coding. The former uses a small frame (10-11 bits) while the latter uses larger frame sizes and a larger preamble. We study how the receiver might do clock recovery in both cases and discuss advantages and disadvantages.

4.1: Asynchronous Coding

In asynchronous coding, the sender physical layer is given a character of data to transmit. This data character can be 5-8 bits and may also include a parity bit. The most common example is the ASCII code in which letters of the alphabet and numbers can be encoded in 7 bits. In addition most asynchronous codes also add an additional parity bit.[3]

The actually coding consists of the following rules. First, low amplitudes are used to encode a 1, and high amplitudes are used to encode a 0 (this is the opposite of the convention we have followed so far!). We add a parity bit to each character and then "frame" the character with an extra start bit and 1-2 stop bits. The start bit is a 0 and the stop bit is a 1. Thus regardless of the value of the data character, the first bit sent is a 0 and the last bit is a 1. This is shown in Figure 9. Notice that the start bit is a high amplitude because of the peculiar way we code bits!

The times between the sending of characters can be arbitrary and hence this form of coding has been called *asynchronous*. The word synchronous refers to a process that involves a clock or some other time constraint; asynchronous implies that the time between *characters* is arbitrary. However, please note that the time between *bits* is not at all arbitrary but is controlled by the sender clock. If the time between characters is arbitrary, what happens to the line between characters? Essentially, the line amplitude is low (i.e., a 1).

Let's see how we solve the two clock recovery problems. Recall that the last bit of a coded character is a 1 and the line reverts to a 1 level between characters. Thus since the first bit of a character is a 0, *we always guarantee a transition at the start of a new character* regardless of whether the time between characters is zero or some other arbitrary time. Thus the receiver gets into phase by watching for the low to high transition.

The second problem, that of staying in phase, is solved by a brute-force approach. We assume the receiver and sender clocks are fairly close. Since there are only 10-11 bits in a frame, even if the receiver clock drifts, the receiver sampling points cannot drift too much by the end of a frame. More precisely, suppose the receiver begins sampling at the middle of the start bit. Then, even if the receiver clock differs from the sender clock by 5 percent, over 10 bits the receiver sampling point will not drift by more than 50 percent. Thus the receiver sampling point will still stay within the corresponding sender bit period. In practice, we would want the receiver clock to be more accurate so that the last few bits are still sampled close to the middle of a bit period (and not close to the edge of a bit where unpredictable things can happen).

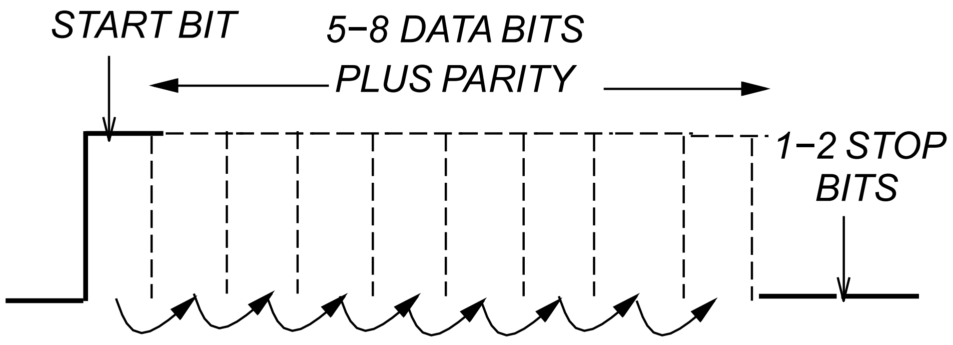


Figure 9: The extra start bit allows the receiver to start sampling (see arrows for ideal sampling points) and then we rely on bounded clock drift.

Figure 9 shows the ideal sampling points. Given this picture, it is easy to model the receiver bit recovery process by a piece of pseudo-code (most real UARTs and other devices tend to do this process in hardware). The code models the waiting of timer by calling a routine *StartTimer()* and the sampling of a signal by calling *SampleSignal()*.

Receiver Code:

```
Data Structure:
 C[0..10]; ARRAY, to store bits in current character

On Transition:
        StartTimer (1/2 bit)
        For (i = 0 to 10) do
           Wait (TimerExpiry);
            C[i] = SampleSignal(); StartTimer (1 bit)
        End;
        If (C[0] = 1) and (C[9] = C[10] = 0) then Output C[1..8]
```

4.2: Synchronous Coding

In asynchronous coding, the receiver gets locked in phase when the voltage goes from low to high (start bit). To make this process reliable, we need to have a fairly large idle time between characters for the receiver to get locked in phase for each character. This slows transmission and limits it to low rates (e.g., 9600 bps). Thus asynchronous coding has two overheads: there is clearly the overhead to add an extra start and stop bit for every 8 bits of data; in addition there is the extra time needed for reliable detection and starting up the receiver clock for each character. (In some sense, we need to restart the receiver stop-watch for every character.)

The so-called *synchronous coding* techniques attacks both these problems by using a frame size that is (typically) much larger than a character, in the order of thousands of bits. In this case, the overhead of the preamble (i.e., start bits) and postamble (i.e., stop bits) is amortized over a large number of bits and can become small. In addition, the phase information learnt from the preamble is used to clock the remaining bits in the frame as well. Thus the inter-frame time needed for reliable phase detection is again paid once for a large group of bits. The bits within a frame can now be sent at very high speeds.

However, if we make the frame size larger, we can no longer use the brute force approach of relying on the bounded inaccuracy between the sender and receiver clocks to *stay in phase* till the end of the frame. Thus all synchronous coding techniques will also code the data bits to guarantee transitions *within* a frame and not just *between* frames as in asynchronous coding.

The terminology is somewhat unfortunate. Both asynchronous and synchronous coding methods transmit data synchronously (i.e., using a fixed clock) *within* a frame. Both are asynchronous (i.e., there can be arbitrary intervals of time) between frames. The only difference is the size of a frame, which in asynchronous is limited to a few bits, while in synchronous it is much larger.

Types of Synchronous Coding: As in asynchronous coding, we can always represent a 1 by one voltage, and a 0 by another. This is called NRZ (non-return to zero) coding. We could always prefix a frame of data by an alternating string of 1's and 0's to get a good preamble to get into phase synchronization. However, the main problems with this coding scheme is that a long series of 1's (or 0's) will contain no transitions and the receiver will not be able to stay in synch.

A more popular scheme (used in Ethernets) is called Manchester encoding (see Figure 10). Here a 0 is encoded as -1.5 V for half a bit followed by 1.5 V for the remaining half bit. A 1 is encoded symmetrically as 1.5 V followed by -1.5V. In some sense, in Manchester we are encoding a 0 by 01 in NRZ, and a 1 by 10 in NRZ. Effectively, we are sending two coded bits for every real bit transmitted. Thus the transmission efficiency of Manchester is poor (real bits to coded bits, 50 percent efficiency).

However, the great advantage of Manchester is that it is *self-clocking*. This refers to the fact that every bit, regardless of its value, provides a transition that can be used to sample the bit value. For example, if we knew approximately where the half bit period was, our decoding algorithm could be: *Wait for a transition at roughly the mid bit. Then start a timer for a quarter bit after that. If the value is above some threshold, it's a 0; else it's a 1.* In fact, many simple Ethernet clock recovery circuits used this algorithm.

This algorithm does not really help to get into phase because a string of consecutive 1's and a string of consecutive 0's both produce almost identical waveforms (with transitions at every half bit) except for a phase shift.[4] To solve the phase problems, most Manchester encoders start a frame with the preamble 010101…01. This string has transitions only at the mid bits, which allows the receiver to easily get in phase.

(Digression: The Ethernet frame format starts with a string of this sort followed by the bits 11. The reason for this is that even if the sender sends a fixed length string of 0101…01, the receiver may get in phase after an unknown number of bits and lose the first few bits. The last 11 at the end helps the receiver realize that the preamble is over and the data bits are beginning.)

Manchester lack of efficiency makes it quite unpopular at higher speeds. Higher speed physical layers (for example the 100 Mbit token rings) often use 4-5 codes, in which 4 data bits are encoded as 5 transmitted bits. This can be used to convert 4 consecutive 0's to 5 bits with at least one 1 in them which can guarantee transitions.

Based on this discussion we can compare different types of codes based on:

Ability to Guarantee Transitions: This is probably the most important factor.

Transmission Efficiency: The ratio of real to transmitted bits.

Signal to Noise Ratio: Given the same maximum and minimum transmission levels, the amount of noise required to render a received signal ambiguous.

Implementation Complexity: This is somewhat qualitative. Schemes like Manchester coding are simpler to implement than, say, 4-5 coding or AMI which require memory of the previous bits.

4.3 Broadband Coding

So far we have only talked of *baseband* coding which is coding using energy levels (say voltage or light). Another popular form of coding used by modems is *broadband coding* in which the information is *modulated* on a carrier wave of a certain frequency. For example, when a modem transmits over a voice line, it is easier if the transmission is based on tones that are in the frequency bandwidth of the line.

The term modulation refers to changing the properties of the carrier to convey the required information. In *Frequency Shift Keying* (FSK), we let a high frequency encode a 0 and (say) a low frequency encode a 1. (Colloquially this is the same as singing high to encode a 0 and low to encode a 1.) Clearly this is done to make the data "sound" like voice so it can pass over a telephone line. In *Amplitude Shift Keying*, we change the amplitude of the sine wave to encode a 1 or a 0. (Colloquially this is the same as singing loudly to encode a 0 and softly to encode a 1.) In phase shift keying, we change the phase of the sine wave whenever we transition from a 1 to a 0 or vice versa. When you change phase, the resulting sine wave appears to be discontinuous. For example, if you are at the peak of the sine wave and you have a phase change, you may instantaneously go to the crest of the sine wave.

Even in these broadband coding schemes that are modulated on a carrier, the issues of clock recovery and Nyquist and Shannon limits still remain. (Although, all the examples we have given motivating these issues used simple on-off encoding.) For example, in FSK we still have to sample the sine wave periodically to decide its frequency at that point (or in that interval). Thus those issues are orthogonal to the type of coding used.

It is probably accurate to describe broadband coding as: *analog coding of digital data* and baseband coding as *digital coding of digital data*. In baseband coding, the source puts out a digital signal (though it becomes an analog signal after channel distortion and noise.)

5.0: Media dependent sublayer: Types of Media

In the example of using a flashlight to send Morse code, we saw that limits to signalling speed corresponded to issues in the transmission sublayer; there were also issues having to do with clock recovery. For the flashlight example, the media dependent sublayer corresponds to issues concerning the technical aspects of the flashlight: for example, battery life, the range of the flashlight and its clarity. These issues clearly affect data communication. If the user is mobile, we prefer a large battery life; long distance communication requires flashlights with a large range.

Similarly, it is worth studying some aspects of the various kinds of media available because:

- *Media Affects Protocols:* The choice of media affects the way protocols have been designed or are evolving. We give examples below.
- *Each Type of Media has its Range of Applicability:* It is important to study and understand the pros and cons of different media in order to design a network.

Thus some minimal information about the technical aspects of media is useful not just for the communication engineers working on the physical layer but for software engineers and protocol designers working at higher layers. We now elaborate.

4.1: Media Affects Protocols

We give some examples of how media has affected (and *is* affecting) protocol design:

Bandwidth and Message Formats: The earliest networks used existing phone lines and were limited to low bandwidth voice links. Thus earlier protocols (e.g., current Internet protocols) made valiant attempts to encode messages to save bits, and to send fewer messages. With the advent of fiber, this is becoming less of an issue and the latest Internet protocols use more spacious formats and are less worried about sending messages.

Broadcast LANs and use of Broadcast: With the personal computer came the invention of the Ethernet (and other local area networks or LANs). The use of coaxial cable in Ethernet made broadcasting essentially free (many stations could listen). This was heavily utilized by protocols that work over such LANs. For example, the Internet protocols use a broadcast mechanism to find the address of an unknown station using the ARP protocol. Other LAN protocols used broadcast to distribute multiple copies of a packet for free (e.g., a distributed flight simulation.) With the advent of fiber, which is inherently point-to-point, broadcast is no longer free. Nevertheless, software simulated broadcast is now considered essential and is now available on the Internet.

Building wiring affects Switching: The cost of wiring is significant because of labor costs and the need for right-of-way. Most buildings are wired hierarchically with lines running from a wiring closet on each floor. Thus a recent trend has been to replace these wiring closets with more active devices like bridges and routers (intelligent hubs) and possibly ATM switches[7]

Fiber to Coaxial Cable, Ethernets to rings: As fiber allows higher transmission rates than coaxial cable, vendors wanted to design higher speed LANs using fiber. Because fiber is point-to-point (one sender and one receiver) and hard to tap, it is hard to build an Ethernet like LAN over fiber. Thus token rings, which consist of a number of point to point links in a ring topology, began to get more popular in standards like 802.5 and FDDI.

Twisted Pair to Fiber, Analog to Digital: The easiest way to send bits over fiber is to turn light on and off, which is inherently digital. Thus as the telephone companies replaced more of their long haul network with fiber, they began to use baseband instead of broadband signalling. However, more recent advances in fiber and all-optical technology are based on analog transmission. This may cause further changes.

Infrared and wireless and low bandwidths again: Infrared technology, while inherently low bandwidth and short range, is becoming popular as a cheap, wireless connection for laptops. The difficulty is that protocol designers can no longer assume high bandwidths everywhere but must assume a large dynamic range. For example, there have been recent efforts to find a standard for compressing the spacious IPv6 (new Internet protocol) headers over mobile links!

The details are interesting but not as important as the message: it is good to be aware of the details of technology and media as they greatly influence protocols. As an analogy, architects are trained to understand the details of types of construction material. Clearly, reinforced concrete and similar materials made skyscrapers possible.

4.2: Types of Media: Pros and Cons

The most common media used for data communications are twisted pair, baseband and broadband coaxial cable, fiber, satellite, microwave, and infrared. We will spend a paragraph discussing each type of media, and then survey their advantages and disadvantages. More details can be found in other texts such as Tanenbaum and Stallings.

Notice that in our Hazel's hats analogy, similar tradeoffs (speed, distance, latency, cost) occur in many types of transport mechanisms. We use planes for long distance and cars for short distances. In some cases, for example, traveling in the mountains, a simple brute-force solution (!) may be to use a donkey.

Similarly, in networking it is important never to forget the brute force solution of shipping a storage device (e.g., magnetic tape, disk) by other transportation methods to the receiver. Tanenbaum has the beautiful example of shipping video tapes overnight by Federal Express across the U.S. Calculations show that the data rate is as fast as the fastest network speed and cheaper than most network transmission (except if the Internet is free for you!). One disadvantage of this scheme is that it has a high *latency* (1 day) if you have a small amount of data to send, a very common feature of most network traffic.[8] However, it is a reasonable solution to the problem of backing up the data at a site to a remote location because of its high bit rate or *throughput.*

Some important aspects of media are: distance or span, speed or throughput, latency, whether the media is wireless or wired, whether the media allows broadcast, the ability to eavesdrop on the media, the ease of installing the media, noise immunity, and the power required for transmission. The pros and cons of the various media are summarized in Figure 12.

Distance influences whether the medium will be used in the local area, office, or backbone networks; speed affects the protocols that can be used over the lines (the backbone networks require high speeds).

Wired media like twisted pair, coax cable, and fiber require right of way in order to install the wire, which is not easy in crowded cities and expensive even in offices and campuses.[9] We have already seen that broadcast media like satellite and coaxial cable make possible cheap broadcast protocols. Most broadcast media have problems with security because it is hard to prevent unauthorized eavesdropping without using encryption.

Ease of installation is an important issue. Again wireless media are excellent bypass technologies in developing countries because they get around the lack of infrastructure and the need for right of way. Among wired media, fiber is light and easy to install. The old bulky Ethernet coaxial cable has since been replaced by lighter weight Thinwire Ethernet. Finally, fiber has excellent noise immunity to electrical noise and disturbances as it carries light; at the other extreme microwaves can be affected by rain, and infrared is easily absorbed by obstacles. Finally, low power radio waves and infrared are useful for wireless computing and laptops because they do not require wires or much power.

Notice that each media type has its niche. Twisted pair, baseband coax, fiber, and infrared are used in the local area and office. Broadband coaxial (e.g., television cable systems) are used in the intermediate (sometimes called metropolitan area and long distance); fiber and satellites are used in the long distance. However, these niches change as technology evolves: for example, broadband coaxial cable is being replaced by fiber in the backbone networks

4.3: Quick Description of Various Types of Media

We now give a quick description of the various kinds of media described in Figure 12.

Twisted Pair: Twisted pair is the standard telephone wire and easily available in the office and the home. Sometimes known as Unshielded Twisted Pair (UTP) this is a bunch of 8 wires twisted together in four pairs to reduce electrical interference. This works because noise tends to go into each pair equally and if the signal is taken as the difference between the signals in each pair, then noise cancels out. The corresponding connectors are RJ-45 plugs. UTP is classified into different levels of quality according to its ability to transmit signals.

Standard Telephone wire is probably Category (or Cat) 1 and was unsuitable for sending data at high rates. Using the standard Shannon Limit and a telephone bandwidth of 4 Hz, it is hard to send data using standard data modems faster than 50 Kbps (without counting compression) and possible at slightly over 100 kbps (with compression).

Two neat exceptions to this rule that you see every day are Ethernet Cables (the so-called Cat x series) and DSL (that some people prefer to Cable). Let's start with Ethernet cables. These are much higher quality twisted pair than telephone cables that effectively allow us a bigger bandwidth and higher data rate. You can't do this over your telephone wiring but you can buy them say at Fry's.

While original Ethernet was carried over coaxial cable at 10 Mbps, so called Cat 3 cable was soon preferred because it is lighted and easier to wire. You can go to Fry's and purchase Fast Ethernet cable as Cat 5 to go up to 100 Mbps. This is because Cat 5 cable can send symbols as fast as 100 million times a second. Recall that Manchester encoding effectively sends 2 bits for every bit. Thus instead, 100 Mbps Ethernet uses 4-5 encoding. Cat 6 (or even enhanced Cat 5) can carry 1 Gbps.

While the Cat 3, 5, and 6 cables are used by users to construct small local area networks, the telephone company also figured out a way to provide fairly high data rates (less than 10 Mbps downstream and much less upstream) using so called DSL or ADSL (Asynchronous Digital Subscriber Line). The key trick here is to realize that the standard twisted pair actually has a lot of bandwidth (it can every carry Ethernet) but it is artifically limited to 4 Khz by the telephone company using so-called loading coils. The loading coils are periodically inserted amplifiers allow telephone calls to travel for long distances. The 4 Khz limit is also not a problem for voice because human voice stays in that range.

To do DSL, they have to remove the loading coils (thus getting a much larger bandwidth), limit you to a small distance from the central office (because there are no loading coils to amplify your signal), and also need to make you asymmetric. Thus the closer you happen to be on the cable path from the central office the higher your bandwidth. Many DSL lines are often terminated by a so-called DSLAM (DSL Access Multiplexor) which then leads on to the Internet.

Coaxial cable: consists of two concentric conductors in which the signal propagates as a wave. Baseband coaxial cable refers to using baseband signaling (e.g., digital signalling as in Ethernet) over coax cable. While this was the original form of 10 Mbps Ethernet, its thick clunky layout has lost out to UTP Cable. However, broadband cable is still very common.

Broadband coaxial cable refers to using broadband signaling (e.g., sending some modulated high frequency signal as in Cable TV.) Baseband signaling has a smaller distance than broadband signaling; but broadband systems require amplifiers to periodically boost the signal. If you have cable at your home its because the cable company has run cable (with some fiber sometimes) to your home. Each cable's bandwidth is divided into multiple 6 Mhz channels using frequency division multiplexing, each of which can carry a TV channel (e.g., NBC, Fox).

However, cable modems work by reserving another 6 Mhz channel (just like a pseudo-channel) for downstream data sent from the cable head end (called the CMTS) to the individual data. Upstream data is sent in a smaller 2 Mhz channel. The 6 Mhz channel downstream allows potentially fast downloading times because using several bits per signal (they use Quadrature Amplitude Modulation to use multiple amplitudes and phases), cable can download at say 30 Mbps. The catch is that this bandwidth may be shared by lots of users and the cable companies impose rate caps on individual users.

Fiber: Fiber is analogous to a flashlight being turned on and off and detected by an eye. The flashlight is provided by a Light Emitting Diode (LED) or Laser that converts voltage into light; the eye is provided by a photodiode that outputs a voltage when light arrives. Bits are sent by turning the LED or laser on and off to emit on-off pulses of light. In fiber, the Nyquist and Shannon limits are very high. The real limits occur due to *chromatic* and *modal dispersion.*

Dispersion refers to the spreading out of a narrow pulse of light when it is passed through a fiber. It is caused by a light signal splitting up into different components that take different amounts of time to reach the destination. The first form of dispersion is modal dispersion in which a light signal can be thought of as splitting into two components: one component travels straight through the fiber and the other takes a longer path, reflecting off the walls (see Figure 13).

If the difference in arrival times between the straight and longer paths is x seconds, then the output pulse seen by the receiver will be at least x seconds wide. This happens even if the input pulse width is much smaller than x seconds. Hence, the term dispersion or spreading out. Dispersion places a limit on the bit rate: if the second bit is sent less than x seconds after the first bit, then the faster component of the second bit can interfere (cause Intersymbol Interference) with the slower component of the first bit, and can lead to bit errors.

Chromatic dispersion is similar except that the light breaks up into different frequencies (i.e., colors) each of which have a different speed through the fiber and hence have different arrival times.

Thus the transmission rate is limited to one bit every x seconds, where x is the difference in arrival times between the fastest and slowest components of a light pulse. We can improve the transmission rate by reducing x. To reduce or eliminate modal dispersion, the fiber width is reduced till there the only way (or mode) for light to travel is "straight down the middle". This is called *single mode* fiber as opposed to *multimode* fiber. Single mode fiber is often at most 1 micron in diameter and more expensive than multimode fiber. To eliminate chromatic dispersion, we use more accurate sources, like lasers, that emit light in a very small frequency range. Many high speed fiber links use lasers together with single mode fiber, despite the increased cost when compared to using LEDs and multimode fiber.

Satellites: Satellite transmission is roughly analogous to a person shouting in a valley and having others listen to the echoes. Transmitters are equipped with a transmitter (antenna) than sends high frequency waves (suitably modulated to convey the bit stream) to a satellite rotating around the earth. High frequency waves are very directional and propagate along the line of transmission. The satellite acts like a repeater in the sky and repeats the signal on its downlink. Since the satellite is fairly high above the earth, a small beam signal repeated by the satellite spreads out into a larger beam by the time it arrives at the earth. Any receivers in the "shadow" of the satellite beam can use a dish, appropriately tuned, to receive the signal.

The earliest satellites were *geosynchronous*: by orbiting at 36,000 km, they could rotate at the same speed as the earth.[10] Thus at all times, the antenna could point in the same direction without the need for an expensive steerable antenna. At this radius, the satellites can only be spaced 2 degrees apart, which limits the number of geosynchronous satellites to 180.

However, there has been a flurry of recent activity involving low flying (and hence not geosynchronous) satellites. Since such satellites are visible only for a small period of time, these systems use a necklace of such satellites that cover the sky. As one satellite moves out of range, another satellite comes in view and there is a handoff procedure between the two satellites as in cellular phones. The Motorola Irridium project is the canonical example with 66 satellites in the sky allowing anyone (even on the North Pole) to talk to anyone in the world. Unfortunately, cell phone calls using this technology cost too much (several dollars per minute) and so this reall cool idea went bankrupt.

A second recent development is the use of small, low cost dishes which allows small users to afford the use of satellite technology.

The high frequency bands used allow fairly large transmission speeds. However, the need to travel a large distance in the sky makes the latency (i.e., the time for the first bit of a signal to travel from sender to receiver) large, in the order of 250 msec. This can be a serious problem for interactive applications. TCP does poorly over satellite because it can increase its window size only after a round-trip delay because of its so-called slow start algorithm. However, satellites score over other media, including fiber by: being wireless (and hence avoiding right of way), being broadcast (which benefits broadcast applications such as publishing or video distribution, as well as mobile nodes that stay within the satellite shadow) and by being distance independent (unlike wired transmission whose cost increases with distance.)

Microwaves, Infrared, and RF: Microwave, infrared, and radio frequency refer to the frequency of the waves used for transmission. Microwave transmission is similar to satellite transmission except that the microwave signals (2-40 GHz) are sent between high towers without mediation through a satellite. Towers are used so that the signal path can avoid obstacles. Microwave was used heavily in the long distance phone network because of the ability to avoid right-of-way and labor costs, especially in crowded cities.

[10]Kepler's law states that the period of a satellite is proportional to $r^{1.5}$, where r is the orbital radius. Satellites flying close to the earth have a period of 90 minutes. We need to go up to 36,000 km to get a period of 24 hours.

Infrared, also high frequency, fell out of favor many years ago because, like microwave and satellite signals, it can be absorbed by obstacles. That can be an advantage in a single office, however, for a very local connection between a laptop and a server, which does not interfere with similar connections in nearby offices. Many recent laptops are coming equipped with an IR port for wireless docking. Older IR ports offered speeds of around 100 kbps, but a new standard called fast IR offers speeds of 4 Mbps.

Radio frequencies are lower frequencies than that of microwave, satellite and infrared. Lower frequency waves are omni-directional (spread out in all directions) and pass through obstacles. Recall that high frequency waves tend to be directional and absorbed by obstacles. The advantage of radio (or RF) transmission is the lack of a need for dishes (to receive the signal) and line-of-sight transmission. The last factor is especially important for wireless LANs in, say, a college campus. However, the lack of directionality and the ability to penetrate obstacles implies that radio signals at the same frequency can interfere. Thus early and current data transmission technologies that used radio (e.g., ALOHA and MACAW, see later) have to detect and resolve collisions when users transmit simultaneously.

5.4: Wireless Technologies

Wireless technologies are probably the most exciting development in physical layer technologies in the last 10 years. All of us have used a wireless hot spot. The key is that we do not need a CAT-5 Ethernet cable for our laptop to connect to the network. We use a wireless NIC Card to connect through the air to a router that can connect us to the network.

Wireless networks typically work in so-called infrastructure mode where communication occurs through a so-called wireless access point or AP. A hot-spot at say Starbucks typically has an access point. Communication is via radio frequencies. For example, the most common technology 802.11b transmits at up to 11 Mbps in the 2.4 to 2.485 Ghz frequency range. As we have seen, such radio waves are good because they move in all directions and pass through obstacles. However, they do interfere with microwaves and some cell phones. There are faster versions that are also being built such as 802.11g at up to 54 Mbps which (unlike standard physical layer technologies) are backwards compatible with 802.11 b.

When a network adminstrator configures an AP (this can be done at home or at Starbucks), it gives the AP a so-called SSID (or Service Set Identifier). When you do “view available networks” you see a list of SSIDs of all networks in your range. Each AP is also assigned a channel number. The 85 Mhz region from 2.4 to 2.485 is divided by frequency division multiplexing into 11 partially overlapping channels. 1, 6, and 11 are non-overlapping. One could install three APs and configure them at channels 1, 6, and 11 and get an overall bandwidth of up to 33 Mbps without interference.

In places like airports (or even homes, where you can often hear your neighbor’s AP), when a wireless station first moves in, it can hear the so-called beacon frames containing the SSID sent by each AP in the area. In passive mode, the wireless station, scans all 11 channels, listening for beacon frames. It may receive two or more beacons from different APs on the same channel. If each AP has sufficient signal strength, then there is some selection algorithm in the laptop’s firmware to pick one (or the user specifies), after which an “association” is made to the right AP by sending some handshake messages. At that point, IP level connectivity is made via DHCP discovery messages (we will study this later). Instead of passively waiting for APs, wireless stations can also actively send broadcast probe requests to APs in the area.

We will study the 802.11b access protocol after we have studied Ethernet. Like Ethernet, it fundamentally tries to send. If it detects a collision, it retransmits. However, the physical layer issues cause additional problems in 802.11b. In radio, it is possible to have three stations A, B, and C in a line with B (say the AP) in the center. Thus it possible that A can hear B, B can hear A and B, and C can only hear B. This is a headache because A could be transmitting happily to its AP. C who does not hear A now comes in brightly and starts sending to the AP B. However, C's frame now collides with that of A at B, because B can hear both A's frames and C's frames. This is called the hidden terminal problem and it does not occur in wired networks.

Wireless networks can also work in so-called ad-hoc mode where wireless endpoints communicate directly with each other without an intermediary like an AP. This is an advantage when there is no infrastructure such as a hot-spot. An example is Bluetooth. Two cell phones that are Bluetooth-enabled can communicate directly within short distances of each other. Although there is no AP, a group of Bluetooth enabled devices organize themselves into what is called a Piconet and elect a master. All communication is mediated by the master. Thus without infrastructure, one of the Bluetooth stations functions almost the same way as an AP in an 802.11b network.

While 802.11b and 802.11g offer considerable bandwidths, they require so-called hotspots where there are APs, and so can communicate over a small distance (10 to 100 m). An extension of this data-centric approach is so-called WiMax which allows larger distances via some different technologies but at smaller speeds of say 4 Mbps. An alternative approach is by the cell-phone carriers who already have considerable infrastructure and are masters of mobility as cars move from cell to cell. Their idea is to extend the cell network to also carry data so you can do Internet access in your car. For example, some users purchase a so-called EVDO card from anywhere to allow Internet access during a road-trip across the United States.

As the two proponents have different stakeholders (cell phone manufacturers and operators for 3G and 5G, and ISPs and cities who install 802.11 hotspots), both are being pushed hard. The 802.11 technologies have the edge in terms of cost and bandwidth, but the 3G/5G networks have the edge in terms of mobility and availability anywhere (even on a beach). As in all technology wars, it is likely they will co-exist.

6.0: Multiplexing

Because physical wires are expensive to buy, install and maintain, it pays to use the highest speed lines possible and to then share the use of this high speed line among multiple slower speed lines. *Multiplexing* is just a fancy term for sharing multiple streams of data between multiple senders (say S1, S2) and multiple receivers (say R1, R2). We want to carry signals from S1 to R1 concurrently with signals from S2 to R2 without interference.

Frequency Division Multiplexing (FDM) uses different parts of the line bandwidth for the two data streams so they do not interfere. FDM was widely used in the older long haul telephone network for multiplexing many voice channels before transmission on higher bandwidth coax or microwave links. *Time Division Multiplexing (TDM)* simply alternates among the different users such that a certain time slot is reserved for each user. An example is in T1 and SONET systems. FDM is natural for analog signals, while TDM is natural for digital signals.

In recent years, a form of FDM that has become popular in optics is to send different signals on the same fiber using different wavelength lights and to separate them out at the receiver using a prism. The major difference between *wavelength division multiplexing (WDM)* and standard FDM is that WDM can be done entirely using optics without any electronics. The current bottleneck in high speed data transfer is the electronics. Thus all-optical methods of switching like WDM offer the potential for very fast, possibly Terabit speed switching.

Most cell-phones use a different multiplexing technology called CDMA (Code Division Multiple Access). Each user is given a certain code which is a sequence of -1's and 1's say -1, 1, 1, -1. The voice is converted to digital and sent a bit at a time. Instead of sending a single bit 1, the bit is multiplied by the code and sent. A 1 is written as 1 as usual but a 0 is represented by -1. Thus a 1 would be sent as -1, 1, 1, -1. On the other hand, a 0 is sent as 1,-1, -1, 1. The key is that each user is given a different code as part of their SIM card. Even if they transmit at the same time, a property of the codes allow them not to interfere. The slides in class give you an example.

The standard form of CDMA requires each user to use a different code and to be synchronized so they start at the same time. This works fine for Base to Mobile communication, but not for two mobiles talking to the same base station because the two mobiles will not necessarily synchronize when talking at the same time. Thus what really is used in our Qualcomm phones is what is called asynchronous CDMA. Instead of using a fixed code each user is given a psuedo-random sequence such that if there are only a small number of users at the same time, the probability that there will be appreciable interference is small.

In other words, TDM, FDM, and even standard CDMA are fixed multiplexing systems while CDMA is a statistical multiplexing technique. Thus asynchronous CDMA can use the bandwidth more efficiently. As Wikipedia says, "In a bursty traffic environment like mobile telephony, the advantage afforded by Asynchronous CDMA is that the performance (bit error rate) is allowed to fluctuate randomly, with an average value determined by the number of users times the percentage of utilization. Suppose there are 2N users that only talk half of the time, then 2N users can be accommodated with the same average bit error probability as N users that talk all of the time."

Bluetooth uses another approach to having multiple users share the channel which is analogous to CDMA, as FDM is analogous to TDM. The idea is that users hop between frequencies (as opposed to using different codes in different time slots).

7.0: Interconnection

We have already seen that the telephone network can *appear* to be a single physical medium to data users when it actually consists of a number of different physical media that are *interconnected*. For example, in the T1 and SONET discussion, we had interconnections through T1 and SONET switches. Even ordinary twisted pair and coaxial cable are often extended through the use of repeaters to allow larger distance coverage and better signal-to-noise ratio of the received signals. A *repeater* is a device that clocks bits coming in on one physical line and clocks them out on an outgoing line; thus repeating bits will restore the signal levels of a bit whose level is "drooping".

Note therefore that the physical layer (and the Data Link layer) may *appear* to be working over a single hop (as we sketched in our introductory lectures). However, they actually may consist of a number of interconnected pieces of wire. However, the details of the interconnection must be *transparent* to the end users: the end users can therefore imagine that they are working over a single hop. Thus T1 switches and Ethernet repeaters are transparent to end users. We will see later that routing and transport interconnections are not transparent and the user is actually aware of the possible multiple hops in the path.

8.0 Review

After all this information, we stop to catch our breaths and ask what the physical layer was about. Recall that we divided the task of the physical layer (see Figure 2) into three parts: the media or transmission sublayer, the media dependent sublayer, and the coding and clock recovery sublayers.

In the Media Sublayer, we studied the use of Fourier Analysis to figure output signal behavior. We then studied the *Nyquist limit* which shows that signaling faster than the channel bandwidth causes Inter-Symbol Interference. Finally, we studied the *Shannon limit* which showed that signal levels can't be finer than noise, so we can only increase the bit rate to a certain point by increasing the number of levels.

We then studied the *Coding and Clock Recovery Sublayer*. We showed that clock recovery was the problem of calculating optimal receiver sampling times in the presence of clock drift and noise. We showed how to get into phase with a preamble and to correct for clock drift by using information provided by transitions in the bit stream. We saw many coding techniques to generate preambles and to force transitions, including asynchronous and synchronous codes.

We then studied the details of several kinds of media. Don't be confused by the details but recall the two important lessons: media affects protocol design, and each media has pros and cons (see Figure 12). We also studied the telephone system, with its collection of media, and argued that some aspects of the telephone system (partly because of historical reasons and partly because of future trends) greatly influence data communications.

The sublayer model can help you break up a complex implementation into manageable parts that you can understand. As in all layered models, it also shows you that parts can be interchanged if desired as long as the interfaces are maintained. For example, Figure 15 shows a sublayered model of a typical modem transmission that uses asynchronous coding, frequency shift keying for media transmission, and telephone wire. Figure 16 shows a sublayered model of Ethernet transmission using Manchester encoding, baseband on-off voltage signaling, and coaxial cable. The sublayer model makes you realize that one could, in principle, do Manchester encoding combined with Phase Shift Keying over a telephone line. This is no different conceptually from using say QUIC to replace the TCP Transport layer.

Finally, we studied the problems of multiplexing and interconnection at the physical layer. Multiplexing is a way of sharing several logical signals on a single physical line using techniques such as FDM and TDM. Interconnection is a way to combine multiple physical layer pipes to look like one logical bit pipe.

9.0 Conclusion

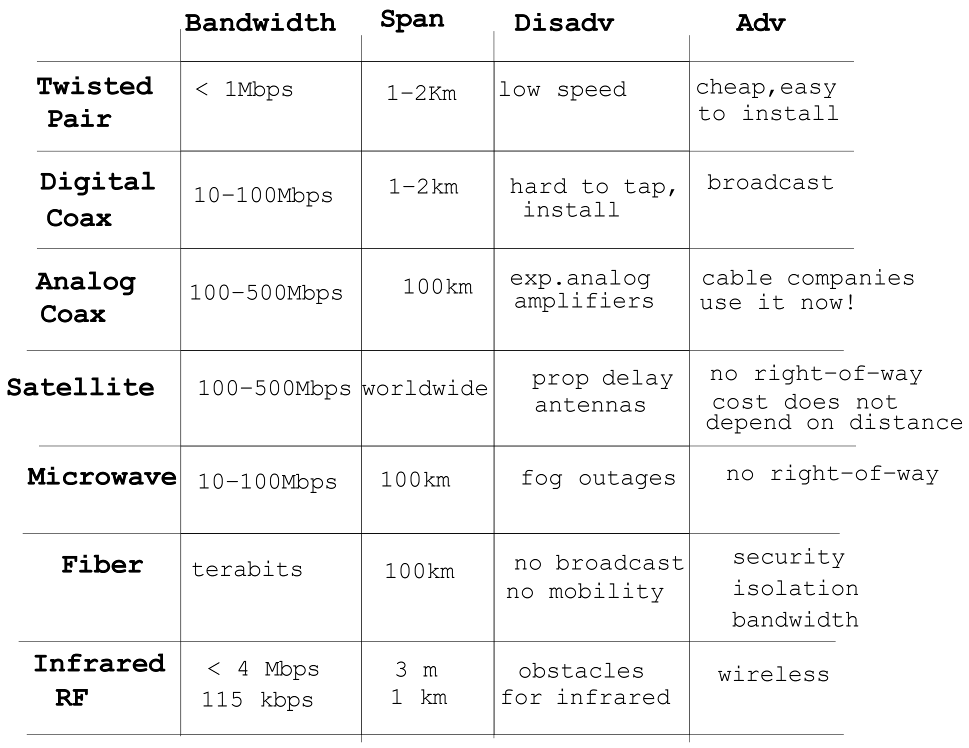
We have finished studying the details of the physical layer. As we study the Data Link and higher layers, we can simply think of the physical layer as providing a bit or symbol pipe between two nodes that can sometimes lose or corrupt bits/symbols. We next move on to studying the Data Link layer, which (we will see) provides a frame pipe between two nodes. We will see why this form of framing is necessary and why it makes possible new functions such as addressing and error detection.

Before we move on, we remark that all layers tend to solve similar abstract problems although they appear in different guises at each layer. Here is a list of some of these problems and a corresponding example in the physical layer:

- *Multiplexing and resource allocation*: Handling multiple users economically using a common set of resources (e.g., TDM, FDM at the physical layer)
- *Addressing*: Separating out traffic for multiple receivers and senders. (Null issue for the physical layer)
- *Error Control:* Techniques to deal with common errors. (At the physical layer we deal with noise and lack of bandwidth by spacing levels far apart, sending below Nyquist limit, sampling at mid bit, etc.)
- *Synchronization*: This is a generic term used to ensure that state variables in the receiver and sender are kept in some prescribed relation despite errors and uncertainty in timing. (The classic synchronization problem at the physical layer is to keep the intervals between the sender and receiver clock instants very close to each other despite clock drift and noise.)
- *Interconnection:* This is mostly unique to network systems but can be done at every layer. At the physical layer this done by repeaters that read bits in from one line and transmit them to other lines. It is also done by T1 switches that perform a form of routing function besides being a bit repeater. It is important that each layer interconnection device only uses the information available at that layer — i.e., bits at the physical layer, frames at the Data Link Layer, packets at the routing layer.

We will see that other layers have other forms of these problems to solve. In fact, most systems solve the same problems. For example, an Operating System must multiplex and share CPU time and memory among many users; it must have techniques for addressing resources like memory, devices and disk; it must ensure consistency in the face of errors like crashes; it must ensure synchronization in the face of interrupts and other concurrent activity. You will realize this more and more as you study more kinds of systems in databases and computer architecture.

In network systems another observation is that many layers solve the same problems repeatedly e.g., interconnection, multiplexing and addressing are solved at different layers in different ways. Part of this is because solutions at different layers have different tradeoffs (for example, solutions at low layers can be applicable more generally to most protocols but may be less efficient because they do not have enough information to specialize to the particular protocols running over them), but part of the problem is that each layer evolved and solved its problems independently.



	Bandwidth	Span	Disadv	Adv
Twisted Pair	< 1Mbps	1-2Km	low speed	cheap,easy to install
Digital Coax	10-100Mbps	1-2km	hard to tap, install	broadcast
Analog Coax	100-500Mbps	100km	exp.analog amplifiers	cable companies use it now!
Satellite	100-500Mbps	worldwide	prop delay antennas	no right-of-way cost does not depend on distance
Microwave	10-100Mbps	100km	fog outages	no right-of-way
Fiber	terabits	100km	no broadcast no mobility	security isolation bandwidth
Infrared RF	< 4 Mbps 115 kbps	3 m 1 km	obstacles for infrared	wireless

Figure 12: Pros and Cons of Various Media: a summary