Napster, MP3, and Perception

Robert Sekuler (copyright 2004)

It got a lot of bad press. The rock group Metallica denounced people who used it, proclaiming that they "exhibited the moral fiber of common looters." Metallica's drummer, Lars Ulrich, added that it is "sickening to know that our art is being traded like a commodity rather than the art that it is." The rapper Dr. Dre expressed similar outrage. And the Recording Industry Association of America sued to put the service out of business. But that very same object of outrage brought smiles to the faces of Limp Bizkit and other musical groups. And it certainly made many, many users **very** happy.

Yes, I'm talking about Napster, the late lamented sharing service. Love it or hate it, until not so long ago *www.napster.com* was one of the most frequently-visited sites on the internet. For example, during a typical week in its heyday (the fall of 2000), www.napster.com attracted about 3 million unique visitors (this statistic does not count return visits by any one individual, which happens quite a bit). A good deal of the internet traffic on many college campuses --on some music-obsessed campuses as high as 75%-- is consumed by downloading files from Napster or newer Napster replacements, such as GNUtella or Morpheus.

Downloaders may not realize it, but the success of these music services hinges upon the idiosyncracies of the human ear. If the human ear were much better than it is, Napster et al. would be impossible. This e-handout is a very brief introduction to how your ear makes digital music technology possible.

First, we need to get straight what exactly Napster is. Napster's internet site includes a search engine and master music list running on a bank of centralized "server" computers. These servers dish out information to other, client computers. "Napster" is also the name of software that runs on client computers, while allowing those clients to do double duty as servers.

When you use Napster, you log onto the website, and tell Napster's server what IP (Internet Protocol) address your machine is using. You can then upload to the server a list of music files you want to share with other Napster users. Now you can find other people using Napster, and they can find you. You can chat with them, see what music files they want to trade and, using the search engine, find specific music files, which you can download to your machine. All these music files are in the same digital format, MP3, which makes it possible to share music files on the internet.

MP3 is actually the format's nickname. Its more formal name is MPEG layer 3. ("MPEG" stands for "Motion Picture Experts Group", a collection of hundreds of people who design standards for the transmission and storage of digital information, including video information. Here's more on this group's work, including its future plans.

MP3 prescribes a set of rules for compressing digital audio information, such as the information contained on a music CD. These rules make it possible to send music over the internet, or to store that music on the harddisk of your computer. By the way, this same scheme is incorporated into many CD-ROM games, solid-state, audio memories,

digital audio broadcasting systems, and the like. And in all these applications, MP3's rules work by exploiting the way that the human ear and brain work. Which is why MP3 should interest students of perception, as well as other people who are consumers of internet music. (A good, complete discussion of MP3 appears in *MP3: The Definitive Guide*, by Scot Hacker; published in 2000 by O'Reilly Press.)

To understand how MP3 works, you must know that the contents of an ordinary music CD represent acoustic frequencies as high as 22 KHz, which is about the upper limits of human hearing underideal conditions. Digital encoding of an acoustic signal is done by a sampling process --that's a fundamental difference between old-fashioned, analog, recordings on LPs and analog tape, and their modern, digital counterparts on CDs. The continuously-varying analog signal is measured (sampled) at a fixed rate, say every 50 microseconds, and each one of the resulting measurements is represented in digital form, as a set of binary digits, 0's and 1's. (By the way, as you probably know, a microsecond is one millionth of a second.)

The amount of data produced by this process depends upon the sampling rate. If you measured an hour-long analog signal only once a minute, you would have a lot less data than if you measured that same hour-long signal once every 25 microseconds. In fact, the slower sampling rate would produce just 60 samples from that hour-long music, while the faster rate would produce 144,000,000 samples (40,000 samples per second x 60 seconds x 60 minutes). The computer file containing the results of the slower sampling rate would be far, far smaller than a file of results with the faster rate. So from the perspective of storage space, a slower sampling rate is more desirable. Because a smaller file can be sent over the internet (or any other communication channel) in a shorter time than a big file, the slower sampling produces a file that is easier to send via internet from one computer to another.

But, as the economist Milton Friedman advised in his 1975 book of that title, "There's no such thing as a free lunch." Greater compression of some material, such as a song, means that greater computational power will be required to decompress the file containing that compressed material. The efficiency gained by slower sampling rates carries a very heavy cost. If you sampled an hour's worth of music just once per minute --hoping to minimize storage space and to optimize transmission time-- you'd have a pretty lousy representation of that music. Because you sampled the acoustic energy (waveform) just once every minute, your representation would have failed to capture most of the variations in that waveform. In fact, any changes in intensity that happen faster than once per minute would be lost. OK, suppose that you did not want to lose any of the music in that acoustic signal. Just how fast would you have to sample so that you did not lose any information?

The answer was produced in the 1920's --long, long before MP3 or Napster--- by Harry Nyquist, a physicist who worked at American Telephone & Telegraph's Bell Laboratories in New Jersey. Nyquist was interested, as you might imagine, in finding efficient ways to send telephone and telegraph signals over AT&T's network of wires (in 1924, he was an inventor of a little thing called the fax machine). He determined that if one had a continuous signal, sound as the acoustic energy in speech, a set of samples could preserve the frequencies in that signal if the samples were taken at a high enough rate. In particular, if the highest frequency in some continuous signal --of speech or music, for example-- was f, samples taken at twice that frequency, or $2f$, would capture that frequency --and all lower frequencies, as well. This rule is called the **Nyquist sampling theorem**; the frequency of sampling, $2f$, that is sufficient to capture a signal's highest

frequency, *f*, is called the **Nyquist frequency**. Note that the Nyquist frequency varies with the frequency content of the continuous signal. The higher the signal's frequency spectrum, the faster must the sampling process be. Here's more, basic information on sampling.

Now we can apply Nyquist's theorem to the problem at hand: how often must we sample a music signal in order to produce a digital version that preserves the frequency information of the original? But it gets worse. Each sample requires many bits in order to represent the signal's amplitude at the time the sample was taken. For example, if you were truly miserly with your bits, restricting each sample to either 0 or 1 (a single bit), you'd have no way to represent subtle variations in signal intensity. You'd have the audio equivalent to a photograph that had just two light levels --black or white, with no gray shades, and certainly no colors. That would save storage space, but, again, at a considerable price. The bad news is that capture all the subtleties (intensity variations) of music, each sample can take up as much as 16 bits --a string of 16 0's and 1's. This allows you to capture 65,536 differences in intensity (65,536 is 2 to the 16th power).

On a music CD, the information is written on the CD at a rate of a tad over 22 KHz, for each of the two stereo channels. Suppose you wanted to take the information from that CD and transmit it over the internet. Nyquist's theorem tells us that to capture all that information, we must sample the CD at 44.1 kHz, stereo, with 16 bits per sample. Multiply that value by 2 (for two stereo channels) and then multiply by another factor of 2 because you have 2 bytes per value (that's what 16 bit means). As a result, a single minute's worth of sampled information from the song would take up.

44,100 samples/s * 2 channels * 2 bytes/sample * 60 s/min = 10MBytes

of storage space on your hard disk. To download that file over the internet, using an ordinary 56kB modem, would take you

10,000,000 bytes * 8 bits/byte / (56,000 bits/s * 60 s/min) = 24.5 minutes.

So to download just one minute's worth of music would take nearly a half hour. And heaven forbid that you wanted to download a song that lasted even longer, say three or four minutes. You could go out for a very good, very leisurely meal while the song downloaded.

How can we knock down the needed storage to manageable levels? Obviously, if we reduce the sampling frequency or shrink the number of bits per sample, we will have a loss in quality --some information will be lost. This is where MP-3 comes to the rescue. MP3 is one of a class of schemes called "perceptual codecs". The word **codec** is short for coding-decoding. Any perceptual codec builds on the fact that there's no point to storing information that cannot be perceived by humans. The **JPEG** format is a codec that is familiar to many computer users. This is a codec designed to compress and decompress photographic images. You may wonder why JPEG, which is a perfectly successful codec, couldn't be applied to audio data, and why another, newer codec, MP3, was needed. The answer lies in perceptual differences between vision and hearing.

A CD recording contains a tremendous amount of audio data that no human listener can hear. So if we have a sampling scheme that throws away the imperceptible data, then the sampled data will be indistinguishable from the original --even though the sampled

version contains lots fewer data. The trick, though, is to identify what information can be safely discarded. A compression-decompression scheme that discards information is called a **lossy** scheme (as opposed to a lossless scheme, one that saves all the information, but recodes it into more efficient format). With a lossy compression method, when you take some data set, compress it, and then decompress it, what you have is not identical to the original; with lossless compression, however, the product **is** identical to the original.

In his book on MP3, Scot Hacker puts the matter very well:

> "*Many lossy compression formats work by scanning for redundant data and reducing it to a mathematical depiction which can be 'unpacked' later on. Think for a moment of a photograph depicting a clear blue sky, and below it a beach. If you were to scan and store this image on your hard drive, you could end up storing hundreds of thousands of pixels of perfect blue, all identical to one another, and therefore redundant. The secret of a photographic compression method like GIF is that this redundant information is reduced to a single description. Rather than store all the bits individually, they may be represented as the mathematical equivalent of 'repeat blue pixel 273,000 times.' When the part of the image depicting the sand is encountered, the sand is analyzed for redundancy and similar reductions can be achieved. This is why simple images can be stored as small files, while complex images don't compress as well they contain less redundancy. On the other hand, JPEG compression works in accord with user-defined 'tolerance thresholds'; determining how similar two adjacent pixels (or, more accurately, frequencies) have to be before they're considered redundant with one another is the key to determining the degree of lossiness. If JPEG compression is set high, light blue and medium blue pixels may be treated as being redundant with one another. If JPEG compression is set low, the codec will be more fussy about determining which pixels are redundant. The end result will be a clearer picture and a larger image file*."

MP3 uses a coordinated set of perceptually-sensitive tricks to produce lossy versions of acoustic data. Working together, these tricks shrink the original data by somewhere between 12 and 14 times. The result is a version that can be stored and transmitted with considerable economy relative to the original, but with no perceptually distinguishable difference.

One of MP3's most complicated and interesting tricks exploits a phenomenon known as **auditory masking.** As you read in Chapter 11 of Sekuler and Blake's *Perception* textbook, masking means that if two tones of different frequencies are played simultaneously, the more intense of the two will mask its less intense competitor. (As you saw in the textbook, the amount of masking depends upon other factors too, such how close the tones are to one another in frequency). Basically, when two tones of similar frequency, the louder one masks the quieter one. MP3 exploits this quirk of the human ear. Several dozen times each second, MP3 splits the sound spectrum into 32 narrow, frequency bands. It then analyzes which of the bands would be masked by others. MP3 then digitally codes only those sounds that would be heard. This maneuver saves bits without affecting how the music sounds to listener. So masking can occur before, during or after a particular signal.

To accomplish this bit-saving trick, a computer running MP3 must analyze, moment by moment, the frequency content of an audio signal, such as the signal produced when some music CD is playing. And bear in mind that masking occurs not only between

frequencies that are present at exactly the same moment. A brief but intense signal can exert a masking effect for as much as one-tenth of a second after the signal has ceased. (The amount of masking diminishes with time.) MP3's identification of audible signals takes account of this time-delayed masking. This boosts the efficiency of MP3's version of the incoming signal.

In sum, MP3 doesn't waste precious bits on frequencies that wouldn't be heard anyway. To accomplish this selective pruning of masked signals, a computer running MP3's rules must incorporate a representation of the human ear's masking and audibility functions. Those functions allow the computer to know when particular sounds would and would not be heard. The complete process of pruning, which is quite a bit more complicated than the simple rules described here, requires an enormous amount of computing power for every minute of music that is processed.

Oh, I nearly left out the beauty part. Once the music has been sampled, encoded according to MP3's rules, and transmitted, the decoder part of MP3's codec comes into play. This decoder is a piece of software in a personal computer or in a dedicated MP3-player. The decoder translates the encoded digital signals back into audio signals that a listener can hear and enjoy, never having to stop and think about the **digital** and computational complexities that made all this possibility. Come to think of it, that's not much different from never having to stop and think about the **neural** complexities that make it possible to hear in general. *Thanks to Mark Hosang of Brandeis University for his valuable comments on this document.*