INGLES/ INGLES II

TRABAJO PRÁCTICO Nº 9

TUPAR, TUDAI E INGENIERÍA DE SISTEMAS

➢ Actividad de Pre lectura

1- **OBSERVE** las palabras técnicas que se adjuntan en el recuadro. **BUSQUE** en el diccionario aquellas que desconozca. **REDACTE** un párrafo consignando lo que sabe del tema `Multimedia` y **UTILICE** alguna de las palabras del recuadro en su elaboración.

Skin-decompress-concurrent- jitter- disjoint- metafile- interface- streaming media-buffer-bandwidth-combination-continuous media-push server-buffer

……

……

……

……

……

……

➢ GRAMÁTICA

2- (7.4. y 7.4.1**) IDENTIFIQUE** y **SUBRAYE** ejemplos de:

- Conectores que indiquen contraste y consecuencia
- Cognados
- Falsos cognados

- Oraciones condicionales
- Estructuras comparativas

➢ Actividades de Lectura

3- (7.4**) LEA** el texto e **IDENTIFIQUE** a qué hacen referencia las expresiones *the former* y *the latter*. ¿Qué tipo de referencia son? **CLASIFÍQUELAS** en anáfora o catáfora.

4- **CONTESTE** las siguientes preguntas relativas a la introducción a *Multimedia* (7.4.) y a la *Introducción a Audio digital* (7.4.1).

5- La siguiente cita fue extraída del texto de Tannenbaum: ***"Los recursos multimediales son el santo grial de la red"***. ¿Adhiere a esta postura? **EXPLIQUE**.

- **EXPLIQUE** por qué podría decirse que este libro es un ejemplo de `multimedia.`
- **EXPLIQUE** qué implica utilizar la palabra `multimedia.`
- **EXPLIQUE** por qué muchos comienzan a `salivar` ni bien escuchan la palabra multimedia. ¿Quiénes serían estos sujetos? ¿Por qué?
- Teniendo en cuenta las posturas propuestas, **CONSIGNE** a cuál adhiere el autor. **JUSTIFIQUE** su elección con una cita del texto o **MÁRQUELA**.
- **EXPLIQUE** de qué manera se compara el oído humano con un micrófono.
- **EXPLIQUE** por qué se establece una comparación entre el sistema auditivo y el visual.
- **EXPLIQUE** cómo funciona el convertidor análogo digital.

7- **LEA** el apartado 7. 4. 2. "***Audio Compression***". **REDACTE** un resumen utilizando el siguiente comienzo.

"Se pueden comprimir archivos de audio utilizando dos vías diferentes: la codificación de ondas o la codificación perceptual"

……

……

……

……

……

……

……

8- **LEA** el apartado 7.4.3 "Streaming Audio" e **IDENTIFÍQUE** los procedimientos diferentes para reproducir música. **CARACTERÍCELOS**.

……

……

……

……

……

……

……

……

……

9- **LEA** las afirmaciones propuestas y **ESCRIBA VERDADERO** o **FALSO**. **JUSTIFIQUE** cada respuesta con información del texto.

- ……. "Los sitios de música han resuelto problemas de velocidad de reproducción a través de los metarchivos ya que en caso de pérdida de paquetes, sólo se compromete la calidad de resolución temporariamente
- ……. "El entrelazado se puede utilizar tanto con audio comprimido o sin comprimir"**.**
-

10- **COMPARE** el procedimiento que utiliza la tecnología *push* y el que utiliza la tecnología *pull.* **REDACTE** un párrafo.

7.4 MULTIMEDIA

The wireless Web is an exciting new development, but it is not the only one. For many people, multimedia is the holy grail of networking. When the word is mentioned, both the propeller heads and the suits begin salivating as if on cue. The former see immense technical challenges in providing (interactive) video on demand to every home. The latter see equally immense profits in it. Since multimedia requires high bandwidth, getting it to work over fixed connections is hard enough. Even VHS-quality video over wireless is a few years away, so our treatment will focus on wired systems.

Literally, multimedia is just two or more media. If the publisher of this book wanted to join the current hype about multimedia, it could advertise the book as using multimedia technology. After all, it contains two media: text and graphics (the figures). Nevertheless, when most people refer to multimedia, they generally mean the combination of two or more **continuous media**, that is, media that have to be played during some well-defined time interval, usually with some user interaction. In practice, the two media are normally audio and video, that is, sound plus moving pictures.

However, many people often refer to pure audio, such as Internet telephony or Internet radio as multimedia as well, which it is clearly not. Actually, a better term is **streaming media**, but we will follow the herd and consider real-time audio to be multimedia as well. In the following sections we will examine how computers process audio and video, how they are compressed, and some network applications of these technologies. For a comprehensive (three volume) treatment on networked multimedia, see (Steinmetz and Nahrstedt, 2002; Steinmetz and Nahrstedt, 2003a; and Steinmetz and Nahrstedt, 2003b).

7.4.1 Introduction to Digital Audio

An audio (sound) wave is a one-dimensional acoustic (pressure) wave. When an acoustic wave enters the ear, the eardrum vibrates, causing the tiny bones of the inner ear to vibrate along with it, sending nerve pulses to the brain. These pulses are perceived as sound by the listener. In a similar way, when an acoustic wave strikes a microphone, the microphone generates an electrical signal, representing the sound amplitude as a function of time. The representation, processing, storage, and transmission of such audio signals are a major part of the study of multimedia systems.

The frequency range of the human ear runs from 20 Hz to 20,000 Hz. Some animals, notably dogs, can hear higher frequencies. The ear hears logarithmically, so the ratio of two sounds with power A and B is conventionally expressed in **dB** (**decibels**) according to the formula

$$\text{dB} = 10 \log_{10}(A/B)$$

If we define the lower limit of audibility (a pressure of about 0.0003 dyne/cm^2) for a 1-kHz sine wave as 0 dB, an ordinary conversation is about 50 dB and the pain threshold is about 120 dB, a dynamic range of a factor of 1 million.

The ear is surprisingly sensitive to sound variations lasting only a few milliseconds. The eye, in contrast, does not notice changes in light level that last only a few milliseconds. The result of this observation is that jitter of only a few milliseconds during a multimedia transmission affects the perceived sound quality more than it affects the perceived image quality.

Audio waves can be converted to digital form by an **ADC** (**Analog Digital Converter**). An ADC takes an electrical voltage as input and generates a binary number as output. In Fig. 7-1(a) we see an example of a sine wave. To represent this signal digitally, we can sample it every ΔT seconds, as shown by the bar heights in Fig. 7-1(b). If a sound wave is not a pure sine wave but a linear superposition of sine waves where the highest frequency component present is f, then the Nyquist theorem (see Chap. 2) states that it is sufficient to make samples at a frequency $2f$. Sampling more often is of no value since the higher frequencies that such sampling could detect are not present.

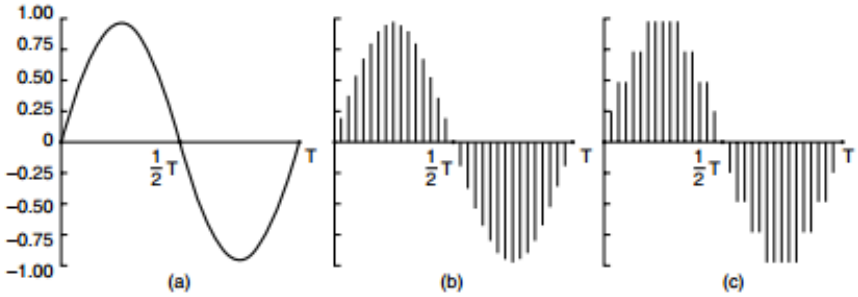

Figure 7-1. (a) A sine wave. (b) Sampling the sine wave. (c) Quantizing the samples to 4 bits.

Digital samples are never exact. The samples of Fig. 7-1(c) allow only nine values, from −1.00 to +1.00 in steps of 0.25. An 8-bit sample would allow 256 distinct values. A 16-bit sample would allow 65,536 distinct values. The error introduced by the finite number of bits per sample is called the **quantization noise**. If it is too large, the ear detects it.

Two well-known examples where sampled sound is used are the telephone and audio compact discs. Pulse code modulation, as used within the telephone system, uses 8-bit samples made 8000 times per second. In North America and Japan, 7 bits are for data and 1 is for control; in Europe all 8 bits are for data. This system gives a data rate of 56,000 bps or 64,000 bps. With only 8000 samples/sec, frequencies above 4 kHz are lost.

Audio CDs are digital with a sampling rate of 44,100 samples/sec, enough to capture frequencies up to 22,050 Hz, which is good enough for people, but bad for canine music lovers. The samples are 16 bits each and are linear over the range of amplitudes. Note that 16-bit samples allow only 65,536 distinct values, even though the dynamic range of the ear is about 1 million when measured in steps of the smallest audible sound. Thus, using only 16 bits per sample introduces some quantization noise (although the full dynamic range is not covered—CDs are not supposed to hurt). With 44,100 samples/sec of 16 bits each, an audio CD needs a bandwidth of 705.6 kbps for monaural and 1.411 Mbps for stereo. While this is lower than what video needs (see below), it still takes almost a full T1 channel to transmit uncompressed CD quality stereo sound in real time.

Digitized sound can be easily processed by computers in software. Dozens of programs exist for personal computers to allow users to record, display, edit, mix, and store sound waves from multiple sources. Virtually all professional sound recording and editing are digital nowadays.

Music, of course, is just a special case of general audio, but an important one. Another important special case is speech. Human speech tends to be in the 600-Hz to 6000-Hz range. Speech is made up of vowels and consonants, which have different properties. Vowels are produced when the vocal tract is unobstructed, producing resonances whose fundamental frequency depends on the size and shape of the vocal system and the position of the speaker's tongue and jaw. These sounds are almost periodic for intervals of about 30 msec. Consonants are produced when the vocal tract is partially blocked. These sounds are less regular than vowels.

Some speech generation and transmission systems make use of models of the vocal system to reduce speech to a few parameters (e.g., the sizes and shapes of various cavities), rather than just sampling the speech waveform. How these vocoders work is beyond the scope of this book, however.

7.4.2 Audio Compression

CD-quality audio requires a transmission bandwidth of 1.411 Mbps, as we just saw. Clearly, substantial compression is needed to make transmission over the Internet practical. For this reason, various audio compression algorithms have been developed. Probably the most popular one is MPEG audio, which has three layers (variants), of which **MP3** (**MPEG audio layer 3**) is the most powerful and best known. Large amounts of music in MP3 format are available on the Internet, not all of it legal, which has resulted in numerous lawsuits from the artists and copyright owners. MP3 belongs to the audio portion of the MPEG video compression standard. We will discuss video compression later in this chapter; let us look at audio compression now.

Audio compression can be done in one of two ways. In **waveform coding** the signal is transformed mathematically by a Fourier transform into its frequency components. Figure 2-1(a) shows an example function of time and its Fourier amplitudes. The amplitude of each component is then encoded in a minimal way. The goal is to reproduce the waveform accurately at the other end in as few bits as possible.

The other way, **perceptual coding**, exploits certain flaws in the human auditory system to encode a signal in such a way that it sounds the same to a human listener, even if it looks quite different on an oscilloscope. Perceptual coding is based on the science of **psychoacoustics**—how people perceive sound. MP3 is based on perceptual coding.

The key property of perceptual coding is that some sounds can **mask** other sounds. Imagine you are broadcasting a live flute concert on a warm summer day. Then all of a sudden, a crew of workmen nearby turn on their jackhammers and start tearing up the street. No one can hear the flute any more. Its sounds have been masked by the jackhammers. For transmission purposes, it is now sufficient to encode just the frequency band used by the jackhammers because the listeners cannot hear the flute anyway. This is called **frequency masking**—the ability of a loud sound in one frequency band to hide a softer sound in another frequency band that would have been audible in the absence of the loud sound. In fact, even after the jackhammers stop, the flute will be inaudible for a short period of time because the ear turns down its gain when they start and it takes a finite time to turn it up again. This effect is called **temporal masking**.

To make these effects more quantitative, imagine experiment 1. A person in a quiet room puts on headphones connected to a computer's sound card. The computer generates a pure sine wave at 100 Hz at low, but gradually increasing power. The person is instructed to strike a key when she hears the tone. The computer records the current power level and then repeats the experiment at 200 Hz, 300 Hz, and all the other frequencies up to the limit of human hearing. When averaged over many people, a log-log graph of how much power it takes for a tone to be audible looks like that of Fig. 7-2(a). A direct consequence of this curve is that it is never necessary to encode any frequencies whose power falls below the threshold of audibility. For example, if the power at 100 Hz were 20 dB in Fig. 7-2(a), it could be omitted from the output with no perceptible loss of quality because 20 dB at 100 Hz falls below the level of audibility.

Now consider Experiment 2. The computer runs experiment 1 again, but this time with a constant-amplitude sine wave at, say, 150 Hz, superimposed on the test frequency. What we discover is that the threshold of audibility for frequencies near 150 Hz is raised, as shown in Fig. 7-2(b).

The consequence of this new observation is that by keeping track of which signals are being masked by more powerful signals in nearby frequency bands, we can omit more and more frequencies in the encoded signal, saving bits. In Fig. 7-2, the 125-Hz signal can be completely omitted from the output and no one will be able to hear the difference. Even after a powerful signal stops in some frequency band, knowledge of its temporal masking properties allow us to continue to omit

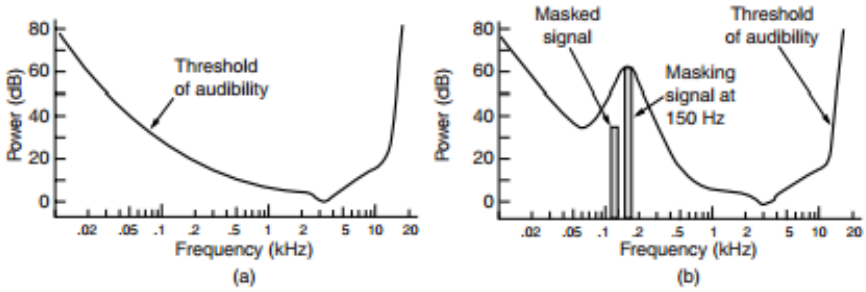

Figure 7-2. (a) The threshold of audibility as a function of frequency. (b) The masking effect.

the masked frequencies for some time interval as the ear recovers. The essence of MP3 is to Fourier-transform the sound to get the power at each frequency and then transmit only the unmasked frequencies, encoding these in as few bits as possible.

With this information as background, we can now see how the encoding is done. The audio compression is done by sampling the waveform at 32 kHz, 44.1 kHz, or 48 kHz. Sampling can be done on one or two channels, in any of four configurations:

1. Monophonic (a single input stream).
2. Dual monophonic (e.g., an English and a Japanese soundtrack).
3. Disjoint stereo (each channel compressed separately).
4. Joint stereo (interchannel redundancy fully exploited).

First, the output bit rate is chosen. MP3 can compress a stereo rock 'n roll CD down to 96 kbps with little perceptible loss in quality, even for rock 'n roll fans with no hearing loss. For a piano concert, at least 128 kbps are needed. These differ because the signal-to-noise ratio for rock 'n roll is much higher than for a piano concert (in an engineering sense, anyway). It is also possible to choose lower output rates and accept some loss in quality.

Then the samples are processed in groups of 1152 (about 26 msec worth). Each group is first passed through 32 digital filters to get 32 frequency bands. At the same time, the input is fed into a psychoacoustic model in order to determine the masked frequencies. Next, each of the 32 frequency bands is further transformed to provide a finer spectral resolution.

In the next phase the available bit budget is divided among the bands, with more bits allocated to the bands with the most unmasked spectral power, fewer bits allocated to unmasked bands with less spectral power, and no bits allocated to masked bands. Finally, the bits are encoded using Huffman encoding, which assigns short codes to numbers that appear frequently and long codes to those that occur infrequently.

There is actually more to the story. Various techniques are also used for noise reduction, antialiasing, and exploiting the interchannel redundancy, if possible, but these are beyond the scope of this book. A more formal mathematical introduction to the process is given in (Pan, 1995).

7.4.3 Streaming Audio

Let us now move from the technology of digital audio to three of its network applications. Our first one is streaming audio, that is, listening to sound over the Internet. This is also called music on demand. In the next two, we will look at Internet radio and voice over IP, respectively.

The Internet is full of music Web sites, many of which list song titles that users can click on to play the songs. Some of these are free sites (e.g., new bands looking for publicity); others require payment in return for music, although these often offer some free samples as well (e.g., the first 15 seconds of a song). The most straightforward way to make the music play is illustrated in Fig. 7-3.

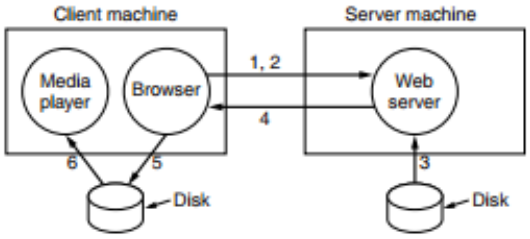

Figure 7-3. A straightforward way to implement clickable music on a Web page.

The process starts when the user clicks on a song. Then the browser goes into action. Step 1 is for it to establish a TCP connection to the Web server to which the song is hyperlinked. Step 2 is to send over a *GET* request in HTTP to request the song. Next (steps 3 and 4), the server fetches the song (which is just a file in MP3 or some other format) from the disk and sends it back to the browser. If the file is larger than the server's memory, it may fetch and send the music a block at a time.

Using the MIME type, for example, *audio/mp3*, (or the file extension), the browser looks up how it is supposed to display the file. Normally, there will be a helper application such as RealOne Player, Windows Media Player, or Winamp, associated with this type of file. Since the usual way for the browser to communicate with a helper is to write the content to a scratch file, it will save the entire music file as a scratch file on the disk (step 5) first. Then it will start the media player and pass it the name of the scratch file. In step 6, the media player starts fetching and playing the music, block by block.

In principle, this approach is completely correct and will play the music. The only trouble is that the entire song must be transmitted over the network before the music starts. If the song is 4 MB (a typical size for an MP3 song) and the modem is 56 kbps, the user will be greeted by almost 10 minutes of silence while the song is being downloaded. Not all music lovers like this idea. Especially since the next song will also start with 10 minutes of download time, and the one after that as well.

To get around this problem without changing how the browser works, music sites have come up with the following scheme. The file linked to the song title is not the actual music file. Instead, it is what is called a **metafile**, a very short file just naming the music. A typical metafile might be only one line of ASCII text and look like this:

```
rtsp://joes-audio-server/song-0025.mp3
```

In most cases, the server named in the metafile is not the same as the Web server. In fact, it is generally not even an HTTP server, but a specialized media server. In this example, the media server uses **RTSP (Real Time Streaming Protocol)**, as indicated by the scheme name *rtsp*. It is described in RFC 2326.

The media player has four major jobs to do:

1. Manage the user interface.
2. Handle transmission errors.
3. Decompress the music.
4. Eliminate jitter.

Most media players nowadays have a glitzy user interface, sometimes simulating a stereo unit, with buttons, knobs, sliders, and visual displays. Often there are interchangeable front panels, called **skins**, that the user can drop onto the player. The media player has to manage all this and interact with the user.

Its second job is dealing with errors. Real-time music transmission rarely uses TCP because an error and retransmission might introduce an unacceptably long gap in the music. Instead, the actual transmission is usually done with a protocol like RTP, which we studied in Chap. 6. Like most real-time protocols, RTP is layered on top of UDP, so packets may be lost. It is up to the player to deal with this.

In some cases, the music is interleaved to make error handling easier to do. For example, a packet might contain 220 stereo samples, each containing a pair of 16-bit numbers, normally good for 5 msec of music. But the protocol might send all the odd samples for a 10-msec interval in one packet and all the even samples in the next one. A lost packet then does not represent a 5 msec gap in the music, but loss of every other sample for 10 msec. This loss can be handled easily by having the media player interpolate using the previous and succeeding samples. estimate the missing value.

The use of interleaving to achieve error recovery is illustrated in Fig. 7-4. Here each packet holds the alternate time samples for an interval of 10 msec. Consequently, losing packet 3, as shown, does not create a gap in the music, but only lowers the temporal resolution for some interval. The missing values can be interpolated to provide continuous music. This particular scheme only works with uncompressed sampling, but shows how clever coding can convert a lost packet into lower quality rather than a time gap. However, RFC 3119 gives a scheme that works with compressed audio.

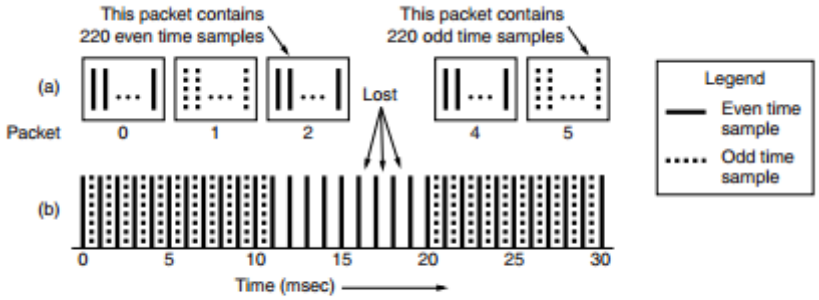

Figure 7-4. When packets carry alternate samples, the loss of a packet reduces the temporal resolution rather than creating a gap in time.

Two approaches can be used to keep the buffer filled. With a **pull server**, as long as there is room in the buffer for another block, the media player just keeps

Two approaches can be used to keep the buffer filled. With a **pull server**, as long as there is room in the buffer for another block, the media player just keeps

sending requests for an additional block to the server. Its goal is to keep the buffer as full as possible.

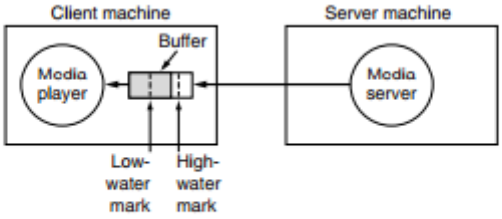

Figure 7-5. The media player buffers input from the media server and plays from the buffer rather than directly from the network.

The disadvantage of a pull server is all the unnecessary data requests. The server knows it has sent the whole file, so why have the player keep asking? For this reason, it is rarely used.

With a **push server**, the media player sends a *PLAY* request and the server just keeps pushing data at it. There are two possibilities here: the media server runs at normal playback speed or it runs faster. In both cases, some data is buffered before playback begins. If the server runs at normal playback speed, data arriving from it are appended to the end of the buffer and the player removes data from the front of the buffer for playing. As long as everything works perfectly, the amount of data in the buffer remains constant in time. This scheme is simple because no control messages are required in either direction.

The other push scheme is to have the server pump out data faster than it is needed. The advantage here is that if the server cannot be guaranteed to run at a regular rate, it has the opportunity to catch up if it ever gets behind. A problem here, however, is potential buffer overruns if the server can pump out data faster than it is consumed (and it has to be able to do this to avoid gaps).

To operate a push server, the media player needs a remote control for it. This is what RTSP provides. It is defined in RFC 2326 and provides the mechanism for the player to control the server. It does not provide for the data stream, which is usually RTP. The main commands provided for by RTSP are listed in Fig. 7-6.

Command	Server action
DESCRIBE	List media parameters
SETUP	Establish a logical channel between the player and the server
PLAY	Start sending data to the client
RECORD	Start accepting data from the client
PAUSE	Temporarily stop sending data
TEARDOWN	Release the logical channel

Figure 7-6. RTSP commands from the player to the server.

Fuente:

Tannenbaum, A. (: fourth edition) *Computer Networks.* London: Pearson Education, Inc. pps. 701- 710. Disponible en: http://authors.phptr.com/tanenbaumcn4/ [Último acceso: 18 de agosto de 2015]