Multimedia Communications

06EC841

PART - A

UNIT – 1:

MULTIMEDIA COMMUNICATIONS

Introduction, multimedia information representation, multimedia networks, multimedia applications, media types, communication modes, network types, multipoint conferencing, network QoS application QoS. 7 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & Proceedings:

1. M. Tatipamula and B. Khasnabish (Eds.), Multimedia Communication Networks Technologies and Services, Artech House, Boston, 1998.
2. ISO8348 OSI Data Communication – Network Service Definition, 1997.
3. K. R. Rao and Z. S. Bojkovic, Packet Video Communications Over ATM Networks,
Prentice Hall PTR, Upper Saddle River, NJ, 2000.
4. ITU MEDIACOM2004, Project Description – Version 3.0, March 2002.
5. ISO7498/1-4 OSI, Information Processing Systems – Basic Reference Model of OSI,
1998.
6. N. Modiri, The ISO reference model entities, IEEE Network Magazine, 5, 24–33 (1991).

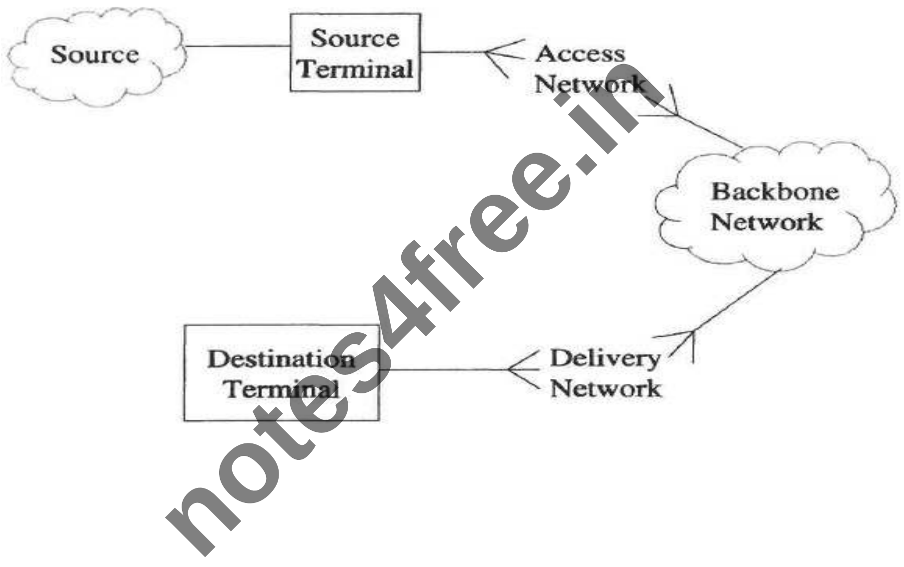
7. ISO8072, Information Processing Systems – Open Systems Interconnection – Oriented Transport Service Definition, 1987.
8. A. R. Modarressi and S. Mohan, Control and management in next-generation networks: challenges and opportunities, IEEE Comm. Magazine, 38, 94–102 (2000).

Multimedia communications have emerged as a major research and development area. In particular, computers in multimedia open a wide range of possibilities by combining different types of digital media such as text, graphics, audio, and video. The emergence of the World Wide Web (WWW), two decades ago, has fuelled the growth of multimedia computing.

Multimedia – an interactive presentation of speech, audio, video, graphics, and text, has become a major theme in today's information technology that merges the practices of communications, computing, and information processing into an interdisciplinary field. In recent years, there has been a tremendous amount of activity in the area of multimedia communications: applications, middleware, and networking. A variety of techniques from various disciplines such as image and video processing, computer vision, audio and speech processing, statistical pattern recognition, learning theory, and data-based research have been employed.

In this chapter, we are interested in multimedia communications; that is, we are interested in the transmission of multimedia information over networks. By *multimedia,* we mean data, voice, graphics, still images, audio, and video, and we require that the networks support the transmission of multiple media, often at the same time.

Fig 1.1: components of multimedia communication network



In Figure 1.1 the Source consists of any one or more of the multimedia sources, and the job of the Source Terminal is to compress the Source such that the bit rate delivered to the network connection between the Source Terminal and the Destination Terminal is at least approximately appropriate. Other factors may be considered by the Source Terminal as well. For example, the Source Terminal may be a battery-power-limited device or may be aware that the Destination Terminal is limited in signal processing power or display capability.

Further, the Source Terminal may packetize the data in a special way to guard against packet loss and aid error concealment at the Destination Terminal. All such factors impinge on the design of the Source Terminal. The Access Network may be reasonably modeled by a single line connection, such as a 28.8 Kbit/s modem, a 56 Kbit/s modem, a 1.5 Mbit/s Asymmetric Digital Subscriber Line (ADSL) line, and so on, or it may actually be a network that has shared capacity, and hence have packet loss and delay characteristics in addition to certain rate constraints. The Backbone Network may consist of a physical circuit switched connection, a dedicated virtual path through a packet-switched network, or a standard best-effort Transmission Control Protocol/Internet Protocol (TCP/IP) connection, among other possibilities. Thus, this network has characteristics such as bandwidth, latency, jitter, and packet loss, and may or may not have the possibility of Quality of Service (QoS) guarantees. The Delivery Network may have the same general set of characteristics as the Access Network, or one may envision that in a one-to-many transmission that the Delivery Network might be a corporate intranet.

Finally, the Destination Terminal may have varying power, mobility, display or audio capabilities.

- "Multimedia" indicate that the information/data being transferred over the network may be composed of one or more of the following media types:

- Text
- Images
- Audio
- video

- <u>Media types</u>
  - Text: unformatted text, formatted text
  - Images: computer-generated, pictures
  - Audio: speech, music, general audio
  - Video: video clips, movies, films
- Network types
- Multimedia + Network → multimedia communications

<u>Multimedia Information Representation</u>

- Text, images
  - Blocks of digital data
  - Does not vary with time (time-independent)
  - Audio, video
  - Vary with time (time-dependent)
  - Analog signal
  - Must be converted into digital form for integration
- Communication networks cannot support the high bit rates of audio, video → Compression is applied to digitized signals.

Multimedia Networks:

Many applications, such as video mail, video conferencing, and collaborative work systems, require networked multimedia. In these applications, the multimedia objects are stored at a server and played back at the client's sites. Such applications might require broadcasting multimedia data to various remote locations or accessing large depositories of multimedia sources. Multimedia networks require a very high transfer rate or bandwidth, even when the data is compressed. Traditional networks are used to provide error-free transmission. However, most multimedia applications can tolerate errors in transmission due to corruption or packet loss without retransmission or correction. In some cases, to meet real-time delivery requirements or to achieve synchronization, some packets are even discarded. As a result, we can apply lightweight transmission protocols to multimedia networks. These protocols cannot accept retransmission, since that might introduce unacceptable delays.

Multimedia networks must provide the low latency required for interactive operation. Since multimedia data must be synchronized when it arrives at the destination site, networks should provide synchronized transmission with low jitter. In multimedia networks, most communications are multipoint as opposed to traditional point-to-point communication. For example, conferences involving more than two participants need to distribute information in different media to each participant.

Conference networks use multicasting and bridging distribution methods. Multicasting replicates a single input signal and delivers it to multiple destinations. Bridging combines multiple input signals into one or more output signals, which then deliver to the participants.

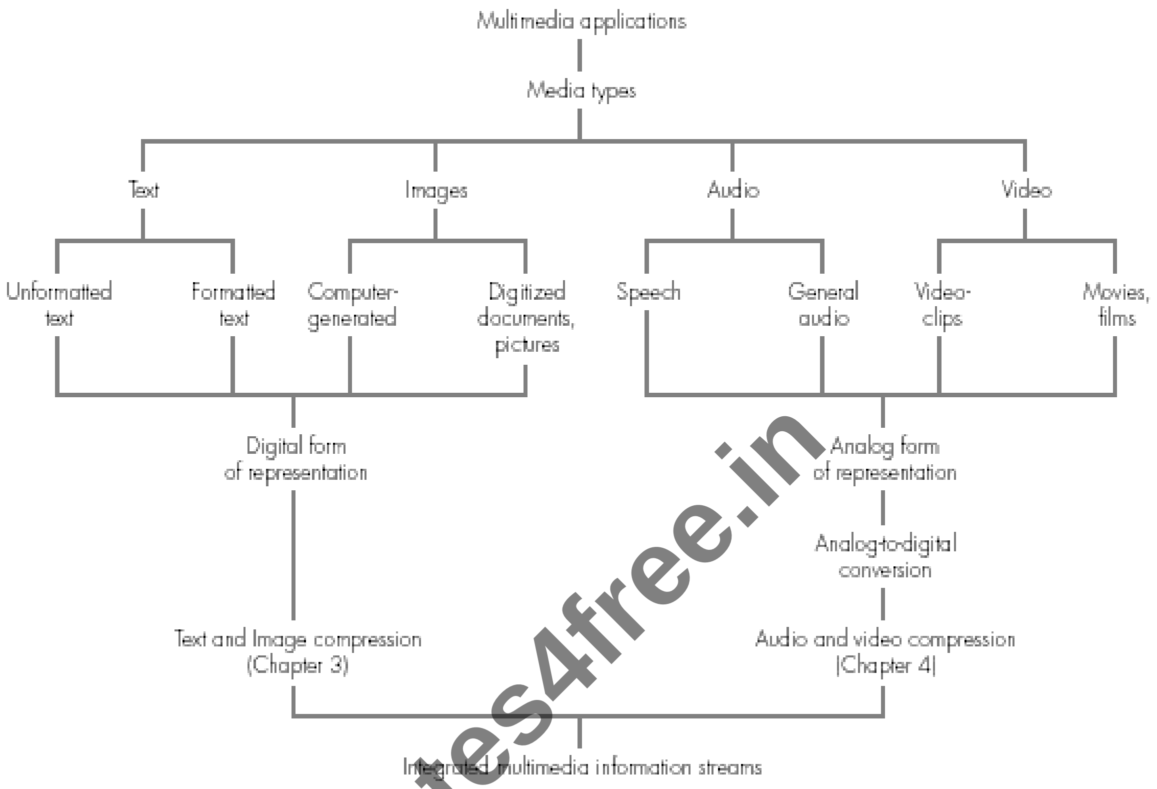
Traditional networks do not suit multimedia Ethernet, which provides only 10 Mbps, its access time is not bounded, and its latency and jitter are unpredictable. Token-ring networks provide 16 Mbps and are deterministic. From this point of view, they can handle multimedia. However, the predictable worst case access latency can be very high.

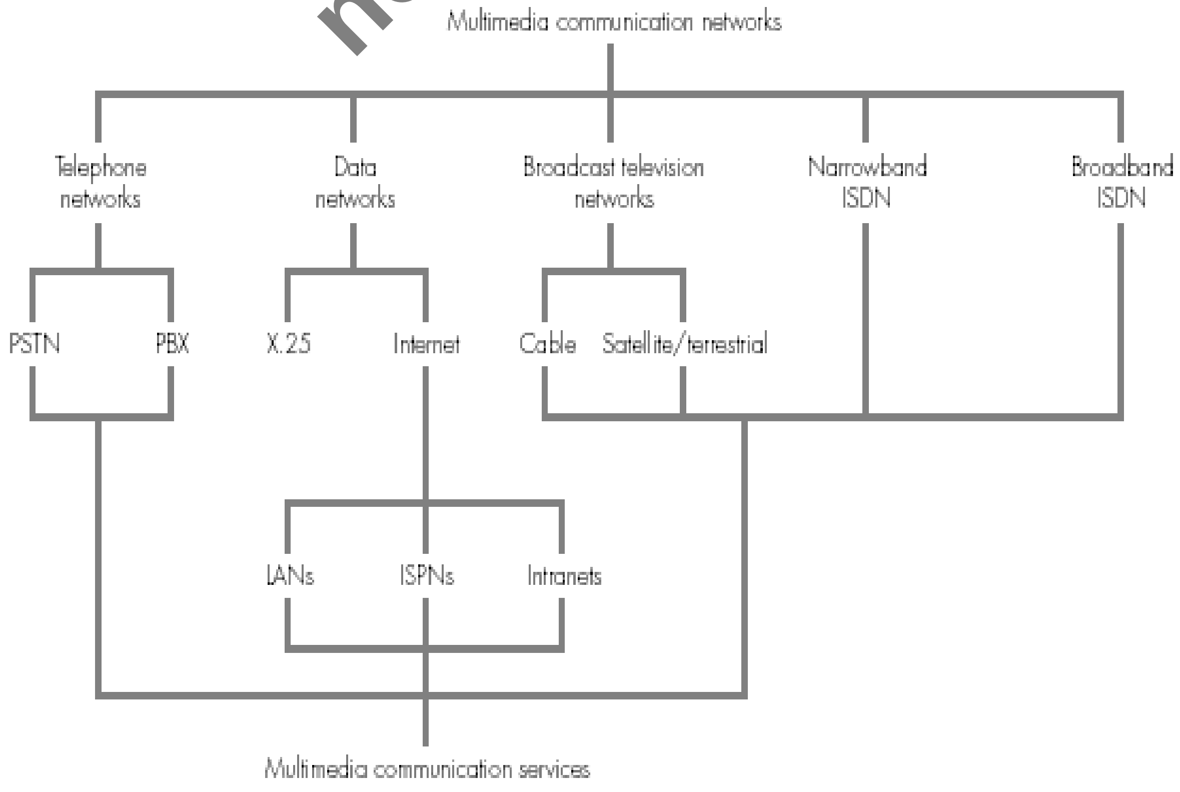
A fiber distributed data interface (FDDI) network provides 100 Mb/s bandwidth, sufficient for multimedia. In the synchronized mode, FDDI has a low access latency and low jitter. It also guarantees a bounded access delay and a predictable average bandwidth for synchronous traffic. However, due to the high cost, FDDI networks are used primarily for backbone networks, rather than networks of workstations.

- **Telephone networks**
- **Data networks**
- **Broadcast television networks**
- **Integrated services digital networks**

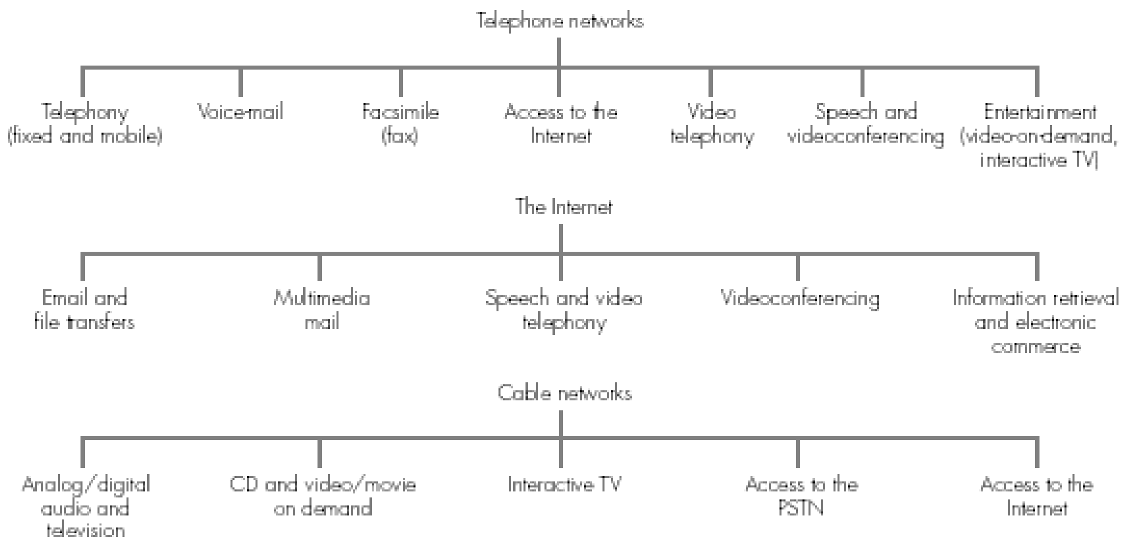
- **Broadcast multiservice networks**

Media Types

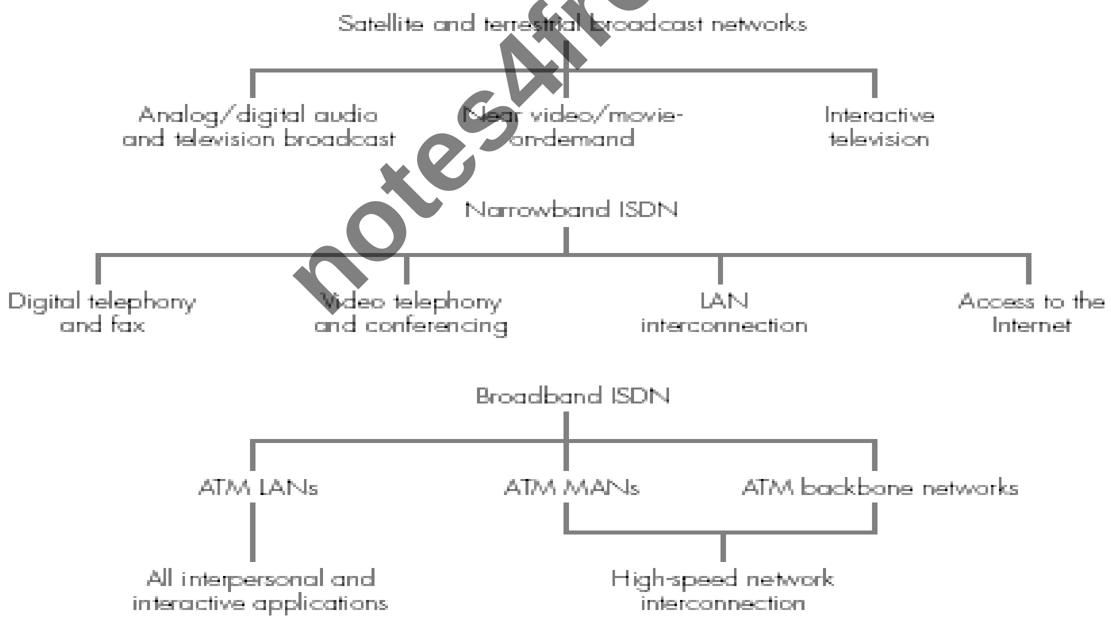

Multimedia applications
Media types
Text
Images
Audio
Video
Unformatted text
Formatted text
Computer-generated
Digitized documents, pictures
Speech
General audio
Video-clips
Movies, films
Digital form of representation
Analog form of representation
Analog-to-digital conversion
Text and Image compression (Chapter 3)
Audio and video compression (Chapter 4)
Integrated multimedia information streams



PSTN = public switched telephone network
PBX = private branch exchange
ISDN = integrated services digital network
LANs = local area networks
ISPNs = internet service provider networks


Telephone networks
Telephony (fixed and mobile)
Voice-mail
Facsimile (fax)
Access to the Internet
Video telephony
Speech and videoconferencing
Entertainment (video-on-demand, interactive TV)
The Internet
Email and file transfers
Multimedia mail
Speech and video telephony
Videoconferencing
Information retrieval and electronic commerce
Cable networks
Analog/digital audio and television
CD and video/movie on demand
Interactive TV
Access to the PSTN
Access to the Internet

Multimedia applications


Satellite and terrestrial broadcast networks
Analog/digital audio and television broadcast
Near video/movie-on-demand
Interactive television
Narrowband ISDN
Digital telephony and fax
Video telephony and conferencing
LAN interconnection
Access to the Internet
Broadband ISDN
ATM LANs
ATM MANs
ATM backbone networks
All interpersonal and interactive applications
High-speed network interconnection

UNIT - 2

MULTIMEDIA INFORMATION REPRESENTATION

Introduction, digital principles, text, images, audio, video.

5 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & Proceedings:

1. . M. Mampaey, TINA for services and advanced signaling and control in next-generation networks, IEEE Comm. Magazine, 38, 104–110 (2000).
2. G. Karlson, Asynchronous Transfer of Video, SICS Research report R95:14, Sweden,
   1997.
3. M. R. Pickering and J. F. Arnold, A perceptually efficient VBR rate control algorithm,
   IEEE Trans. Image Processing, 3, 527–531 (1994).
4. A. Ortega et al., Rate constraints for video transmission over ATM networks based on joint source/network criteria, Annales des Telecommunications, 50, 603–616 (1995).
5. Y. Shoham and A. Gersho, Efficient bit allocation for an arbitrary set of quantizers, IEEE Trans ASSP, 36, 1445–1453 (1988).
6. K. Ramchandran, A. Ortega, and M. Vetterli, Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders, IEEE Trans Image Processing,

2.1 Introduction

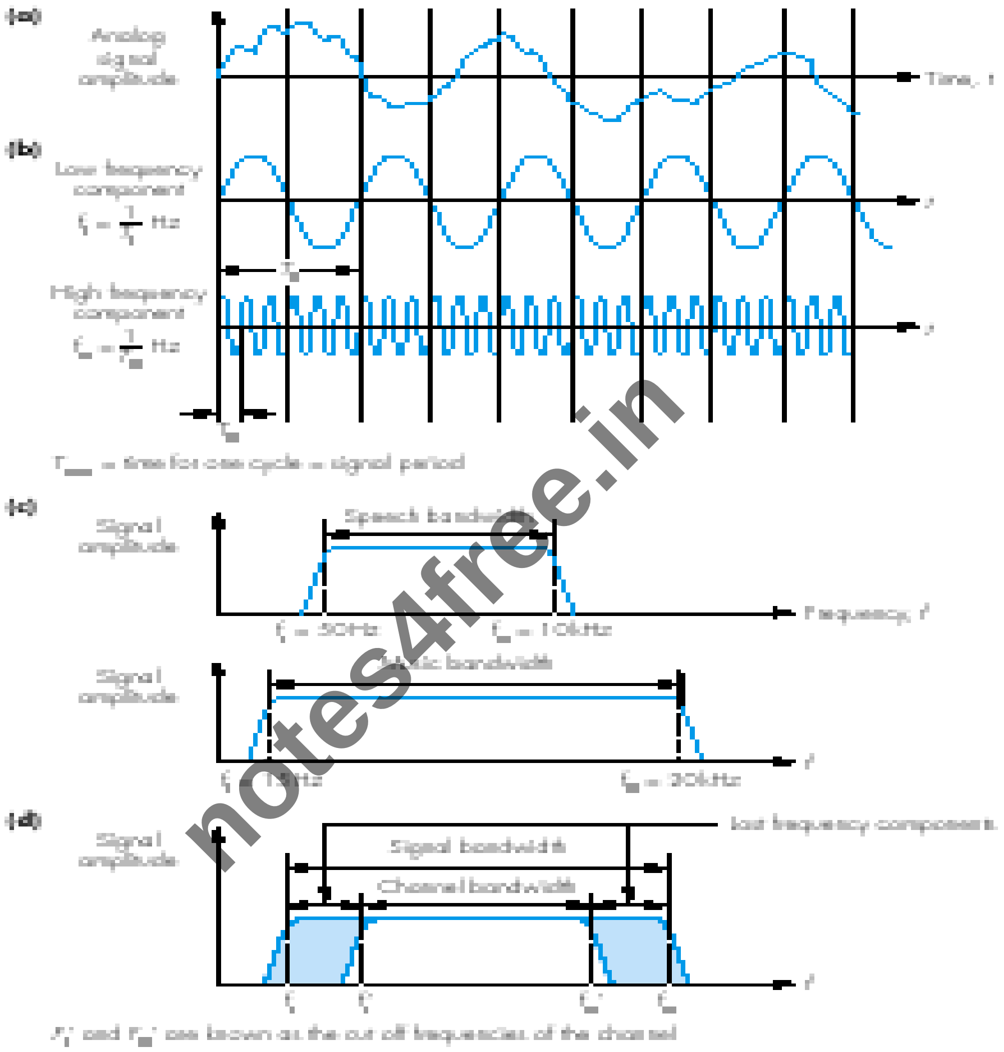
- The conversion of an analog signal into a digital form
- Signal encoder, sampling, signal decoder

2.2 Digitization principles

2.2.1 Analog signals

- Fourier analysis can be used to show that any time-varying analog signal is made up of a possibly infinite number of single-frequency sinusoidal signals whose amplitude and phase vary continuously with time relative to each other
- Signal bandwidth
- Fig2.1
- The bandwidth of the transmission channel should be equal to or greater than the bandwidth of the signal─bandlimiting channel

Figure 2.1 Signal properties: (a) time-varying analog signal; (b) sinusoidial frequency components; (c) signal bandwidth examples; (d) effect of a limited bandwidth transmission channel.

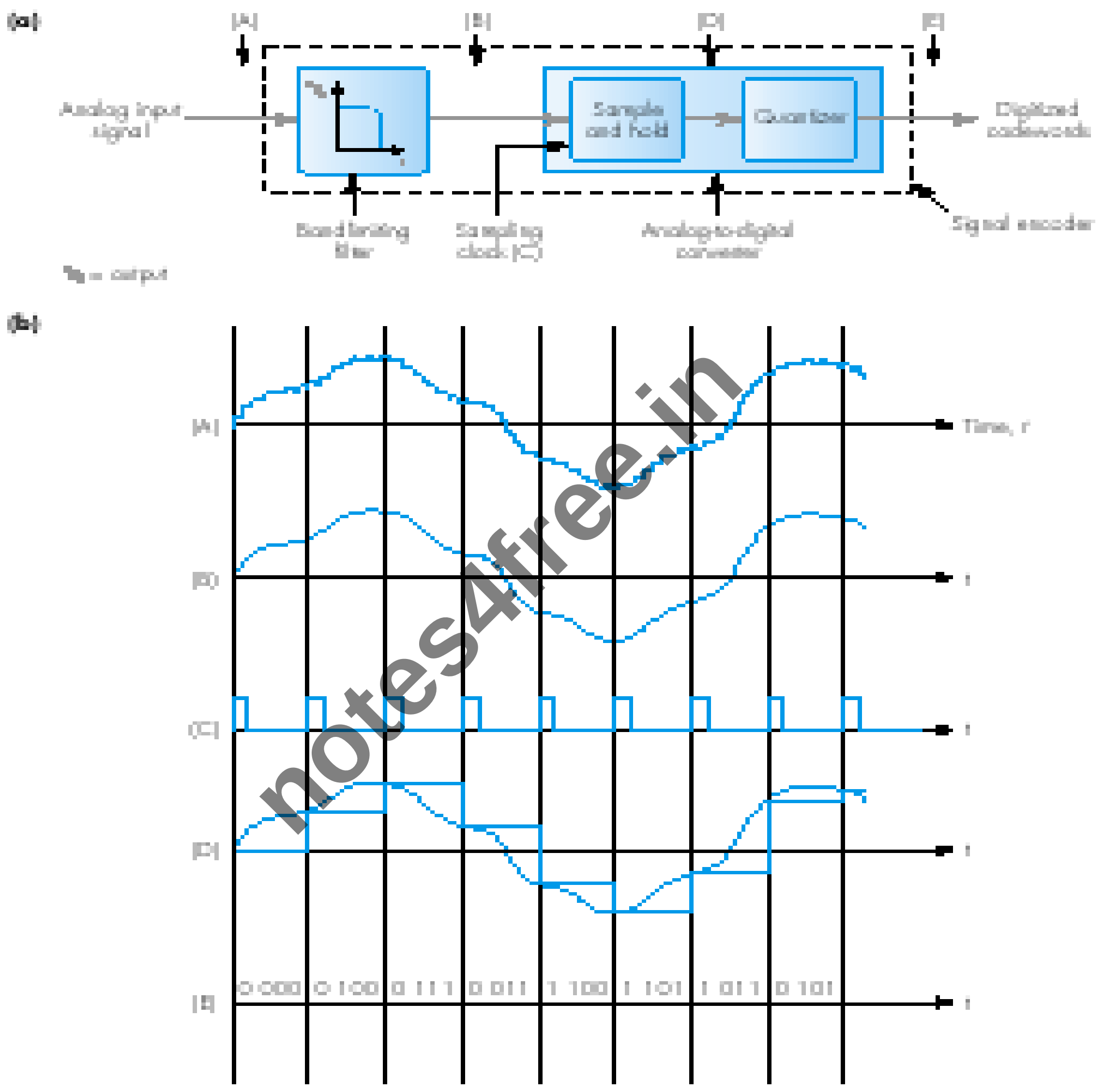


2.2.2 Encoder design

- A bandlimiting filter and an analog-to-digital converter(ADC), the latter comprising a sample-and-hold and a quantizer
- Fig2.2

- Remove selected higher-frequency components from the source signal (A)
- (B) is then fed to the sample-and-hold circuit
- Sample the amplitude of the filtered signal at regular time intervals (C) and hold the sample amplitude constant between samples (D)

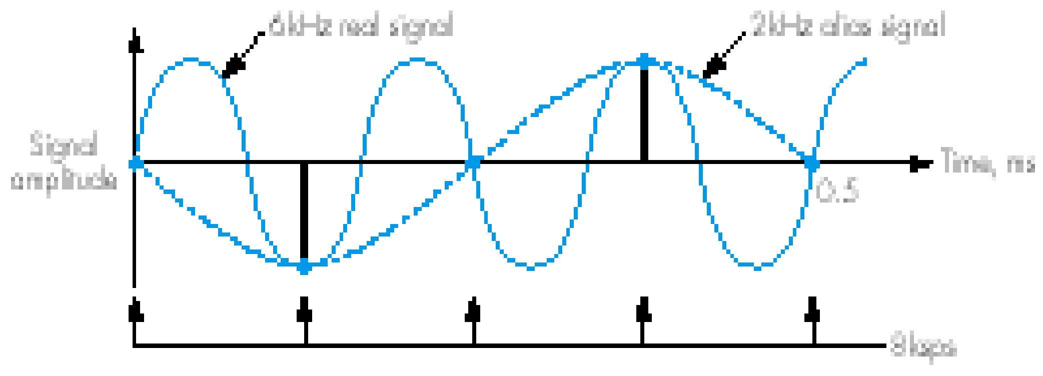
Figure 2.2 Signal encoder design: (a) circuit components; (b) associated waveform set.



- Quantizer circuit which converts each sample amplitude into a binary value known as a codeword (E)
- The signal to be sampled at a rate which is higher than the maximum rate of change of the signal amplitude

- The number of different quantization levels used to be as large as possible
- Nyquist sampling theorem states that: in order to obtain an accurate representation of a time-varing analog signal, its amplitude must be sampled at a minimum rate that is equal to or greater than twice the highest sinusoidal frequency component that is present in the signal
- Nyquist rate: samples per second (sps)
- The distortion caused by sampling a signal at a rate lower than the Nyquist rate
- Fig2.3
- Alias signals: they replace the corresponding original signals

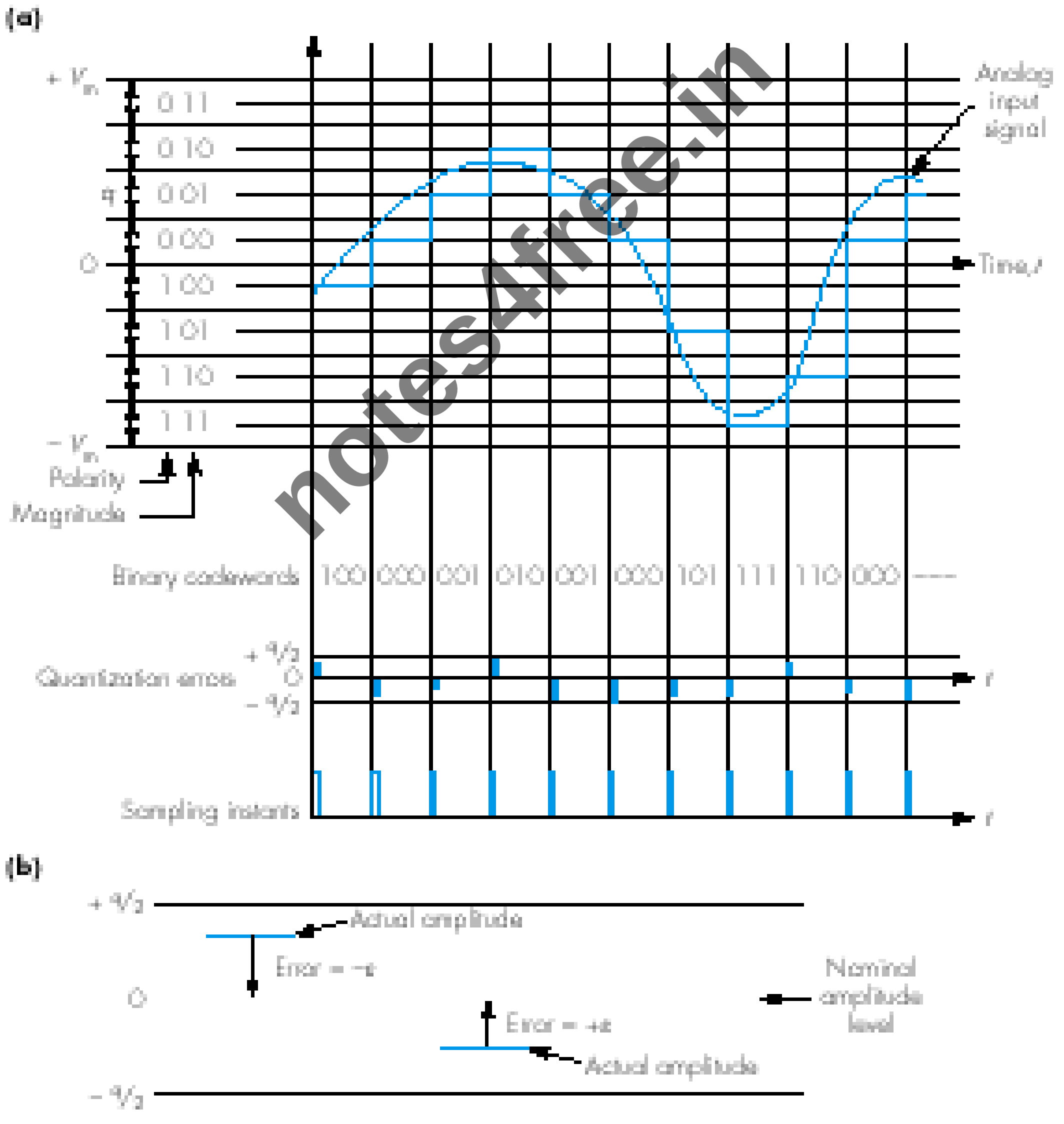
Figure 2.3 Alias signal generation due to undersampling.



- Quantization intervals
- A finite number of digits is used, each sample can only be represented by a corresponding number of discrete levels

- Fig2.4
- If Vmax is the maximum positive and negative signal amplitude and n is the number of binary bits used, then the magnitude of each quantization interval, q

Figure 2.4 Quantization procedure: (a) source of errors; (b) noise polarity.



- Each codeword corresponds to a nominal amplitude level which is at the center of the corresponding quantization interval
- The difference between the actual signal amplitude and the corresponding nominal amplitude is called the quantization error (Quantization noise)
- The ratio of the peak amplitude of a signal to its minimum amplitude is known as the dynamic range of the signal, *D* (decibels or dB)

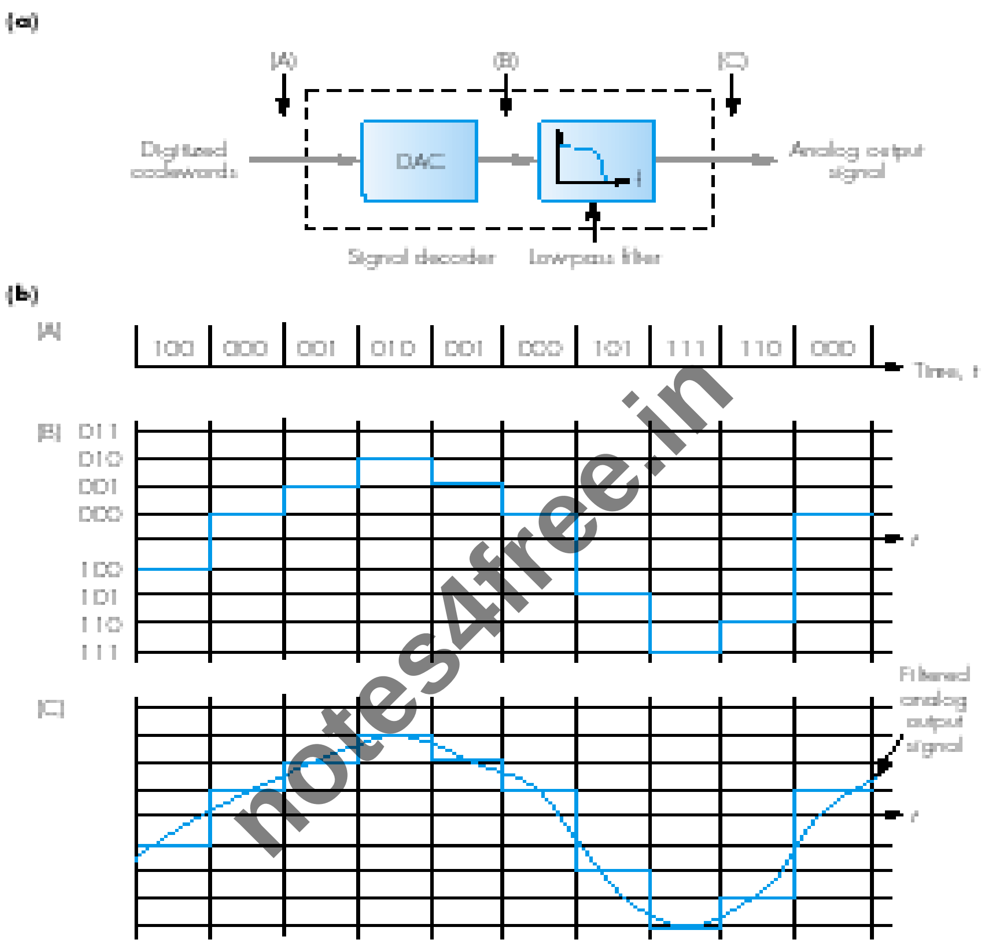
$$D = 20\log_{10}\left(V_{\max}/V_{\min}\right) dB$$

- It is necessary to ensure that the level of quantization noise relative to the smallest signal amplitude is acceptable
- Example 2.2

2.2.3 Decoder design

- Fig2.5
- Reproduce the original signal, the output of the DAC is passed through a low-pass filter which only passes those frequency components that made up the original filtered signal (C)
- Audio/video encoder-decoder or audio/video codec

Figure 2.5 Signal decoder design: (a) circuit components; (b) associated waveform set.



2.3 Text

- Three types of text
    - Unformatted text
    - Formatted text
    - hypertext

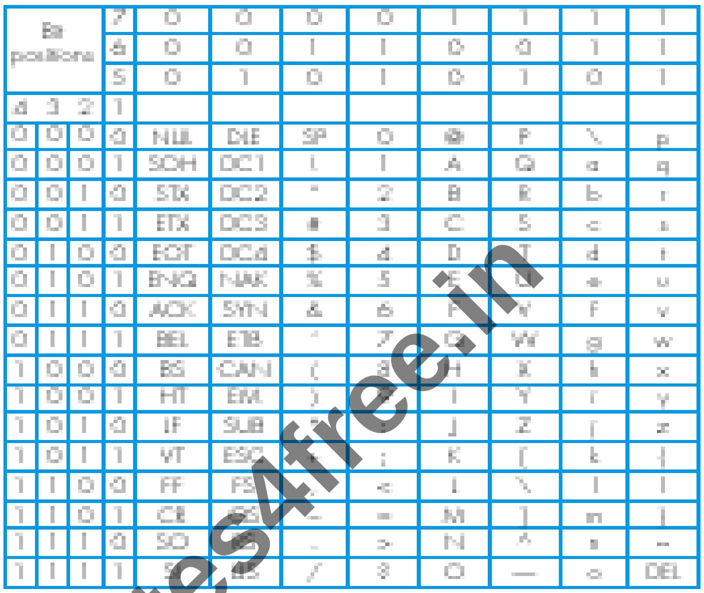
2.3.1 Unformatted text

- American Standard Code for Information Interchange (ASCII character set)

- Fig2.6
- Mosaic characters create relatively simple graphical images

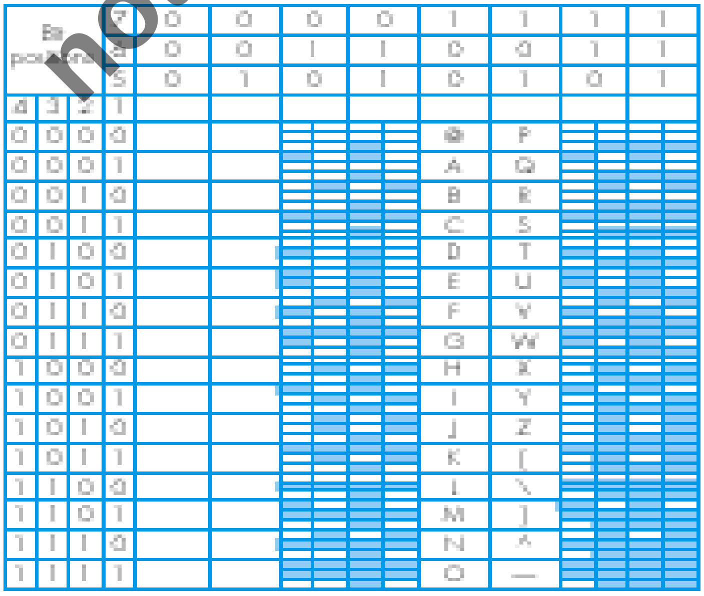
Figure 2.6 Two example character sets to produce unformatted text: (a) the basic ASCII character set; (b) supplementary set of mosaic characters.

(a)



Bit positions			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0	NUL	DLE	SP	0	@	P	\`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	—	o	DEL

(b)

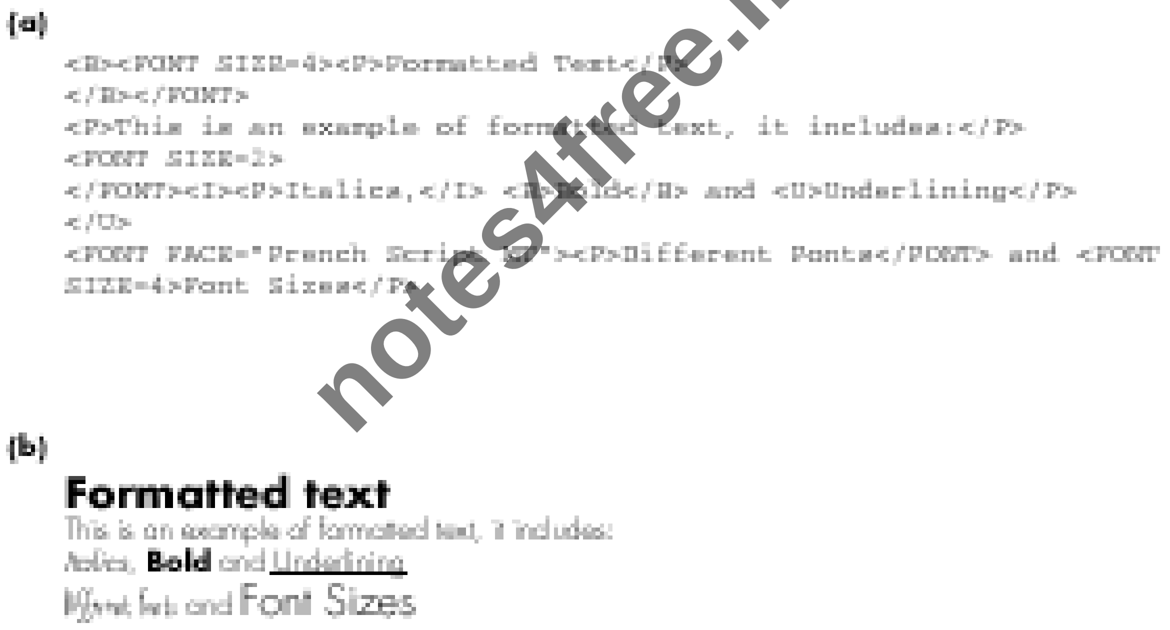


Bit positions			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0			▦	▦	@	P	▦	▦
0	0	0	1			▦	▦	A	Q	▦	▦
0	0	1	0			▦	▦	B	R	▦	▦
0	0	1	1			▦	▦	C	S	▦	▦
0	1	0	0			▦	▦	D	T	▦	▦
0	1	0	1			▦	▦	E	U	▦	▦
0	1	1	0			▦	▦	F	V	▦	▦
0	1	1	1			▦	▦	G	W	▦	▦
1	0	0	0			▦	▦	H	X	▦	▦
1	0	0	1			▦	▦	I	Y	▦	▦
1	0	1	0			▦	▦	J	Z	▦	▦
1	0	1	1			▦	▦	K	[	▦	▦
1	1	0	0			▦	▦	L	\	▦	▦
1	1	0	1			▦	▦	M	]	▦	▦
1	1	1	0			▦	▦	N	^	▦	▦
1	1	1	1			▦	▦	O	—	▦	▦

2.3.2 Formatted text

- Produced by most word processing packages
- Each with different headings and with tables, graphics, and pictures inserted at appropriate points
- Fig2.8
- WYSIWYG: an acronym for what-you-see-is-what-you-get

Figure 2.8 Formatted text: (a) an example formatted text string;(b) printed version of the string.



(a)

```
<B><FONT SIZE=4><P>Formatted Text</P>
</B></FONT>
<P>This is an example of formatted text, it includes:</P>
<FONT SIZE=2>
</FONT><I><P>Italics,</I> <B>Bold</B> and <U>Underlining</P>
</U>
<FONT FACE="French Script MT"><P>Different Fonts</FONT> and <FONT
SIZE=4>Font Sizes</P>
```

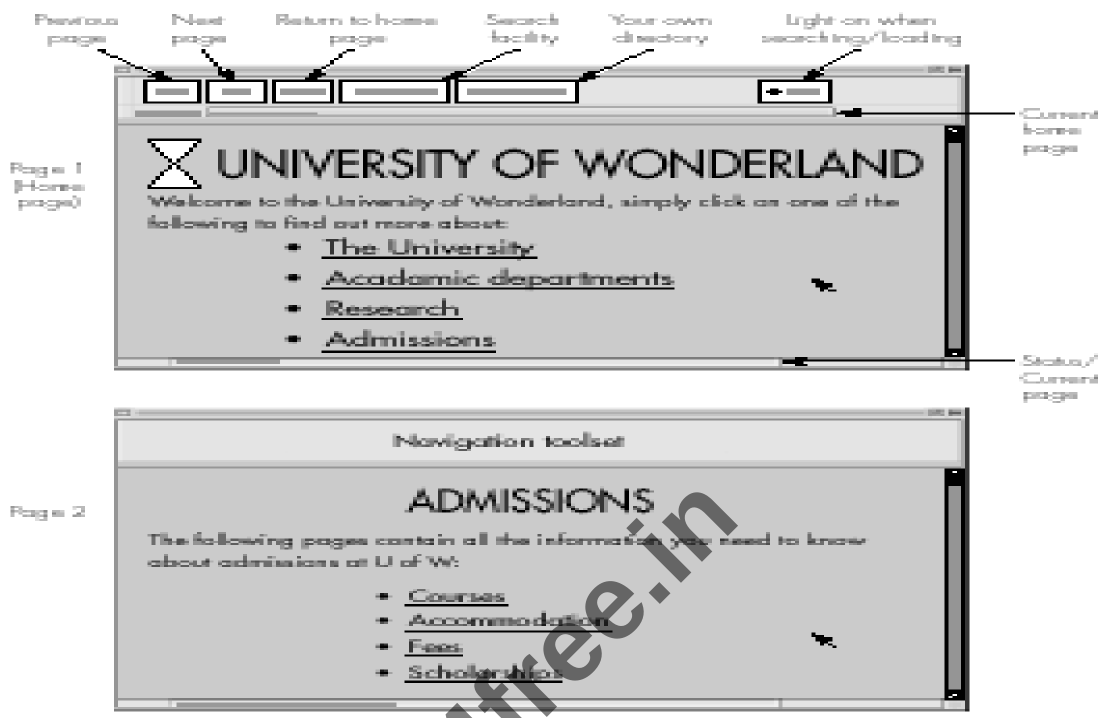
(b)

Formatted text
This is an example of formatted text, it includes:
Italics, **Bold** and Underlining
Different fonts and Font Sizes

2.3.3 Hypertext

- Formatted text that enables a related set of documents–normally referred to as pages–to be created which have defined linkage points–referred to as hyperlinks –between each other
- Fig2.9

Figure 2.9 Example of an electronic document written in hypertext.



UNIVERSITY OF WONDERLAND

Welcome to the University of Wonderland, simply click on one of the following to find out more about:

- The University
- Academic departments
- Research
- Admissions

ADMISSIONS

The following pages contain all the information you need to know about admissions at U of W:

- Courses
- Accommodation
- Fees
- Scholarships

Note:
- Page 2 is displayed after clicking the cursor on •Admissions of Page 1
- Selected images can be used as a background.
- Hyperlinks can be either underlined (as shown) or in a different color

2.4 Images

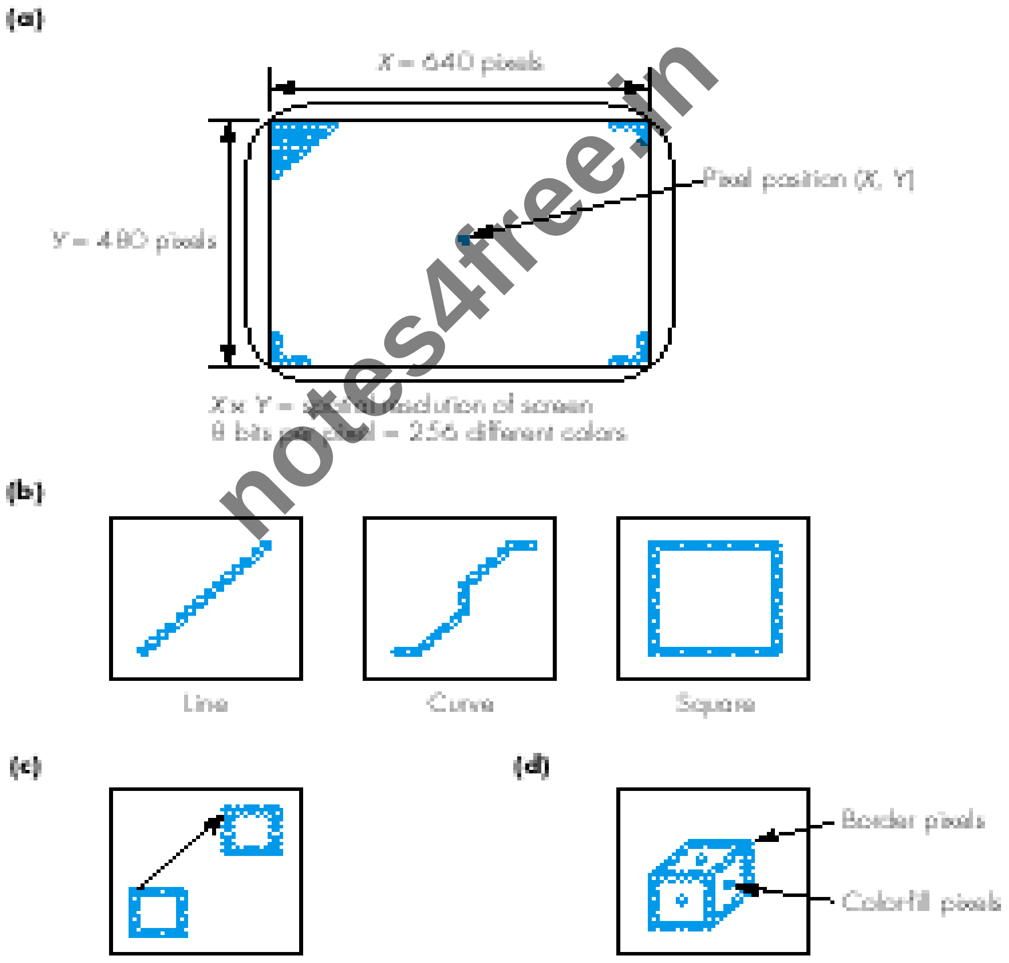
- Image are displayed in the form of a two-dimensional matrix of individual picture elements—known as pixels or pels

2.4.1 Graphics

- Fig 2.10
- Two forms of representation of a computer graphic: a high-level version (similar to the source code of a high-level program) and the actual pixel-image of the graphic (similar to the byte-string corresponding to the low-level machine code—bit-map format)

- Standardized forms of representation such as GIF (graphical interchange format) and TIFF (tagged image file format)

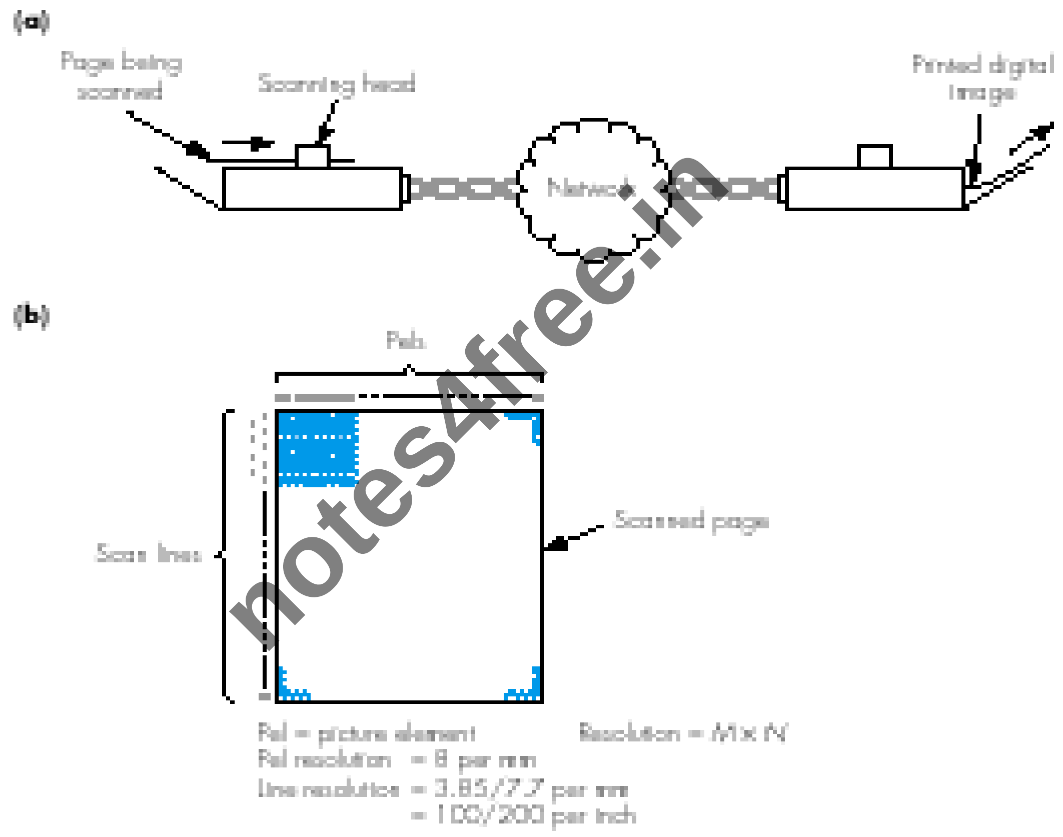
Figure 2.10 Graphics principles: (a) example screen format; (b) some simple object examples; (c) effect of changing position attribute; (d) solid objects.



2.4.2 Digitized documents

- **Fig 2.11**
- **A single binary digit to represent each pel, a 0 for a white pel and a 1 for a black pel**

Figure 2.11 Facsimile machine principles: (a) schematic; (b) digitization format.



2.4.3 Digitized pictures

- Color principles
- A whole spectrum of colors–known as a color gamut –can be produced by using different proportions of red(R), green(G), and blue (B)
- Fig 2.12
- Additive color mixing producing a color image on a black surface

- Subtractive color mixing for producing a color image on a white surface
- Fig 2.13

Figure 2.12 Color derivation principles: (a) additive color mixing; (b) subtractive color mixing.

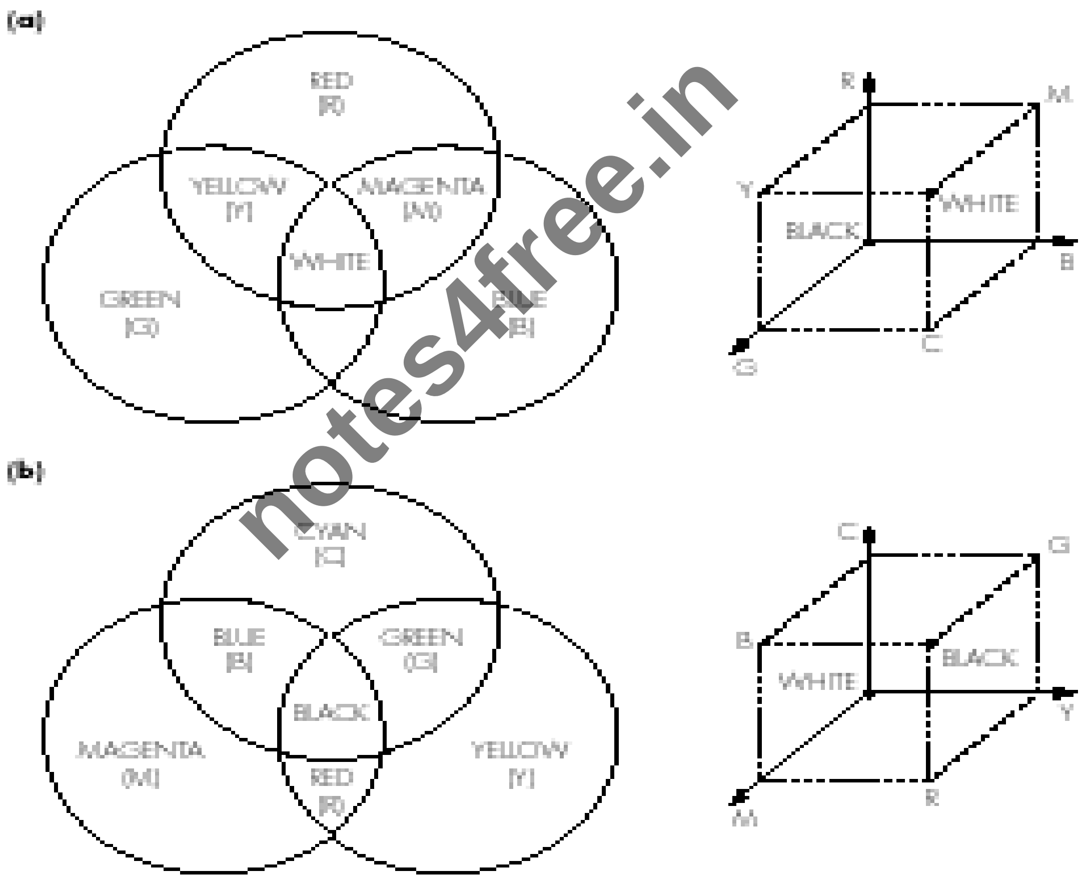
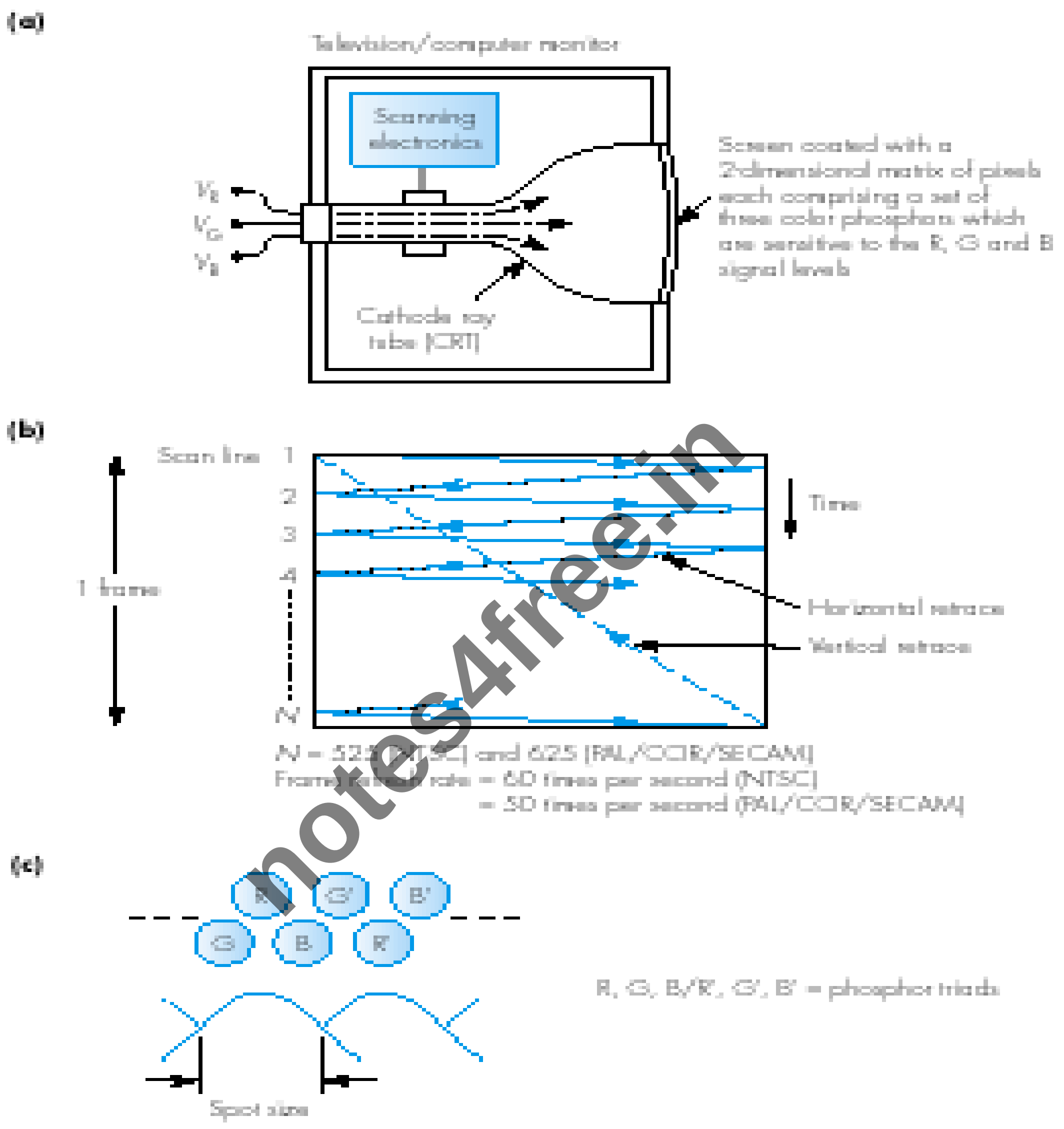


Figure 2.13 Television/computer monitor principles: (a) schematic; (b) raster-scan principles; (c) pixel format on each scan line.


(a)
Television/computer monitor
Scanning electronics
V_R
V_G
V_B
Screen coated with a 2-dimensional matrix of pixels each comprising a set of three color phosphors which are sensitive to the R, G and B signal levels
Cathode ray tube (CRT)
(b)
Scan line 1
2
3
4
N
1 frame
Time
Horizontal retrace
Vertical retrace
N = 525 (NTSC) and 625 (PAL/CCIR/SECAM)
Frame refresh rate = 60 times per second (NTSC)
= 50 times per second (PAL/CCIR/SECAM)
(c)
R
G'
B'
G
B
R'
R, G, B/R', G', B' = phosphor triads
Spot size

2.4.3 Digitized pictures

- Raster-scan principles
- Progressive scanning
- Each complete set of horizontal scan is called a frame
- The number of bits per pixel is known as the pixel depth and determines the range of different colors
- Aspect ratio

- Both the number of pixels per scanned line and the number of lines per frame
- The ratio of the screen width to the screen height
- National Television Standards Committee (NTSC), PAL(UK), CCIR(Germany), SECAM (France)
- Table 2.1

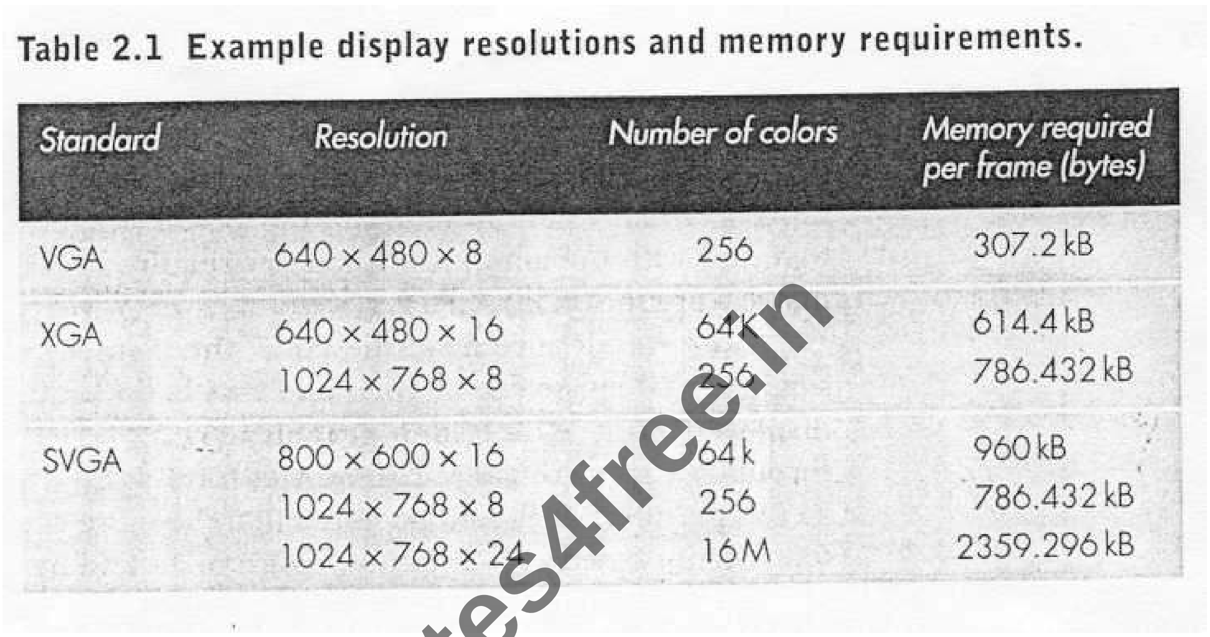
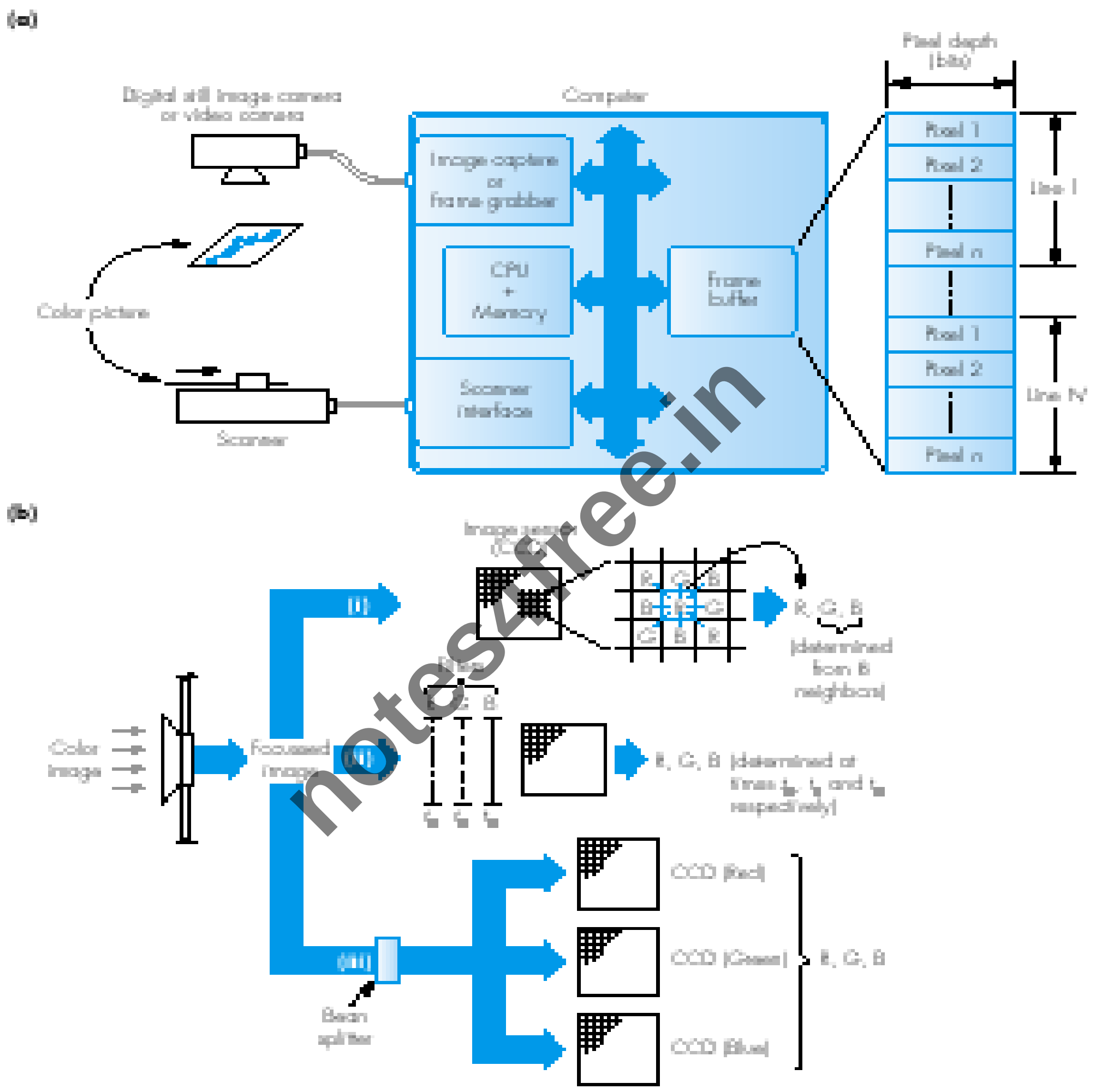


Table 2.1 Example display resolutions and memory requirements.

Standard	Resolution	Number of colors	Memory required per frame (bytes)
VGA	640 × 480 × 8	256	307.2 kB
XGA	640 × 480 × 16	64K	614.4 kB
	1024 × 768 × 8	256	786.432 kB
SVGA	800 × 600 × 16	64 k	960 kB
	1024 × 768 × 8	256	786.432 kB
	1024 × 768 × 24	16M	2359.296 kB

- Digital cameras and scanners
- An image is captured within the camera/scanner using an image sensor
- A two-dimensional grid of light-sensitive cells called photosites
- A widely-used image sensor is a charge-coupled device (CCD)
- Fig 2.16

Figure 2.16 Color image capture: (a) schematic; (b) RGB signal generation alternatives.



2.5 Audio

- The bandwidth of a typical speech signal is from 50Hz through to 10kHz; music signal from 15 Hz through to 20kHz
- The sampling rate: 20ksps (2*10kHz) for speech and 40ksps (2*20kHz) for music

- Music stereophonic (stereo) results in a bit rate double that of a monaural(mono) signal
- Example 2.4

2.5.2 CD-quality audio

- Bit rate per channel

=sampling rate*bits $= 44.1 \times 10^3 \times 16 = 705.6kbps$

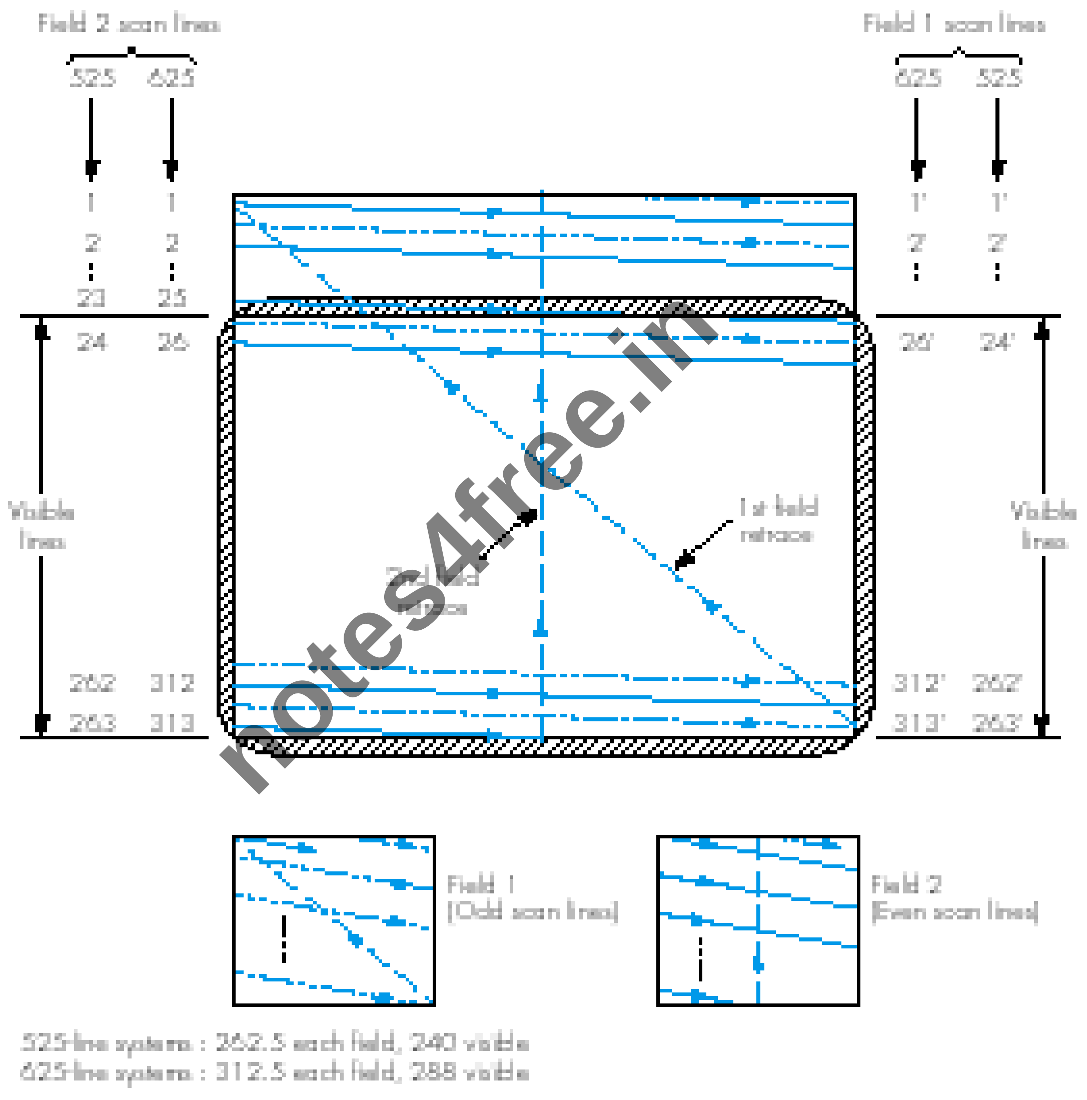
- Total bit rate = 2*705.6=1.411Mbps
- Example 2.5

2.6 Video

2.6.1 Broadcast television

- Scanning sequence
- It is necessary to use a minimum refresh rate of 50 times per second to avoid flicker
- A refresh rate of 25 times per second is sufficient
- Field:the first comprising only the odd scan lines and the second the even scan lines
- The two field are then integrated together in the television receiver using a technique known as interlaced scanning
- Fig 2.19
- The three main properties of a color source
  - Brightness
  - Hue:this represents the actual color of the source
  - Saturation:this represents the strength or vividness of the color

Figure 2.19 Interlaced scanning principles.



- The term luminance is used to refer to the brightness of a source
- The hue and saturation are referred to as its chrominance

$$Y_s = 0.299R_s + 0.587G_s + 0.144B_s$$

- Where Ys is the amplitude of the luminance signal and Rs,Gs and Bs are the magnitudes of the three color component signals

- The blue chrominance (Cb), and the red chrominance (Cr) are then used to represent hue and saturation
- The two color difference signals:

$$C_b = B_s - Y_s \qquad C_r = R_s - Y_s$$

- In the PAL system, Cb and Cr are referred to as *U* and *V* respectively

$$PAL: Y = 0.299R + 0.587G + 0.114B$$
$$U = 0.493(B - Y)$$
$$V = 0.877(R - Y)$$

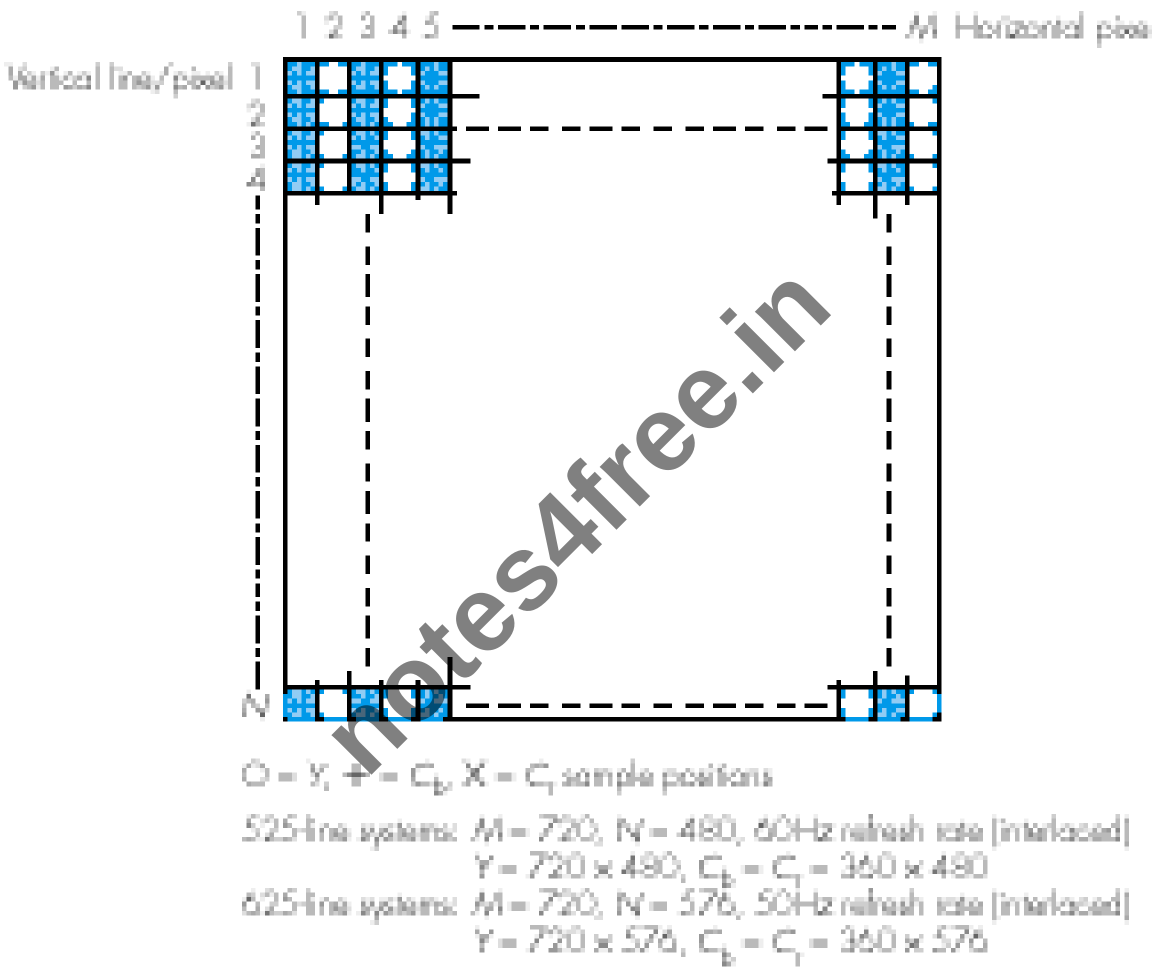
- The NTSC system form two different signals referred to as *I* and *Q*

$$NTSC: Y = 0.299R + 0.587G + 0.114B$$
$$I = 0.74(R - Y) - 0.27(B - Y)$$
$$Q = 0.48(R - Y) + 0.41(B - Y)$$

2.6.2 Digital video

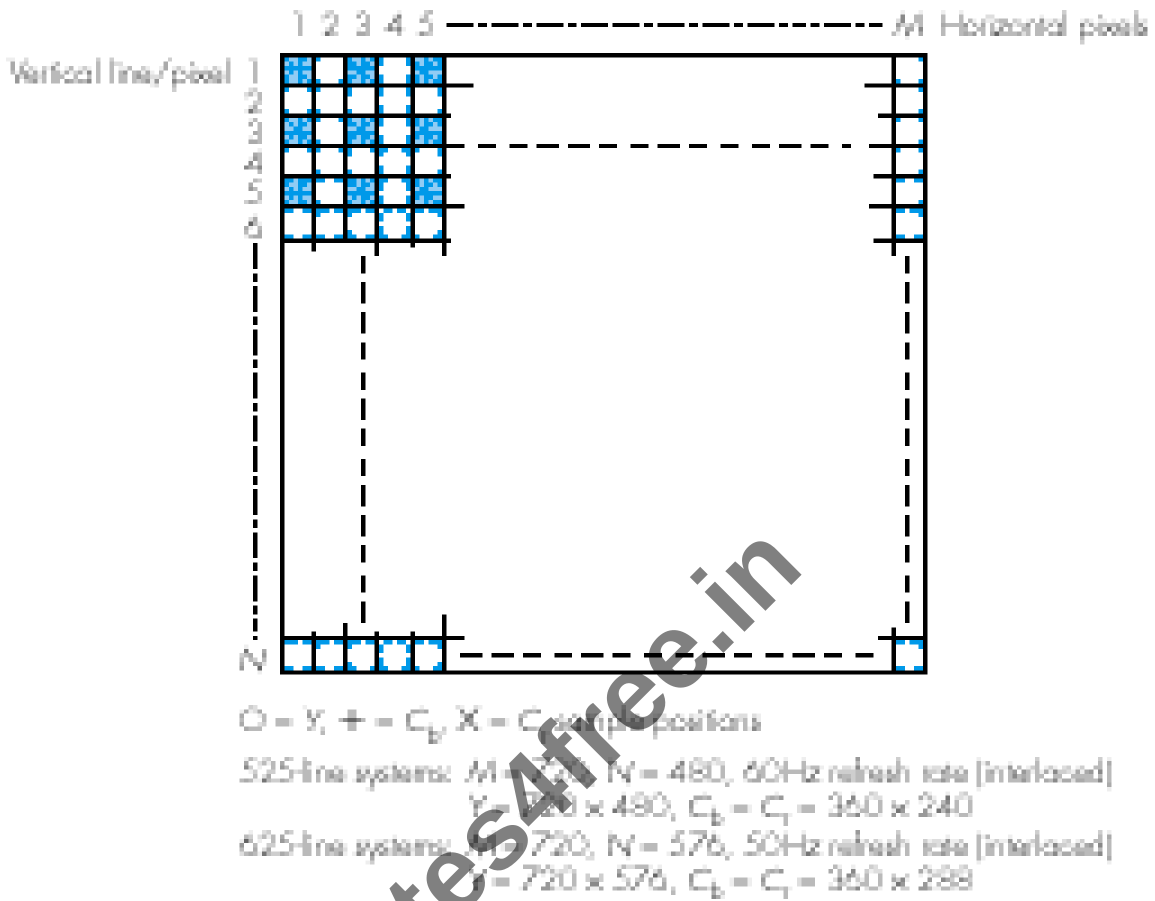
- Eye have shown that the resolution of the eye is less sensitive for color than it is for luminance
- 4：2：2 format
- The original digitization format used in Recommendation CCIR-601
- A line sampling rate of 13.5MHz for luminance and 6.75MHz for the two chrominance signals
- The number of samples per line is increased to 720
- The corresponding number of samples for each of the two chrominance signals is 360 samples per active line
- This results in 4Y samples for every 2Cb, and 2Cr samples
- The numbers 480 and 576 being the number of active (visible) lines in the respective system
- Fig. 2.21
- Example 2.7

Figure 2.21 Sample positions with 4:2:2 digitization format.


1 2 3 4 5
M Horizontal pixel
Vertical line/pixel 1
2
3
4
N
O = Y, + = C_b, X = C_r sample positions
525-line system: M = 720, N = 480, 60Hz refresh rate (interlaced)
Y = 720 × 480, C_b = C_r = 360 × 480
625-line system: M = 720, N = 576, 50Hz refresh rate (interlaced)
Y = 720 × 576, C_b = C_r = 360 × 576

- 4 : 2 : 0 format is used in digital video broadcast applications
- Interlaced scanning is used and the absence of chrominance samples in alternative lines
- The same luminance resolution but half the chrominance resolution
- Fig2.22

Figure 2.22 Sample positions in 4:2:0 digitization format.


1 2 3 4 5 M Horizontal pixels
Vertical line/pixel 1 2 3 4 5 6 N
○ = Y, + = C_b, X = C_r sample positions
525-line systems: M = 720, N = 480, 60Hz refresh rate (interlaced)
Y = 720 × 480, C_b = C_r = 360 × 240
625-line systems: M = 720, N = 576, 50Hz refresh rate (interlaced)
Y = 720 × 576, C_b = C_r = 360 × 288

525-line system

$$Y = 720 \times 480$$

$$C_b = C_r = 360 \times 240$$

625-line system

$$Y = 720 \times 480$$

$$C_b = C_r = 360 \times 288$$

$$13.5 \times 10^6 \times 8 + 2(3.375 \times 10^6 \times 8) = 162 Mbps$$

- HDTV formats: the resolution to the newer 16/9 wide-screen tubes can be up to 1920*1152 pixels
- The source intermediate format (SIF) give a picture quality comparable with video recorders(VCRs)

- The common intermediate format (CIF) for use in videoconferencing applications
- Fig 2.23
- The quarter CIF (QCIF) for use in video telephony applications
- **Fig 2.24**
- **Table 2.2**

Figure 2.23 Sample positions for SIF and CIF.

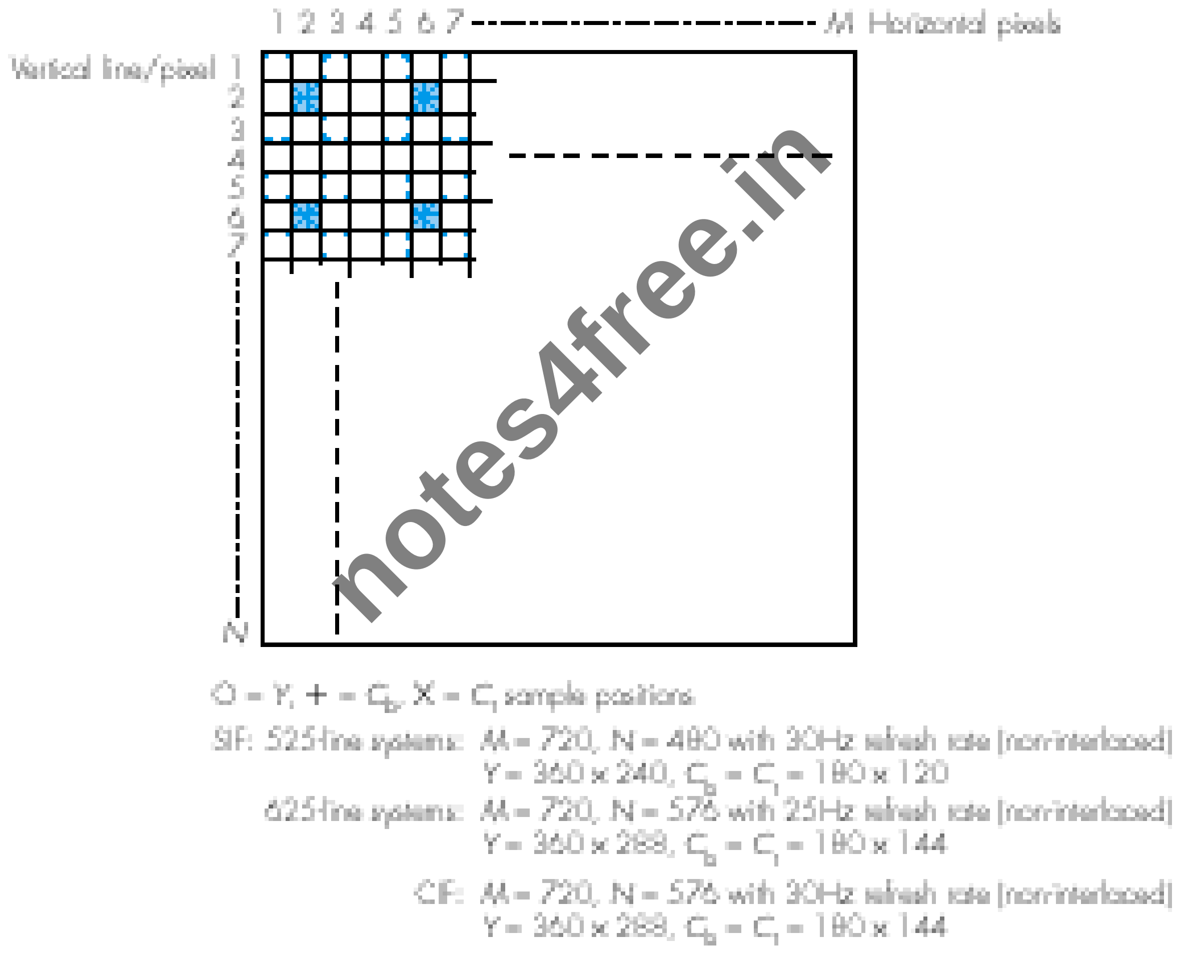
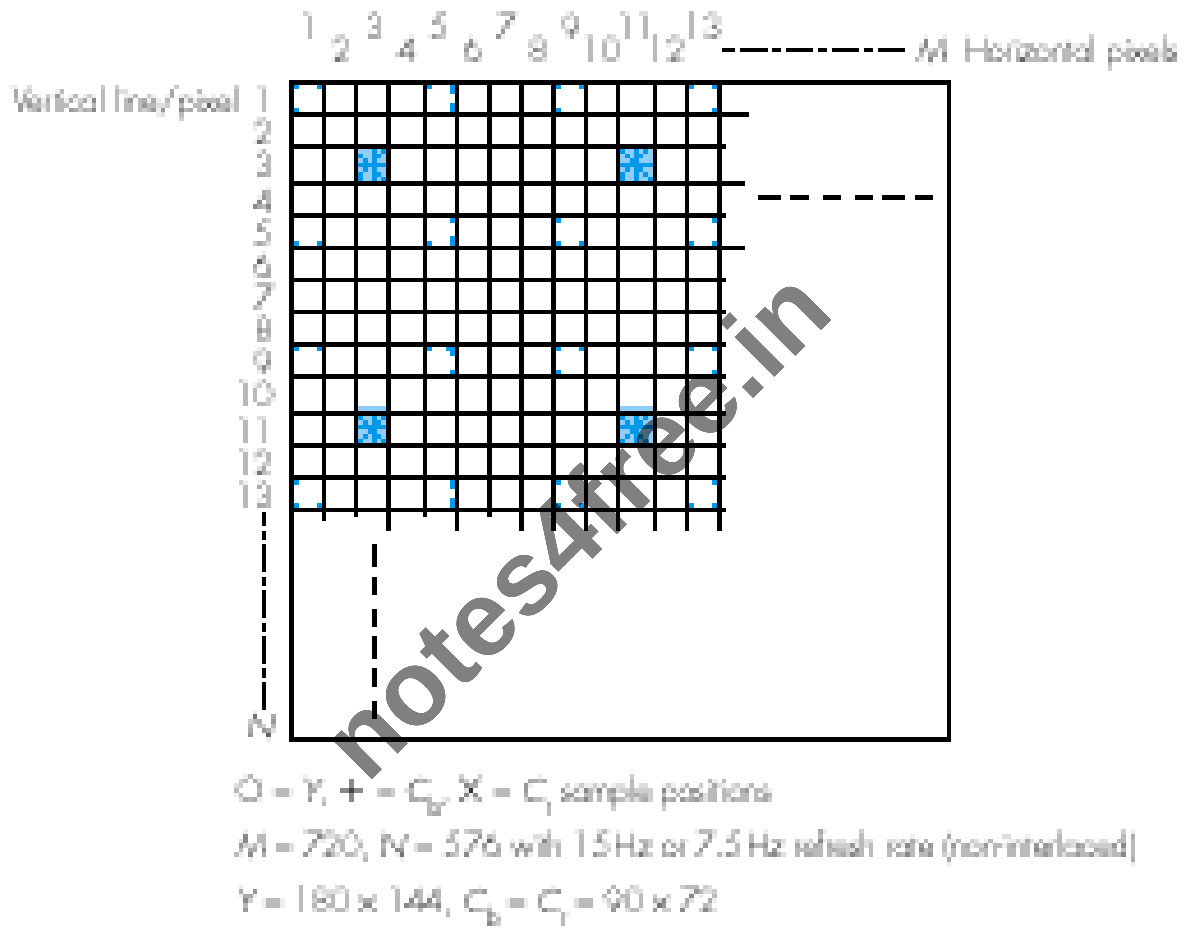

1 2 3 4 5 6 7 M Horizontal pixels
Vertical line/pixel 1 2 3 4 5 6 7 N
○ = Y, + = C_b, X = C_r sample positions
SIF: 525-line systems: M = 720, N = 480 with 30Hz refresh rate (non-interlaced)
Y = 360 x 240, C_b = C_r = 180 x 120
625-line systems: M = 720, N = 576 with 25Hz refresh rate (non-interlaced)
Y = 360 x 288, C_b = C_r = 180 x 144
CIF: M = 720, N = 576 with 30Hz refresh rate (non-interlaced)
Y = 360 x 288, C_b = C_r = 180 x 144

Figure 2.24 Sample positions for QCIF.


1 2 3 4 5 6 7 8 9 10 11 12 13
M Horizontal pixels
Vertical line/pixel
1 2 3 4 5 6 7 8 9 10 11 12 13
N
O = Y, + = C_b, X = C_r sample positions
M = 720, N = 576 with 15 Hz or 7.5 Hz refresh rate (non-interlaced)
Y = 180 x 144, C_b = C_r = 90 x 72

2.6.3 PC video

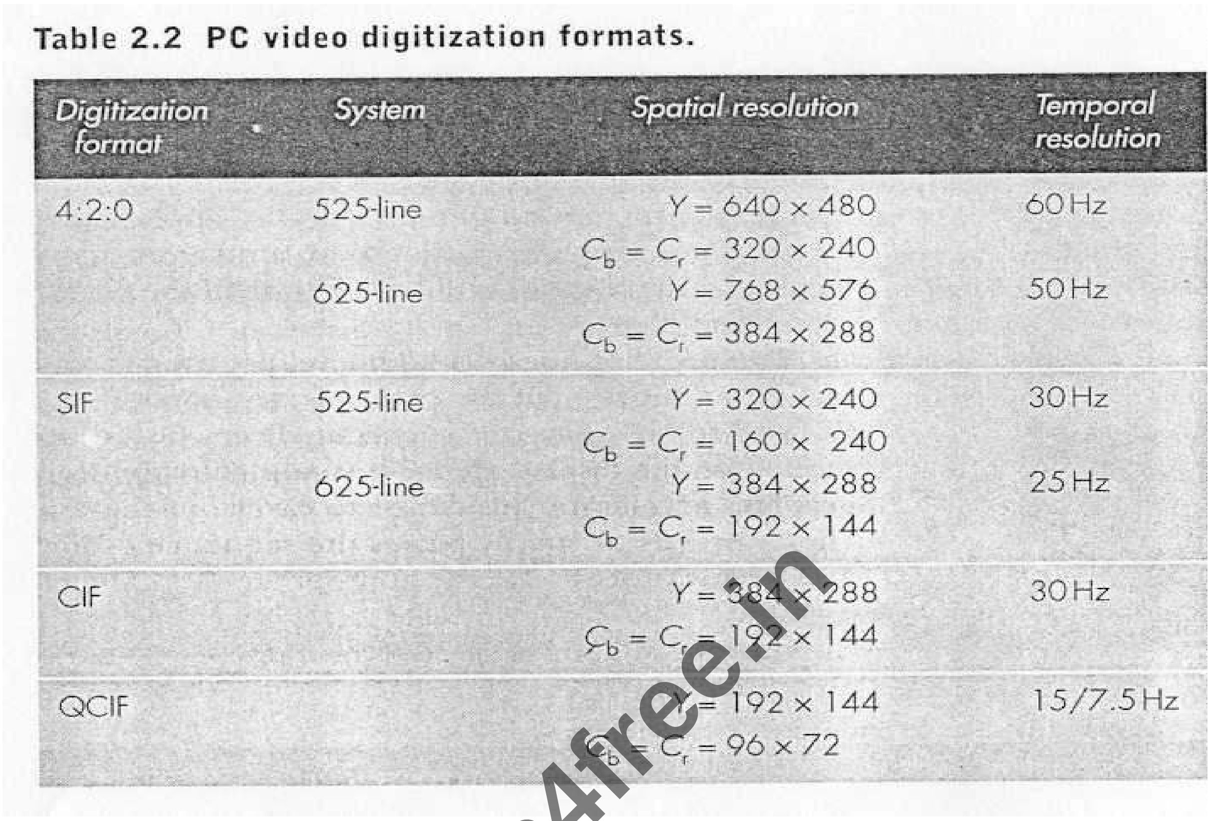


Table 2.2 PC video digitization formats.

Digitization format	System	Spatial resolution	Temporal resolution
4:2:0	525-line	$Y = 640 \times 480$	60 Hz
		$C_b = C_r = 320 \times 240$	
	625-line	$Y = 768 \times 576$	50 Hz
		$C_b = C_r = 384 \times 288$	
SIF	525-line	$Y = 320 \times 240$	30 Hz
		$C_b = C_r = 160 \times 240$	
	625-line	$Y = 384 \times 288$	25 Hz
		$C_b = C_r = 192 \times 144$	
CIF		$Y = 384 \times 288$	30 Hz
		$C_b = C_r = 192 \times 144$	
QCIF		$Y = 192 \times 144$	15/7.5 Hz
		$C_b = C_r = 96 \times 72$	

UNIT – 3

TEXT AND IMAGE COMPRESSION

Introduction, compression principles, text compression, image compression.

7 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & publications:

1. J. Choi and D. Park, A stable feedback control of the buffer state using the controlled
   Lagrange multiplier method, IEEE Trans. Image Processing, 3, 546–558 (1994).
2. Y. L. Lin and A. Ortega, Bit rate control using piecewise approximated rate-distortion
   characteristics, IEEE Trans CSVT, 8, 446–459 (1998).
3. W. Ding, Rate control of MPEG-video coding and recording by rate quantization modeling, IEEE Trans CSVT, 6, 12–20 (1966).
4. B. Tao, H. A. Peterson, and B. W. Dickinson, A rate-quantization model for MPEG encoders, Proc. IEEE ICIP, 1, 338–341 1997.
5. K. H. Yang, A. Jacquin, and N. S. Jayant, A normalized rate distortion model for H.263- compatible codecs and its application to quantizer selection, Proc. IEEE ICIP, 1, 41–44 (1997).

6. A. Velio, H. F. Sun, and Y. Wang, MPEG-4 rate control for multiple video objects, IEEE Trans. CSVT, 9, 186–199 (1999).
7. Y. Ribos-Corbero and S. M. Lei, JTC1/SC29/WG11 MPEG96/M1820, Contribution to
Rate Control Q2 Experiment: A Quantization Control Tool for Achieving Target Bitrate Accurately, Sevilla, Spain (1997).

Introduction

Compression is used just about everywhere. All the images you get on the web are compressed, typically in the JPEG or GIF formats, most modems use compression, HDTV will be compressed using MPEG-2, and several file systems automatically compress files when stored, and the rest of us do it by hand. The neat thing about compression, as with the other topics we will cover in this course, is that the algorithms used in the real world make heavy use of a wide set of algorithmic tools, including sorting, hash tables, tries, and FFTs. Furthermore, algorithms with strong theoretical foundations play a critical role in real-world applications.

In this chapter we will use the generic term *message* for the objects we want to compress, which could be either files or messages. The task of compression consists of two components, an *encoding* algorithm that takes a message and generates a “compressed” representation (hopefully with fewer bits), and a *decoding* algorithm that reconstructs the original message or some approximation of it from the compressed representation. These two components are typically intricately tied together since they both have to understand the shared compressed representation.

We distinguish between *lossless algorithms*, which can reconstruct the original message exactly from the compressed message, and *lossy algorithms*, which can only reconstruct an approximation of the original message. Lossless algorithms are typically used for text, and lossy for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable. Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise. For example, one might think that lossy text compression would be unacceptable because they are imagining missing or switched characters.

Consider instead a system that reworded sentences into a more standard form, or replaced words with synonyms so that the file can be better compressed. Technically the compression would be lossy since the text has changed, but the “meaning” and clarity of the message might be fully maintained, or even improved. In fact Strunk and White might argue that good writing is the art of lossy text compression.

Because one can’t hope to compress everything, all compression algorithms must assume that there is some bias on the input messages so that some inputs are more likely than others, *i.e.* that there is some unbalanced probability distribution over the possible messages. Most compression algorithms base this “bias” on the structure of the messages – *i.e.*, an assumption that repeated characters are more likely than random characters, or that large white patches occur in “typical” images. Compression is therefore all about probability.

When discussing compression algorithms it is important to make a distinction between two components: the model and the coder. The *model* component somehow captures the probability distribution of the messages by knowing or discovering something about the structure of the input.

The *coder* component then takes advantage of the probability biases generated in the model to generate codes. It does this by effectively lengthening low probability messages and shortening high-probability messages. A model, for example, might have a generic “understanding” of human faces knowing that some “faces” are more likely than others (*e.g.*, a teapot would not be a very likely face). The coder would then be able to send shorter messages for objects that look like faces. This could work well for compressing teleconference calls. The models in most current real-world compression algorithms, however, are not so sophisticated, and use more mundane measures such as repeated patterns in text. Although there are many different ways to design the model component of compression algorithms and a huge range of levels of sophistication, the coder components tend to be quite generic—in current algorithms are almost exclusively based on either Huffman or arithmetic codes. Lest we try to make to fine of a distinction here, it should be pointed out that the line between model and coder components of algorithms is not always well defined. It turns out that information theory is the glue that ties the model and coder components together. In particular it gives a very nice theory about how probabilities are related to information content and code length. As we will see, this theory matches practice almost perfectly, and we can achieve code lengths almost identical to what the theory predicts.

Another question about compression algorithms is how does one judge the quality of one versus another. In the case of lossless compression there are several criteria I can think of, the time to compress, the time to reconstruct, the size of the compressed messages, and the generality—*i.e.*, does it only work on Shakespeare or does it do Byron too. In the case of lossy compression the judgement is further complicated since we also have to worry about how good the lossy approximation is. There are typically tradeoffs between the amount of compression, the runtime, and the quality of the reconstruction. Depending on your application one might be more important than another and one would want to pick your algorithm appropriately. Perhaps the best attempt to systematically compare lossless compression algorithms is the Archive Comparison Test (ACT) by Jeff Gilchrist. It reports times and compression ratios for 100s of compression algorithms over many databases. It also gives a score based on a weighted average of runtime and the compression ratio.

<u>Compression principles:</u>

Compression in Multimedia Data:

Compression basically employs redundancy in the data:

_ Temporal — in 1D data, 1D signals, Audio etc.

_ Spatial — correlation between neighbouring pixels or data items

_ Spectral — correlation between colour or luminescence components. This uses the frequency domain to exploit relationships between frequency of change in data.
_ Psycho-visual — exploit perceptual properties of the human visual system.

Lossless v Lossy Compression
Compression can be categorised in two broad ways:
Lossless Compression — after decompression gives an exact copy of the original data
Examples: Entropy Encoding Schemes (Shannon-Fano, Huffman coding), arithmetic coding,LZW algorithm used in GIF image file format.

Lossy Compression — after decompression gives ideally a 'close' approximation of the original data, in many cases perceptually lossless but a byte-by-byte comparision of files shows differences.
Examples: Transform Coding — FFT/DCT based quantisation used in JPEG/MPEG differential encoding, vector quantisation.

Why do we need Lossy Compression?
Lossy methods for **typically** applied to high resoultion audio, image compression **Have to be employed** in video compression (apart from special cases).

Basic reason: Compression ratio of lossless methods (e.g., Huffman coding, arithmetic coding, LZW) is not high enough.

Lossless Compression Algorithms
Entropy Encoding
_ Lossless Compression frequently involves some form of **entropy encoding**
_ Based on **information theoretic techniques**.

Basics of Information Theory
According to Shannon, the **entropy** of an information source S is defined as:

$$H(S) = \eta = \sum_i \; p_i \log_2 \frac{1}{p_i}$$

where pi is the probability that symbol Si in S will occur.

$$\log_2 \frac{1}{p_i}$$

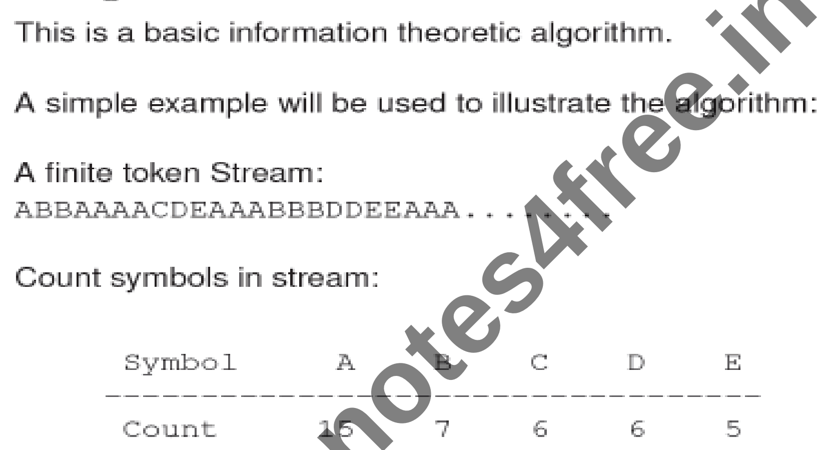
indicates the amount of information contained in Si, i.e., the number of bits needed to code Si.

_ For example, in an image with uniform distribution of gray-level intensity, i.e. pi = 1/256, then

– The number of bits needed to code each gray level is 8 bits.

– The entropy of this image is 8.

The Shannon-Fano Algorithm — Learn by Example



This is a basic information theoretic algorithm.

A simple example will be used to illustrate the algorithm:

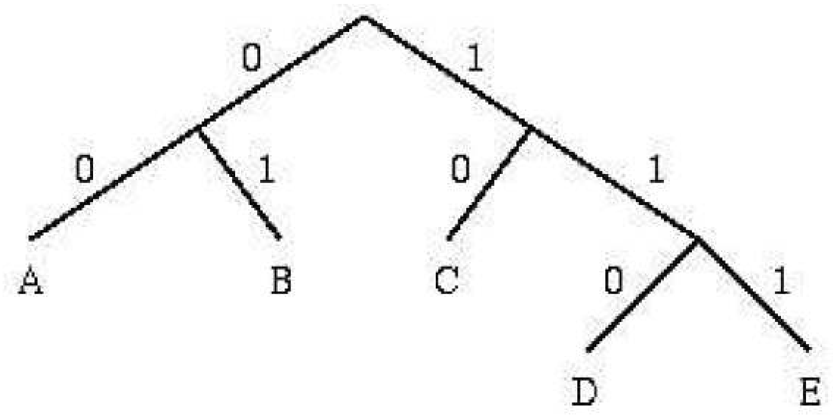
A finite token Stream:
```
ABBAAAACDEAAABBBDDEEAAA........
```

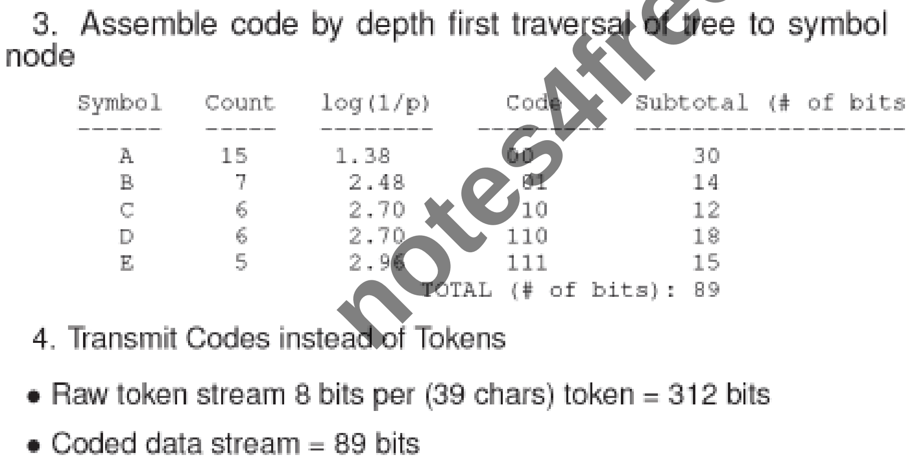
Count symbols in stream:

Symbol	A	B	C	D	E
Count	15	7	6	6	5

A top-down approach

1. Sort symbols (Tree Sort) according to their frequencies/probabilities, e.g., ABCDE.
2. Recursively divide into two parts, each with approx. Same number of counts.




3. Assemble code by depth first traversal of tree to symbol node

Symbol	Count	log(1/p)	Code	Subtotal (# of bits
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15
			TOTAL (# of bits):	89

4. Transmit Codes instead of Tokens

- Raw token stream 8 bits per (39 chars) token = 312 bits
- Coded data stream = 89 bits

Huffman Coding

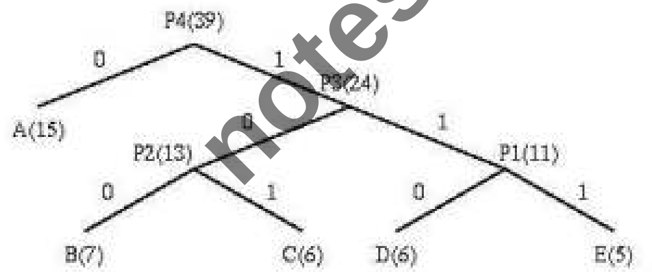
_ Based on the frequency of occurrence of a data item (pixels or small blocks of pixels in images).
_ Use a lower number of bits to encode more frequent data
_ Codes are stored in a **Code Book**—as for Shannon (previous slides)
_ Code book constructed for each image or a set of images.

_ Code book **plus** encoded data **must** be transmitted to enable decoding.

Encoding for Huffman Algorithm:
A bottom-up approach
1. Initialization: Put all nodes in an OPEN list, keep it sorted at all times (e.g., ABCDE).
2. Repeat until the OPEN list has only one node left:
(a) From OPEN pick two nodes having the lowest frequencies/probabilities, create a parent node of them.
(b) Assign the sum of the children's frequencies/probabilities to the parent node and insert it into OPEN.
(c) Assign code 0, 1 to the two branches of the tree, and delete the children from OPEN.
3. Coding of each node is a top-down label of branch labels.

Huffman Encoding Example:

ABBAAAACDEAAABBBDDEEAAA........ (Same as Shannon-Fano E.g.)



```
 Symbol   Count    log(1/p)      Code    Subtotal (# of bits)
 ------   -----   ---------   ---------  ------------------
   A       15       1.38          0           15
   B        7       2.48        100           21
   C        6       2.70        101           18
   D        6       2.70        110           18
   E        5       2.96        111           15
                          TOTAL (# of bits):  87
```

Huffman Encoder Analysis

The following points are worth noting about the above algorithm:

_ Decoding for the above two algorithms is trivial as long as the coding table/book is sent before the data.

– There is a bit of an overhead for sending this.

– But negligible if the data file is big.

_ **Unique Prefix Property**: no code is a prefix to any other code (all symbols are at the leaf nodes) –> great for decoder, unambiguous.

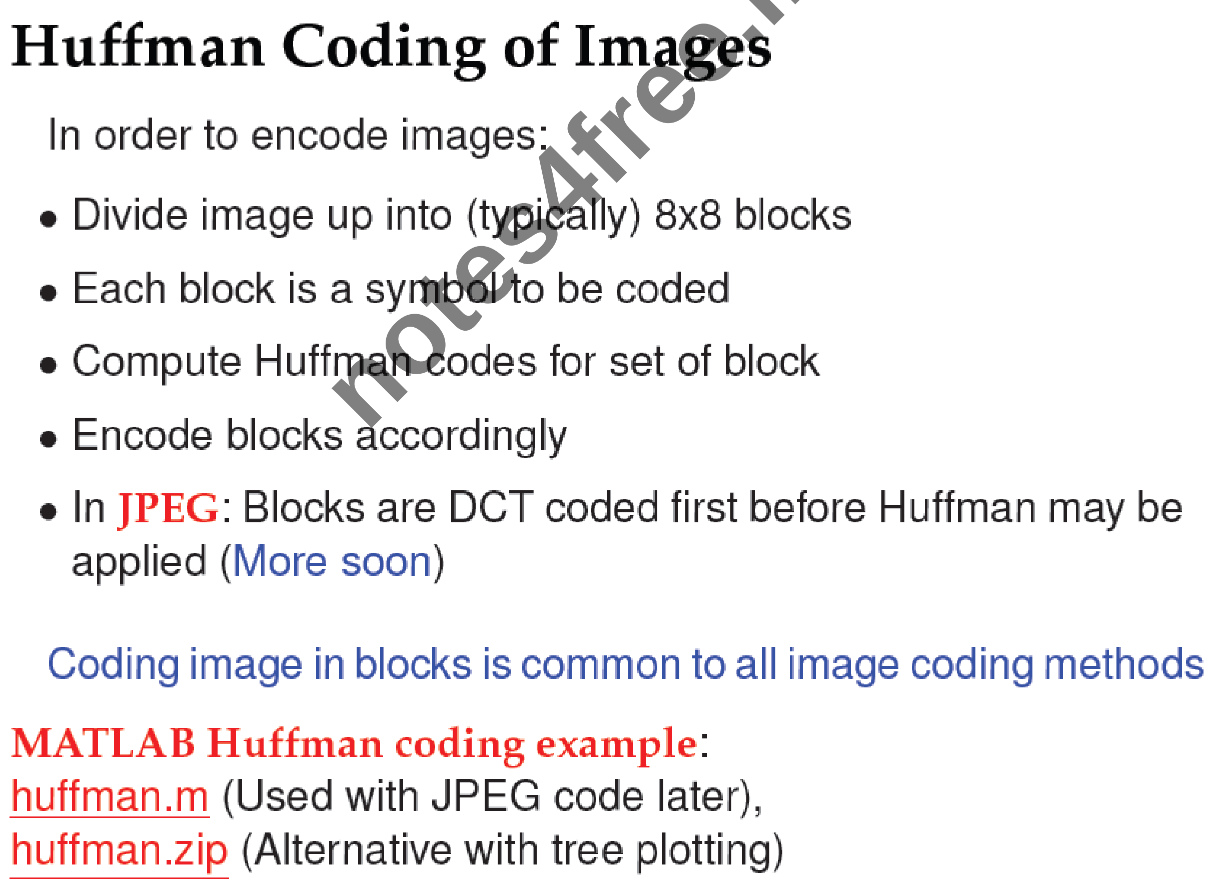
_ If prior statistics are available and accurate, then Huffman coding is very good.

Huffman Entropy

In the above example:

$$\begin{aligned}\text{Idealentropy} &= (15 * 1.38 + 7 * 2.48 + 6 * 2.7 \\ &\quad +6 * 2.7 + 5 * 2.96)/39 \\ &= 85.26/39 \\ &= 2.19\end{aligned}$$

Number of bits needed for Huffman Coding is: $87/39 = 2.23$



Huffman Coding of Images

In order to encode images:

- Divide image up into (typically) 8x8 blocks
- Each block is a symbol to be coded
- Compute Huffman codes for set of block
- Encode blocks accordingly
- In **JPEG**: Blocks are DCT coded first before Huffman may be applied (More soon)

Coding image in blocks is common to all image coding methods

MATLAB Huffman coding example:
huffman.m (Used with JPEG code later),
huffman.zip (Alternative with tree plotting)

Arithmetic Coding

- A widely used entropy coder
- Also used in JPEG — more soon
- Only problem is it's speed due possibly complex computations due to large symbol tables,
- Good compression ratio (better than Huffman coding), entropy around the Shannon Ideal value.

Why better than Huffman?

- Huffman coding etc. use an integer number (k) of bits for each symbol,
  - hence k is never less than 1.
- Sometimes, e.g., when sending a 1-bit image, compression becomes impossible.

Decimal Static Arithmetic Coding

- Here we describe basic approach of Arithmetic Coding
- Initially basic static coding mode of operation.
- Initial example decimal coding
- Extend to Binary and then machine word length later

Basic Idea

The idea behind arithmetic coding is

- To have a probability line, 0–1, and
- Assign to every symbol a range in this line based on its probability,
- The higher the probability, the higher range which assigns to it.

Once we have defined the ranges and the probability line,

- Start to encode symbols,
- Every symbol defines where the output floating point number lands within the range.

Simple Basic Arithmetic Coding Example

Assume we have the following token symbol stream

BACA

Therefore

- A occurs with probability 0.5,
- B and C with probabilities 0.25.

Basic Arithmetic Coding Algorithm

Start by assigning each symbol to the probability range 0–1.

- Sort symbols highest probability first

Symbol	Range
A	[0.0, 0.5)
B	[0.5, 0.75)
C	[0.75, 1.0)

- The first symbol in our example stream is B

We now know that the code will be in the range 0.5 to 0.74999 . . ..

Range is not yet unique

- Need to narrow down the range to give us a unique code.

Basic arithmetic coding iteration

- Subdivide the range for the first token given the probabilities of the second token then the third etc.

Subdivide the range as follows

For all the symbols:

```
range = high - low;
high = low + range * high_range of the symbol being coded;
low = low + range * low_range of the symbol being coded;
```

Where:

- `range`, keeps track of where the next range should be.
- `high` and `low`, specify the output number.
- Initially `high = 1.0`, `low = 0.0`

Back to our example

The second symbols we have
(now `range` = 0.25, `low` = 0.5, `high` = 0.75):

Symbol	Range
BA	[0.5, 0.625)
BB	[0.625, 0.6875)
BC	[0.6875, 0.75)

Third Iteration

We now reapply the subdivision of our scale again to get for our third symbol
(range = 0.125, low = 0.5, high = 0.625):

Symbol	Range
BAA	[0.5, 0.5625)
BAB	[0.5625, 0.59375)
BAC	[0.59375, 0.625)

Fourth Iteration

Subdivide again
(range = 0.03125, low = 0.59375, high = 0.625):

Symbol	Range
BACA	[0.59375, 0.60937)
BACB	[0.609375, 0.6171875)
BACC	[0.6171875, 0.625)

So the (Unique) output code for BACA is any number in the range:

[0.59375, 0.60937).

Decoding

To **decode** is essentially the opposite

- We compile the table for the sequence given probabilities.
- Find the range of number within which the code number lies and carry on

Binary static algorithmic coding

This is very similar to above:

- **Except** we us **binary fractions**.

Binary fractions are simply an extension of the binary systems into fractions much like decimal fractions.

Binary Fractions — Quick Guide

Fractions in **decimal**:

0.1 decimal = $\frac{1}{10^1} = 1/10$
0.01 decimal = $\frac{1}{10^2} = 1/100$
0.11 decimal = $\frac{1}{10^1} + \frac{1}{10^2} = 11/100$

So in **binary** we get

0.1 binary = $\frac{1}{2^1} = 1/2$ decimal
0.01 binary = $\frac{1}{2^2} = 1/4$ decimal
0.11 binary = $\frac{1}{2^1} + \frac{1}{2^2}$ = 3/4 decimal

Binary Arithmetic Coding Example

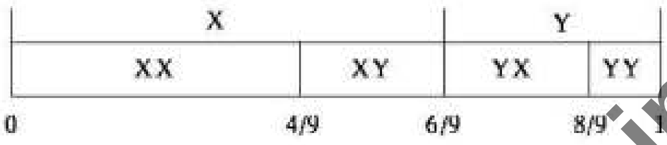
- Idea: Suppose alphabet was X, Y and token stream:

  XXY

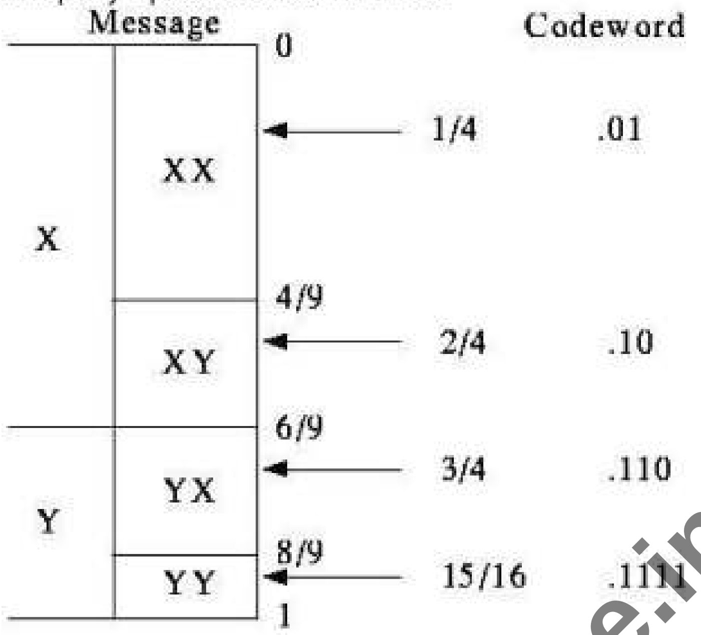
  Therefore:

  ```
      prob(X) = 2/3
      prob(Y) = 1/3
  ```

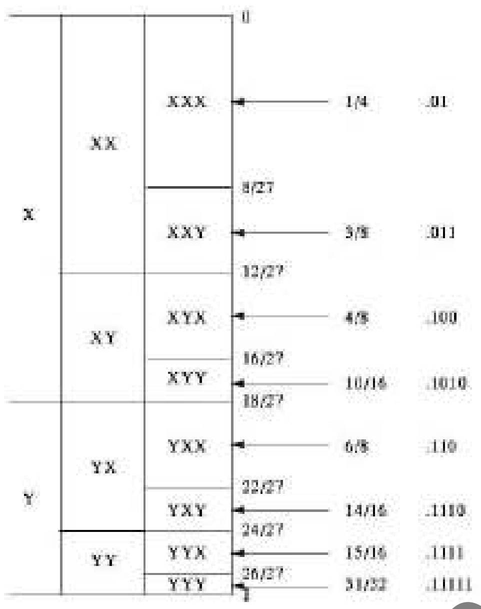
- If we are only concerned with encoding length 2 messages, then we can map all possible messages to intervals in the range [0..1]:



- To encode message, just send enough bits of a binary fraction that uniquely specifies the interval.



- Similarly, we can map all possible length 3 messages to intervals in the range [0..1]:



Implementation Issues

FPU Precision

- Resolution of the of the number we represent is limited by FPU precision
- Binary coding extreme example of rounding
- Decimal coding is the other extreme — theoretically no rounding.
- Some FPUs may us up to 80 bits
- As an example let us consider working with 16 bit resolution.

16-bit arithmetic coding

We now encode the range 0–1 into 65535 segments:

0.000	0.250	0.500	0.750	1.000
0000h	4000h	8000h	C000h	FFFFh

If we take a number and divide it by the maximum (FFFFh) we will clearly see this:

```
0000h: 0/65535 = 0.0
4000h: 16384/65535 = 0.25
8000h: 32768/65535 = 0.5
C000h: 49152/65535 = 0.75
FFFFh: 65535/65535 = 1.0
```

The operation of coding is similar to what we have seen with the binary coding:

- Adjust the probabilities so the bits needed for operating with the number aren't above 16 bits.
- Define a new interval
- The way to deal with the infinite number is
  - to have only loaded the 16 first bits, and when needed shift more onto it:
    `1100 0110 0001 000 0011 0100 0100 ...`
  - work only with those bytes
  - as new bits are needed they'll be shifted.

Memory Intensive

What about an alphabet with 26 symbols, or 256 symbols, ...?

- In general, number of bits is determined by the size of the interval.
- In general, (from entropy) need $-\log p$ bits to represent interval of size p.
- Can be memory and CPU intensive

MATLAB Arithmetic coding examples:
Arith06.m (Version 1),
Arith07.m (Version 2)

Lempel-Ziv-Welch (LZW) Algorithm

- A very common compression technique.
- Used in GIF files (LZW), Adobe PDF file (LZW), UNIX `compress` (LZ Only)
- Patented — LZW not LZ.

Basic idea/Example by Analogy:

Suppose we want to encode the Oxford Concise English dictionary which contains about 159,000 entries.

Why not just transmit each word as an 18 bit number?

LZW Constructs Its Own Dictionary

Problems:

- Too many bits per word,
- Everyone needs a dictionary,
- Only works for English text.

Solution:

- Find a way to build the dictionary adaptively.
- Original methods (LZ) due to Lempel and Ziv in 1977/8.
- Quite a few variations on LZ.
- Terry Welch improvement (1984), **Patented LZW Algorithm**
  - LZW introduced the idea that only the **initial dictionary** needs to be transmitted to enable **decoding**:
    The decoder is able to **build** the **rest** of the table from the **encoded sequence**.

LZW Compression Algorithm

The LZW Compression Algorithm can summarised as follows:

```
w = NIL;
while ( read a character k )
  {    if wk exists in the dictionary
             w = wk;
         else
             { add wk to the dictionary;
                output the code for w;
                 w = k;
              }
   }
```

- Original LZW used dictionary with 4K entries, first 256 (0-255) are ASCII codes.

LZW Compression Algorithm Example:

Input string is "^WED^WE^WEE^WEB^WET".

w	k	output	index	symbol
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

- A 19-symbol input has been reduced to 7-symbol plus 5-code output. Each code/symbol will need more than 8 bits, say 9 bits.
- Usually, compression doesn't start until a large number of bytes (e.g., > 100) are read in.

LZW Decompression Algorithm

The LZW Decompression Algorithm is as follows:

```
read a character k;
output k;
w = k;
while ( read a character k )
/* k could be a character or a code. */
  {
          entry = dictionary entry for k;
          output entry;
          add w + entry[0] to dictionary;
          w = entry;
  }
```

Note (Recall):
LZW decoder only needs the **initial dictionary**:
The decoder is able to **build** the **rest** of the table from the **encoded sequence**.

LZW Decompression Algorithm Example:

Input string is
"^WED<256>E<260><261><257>B<260>T"

w	k	output	index	symbol
^	^			
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
<256>	E	E	260	^WE
E	<260>	^WE	261	E^
<260>	<261>	E^	262	^WEE
<261>	<257>	WE	263	E^W
<257>	B	B	264	WEB
B	<260>	^WE	265	B^
<260>	T	T	266	^WET

Lossy Compression: Source Coding Techniques

Source coding is based changing on the content of the original signal.

Also called *semantic-based coding*

High compression rates may be high but a price of loss of information. Good compression rates make be achieved with source encoding with (occasionally) **lossless** or (mostly) little **perceivable** loss of information.

There are three broad methods that exist:

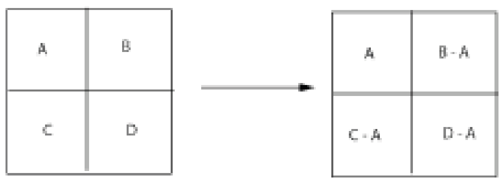
- Transform Coding
- Differential Encoding
- Vector Quantisation

Transform Coding

A simple transform coding example

A Simple Transform Encoding procedure maybe described by the following steps for a 2x2 block of monochrome pixels:

1. Take top left pixel as the base value for the block, pixel A.
2. Calculate three other transformed values by taking the difference between these (respective) pixels and pixel A, Ii.e. B-A, C-A, D-A.
3. Store the base pixel and the differences as the values of the transform.



Simple Transforms

Given the above we can easily form the forward transform:

$$\begin{aligned} X_0 &= A \\ X_1 &= B - A \\ X_2 &= C - A \\ X_3 &= D - A \end{aligned}$$

and the inverse transform is:

$$\begin{aligned} A_n &= X_0 \\ B_n &= X_1 + X_0 \\ C_n &= X_2 + X_0 \\ D_n &= X_3 + X_0 \end{aligned}$$

Compressing data with this Transform?

Exploit redundancy in the data:

- Redundancy transformed to values, X_i.
- Compress the data by using fewer bits to represent the differences — *Quantisation*.
  - I.e if we use 8 bits per pixel then the 2x2 block uses 32 bits
  - If we keep 8 bits for the base pixel, X0,
  - Assign 4 bits for each difference then we only use 20 bits.
  - Better than an average 5 bits/pixel

Example

Differential Encoding

Simple example of transform coding mentioned earlier and instance of this approach.

Here:

- The difference between the actual value of a sample and a prediction of that values is encoded.
- Also known as **predictive encoding**.
- Example of technique include: differential pulse code modulation, delta modulation and adaptive pulse code modulation — differ in prediction part.
- Suitable where successive signal samples do not differ much, but are not zero. **E.g.** Video — difference between frames, some audio signals.

Differential Encoding Methods

- **Differential pulse code modulation** (DPCM)

Simple prediction (also used in JPEG):

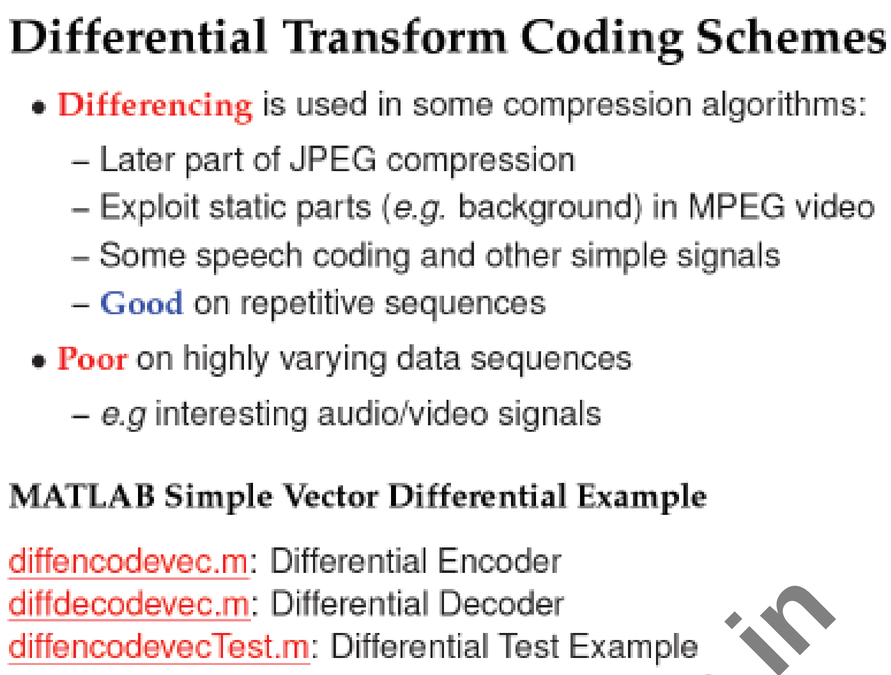
$$f_{predict}(t_i) = f_{actual}(t_{i-1})$$

I.e. a simple Markov model where current value is the predict next value.

So we simply need to encode:

$$\Delta f(t_i) = f_{actual}(t_i) - f_{actual}(t_{i-1})$$

If successive sample are close to each other we only need to encode first sample with a large number of bits:



Differential Transform Coding Schemes

- Differencing is used in some compression algorithms:
  - Later part of JPEG compression
  - Exploit static parts (*e.g.* background) in MPEG video
  - Some speech coding and other simple signals
  - Good on repetitive sequences
- Poor on highly varying data sequences
  - *e.g* interesting audio/video signals

MATLAB Simple Vector Differential Example

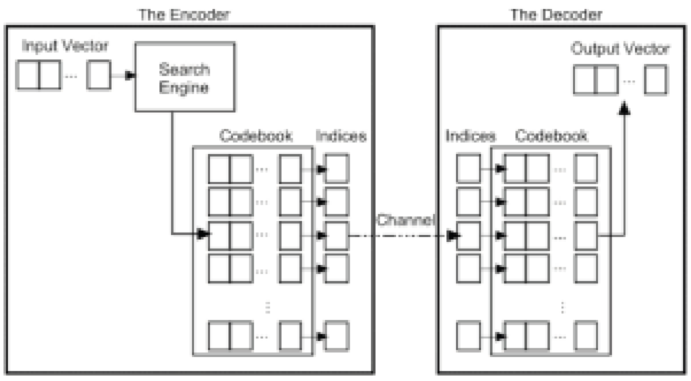
diffencodevec.m: Differential Encoder
diffdecodevec.m: Differential Decoder
diffencodevecTest.m: Differential Test Example

Vector Quantisation

The basic outline of this approach is:

- Data stream divided into (1D or 2D square) blocks — **vectors**
- A table or **code book** is used to find a pattern for each block.
- Code book can be dynamically constructed or predefined.
- Each pattern for block encoded as a look value in table
- Compression achieved as data is effectively subsampled and coded at this level.
- Used in MPEG4, Video Codecs (Cinepak, Sorenson), Speech coding, Ogg Vorbis.

Vector Quantisation Encoding/Decoding

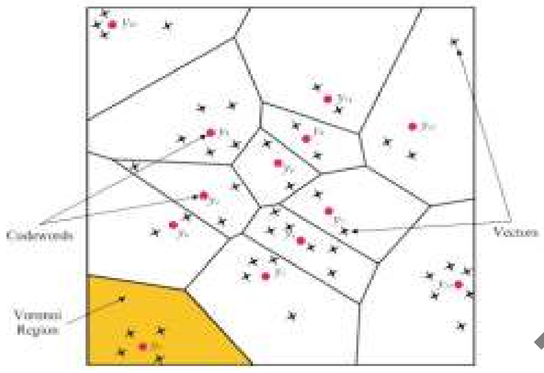


- Search Engine:
  - Group (Cluster) data into vectors
  - Find closest code vectors
- On decode output need to *unblock* (smooth) data

Vector Quantisation Code Book Construction

How to cluster data?

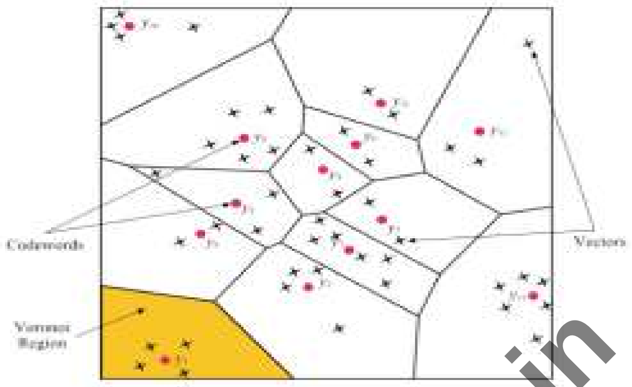
- Use some clustering technique,
  e.g. K-means, Voronoi decomposition
  Essentially cluster on some closeness measure, minimise inter-sample variance or distance.



Vector Quantisation Code Book Construction

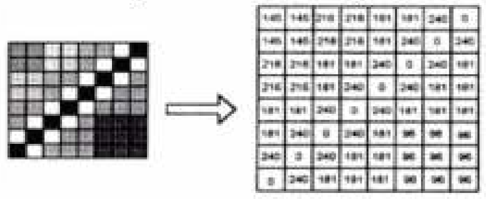
How to code?

- For each cluster choose a mean (median) point as representative code for all points in cluster.



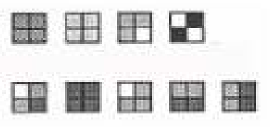
Vector Quantisation Image Coding Example

- A small block of images and intensity values



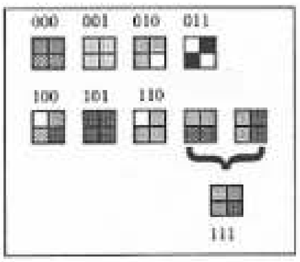
145	145	216	216	181	181	240	0
145	145	216	216	181	240	0	240
216	216	181	181	240	0	240	181
216	216	181	240	0	240	181	181
181	181	240	0	240	181	181	181
181	240	0	240	181	96	96	96
240	0	240	181	181	96	96	96
0	240	181	181	181	96	96	96

- Consider Vectors of 2x2 blocks, and only allow 8 codes in table.
- 9 vector blocks present in above:



Vector Quantisation Image Coding Example (Cont.)

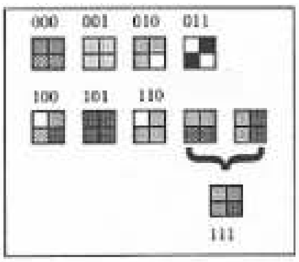
- 9 vector blocks, so **only one** has to be **vector quantised** here.
- Resulting code book for above image



MATLAB EXAMPLE: vectorquantise.m

Vector Quantisation Image Coding Example (Cont.)

- 9 vector blocks, so **only one** has to be **vector quantised** here.
- Resulting code book for above image



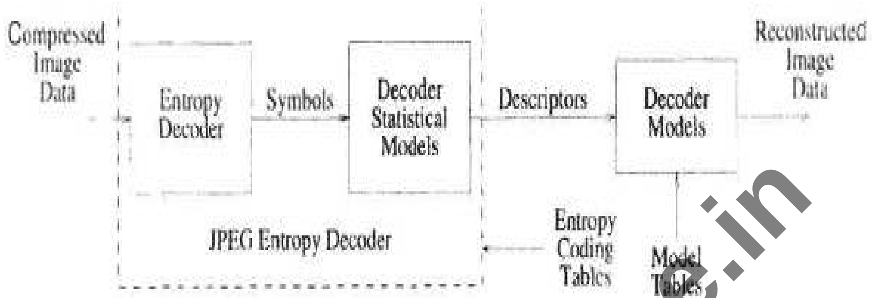
MATLAB EXAMPLE: vectorquantise.m

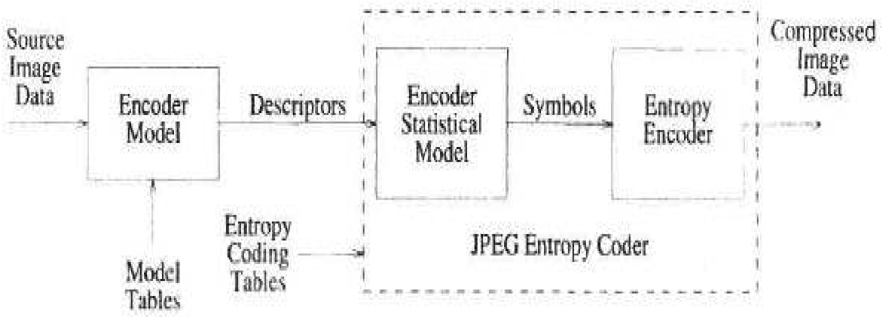
LOSSY COMPRESSION

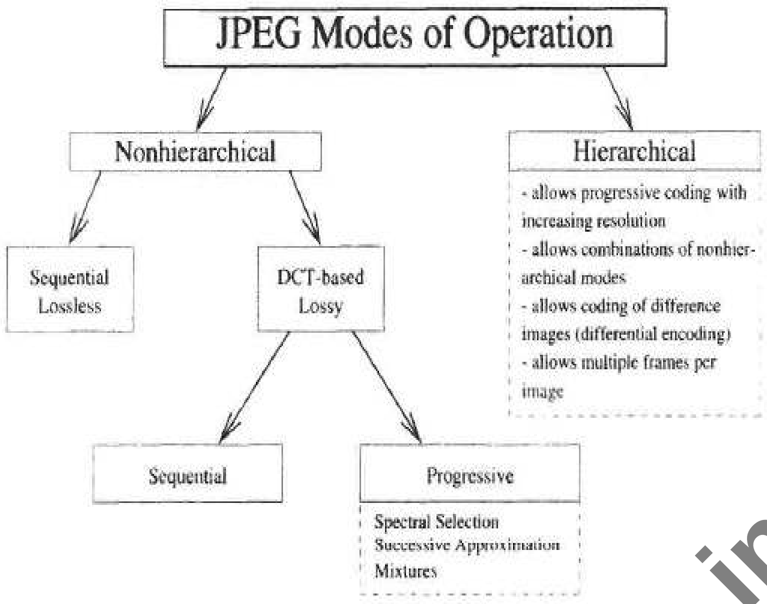
Any compression scheme can be viewed as a sequence of two steps. The first step involves removal of redundancy based on implicit assumptions about the structure in the data, and the second step is the assignment of binary code words to the information deemed no redundant The lossy compression portion of JPEG and JPEG2000 relies on a blockwise spectral decomposition of the image. The current JPEG standard uses Discrete Cosine Transform (DCT) and the JPEG2000 standard uses wavelets.

JPEG

JPEG refers to a wide variety of possible image compression approaches that have been collected into a single standard. In this section we attempt to describe JPEG in a somewhat general but comprehensible way. A very complete description of the JPEG standard has been presented in Pennabaker and Mitchell [1993]. As mentioned in the preceding, the JPEG standard has both lossless and lossy components. In addition, the entropy coding employed by JPEG can be either Huffman coding or binary arithmetic coding. Figures 5.1 and 5.2 present very general image compression models that help describe the JPEG standard. In Figure 5.1 the compression process is broken into two basic functions: modeling the image data and entropy coding the description provided by a particular model. As the figure indicates, the modeling and entropy coding are separate. Hence whether Huffman or arithmetic entropy codes are used is irrelevant to the modeling. Any standard application-specific or image-specific coding tables can be used for entropy coding. The reverse process is illustrated in Figure 5.2. The modes of operation for JPEG are depicted in Figure 5.3. Two basic functional modes exist: nonhierarchical and hierarchical. Within the nonhierarchical modes are the sequential lossless and the lossy DCT-based sequential and progressive modes. The sequential modes progress through an image segment in a strict left-to-right, top-to-bottom pass. The progressive modes allow several refinements through an image segment, with increasing quality after each JPEG modes of operation. coding with increasing resolution, coding of difference images, and multiple frames per image (the nonhierarchical modes allow only a single frame per image).






JPEG Modes of Operation
Nonhierarchical
Hierarchical
- allows progressive coding with increasing resolution
- allows combinations of nonhierarchical modes
- allows coding of difference images (differential encoding)
- allows multiple frames per image
Sequential Lossless
DCT-based Lossy
Sequential
Progressive
Spectral Selection
Successive Approximation
Mixtures

DCT-Based Image Compression

The basis for JPEG's lossy compression is the two-dimensional OCT. An image is broken into 8 x 8 blocks on which the transform is computed. The transform allows different two-dimensional frequency components to be coded separately. Image compression is obtained through quantization of these DCT coefficients to a relatively small set of finite values. These values (or some representation of them) are entropy coded and stored as a compressed version of the image.

The DCT-based compression is effective because the human visual system is relatively less sensitive to higher (spatial) frequency image components and the quantization is uneven for the various DCT frequencies. In addition, many images have very few significant high-frequency components, so many of the quantized DCT coefficients are zero and can be efficiently coded. Figure 5.4 illustrates a typical JPEG quantization table. The DC coefficient (top left) is encoded differentially from block to block and the quantization step size is given for the *difference* between the current block's DC coefficient and the previous block's quantized DC coefficient. The other 63 coefficients in the transform are referred to as the AC coefficients. These are quantized directly by the respective step size values in the table. All quantization is done assuming a midtread quantizer to allow for zero values in the quantizer output. The scaling in the JPEG quantization tables assumes that a 1 is equal to the least significant bit in a $P+3$ bit binary word when the original pixels have P bits of precision. This results from the DCT scaling providing a multiplicative factor of 8. JPEG users are free to define and use their own quantization tables (and hence transmit or store these with images) or use the tables provided by JPEG. Note that in the lossy DCT-based JPEG modes the allowable values of P are 8 or 12, with a value of 128 or 2048 being subtracted from the raw image data for 8- or 12-bit precision, respectively, prior to processing. Adaptive quantization has been added as a JPEG extension. The adaptive quantization allows either selection of a new quantization table or a modification (e.g., scaling) of an existing quantization table.

Additional compression of the AC coefficients is obtained because these coefficients are scanned in a zigzag fashion from lowest frequency to highest frequency, as illustrated in Figure 5.5. The zigzag scan in combination with the quantization table and midtread quantizer design results in long runs of zero coefficients for many image blocks. This substantially enhances the compression achieved with the subsequent entropy coding. In addition, this ordering is exploited by some of JPEG's progressive transmission schemes.

UNIT - 4

AUDIO AND VIDEO COMPRESSION

Introduction, audio compression, DPCM, ADPCM, APC, LPC, video compression, video compression principles, H.261, H.263, MPEG, MPEG-1, MPEG-2, and MPEG-4

7 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & Proceedings:

1. J. Choi and D. Park, A stable feedback control of the buffer state using the controlled Lagrange multiplier method, IEEE Trans. Image Processing, 3, 546–558 (1994).
2. Y. L. Lin and A. Ortega, Bit rate control using piecewise approximated rate-distortion characteristics, IEEE Trans CSVT, 8, 446–459 (1998).
3. W. Ding, Rate control of MPEG-video coding and recording by rate quantization modeling, IEEE Trans CSVT, 6, 12–20 (1966).
4. B. Tao, H. A. Peterson, and B. W. Dickinson, A rate-quantization model for MPEG encoders, Proc. IEEE ICIP, 1, 338–341 1997.
5. K. H. Yang, A. Jacquin, and N. S. Jayant, A normalized rate distortion model for H.263- compatible codecs and its application to quantizer selection, Proc. IEEE ICIP, 1, 41–44
(1997).

6. A. Velio, H. F. Sun, and Y. Wang, MPEG-4 rate control for multiple video objects, IEEE
Trans. CSVT, 9, 186–199 (1999).
7. Y. Ribos-Corbero and S. M. Lei, JTC1/SC29/WG11 MPEG96/M1820, Contribution to Rate Control Q2 Experiment: A Quantization Control Tool for Achieving Target Bitrate Accurately, Sevilla, Spain (1997).

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 dB ⁘10 log10(*A* /*B*).

If we define the lower limit of audibility (a pressure of about 0.0003 dyne/cm2) 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 ▪ *T* 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 2*f*. Sampling more often is of no value since the higher frequencies that such sampling could detect are not present.

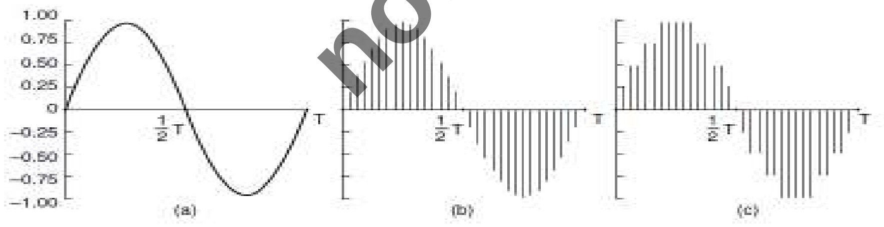


Figure 7-1. (a) A sine wave. (b) Sampling the sine wave. (c) Quantizing the samples to 4 bits.

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.

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 frequencycomponents. 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 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.

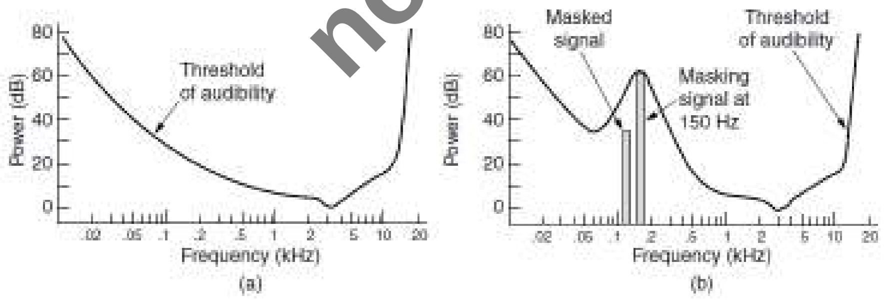


Figure 7-2. (a) The threshold of audibility as a function of frequency. (b) The masking effect.

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).

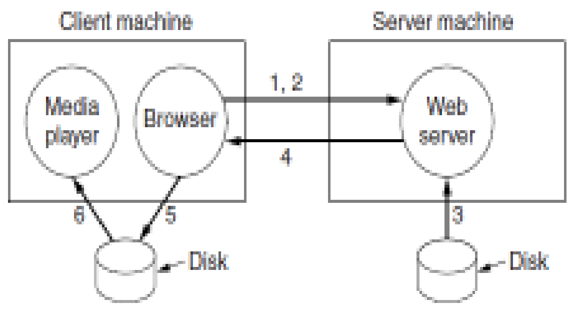
First, the output bit rate is chosen. MP3 can compress a stereo rock ’n roll CD down to 96 kbps with little perceptible loss in quality, even for rock ’n roll fans with no hearing loss. For a piano concert, at least 128 kbps are needed. These differ because the signal-to-noise ratio for rock ’n roll 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).

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.



1. Establish TCP connection
2. Send HTTP GET request
3. Server gets file from disk
4. File sent back
5. Browser writes file to disk
6. Media player fetches file block by block and plays it

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

When the browser gets the 1-line file, it writes it to disk on a scratch file, starts the media player as a helper, and hands it the name of the scratch file, as usual. The media player then reads the file and sees that it contains a URL. Then it contacts *joes-audio-server* and asks for the song. Note that the browser is not in the loop any more.

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.

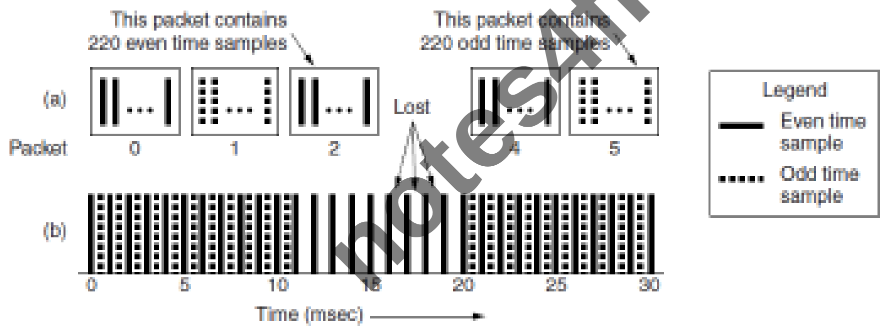

This packet contains 220 even time samples
This packet contains 220 odd time samples
(a)
Packet
0
1
2
Lost
4
5
(b)
0
5
10
15
20
25
30
Time (msec)
Legend
Even time sample
Odd time sample

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

The media player's third job is decompressing the music. Although this task is computationally intensive, it is fairly straightforward. The fourth job is to eliminate jitter, the bane of all real-time systems. All streaming audio systems start by buffering about 10–15 sec worth of music before starting to play, as shown in Fig. 7-5. Ideally, the server will continue to fill the buffer at the exact rate it is being drained by the media player, but in reality this may not happen, so feedback in the loop may be helpful.

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.

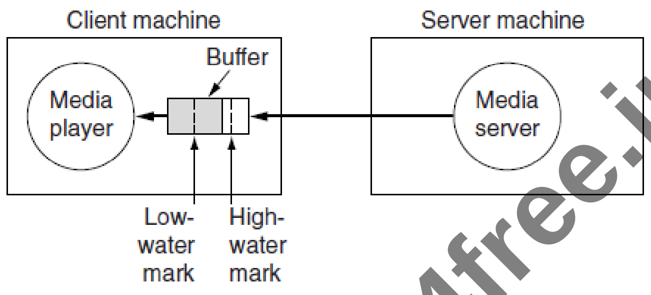


Figure 7-5. The media player buffers input from the media server and plays from the buffer rather than directly from the network.

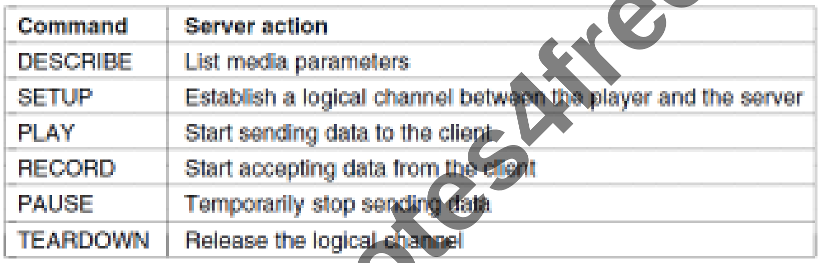
Its goal is to keep the buffer as full as possible. The disadvantage of a pull server is all the unnecessary data requests. The server knows it has sent the whole file, so why have the player keep asking? For this reason, it is rarely used. With a **push server**, the media player sends a *PLAY* request 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).

The solution is for the media player to define a **low-water mark** and a **high-water mark** in the buffer. Basically, the server just pumps out data until the buffer is filled to the high-water mark. Then the media player tells it to pause. Since data will continue to pour in until the server has gotten the pause request, the distance between the high-water mark and the end of the buffer has to be greater than the bandwidth-delay product of the network. After the server has stopped, the buffer will begin to empty. When it hits the low-water mark, the media player tells the media server to start again. The low-water mark has to be positioned so that buffer underrun does not occur.

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.

Differential Encoding Methods

- **Differential pulse code modulation** (DPCM)

Simple prediction (also used in JPEG):

Simple Differential Pulse Code Modulation Example

Actual Data: 9 10 7 6

Predicted Data: 0 9 10 7

$\triangle f(t)$: +9, +1, -3, -1.

MATLAB Complete (with quantisation) DPCM Example

dpcm_demo.m.m: DPCM Complete Example
dpcm.zip.m: DPCM Support FIles

Differential Encoding Methods (Cont.)

- Delta modulation is a special case of DPCM:
  - Same predictor function,
  - Coding error is a single bit or digit that indicates the current sample should be increased or decreased by a step.
  - Not Suitable for rapidly changing signals.
- Adaptive pulse code modulation

Fuller Temporal/Markov model:

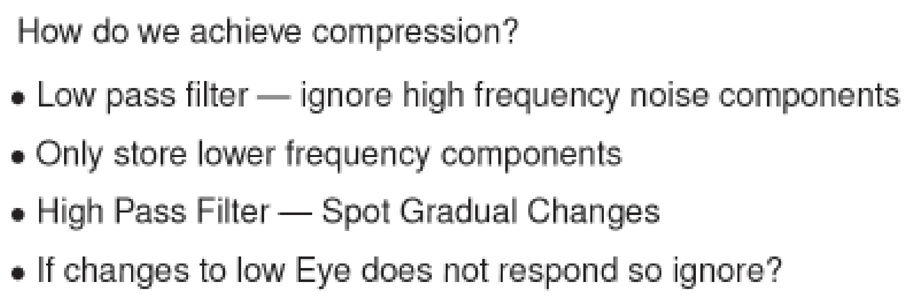
- Data is extracted from a function of a series of previous values
- E.g. Average of last n samples.
- Characteristics of sample better preserved.

Frequency Domain Methods

Another form of Transform Coding

Transformation from one domain —time (*e.g.* 1D audio, video:2D imagery over time) or Spatial (*e.g.* 2D imagery) domain to the frequency domain via

- Discrete Cosine Transform (DCT)— Heart of JPEG and MPEG Video, (alt.) MPEG Audio.
- Fourier Transform (FT) — MPEG Audio


How do we achieve compression?

- Low pass filter — ignore high frequency noise components
- Only store lower frequency components
- High Pass Filter — Spot Gradual Changes
- If changes to low Eye does not respond so ignore?

Video Compression

It should be obvious by now that transmitting uncompressed video is completely out of the question. The only hope is that massive compression is possible. Fortunately, a large body of research over the past few decades has led to many compression techniques and algorithms that make video transmission feasible.

In this section we will study how video compression is accomplished. All compression systems require two algorithms: one for compressing the data at the source, and another for decompressing it at the destination. In the literature, these algorithms are referred to as the **encoding** and **decoding** algorithms, respectively.

We will use this terminology here, too. These algorithms exhibit certain asymmetries that are important to understand. First, for many applications, a multimedia document, say, a movie will only be encoded once (when it is stored on the multimedia server) but will be decoded thousands of times (when it is viewed by customers). This asymmetry means that it is acceptable for the encoding algorithm to be slow and require expensive hardware provided that the decoding algorithm is fast and does not require expensive hardware. After all, the operator of a multimedia server might be quite willing to rent a parallel supercomputer for a few weeks to encode its entire video library, but requiring consumers to rent a supercomputer for 2 hours to view a video is not likely to be a big success. Many practical compression systems go to great lengths to make decoding fast and simple, even at the price of making encoding slow and complicated.

On the other hand, for real-time multimedia, such as video conferencing, slow encoding is unacceptable. Encoding must happen on-the-fly, in real time. Consequently, real-time multimedia uses different algorithms or parameters than storing videos on disk, often with appreciably less compression.

A second asymmetry is that the encode/decode process need not be invertible. That is, when compressing a file, transmitting it, and then decompressing it, the user expects to get the original back, accurate down to the last bit. With multimedia, this requirement does not exist. It is usually acceptable to have the video signal after encoding and then decoding be slightly different from the original.

When the decoded output is not exactly equal to the original input, the system is said to be **lossy**. If the input and output are identical, the system is **lossless**. Lossy systems are important because accepting a small amount of information loss can give a huge payoff in terms of the compression ratio possible.

The JPEG Standard

A video is just a sequence of images (plus sound). If we could find a good algorithm for encoding a single image, this algorithm could be applied to each image in succession to achieve video compression. Good still image compression algorithms exist, so let us start our study of video compression there. The **JPEG** (**Joint Photographic Experts Group**) standard for compressing continuous-tone still pictures (e.g., photographs) was developed by photographic experts working under the joint auspices of ITU, ISO, and IEC, another standards body. It is important for multimedia because, to a first approximation, the multimedia standard for moving pictures, MPEG, is just the JPEG encoding of each frame separately, plus some extra features for interframe compression and motion detection. JPEG is defined in International Standard 10918.

JPEG has four modes and many options. It is more like a shopping list than a single algorithm. For our purposes, though, only the lossy sequential mode is relevant, and that one is illustrated in Fig. 7-15. Furthermore, we will concentrate on the way JPEG is normally used to encode 24-bit RGB video images and will leave out some of the minor details for the sake of simplicity.

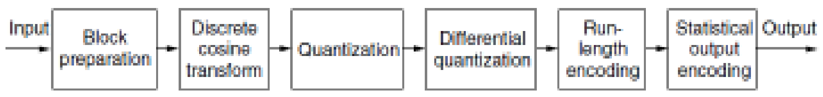

Input
Block preparation
Discrete cosine transform
Quantization
Differential quantization
Run-length encoding
Statistical output encoding
Output

Figure 7-15. The operation of JPEG in lossy sequential mode.

Step 1 of encoding an image with JPEG is block preparation. For the sake of specificity, let us assume that the JPEG input is a 640 × 480 RGB image with 24 bits/pixel, as shown in Fig. 7-16(a). Since using luminance and chrominance gives better compression, we first compute the luminance, *Y*, and the two chrominances, *I* and *Q* (for NTSC), according to the following formulas:

$Y = 0.30R + 0.59G + 0.11B$

$I = 0.60R - 0.28G - 0.32B$

$Q = 0.21R - 0.52G + 0.31B$

For PAL, the chrominances are called *U* and *V* and the coefficients are different, but the idea is the same. SECAM is different from both NTSC and PAL. Separate matrices are constructed for *Y*, *I*, and *Q*, each with elements in the range 0 to 255.

Next, square blocks of four pixels are averaged in the *I* and *Q* matrices to reduce them to 320 × 240. This reduction is lossy, but the eye barely notices it since the eye responds to luminance more than to chrominance. Nevertheless, it compresses the total amount of data by a factor of two. Now 128 is subtracted from each element of all three matrices to put 0 in the middle of the

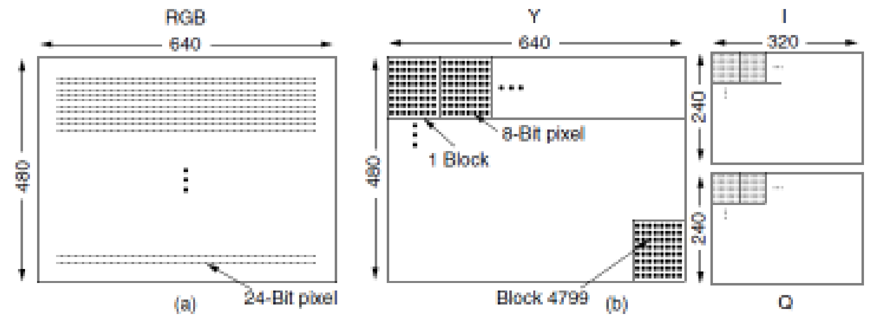


Figure 7-16. (a) RGB input data. (b) After block preparation.

range. Finally, each matrix is divided up into 8 × 8 blocks. The *Y* matrix has 4800 blocks; the other two have 1200 blocks each, as shown in Fig. 7-16(b). Step 2 of JPEG is to apply a **DCT** (**Discrete Cosine Transformation**) to each of the 7200 blocks separately. The output of each DCT is an 8 × 8 matrix of DCT coefficients. DCT element (0, 0) is the average value of the block. The other elements tell how much spectral power is present at each spatial frequency. In theory, a DCT is lossless, but in practice, using floating-point numbers and transcendental functions always introduces some roundoff error that results in a little information loss. Normally, these elements decay rapidly with distance from the origin, (0, 0), as suggested by Fig. 7-17.

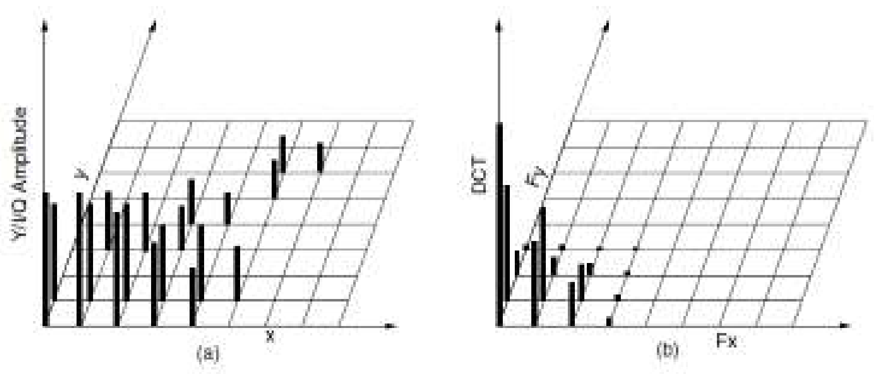
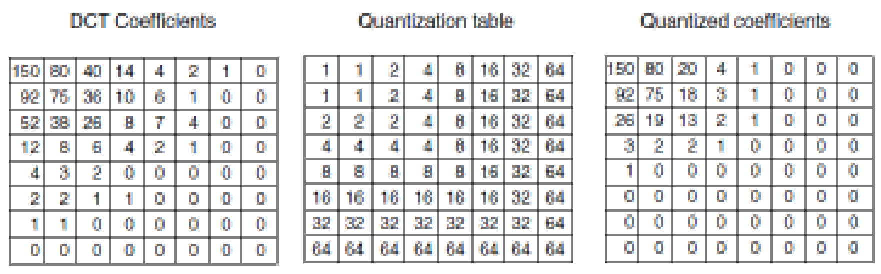


Figure 7-17. (a) One block of the Y matrix. (b) The DCT coefficients.

Once the DCT is complete, JPEG moves on to step 3, called **quantization**, in which the less important DCT coefficients are wiped out. This (lossy) transformation is done by dividing each of the coefficients in the 8 ▪ 8 DCT matrix by a weight taken from a table. If all the weights are 1, the transformation does nothing.

However, if the weights increase sharply from the origin, higher spatial frequenciesare dropped quickly. An example of this step is given in Fig. 7-18. Here we see the initial DCT matrix, the quantization table, and the result obtained by dividing each DCT element by the corresponding quantization table element. The values in the quantization table are not part of the JPEG standard. Each application must supply its own, allowing it to control the loss-compression trade-off.



DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quanitized coefficients

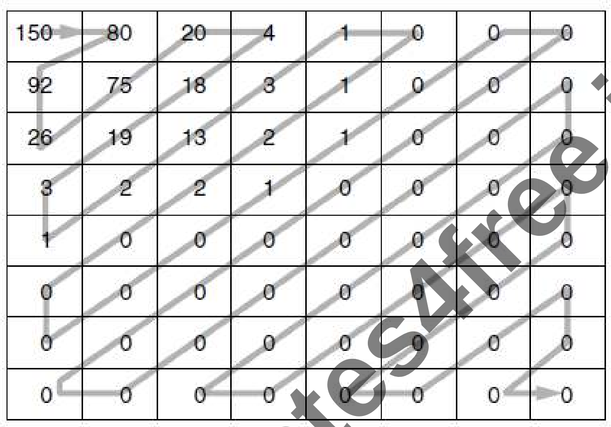
150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 7-18. Computation of the quantized DCT coefficients.

Step 4 reduces the (0, 0) value of each block (the one in the upper-left corner) by replacing it with the amount it differs from the corresponding element in the previous block. Since these elements are the averages of their respective blocks, they should change slowly, so taking the differential values should reduce most of them to small values. No differentials are computed from the other values. The (0, 0) values are referred to as the DC components; the other values are the AC components.

Step 5 linearizes the 64 elements and applies run-length encoding to the list. Scanning the block from left to right and then top to bottom will not concentrate the zeros together, so a zigzag scanning pattern is used, as shown in Fig. 7-19. In this example, the zig zag pattern produces 38 consecutive 0s at the end of the matrix. This string can be reduced to a single count saying there are 38 zeros, a technique known as **run-length encoding**. Now we have a list of numbers that represent the image (in transform space).

Step 6 Huffman-encodes the numbers for storage or transmission, assigning common numbers shorter codes that uncommon ones. JPEG may seem complicated, but that is because it *is* complicated. Still, since it often produces a 20:1 compression or better, it is widely used. Decoding a JPEG image requires running the algorithm backward. JPEG is roughly symmetric: decoding takes as long as encoding. This property is not true of all compression algorithms, as we shall now see.



150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 7-19. The order in which the quantized values are transmitted.

Compression:

Video Compression (MPEG and others)

We need to compress video (more so than audio/images) in practice since:

1. Uncompressed video (and audio) data are huge. In HDTV, the bit rate easily exceeds 1 Gbps. —big problems for storage and

network communications. **E.g.** HDTV: 1920 x 1080 at 30 frames per second, 8 bits per YCrCb (PAL) channel = 1.5 Gbps.

2. Lossy methods have to employed since the **compression ratio** of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression.

Not the complete picture studied here

Much more to MPEG — Plenty of other tricks employed. We only concentrate on some basic principles of video compression:

_ Earlier H.261 and MPEG 1 and 2 standards.

Compression Standards Committees

Image, Video and Audio Compression standards have been specifies and released by two main groups since 1985:

ISO - International Standards Organisation: JPEG, MPEG.

ITU - International Telecommunications Union: H.261 — 264.

Compression Standards

Whilst in many cases one of the groups have specified separate standards there is some crossover between the groups.

For example:

_ JPEG issued by ISO in 1989 (but adopted by ITU as ITU T.81)

_ MPEG 1 released by ISO in 1991,

_ H.261 released by ITU in 1993 (based on CCITT 1990 draft).

CCITT stands for **Comit´e Consultatif International T´el ´ephonique et**

T´el´egraphique whose parent company is ITU.

_ H.262 is alternatively better known as MPEG-2 released in 1994.

_ H.263 released in 1996 extended as H.263+, H.263++.

_ MPEG 4 release in 1998.

_ H.264 releases in 2002 for DVD quality and is now part of MPEG 4 .

How to compress video?

Basic Idea of Video Compression:

Motion Estimation/Compensation

_ Spatial Redundancy Removal – Intraframe coding (JPEG)

NOT ENOUGH BY ITSELF?

_ Temporal — Greater compression by noting the temporal coherence/incoherence over frames. Essentially we note the difference between frames.

_ Spatial and Temporal Redundancy Removal – Intraframe and Interframe coding (H.261, MPEG).

Simple Motion Estimation/Compensation

Example: Things are much more complex in practice of course. Which Format to represent the compressed data?

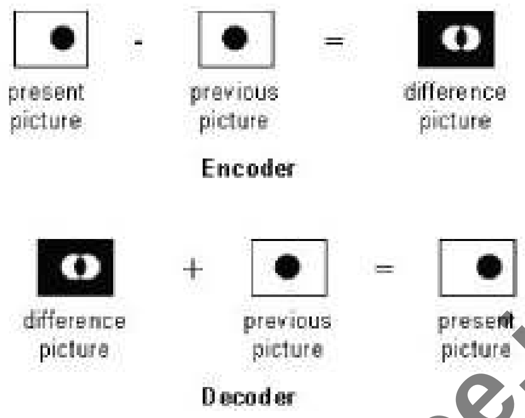
_ Simply based on Differential Pulse Code Modulation (DPCM).

Simple Motion Example (Cont.)

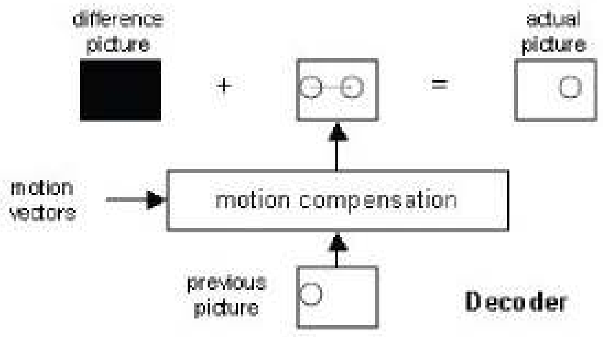
Consider a simple image (block) of a moving circle.

Lets just consider the difference between 2 frames.

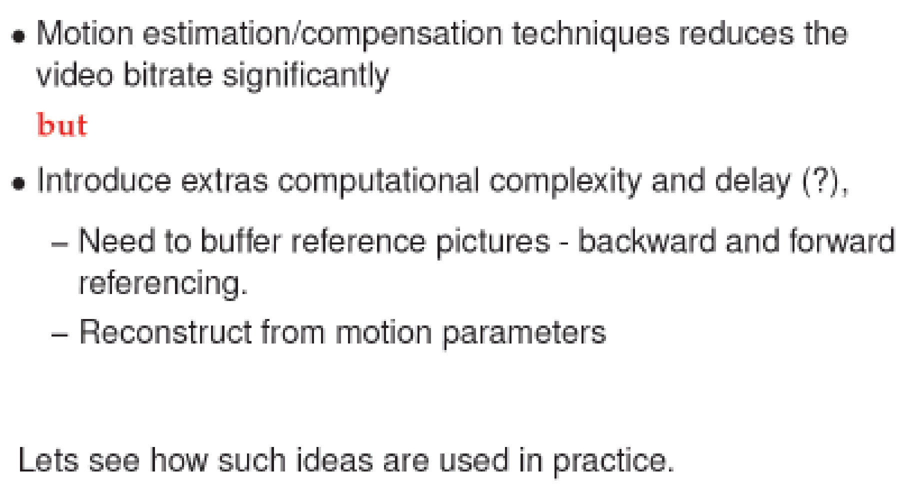
It simple to encode/decode:



Decoding Motion of blocks



Why is this a better method than just frame differencing?



- Motion estimation/compensation techniques reduces the video bitrate significantly

  but

- Introduce extras computational complexity and delay (?),
  - Need to buffer reference pictures - backward and forward referencing.
  - Reconstruct from motion parameters

Lets see how such ideas are used in practice.

H.261 Compression

The basic approach to H. 261 Compression is summarised as follows:

H. 261 Compression has been specifically designed for video telecommunication applications:

_ Developed by CCITT in 1988-1990

_ Meant for videoconferencing, videotelephone applications over ISDN telephone lines.

_ Baseline ISDN is 64 kbits/sec, and integral multiples (px64)

Overview of H.261

_ Frame types are CCIR 601 CIF (352x288) and QCIF (176x144) images with 4:2:0 subsampling.

_ Two frame types:

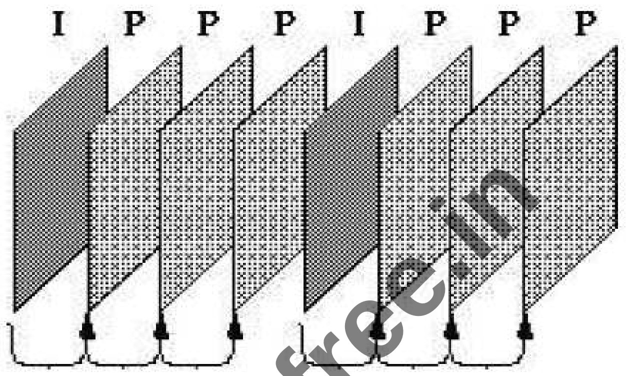
Intraframes (I-frames**)** and **Interframes (**P-frames**)**

_ I-frames use basically JPEG—**but** YUV (YCrCb) and **larger** DCT windows, **different** quantisation

_ I-frame provide us with a refresh accessing point — **Key Frames**

_ P-frames use **pseudo-differences** from previous frame (predicted), so frames depend on each other.

H.261 Group of Pictures



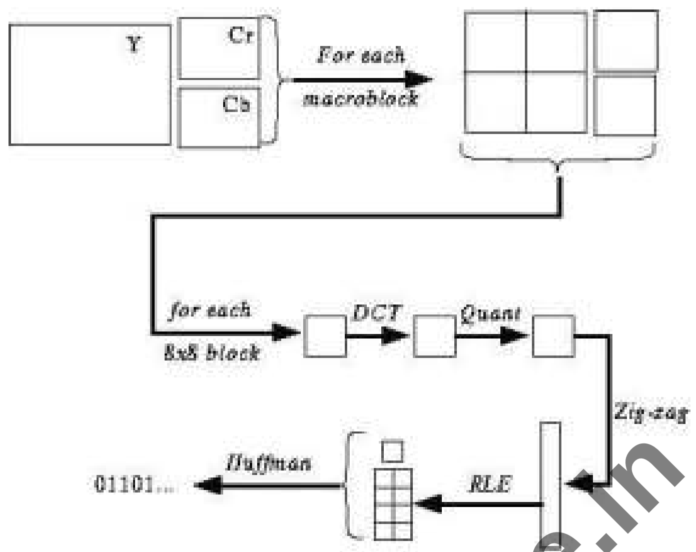
- We typically have a group of pictures — one *I-frame* followed by several *P-frames* — a **group of pictures**
- Number of *P-frame* defined by a **quality measure** — more = less quality — defined in MPEG standard.

Intra Frame Coding

- Various lossless and lossy compression techniques use — like JPEG.
- Compression contained only within the current frame
- Simpler coding – Not enough by itself for high compression.
- Cant rely on intra frame coding alone not enough compression
  - Motion JPEG (MJPEG) standard does exist — not commonly used.
  - So introduce idea of inter frame difference coding
- However, cant rely on inter frame differences across a large number of frames
  - So when Errors get too large: Start a new I-Frame

Intra Frame Coding (Cont.)

Intraframe coding is very similar to JPEG:

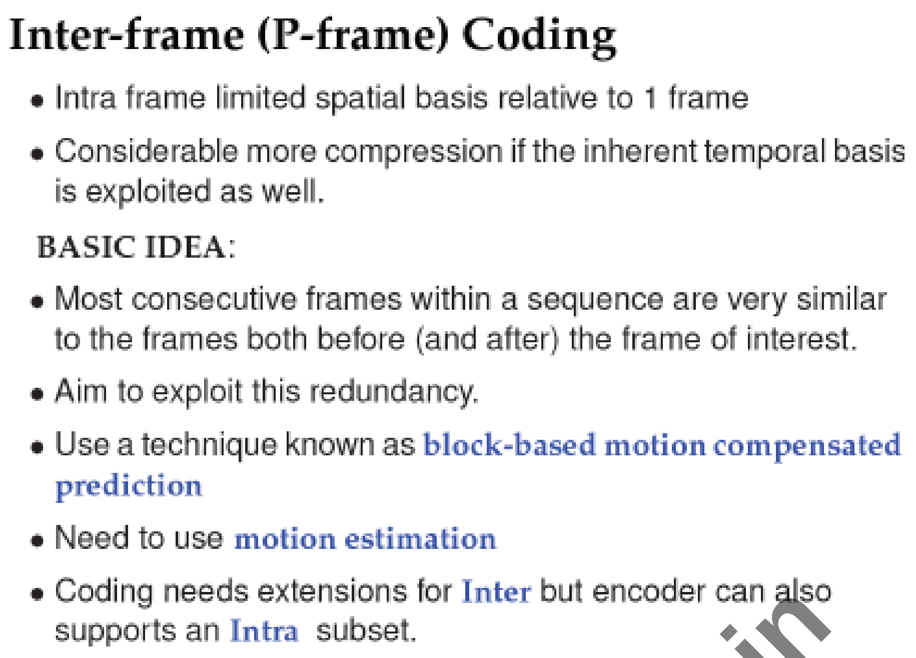


What are the differences between this and JPEG?

Intra Frame Coding (Cont.)

This is a basic Intra Frame Coding Scheme is as follows:

- Macroblocks are typically 16x16 pixel areas on Y plane of original image.
- A **macroblock** usually consists of 4 Y blocks, 1 Cr block, and 1 Cb block. (4:2:0 chroma subsampling)
  - Eye most sensitive luminance, less sensitive chrominance.
    - We operate on a **more effective** color space: YUV (YCbCr) colour which we studied earlier.
  - Typical to use 4:2:0 macroblocks: one quarter of the chrominance information used.
- Quantization is by constant value for all DCT coefficients. **I.e., no quantization table as in JPEG**.



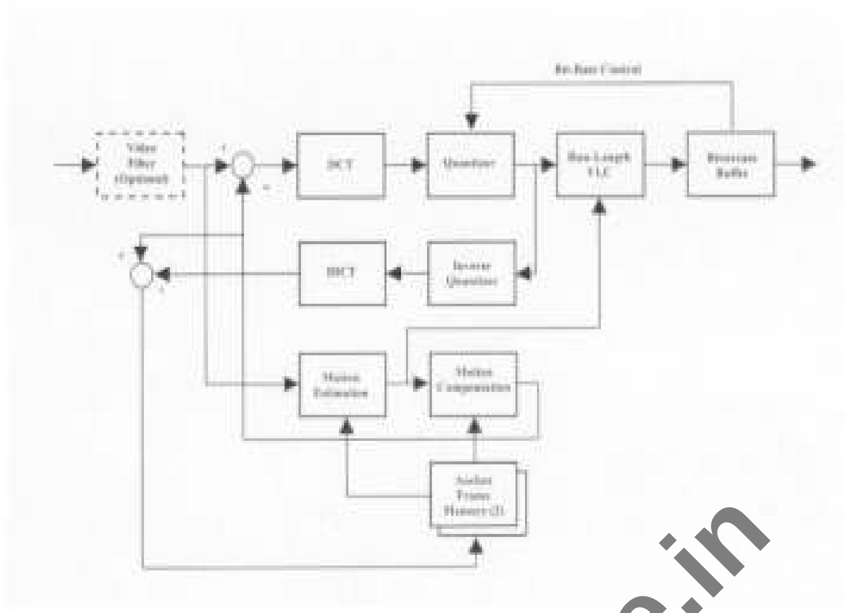
Inter-frame (P-frame) Coding

- Intra frame limited spatial basis relative to 1 frame
- Considerable more compression if the inherent temporal basis is exploited as well.

BASIC IDEA:

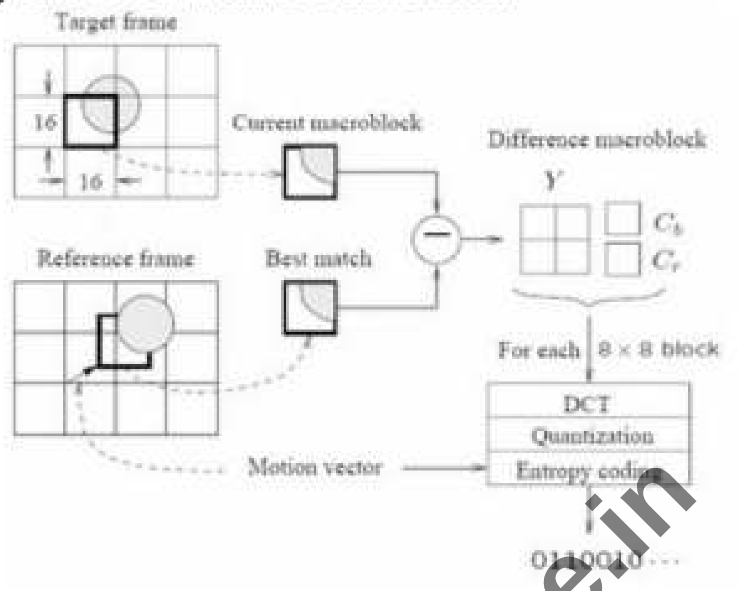
- Most consecutive frames within a sequence are very similar to the frames both before (and after) the frame of interest.
- Aim to exploit this redundancy.
- Use a technique known as **block-based motion compensated prediction**
- Need to use **motion estimation**
- Coding needs extensions for **Inter** but encoder can also supports an **Intra** subset.

Inter-frame (P-frame) Coding (Cont.)



Inter-frame (P-frame) Coding (Cont.)

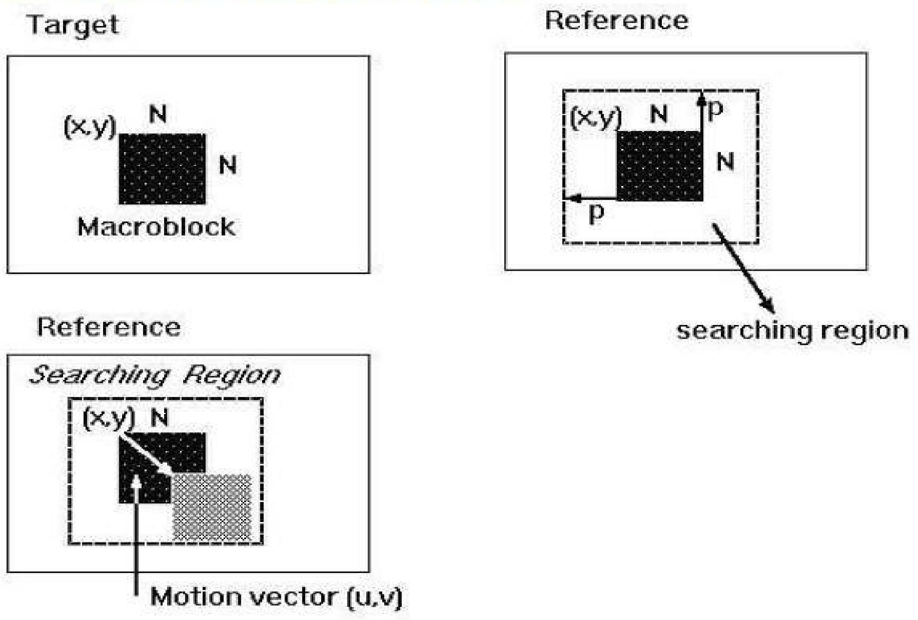
P-coding can be summarised as follows:



Motion Vector Search

So we know how to encode a P-block.

How do we find the motion vector?



Motion Estimation

- The temporal prediction technique used in MPEG video is based on motion estimation.

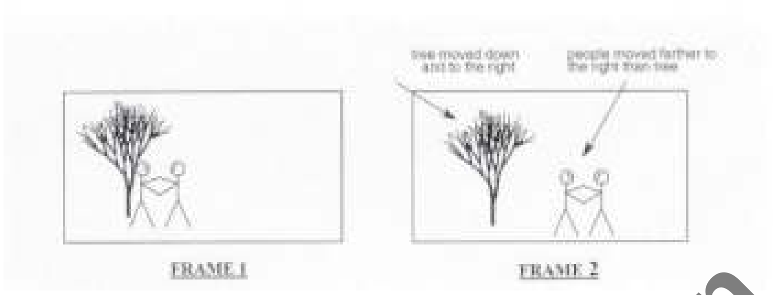
The basic premise:

- Consecutive video frames will be similar except for changes induced by objects moving within the frames.
- Trivial case of zero motion between frames — no other differences except noise, etc.),
- Easy for the encoder to predict the current frame as a duplicate of the prediction frame.
- When there is motion in the images, the situation is not as simple.

Example of a frame with 2 stick figures and a tree

The problem for motion estimation to solve is :

- How to adequately represent the changes, or differences, between these two video frames.

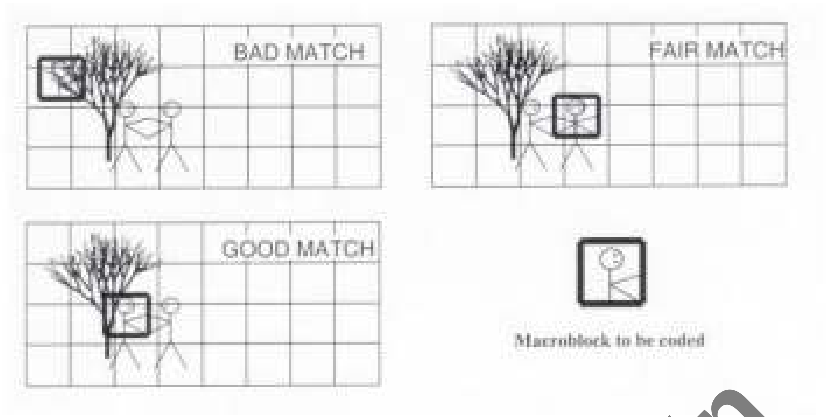

tree moved down and to the right
people moved farther to the right than tree
FRAME 1
FRAME 2

Solution:

A comprehensive 2-dimensional spatial search is performed for each luminance macroblock.

- Motion estimation is not applied directly to chrominance in MPEG
- MPEG does not define how this search should be performed.
- A detail that the system designer can choose to implement in one of many possible ways.
- Well known that a full, exhaustive search over a wide 2-D area yields the best matching results in most cases, but at extreme computational cost to the encoder.
- Motion estimation usually is the most computationally expensive portion of the video encode

Motion Estimation Example

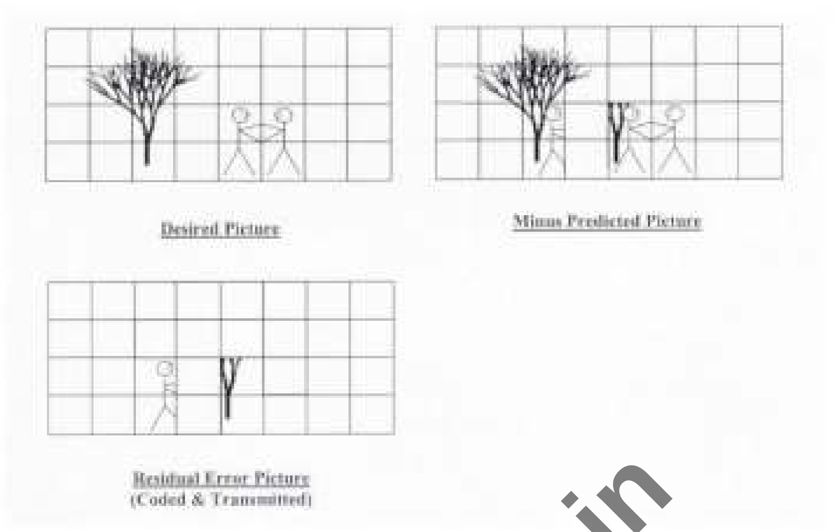


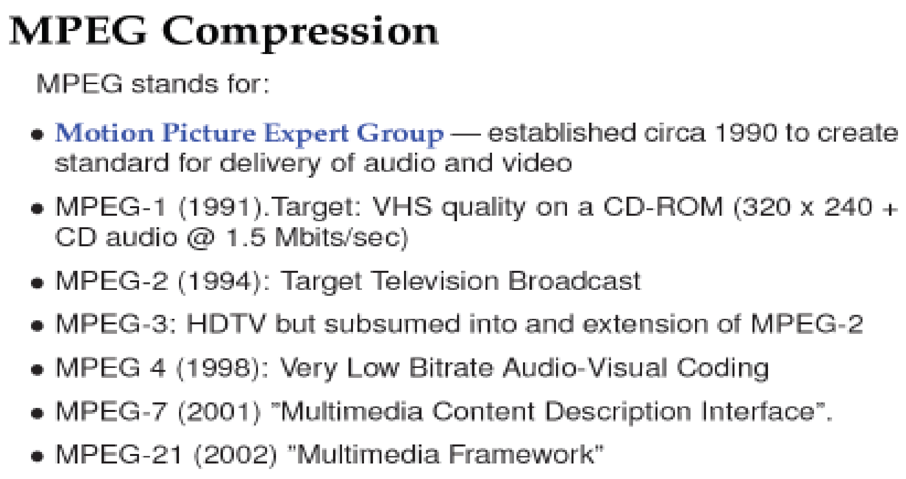
Motion Vectors, Matching Blocks

Previous figure shows an example of a particular macroblock from Frame 2 of earlier example, relative to various macroblocks of Frame 1:

- The top frame has a bad match with the macroblock to be coded.
- The middle frame has a fair match, as there is some commonality between the 2 macroblocks.
- The bottom frame has the best match, with only a slight error between the 2 macroblocks.
- Because a relatively good match has been found, the encoder assigns motion vectors to that macroblock,

Final Motion Estimation Prediction

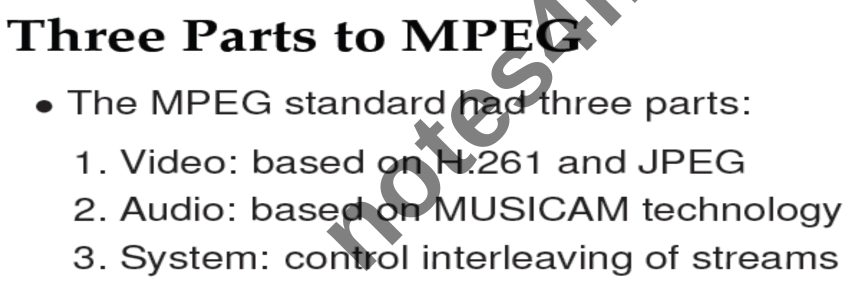




MPEG Compression

MPEG stands for:

- Motion Picture Expert Group — established circa 1990 to create standard for delivery of audio and video
- MPEG-1 (1991).Target: VHS quality on a CD-ROM (320 x 240 + CD audio @ 1.5 Mbits/sec)
- MPEG-2 (1994): Target Television Broadcast
- MPEG-3: HDTV but subsumed into and extension of MPEG-2
- MPEG 4 (1998): Very Low Bitrate Audio-Visual Coding
- MPEG-7 (2001) "Multimedia Content Description Interface".
- MPEG-21 (2002) "Multimedia Framework"



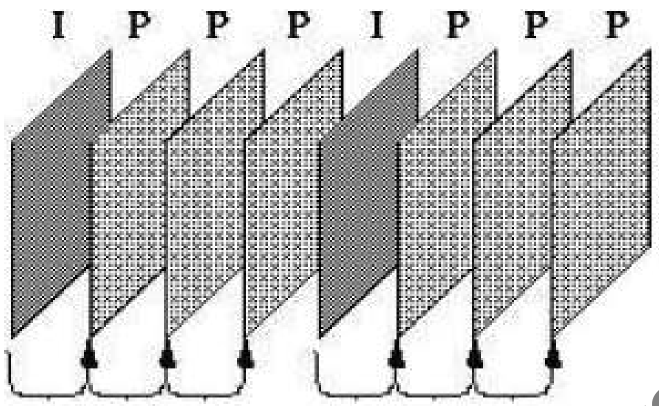
Three Parts to MPEG

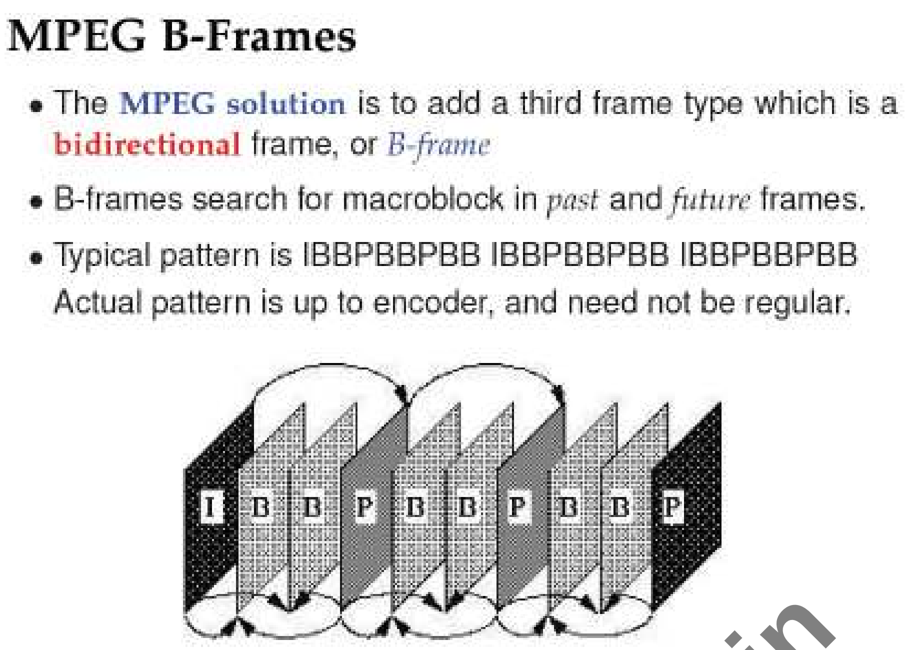
- The MPEG standard had three parts:
  1. Video: based on H.261 and JPEG
  2. Audio: based on MUSICAM technology
  3. System: control interleaving of streams

MPEG Video

MPEG compression is essentially a attempts to over come some shortcomings of H.261 and JPEG:

- Recall H.261 dependencies:





MPEG B-Frames

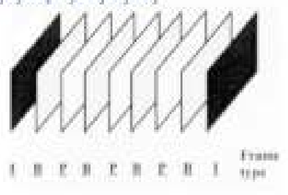
- The **MPEG solution** is to add a third frame type which is a **bidirectional** frame, or *B-frame*
- B-frames search for macroblock in *past* and *future* frames.
- Typical pattern is IBBPBBPBB IBBPBBPBB IBBPBBPBB
  Actual pattern is up to encoder, and need not be regular.

I
B
B
P
B
B
P
B
B
P

Example I, P, and B frames

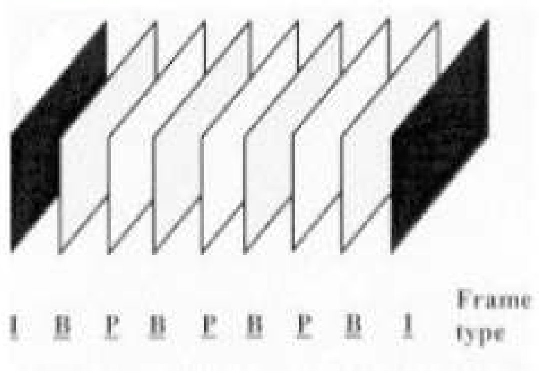
Consider a group of pictures that lasts for 6 frames:

- Given: I,B,P,B,P,B,I,B,P,B,P,B,



- I frames are coded spatially only (as before in H.261).
- P frames are forward predicted based on previous I and P frames (as before in H.261).
- B frames are coded based on a forward prediction from a **previous** I or P frame, **as well** as a **backward prediction** from a **succeeding** I or P frame.

- 1st B frame is predicted from the 1st I frame and 1st P frame.
- 2nd B frame is predicted from the 1st and 2nd P frames.
- 3rd B frame is predicted from the 2nd and 3rd P frames.
- 4th B frame is predicted from the 3rd P frame and the 1st I frame of the next group of pictures.

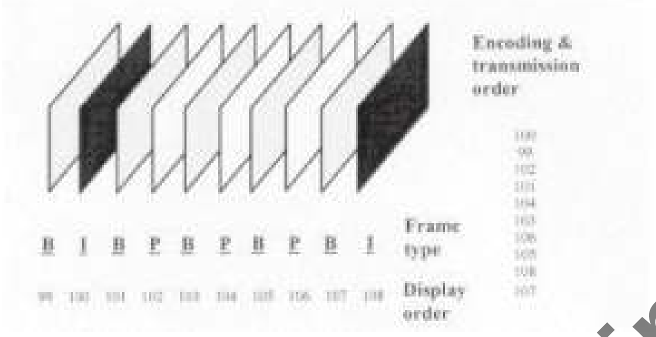


Backward Prediction Implications

Note: Backward prediction requires that the **future frames** that are to be used for **backward prediction** be

- Encoded and Transmitted **first**, *I.e.* **out of order**.

This process is summarized:



Also NOTE:

- No defined limit to the number of consecutive B frames that may be used in a group of pictures,
- Optimal number is application dependent.
- Most broadcast quality applications however, have tended to use 2 consecutive B frames (I,B,B,P,B,B,P,) as the ideal trade-off between compression efficiency and video quality.
- MPEG suggests some standard groupings.

Advantage of the usage of B frames

- Coding efficiency.
- Most B frames use less bits.
- Quality can also be improved in the case of moving objects that reveal hidden areas within a video sequence.
- Better Error propagation: B frames are not used to predict future frames, errors generated will not be propagated further within the sequence.

Disadvantage:

- Frame reconstruction memory buffers within the encoder and decoder must be doubled in size to accommodate the 2 anchor frames.

MPEG-2, MPEG-3, and MPEG-4

- MPEG-2 target applications

Level	size (Mbits)	Pixels/sec	bit-rate	Application
Low	352 x 240	3 M	4	VHS tape equiv.
Main	720 x 480	10 M	15	studio TV
High 1440	1440 x 1152	47 M	60	consumer HDTV
High	1920 x 1080	63 M	80	film production

- MPEG-2 differences from MPEG-1
  1. Search on fields, not just frames.
  2. 4:2:2 and 4:4:4 macroblocks
  3. Frame sizes as large as 16383 x 16383
  4. Scalable modes: Temporal, Progressive,...
  5. Non-linear macroblock quantization factor
  6. A bunch of minor fixes

- MPEG-3: Originally for HDTV (1920 x 1080), got folded into MPEG-2
- MPEG-4: very low bit-rate communication (4.8 to 64 kb/sec). Video processing

8.2 MPEG-4

Scanning the applications of MPEG-1 and MPEG-2 listed in Section 8.1, it is clear that MPEG standards are employed in communications, computing, and entertainment, and it is expected that these three areas will continue to converge in the future. Indeed, it is this expected convergence that is driving many international standards setting bodies. Thus, a goal of MPEG-4 was to develop functionalities to support applications from this convergence. An excellent starting point for our discussions is a comparison of the general reference models for the several MPEG standards. The general reference models for MPEG-1 and MPEG-2 are shown in Figures 8.1 and 8.2, respectively. While these figures appear very simple, it is important to note that the goals of MPEG-1 and MPEG-2 were not considered trivial at the time. Indeed, the target bit rates and quality sought by MPEG-1 and MPEG-2 were considered ambitious, and there was a real demand for the functionality implied by the "Interaction" box in Figure 8.2. Recognizing this demand and realizing the limitations of the MPEG-2 interactive capabilities, the Multimedia and Hypermedia Experts Group (MHEG) created a standard called MHEG-5, which supports extended functionalities that can work with MPEG-2 and other compression methods.

MHEG-5 is not a compression standard, but it provides the capability of composing a scene with text strings, still images, and graphic animations in addition to audio-visual streams. These parts can be provided by the author, and there is some capability for the user to modify the evolution of the scene, through text input and selection menus. The reference model for MHEG-5 is shown in Figure 8.3 and can be contrasted with the MPEG-1 and MPEG-2 compression-dominated standards.

MPEG-4 takes these ideas a step further by allowing more interaction by the user, and also includes some new video compression methods that are more object based; it also offers extended capabilities and choices for audio and voice encoding. Furthermore, MPEG-4 allows audio and video information that may be natural or synthetic, or even a combination of the two. Figure 8.4 specifies the general reference model for MPEG-4, where it is clearly quite a different standard than, say MPEG-1 and -2, and where it is obviously an extension of the MHEG-5 enabled MPEG structure.

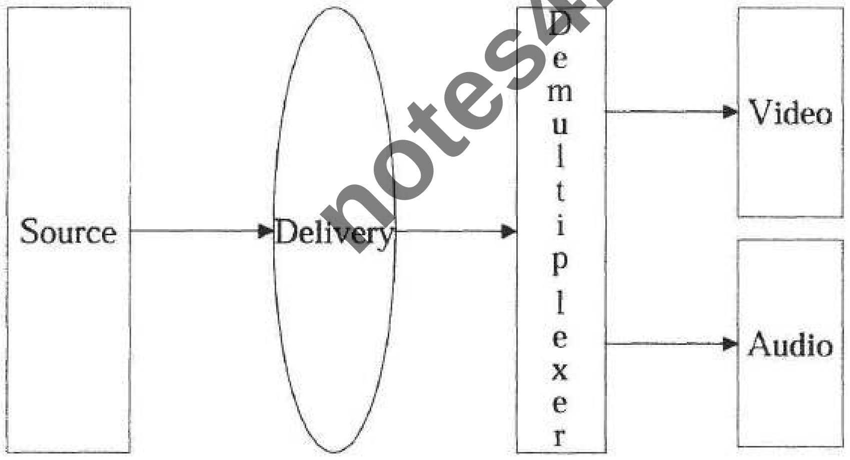


FIGURE 8.1
General reference model for MPEG-1.

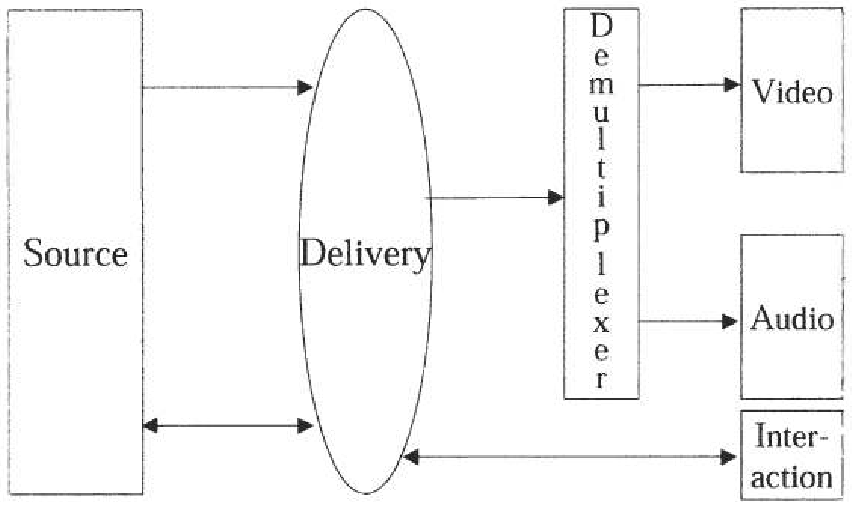


FIGURE 8.2
General reference model for MPEG-2.

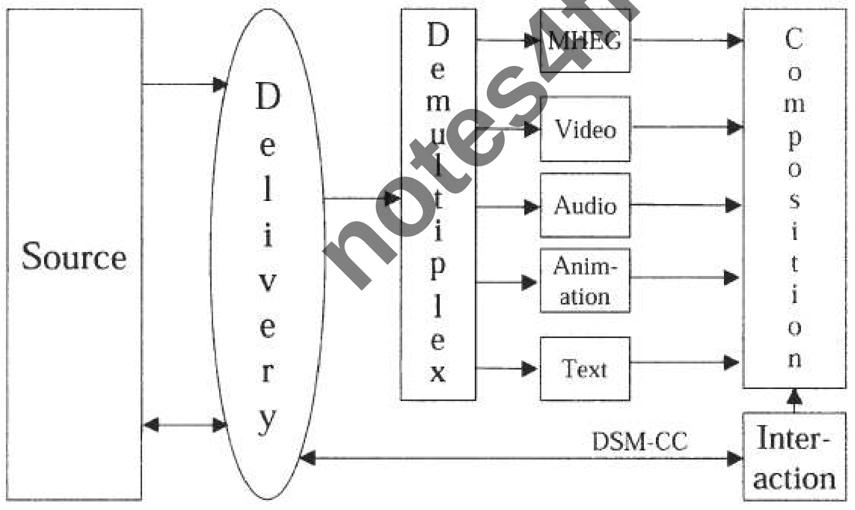


FIGURE 8.3
MHEG-5 enabled MPEG-2 reference model.

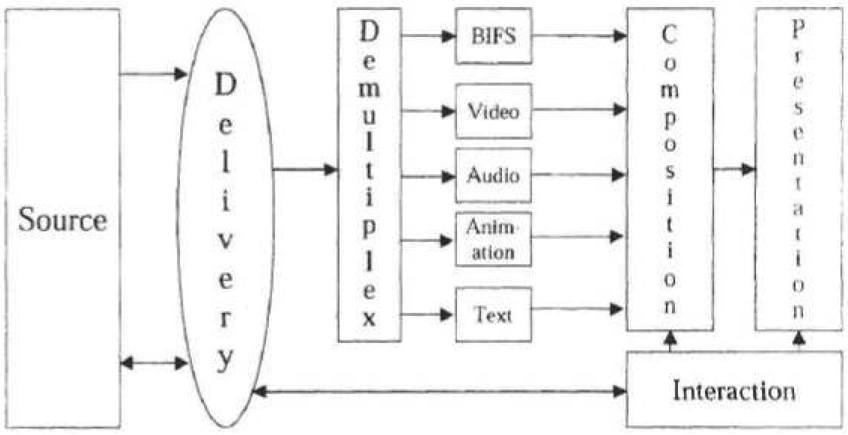


FIGURE 8.4
General reference model for MPEG-4.

8.2.1 MPEG-4 Systems Model

A major difference between MPEG-4 and the MPEG-1 and -2 standards is that MPEG-4 breal
a scene down into components called audio-visual objects, codes these objects, and then recoi
structs the scene from these objects. Figure 8.5 is a representation of the MPEG-4 encoding ar
decoding process [3]. A number of key ideas are represented by this figure. The Binary Form
for Scenes (BIFS) data stream specifies the scene composition, such as the spatial and tempor

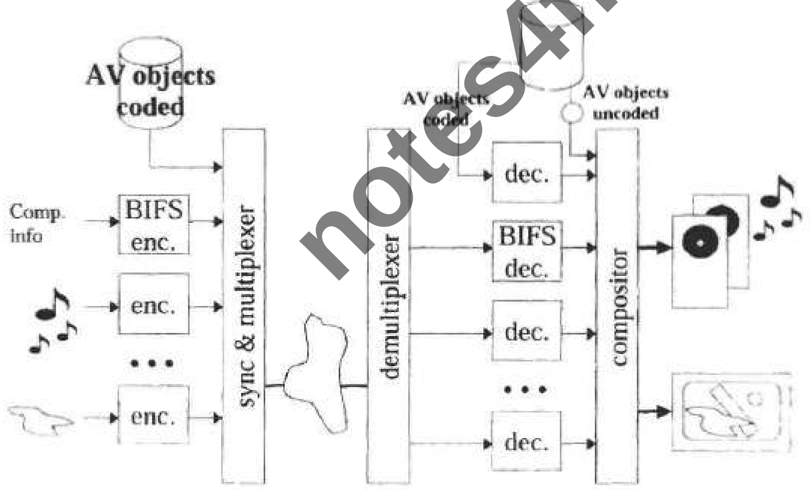


FIGURE 8.5
The MPEG-4 scene decomposition/object-based architecture.

locations of objects in scenes. There are different types of objects to be coded, including natural images, natural audio, voice, synthetic audio and video, textures, and physical objects, and thus there are several encoders shown in the figure. Thus, the scene is decomposed, the objects are compressed separately, and this information is multiplexed with the BIFS composition information for transmission to the decoder. At the decoder, the BIFS information is decoded, the compressed versions of the scene components are decoded, and the scene is recomposed.

There are a number of advantages to utilizing a scene decomposition approach in conjunction with separate coding of the components [4]. The most obvious advantage is that one compression method does not have to compress a complicated scene that includes people, arbitrarily shaped objects, and (maybe) text, which often produces visual artifacts. Once decomposition is achieved, each component can be compressed with an encoder that is much better matched to the specific source to be encoded. Another advantage is that the data stream can be scaled based upon content. That is, the data stream can be modified by removing or adapting content, depending upon bandwidth or complexity requirements. This is really a new idea, and can be very powerful. For example, if there is a need to lower the transmitted bit rate, rather than discard bits that affect the quality of the entire scene, it may be possible to discard an unimportant object in the scene that provides a sufficient bit-rate reduction without reducing the quality of the delivered scene as determined by the particular application. Another advantage of using scene decomposition and object-based compression is that the user can be allowed to access various objects in the scene and change the scene content.

The hierarchical description of a scene is an essential element of MPEG-4 functionality. Figure 8.6 shows a possible breakdown of a video scene into several possible layers. The video session consists of a Visual Object Sequence (VS), as shown at the top of the figure. Within the scene there will be Video Objects (VO), which can then be encoded in the Video Object Layer (VOL). The Video Object Plane (VOP) consists of time samples of a video object. The layer above the VOP in the figure is the Group of Video Object Planes (GOV), and this layer provides the opportunity for the VOPs to be coded independently of each other, thus allowing random access in the bit stream.

Figure 8.7 provides a less abstract view of how a scene might be decomposed. Here we see the scene broken down into multiple audio and video objects, including a person, the background, furniture, and an audio-visual presentation. Thus, there are objects and other components of the scene, such as the background, that one may expect to remain in the scene and to not change for some amount of time. These components are coded separately, and if they do not change, they need not be encoded and transmitted again until they do change. Notice that the person object is further broken down into a sprite and a voice. The *sprite*, which represents the video of the person excised from the rest of the image, can be coded separately, and the voice can be sent to the appropriate voice coding method. Of course, the sprite and voice information is likely to be changing constantly, and thus it must be continually coded and transmitted, but the background image may not change appreciably over relatively long periods of time, and thus need not be retransmitted very often. The audio-visual presentation could contain high quality audio, and in this case, the audio is coded by one of the coders designed specifically for this task. It should be clear from this description and Figure 8.7 that the object-based approach offers the possibility of much better compression because a particular compression algorithm does not have to handle such a wide variety of inputs.

Just as in MPEG-1 and MPEG-2, a major component of the MPEG-4 standard is the specification of a System Decoder Model (SDM). The architecture of the MPEG-4 Systems Decoder Model is shown in Figure 8.8, along with the network interfaces [4]. The composition and separate encoding of the several audio-visual objects is clearly represented here, but one also sees some network layer interfaces. Delivery Multimedia Integration Framework (DMIF) utilization needs [4]. Perhaps the main impact of Figure 8.8 is that it highlights the wide range of functionalities that MPEG-4 attempts to address [7].

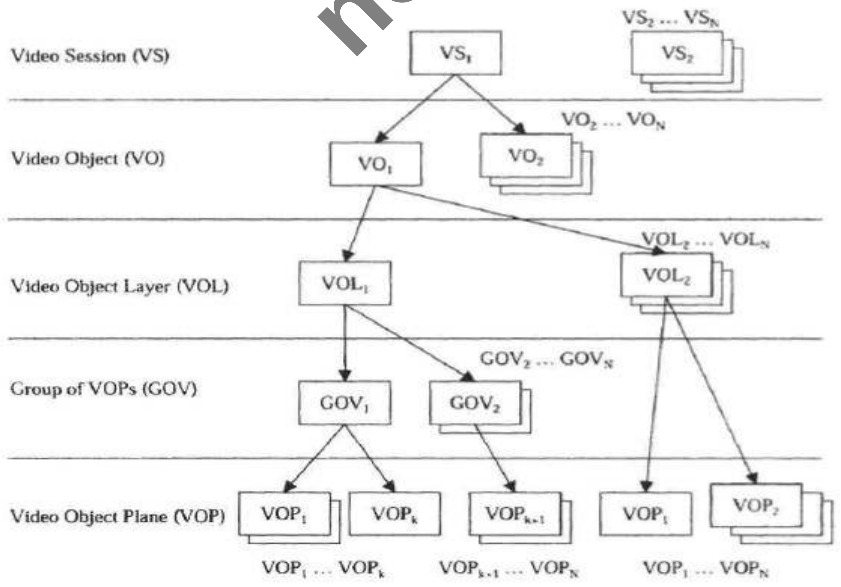
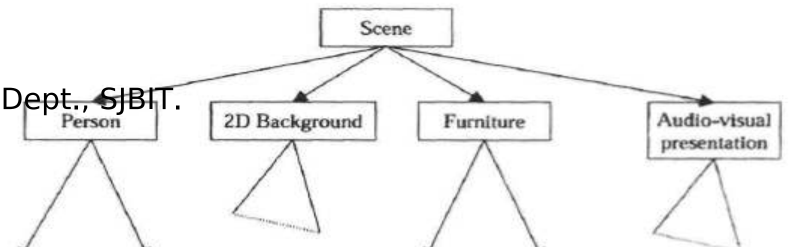


FIGURE 8.6
MPEG-4 video hierarchical description of a scene.



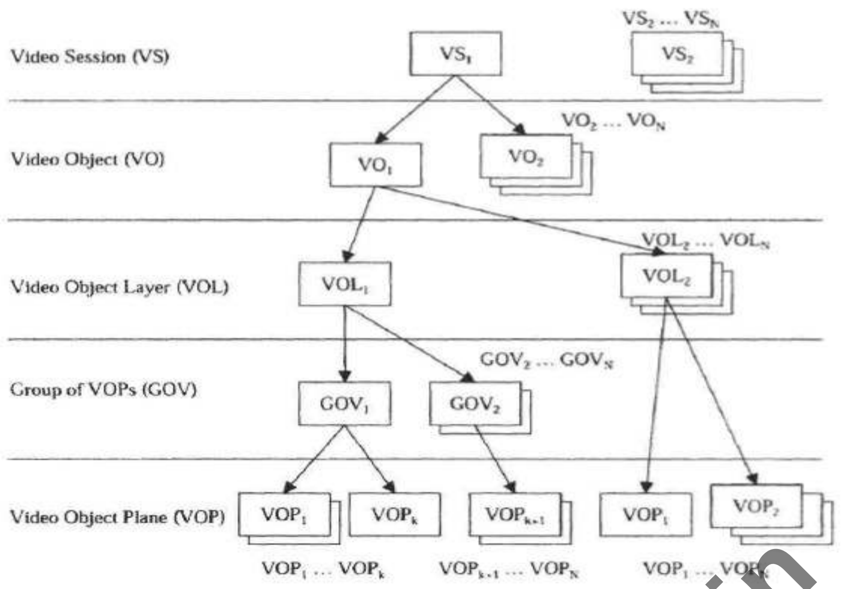


FIGURE 8.6
MPEG-4 video hierarchical description of a scene.

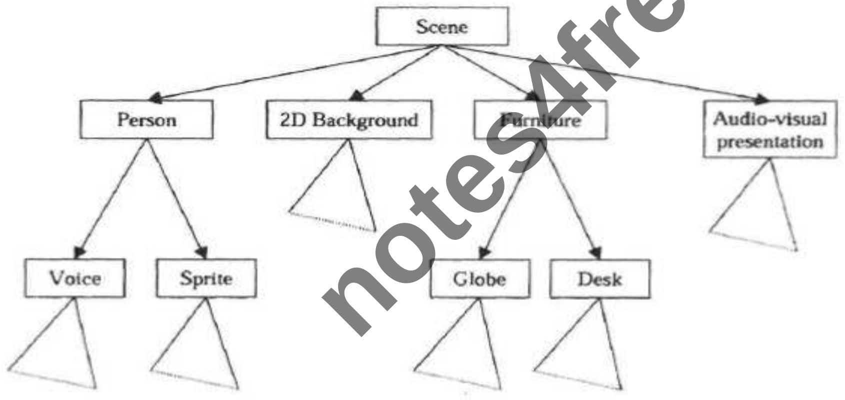


FIGURE 8.7
Hierarchical breakdown of a typical scene.

8.3 MPEG-7

The MPEG-7 standard under development continues the general trend evident in MPEG-4, where the standards are as concerned about functionality as they are about compression. MPEG-7 is designated Multimedia Content Description Interface and is concerned with the interpretation of information in such a way that it can be used or searched by devices or computers. Possible applications include content-based searches or queries, video and audio summarization, and accelerated browsing of Internet sites. The diagram in Figure 8.10 shows a broad view of possible applications for the MPEG-7 standard. The standard intends to specify a set of descriptors, a set of description schemes, and a description definition language, and these are shown in Figure 8.10. MPEG-7 implies a high level of "understanding" by the computing devices in the terminals and networks.

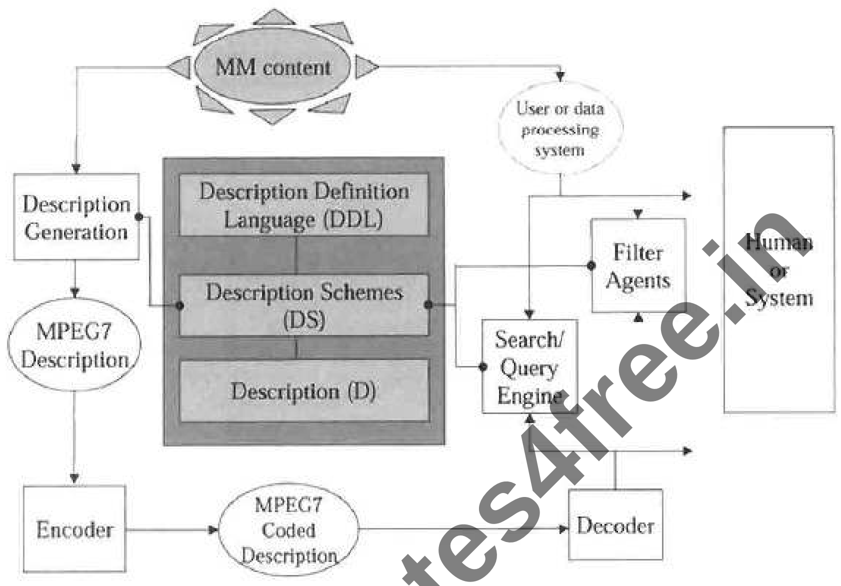
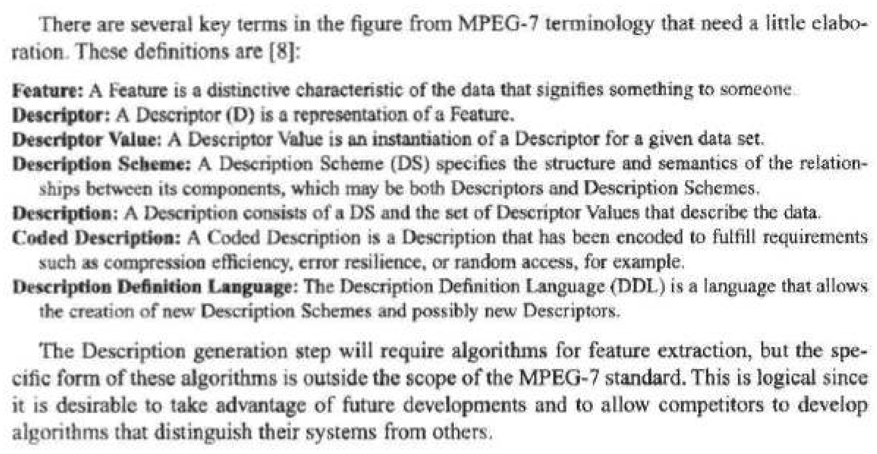


FIGURE 8.10
MPEG-7 applications overview. (MM = multimedia.)



There are several key terms in the figure from MPEG-7 terminology that need a little elaboration. These definitions are [8]:

Feature: A Feature is a distinctive characteristic of the data that signifies something to someone.
Descriptor: A Descriptor (D) is a representation of a Feature.
Descriptor Value: A Descriptor Value is an instantiation of a Descriptor for a given data set.
Description Scheme: A Description Scheme (DS) specifies the structure and semantics of the relationships between its components, which may be both Descriptors and Description Schemes.
Description: A Description consists of a DS and the set of Descriptor Values that describe the data.
Coded Description: A Coded Description is a Description that has been encoded to fulfill requirements such as compression efficiency, error resilience, or random access, for example.
Description Definition Language: The Description Definition Language (DDL) is a language that allows the creation of new Description Schemes and possibly new Descriptors.

The Description generation step will require algorithms for feature extraction, but the specific form of these algorithms is outside the scope of the MPEG-7 standard. This is logical since it is desirable to take advantage of future developments and to allow competitors to develop algorithms that distinguish their systems from others.

PART - B

UNIT - 5

MULTIMEDIA INFORMATION NETWORKS

Introduction, LANs, Ethernet, Token ring, Bridges, FDDI High-speed LANs, LAN protocol.

7 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & publications:

1. H. J. Lee, T. Chiang, and Y. Q. Zhang, Scalable rate control for very low bit rate video, Proc. IEEE ICIP, 2, 768–771 (1997).
2. I. E. G. Richardson, H.264 and MPEG-4 Video Compression – Video Coding for Next- Generation Multimedia, Wiley, Chichester, 2003.
3. Z. Li et al., ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 Document JVT-H014, Adaptive Rate Control with HRD Consideration, May 2003.
4. ISO/IEC 14496–2, Coding of Audio-Visual Objects, Part 2: Visual, Annex L, 2001.
5. Y. S. Saw, Rate-Quality Optimized Video Coding, Kluwer, Norwood, MA, November 1998.
6. H. Lin and P. Mouchatias, Voice over IP signaling: H.323 and beyond, IEEE Comm.Magazine, 38, 142–148 (2000).

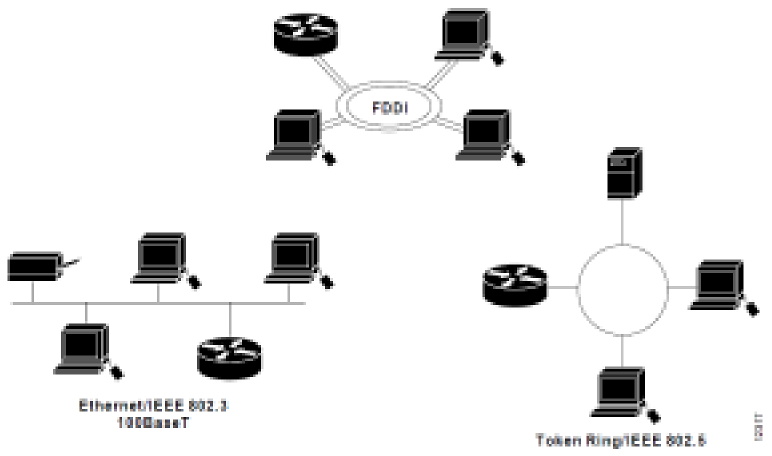
7. H. Schulzrine et al., RTP: A Transport Protocol for Real-Time Applications, IETF RFC1889, IETF, January 1996.
8. M. Handley et al., SIP Session Initiation Protocol, IETF RFC2543, IETF, March 1999.
9. M. Mampaey, TINA for services and signaling and control in next-generation networks, IEEE Comm. Magazine, 38, 104–110 (2000).
10. L. Dimopolou et al., QM tool: An XML-based management platform for QoS-aware IP networks, IEEE Network, 17, 8–14 (2003).
11. G. Pavlov et al., On management technologies and the potential of Web services, IEEE Comm. Magazine, 42, 58–66 (2004).
12. J. Case et al., A Simple Network Management Protocol (SNMP), IETF RFC1157, IETF, May 1990.
13. Common Information Model (CMIP) Version 2.2, Distributed Management Task Force, June 1999.
14. X. Xiao and L. Ni, Internet QoS: a big picture, IEEE Network, 13, 8–18 (1999).
15. Extensible Markup Language (XML) 1.0, W#C Recomm. REC-XML-2001006, October 2000.
16. G. Pavlov, From protocol-based to distributed object-based management architectures, in Proc. 8th IFIP/IEEE Int. Workshop Distributed Systems and Management, 25–40, Sydney, Australia, October 1997.
17. ITU-T Recomm. X.701, Information Technology – Open System Interconnection, System Management Overview, 1992.
18. J. Y. Kim et al., Towards TMN-based integrated network management using CORBA and Java technologies, IEICE Trans Commun., E82-B, 1729–1741 (1999).
19. ITU-T Recomm. M.3010, Principles for a Telecommunications Management Network, ITU, Geneva, Switzerland, May 1996.
20. ITU-T Recomm. X.900, Information Technologies – Open Distributed Processing, Basic Reference Model of Open Distributed Processing, ITU, 1995.

With the rapid paradigm shift from conventional circuit-switching telephone networks to the packet-switching, data-centric, and IP-based Internet, networked multimedia computer applications have created a tremendous impact on computing and network infrastructures. More specifically, most multimedia content providers, such as news, television, and the entertainment industry have started their own streaming infrastructures to deliver their content, either live or on-demand. Numerous multimedia networking applications have also matured in the past few years, ranging from distance learning to desktop video conferencing, instant messaging, workgroup collaboration, multimedia kiosks, entertainment, and imaging.

What is a LAN?

A LAN is a high-speed, fault-tolerant data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications.

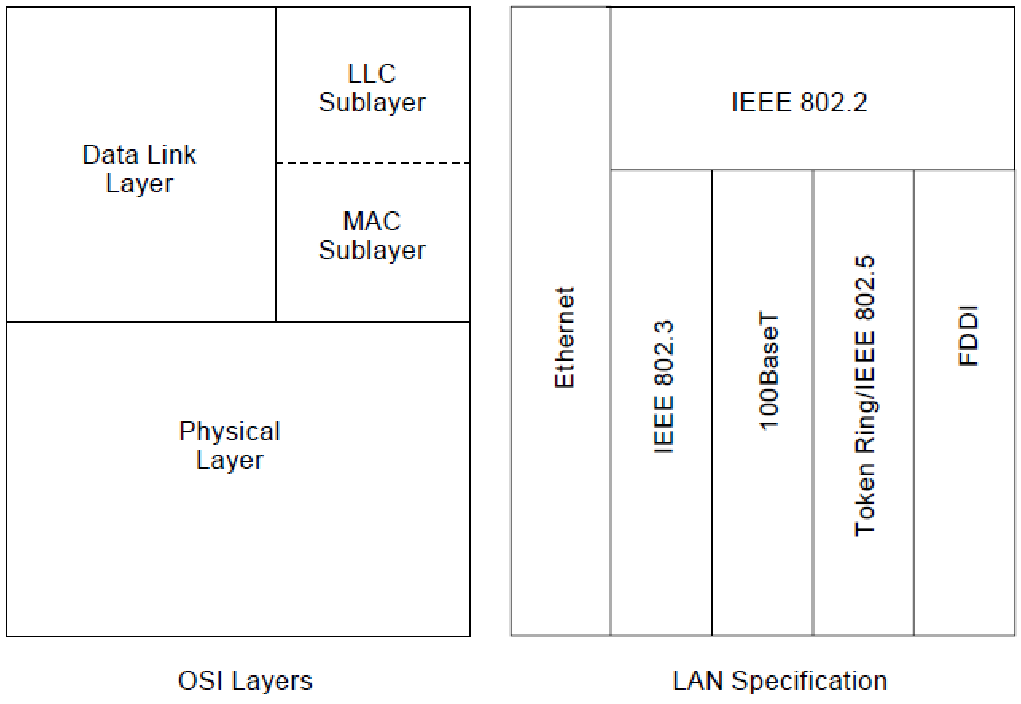
Three LAN implementations are used most commonly:



LAN Protocols and the OSI Reference Model

LAN protocols function at the lowest two layers of the OSI reference model, as discussed in Chapter 1, "Internetworking Basics," between the physical layer and the data link layer. Figure 2-2 illustrates how several popular LAN protocols map to the OSI reference model.

Figure 2-2 Popular LAN protocols mapped to the OSI reference model



LAN Media-Access Methods:

LAN protocols typically use one of two methods to access the physical network medium: *carrier sense multiple access collision detect* (CSMA/CD) and *token passing*. In the CSMA/CD media-access scheme, network devices contend for use of the physical network medium. CSMA/CD is therefore sometimes called *contention access*. Examples of LANs that use the CSMA/CD media-access scheme are Ethernet/IEEE 802.3 networks, including 100BaseT.

In the token-passing media-access scheme, network devices access the physical medium based on possession of a token. Examples of LANs that use the token-passing media-access scheme are Token Ring/IEEE 802.5 and FDDI.

LAN Transmission Methods:

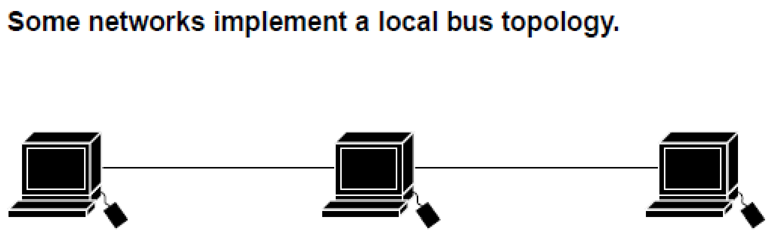
LAN data transmissions fall into three classifications: *unicast*, *multicast*, and *broadcast*. In each type of transmission, a single packet is sent to one or more nodes. In a unicast transmission, a single packet is sent from the source to a destination on a network.

First, the source node addresses the packet by using the address of the destination node. The package is then sent onto the network, and finally, the network passes the packet to its destination. A multicast transmission consists of a single data

packet that is copied and sent to a specific subset of nodes on the network. First, the source node addresses the packet by using a multicast address. The packet is then sent into the network, which makes copies of the packet and sends a copy to each node that is part of the multicast address. A broadcast transmission consists of a single data packet that is copied and sent to all nodes on the
network. In these types of transmissions, the source node addresses the packet by using the broadcast address. The packet is then sent into the network, which makes copies of the packet and sends a copy to every node on the network.

LAN Topologies:

LAN topologies define the manner in which network devices are organized. Four common LANtopologies exist: bus, ring, star, and tree. These topologies are logical architectures, but the actual devices need not be physically organized in these configurations. Logical bus and ring topologies, for example, are commonly organized physically as a star. A bus topology is a linear LAN architecture in which transmissions from network stations propagate the length of the medium and are received by all other stations. Of the three most widely used LAN implementations, Ethernet/IEEE 802.3 networks including 100BaseT, implement a bus topology, which is illustrated in Figure 2-3.

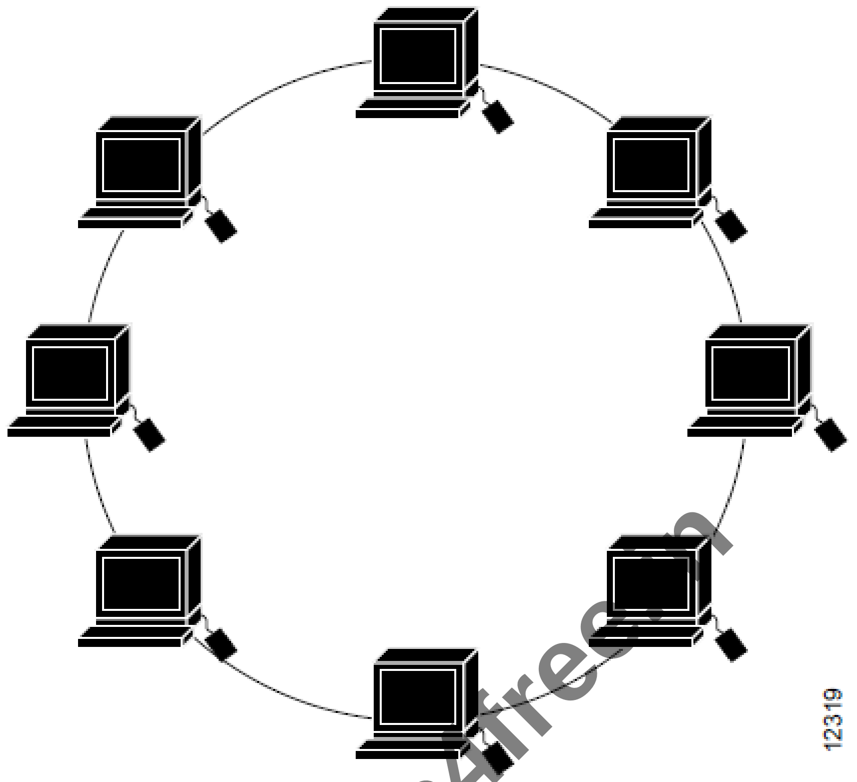


A ring topology is a LAN architecture that consists of a series of devices connected to one another by unidirectional transmission links to form a single closed loop. Both Token Ring/IEEE 802.5 and FDDI networks implement a ring topology. Figure 2-4 depicts a logical ring topology.

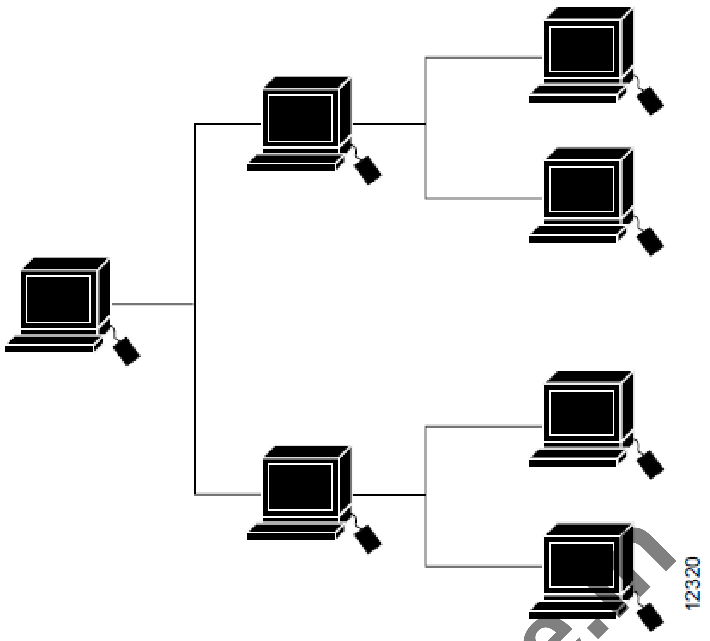
A star topology is a LAN architecture in which the endpoints on a network are connected to a common central hub, or switch, by dedicated links. Logical bus and ring topologies are often implemented physically in a star topology, which is illustrated in Figure 2-5.

A tree topology is a LAN architecture that is identical to the bus topology, except that branches with multiple nodes are possible in this case. Figure 2-5 illustrates a logical tree topology.

Some networks implement a logical ring topology.



A logical tree topology can contain multiple nodes.

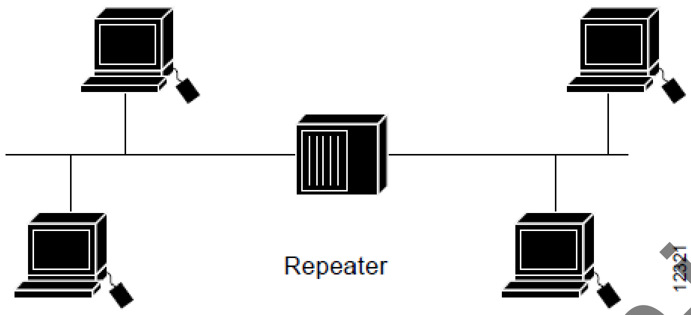


LAN Devices:
Devices commonly used in LANs include *repeaters*, *hubs*, *LAN extenders*, *bridges*, *LAN switches*, and *routers*.

Note Repeaters, hubs, and LAN extenders are discussed briefly in this section.

A *repeater* is a physical layer device used to interconnect the media segments of an extended network. A repeater essentially enables a series of cable segments to be treated as a single cable. Repeaters receive signals from one network segment and amplify, retime, and retransmit those signals to another network segment. These actions prevent signal deterioration caused by long cable lengths and large numbers of connected devices. Repeaters are incapable of performing complex filtering and other traffic processing. In addition, all electrical signals, including electrical disturbances and other errors, are repeated and amplified. The total number of repeaters and network segments that can be connected is limited due to timing and other issues. Figure 2-6 illustrates a repeater connecting two network segments.

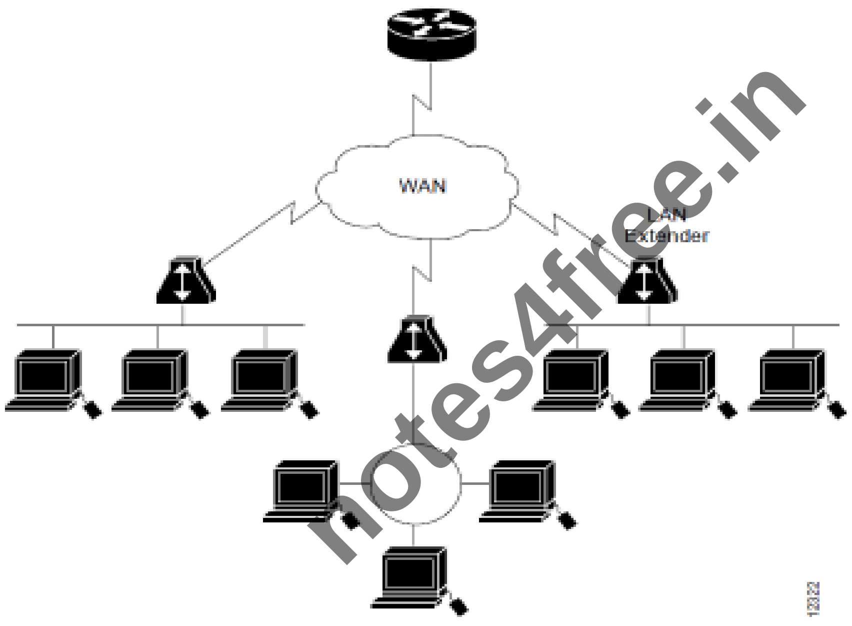
Figure 2-6 A repeater connects two network segments.



A *hub* is a physical-layer device that connects multiple user stations, each via a dedicated cable. Electrical interconnections are established inside the hub. Hubs are used to create a physical star network while maintaining the logical bus or ring configuration of the LAN. In some respects, a hub functions as a multiport repeater.

A LAN *extender* is a remote-access multilayer switch that connects to a host router. LAN extenders forward traffic from all the standard network-layer protocols (such as IP, IPX, and AppleTalk), and filter traffic based on the MAC address or network-layer protocol type. LAN extenders scale well because the host router filters out unwanted broadcasts and multicasts. LAN extenders, however, are not capable of segmenting traffic or creating security firewalls. Figure 2-7 illustrates multiple LAN extenders connected to the host router through a WAN.

Figure 2-7 Multiple LAN extenders can connect to the host router through a WAN.



<u>TOKEN RINGS/BUSES, HIGH SPEED LANS AND BRIDGES:</u>

IEEE 802.4 Standard

PASSING THE TOKEN :

- **IN CSMA/CD (802.3) STARVATION MAY OCCUR, I.E., STATIONS CAN WAIT FOREVER TO TRANSMIT.**
- **IN TOKEN BUS, EVERY STATION HAS A CHANCE TO TRANSMIT (TOKEN) THEREFORE - NO COLLISIONS. IT IS CONTENTION-FREE.**
- **TOKEN PASSES AROUND IN PRE-DEFINED ORDER AND ONCE STATION ACQUIRES TOKEN, IT CAN START TRANSMITTING.**
- **WHEN COMPLETE, THE TOKEN IS PASSED ONTO THE NEXT STATION.**

STATIONS TRANSMITTING :

- **HOWEVER THERE IS LIMITED EFFICIENCY DUE TO PASSING OF THE TOKEN. IT IS MOST COMMONLY**

USED MAC PROTOCOL FOR RING TOPOLOGIES. IT ALSO USES A SPECIAL-PURPOSE, CIRCULATING FRAME, OR TOKEN (3 BYTES).

- Z STATION THAT WANTS TO TRANSMIT WAITS TILL TOKEN PASSES BY.
- Z WHEN STATION WANTS TO TRANSMIT:
  - Y WAITS FOR TOKEN.
  - Y SEIZES IT BY CHANGING 1 BIT AND TOKEN BECOMES START-OF-FRAME SEQUENCE.
  - Y STATION APPENDS REMAINDER OF FRAME.
  - Y WHEN STATION SEIZES TOKEN AND BEGINS TRANSMISSION, THERE'S NO TOKEN ON THE RING; SO NOBODY ELSE CAN TRANSMIT.
- Z TRANSMITTING STATION INSERTS A NEW TOKEN WHEN THE STATION COMPLETES FRAME TRANSMISSION AND THE LEADING EDGE OF FRAME RETURNS TO IT AFTER A ROUND-TRIP. UNDER LIGHT LOAD, INEFFICIENCY DUE TO WAITING FOR THE TOKEN TO TRANSMIT.
- Z UNDER A HEAVY LOAD, ROUND-ROBIN IS FAIR AND EFFICIENT. IN FACT THIS IS ONE OF THE MAJOR ADVANTAGES OF THE TOKEN RING….
- Z THE MONITORING STATION HOWEVER IS RESPONSIBLE FOR RING MAINTENANCE (REMOVING DUPLICATES, INSERTING TOKEN)

TOKEN RING FRAME FORMAT

SD: STARTING DELIMITER; INDICATES STARTING OF FRAME.

AC: ACCESS CONTROL; PPPTMRRR; PPP AND RRR PRIORITY AND RESERVATION; M MONITOR BIT; T TOKEN OR DATA FRAME.

FC: FRAME CONTROL; IF LLC DATA OR CONTROL.

DA AND SA: DESTINATION AND SOURCE ADDRESSES.

FCS: FRAME CHECK SEQUENCE.

ED: ENDING DELIMITER; CONTAINS THE ERROR DETECTION BIT E; CONTAINS FRAME CONTINUATION BIT I (MULTIPLE FRAME TRANSMISSIONS).

FS: FRAME STATUS.

TOKEN RING PRIORITIES:

THERE IS AN OPTIONAL PRIORITY MECHANISM IN 802.5. IT HAS 3 PRIORITY BITS: 8 PRIORITY LEVELS.

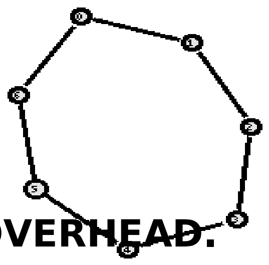
- **Z SERVICE PRIORITY: PRIORITY OF CURRENT TOKEN.**
  - **Y STATION CAN ONLY TRANSMIT FRAME WITH PRIORITY >= SERVICE PRIORITY.**
  - **Y RESERVATION BITS ALLOW STATION TO INFLUENCE PRIORITY LEVELS TRYING TO RESERVE NEXT TOKEN.**
  - **Y GENERALLY A STATION WAITS FOR FRAME TO COME BACK BEFORE ISSUING A NEW TOKEN. THIS CAN LEAD TO LOW RING UTILIZATION.**

- THEREFORE THERE IS AN EARLY TOKEN RELEASE (ETR) OPTION WHERE A STATION MAY RELEASE TOKEN AS SOON AS IT COMPLETES TRANSMISSION.

TOKEN RING SUMMARY:

A TOKEN RING IS:

- EFFICIENT AT HEAVY TRAFFIC.
- GUARANTEED DELAY.
- FAIR.
- SUPPORTS PRIORITIES.
- BUT, RING/TOKEN MAINTENANCE OVERHEAD.
  - CENTRALIZED MONITORING.



HIGH-SPEED LANS – FDDI:

- FIBER DISTRIBUTED DATA INTERFACE IS SIMILAR TO 802.5 WITH SOME CHANGES DUE TO HIGHER DATA RATES.
- 100-1000 MBPS, TOKEN RING LAN - SUITABLE FOR MANS WITH FIBER OR TP AS TRANSMISSION MEDIUM.
- UP TO 100 REPEATERS AND UP TO 2 KM (FIBER) OR 100M (TP) BETWEEN REPEATERS.
- BASIC DIFFERENCES TO 802.5:
  - STATION WAITING FOR TOKEN, SEIZES TOKEN BY FAILING TO REPEAT IT (COMPLETELY REMOVES IT). ORIGINAL 802.5 TECHNIQUE IMPRACTICAL (HIGH DATA RATE).
  - STATION INSERTS NEW FRAME AND EARLY TOKEN RELEASE BY DEFAULT.

HIGH-SPEED LANS – FDDI:

TWO COUNTER-ROTATING FIBER RINGS; ONLY ONE USED FOR TRANSMISSION; THE OTHER FOR RELIABILITY, I.E., SELF-HEALING RING.

High-Speed LANs - 100VG-ANYLAN:

- z voice grade; ANYLAN: support multiple frame types.
- z 802.12 (uses new MAC scheme and not CSMA/CD).
- z Intended to be 100Mbps extension to Ethernet like 100BASE-T.
- z MAC scheme: demand priority (determines order in which nodes share network).
- z Supports both 802.3 and 802.5 frames.

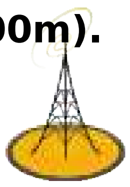
High-Speed LANs - 100VG-ANYLAN:

Topology: hierarchical star.

High-Speed LANs - Fast Ethernet:

- z 100 Mbps Ethernet.
- z IEEE 802.3u standard.
- z Medium alternatives: 100BASE-TX (twisted pair) 100BASE-FX (fiber).
- z IEEE 802.3 MAC and frame format.
- z 10-fold increase in speed => 10-fold reduction in diameter (200m).

Wireless LANs



- z IEEE 802.11.
- z Distributed access control mechanism (DCF) based on CSMA with optional centralized control (PCF).

MAC in Wireless LANs:

- **Distributed coordination function (DCF) uses CSMA-based protocol (e.g., ad hoc networks).**
- **CD does not make sense in wireless.**
  - **Hard for transmitter to distinguish its own transmission from incoming weak signals and noise.**
- **Point coordination function (PCF) uses polling to grant stations their turn to transmit (e.g., cellular networks).**

Switched Ethernet:

- **Point-to-point connections to multi-port hub acting like switch; no collisions.**
- **More efficient under high traffic load: break large shared Ethernet into smaller segments.**

LAN Interconnection Schemes:

- **Extend LAN coverage and merge different types of LAN.**
- **Connect to an internetwork.**
- **Reliability and security.**
- **Hubs or repeaters: physical-level interconnection.**
  - **Devices repeat/amplify signal.**
  - **No buffering/routing capability.**
  - **Bridges: link-layer interconnection.**
  - **Store-and-forward frames to destination LAN.**
  - **Need to speak protocols of LANs it interconnect.**
  - **Routers: network-layer interconnection.**
  - **Interconnect different types of networks.**

Bridges:

- **Operate at the MAC layer.**
  - **Interconnect LANs of the same type, or**
  - **LANs that speak different MAC protocols.**

BRIDGES:

- A BRIDGES FUNCTION IS TO:
  - LISTEN TO ALL FRAMES ON LAN A AND ACCEPTS THOSE ADDRESSED TO STATIONS ON LAN B.
  - USING B'S MAC PROTOCOL RETRANSMITS THE FRAMES ONTO B.
  - DO THE SAME FOR B-TO-A TRAFFIC.
  - BEHAVE LIKE A STATION; HAVE MULTIPLE INTERFACES, 1 PER LAN.
- USE DESTINATION ADDRESS TO FORWARD UNICAST FRAMES; IF DESTINATION IS ON THE SAME LAN, DROPS FRAME; OTHERWISE FORWARDS IT.
- FORWARD ALL BROADCAST FRAMES AND HAVE STORAGE AND ROUTING CAPABILITY
- NO ADDITIONAL ENCAPSULATION.
- BUT THEY MAY HAVE TO DO HEADER CONVERSION IF INTERCONNECTING DIFFERENT LANS (E.G., BUS TO RING FRAME).
- MAY INTERCONNECT MORE THAN 2 LANS AND LANS MAY BE INTERCONNECTED BY MORE THAN 1 BRIDGE.
- IEEE 802.1D SPECIFICATION FOR MAC BRIDGES.

ROUTING WITH BRIDGES:

- BRIDGE DECIDES TO RELAY FRAME BASED ON DESTINATION MAC ADDRESS.
- IF ONLY 2 LANS, DECISION IS SIMPLE.
- IF MORE COMPLEX TOPOLOGIES, ROUTING IS NEEDED, I.E., FRAME MAY TRAVERSE MORE THAN 1 BRIDGE.

- Z DETERMINING WHERE TO SEND FRAME SO THAT IT REACHES THE DESTINATION.
- Z ROUTING BY LEARNING: ADAPTIVE OR BACKWARD LEARNING.

REPEATERS & BRIDGES:

REPEATERS:

- Y EXTEND THE SCOPE OF LANS.
- Y THEY SERVE AS AMPLIFIERS 'REBOOSTING' THE SIGNAL.
- Y THEY HAVE NO STORAGE OR ROUTING CAPABILITIES.

BRIDGES:

- Y ALSO EXTEND SCOPE OF LANS.
- Y ROUTING/STORAGE CAPABILITIES.
- Y OPERATE AT THE DATA LINK LAYER.
- Y ONLY EXAMINE DLL HEADER INFORMATION.
- Y DO NOT LOOK AT THE NETWORK LAYER HEADER.

UNIT - 6

THE INTERNET

Introduction, IP Datagrams, Fragmentation, IP Address, ARP and RARP, QoS Support, IPv8.

7 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **“Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing”**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **“Multimedia Systems Design”**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & publications:

1. ITU-T Recomm. X.900, Information Technologies – Open Distributed Processing, Basic Reference Model of Open Distributed Processing, ITU, 1995.
2. W3D, Web Services Activity Docs., available at http://www.w3c.org/2002/ws.
3. K. B. Song et al., Dynamic spectrum management for next-generation DSL systems, IEEE Comm. Magazine, 40, 101–109 (2002).
4. T. Starr et al., DSL Advances, Prentice Hall, Upper Saddle River, NJ, 2003.
5. K. Kerpez et al., Advanced DSL management, IEEE Commun. Magazine, 41, 116–123 (2003).
6. 3GPP TS29.198, Open Service Access (OSA): Application Programming Interface (API), Part 1–2.
7. Parlay Group, Parlay API Spec. 3.0, December 2001, available at http://www.parlay. org/specs/index.asp.
8. S. Panagiotis, A. Alonistioti, and L. Merkas, An advanced location information management scheme for supporting flexible service provisioning in reconfigurable mobile networks, IEEE Commun. Magazine, 41, 88–98 (2003).
9. 3GPP TS23.271, Functional, Stage 2 Description of LCS.
10. 3GPP TS23.240, 3GPP Generic User Profile – Architecture, Stage 2.
11. 3GPP TS22.071, Location Services (LCS): Service Description, Stage 1.
12. B. Shafiq et al., Wireless network resource management for web-based multimedia documents services, IEEE Commun. Magazine, 41, 138–145 (2003).
13. S. Baqai, M. Woo, and A. Ghafoor, Network resource management for enterprise-wide multimedia services, IEEE Commun. Magazine, 34, 78–85 (1996).
14. T. D. C. Little and A. Ghafoor, Multimedia synchronization protocols for integrated services, IEEE J. Selected Areas in Comm., 9, 1368–1382 (1991). REFERENCES.
15. ITU-T Recomm. G.709, Synchronous Multiplexing Structure, ITU, March 1993.
16. K. Sato, S.Okomoto, and H.Hadama,Network performance and integrity enhancement with optical path layer technologies, IEEE J. Selected Areas in Commun., 12, 159–170 (1994).
17. ISO/IEC 13818 MPEG-2, Information Technology: Generic Coding of Moving Pictures and Associate Audio Information, 1995.
18. IETF RFC1889, RTP: A Transport Protocol for Real-Time Applications, IETF, January 1996.
19. IETF RFC793, Transmission Control Protocol, IETF, September 1981.
20. IETF RFC2357, IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols, IETF, June 1998.
21. J. W. Atwood, A classification of reliable multicast protocols, IEEE Network, 18, 24–34 (2004).

What is Internet?

Interconnection of computers and computer networks using TCP/IP communication protocol

• Transport Control Protocol/ Internet Protocol (TCP/IP)

What is a protocol?

A protocol is a set of rules defining communication between systems

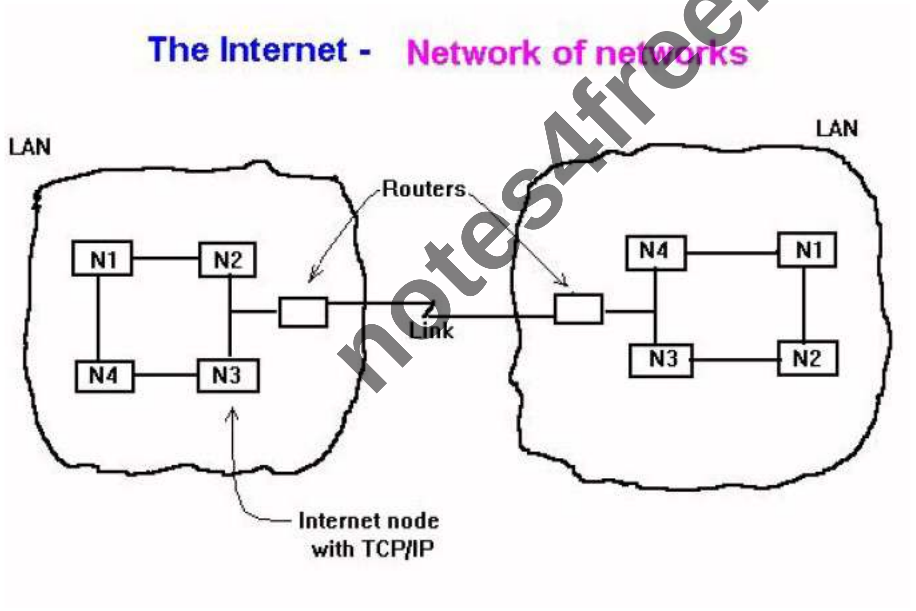
• Intranet: Use of TCP/IP to connect computers on an organizational LAN

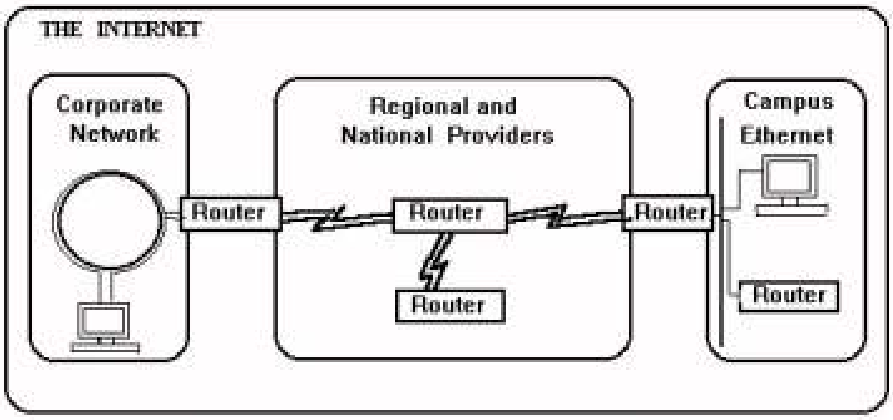
• Internet: Use of TCP/IP to interconnect such networks

• TCP/IP is the basic Internet protocol

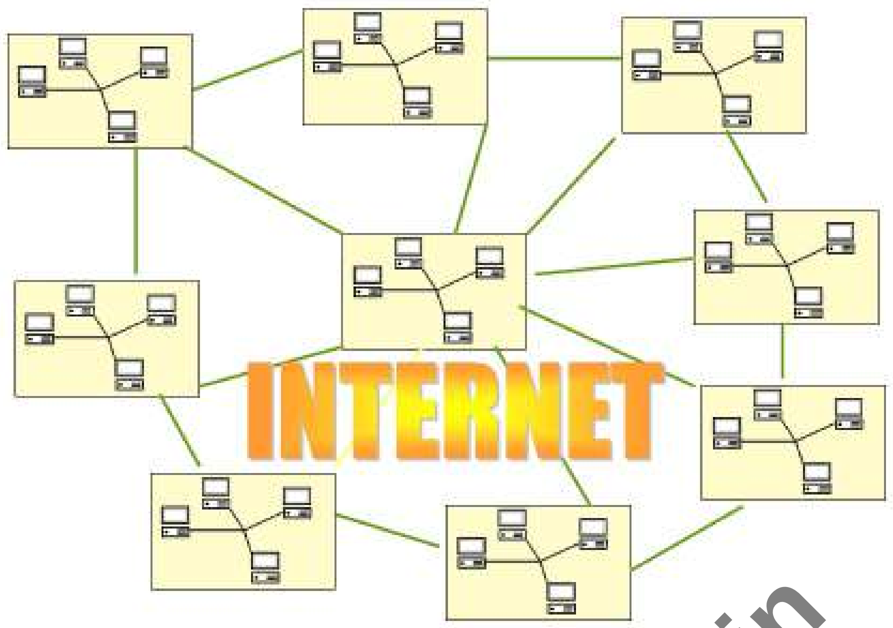
• Various application protocols operate over TCP/IP

– SMTP (E-Mail), HTTP (Web), IRC (Chat), FTP (File transfer), etc.





INTERNET HARDWARE



TCP/IP

• Rules for information exchange between computers over a network

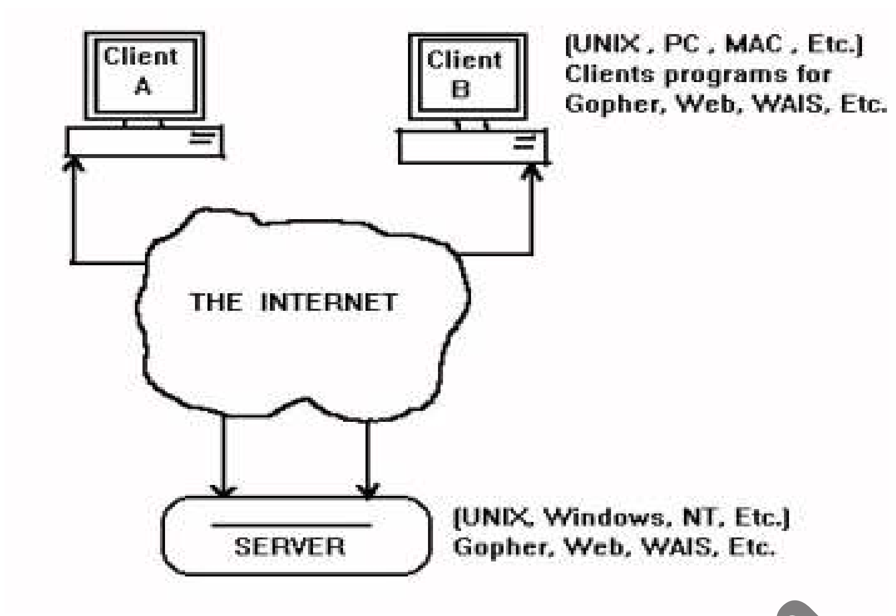
• ‘Packet’ based – segment/ de-segment information

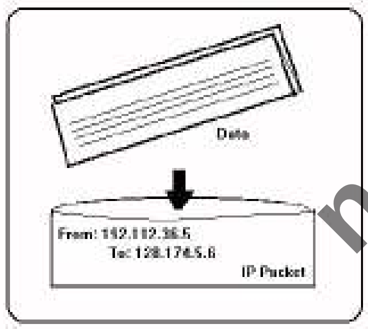
• Client-Server (Request/ Response)

– Web browser (client), Website (Server)

• TCP – Handles data part

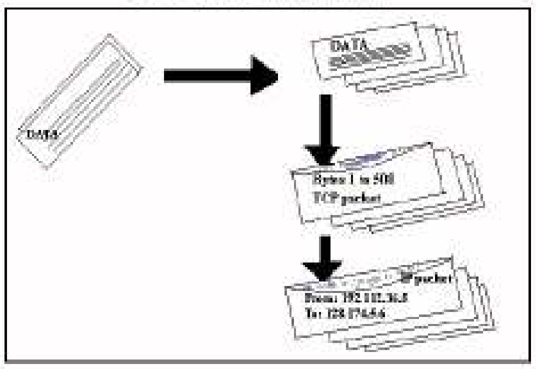
• IP – Handles address part – Identification of every computer on the Internet – IP address

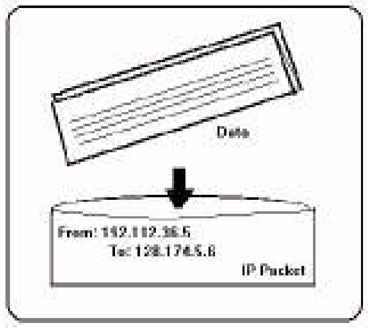




IP Envelopes

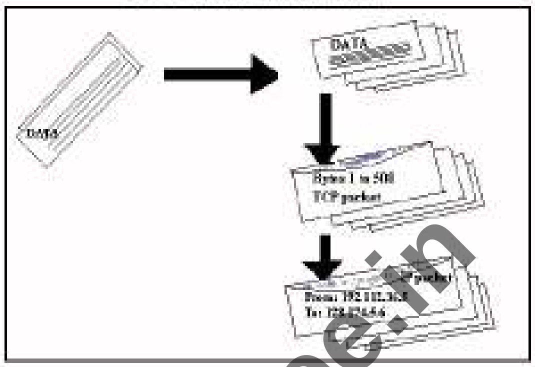
TCP Packet Encapsulation





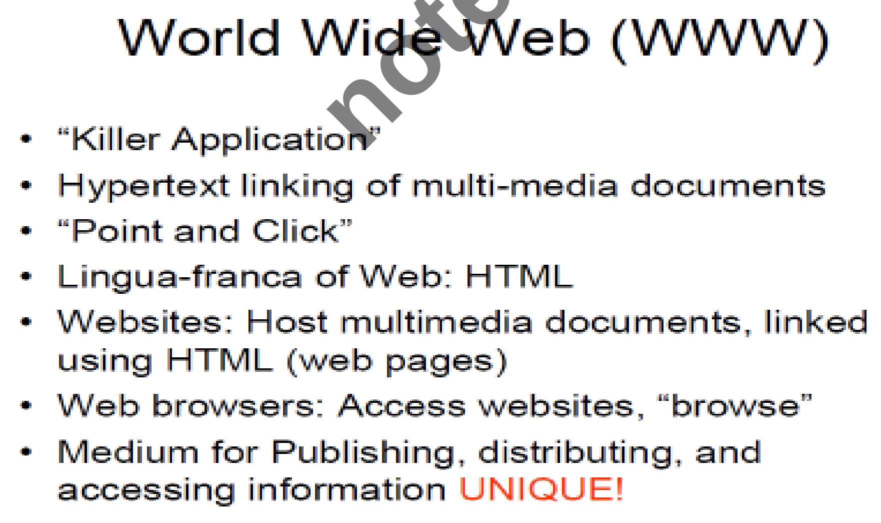
IP Envelopes

TCP Packet Encapsulation



Internet Services

- E-Mail
  - One-to-one
  - One-to-many (e.g. discussion forums)
- Telnet – remote login (e.g. library catalogue access)
- FTP: File transfer (e.g. software packages)
- Web (HTTP): Hypertext linking/navigation
- IRC: Internet Relay Chat
- Internet telephony, mobile access, etc.



World Wide Web (WWW)

- "Killer Application"
- Hypertext linking of multi-media documents
- "Point and Click"
- Lingua-franca of Web: HTML
- Websites: Host multimedia documents, linked using HTML (web pages)
- Web browsers: Access websites, "browse"
- Medium for Publishing, distributing, and accessing information UNIQUE!

Bring customers to your website with Google AdWords

Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil

Advertising Programs - Business Solutions - About Google - Go to Google.com

Make Google Your Homepage!

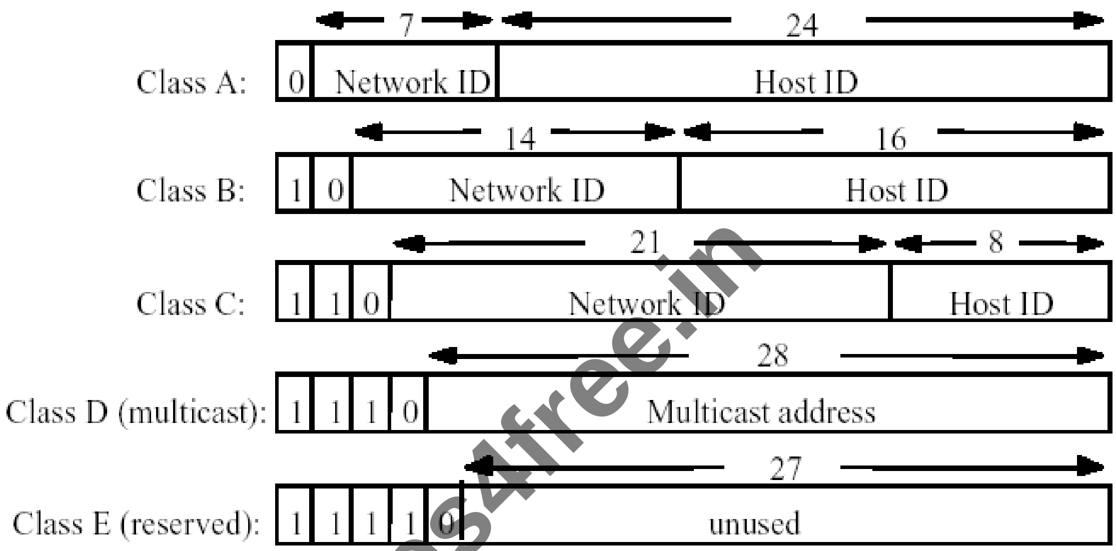
©2004 Google - Searching 4,285,199,774 web pages

Introduction to IP:

- **The network protocol in the Internet**
- **IP address**
  → Network ID + Host ID
- **Transmit datagrams from one host to another. if necessary, via intermediate routers**
- **Network Information Center(NIC)**
- **Unreliable packet delivery**
- **Only header check sum**
- **When an IP datagram is longer than the MTU of the underlying network**
  → broken into smaller packets at the source, and
  → reassembled at the final destination
- **Address resolution**
  - convert IP address to network address of a host for a specific underlying network
  (Ex) convert 32 bits IP address to 48 bits Ethernet address
- **Translation is network Technology-dependent**

IPV4 Addressing:

- **IETF RFC 791 (1981)**
- **32 bits numeric id - 4 billion hosts (Ex 223.1.1.219)**
- **Using Dot notation**
- **Network has an address, using network mask, (Ex 223.1.1.0 /24)**
-



Classless InterDomain Routing(CIDR)

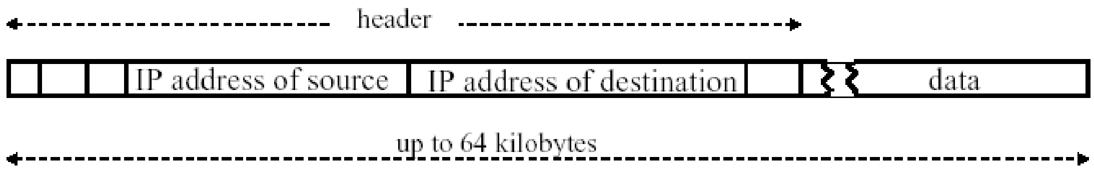
→ **Problem of shortage of IP address (scarify of class B, plenty of class C)**

→ **Allocate a batch of contiguous class C address to a subnet requiring more than 255 addresses**

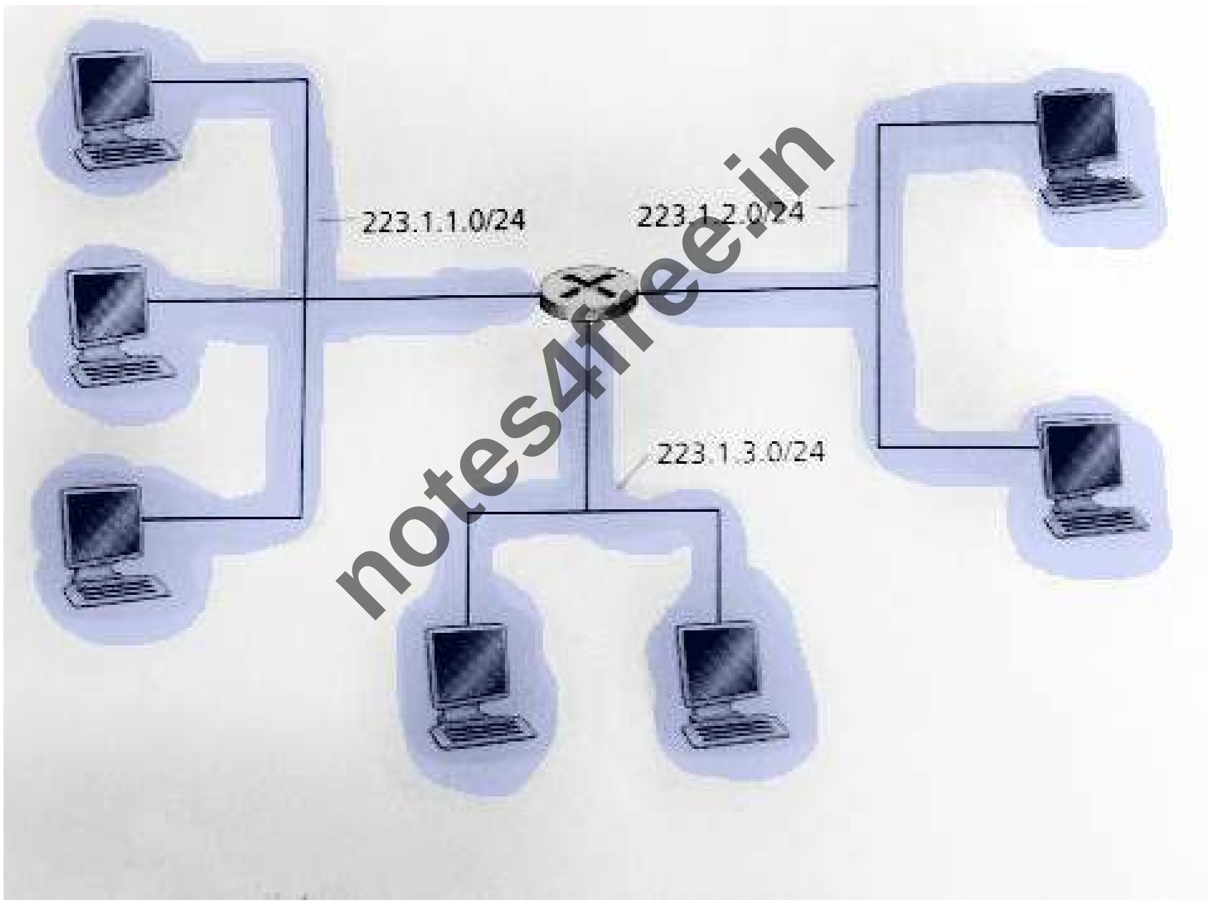
à Add a mask field to the routing table

→ **IETF RFC 1519 (1993)**

Moving a Datagram from Source to Destination:



IP packet layout



IP Datagram Format (IPv4):

IP Datagram Fragmentation:

- **When an IP datagram is longer than the MTU of the underlying network**
  à broken into smaller packets at the source, and
  → reassembled at the final destination

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in data filed	Identification = 777	Offset = 0	flag = 1 (there is more)
2nd fragment	1,480 bytes in dat filed	Identification = 777	Offset = 1,480	flag = 1 (there is more)
3rd fragment	1020 bytes (3,980 - 1,480 - 1,480)	Identification = 777	Offset = 2,906	flag = 0 (last fragment)

IP Addresses

In order for systems to locate each other in a distributed environment, nodes are given explicit addresses that uniquely identify the particular network the system is on and uniquely identify the system to that particular network. When these two identifiers are combined, the result is a globally-unique address.

This address, known as ?IP address?, as ?IP number?, or merely as ?IP? is a code made up of numbers separated by three dots that identifies a particular computer on the Internet. These addresses are actually 32-bit binary numbers, consisting of the two sub addresses (identifiers) mentioned above which, respectively, identify the network and the host to the network, with an imaginary boundary separating the two. An IP address is, as such, generally shown as 4 octets of numbers from 0-255 represented in decimal form instead of binary form.

For example, the address 168.212.226.204 represents the 32-bit binary number 10101000.11010100.11100010.11001100.

The binary number is important because that will determine which class of network the IP address belongs to. The Class of the address determines which part belongs to the network address and which part belongs to the node address (see IP address Classes further on).

The location of the boundary between the network and host portions of an IP address is determined through the use of a subnet mask. This is another 32-bit binary number which acts like a filter when it is applied to the 32-bit IP address. By comparing a subnet mask with an IP address, systems can determine which portion of the IP address relates to the network and which portion relates to the host. Anywhere the subnet mask has a bit set to ?1?, the underlying bit in the IP address is part of the network address. Anywhere the subnet mask is set to ?0?, the related bit in the IP address is part of the host address.

The size of a network is a function of the number of bits used to identify the host portion of the address. If a subnet mask shows that 8 bits are used for the host portion of the address block, a maximum of 256 host addresses are available for that specific network. If a subnet mask shows that 16 bits are used for the host portion of the address block, a maximum of 65,536 possible host addresses are available for use on that network.

An Internet Service Provider (ISP) will generally assign either a static IP address (always the same) or a dynamic address (changes every time one logs on). ISPs and organizations usually apply to the InterNIC for a range of IP addresses so that all clients have similar addresses. There are about 4.3 billion IP addresses. The class-based, legacy addressing scheme places heavy restrictions on the distribution of these addresses. TCP/IP networks are inherently router-based, and it takes much less overhead to keep track of a few networks than millions of them.

IP Classes

Class A addresses always have the first bit of their IP addresses set to ?0?. Since Class A networks have an 8-bit network mask, the use of a leading zero leaves only 7 bits for the network portion of the address, allowing for a maximum of 128 possible network numbers, ranging from 0.0.0.0 ? 127.0.0.0. Number 127.x.x.x is reserved for loopback, used for internal testing on the local machine.

Class B addresses always have the first bit set to ?1? and their second bit set to ?0?. Since Class B addresses have a 16-bit network mask, the use of a leading ?10? bit-pattern leaves 14 bits for the network portion of the address, allowing for a maximum of 16,384 networks, ranging from 128.0.0.0 ? 181.255.0.0.

Class C addresses have their first two bits set to ?1? and their third bit set to ?0?. Since Class C addresses have a 24-bit network mask, this leaves 21 bits for the network portion of the address, allowing for a maximum of 2,097,152 network addresses, ranging from 192.0.0.0 ? 223.255.255.0.

Class D addresses are used for multicasting applications. Class D addresses have their first three bits set to ?1? and their fourth bit set to ?0?. Class D addresses are 32-bit network addresses, meaning that all the values within the range of 224.0.0.0 ? 239.255.255.255 are used to uniquely identify multicast groups. There are no host addresses within the Class D address space, since all the hosts within a group share the group?s IP address for receiver purposes.

Class E addresses are defined as experimental and are reserved for future testing purposes. They have never been documented or utilized in a standard way.

The Paessler network monitoring products PRTG Traffic Grapher and IPCheck Server Monitor use the IP address in order to connect to the respective machines they are intended to monitor / graph.

If you happen to know the IP address of your provider's DNS server, the mailserver, the news server and possibly some other machines, you will realize that very often the first three octets of their IP addresses are the same, for example 192.168.43.4 for the DNS server, 192.168.43.5 for the mail server, 192.168.43.7 for the news server and 192.168.43.25 for the secondary DNS server. This is not just by chance.

Instead of giving out one IP address by one, there are classes which are assigned to organizations. A, B, and C classes are the most known ones, with the C-class the most common one. There are only 127 A-class ranges, or networks, but each of them has 16,777,214 addresses for hosts. There are 16,384 possible B-class networks with 65,534 addresses for hosts each and 2,097,152 C-class networks with 254 possible host addresses each.

ARP and RARP

The Address Resolution Problem:

- *? How does a host or gateway map an IP address to the correct physical address when it needs to send a packet over a physical network ?*
- Devise a low-level software that hides physical addresses and allows higher-level programs to work only with internet addresses.
- Mapping of high-level to low-level addresses is the *address resolution problem*.

IP address A	conceptual	B IP address
Physical address A	physical	B Physical address

Physical Addresses:

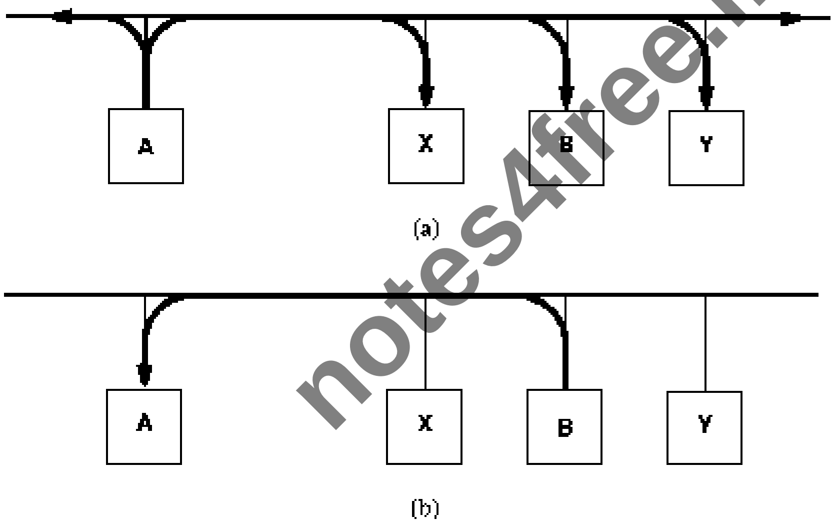
- **Two basic types of physical addresses:**
    - *Large*
        - Ethernet
        - 48 bits
    - *Small*
        - Token Ring (proNET-10)
        - 8 bits
- *proNET-10* - uses Class C addresses with host-id portion = 1, 2, ... , 255

Physical Addresses: Mapping

- *Mapping must be computationally efficient* (simply mask all portions of the address excluding the host-id portion)
- When address mapping can only be done via an address table (X.25 to IP), *hashing is used to speed up lookup.*
- Problem with representation of 48-bit Ethernet addresses within a 32-bit IP address and allowing new machines to be added without recompilation.
- Avoid maintaining a static table of mappings by using the *ARP* (*Address Resolution Protocol*).

Address Resolution Protocol (ARP):

- *ARP* - is a low-level protocol used to bind addresses dynamically.
- ARP allows a host to find a physical address of a target host on the same physical network, given only it's IP address.
- ARP hides the underlying network physical addressing. It can be thought of as part of physical network system and not the internet protocols.



- ARP broadcasts special packets with the destination's IP address to ALL hosts.
- The destination host (only) will respond with it's physical address.
- When the response is received, the sender uses the physical address of destination host to send all packets.

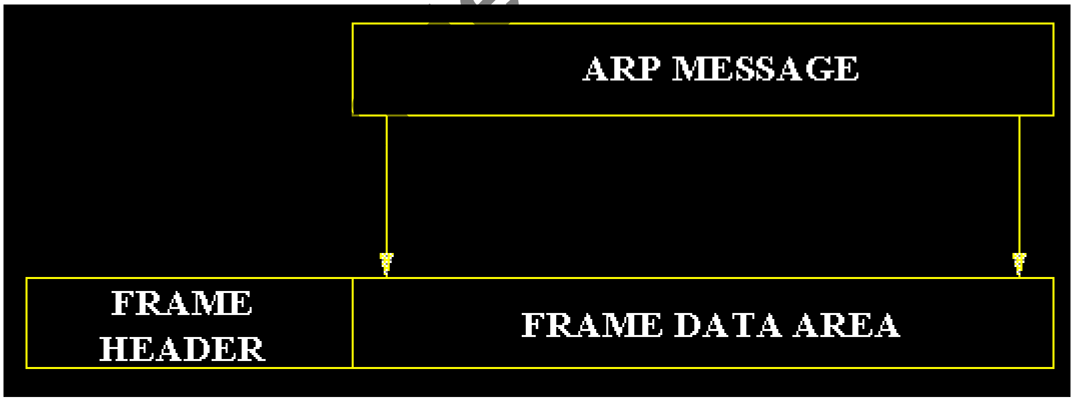
ARP Inefficiencies and Improvements:

- Broadcasting is expensive on network resources, so an *ARP cache* of recently acquired IP-to-Physical address bindings is kept.
- **Other Broadcast Improvements:**
  - Include the sender's IP-to-Physical address binding along with the request, to the destination. This reduces future traffic.
  - During each broadcast, ALL machines can find out the senders physical address and record it locally in it's ARP cache.
  - When a new machine comes on-line, it immediately broadcasts it's IP-to-Physical address binding to all nodes.

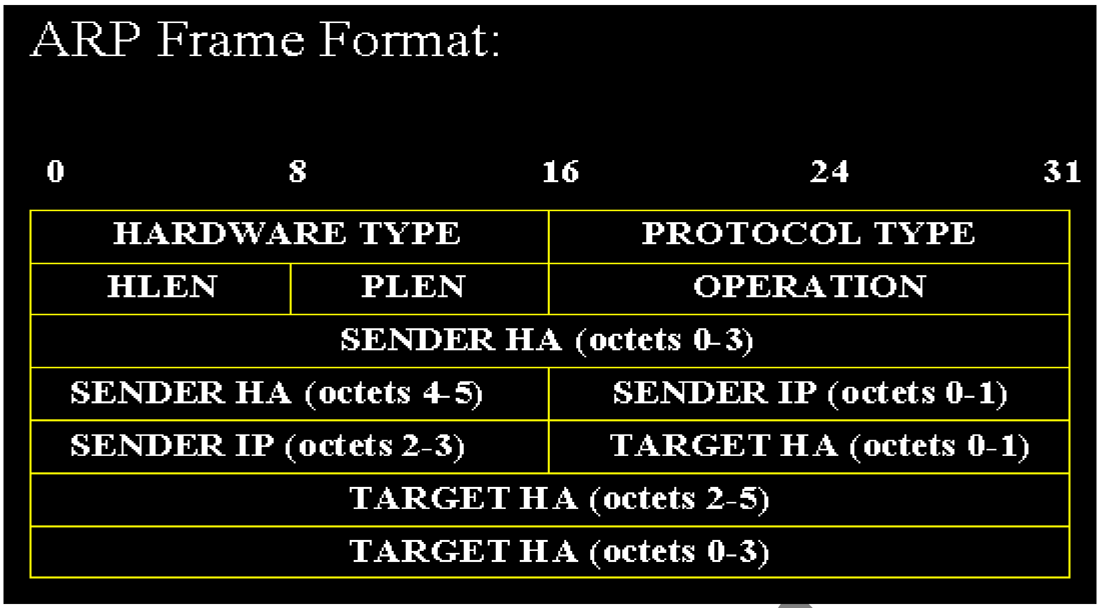
ARP Functionality:

- **There are two main functional parts of the address resolution protocol:**
  - Determine the destination's physical address before sending a packet.
  - Answer requests that arrive for it's own Physical-to-IP address binding.
- Because of lost/duplicate packets, ARP must handle this to avoid many re-broadcasts.

- Bindings in ARP cache (actual cache table) must be removed after a fixed period of time to ensure validity.
- When a packet is received, the sender's IP address is stripped and the local table is updated (ARP cache), then the rest of the packet is processed.
- **Two types of incoming packets:**
    - Those to be processed (correct destination).
    - Stray broadcast packets (can be dropped after updating the ARP cache).
- Application programs may *request the destination address many times before the binding is complete.* This must be handled, by discarding enqueued requests, when the correct binding returns.



- ARP sets the field "*TYPE*" for the *ID of a frame*.
- ARP packets DO NOT have a fixed format header, so they can be used with arbitrary physical addresses and arbitrary protocol addresses.
- The lengths of physical addresses may vary up to 48-bits.



ARP Frame Format:

0	8	16	24 – 31
HARDWARE TYPE		PROTOCOL TYPE	
HLEN	PLEN	OPERATION	
SENDER HA (octets 0-3)			
SENDER HA (octets 4-5)		SENDER IP (octets 0-1)	
SENDER IP (octets 2-3)		TARGET HA (octets 0-1)	
TARGET HA (octets 2-5)			
TARGET HA (octets 0-3)			

Introduction to QoS

• QoS developments in IP networks is inspired by new types of applications: VoIP, audio/video streaming, networked virtual environments, interactive gaming, videoconferencing, video distribution, e-commerce, GRIDs & collaborative enviroments, etc.

• **Quality-of-Service (QoS)** is a set of service requirements (performance guarantees) to be met by the network while transporting a flow.

QoS Architectures

• Best Effort Internet

• Integrated Services

– Performance guarantees to traffic and resource reservations are provided on per-flow basis.

– Guaranteed & Controlled Load Service

– Scaling issues (per flow state information)

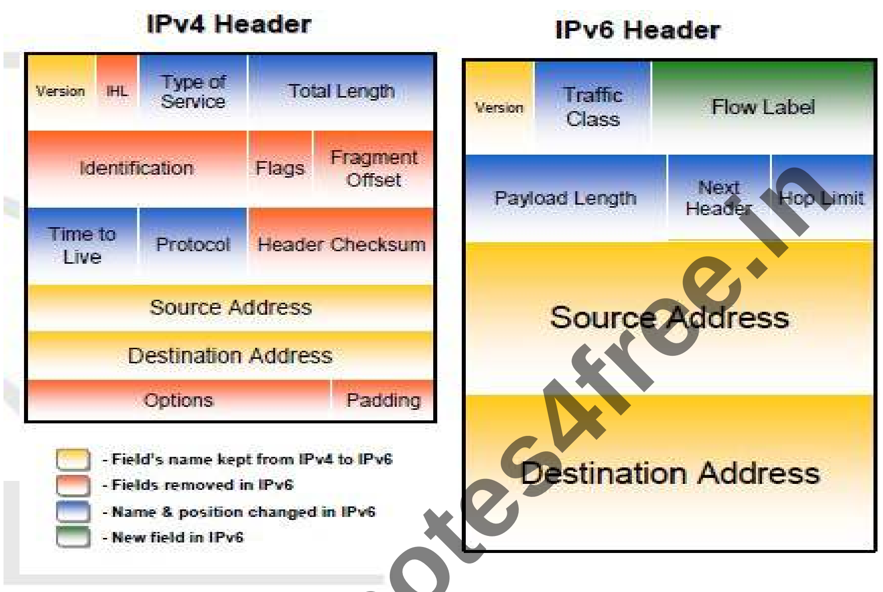
• Differentiated Services

– Performance guarantees are provided to traffic aggregates rather than to flows.

– Per-Hop Behaviours (PHB): EF & AF

– Lack of any signalling protocol for resource allocation (admission control) and QoS mechanisms control.

– Example of services: Premium, “Silver”, LBE



IPv8: Peer-to-Peer overlay network

In short: a library for networking in distributed applications based on a P2P-overlay which handles IP changes, strong identities, trust levels, and neighbourhood graphs.

Overview

Problems with the very fabric of The Internet, IPv4, are mounting. The approach of IPv6, Mobile IP, and IPSec is hampered by fundamental architectural problems. A superior solution is moving the intelligence up to a higher layer in the protocol stack and towards the end points.

We have the expertise to design and build innovative P2P overlay software. Our overlay will offer a secure network connection to either a known person or a specific computer which is robust against eavesdropping, man-in-the-middle attacks, peer failure, network failure, packet loss, change of IP numbers, network mobility, and blocking by NAT/Firewalls. *Our solution exposes trust and reputation levels to the networking layer to lower the risk of DDOS attacks.*

Functionality

IPv8 is an P2P overlay network which unlocks more advanced functionality. Over the coming 5 years we aim to evolve this technology and offer the following functionality:

- Direct, safe, and robust communication between you and any other node
- Determine the friendship paths between you and any other node by integrating existing web-based social networks
- Estimate the trust level between you and any other node
- Exchange of multimedia information of any size or popularity
- Transfer of virtual currency (credits) or real money to any other node

This is a protected section. You will not be able to view this without a correct authentication.

ToDo?: Also manage internal network addresses, discover external network address, connect to peers within subnet with internal IP address. Expand with NAT/Firewall puncturing, UDP/HTTP encapculation, user space TCP rate control, relaying through proxies.

Performance and awareness

IPv8 also enables a new interface for performance and network awareness. Currently every application has to guess the available bandwidth, latency, etc. while all this information is availbe in the hidden TCP state. Especially for network-dependent applications this can boost effectiveness and efficiency. (As nicely described years ago by MIT people in the Daytona paper)

TCP manages each stream/connection separately; when working with multiple concurrent streams, TCP has issues. As P2P routinely employs numerous connections, that issues surface . E.g. BitTorrent has 4 upload connection slots - otherwise, Cohen claims, TCP performance is suboptimal.

So, managing all streams by a single control loop may bring some benefits.

BROADBAND ATM NETWORKS

Introduction, Cell format, Switfh and Protocol Architecture ATM LANs.

6 Hours

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

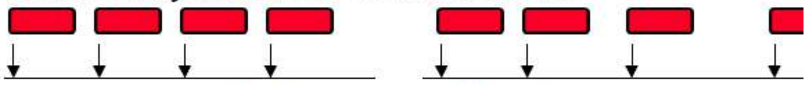
Journals & Publications:

1. T. W. Strayer, B. J. Dempsey, and A. C. Weaver, XTP – The Xpress Transfer Protocol, Addison-Wesley, 1992.
2. W. T. Strayer, Xpress Transport Protocol (XTP) Specification Version 4.0b, Technical Report, XTP Forum, June 1998.
3. S. Floyd et al., A reliable multicast framework for lightweight sessions and application level framing, IEEE/ACM Trans. Network, 5, 784–803 (1997).
4. IETF RFC3208, PGM Reliable Transport Protocol Specification, IETF, December 2001.
5. M. Hofmann, Enabling group communications in global networks, in Proc. Global Networking, II, 321–330 (1997).
6. S. Paul et al., Reliable multicast transport protocol (RMTP), IEEE J. Selected Areas in Comm., 15, 407–421 (1997).
7. K. R. Rao, Z. S. Bojkovic, and D. A. Milovanic, Multimedia Communication Systems, Prentice Hall, Upper Saddle River, NJ, 2002.
8. P. H. Ho and H. T. Mouftah, A novel distributed control protocol in dynamic wavelength routed optical networks, IEEE Commun. Magazine, 40, 38–45 (2002).
9. I. Chlamac, A. Ferego, and T. Zhang, Light path (wavelength) routing in large WDMnetworks, IEEE J. Selected Areas in Comm., 14, 909–913 (1996).
10. P. H. Ho and H. T. Mouftah, A framework of service guaranteed shared protection for optical networks, IEEE Commun. Magazine, 40, 97–103 (2002).
11. P. H. Ho and H. T. Mouftah, Capacity-balanced alternate routing for MPLS traffic engineering, in Proc. IEEE Int. Symp. Comp. Commun., 927–932, Taormina, Italy, July 2002.
12. L. C. Wolf, C. Griwodz, and R. Steinmetz, Multimedia communication, Proc. IEEE, 85, 1915–1933 (1997).
13. D. Pei and L. Zhang, A framework for resilient internet routing protocols, IEEE Network 18, 5–12 (2004).
14. M. Tatipamula and B. Khasnabish (Eds.), Multimedia Communications Networks Technologies and Services, Artech House, Boston, 1998.
15. ITU-T Q29/11, Recomm. Q.NSEC, ITU, July 1995.
16. A. Chakrabarti and G. Mammaran, Internet infrastructure security: a taxonomy, IEEE Network, 16, 13– (2002). 720 NETWORK LAYER
17. C. P. Pfleeger, Security in Computing, Prentice Hall, Upper Saddle River, NJ, 1996.

18. GSM 03.60, GPRS Service Description, Stage 2, 1998.
19. 3GPP TS 33.120, 3G Security: Security Principles and Objectives, May 1999.
20. Ch. Xenakis and L. Merakos, On demand network-wide VPN deployment in GPRS, IEEE Network, 16, 28–37 (2002).

ATM Networks: Overview

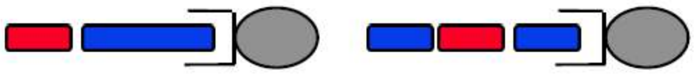
- STM = Synchronous Transfer Mode,
  ATM = Asynchronous Transfer Mode



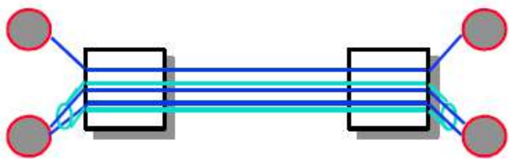
Allows any-speed and even variable rate connection

Broadband = Rate greater than primary rate (1.5 Mbps)

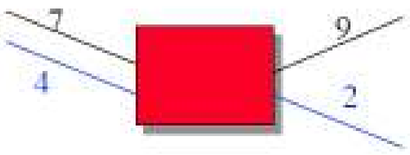
- ATM = Short fixed size 53-byte cells



- Connection oriented ⇒ Virtual Channels (VC)

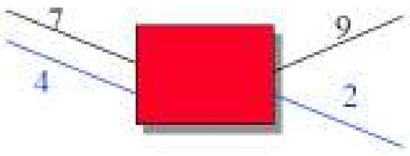


- Labels vs addresses
  ⇒ Better scalability in number of nodes



- Slotted system ⇒ Better scalability in distance-bandwidth
- Switches vs routers
  ⇒ Cheaper due to fixed size, short address, simplicity
- Seamless ⇒ Same technology for LAN, MAN, WAN
- Data, voice, video integration
- Everyone else is doing it

- Labels vs addresses
  ⇒ Better scalability in number of nodes



- Slotted system ⇒ Better scalability in distance-bandwidth
- Switches vs routers
  ⇒ Cheaper due to fixed size, short address, simplicity
- Seamless ⇒ Same technology for LAN, MAN, WAN
- Data, voice, video integration
- Everyone else is doing it

History of ATM

q 1980: Narrowband ISDN adopted

q Early 80's: Research on Fast Packets

q Mid 80's: B-ISDN Study Group formed

q 1986 ATM approach chosen for B-ISDN

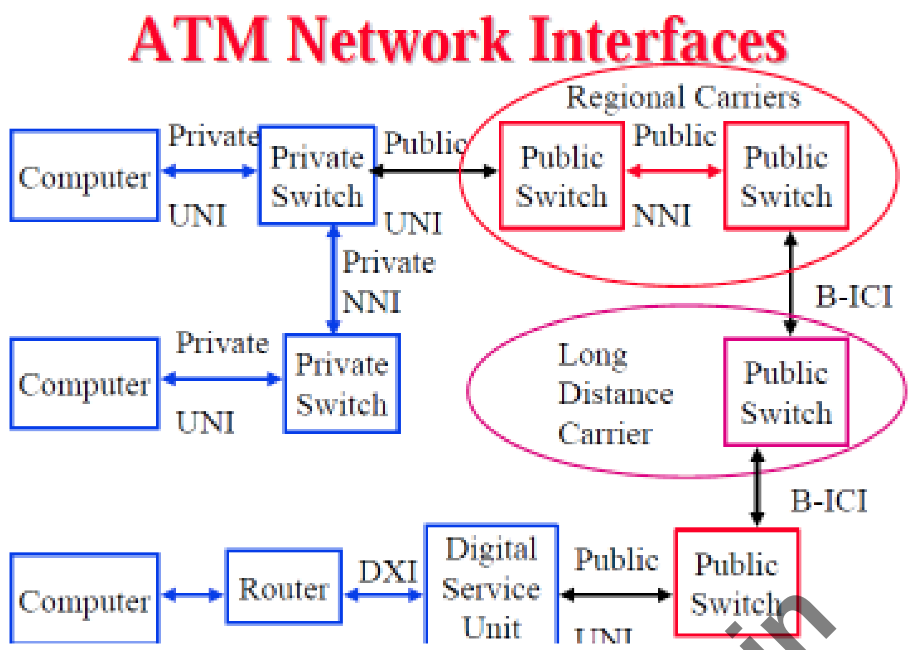
q June 1989: 48+5 chosen (64+5 vs 32+4).

q October 1991: ATM Forum founded

q July 1992: UNI V2 released by ATM Forum

q 1993: UNI V3 and DXI V1

q 1994: B-ICI V1



ATM Network Interfaces

q User to Network Interface (UNI):

Public UNI, Private UNI

q Network to Node Interface (NNI):

q Private NNI (P-NNI)

q Public NNI = Inter-Switching System Interface (ISSI)

Intra-LATA ISSI (Regional Bell Operating Co)

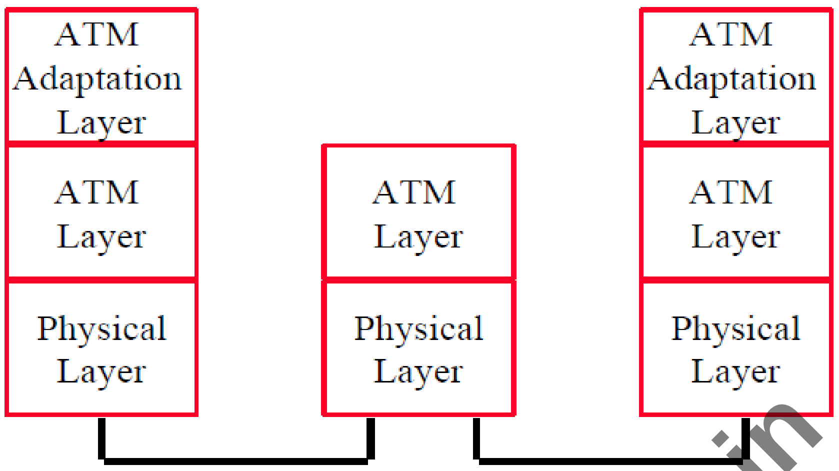
q Inter-LATA ISSI (Inter-exchange Carriers)

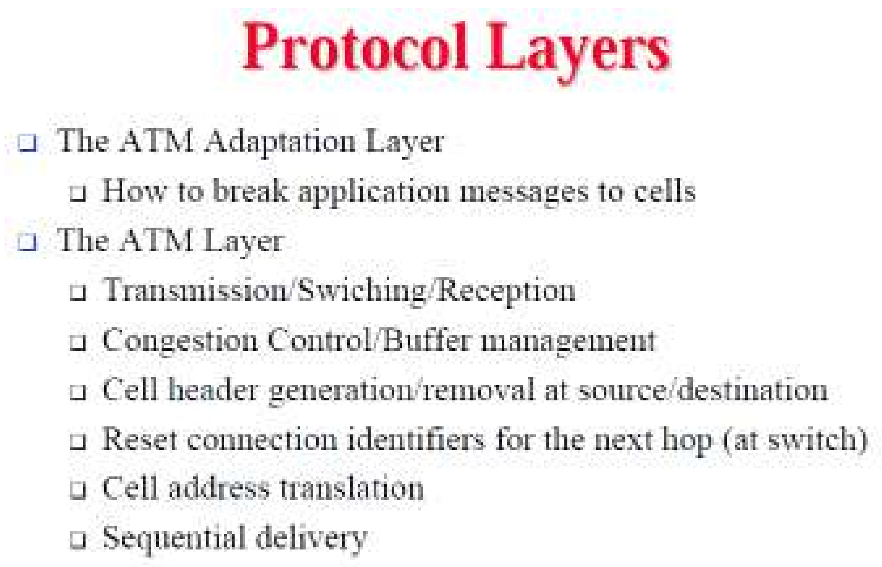
▪ Broadband Inter-Carrier Interface (B-ICI)

q Data Exchange Interface (DXI)

Between packet routers and ATM Digital Service Units (DSU)

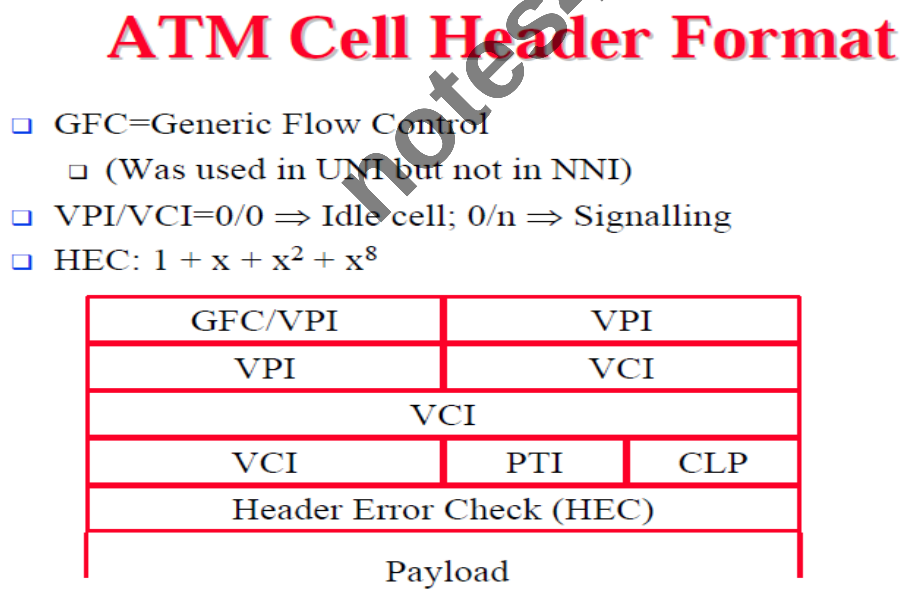
Protocol Layers





Protocol Layers

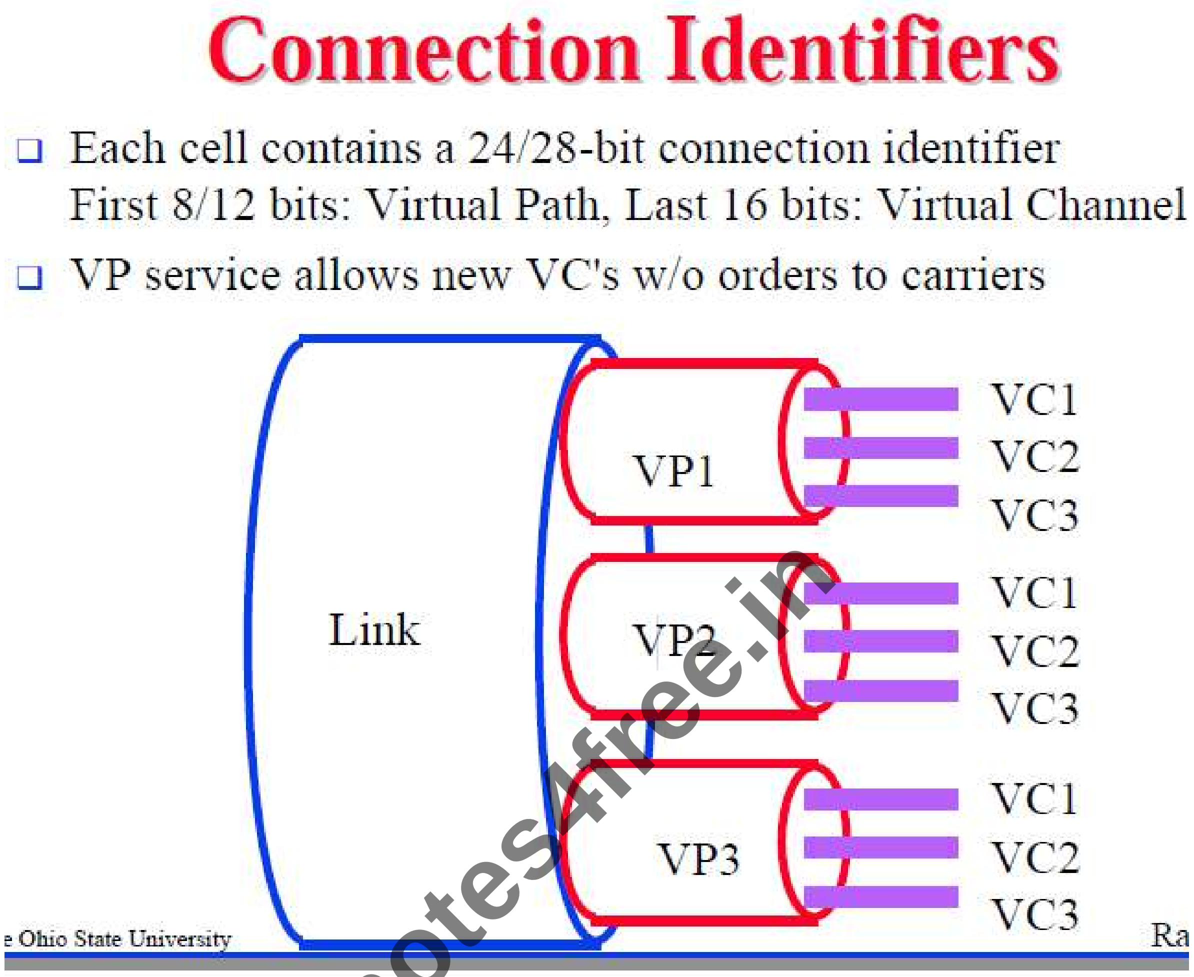
- The ATM Adaptation Layer
  - How to break application messages to cells
- The ATM Layer
  - Transmission/Swiching/Reception
  - Congestion Control/Buffer management
  - Cell header generation/removal at source/destination
  - Reset connection identifiers for the next hop (at switch)
  - Cell address translation
  - Sequential delivery



ATM Cell Header Format

- GFC=Generic Flow Control
  - (Was used in UNI but not in NNI)
- VPI/VCI=0/0 $\Rightarrow$ Idle cell; 0/n $\Rightarrow$ Signalling
- HEC: $1 + x + x^2 + x^8$

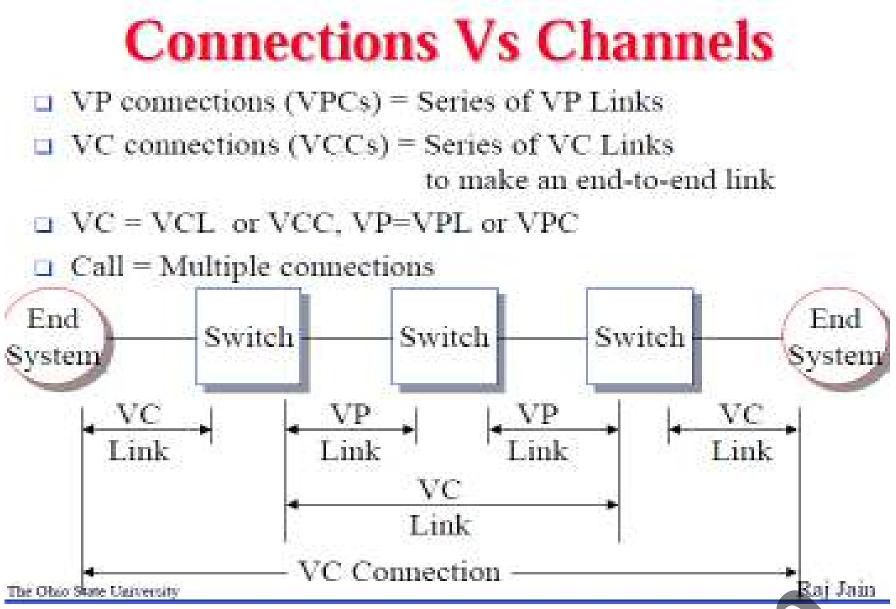
GFC/VPI	VPI	
VPI	VCI	
VCI		
VCI	PTI	CLP
Header Error Check (HEC)		
Payload		



Connection Identifiers

- Each cell contains a 24/28-bit connection identifier
  First 8/12 bits: Virtual Path, Last 16 bits: Virtual Channel
- VP service allows new VC's w/o orders to carriers

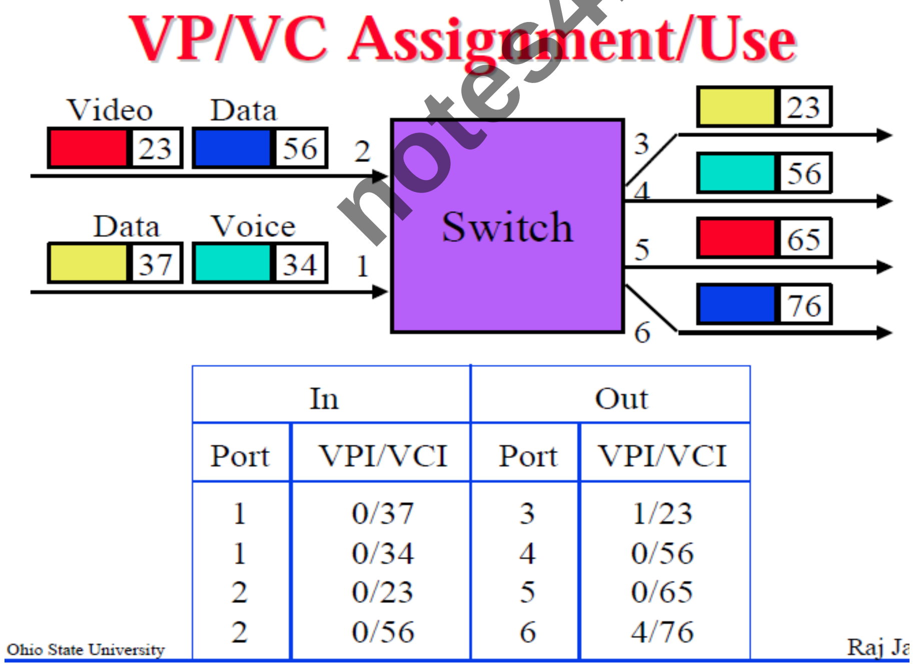
e Ohio State University Ra



Connections Vs Channels

- VP connections (VPCs) = Series of VP Links
- VC connections (VCCs) = Series of VC Links to make an end-to-end link
- VC = VCL or VCC, VP=VPL or VPC
- Call = Multiple connections

The Ohio State University Raj Jain



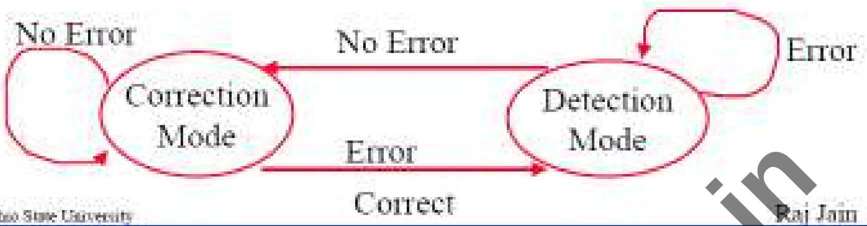
VP/VC Assignment/Use

In		Out	
Port	VPI/VCI	Port	VPI/VCI
1	0/37	3	1/23
1	0/34	4	0/56
2	0/23	5	0/65
2	0/56	6	4/76

Ohio State University Raj Ja

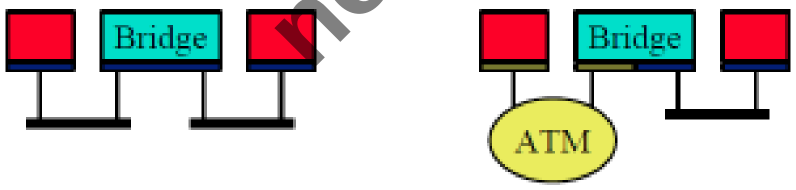
Header Error Check (HEC)

- Protects header only
- Optional Correction mode: Correct one bit errors if no earlier errors
- Discard cells with bad HEC
- Used for cell delineation in SONET
- Recalculated on each hop

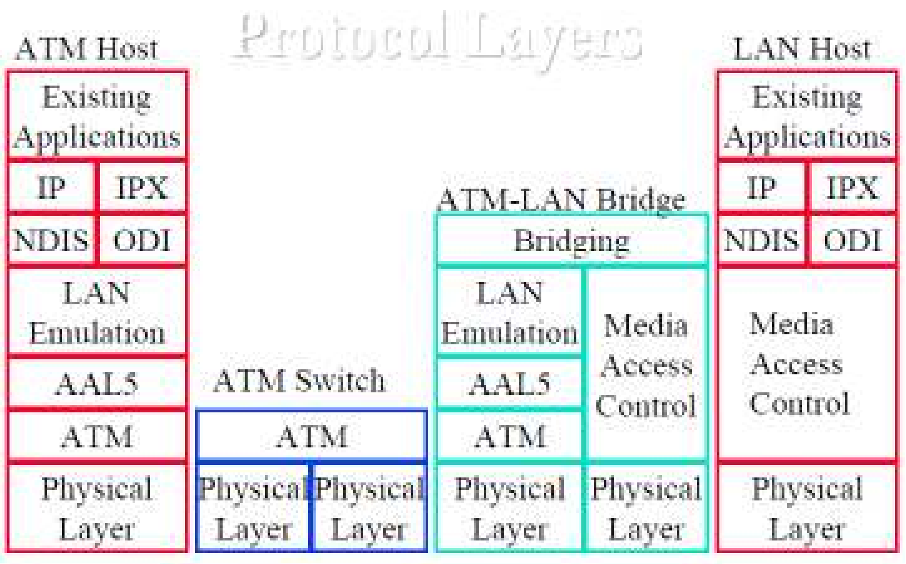


Ohio State University Raj Jain

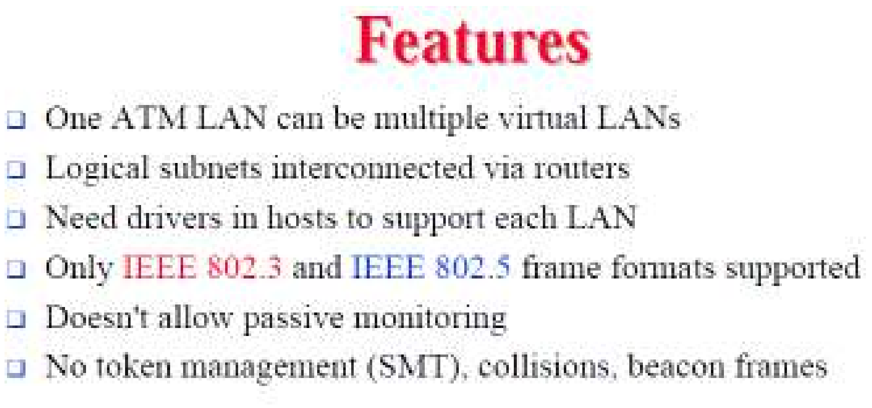
LAN Emulation



- Problem: Need new networking s/w for ATM
- Solution: Let ATM network appear as a virtual LAN
- LAN emulation implemented as a device driver below the network layer



- NDIS = Network Driver Interface Specification
- ODI = Open Datalink Interface



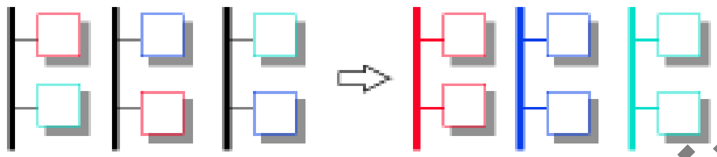
Features

- One ATM LAN can be multiple virtual LANs
- Logical subnets interconnected via routers
- Need drivers in hosts to support each LAN
- Only IEEE 802.3 and IEEE 802.5 frame formats supported
- Doesn't allow passive monitoring
- No token management (SMT), collisions, beacon frames

LE Header (2 Bytes)	Standard IEEE 802.3 or 802.5 Frame

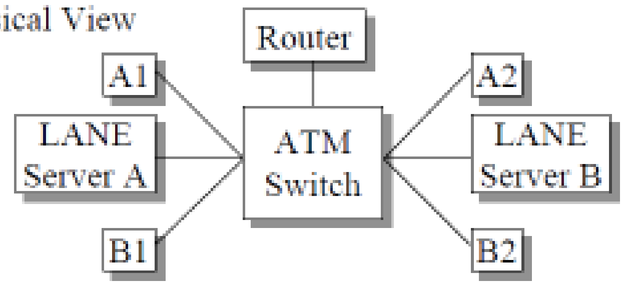
Virtual LANs

- Group of users that appear to be interconnected by one LAN
  One LAN = One broadcast domain
- They may be on physically different LANs
- Stations can be grouped by:
  - All stations that have the same IP subnet address
  - All stations that are connected to the same switch port
  - Stations whose specific addresses are specified

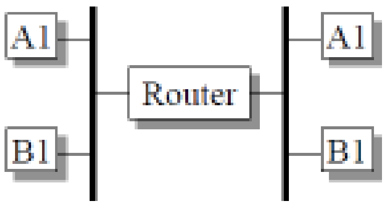


ATM Virtual LANs

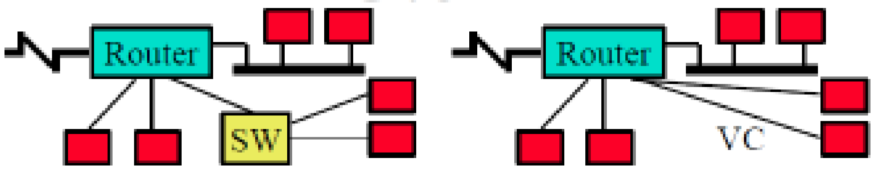
- Physical View



- Logical View

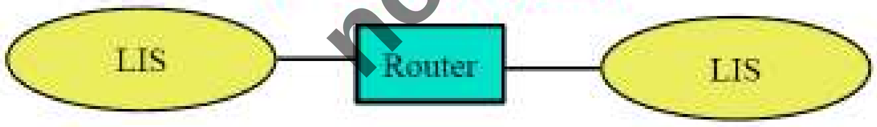


IP Over ATM



- ATM similar to point-to-point WANs.
  Simpler than LAN emulation
- IP address:123.145.134.65
  ATM address:…1-614-999-2345-…
- Issue: IP Address ⇔ ATM Address translation
  - Address Resolution Protocol (ARP)
  - Inverse ATM ARP: VC ⇒ IP Address
- Solution: Logical IP Subnet (LIS) Server
- Ref: RFC 1577

IP Over ATM



- Clients within LIS use direct VCs
- All traffic between LIS passes through a router
- ATM AAL5 PDU size = 9180 + 8 LLC/SNAP header
- Problem: Need router even if ATM connection between LIS
- Solution: Routing Over Large Clouds (ROLC)

ARP Over ATM

- Only one ATM ARP server per subnet
  ⇒ No database synchronization issues
- Clients are configured with server's ATM address
- Clients setup a VC with the server
- Server sends an inverse ARP request
  (What's your IP Address?)
- Client responds with its IP Address
- Clients ask server by ARP request
  (What's ATM address of 123.145.134.65?)
- Server replies with ATM address
- Server sends NAK if not in table
- ARP requests are NOT broadcast to all LIS members

ARP Database Maintenance

- Clients register with the server at startup
- Can use ARP requests to update entry for requester
- Entries at clients age out after 15 minutes
- Entries at servers age out after 20 minutes
- Server sends inverse ARP on active VC before aging out
- Otherwise clients resend registration every 20 minutes

UNIT - 8

TRANSPORT PROTOCOL

Introduction, TCP/IP, TCP, UDP, RTP and RTCP. **6 Hours**

TEXT BOOK:

1. **Multimedia Communications: Applications, Networks, Protocols and Standards**, Fred Halsall, Pearson Education, Asia, Second Indian reprint 2002.

REFERENCE BOOKS:

1. **Multimedia Information Networking**, Nalin K. Sharda, PHI, 2003.
2. **"Multimedia Fundamentals: Vol 1 - Media Coding and Content Processing"**, Ralf Steinmetz, Klara Narstedt, Pearson Education, 2004.
3. **"Multimedia Systems Design"**, Prabhat K. Andleigh, Kiran Thakrar, PHI, 2004.

Journals & Publications:

1. 100. G. Coulson, A. Campbell, and P. Robin, Design of a QoS controlled ATM based communication
2. system in chorus, IEEE J. Selected Areas in Comm., 13, 686–699 (1995).
3. 101. W. Doeringer et al., A survey of light-weight transport protocol for high-speed networks, IEEE Trans Commun., 38, 2025–2039 (1990).
4. 102. D. Ferrari and D. C. Verma, A scheme for real-time channel establishment in wide-area networks, IEEE J. Selected Areas in Comm., 8, 368–377 (1990).
5. 103. J. A. Stukovic et al., Implications of classical scheduling results for real-time systems, IEEE Computer, 28, 16–25 (1995). REFERENCES 721
6. 104. T. D. C. Little and A. Ghafoor, Synchronization properties and storage models for multimedia objects, IEEE J. Selected Areas in Commun., 8, 229–238 (1990).
7. 105. J. F. Kurose, Open issues and challenges in providing quality of service guarantees in high speed networks, ACM Computer Commun. Rev., 23, 6–15 (1993).
8. 106. ISO/IEC JTC1/SC21/WG1 N9680, Quality of Service Framework, UK, 1995.
9. 107. A. T. Campbell et al., Integrated quality of service for multimedia communications, in Proc. IEEE INFOCOM, 732–739, San Francisco, CA, April 1993.
10. 108. A. A. Lazar, A real-time control management and information transport architecture for broadband networks, Proc. Int. Zurich Sem. Digital Communications, 281–295, May 1992.
11. 109. D. Tenkenhouse, Layered multiplexing considered harmful, in Protocols for High-Speed Network, Elsevier Science Publishers, New York, NY, 1990.
12. 110. M. Zilterbart, B. Stiller, and A. Tantewy, A model for flexible high performance communication subsystems, IEEE J. Selected Areas in Commun., 11, 507–518 (1992).
13. 111. D. D. Clark, S. Sheuder, and L. Zhang, Supporting real-time applications in an integrated services packet network: architecture and mechanisms, Proc. ACM SIGCOMM, 17–26, Baltimore, MD, August 1992.
14. 112. R. Gonndan and D. P. Anderson, Scheduling and IPC mechanisms for continuous media, Proc. ACM Symp. Operating Systems Principles, 25, 68–80, Pacific Grove, CA, 1991.
15. 113. A. T. Campbell et al., A continuous media transport and orchestration service, in Proc. ACM SIGCOMM, 99–110, Baltimore, MD, 1992.
16. 114. V. O. K. Li and W. Liao, Distributed multimedia systems, Proc. IEEE, 85, 1063–1108 (1997).
17. 115. ISO/IEC JTC1/SC21/WG1 N9680, Quality of Service Framework, UK, 1995.
18. 116. D. J. Wright, Assessment of alternative transport options for video distribution and retrieval
19. over ATM on residential broadband, IEEE Comm. Magazine, 35, 78–87 (1997).
20. 117. N. Ohta, Packet Video: Modeling and Signal Processing, Artech House, Norwood, MA,
21. 1994.
22. 118. M. Hamidi, J. W. Roberts, and P. Rolin, Rate control for VBR video coders in broadband
23. networks, IEEE J. Selected Areas in Comm., 15, 1040–1051 (1997).
24. 119. R. Steinmetz and K. Nahrstedt, Multimedia Computing, Communications and Applications,
25. Prentice-Hall, Englewood Cliffs, NJ, 1995.
26. 120. ATM Forum, ATM User-Network Interface Specification – Version 3, Mountain View,
27. CA, 1996.
28. 121. N. B. Seitz et al., User-oriented measures of telecommunication quality, IEEE Comm.
29. Magazine, 32, 56–66 (1994).
30. 122. J. Croworft et al., The global internet, IEEE J. Selected Areas in Comm., 13, 1366–1370
31. (1995).
32. 123. IETF RFC1633, Integrated Services in the Internet Architecture: An Overview, ITU,
33. June 1994.
34. 124. ITU-T Recomm. I.356, B-ISDN ATM Layer Cell Transfer Performance, ITU, Geneva,
35. Switzerland, November 1993.
36. 125. F. Guillemin, C. Levert, and C. Rosenberg, Cell conformance testing with respect to the
37. peak cell rate in ATM networks, Computer Networks and ISDN Systems, 27, 703–725
38. (1995).

39. 722 NETWORK LAYER
40. 126. C. J. Gallego and R. Grunenfelder, Testing and measurement problems in ATM networks,
41. Proc. IEEE ICC, 653–657, Dallas, TX, June 1996.
42. 127. J. Tarnet, New directions in communications (or which way to the information age?),
43. IEEE Commun. Magazine, 24, 8–15 (1986).
44. 128. F. Kupers et al., An overview of constraint-based path selection algorithms for QoS routing,
45. IEEE Commun. Magazine, 40, 50–55 (2002).
46. 129. D. Medhi, QoS routing computations with path coding: a framework and network performance,
47. IEEE Commun. Magazine, 40, 106–113 (2002).
48. 130. J. Moy, OS-PF-Anatomy of Internet Routing Protocol, Addison-Wesley, Reading, MA,
49. 1998.

TCP/IP Protocols

This chapter discusses the protocols available in the TCP/IP protocol suite. The following figure shows how they correspond to the 5-layer TCP/IP Reference Model. This is not a perfect one-to-one correspondence; for instance, Internet Protocol (IP) uses the Address Resolution Protocol (ARP), but is shown here at the same layer in the stack.

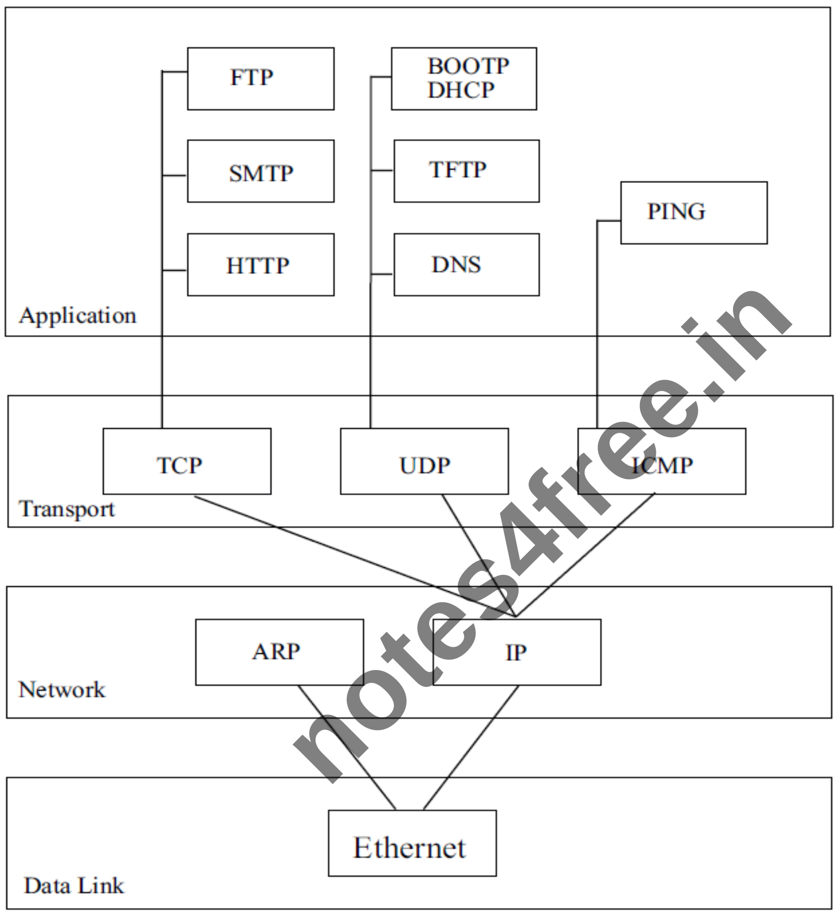


Figure 4. TCP/IP Protocol Flow

IP:

IP provides communication between hosts on different kinds of networks (i.e., different data-link implementations such as Ethenet and Token Ring). It is a connectionless, unreliable packet delivery service. Connectionless means that there is no handshaking, each packet is independent of any other packet. It is unreliable because there is no guarantee that a packet gets delivered; higher level protocols must deal with that.

IP Address

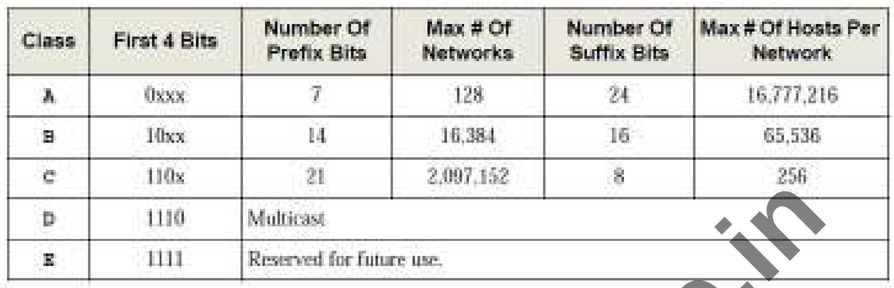
IP defines an addressing scheme that is independent of the underlying physical address (e.g, 48-bit MAC address). IP specifies a unique 32-bit number for each host on a network. This number is known as the Internet Protocol Address, the IP Address or the Internet Address. These terms are interchangeable. Each packet sent across the internet contains the IP address of the source of the packet and the IP address of its destination.

For routing efficiency, the IP address is considered in two parts: the prefix which identifies the physical network, and the suffix which identifies a computer on the network. A unique prefix is needed for each network in an internet. For the global Internet, network numbers are obtained from Internet Service Providers (ISPs). ISPs coordinate with a central organization called the Internet Assigned Number Authority (IANA).

IP Address Classes

The first four bits of an IP address determine the class of the network. The class specifies how many of the remaining bits belong to the prefix (aka Network ID) and to the suffix (aka Host ID).

The first three classes, A, B and C, are the primary network classes.



Class	First 4 Bits	Number Of Prefix Bits	Max # Of Networks	Number Of Suffix Bits	Max # Of Hosts Per Network
A	0xxx	7	128	24	16,777,216
B	10xx	14	16,384	16	65,536
C	110x	21	2,097,152	8	256
D	1110	Multicast			
E	1111	Reserved for future use.			

When interacting with mere humans, software uses dotted decimal notation; each 8 bits is treated as an unsigned binary integer separated by periods. IP reserves host address 0 to denote a network. 140.211.0.0 denotes the network that was assigned the class B prefix 140.211.

Netmasks

Netmasks are used to identify which part of the address is the Network ID and which part is the Host ID. This is done by a logical bitwise-AND of the IP address and the netmask. For class A networks the netmask is always 255.0.0.0; for class B networks it is 255.255.0.0 and for class C networks the netmask is 255.255.255.0.

Subnet Address

All hosts are required to support subnet addressing. While the IP address classes are the convention, IP addresses are typically subnetted to smaller address sets that do not match the class system.

The suffix bits are divided into a subnet ID and a host ID. This makes sense for class A and B networks, since no one attaches as many hosts to these networks as is allowed. Whether to subnet and how many bits to use for the subnet ID is determined by the local network administrator of each network.

If subnetting is used, then the netmask will have to reflect this fact. On a class B network with subnetting, the netmask would not be 255.255.0.0. The bits of the Host ID that were used for the subnet would need to be set in the netmask.

Directed Broadcast Address

IP defines a directed broadcast address for each physical network as all ones in the host ID part of the address. The network ID and the subnet ID must be valid network and subnet values. When a packet is sent to a network's broadcast address, a single copy travels to the network, and then the packet is sent to every host on that network or subnetwork.

Limited Broadcast Address

If the IP address is all ones (255.255.255.255), this is a limited broadcast address; the packet is addressed to all hosts on the current (sub)network. A router will not forward this type of broadcast to other (sub)networks.

5.2 IP Routing

Each IP datagram travels from its source to its destination by means of routers. All hosts and routers on an internet contain IP protocol software and use a routing table to determine where to send a packet next. The destination IP address in the IP header contains the ultimate destination of the IP datagram, but it might go through several other IP addresses (routers) before reaching that destination.

Routing table entries are created when TCP/IP initializes. The entries can be updated manually by a network administrator or automatically by employing a routing protocol such as Routing Information

Protocol (RIP). Routing table entries provide needed information to each local host regarding how to communicate with remote networks and hosts.

When IP receives a packet from a higher-level protocol, like TCP or UDP, the routing table is searched for the route that is the closest match to the destination IP address. The most specific to the least specific route is in the following order:

• A route that matches the destination IP address (host route).

• A route that matches the network ID of the destination IP address (network route).

• The default route.

If a matching route is not found, IP discards the datagram.
IP provides several other services:

• **Fragmentation**: IP packets may be divided into smaller packets. This permits a large packet to travel across a network which only accepts smaller packets. IP fragments and reassembles packets transparent to the higher layers.

• **Timeouts**: Each IP packet has a Time To Live (TTL) field, that is decremented every time a packet moves through a router. If TTL reaches zero, the packet is discarded.

• **Options**: IP allows a packet's sender to set requirements on the path the packet takes through the network (source route); the route taken by a packet may be traced (record route) and packets may be labeled with security features.

ARP

The Address Resolution Protocol is used to translate virtual addresses to physical ones. The network hardware does not understand the software-maintained IP addresses. IP uses ARP to translate the 32-bit IP address to a physical address that matches the addressing scheme of the underlying hardware (for Ethernet, the 48-bit MAC address).

There are three general addressing strategies:

1. Table lookup
2. Translation performed by a mathematical function
3. Message exchange

TCP/IP can use any of the three. ARP employs the third strategy, message exchange. ARP defines a request and a response. A request message is placed in a hardware frame (e.g., an Ethernet frame), and broadcast to all computers on the network. Only the computer whose IP address matches the request sends a response.

The Transport Layer

There are two primary transport layer protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). They provide end-to-end communication services for applications.

UDP

This is a minimal service over IP, adding only optional checksumming of data and multiplexing by port number. UDP is often used by applications that need multicast or broadcast delivery, services not offered by TCP. Like IP, UDP is connectionless and works with datagrams.

TCP

TCP is a connection-oriented transport service; it provides end-to-end reliability, resequencing, and flow control. TCP enables two hosts to establish a connection and exchange streams of data, which are treated in bytes. The delivery of data in the proper order is guaranteed.

TCP can detect errors or lost data and can trigger retransmission until the data is received, complete and without errors.

TCP Connection/Socket

A TCP connection is done with a 3-way handshake between a client and a server. The following is a simplified explanation of this process.

- The client asks for a connection by sending a TCP segment with the SYN control bit set.
- The server responds with its own SYN segment that includes identifying information that was sent by the client in the initial SYN segment.
- The client acknowledges the server's SYN segment.

The connection is then established and is uniquely identified by a 4-tuple called a *socket* or *socket pair*:

(destination IP address, destination port number)

(source IP address, source port number)

During the connection setup phase, these values are entered in a table and saved for the duration of the connection.

TCP Header

Every TCP segment has a header. The header comprises all necessary information for reliable, complete delivery of data. Among other things, such as IP addresses, the header contains the following fields:

Sequence Number - This 32-bit number contains either the sequence number of the first byte of data in this particular segment or the Initial Sequence Number (ISN) that identifies the first byte of data that will be sent for this particular connection.

The ISN is sent during the connection setup phase by setting the SYN control bit. An ISN is chosen by both client and server. The first byte of data sent by either side will be identified by the sequence number ISN + 1 because the SYN control bit consumes a sequence number. The following figure illustrates the three-way handshake.

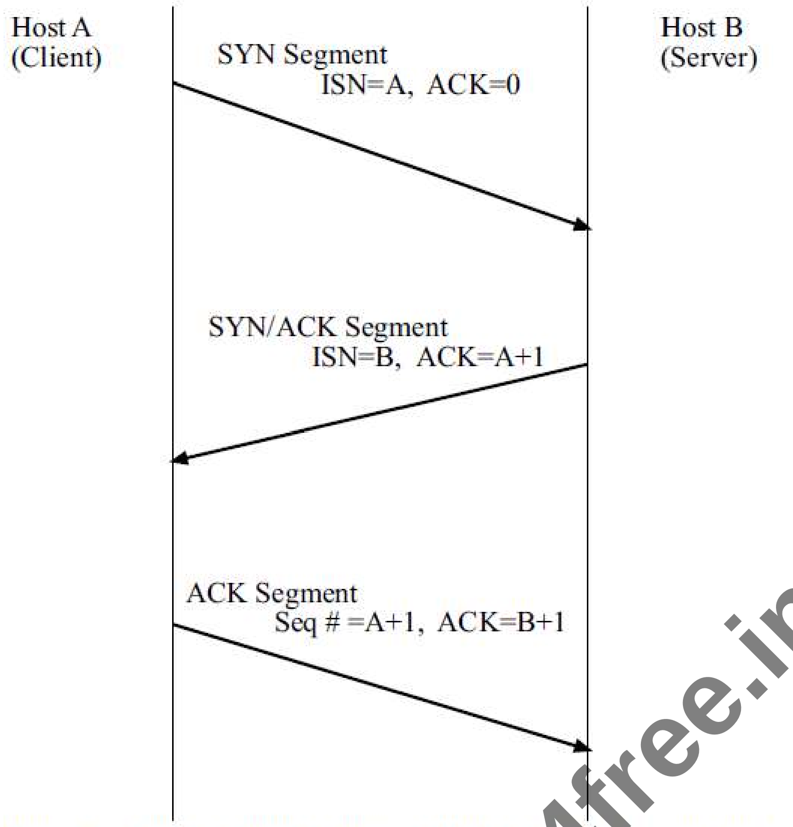


Figure 5. Synchronizing Sequence Numbers for TCP Connection

The sequence number is used to ensure the data is reassembled in the proper order before being passed to an application protocol.

Acknowledgement Number - This 32-bit number is the other host's sequence number + 1 of the last successfully received byte of data. It is the sequence number of the next expected byte of data. This field is only valid when the ACK control bit is set. Since sending an ACK costs nothing, (because it and the Acknowledgement Number field are part of the header) the ACK control bit is always set after a connection has been established. The Acknowledgement Number ensures that the TCP segment arrived at its destination.

Control Bits - This 6-bit field comprises the following 1-bit flags (left to right):

- URG - Makes the Urgent Pointer field significant.
- ACK - Makes the Acknowledgement Number field significant.
- PSH - The Push Function causes TCP to promptly deliver data.
- RST - Reset the connection.
- SYN - Synchronize sequence numbers.
- FIN - No more data from sender, but can still receive data.

Window Size - This 16-bit number states how much data the receiving end of the TCP connection will allow. The sending end of the TCP connection must stop and wait for an acknowledgement after it has sent the amount of data allowed.

Checksum - This 16-bit number is the one's complement of the one's complement sum of all bytes in the TCP header, any data that is in the segment and part of the IP packet. A checksum can only detect some errors, not all, and cannot correct any.

ICMP

Internet Control Message Protocol is a set of messages that communicate errors and other conditions that require attention. ICMP messages, delivered in IP datagrams, are usually acted on by either IP, TCP or UDP. Some ICMP messages are returned to application protocols.

A common use of ICMP is "pinging" a host. The Ping command (Packet INternet Groper) is a utility that determines whether a specific IP address is accessible. It sends an ICMP echo request and waits for a reply. Ping can be used to transmit a series of packets to measure average roundtrip times and packet loss percentages.

The Application Layer

There are many applications available in the TCP/IP suite of protocols. Some of the most useful ones are for sending mail (SMTP), transferring files (FTP), and displaying web pages (HTTP).

These applications are discussed in detail in the *TCP/IP User's Manual*.

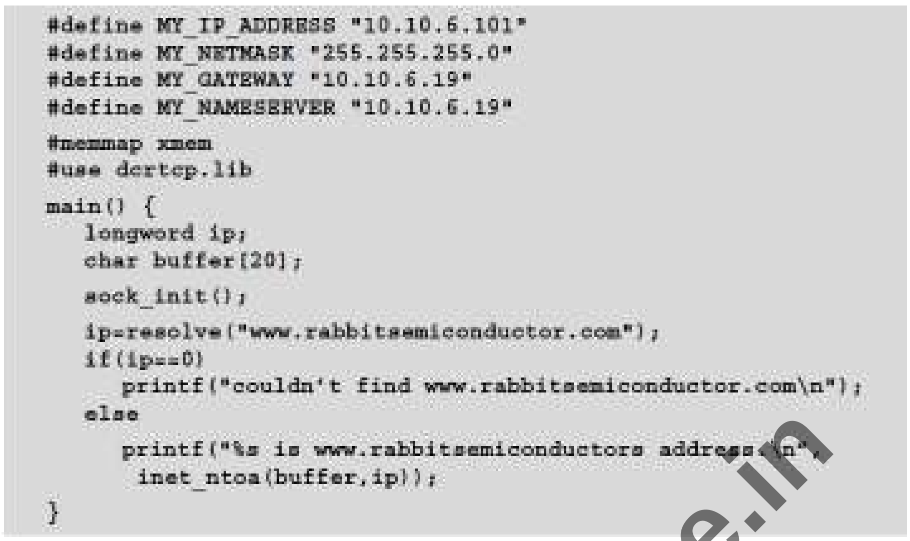
Another important application layer protocol is the Domain Name System (DNS). Domain names are significant because they guide users to where they want to go on the Internet.

DNS

The Domain Name System is a distributed database of domain name and IP address bindings. A domain name is simply an alphanumeric character string separated into segments by periods. It represents a specific and unique place in the "domain name space." DNS makes it possible for us to use identifiers such as zworld.com to refer to an IP address on the Internet. Name servers contain information on some segment of the DNS and make that information available to clients who are called resolvers.

DCRTCP.LIB Implementation of DNS

The **resolve()** function in **DCRTCP.LIB** immediately converts a dotted decimal IP address to its corresponding binary IP address and returns this value. If **resolve()** is passed a domain name, a series of queries take place between the computer that called **resolve()** and computers running name server software. For example, to resolve the domain name www.rabbitsemiconductor.com, the following code (available in **SAMPLES\ TCP\DNS.C**) can be used.



```
#define MY_IP_ADDRESS "10.10.6.101"
#define MY_NETMASK "255.255.255.0"
#define MY_GATEWAY "10.10.6.19"
#define MY_NAMESERVER "10.10.6.19"
#memmap xmem
#use dcrtcp.lib
main() {
   longword ip;
   char buffer[20];
   sock_init();
   ip=resolve("www.rabbitsemiconductor.com");
   if(ip==0)
      printf("couldn't find www.rabbitsemiconductor.com\n");
   else
      printf("%s is www.rabbitsemiconductors address.\n",
       inet_ntoa(buffer,ip));
}
```

Your local name server is specified by the configuration macro **MY_NAMESERVER**. Chances are that your local name server does not have the requested information, so it queries the root server.

The root server will not know the IP address either, but it will know where to find the name server that contains authoritative information for the .com zone. This information is returned to your local name server, which then sends a query to the name server for the .com zone. Again, this name server does not know the requested IP address, but does know the local name server that handles rabbitsemiconductor.com. This information is sent back to your local name server, who sends a final query to the local name server of rabbitsemiconductor.com. This local name server returns the requested IP address of www.rabbitsemiconductor.com to your local name server, who then passes it to your computer.

Introduction: What are RTP and RTCP?

The spread of computers, added to the availability of cheap audio/video computer hardware, and the availability of higher connection speeds have increased interest in using the Internet for sending audio and video, data types which were traditionally reserved for specialist networks, and over several years audio and video conferencing have become common practice. However, the very nature of the Internet means that this network is not suited for data transmission in real time and as a result the quality of audio sent over the Internet is usually of mediocre quality.

This theory specifically addresses the analysis and solution of these problems to enable an audio conferencing or telephone application over the Internet to change its behaviour to maintain an acceptable auditory quality even in cases where the network is quite congested. These solutions, in the form of control mechanisms, have been implemented and tested on the audio conferencing and telephone software on Internet Free Phone which we developed. A study on the effect that these machines would have on an Internet which developed to integrate the Fair Queuing service discipline has shown that while these mechanisms would still be necessary, they would perform even better on this network type.

RTP (Real-time Transport Protocol)

The aim of RTP is to provide a uniform means of transmitting data subject to real time constraints over IP (audio, video, etc.). The principal role of RTP is to implement the sequence numbers of IP packets to reform voice or video information even if the underlying network changes the order of the packets. More generally, RTP makes it possible to:

- identify the type of information carried,
- add temporary markers and sequence numbers to the information carried,

- monitor the packets' arrival at the destination.

In addition, RTP may be conveyed by multicast packets in order to route conversations to multiple recipients.

RTCP (Real-time Transport Control Protocol)

RTCP protocol is based on periodic transmissions of control packets by all participants in the session.

It is a control protocol for RTP flow, making it possible to convey basic information on the participants of a session and the quality of service.

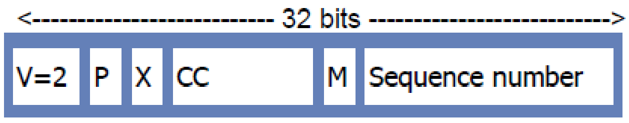
Planned use of RTP and RTCP

RTP allows the management of multimedia flows (voice, video) over IP. RTP works on UDP. The RTP header carries synchronisation and numbering information. The data coding will depend on the compression type. RFCxxxx specifies RTP, on the other hand the adaptation of a compression method to RTP will be described in a specific RFC, for example H261 on RTP is described in RFCxxxx. One RTP channel is used per type of flow: one for audio, one for video. The field xxx is used for synchronisation.

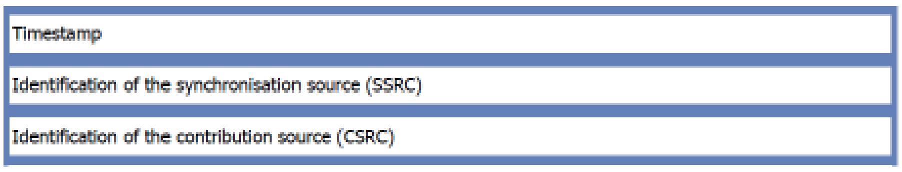
RTP offers an end to end service. It adds a header which provides timing information necessary for the synchronisation of sound and video type real time flow. RTP (Real time Transport Protocol) and its companion RTCP (Real time Transport Control Protocol) make it possible to respectively transport and monitor data blocks which have real time properties. RTP and RTCP are protocols which are located at application level and use underlying TCP or UDP transport protocols. But the use of RTP/RTCP is generally done above UDP. RTP and RTCP

can use the Unicast (point to point) method just as well as the Multicast (multipoint) method. Each of them uses a separate port from a pair of ports. RTP uses the even port and RTCP the next highest odd port **Format of headers and their contents**

The RTP header carries the following information:



<---------------------------- 32 bits ---------------------------->
V=2
P
X
CC
M
Sequence number



Timestamp
Identification of the synchronisation source (SSRC)
Identification of the contribution source (CSRC)

Here are the meanings of the different header fields:

- **Version field V** 2 bits long indicates the protocol version (V=2)
- **Padding field P**: 1 bit, if P is equal to 1, the packet contains additional bytes for padding to finish the last packet.
- **Extension field X**: 1 bit, if X=1 the header is followed by a extension packet
- **CSRC count field CC**: 4 bits, contains the number of CSRC which follow the header
- **Marker field M**: 1 bit, its interpretation is defined by an application profile
- **Payload type field PT**: 7 bits, this field identifies the payload type (audio, video, image, text, html, etc.)
- **Sequence number field**: 16 bits, its initial value is random and it increments by 1 for each packet sent, it can be used to detect lost packets
- **Timestamp field**: 32 bits, reflects the moment when the first byte of the RTP packet has been sampled. This instant must be taken from a clock which increases in a monotonous and linear way in time to enable synchronisation and the calculation of the jitter at the destination.
- **SSRC field**: 32 bits uniquely identify the source, its value is chosen randomly by the application. The SSRC identifies the synchronisation source (simply called "the source"). This identifier is chosen randomly with the intent that it is unique among all the sources of the same session. The list of CSRC identifies the sources (SSRC) which have contributed to obtaining the data contained in the packet which contains

these identifiers. The number of identifiers is given in the CC field.

- **CSRC field**: 32 bits, identifies contributing sources.

RTCP headers

The objective of RTCP is to provide different types of information and a return regarding the quality of reception.

The RTCP header carries the following information:

- **Version field** (2 bits):
- **Padding field** (1 bit) indicates that there is padding the size of which is indicated in the last byte
- **Reception report count field** (5 bits): number of reports in the packet
- **Packet type field** (8 bits) 200 for SR
- **Length field** (16 bits) packet length in 32 bit words
- **SSRC field** (32 bits): identification of the specific originator source
- **NTP timestamp field** (64 bits)
- **RTP timestamp field** (32 bits)
- **Sender's packet count field** (32 bits)

Sender's packet byte field (32 bits) statistics

- **SSRC-n field** (32 bits) number of the source whose flow is analysed
- **Fraction lost field** (8 bits)
- **Cumulative number of packets lost field** (24 bits)
- **Extended highest sequence number received field** (32 bits)
- **Interarrival jitter field** (32 bits). This is an estimation of the time interval for an RTP data packet which is measured with the timestamp and which is in the form of a whole number. This is in fact the relative transit time between two data packets. The formula for calculating it is: $J=J+(|D(i-1,i)|-J)/16$

The interarrival jitter is calculated for each data packet received by the source SSRC_n

i -->First packet

i-1 --> previous packet

D --> difference

J --> second packet

- **Last SR timestamp field** (32 bits)
- **Delay since last SR timestamp field** (32 bits)

How is RTCP used with regards RTP?
RTCP is a control protocol associated with RTP, it measures performances but offers no guarantees. To do so, it must use a reservation protocol such as RSVP or make sure that the communication links used are correctly proportioned in relation to the use which is made of it.

RTP and RTCP operate above which protocols?
RTP/RTCP is above the UDP/TCP transport, but practically above UDP. RTP is a session protocol, but it is placed in the application. It is for the developer to integrate it.

How is the type of flow transported?
RTP has nothing to do with the type of flow, it is above UDP, which itself is above IP. The type of flow is theoretically used in IP. RTP carries a sequence number, timestamp and unique identifier for the source (SSRC)

Question Bank

1. Define Mutimedia.?

2. Design a Huffman code for a source that puts out letters from an alphabet A={a1,a2,a3,a4,a5}with P(a1)=P(a3)=0.2,P(a2)=0.4, and P(a4)=P(a5)=0.1 .

3. For the Q1,Find the entropy for this source and the average length for this code **?**

4. Design a Arithmetic code for a three letter alphabet A={a1,a2,a3} with P(a1)=0.7,P(a2)=0.1, and P(a3)=0.2.Encode input sequence a1a2a3

. Explain with a neat diagram Base line mode of JPEG

6. Explain MIDI standards in detail including Hardware aspects and MIDI messages

7. Explain PCM and DPCM with neat block diagrams. Explain Delta Modulation

8. Explain Non Uniform Quantization. Explain the two algorithms to achieve non uniform quantization

9. Define and Explain the coding using Discrete Cosine Transform and Karhunen Loeve Transformation

10. Write short notes on LZW algorithm

11. Write short note on Run Length Coding

12. Explain Zero tree data structure and successive approximation quantization.

13. Write a note on JPEG-LS standard.

14. Explain neatly the concept of padding in reference to VOP in the context of MPEG4 encoding.

15. Explain with a neat diagram the six hierarchical layers for the bitstream of an MPEG-1 video.

16. Explain the main steps of JPEG2000 Image Compression.

17. Write a note on motion compensation in MPEG-4 Standard.

18. Explain sprite coding in MPEG-4 standard.

19. Write short note on H.263.

20. Write short note on MPEG 7.

21. Draw the block diagram of MPEG-2 video encoder for SNR scalability and explain briefly.

22. What are some of the enhancements of MPEG-2, compared with MPEG-1 ?

23. Why hasn't the MPEG-2 standard superseded the MPEG-1 standard ?

24. Explain Dithering with an example. What is Ordered Dithering ?

25. What are Popular Image formats ?

26. Explain Raster data and Interlacing concept

27. Explain HDTV(High definition television)

28. What are the differences between ordinary TV and HDTV(High definition television)

29. What are different types of Video signals.

30. What are the advantage of Interlacing.

31. Mention briefly about Signal to quantization ratio.

32. What does Weber's law states?

33. Explain Median cut Algorithm.

34. Write short note on Synthetic sounds

35. Write short note on A-law and mu –law

36. Write short note on Color LUT's

37. Write short note on PAL video

38. Write short note on HTTP

39. Write short note on HTML

40. Write short note on XML

41. Write short note on SMIL.