Multimedia Networking

#3 Multimedia Networking

Semester Ganjil 2012

PTIIK Universitas Brawijaya

Schedule of Class Meeting

1. Introduction
2. Applications of MN
3. **Requirements of MN**
4. **Coding and Compression**
5. RTP
6. IP Multicast
7. IP Multicast (cont'd)
8. Overlay Multicast
9. CDN: Solutions
10. CDN: Case Studies
11. QoS on the Internet: Constraints
12. QoS on the Internet: Solutions
13. Discussion
14. Summary

Today's Outline

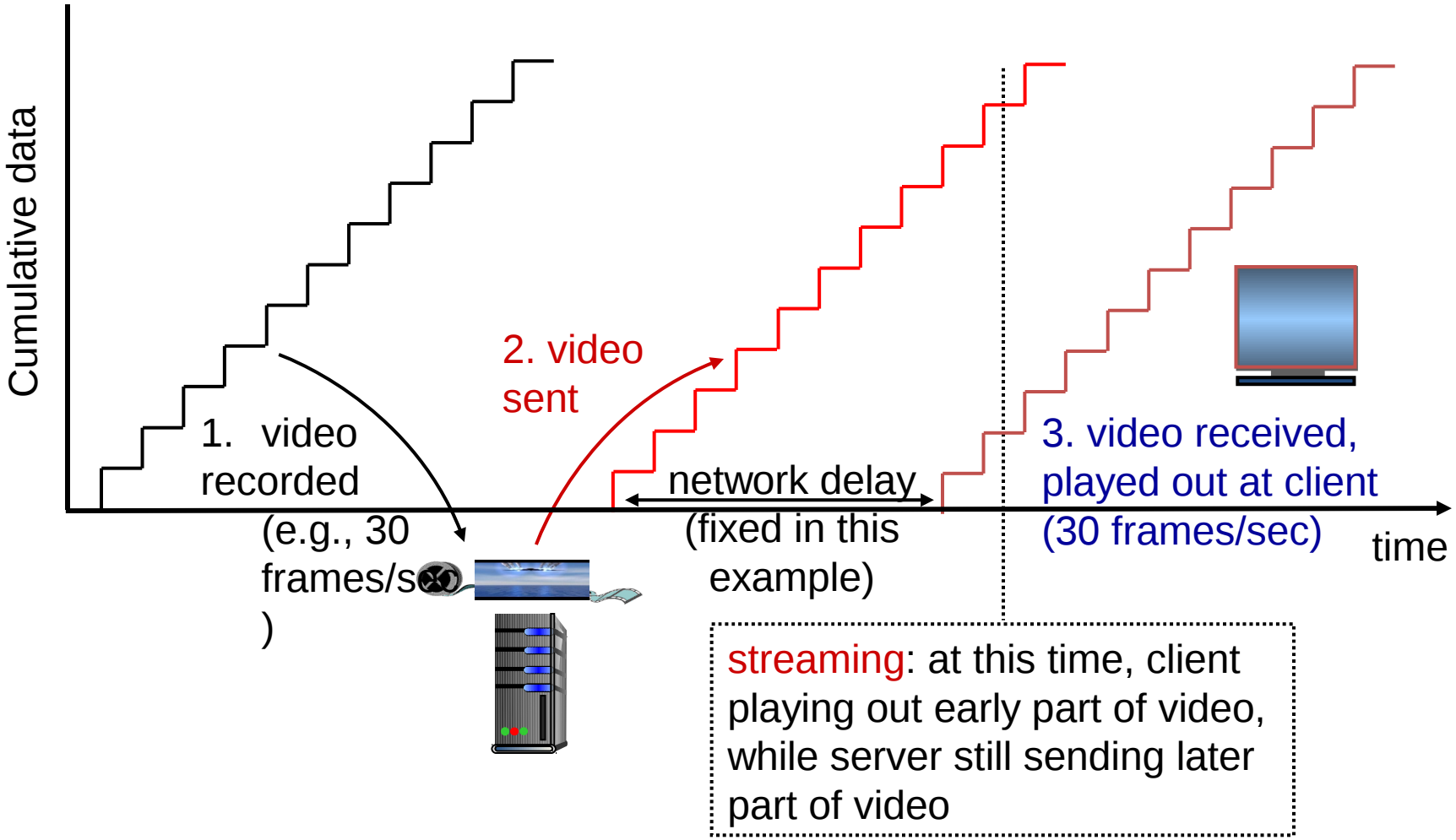
Requirements of Multimedia Networking

- **Client-side buffering**
- Removing Jitter
- Recovering from Packet Loss

Coding and Compression

- Digital Audio
- Digital Video

Streaming stored video:



Streaming stored video: systems

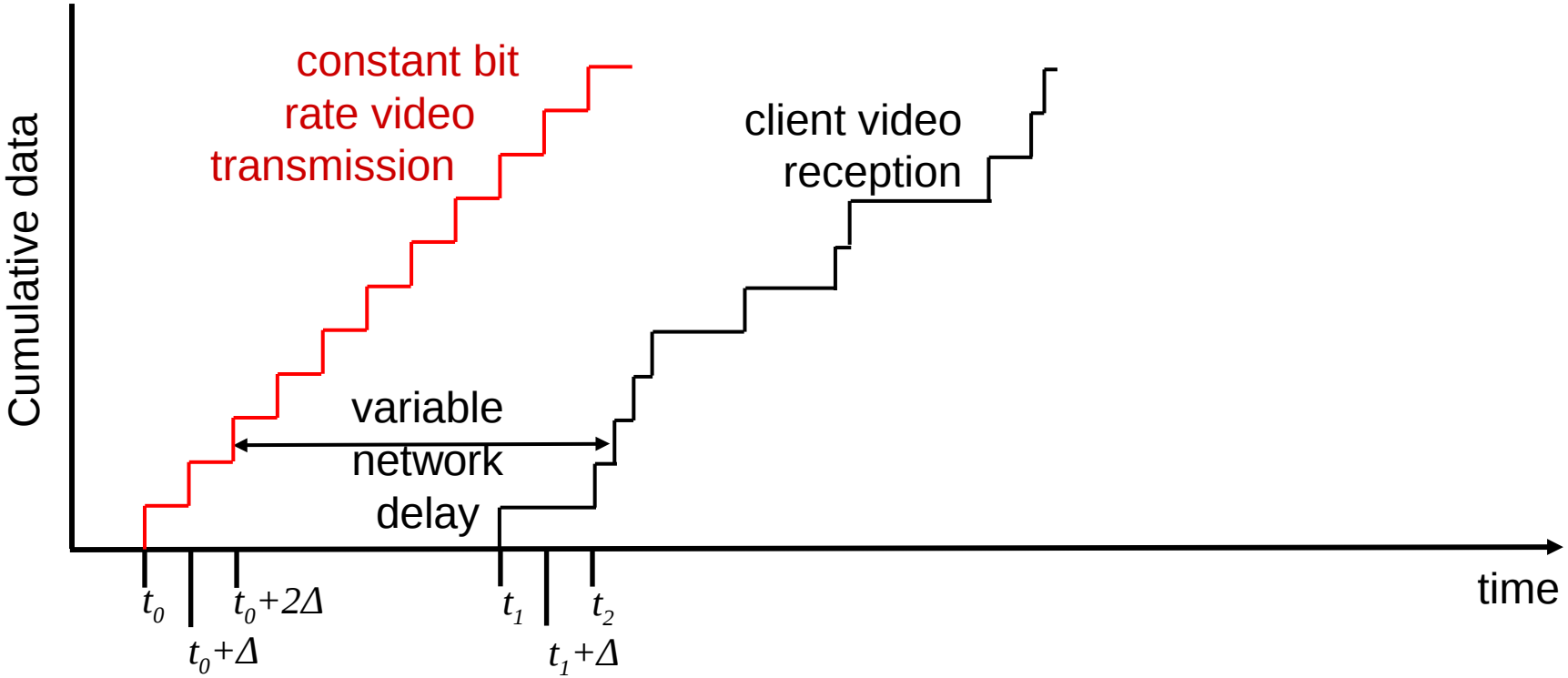
- UDP Streaming
  - unpredictable available bandwidth → constant-rate streaming can fail
  - requires a media control server → interactivity
  - many firewalls block UDP traffic
- HTTP Streaming & Adaptive Streaming
  - TCP congestion control & retransmission → delay playout
  - client-side buffering & prefetching → smooth playout

- Majority of today's systems employ HTTP streaming and adaptive HTTP streaming

Streaming stored video: revisted

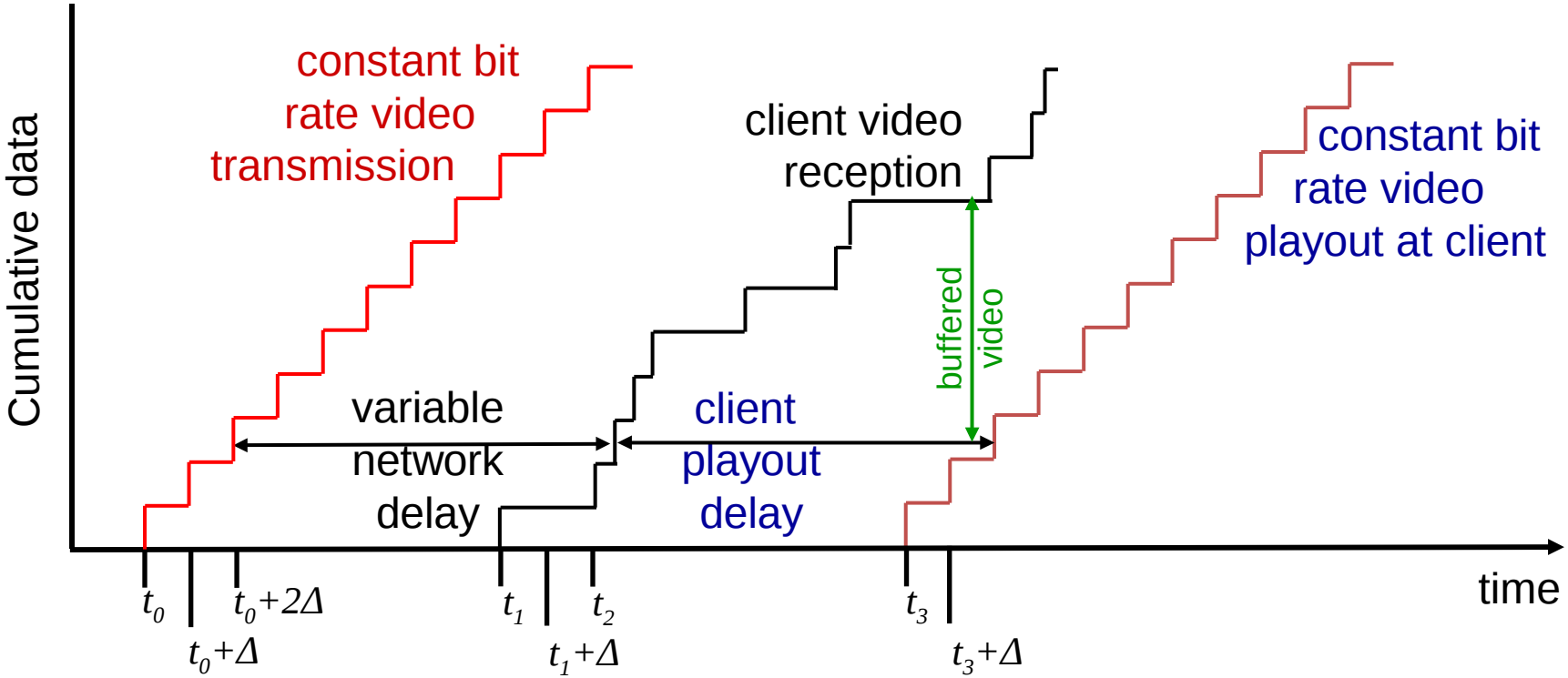
- Δ = fixed time, each video block played out time

Streaming stored video: revisted



- variable end-to-end delays, each video block may experience different delays

Streaming stored video: revisted



- Client-side buffering is used to mitigate:
  - varying end-to-end delays
  - varying available bandwidth

Today's Outline

Requirements of Multimedia Networking

- Client-side buffering
- **Removing Jitter**
- Recovering from Packet Loss

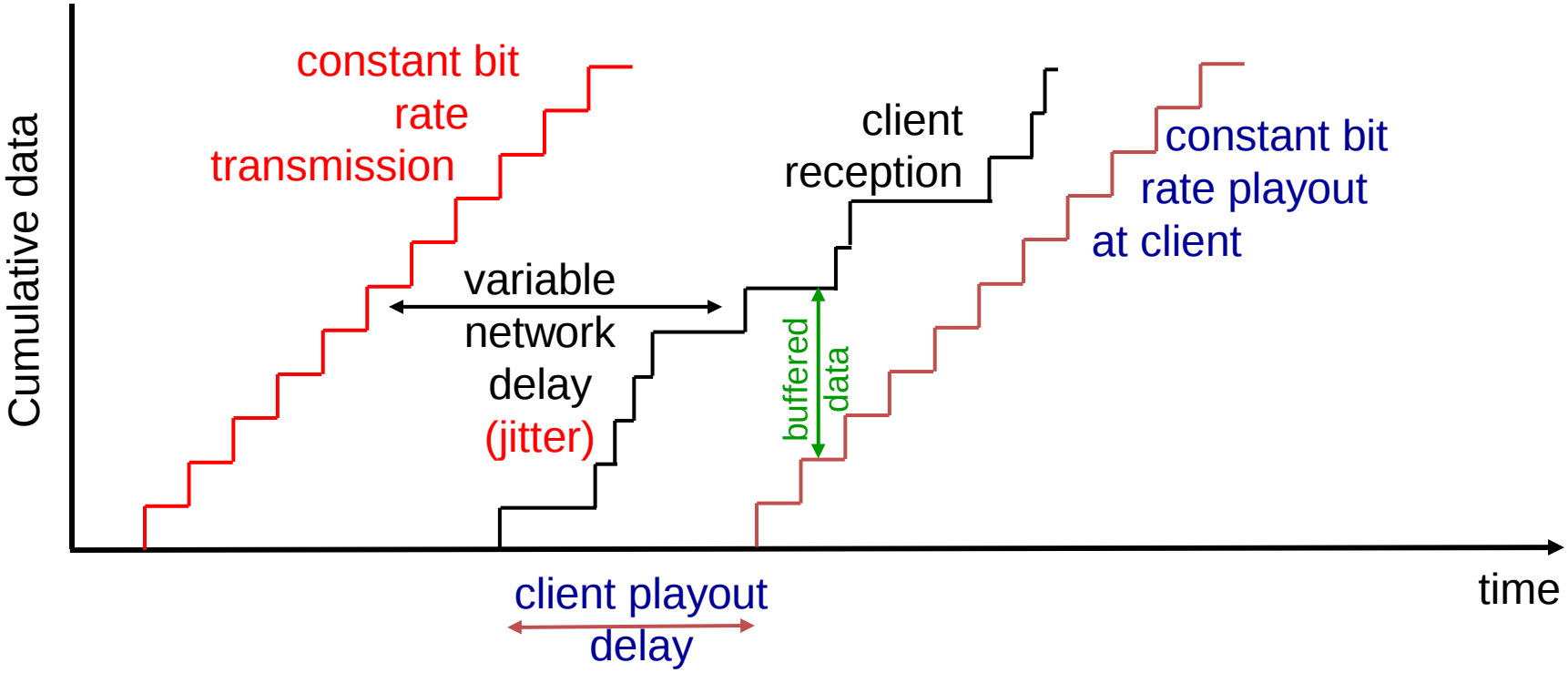
Coding and Compression

- Digital Audio
- Digital Video

Removing Jitter in VoIP

- Jitter can be removed using:
  - prepend each chunk with *sequence number* & *timestamp*,
  - *delaying playout*
- VoIP characteristics:
  - 64 kbps during talk spurt
  - pkts generated only during talk spurts
  - 20 msec chunks at 8 Kbytes/sec: 160 bytes of data
  - typical maximum tolerable delay: 400 ms

Delay ~~jitter~~



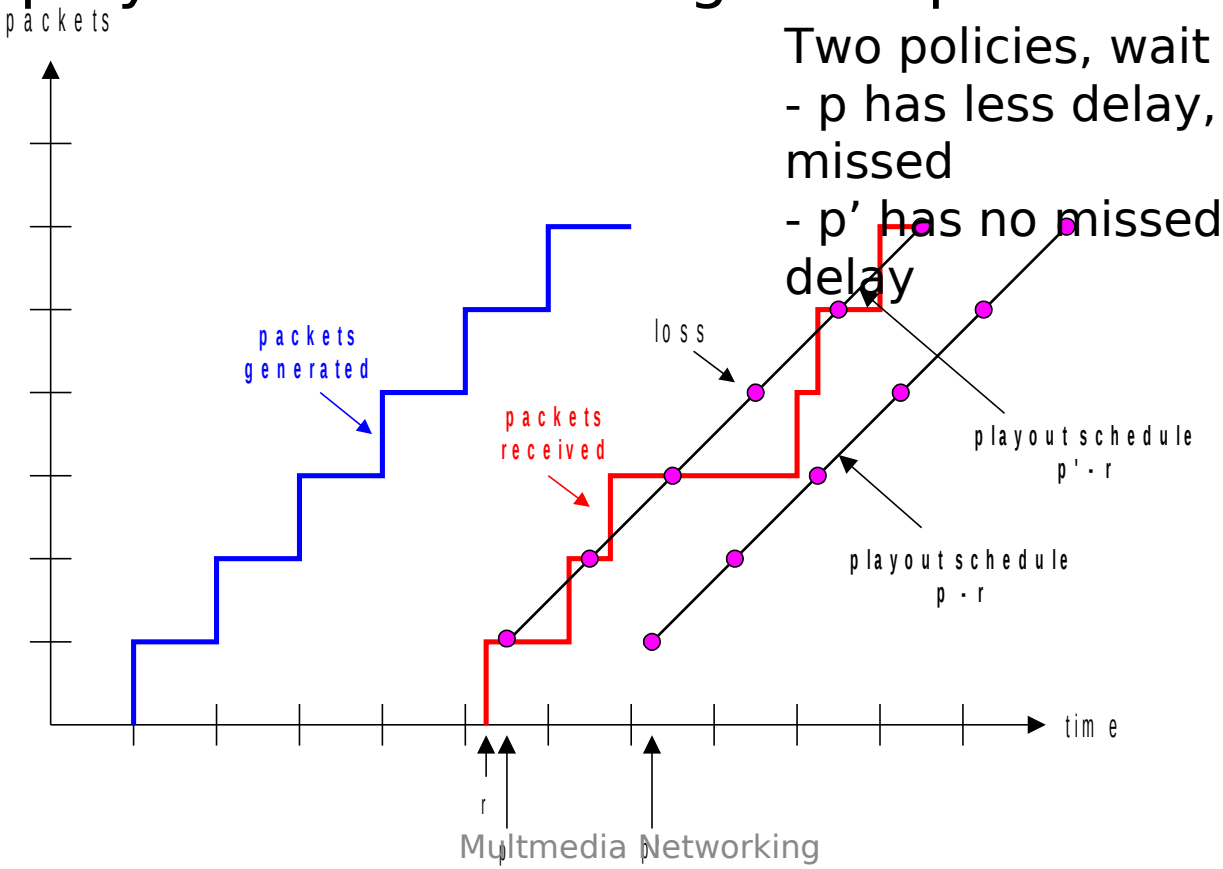
- end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

VoIP: fixed playout delay

- receiver attempts to playout each chunk exactly *q* msecs after chunk was generated.
  - chunk has time stamp *t:* play out chunk at *t*+*q*
  - chunk arrives after *t*+*q*: data arrives too late for playout: data "lost"
- tradeoff in choosing *q*:
  - *large q:* less packet loss
  - *small q:* better interactive experience

VoIP: fixed playout delay

- sender generates packets every 20 msec during talk spu
- first packet received at time r
- first playout schedule: begins at p
- second playout schedule: begins at p’



Two policies, wait p or wait p’
- p has less delay, but one missed
- p’ has no missed, but higher delay

Adaptive playout delay (1)

- *goal:* low playout delay, low late loss rate
- *approach:* adaptive playout delay adjustment:
  - estimate network delay, adjust playout delay at beginning of each talk spurt
  - silent periods compressed and elongated
  - chunks still played out every 20 msec during talk spurt
- adaptively estimate packet delay: (EWMA - exponentially weighted moving average, recall TCP RTT estimate):

$$d_i = (1-\alpha)d_{i-1} + \alpha(r_i - t_i)$$

delay estimate after ith packet

small constant, e.g. 0.1

time received - time sent (timestamp)

measured end-to-end delay of ith packet

Adaptive playout delay (2)

also useful to estimate average deviation of delay, v

$$v_i = (1-\beta)v_{i-1} + \beta\ |r_i - t_i - d_i|$$

- estimates d_i, v_i calculated for every received packet, but used only at start of talk spurt

- for first packet in talk spurt, playout time is:

$$playout\text{-}time_i = t_i + d_i + Kv_i$$

remaining packets in talkspurt are

Adaptive playout delay (3)

Q: How does receiver determine whether packet is first in a talkspurt?

- if no loss, receiver looks at successive timestamps
  - difference of successive stamps > 20 msec → talk spurt begins.
- with loss possible, receiver must look at both time stamps and sequence numbers
  - difference of successive stamps > 20 msec *and* sequence numbers without gaps --> talk spurt begins.

Today's Outline

Requirements of Multimedia Networking

- Client-side buffering
- Removing Jitter
- **Recovering from Packet Loss**

Coding and Compression

- Digital Audio
- Digital Video

VoIP: recovery from packet loss (1)

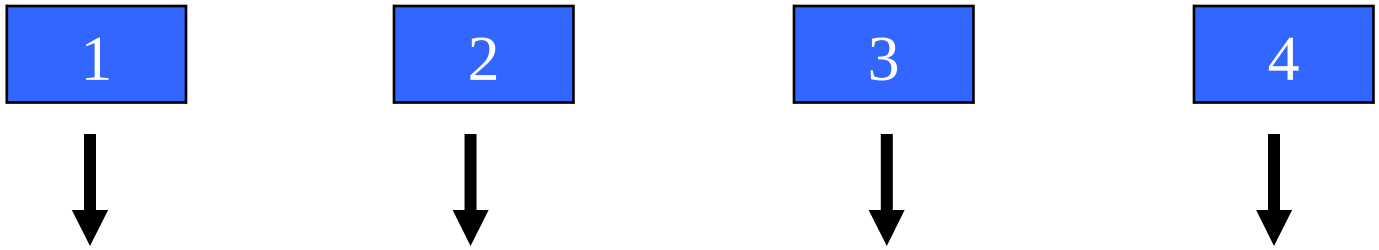
Challenge: recover from packet loss given small tolerable delay between original transmission and playout

- each ACK/NAK takes ~ one RTT
- alternative: *Forward Error Correction (FEC)*
  - send enough bits to allow recovery without retransmission

simple FEC

- for every group of *n* chunks, create redundant chunk by exclusive OR-ing *n* original chunks
- send *n+1* chunks, increasing bandwidth by factor *1/n*
- can reconstruct original *n* chunks if at most one lost chunk from *n+1* chunks, with playout delay

VoIP: Loss

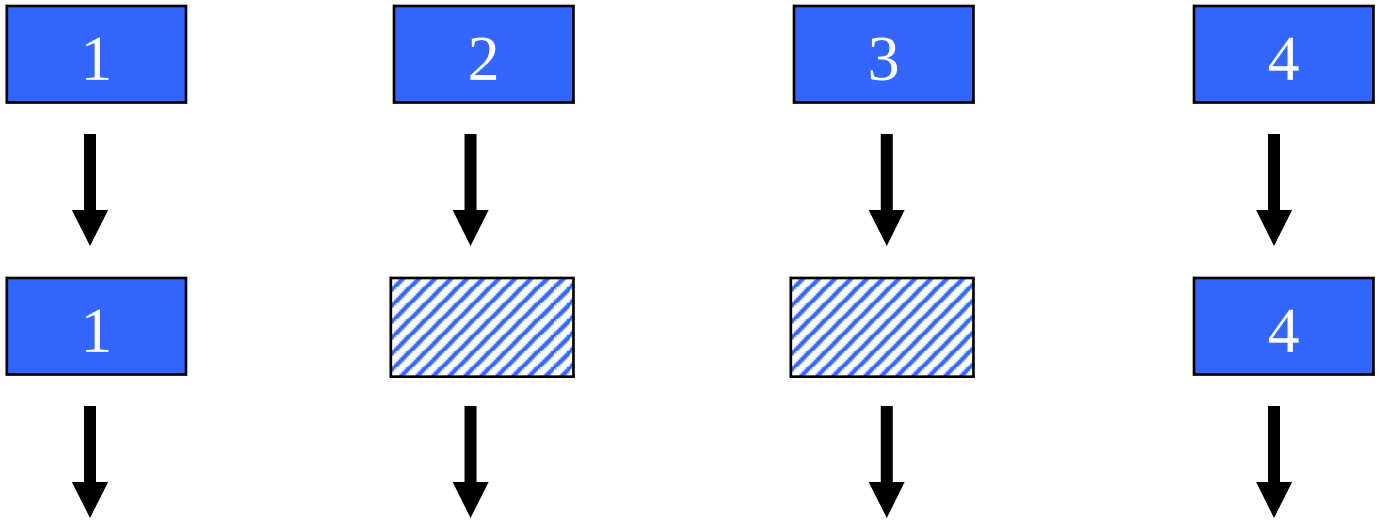


Encode

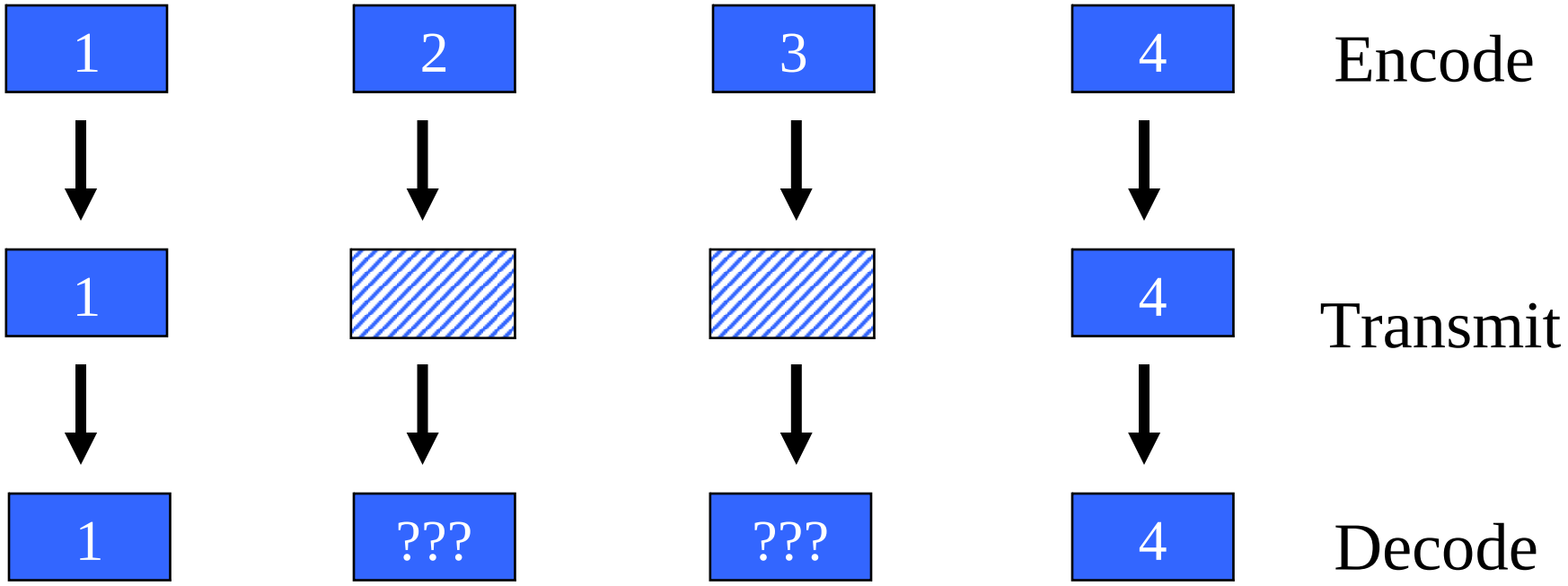
Transmit

Decode

VoIP: Loss

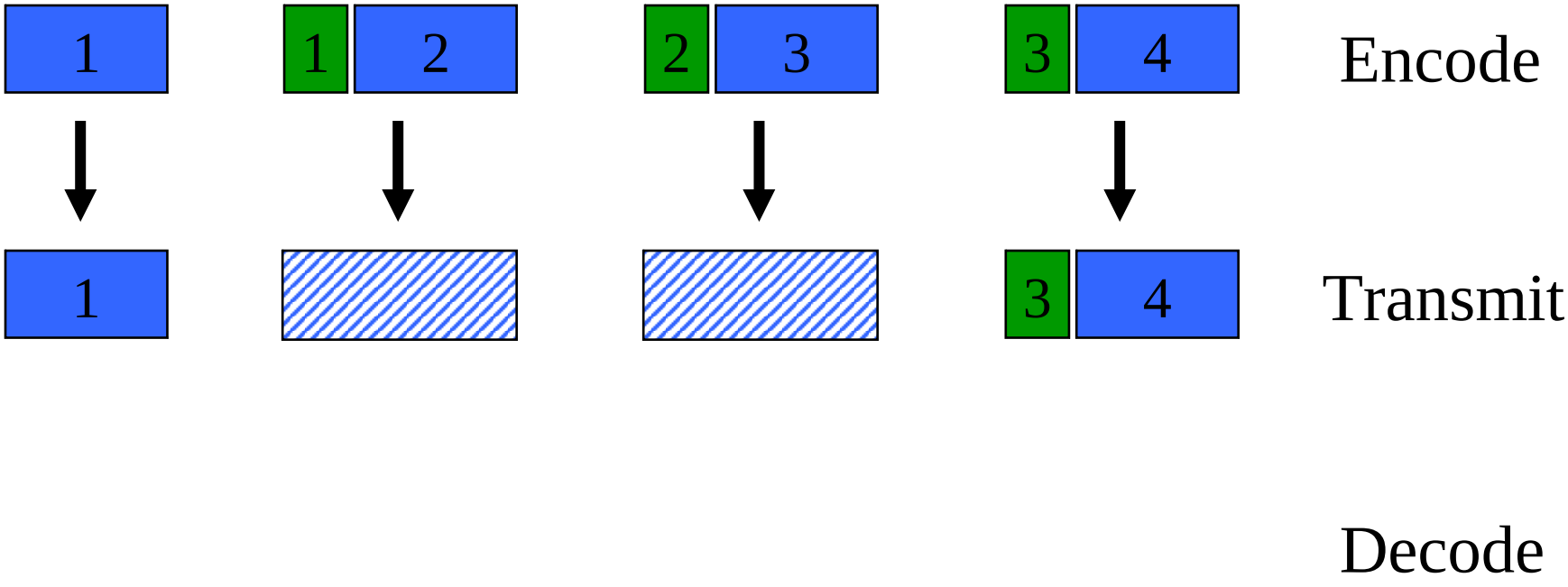


VoIP: Loss

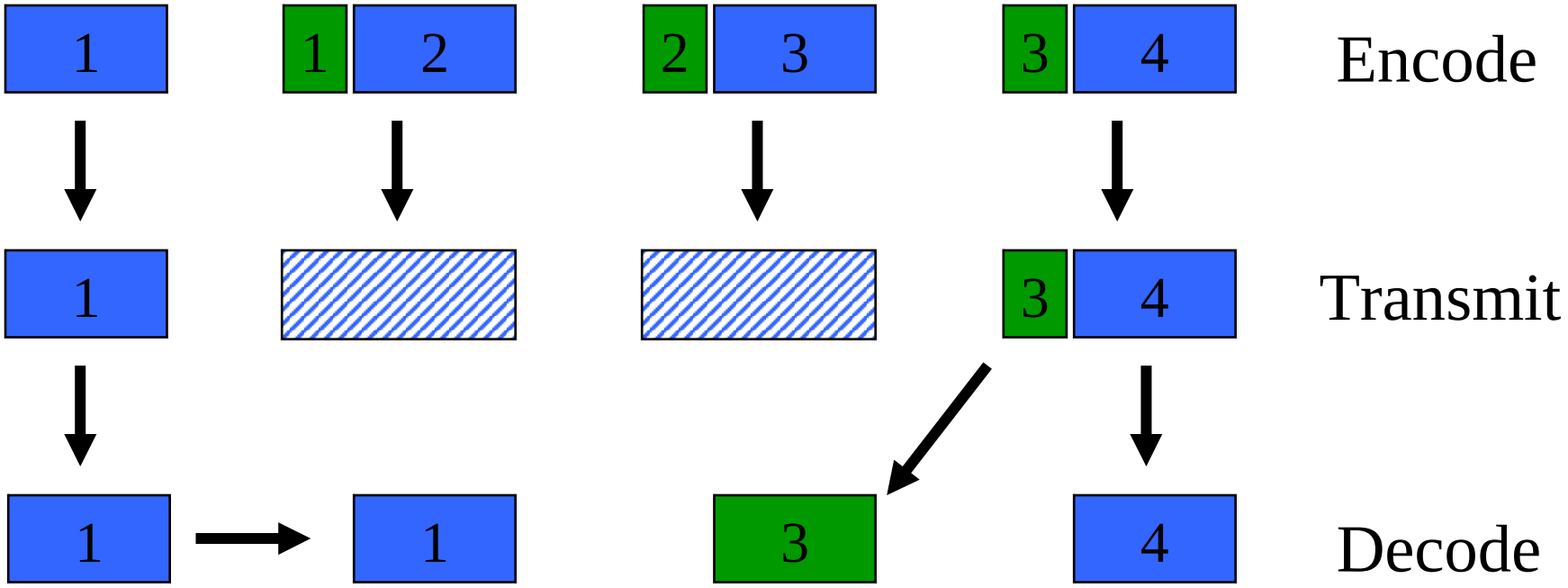


What to do about the missing packets?

VoIP: Recovering from Loss



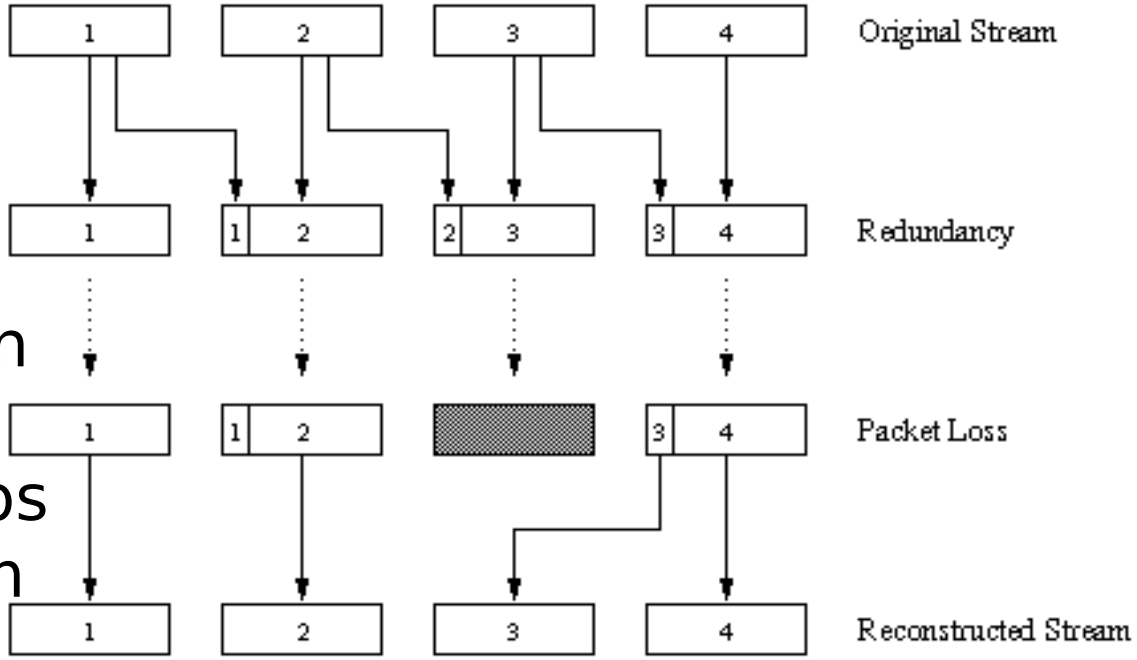
VoIP: Recovering from Loss



VoIP: recovery from packet loss (2)

another FEC scheme:

- "piggyback lower quality stream"
- send lower resolution audio stream as redundant information
- e.g., nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps

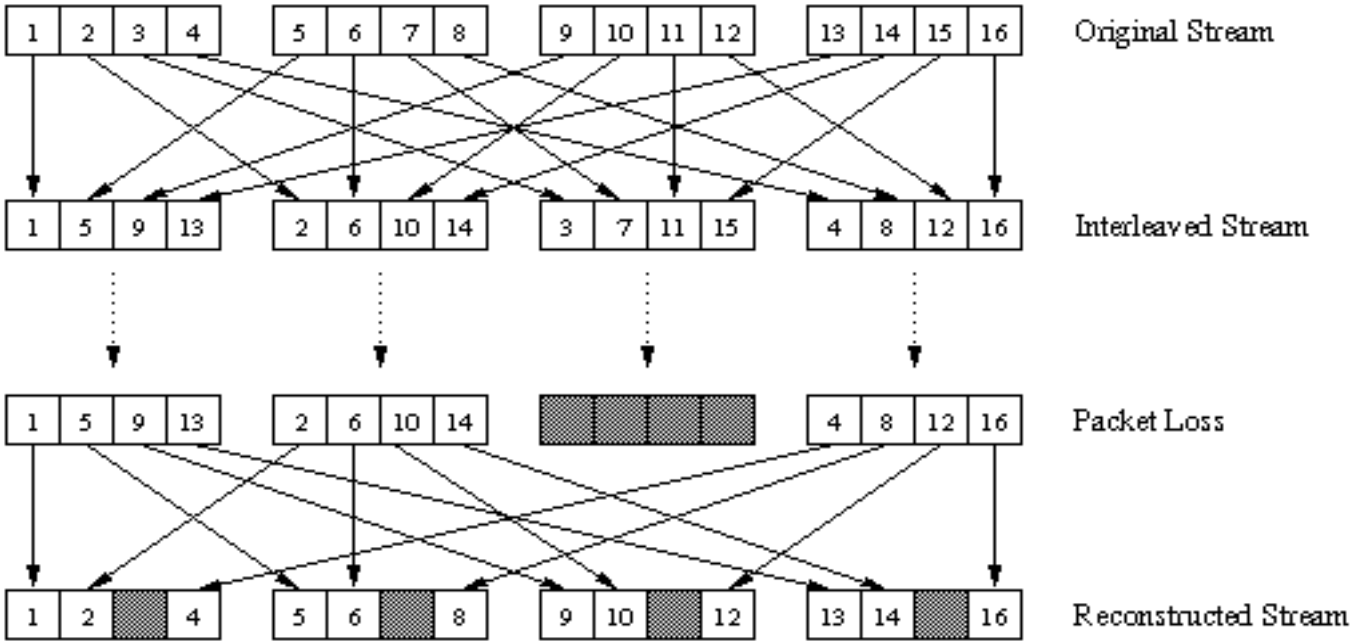


non-consecutive loss: receiver can conceal loss

generalization: can also append (n-1)st and (n-2)nd low-bit r

chunk

VoIP: recovery from packet loss (3)



interleaving to conceal loss:

- audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk
- packet contains small units from different chunks

- if packet lost, still have *most* of every original chunk
- no redundancy overhead, but increases playout delay

Today's Outline

Requirements of Multimedia Networking

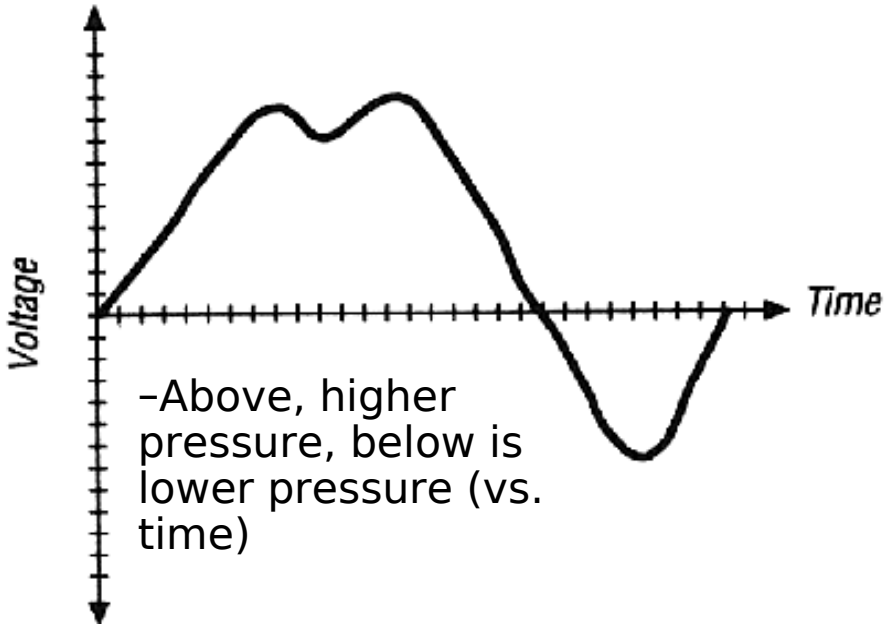
- Client-side buffering
- Removing Jitter
- Recovering from Packet Loss

Coding and Compression

- **Digital Audio**
- Digital Video

Digital Audio

- Sound produced by variations in air pressure
  - Can take any continuous value
  - *Analog* component



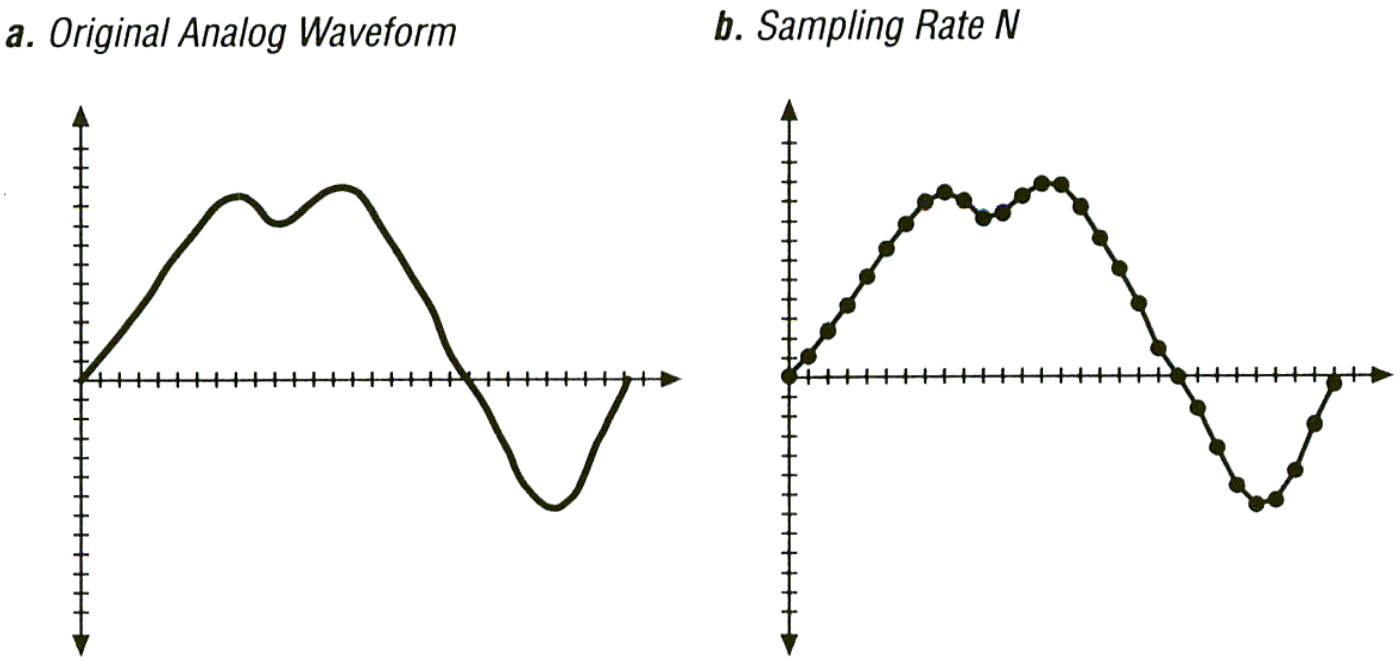
–Above, higher pressure, below is lower pressure (vs. time)

(Show samples with audacity)

- Computers work with *digital*
  - Must convert analog to digital
  - Use *sampling* to get discrete values

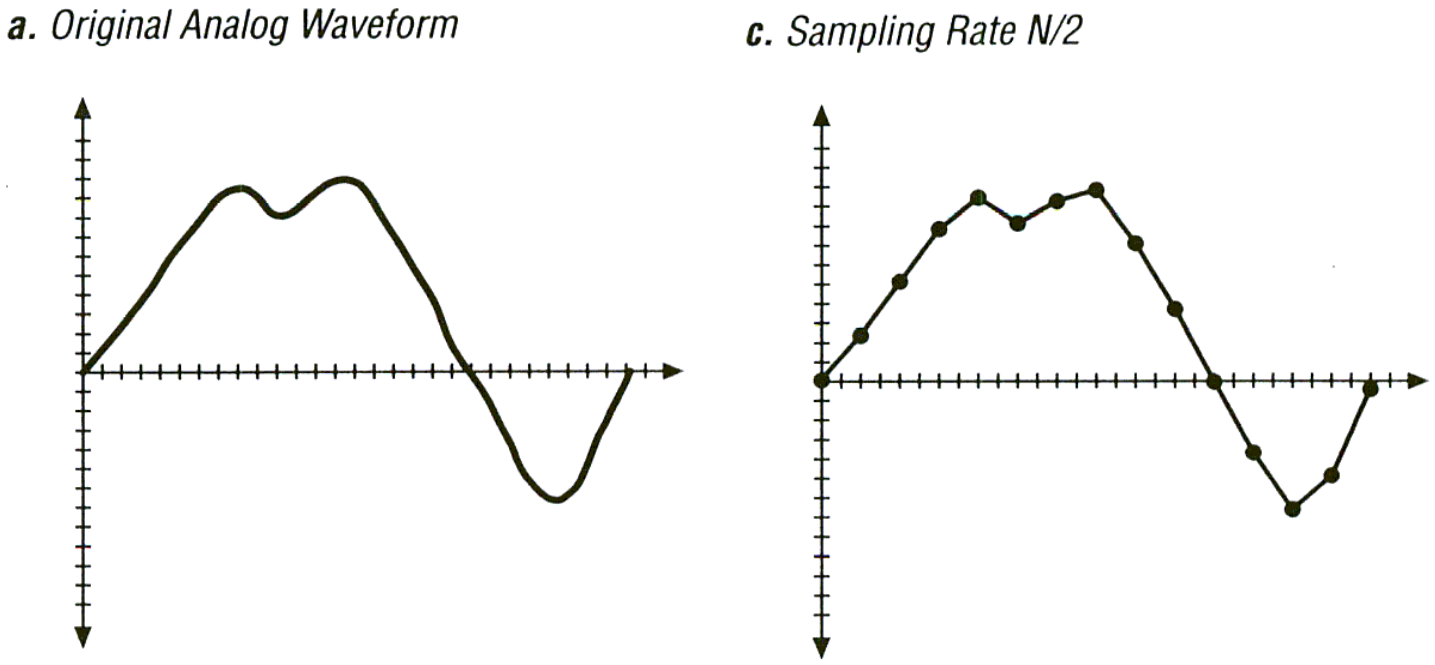
Digital Sampling

- *Sample rate* determines number of discrete values



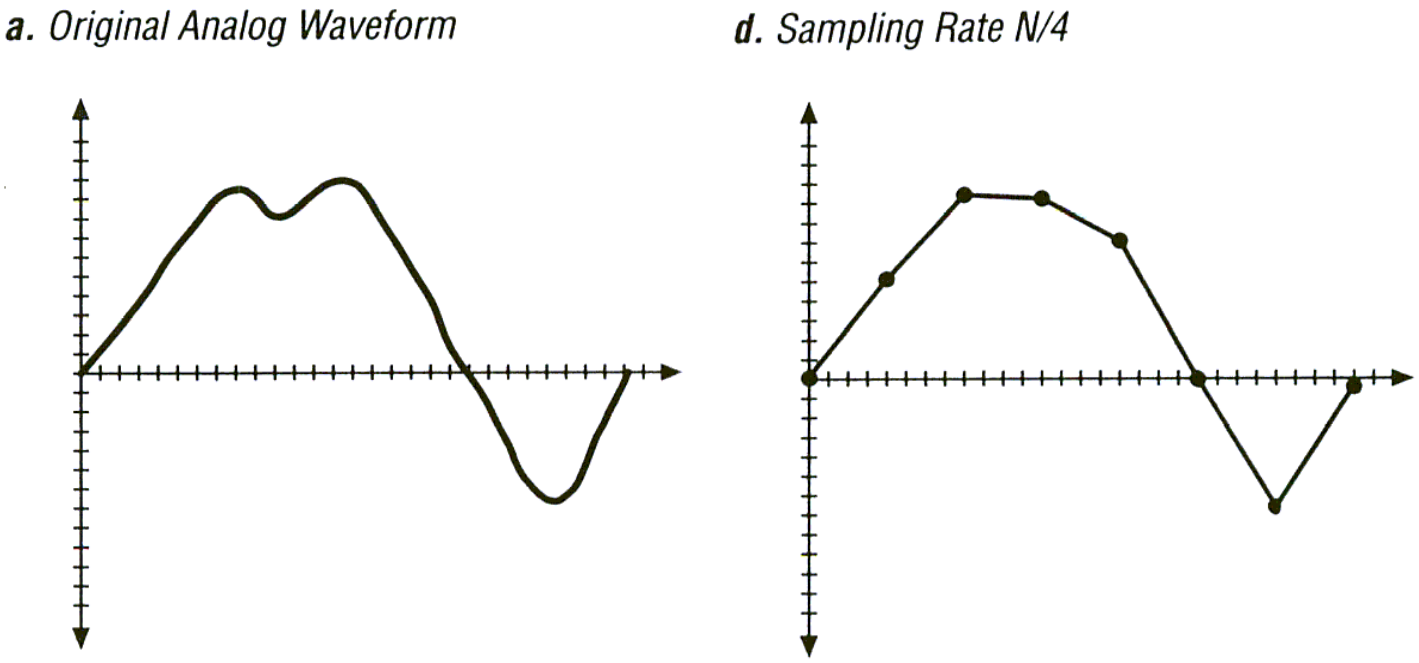
Digital Sampling

- Half the sample rate



Digital Sampling

- Quarter the sample rate



(How often to sample to reproduce curve?)

Sample Rate

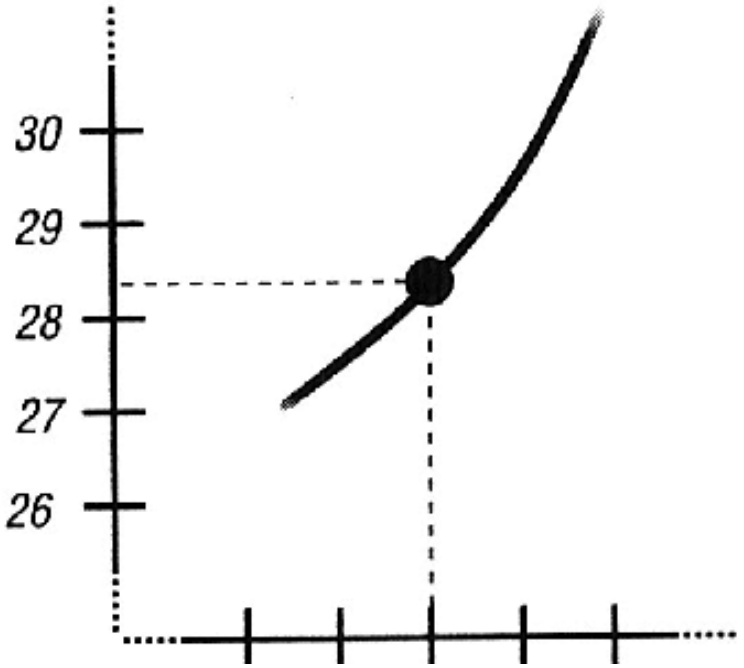
- Shannon’s Theorem: to accurately reproduce signal, must sample at twice the highest frequency
- Why not always use high sampling rate?

Sample Rate

- *Shannon's Theorem*: to accurately reproduce signal, must sample at twice the highest frequency
- Why not always use high sampling rate?
  - Requires more storage
  - Complexity and cost of analog to digital hardware
  - Human's can't always perceive
    - Dog whistle
  - Typically want an "*adequate"* sampling rate
    - "Adequate" depends upon use of reconstructed signal

Sample Size

- Samples have discrete values



- How many possible values?
  - *Sample Size*
  - Say, 256 values from 8 bits

Sample Size

- *Quantization error* from rounding
  - Ex: 28.3 rounded to 28
- Why not always have large sample size?

Sample Size

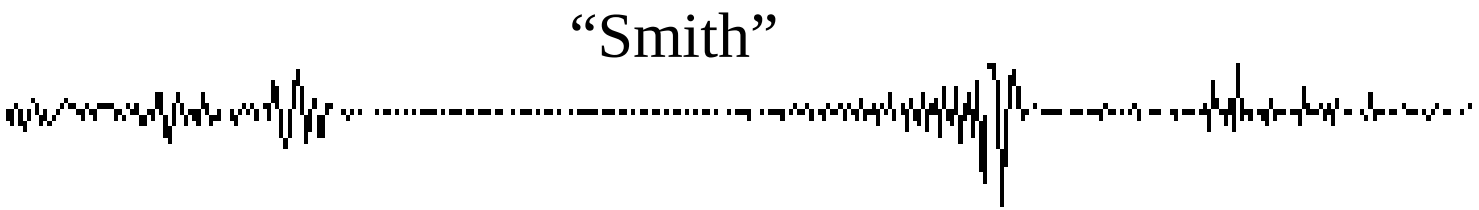
- *Quantization error* from rounding
  - Ex: 28.3 rounded to 28
- Why not always have large sample size?
  - Storage increases per sample
  - Analog to digital hardware becomes more expensive

Groupwork

- Think of as many uses of computer audio as you can
- Which require a high sample rate and large sample size? Which do not? Why?

Audio

- Encode/decode devices are called *codecs*
  - Compression is the complicated part
- For voice compression, can take advantage of speech:



- Many similarities between adjacent samples
  - Send differences (ADPCM)
- Use understanding of speech
  - Can ‘predict’ (CELP)

Audio by People

- Sound by breathing air past vocal cords
  - Use mouth and tongue to shape vocal tract
- Speech made up of phonemes
  - Smallest unit of distinguishable sound
  - Language specific
- Majority of speech sound from 60-8000 Hz
  - Music up to 20,000 Hz
- Hearing sensitive to about 20,000 Hz
  - Stereo important, especially at high frequency
  - Lose frequency sensitivity with age

Typical Encoding of Voice

- Today, telephones carry digitized voice
- 8000 samples per second
  - Adequate for most voice communication
- 8-bit sample size
- For 10 seconds of speech:
  - 10 sec x 8000 samp/sec x 8 bits/samp
    = 640,000 bits or 80 Kbytes
  - Fit 3 minutes of speech on a floppy disk
  - Fit 8 weeks of sound on typical hard disk
- Ok for voice (but Skype better), but what about music?

Typical Encoding of Audio

- Can only represent 4 KHz frequencies (why?)
- Human ear can perceive 10-20 KHz
  - Full range used in music
- CD quality audio:
  - sample rate of 44,100 samples/sec
  - sample size of 16-bits
  - 60 min x 60 secs/min x 44100 samp/sec
    x 2 bytes/samples x 2 channels
    = 635,040,000, about 600 Mbytes (typical CD)
- Can use *compression* to reduce
  - mp3 (“as it sounds)”, RealAudio
  - 10x compression rate, same audible quality

Sound File Formats

- Raw data has samples (interleaved w/stereo)
- Need way to ‘parse’ raw audio file
- Typically a header
  - Sample rate
  - Sample size
  - Number of channels
  - Coding format
  - …
- Examples:
  - .au for Sun μ-law, .wav for IBM/Microsoft
  - .mp3 for MPEG-layer 3

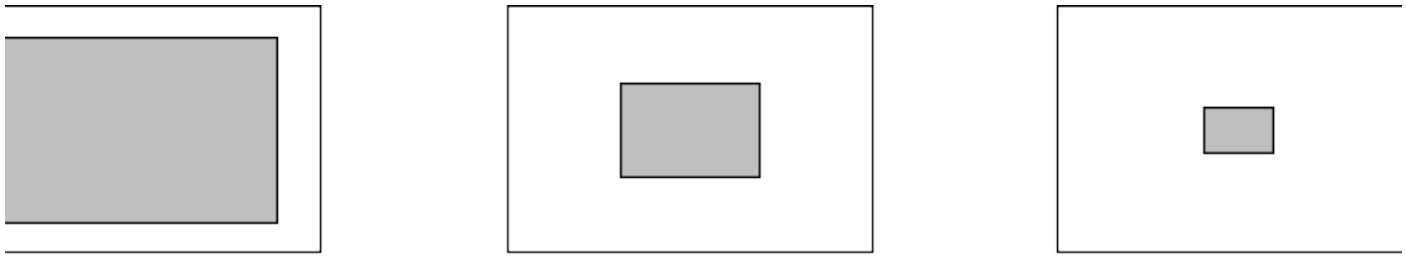
Graphics and Video

"A Picture is Worth a Thousand Words"

- People are visual by nature
- Many concepts hard to explain or draw
- Pictures to the rescue!
- Sequences of pictures can depict motion
  - Video!

Video Images

- Traditional television is 646x486 (NTSC)
- HDTV is 1920x1080 (1080p), 1280x720 (720p), 852x480 (480p)
- Digital video smaller
  - 352x288 (H.261), 176x144 (QCIF)



- Monitors higher resolution than traditional TV
- Computer video often called “Postage Stamp”

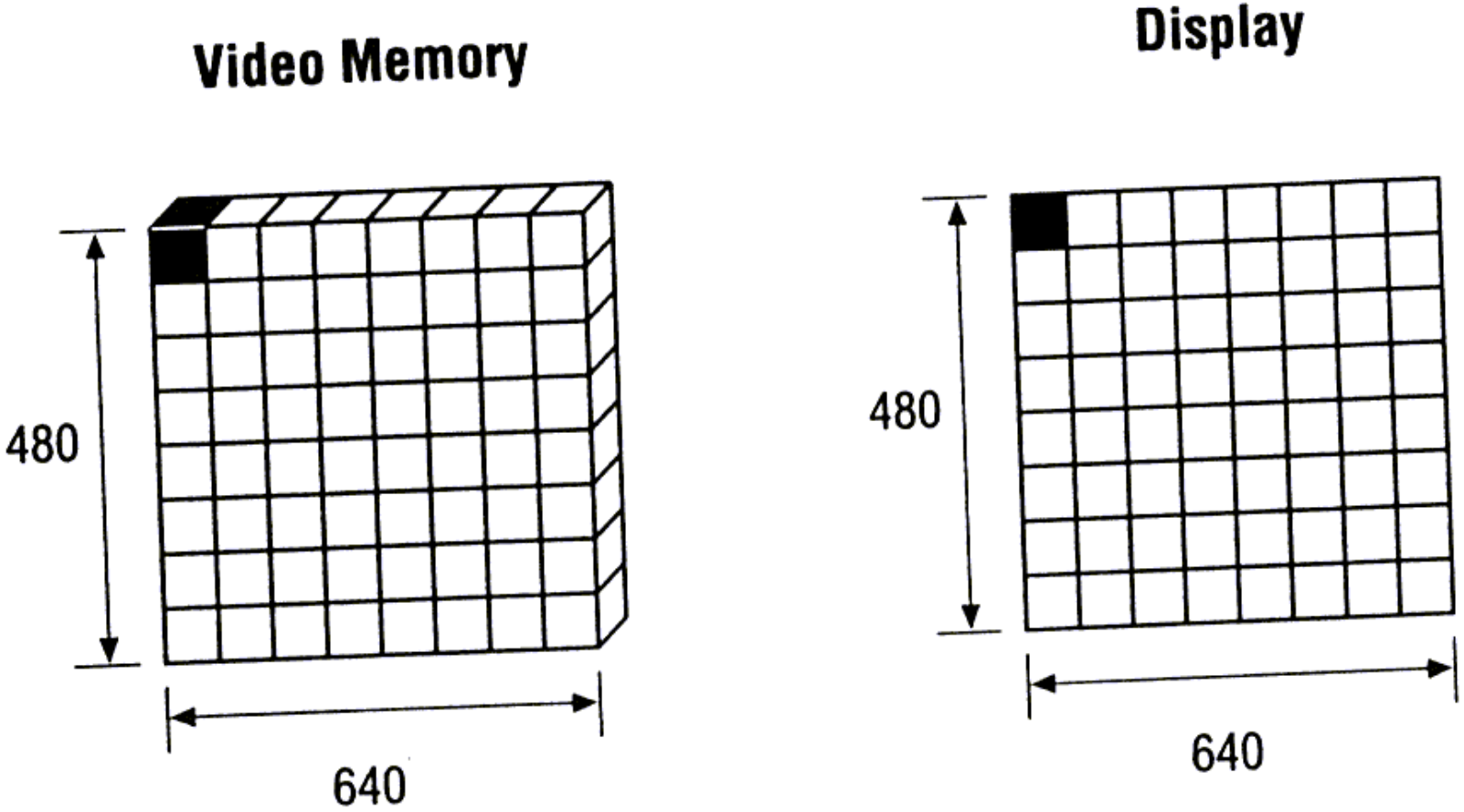
Video Image Components

- Luminance (Y) and Chrominance: Hue (U) and Intensity (V) - *YUV*
  - Human eye less sensitive to color than luminance, so those sampled at less resolution
- YUV has backward compatibility with BW televisions (only had Luminance)
  - Monitors are typically Red Green Blue (RGB)
  - (Why are primary colors Red Yellow Blue?)

Graphics Basics

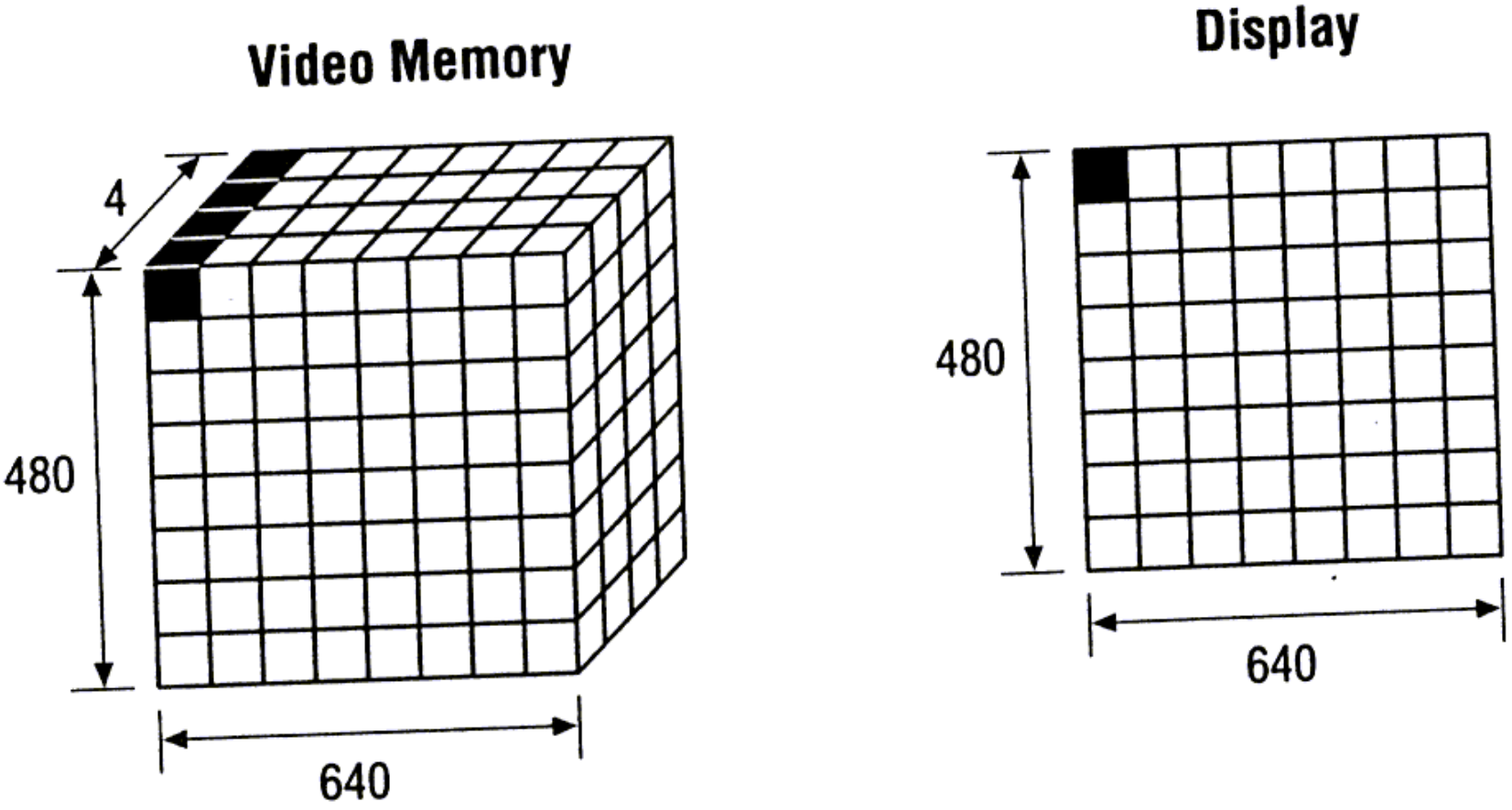
- Display images with graphics hardware
- Computer graphics (pictures) made up of pixels
  - Each pixel corresponds to region of memory
  - Called *video memory* or *frame buffer*
- Write to video memory
  - Traditional monitor displays with raster cannon
  - LCD monitors align crystals with electrodes

Monochrome Display



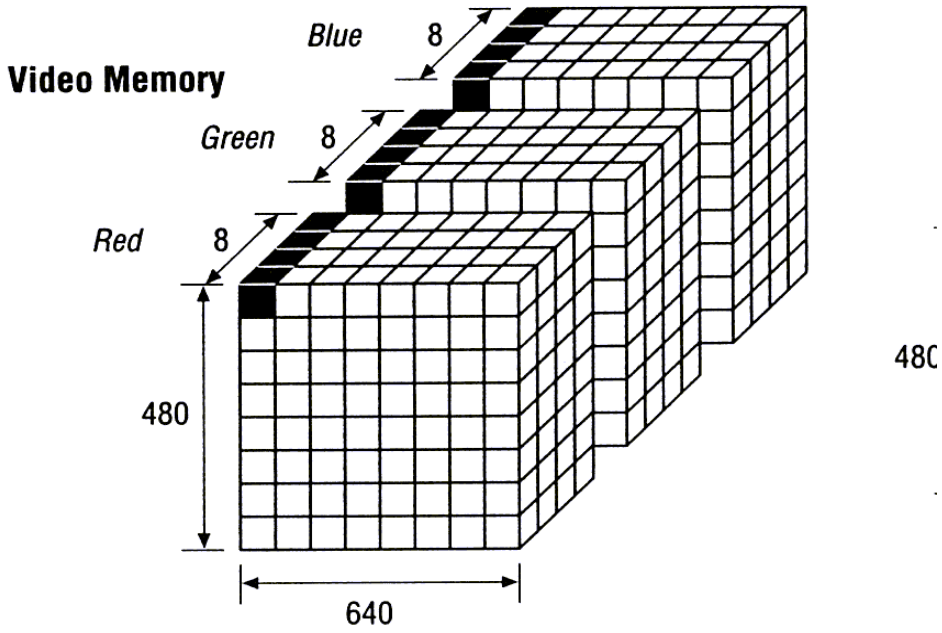
- Pixels are on (black) or off (white)
  - *Dithering* can make area appear gray

Grayscale Display



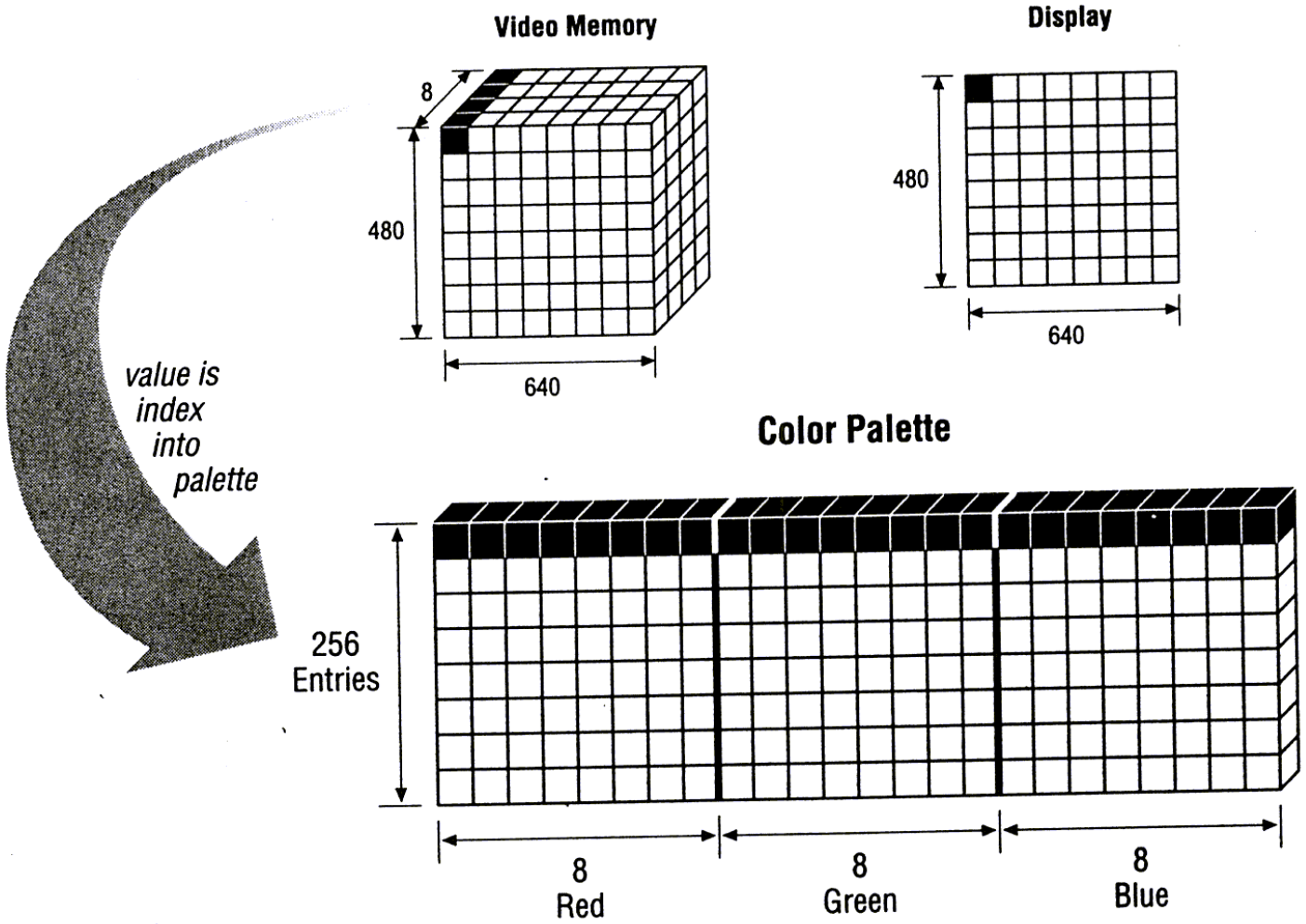
- *Bit-planes*
  - 4 bits per pixel, 2^4 = 16 gray levels

Color Displays



- Humans can perceive far more colors than grayscales
  - Cones (color) and Rods (gray) in eyes
- All colors seen as combo of **red**, **green** and **blue**
- Visual maximum needed
  - 24 bits/pixel, 2^{24} ~ 16 million colors (true color)
- Requires 3 bytes per pixel

Video Palettes



- Still have 16 million colors, only 256 at a time
- Complexity to lookup, color flashing
- Can dither for more colors, too

Graphics Summary

Display Type	Bits Per Pixel	Colors	Resolu-tion	Video Memory
monochrome	1	2 (black and white)	640x480	38 KB
grayscale	4	16 shades of gray	640x480	150 KB
color	24	16 million	640x480	900 KB
color with palette	8	256 from palette of 16 million	640x480	301 KB
monochrome	1	2 (black and white)	1024x768	96 KB
grayscale	4	16 shades of gray	1024x768	384 KB
color	24	16 million	1024x768	2.3 MB
color with palette	8	256 from palette of 16 million	1024x768	769 KB

- Linux: xdpyinfo, display→settings
- Windows: rt click desktop→display properties→settings
- Mac: apple→system preferences→displays

Moving Video Images (Guidelines)

- Series of frames with changes appear as motion (typically ~ 30 fps)
- Unit is Frames Per Second (fps)
  - 24-30 fps: full-motion video
  - 15 fps: full-motion video approximation
  - 7 fps: choppy
  - 3 fps: very choppy
  - Less than 3 fps: slide show

Moving Video Images

- Assume 30 fps uncompressed

Time:Size	640x480	320x240	160x120
1sec	27Mb	6.75Mb	1.68Mb
1min	1.6Gb	400Mb	100Mb
1hour	97Gb	24Gb	6Gb
1000hours	97Tb	24Tb	6Tb

Uncompressed video is enormous!

Video Compression

Time v. Scale	None	3:1	25:1 (JPEG)	100:1 (MPEG)
1 sec	27 Mb	9 Mb	1.1 Mb	270 Kb
1 min	1.6 Gb	540 Mb	65 Mb	16 Mb
1 hour	97 Gb	32 Gb	3.9 Gb	970 Mb

640x480

Time v. Scale	None	3:1	25:1 (JPEG)	100:1 (MPEG)
1 sec	6.75 Mb	2.25 Mb	270 Kb	68 Kb
1 min	400 Mb	133 Mb	16 Mb	4 Mb
1 hour	24 Gb	8 Gb	1 Gb	240 Mb

320x240

- Lossless or Lossy
- Intracoded or Intercoded
  - Take advantage of dependencies between frames
    - Motion
- (More on MPEG later)