Jan-28 Lecture

Multimedia Data

- From a networking perspective, what is different about multimedia data?

Multimedia Data

- From a networking perspective, what is different about multimedia data?
  - In some cases, nothing… if it is just a file transfer, then there are no particular additional requirements
  - Only when data is "real-time" is there any point to study
    - *"soft" real-time*: used/consumed as soon as it is received, the faster the better
    - *"hard" real-time*: if it doesn't make it by a deadline, bad things happen

Multimedia Data

- From a networking perspective, what is different about multimedia data?
  - In some cases, nothing… if it is just a file transfer, then there are no particular additional requirements
  - Only when data is "real-time" is there any point to study
    - *"soft" real-time*: used/consumed as soon as it is received, the faster the better
    - *"hard" real-time*: if it doesn't make it by a deadline, bad things happen

- For improved delivery, there are two choices:
  1. Implement support "in the network" (e.g., quality of service)
     - Better than best-effort
  2. Implement support "on top of the network"
     - Transport-layer support, session-layer support, application-layer support

Multimedia Data

- Support “in the network” (e.g., quality of service)
  - Mark packets according to their data type
  - Queue/forward different types of packets differently
  - Create a (logical) connection-based system and reserve more/less resources for different types of packets

Multimedia Data

- Support “in the network” (e.g., quality of service)
  - Mark packets according to their data type
  - Queue/forward different types of packets differently
  - Create a (logical) connection-based system and reserve more/less resources for different types of packets

- Implement support “on top of the network”
  - Transport layer: TCP or UDP
  - Session layer: UDP+, RTP, RTSP, UPnP, DLNA
  - Application layer: Proprietary solutions and/or at least application-specific solutions

When Streaming Was a Thing

- Links were just able to support bandwidth requirements
  - End-to-end path had to support delivery rate at close to the consumption rate
    - If the path couldn't keep up, playout would stop
  - Could build in buffering at the beginning to avoid interruptions
    - Works okay for non-real-time, but not for VoIP

When Streaming Was a Thing

- Links were just able to support bandwidth requirements
  - End-to-end path had to support delivery rate at close to the consumption rate
    - If the path couldn't keep up, playout would stop
  - Could build in buffering at the beginning to avoid interruptions
    - Works okay for non-real-time, but not for VoIP
- In either situation, it was critical *not* to stop the transmission of data
  - So no TCP
  - Was fine since video could be made robust against errors
- The big challenge was regulating the flow of data from source to decoder

When Streaming Was no Longer a Thing

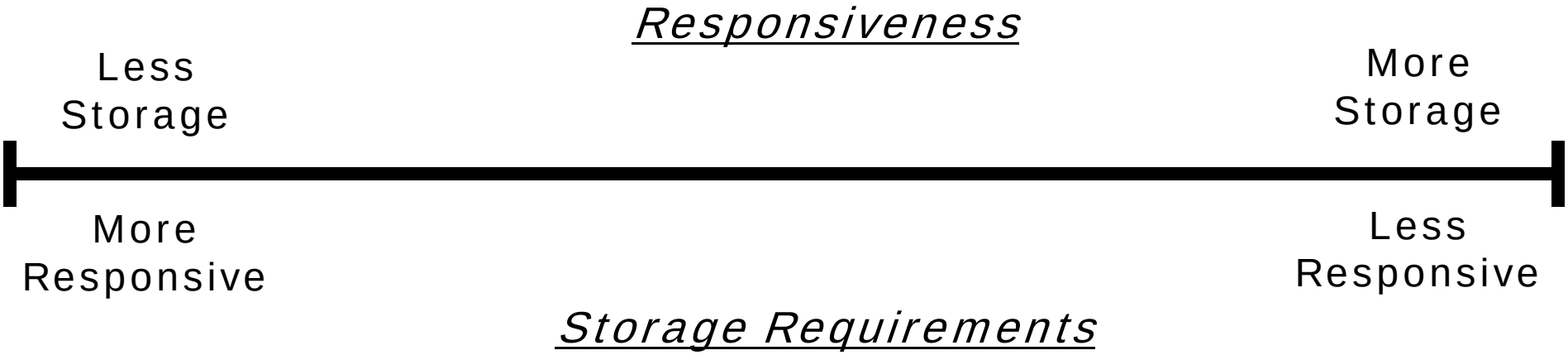
- Links became faster—faster than the rate at which high quality video required more bandwidth
  - The kinds of delays that caused playout interruptions diminished
  - Data could be delivered faster than it was played out
  - TCP became viable again
    - Especially since UDP was used for DoS attacks
    - Especially since UDP was not nice to TCP
  - Sort of still important to regulate the flow of data, but not really
    - Buffering/Memory was increasingly cheap
    - But still some motivation not just to deliver all of the data
  - Still applications like VoIP—data is not available until it is created
    - But here just use minimal buffers, primarily for jitter and TCP behaviors
    - TCP behaviors become increasingly rare in high bandwidth, low congestion environments

Summary of Influences on Throughput

- Rate at which data is made available
  - A factor for live audio/video…
  - …but not for data drawn from a file
- TCP behaviors
  - Certainly slow start, but that doesn't last long
  - Can be a factor if link quality is an issue
- Application layer control
  - Important to have some control of data flow
    - Especially if link is fast and data is coming from a file

Streaming v. Downloading

- *Download*: fetch/retrieve the complete file
- *Streaming*: fetch/retrieve data "just-in-time"



- Lots of techniques to create hybrids of these options

Hybrid Techniques

- *Download*: fetch/retrieve the complete file
- *Streaming*: fetch/retrieve data "just-in-time"

- *Progressive download* (or *buffered streaming*)
  - Download first part of the content first
    - Same as streaming with a significant buffer
  - [Q]: How much to download/buffer before starting?
    - Wait a fixed amount of time
    - Dynamically determine based on download and playout rates
  - Also have to deal with buffer management

Back to Data Type

- The type of application has an impact on the techniques that can be used

- Most demanding application is video conferencing
  - Needs to be interactive and has high bandwidth requirements
  - Voice-over-IP is a close second (same interactive requirements but the bandwidth requirements aren't as high)
  - Also worth mentioning MMOG

- Most other applications in the Internet only have the timing requirement of "I want it as fast as possible"

- Worth talking about errors and resiliency

Sources of Delay in Interactive Voice

- From the time a word is spoken to the time the sound wave is received at the destination user's ear, what are the *major* sources of delay?

Sources of Delay in Interactive Voice

- From the time a word is spoken to the time the sound wave is received at the destination user's ear, what are the *major* sources of delay?

1. Encoding/compression delay
2. Packetization delay
3. Propagation delay
4. Decoding/decompression delay

Sources of Delay in Interactive Voice

- From the time a word is spoken to the time the sound wave is received at the destination user's ear, what are the *major* sources of delay?

1. Encoding/compression delay
2. Packetization delay
3. Propagation delay
4. Decoding/decompression delay

- Another serious concern is jitter
  - In interactive applications, not much you can do about jitter
  - In less stringent streaming applications, can use buffering

Now on to Audio

- Focus today on audio encoding and compression
  - Wednesday on video encoding and compression

Audio Signals

- All sound is an analog signal
  - What we say and what we hear are continuous wave forms

- *Goal*: reduce the analog signal into a series of 1s and 0s so that it can be stored and/or transmitted

Digitizing Analog Audio

- Create a number of samples (per unit time)
- For each sample, identify a discrete *level*, which is then represented digitally

- Levels
  - Represented as bits: usually 8-16 bits (256 – 65356 levels)
  - Sufficient to cover the range of human hearing
  - Fewer levels introduce more *quantizing noise*

- Sample frequency
  - Measure in “kHz”, thousands of samples per second

Uncompressed Digital Audio

- CD is a good example
  - Two channels (stereo)
  - 44,100 samples per second (44.1 kHz)
  - 16 bits per sample
- 2 * 44,100 * 16 = 1.4 Mbps
- Speech (or low quality “anything else”)
  - Mono, 8 kHz, 8 bits per sample
  - Equals 64 kbps
- All of this is without compression

Mu-Law Encoding

- Not much compression at all (so really just an encoding method), but technically there is some compression

- The compression that does exist is to covert 14-bits of sample value using only 8 bit samples

Mu-Law Encoding

- Not much compression at all (so really just an encoding method), but technically there is some compression

- The compression that does exist is to covert 14-bits of sample value using only 8 bit samples
  - Does this by having non-linear levels
  - Based on fact that humans are unable to differentiate very low and very high frequencies
  - So the quantizing noise generated by 8-bit non-linear encoding is the same as a 14-bit linear encoding

- Samples (8 bits) are independent and constant bit rate
  - Really useful for HW#2

- But if we don't want simple, how do we compress?

So Where Does Compression Happen?

- The key fact in audio is that from one sample instant to the next, the information is often the same (or very similar)
- So instead of just encoding the sampled audio value, the difference from the previous value is encoded
  - The difference between Pulse Code Modulation (PCM) and Adaptive Differential Pulse Code Modulation (ADPCM)
- Differential coding often results in long runs of very small to zero sample values
  - These runs can then be compressed using Run Length Coding
  - Other techniques like Huffman coding can be used
- These are all just simple data compression techniques

Next Generation Compression

- Take advantage of the way that the human ear perceives and prioritizes audio
  - We'll see similar ways in video compression that take advantage of the way the human eye perceives and prioritizes video

- Two metrics to consider
  - *Quantitative*: the difference between the *original* analog signal and the *reproduced* digital analog—measured as a noise value (SNR or dB)
  - *Qualitative*: several scales that are used as part of human testing

MP3

- MPEG (Motion Picture Experts Group), Part 3
  - Other two parts are for “system” (audio plus video) and video
  - Not to be confused with the fact that MP3 has three layers
- Wanted 6-to-1 compression
  - Original was ~1.5Mbps (2 channels, 44.1 kHz, 16 bit samples)
  - Goal was ~256 kbps
- “Auditory masking”
  - Strong signal drowns out weaker signal
  - Convert audio signal to frequency
  - Divide the frequencies into sub-bands
  - Digitize the sub-bands

Other Schemes

- Lots of work to develop different ways of collecting, encoding, and compressing samples
  - Transform the data
  - Filter out unimportant data
  - Re-sort data to achieve better compression
- Different schemes optimized for voice, music and other types of audio
  - Voice is surprisingly redundant
- Some protocols also incorporate aspects like security, resiliency, etc.

Tradeoffs to Consider

- The more complex the encoder, the more delay it adds
  - Once upon a time, encoder complexity could overwhelm the processing resources of the device, but not so much anymore
  - Only consideration might be battery power on mobile device
- The better the encoder and the greater the compression the more packet loss will affect the stream

Dealing with Loss

- Many compression schemes are not good at dealing with lost data (e.g., MP3)
    - If you cut data out of an MP3 file, it will likely crash the player
- Work to develop robust encoding schemes has become less important
  - Most streaming just uses TCP and progressive download
  - …except video conferencing and VoIP
- Techniques to deal with packet loss?

Dealing with Loss

- Many compression schemes are not good at dealing with lost data (e.g., MP3)
    - If you cut data out of an MP3 file, it will likely crash the player
- Work to develop robust encoding schemes has become less important
  - Most streaming just uses TCP and progressive download
  - …except video conferencing and VoIP
- Techniques to deal with packet loss
  - Use small packets (adds overhead)
  - Use loss concealment (e.g., play last packet)
  - Scramble and de-scramble the data (but introduces delay)
  - Add forward error correction (adds overhead)

HW#2 Overview

- Straightforward set of programming steps
  - Open source file
  - Read chunks (i.e., packets of data)
  - Decide if received or lost
  - Write data or if lost, write something in its place
    - Ending file size will be exactly the same
  - Open and listen to resulting file
- Use mu-law encoded packets (.au format)
  - Samples are 1 byte (read as unsigned ints) and independent
  - Can replace individual samples easily
  - Always copy the header
    - http://en.wikipedia.org/wiki/Au_file_format
- Run some experiments and report on the results
  - Packet size, loss rates, basic error concealment