Multimedia Networking

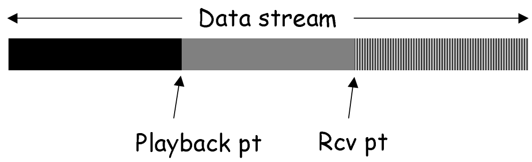
- Application classes
  - streamed stored audio/video
  - one-to-many (multicast) streaming of real-time a/v
  - real-time interactive audio/video
- Typical application issues
  - packet jitter
  - packet loss / recovery
- Internet protocols for multimedia
  - RTSP
  - RTP/RTCP
  - H.323
- Text: Kurose-Ross, Chapter 6

Example Multimedia Apps

- Streamed stored audio/video
  - movies, CS-653 taped lectures (available on MANIC)
- One-to-many streaming
  - News broadcasts, popular movies
- Real-Time Interactive
  - IP telephony, teleconference, distributed gaming

Multimedia terminology

- **Multimedia session**: a session that contains several media types
  - e.g., a movie containing both audio & video
- **Continuous-media session**: a session whose information must be transmitted "continually"
  - e.g., audio, video, but not text (unless ticker-tape)
- **Streaming**: application usage of data during its transmission



Multimedia vs. Raw Data

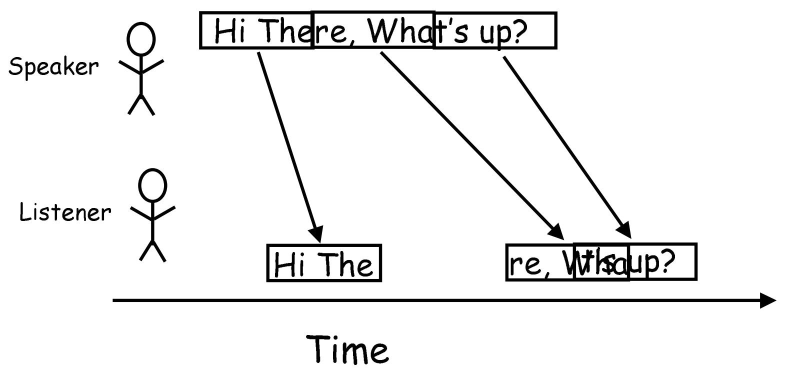
- Multimedia
  - e.g., Audio/Video
  - Tolerates **some** packet loss
  - Packets have timed playout reqmts
- Raw Data
  - e.g., FTP, web page, telnet
  - Lost packets must be recovered
  - Timing: faster delivery always preferred

Why not just use TCP for multimedia traffic?

-
-

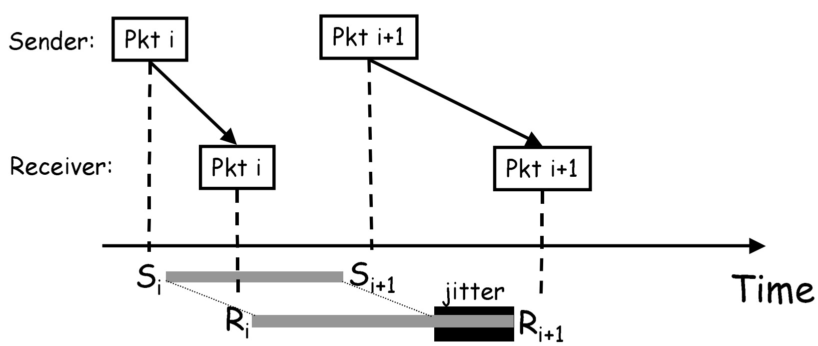
Jitter

- The Internet makes no guarantees about time of delivery of a packet
- Consider an IP telephony session:



Jitter (cont'd)

- A packet pair's jitter is the difference between the transmission time gap and the receive time gap



- Desired time-gap: $S_{i+1} - S_i$ Received time-gap: $R_{i+1} - R_i$
- Jitter between packets i and i+1: $(R_{i+1} - R_i) - (S_{i+1} - S_i)$

Buffering: A Remedy to Jitter

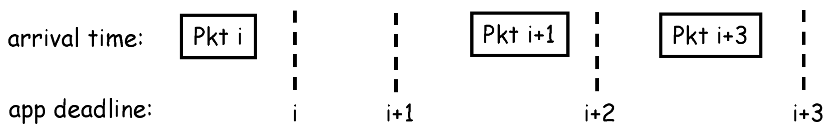
- Delay playout of received packet i until time $S_i + C$ (C is some constant)
- How to choose value for C?
  - Bigger jitter → need bigger C
  - Small C: more likely that $R_i > S_i + C$ ↔ missed deadline
  - Big C:
    - requires more packets to be buffered
    - increased delay prior to playout
  - Application timing reqmts might limit C:
    - Interactive apps (IP telephony) can't impose large playout delays (e.g., the international call effect)
    - non-interactive: more tolerant of delays, but still not infinite...

Adaptive Playout

- For some applications, the playout delay need not be fixed
- e.g., [Ramjee 1994] / p. 430 in Kurose-Ross
  - Speech has talk-spurts w/ large periods of silence
  - Can make small variations in length of silence periods w/o user noticing
  - Can re-adjust playout delay in between spurts to current network conditions

Packet Loss / Recovery

- Problem: Internet might lose / excessively delay packets making them unusable for the session



usage status: …, i used, i+1 late, i+2 lost, i+3 used, …

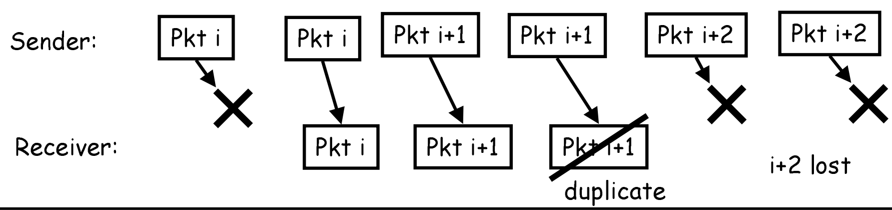
- Solution step 1: Design app to tolerate **some** loss
- Solution step 2: Design techniques to recover **some** lost packets within application's time limits

Applications that tolerate some loss

- Techniques are medium-specific and influence the coding strategy used (beyond scope of course)
  - Video: e.g., MPEG
  - Audio: e.g., GSM, G.729, G.723, replacing missing pkts w/ white-noise, etc.
- Note: loss tolerance is a secondary issue in multimedia coding design
- Primary issue: compression

Reducing loss w/in time bounds

- Problem: packets must be recovered prior to application deadline
- Solution 1: extend deadline, buffer @ rcvr, use ARQ
  - Recall: unacceptable for many apps (e.g., interactive)
- Solution 2: Forward Error Correction (FEC)
  - Send "repair" before a loss is reported
  - Simplest FEC: transmit redundant copies



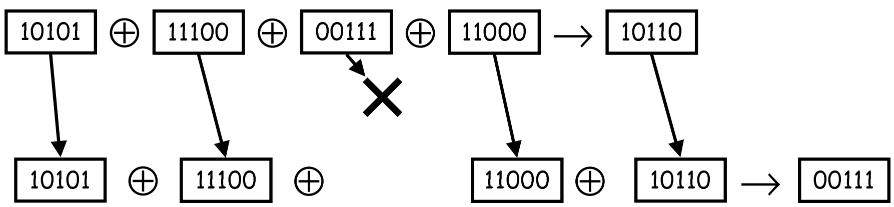
More advanced FEC techniques

- FEC often used at the bit-level to repair corrupt/missing bits (i.e., in the data-link layer)

- Here, we will consider using FEC at the packet layer (special repair packets):

A Simple XOR code

- For low packet loss rates (e.g. 5%), sending duplicates is expensive (wastes bandwidth)
- XOR code
  - XOR a group of data pkts together to produce repair pkt
  - Transmit data + XOR: can recover 1 lost pkt



Reed-Solomon Codes

- Based on simple linear algebra
  - can solve for n unknowns with n equations
  - each data pkt represents a value
  - Sender and receiver know which "equation" is in which pkt (i.e., information in header)
  - Rcvr can reconstruct n data pkts from any set of n data + repair pkts
  - In other words, send n data pkts + k repair packets, then if no more than any k pkts are lost, then all data can be recovered
- In practice
  - To reduce computation, linear algebra is performed over fields that differ from the usual $\Re$

Reed Solomon Example over $\Re$

Pkt 1: Data1

Pkt 2: Data2

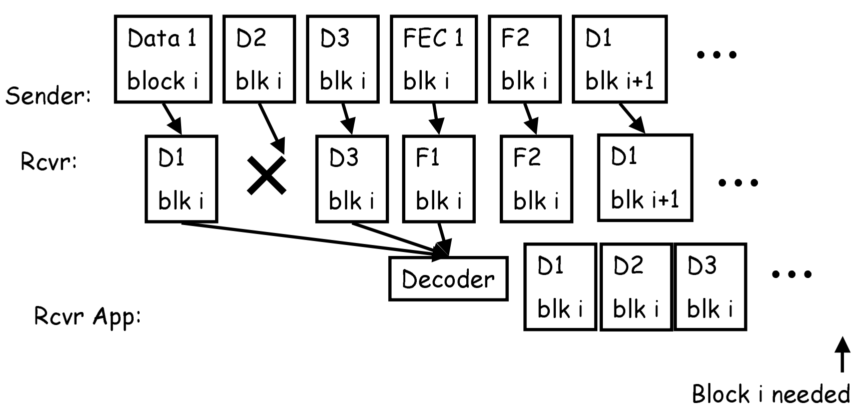
Pkt 3: Data3

Pkt 4: Data1 + Data2 + 2 Data3

Pkt 5: 2 Data1 + Data2 + 3 Data3

- Pkts 1,2,3 are data (Data1, Data2, and Data3)
- Pkts 4,5 are linear combos of data
- Assume 1-5 transmitted, pkts 1 & 3 are lost:
  - Data1 = (2 * Pkt 5 - 3 * Pkt 4 + Pkt 2)
  - Data2 = Pkt 2
  - Data3 = (2 * Pkt 4 - Pkt 5 - Pkt 2)

Using FEC for continuous-media



- Divide data pkts into **blocks**
- Send FEC repair pkts after corresponding data block
- Rcvr decodes and supplies data to app before block i deadline

FEC via variable encodings

- Packet contents:
  - high quality version of frame k
  - low quality version of frame k-c (c is a constant)
  - If packet i containing high quality frame k is lost, then can use packet i+c with low quality frame k in place

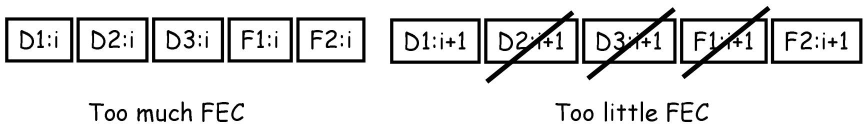
i	low: k-c	high: k
i+1	low: k-c+1	high: k+1
i+2	low: k-c+2	high: k+2

FEC tradeoff

- FEC reduces channel efficiency:
  - Available Bandwidth: B
  - Fraction of pkts that are FEC: f
  - Max data-rate (barring pkt loss): B (1-f)
- Need to be careful how much FEC is used!!!

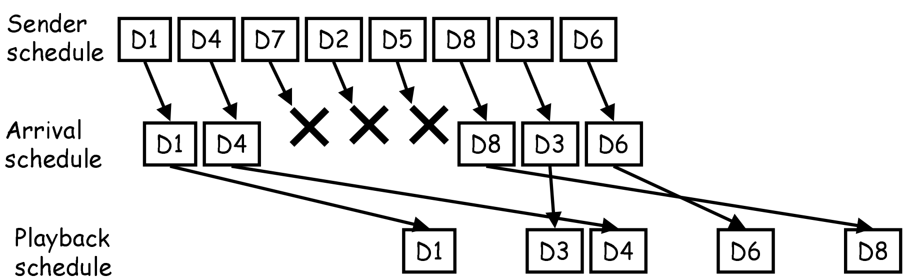
Bursty Loss:

- Many codecs can recover from short (1 or 2 packet) loss outages
- Bursty loss (loss of many pkts in a row) creates long outages: quality deterioration more noticeab
- FEC provides less benefit in a bursty loss scenario (e.g., consider 30% loss in bursts of 3)



Interleaving

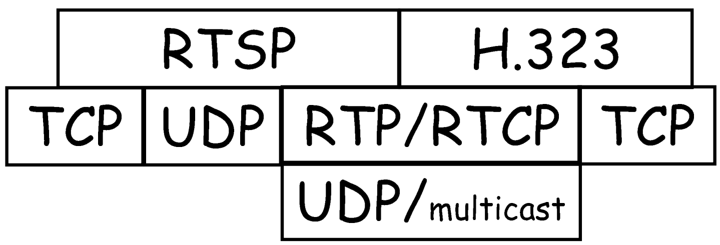
- To reduce effects of burstiness, reorder pkt transmissions



- Drawback: induces buffering and playout delay

Multimedia Internet Protocols

- We'll look at 3:
  - RTP/RTCP: transport layer
  - RTSP: session layer for streaming media applications
  - H.323: session layer for conferencing applications



RTP/RTCP [RFC 1889]

- Session data sent via RTP (Real-time Transfer Protocol)
- RTP components / support:
  - sequence # and timestamps
  - unique source/session ID (SSRC or CSRC)
  - encryption
  - payload type info (codec)
- Rcvr/Sender session status transmitted via RTCP (Real-time Transfer Control Protocol)
  - last sequence # rcvd from various senders
  - observed loss rates from various senders
  - observed jitter info from various senders
  - member information (canonical name, e-mail, etc.)
  - control algorithm (limits RTCP transmission rate)

RTP/RTCP details

- All of a session's RTP/RTCP packets are sent to the same multicast group (by all participants)
  - All RTP pkts sent to some even-numbered port, 2p
  - All RTCP pkts sent to port 2p+1
- Only data senders send RTP packets
- All participants (senders/rcvrs) send RTCP packets

RTP header

Payload Type	Sequence #	Timestamp	Synchronization Source Identifier	Misc

- Why do most (all) multimedia apps require
  - sequence #?
  - timestamp?
  - (unique) Sync Source ID?
- Why should every pkt carry the 7-bit payload type?
  - Why not just when sender initiates session?
- Transmission rate: application specific (no congestion control specified in RTP)

RTCP packets

- There are several types of RTCP packets
  - SR: sender report - transmission & reception stats
  - RR: receiver report - reception stats
  - SDES: Source description items
  - BYE: end-of-participation message
  - APP: application-specific functions
- Typically, several RTCP packets of different types are transmitted w/in a single UDP packet

What RTCP provides

- Info to detect colliding Synch source ID's
- Contact info (e-mail, true name) of participants
- Info to count # of session participants
- Reception quality of all participants

- How does RTCP avoid creating congestion if all participants send RTCP packets?
  - consider a broadcast TV transmission

RTCP congestion control

- A session's aggregate RTCP bandwidth usage should be 5% of the session's RTP bandwidth
  - 75% of the RTCP bandwidth goes to the receivers
  - 25% goes to the senders
- $T_{sender} = \frac{\text{\# senders * avg RTCP pkt size}}{.25 * .05 * \text{RTP bandwidth}}$
- $T_{rcvr} = \frac{\text{\# receivers * avg RTCP pkt size}}{.25 * .05 * \text{RTP bandwidth}}$

Send at (.5 + rand(0,1)) * T : why?

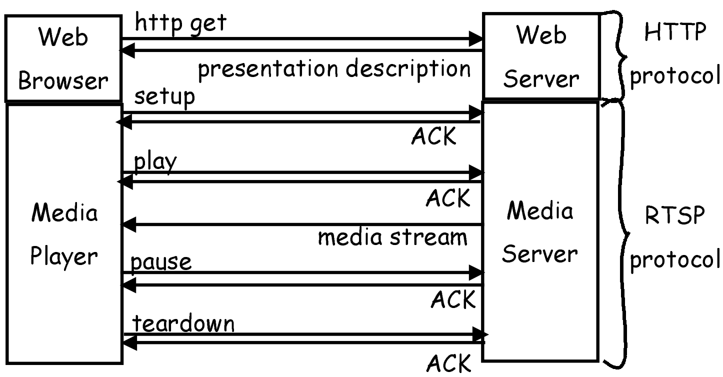
How does each member know # of senders, # rcvrs?

RTCP reconsideration

- Goal: prevent RTCP bandwidth explosion if everybody joins at once
  - Receivers who initially join will count small # of session members
- Solution when first joining:
  1. Compute T, and wait random time interval
  2. At end of interval, reassess # of members
  3. If # of members increased, compute a new T'
  4. If T' < T, send immediately
  5. If T' >= T, wait an additional T', go to step 2
- Other times, use normal wait period

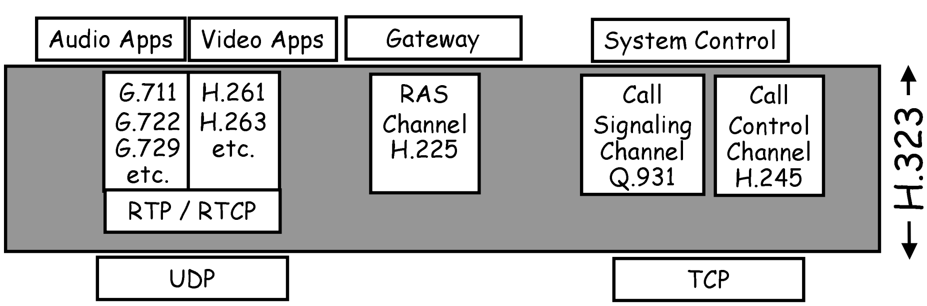
RTSP [RFC 2326]

- RTSP: out-of-band protocol used to control transmission of a media-stream
  - VCR-like functionality (pause/resume, FF, rewind, reposition, etc.)



H.323

- A standard for real-time audio / video teleconferncing on the Internet
- Network Components:
  - **end points**: end-host H.323-compliant app
  - **gateways**: interface between H.323-compliant communication and prior technology (e.g. phone network)
  - **gatekeepers**: provide services at gateway (e.g., address translation, billing, authorization, etc.)



H.323 cont'd

- H.225: notifies gatekeepers of session initiation
- Q.931: signalling protocol for establishing and terminating calls
- H.245: out-of-band protocol negotiates the audio/video codecs used during a session (TCP)