Electrical Engineering E6761
Computer Communication Networks

Lecture 8
Real-Time Internet

Professor Dan Rubenstein
Tues 4:10-6:40, Mudd 1127
Course URL:
http://www.cs.columbia.edu/~danr/EE6761

Overview

- Class schedule adjustments
- HW#3
- Project
  - What's expected
- Lecture: Real-Time communication
  - Basics: jitter, buffering, interleaving, FEC
  - Protocols: RTP/RTCP/RTSP/H.323
  - Congestion Control & Fairness: TCP-friendliness
  - Multicast-specific: rate variation
    - Destination-Set grouping
    - Layering

Class Schedule

- Next week (11/7): no class (Election Day)
- Week after (11/14): no class
  - makeup class (taped) after 11/14?
- I'm out of town from 11/3-11/14
  - If you want to talk to me about your project, you should do it before then…
  - or send me e-mail, but I may not get back to you right away…

Project

- You should have a roadmap of what you will be doing (i.e., a paragraph description) by Thursday
- Due 12/15: ~10 page report
  - basic idea of the project
  - related work (literature survey)
  - results
  - future directions (what you did not do that you would have if there was more time)
- Presentations: maybe (too many groups in class)

Multimedia Networking: Overview

- Application classes
  - streamed stored audio/video
  - one-to-many (multicast) streaming of real-time a/v
  - real-time interactive audio/video
- Multimedia application issues
  - packet jitter
  - packet loss / recovery
- Internet protocols for multimedia
  - RTP/RTCP
  - RTSP
  - H.323
- Multimedia Multicast
  - Destination Set Splitting / Grouping
  - Layering
- TCP-friendly rate adaptation

Lecture Focus

- Today's lecture covers techniques for multimedia implemented at the transport or application layer
- Future lecture: network layer modifications for multimedia (e.g., IntServ, RSVP, Diffserv, revisit queueing, policing, etc.)

Multimedia Application Class

- Typically sensitive to delay, but can tolerate packet loss (would cause minor glitches that can be concealed)
- Data contains audio and video content ("continuous media"), three classes of applications:
  - Streaming
  - Unidirectional Real-Time
  - Interactive Real-Time
- Each class might be broadcast (multicast) or may simply be unicast

Application Classes (more)

- **Streaming**
  - Clients request audio/video files from servers and pipeline reception over the network and display
  - Interactive: user can control operation (similar to VCR: pause, resume, fast forward, rewind, etc.)
  - Delay: from client request until display start typically 1 to 10 seconds
  - e.g., CVN's on-line video transmission of this course!!

Application Classes (more)

- **Unidirectional Real-Time:**
  - similar to existing TV and radio stations, but delivery on the network
  - Non-interactive, just listen/view
- **Interactive Real-Time :**
  - Phone conversation or video conference
  - More stringent delay requirement than Streaming and Unidirectional because of real-time nature
  - Video: < 150 msec acceptable
  - Audio: < 150 msec good, <400 msec acceptable

9

Challenges

- TCP/UDP/IP suite provides best-effort, no guarantees on expectation or variance of packet delay
- Streaming applications delay of 5 to 10 seconds is typical and has been acceptable, but performance deteriorate if links are congested (transoceanic)
- Real-Time Interactive requirements on delay and its jitter have been satisfied by over-provisioning (providing plenty of bandwidth) or "unfair" use of bandwidth, what will happen when the load increases?...

10

Challenges (more)

- Most router implementations use only First-Come-First-Serve (FCFS) packet processing and transmission scheduling
- To mitigate impact of "best-effort" protocols, we can:
  - Use UDP to avoid TCP's rate control
  - Buffer content at client and control playback to remedy jitter
  - Adapt compression level to available bandwidth

11

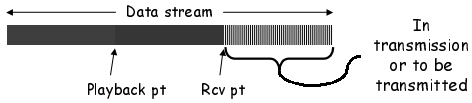
Solution Approaches in IP Networks

- Just add more bandwidth and enhance caching capabilities (over-provisioning)!
- Need major change of the protocols :
  - Incorporate resource reservation (bandwidth, processing, buffering), and new scheduling policies
  - Set up service level agreements with applications, monitor and enforce the agreements, charge accordingly
- Make changes in routing policy (i.e., not just best-effort FIFO)

12

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



13

Streaming

- Important and growing application due to reduction of storage costs, increase in high speed net access from homes, enhancements to caching and introduction of QoS in IP networks
- Audio/Video file is segmented and sent over either TCP or UDP, public segmentation protocol: **Real-Time Protocol (RTP)**

14

Streaming

- User interactive control is provided, e.g. the public protocol **Real Time Streaming Protocol (RTSP)**
- **Helper Application**: displays content, which is typically requested via a Web browser; e.g. RealPlayer; typical functions:
  - Decompression
  - Jitter removal
  - Error correction / loss recovery: use redundant packets to be used for reconstruction of original stream
  - GUI for user control

15

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?

- don't need the high level of reliability
- rate can slow down "too much"

16

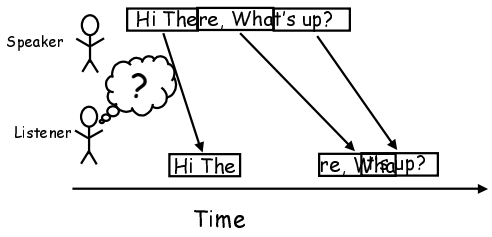
Mmedia Transmission Challenges and Solutions

- Jitter
  - buffering, time-stamps
- Packet loss
  - loss-tolerant apps
  - Interleaving
  - retransmission (ARQ) or Packet-Level Forward Error Correction (FEC)
- Single-rate Multicast
  - Destination Set Splitting
  - Layering

17

Jitter

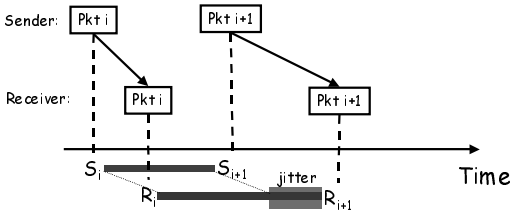
- The Internet makes no guarantees about time of delivery of a packet
- Consider an IP telephony session:



18

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: $(R_{i+1} - R_i) - (S_{i+1} - S_i)$

19

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...

20

Real-Time (Phone) Over IP's Best-Effort

- Internet phone applications generate packets during talk spurts
- Bit rate is 8 KBs, and every 20 msec, the sender forms a packet of 160 Bytes + a header to be discussed below
- The coded voice information is encapsulated into a UDP packet and sent out
  - some packets may be lost; up to 20 % loss is tolerable;
  - using TCP eliminates loss but at a considerable cost: variance in delay;
  - FEC (Forward Error Correction) is sometimes used to fix errors and make up losses

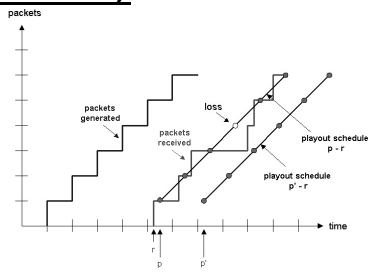
21

Real-Time (Phone) Over IP's Best-Effort

- End-to-end delays above 400 msec cannot be tolerated; packets that are that delayed are ignored at the receiver
- Delay jitter is handled by using timestamps, sequence numbers, and delaying playout at receivers either a fixed or a variable amount
- With fixed playout delay, the delay should be as small as possible without missing too many packets; delay cannot exceed 400 msec

22

Internet Phone with Fixed Playout Delay



23

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

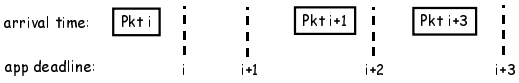
24

Adaptive Playout Delay

- Objective is to use a value for p-r that tracks the network delay performance as it varies during a phone call
- The playout delay is computed for each talk spurt based on observed average delay and observed deviation from this average delay
- Estimated average delay and deviation of average delay are computed in a manner similar to estimates of RTT and deviation in TCP
- The beginning of a talk spurt is identified from examining the timestamps in successive and/or sequence numbers of chunks

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, …

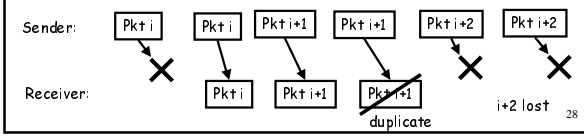
- Solution step 1: Design app to tolerate **some** loss
- Solution step 2: Design techniques to recover **some** lost packets within application's time limits

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

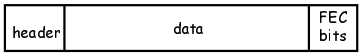
Reducing loss w/in time bounds

- Problem: packets must be recovered prior to application deadline
- Solution 1: extend deadline, buffer @ rcvr, use ARQ (Automatic Repeat Request: i.e., ACKs & NAKs)
  - Recall: unacceptable for many apps (e.g., interactive)
- Solution 2: Forward Error Correction (FEC)
  (Technically: we are using **Erasure Codes**, not FEC codes)
  - Send "repair" before a loss is reported
  - Simplest FEC: transmit redundant copies

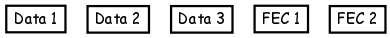


More advanced FEC techniques

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

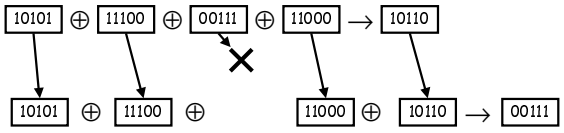


- Here, we will consider using FEC (really Erasure Codes) at the packet layer (special repair packets):



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 $\Re$

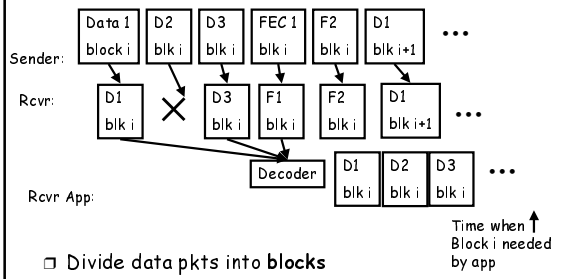
Reed Solomon Example over $\Re$

Pkt 1: Data1
Pkt 2: Data2
Pkt 3: Data3
} Original data

Pkt 4: Data1 + Data2 + 2 Data3
Pkt 5: 2 Data1 + Data2 + 3 Data3
} Special linear combinations

- 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

- Media-specific approach
- Packet contents:
  - high quality version of media frame k
  - low quality version of media 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

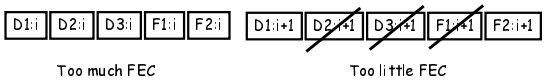
i	low: k-c	high: k
i+1	low: k-c+1	high: k+1
i+2	low: k-c+2	high: k+2

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!!!

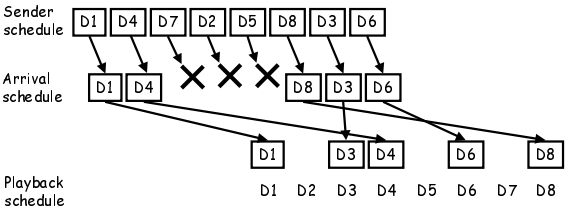
Bursty Loss:

- 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 noticeable
- FEC provides less benefit in a bursty loss scenario (e.g., consider 30% loss in bursts 3 packets long)



Interleaving

- To reduce effects of burstiness, reorder pkt transmissions

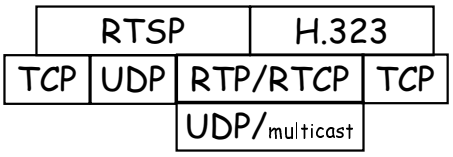


- Drawback: induces buffering and playout delay

37

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



38

RTP/RTCP [RFC 1889 by Prof. Schulzrinne et al]

- General purpose Real-Time Multimedia protocol
  - Scalable to large sessions (many senders, receivers)
- 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)

39

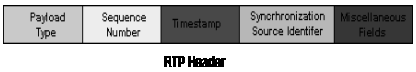
RTP/RTCP details

- 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

40

Real-Time Protocol (RTP)

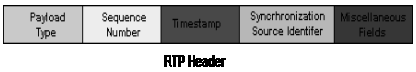
- Provides standard packet format for real-time application
- Typically runs over UDP
- Specifies header fields below
- **Payload Type**: 7 bits, providing 128 possible different types of encoding; eg PCM, MPEG2 video, etc.
- **Sequence Number**: 16 bits; used to detect packet loss



RTP Header

Real-Time Protocol (RTP)

- **Timestamp**: 32 bytes; gives the sampling instant of the first audio/video byte in the packet; used to remove jitter introduced by the network
- **Synchronization Source identifier (SSRC)**: 32 bits; an id for the source of a stream; assigned randomly by the source



RTP Header

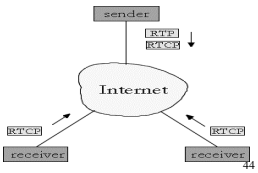
RTP header

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? (Hint: ever showed up late to a movie?)
- Transmission rate: application specific (no congestion control specified in RTP)

RTP Control Protocol (RTCP)

- Protocol specifies report packets exchanged between sources and destinations of multimedia information
- Three reports are defined: Receiver reception, Sender, and Source description
- Reports contain statistics such as the number of packets sent, number of packets lost, inter-arrival jitter
- Used by application to modify sender transmission rates and for diagnostics purposes



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

- Simple rule: 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
- $T_{sender} = \frac{\text{\# senders * avg RTCP pkt size}}{\text{.25 * .05 * RTP bandwidth}}$
- $T_{rcvr} = \frac{\text{\# receivers * avg RTCP pkt size}}{\text{.25 * .05 * RTP bandwidth}}$

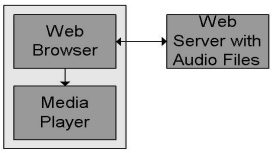
Send at (.5 + rand(0,1)) * $T_{[sender|rcvr]}$: why?
How does each member know # of senders, # rcvrs?

RTCP reconsideration

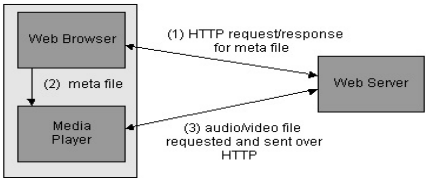
- Goal: prevent RTCP bandwidth explosion if everybody joins at once
  - Receivers who initially join will count small # of session members
- Solution when first joining:
  1. Compute T, and wait random time interval
  2. At end of interval, reassess # of members
  3. If # of members increased, compute 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

Streaming From Web Servers

- Audio: in files sent as HTTP objects
- Video (interleaved audio and images in one file, or two separate files and client synchronizes the display) sent as HTTP object(s)
- A simple architecture is to have the Browser requests the object(s) and after their reception pass them to the player for display
  - No pipelining



Meta file requests



51

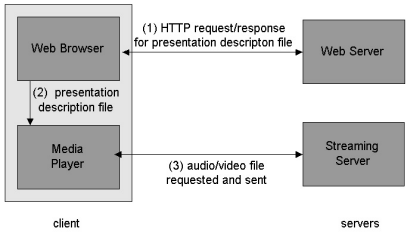
Streaming From Web Server (more)

- Alternative: set up connection between server and player, then download
- Web browser requests and receives a **Meta File** (a file describing the object) instead of receiving the file itself;
- Browser launches the appropriate Player and passes it the Meta File;
- Player sets up a TCP connection with Web Server and downloads the file

50

Using a Streaming Server

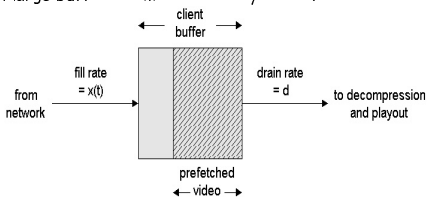
- This gets us around HTTP, allows a choice of UDP vs. TCP and the application layer protocol can be better tailored to Streaming; many enhancements options are possible (see next slide)



52

Options When Using a Streaming Server

- Use UDP, and Server sends at a rate (Compression and Transmission) appropriate for client; to reduce jitter, Player buffers initially for 2-5 seconds, then starts display
- Use TCP, and sender sends at maximum allowable rate under TCP; retransmit when error is encountered; Player uses a much large buffer to smooth delivery rate of TCP



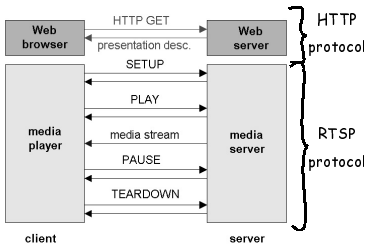
Real Time Streaming Protocol (RTSP)

- For user to control display: rewind, fast forward, pause, resume, etc…
- Out-of-band protocol (uses two connections, one for control messages (Port 554) and for media stream)
- RFC 2326 permits use of either TCP or UDP for the control messages connection, sometimes called the RTSP Channel
- As before, meta file is communicated to web browser which then launches the Player;
- Player sets up an RTSP connection for control messages in addition to the connection for the streaming media

Meta File Example

```
<title>Twister</title>
<session>
      <group language=en lipsync>
            <switch>
               <track type=audio
                     e="PCMU/8000/1"
                     src =
   "rtsp://audio.example.com/twister/audio.en/lofi">
               <track type=audio
                     e="DVI4/16000/2" pt="90 DVI4/8000/1"
   src="rtsp://audio.example.com/twister/audio.en/hifi">
            </switch>
         <track type="video/jpeg"
                     src="rtsp://video.example.com/twister/video">
      </group>
</session>
```


RTSP Operation



RTSP Exchange Example

C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK
Session 4231

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=0-

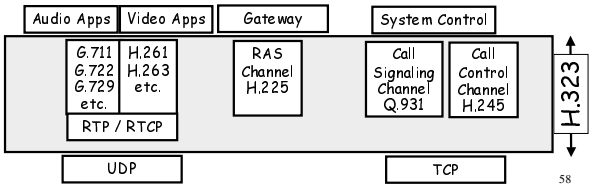
C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=37

C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231

S: 200 3 OK

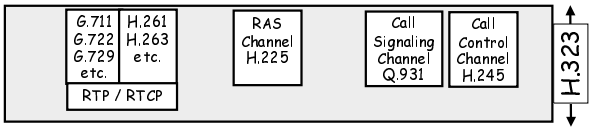
H.323

- 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)

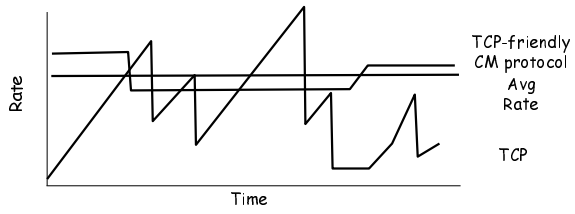


TCP-friendly CM transmission

- Idea: Continuous-media protocols should not use more than their "fair share" of network bandwidth
- Q: What determines a fair share
- One possible answer: TCP could
  - TCP's rate is a function of RTT & loss rate p
  - $\text{Rate}_{TCP} \approx 1.3 / (\text{RTT}\sqrt{p})$ (for "normal" values of p)
  - Over a long time-scale, make the CM-rate match the formula rate

TCP-friendly Congestion Control

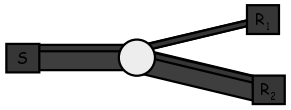
- Average rate same as TCP travelling along same data-path (rate computed via equation), but CM protocol has less rate variance



61

Single-rate Multicast

- In IP Multicast, each data packet is transmitted to all receivers joined to the group
- Each multicast group provides a single-rate stream to all receivers joined to the group

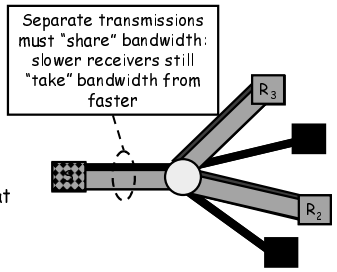


- R_2's rate (and hence quality of transmission) forced down by "slower" receiver R_1
- How can receivers in same session receive at differing rates?

62

Multi-rate Multicast: Destination Set Splitting

- Place session receivers into separate multicast groups that have approximately same bandwidth requirements
- Send transmission at different rates to different groups

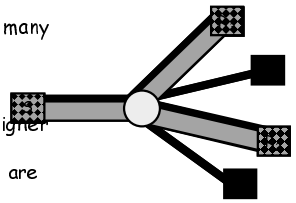


Happy Halloween!!!

63

Multi-rate Multicast: Layering

- Encode signal into layers
- Send layers over separate multicast groups
- Each receiver joins as many layers as links on its network path permit
- More layers joined = higher rate
- Unanswered Question: are layered codecs less efficient than unlayered codecs?



64

Next time (11/21)

- Network Fairness & Pricing or Network Measurement & Inference (or both)

65