Streaming Audio and Video

CS 360 Internet Programming

Daniel Zappala

Brigham Young University
Computer Science Department

Types of Streaming

- stored audio and video
  - content pre-recorded and stored at server
  - user begins playback before entire file received
  - content played continuously, at same rate as original recording
  - user can pause, rewind, fast-forward, index content
- live audio and video
  - content is sent live, rather than pre-recorded, still continuous
  - higher expectation that delays are short, no pauses unless user requests it
- real-time interactive audio and video
  - user interaction - conversations
  - highest expectation for low delay – must be below human thresholds for interactive conversation to be possible
  - delay < 150 *ms* not perceptible, $150 - 400$ *ms* acceptable, > 400 *ms* not tolerable

Challenge

- the Internet offers only best-effort service
  - no guarantee on delay (affects live and interactive streaming)
  - no guarantee on throughput (affects video streaming)
- **multimedia applications are typically delay sensitive but loss tolerant**
- data applications are typically loss sensitive but delay tolerant

Internet Phone Example

- streaming audio: alternating talk spurts, silent periods
  - packets generated only during talk spurts
  - talk spurt = 64 kbps
  - sample every 20 ms: 160 bytes per packet
  - send packets using UDP
- ideally, during a talk spurt a packet arrives every 20ms and you play it
- loss: will make the stream pause or sound choppy
- delay: must be $< 400\ ms$, $< 150\ ms$ is ideal
- jitter: packets experience different amounts of queueing delay

Quality of Service

- performance guarantees: bandwidth, delay, jitter
- **how will the Internet support QoS?**
- do nothing
  - use application-level techniques to mitigate effects of delay and jitter
  - add more bandwidth as needed
  - use CDNs for stored content
- differentiated service
  - classify and treat some traffic differently
  - must be careful of network neutrality – treat all voice or video the same (even your competitor's), unless the user has paid for special treatment

Coping with the Internet

- applications make the best out of best-effort service
  - compression
  - client buffering
  - UDP + error correction or plain TCP
  - multiple encodings – select the best that can be delivered
  - adaptive HTTP streaming
- media player responsibilities
  - remove jitter
  - decompress stream
  - conceal errors
  - user controls: pause, rewind, fast forward

Audio Compression

1. analog signal sampled at constant rate
   - telephone: 8,000 samples/second
   - CD music: 44,100 samples/second
2. each sample quantized (rounded)
   - for example 256 possible values
3. each quantized value represented by bits
   - for example 8 bits for 256 values
   - 9,000 samples/second, 256 quantized values : 64,000 bps
4. receiver converts digital data back to analog signal, some quality reduction

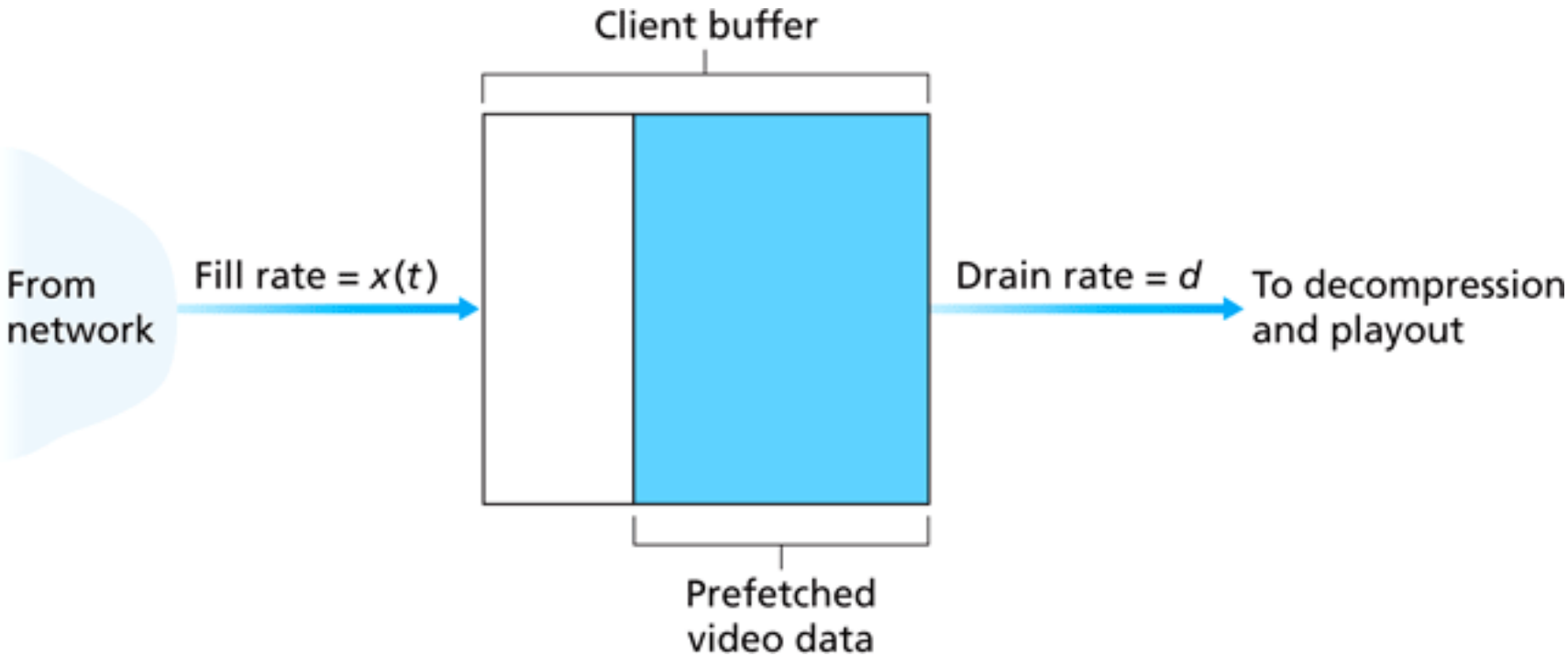
- typical rates
  - CD: 1.411 Mbps
  - MP3: 96, 128, 160 kbps
  - VoIP: 14 kbps

Video Compression

- video is a sequence of images displayed at a constant rate
  - for example 24 frames/second
- each frame is divided into an array of pixels
- each pixel is represented by some bits
- can eliminate spatial and temporal redundancy in and between frames
- example video rates
  - MPEG 1 (CD-ROM): 1.5 Mbps
  - MPEG 2 (DVD): 3 - 6 Mbps
  - MPEG 4 (Internet): $<$ 1 Mbps
- layered video
  - hierarchical encoding into layers
  - additional layers produce increasing resolution
  - send number of layers that path to receiver can handle

Client Buffering

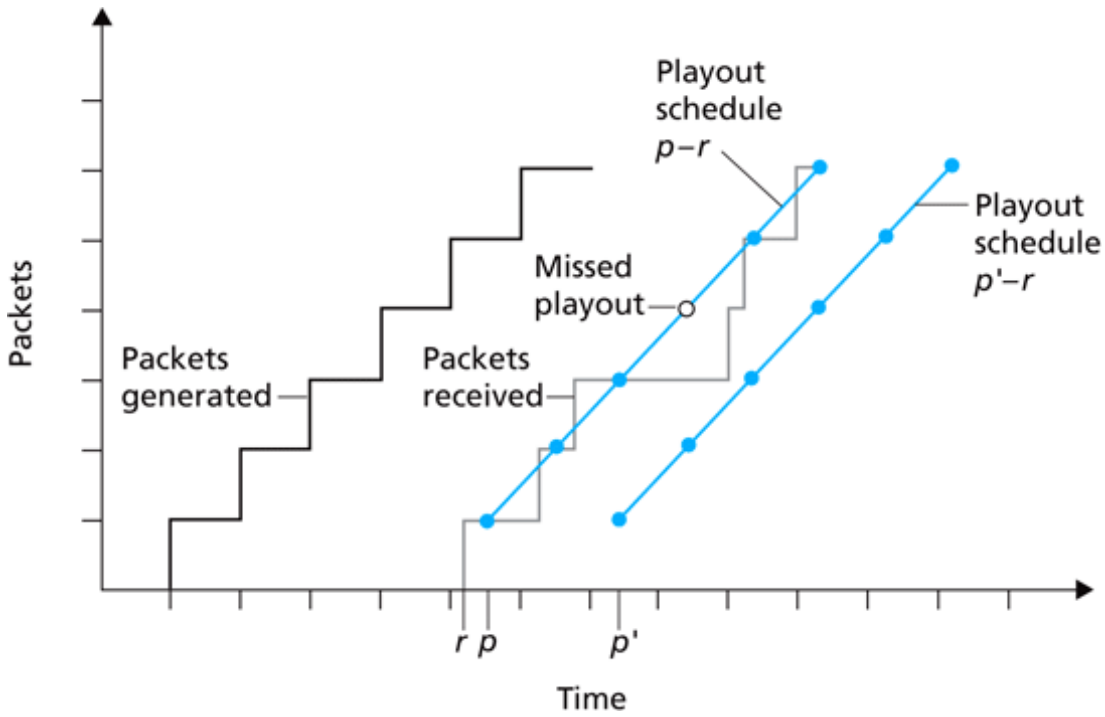
- use buffering to compensate for delay, jitter
  - play packets from the buffer at a certain rate
  - want to prevent draining the buffer too early
  - want to prevent delaying the packets too much: 400 ms maximum, but less is better



UDP vs TCP

- UDP
  - server sends at rate appropriate for client (often ignores congestion control)
    - send rate = encoding rate = constant rate
    - fill rate = constant rate - packet loss
  - short playout delay (2-5 seconds) to compensate for network delay jitter
  - recover errors if time permits
- TCP
  - send at maximum possible rate under TCP: congestion control
  - fill rate fluctuates due to TCP congestion control
  - need a larger playout delay to smooth TCP delivery rate
  - HTTP/TCP passes more easily through firewalls

Fixed Playout Delay



- play packet i at $t_i + q_i$
  - t_i: time the packet was generated
  - small q: better real-time interaction
  - large q: fewer missed playouts

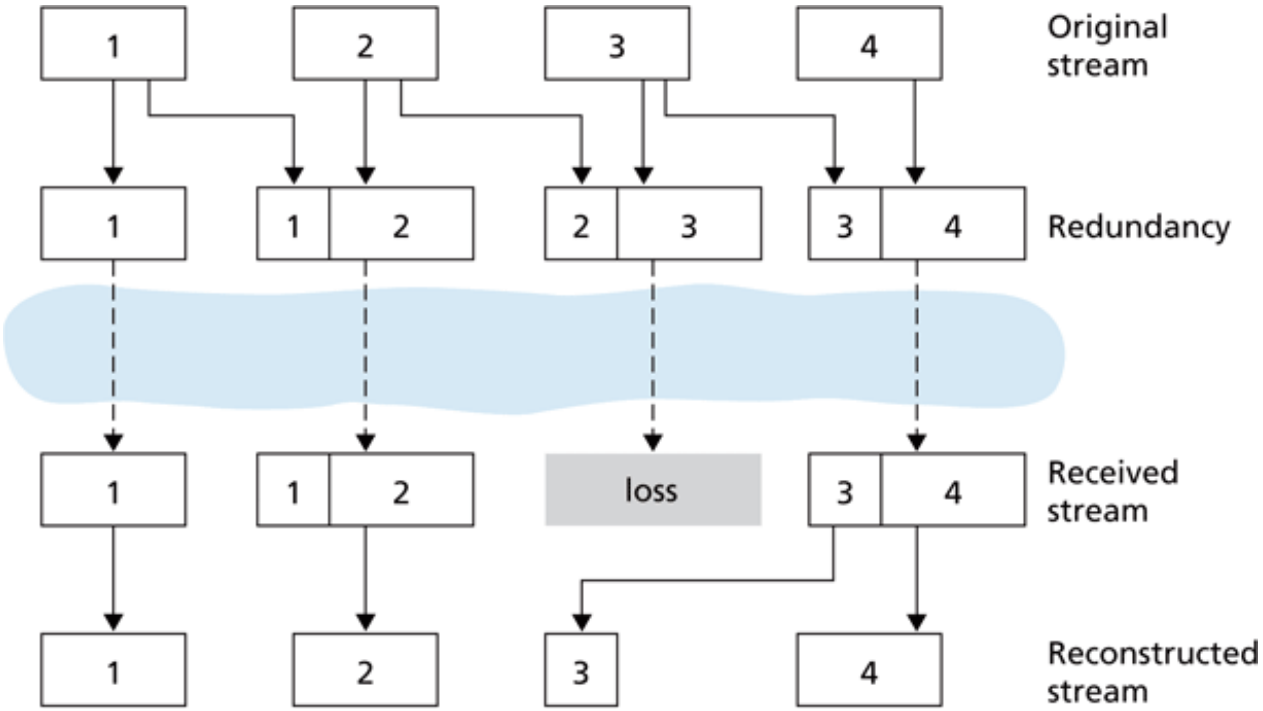
Adaptive Playout Delay

- goal: minimize playout delay, infrequent missed playouts
- for start of talk spurt, play packet i at $p_i = t_i + d_i + Kv_i$
  - d_i = estimate of delay (EWMA)
  - v_i = estimate of delay variation (EWMA)
  - K = constant, e.g. K = 4
- for other packets, $p_j = t_j + p_i - t_i$
  - use the same offset from t as beginning of talk spurt
  - application may denote start of talk spurt
- because playout delay is adaptive, silent periods may be compressed or elongated

Forward Error Correction (FEC)

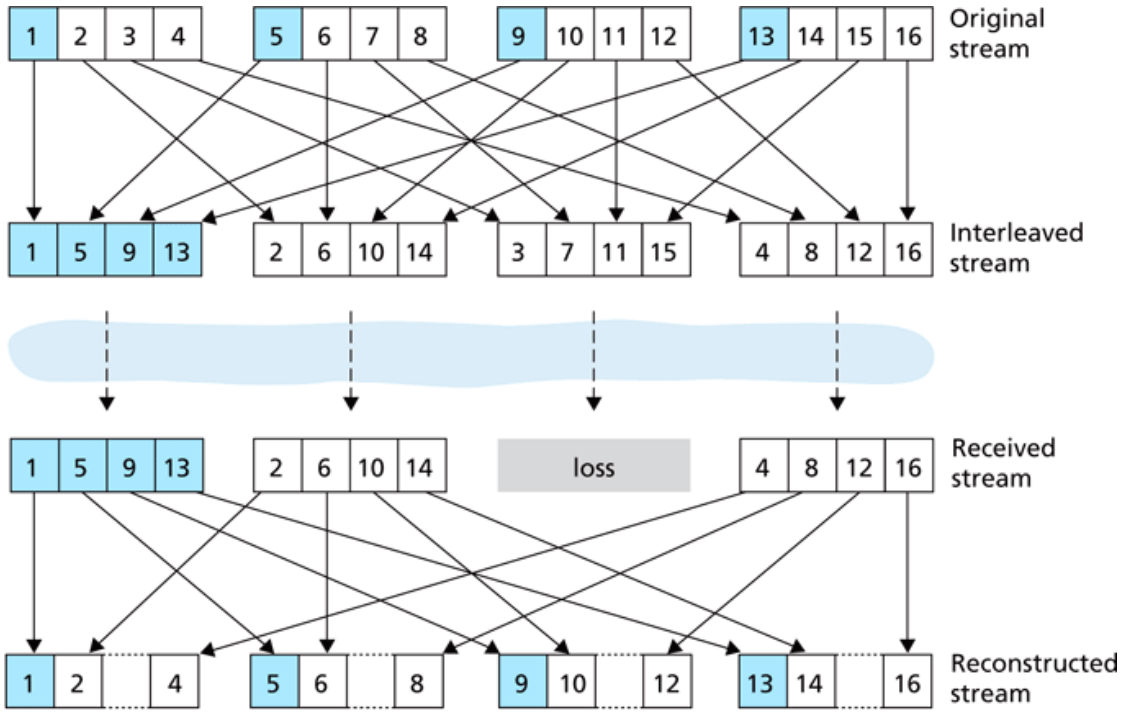
- add some redundant data to the stream
  - for every group of n packets, create a redundant packet: exclusive-OR of the n original packets
  - send $n+1$ packets, increasing bandwidth by $1/n$
- receiver can reconstruct the stream with any n packets
  - must wait for n packets before playout
- trade-offs
  - larger n: less bandwidth
  - smaller n: shorter playout delay, smaller chance of two packets out of n being lost
- simple version of FEC: see RFC 2733

FEC With Lower Quality Redundant Data



- add lower-resolution audio to each packet
- substitute lower quality when needed: Free Phone, RAT
- many other more complex kinds of FEC available

Interleaving



- interleave smaller (5 ms) pieces among packets
- if a packet is lost, still have most of the stream
- no redundancy overhead, but added playout delay

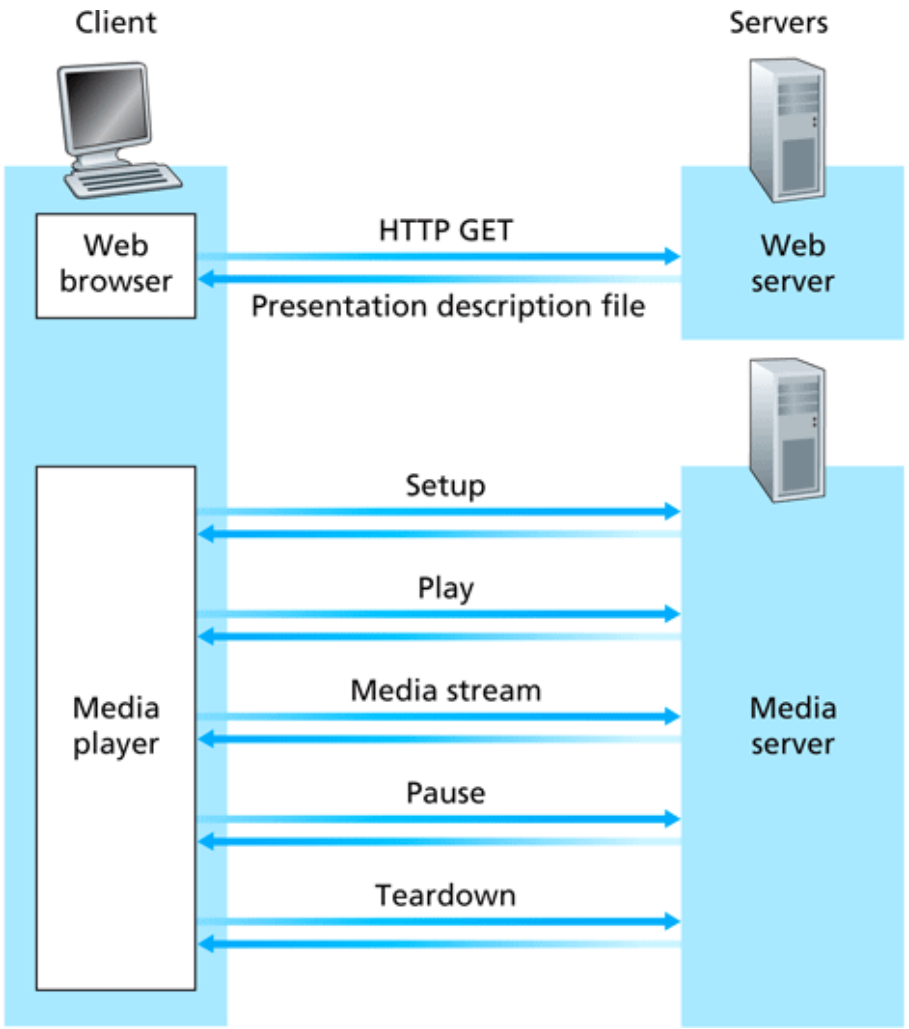
Adaptive HTTP Streaming

- used for streaming stored video (TV shows, movies)
  - Move Networks
  - Microsoft (Smooth Streaming in Silverlight)
  - Adobe (Flash)
  - Apple
- architecture
  - break video into 2 second segments
  - create multiple encodings (resolutions) of each segment
  - store segments on CDNs near users
  - video player downloads the best quality it can in 2 seconds, so it is ready to play for the next 2 second window
  - uses TCP and HTTP
- no buffering
- instantaneous switching to a new part of the movie

RTSP: User Control

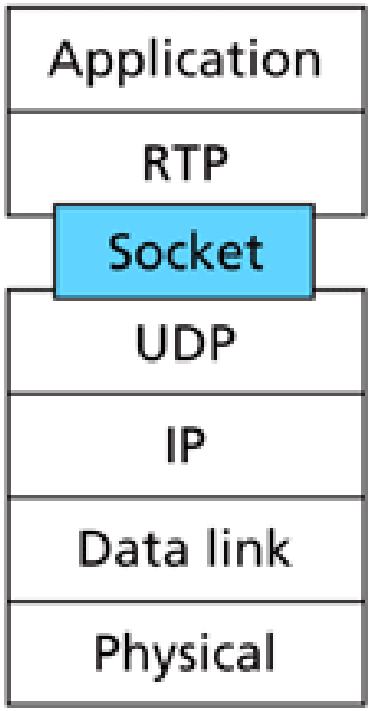
- RFC 2326
- client-server, application layer protocol
- user can pause, rewind, fast forward, index, etc
- does not define
  - how media is streamed over network: packet format
  - transport protocol used
  - client buffering
- messages are *out-of-band* with respect to the streaming media: port 554

RTSP Example



RTP: Real-Time Protocol

- specifies packet structure for audio and video streams
  - payload identification
  - sequence number
  - timestamp
- used for interoperability between multimedia applications
- runs on top of UDP
- RFC 1889

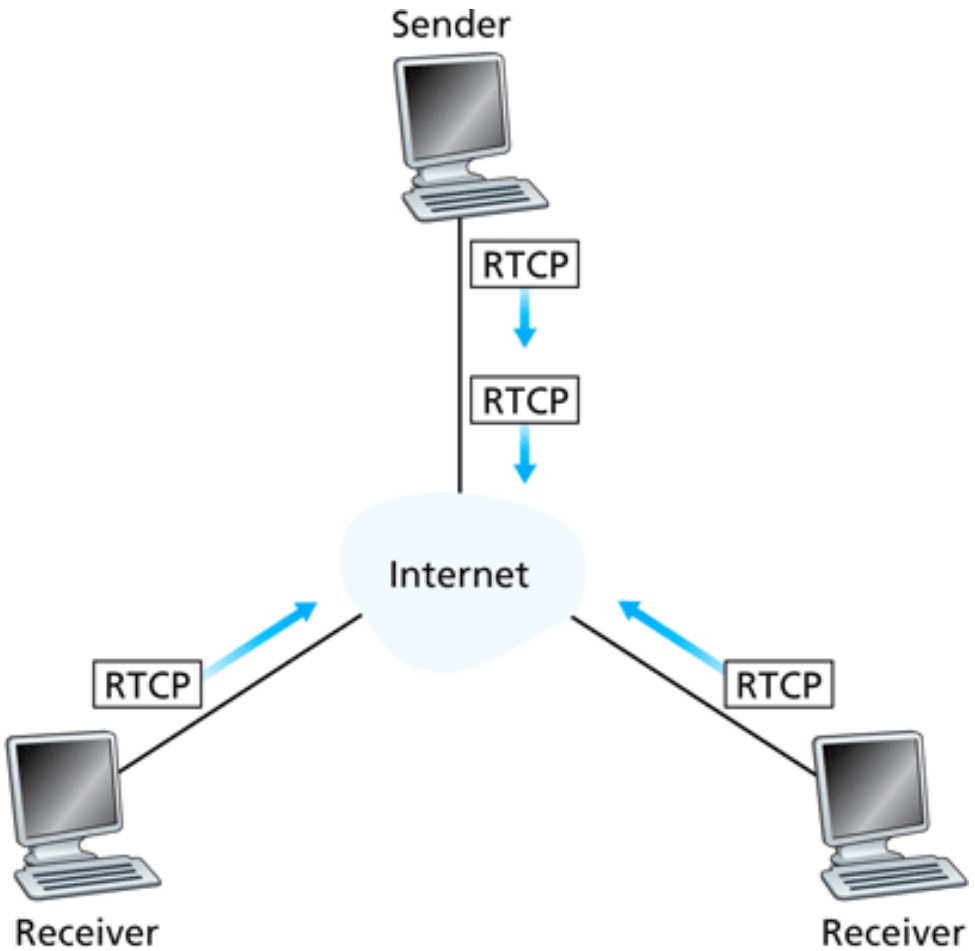


RTCP: Real-Time Control Protocol

- works with RTP
- each participant in RTP session periodically transmits RTCP control packets to other participants
  - statistics from sender and receiver on session quality
- use feedback to control performance, e.g. congestion control, type of encoding
- packets sent using same mechanism as data (unicast or multicast)

RTCP Packets

- source description: sender's name, host name, email address, IP of associated RTP stream
- sender report: IP of RTP stream, current time, number of packets sent, number of bytes sent
- receiver report: fraction of packets lost, last sequence number, average jitter



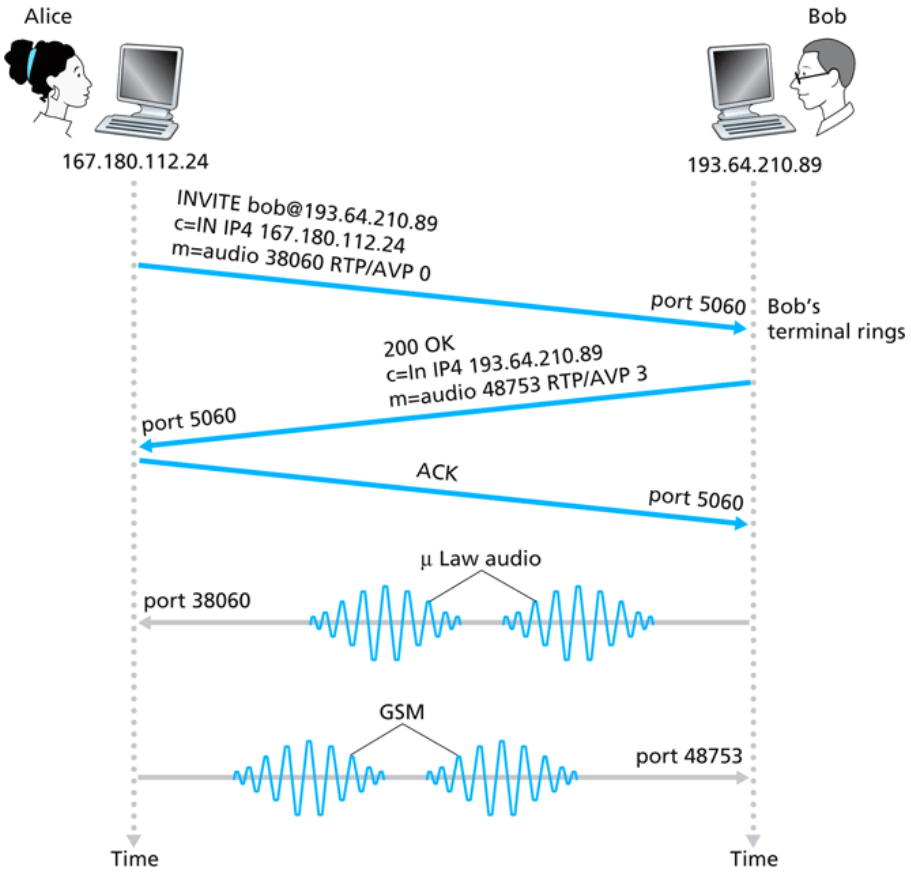
RTCP Extras

- synchronization of different media streams in an RTP session
  - timestamps in audio and video streams tied to sampling clocks, not wall-clock time
  - each RTCP sender report contains timestamp of most recent RTP packet and wall-clock time of when it was created
- bandwidth scaling
  - RTCP will limit control traffic to 5%
  - e.g. if sender transmits video at 2 Mbps, then RTCP tries to use 100 kbps
  - 75% of rate for receivers, 25% to sender
  - receivers try to share 75% equally
  - receiver determines RTCP transmission period by dividing average RTCP packet size by allocated rate

SIP: Session Initiation Protocol

- vision
  - all telephone and video conference calls take place over the Internet
  - people are identified by names and email addresses rather than by phone numbers
  - you can reach the person you are calling wherever she is on the Internet
- call setup
  - start and end call
  - agree on media type and encoding
- map name and email to IP address
- call management
  - add new streams during call
  - change encoding during call
  - invite other users to call
  - transfer and hold calls

SIP Call Setup



- example assumes Alice knows Bob's IP address
- Alice provides port number, IP address, preferred encoding
- Bob responds with new port number, IP address, preferred encoding
- SIP messages can use TCP or UDP, default port 5060

SIP Extras

- encoding negotiation
  - may not have requested encoder
  - may prefer a different one
  - respond with **606 Not Acceptable** and list encoders
  - sender can send a new invitation with new encoder
- can reject a call
  - *busy*, *gone*, *payment required*, *forbidden*
- media can use RTP or any other protocol
- syntax is similar to HTTP

Name Translation and User Location

- need to map user name or email address to IP address
  - mobility
  - changing IP addresses due to DHCP
  - many different IP devices per user
  - call forwarding
- SIP registrar: clients register to provide current location and IP address (similar to instant messaging)
- SIP proxy: find callee on behalf of caller (similar to DNS server)

Session Initiation Example