Multimedia Networking

Computer Networks, WS 2018/19

Lecturers: Prof. Xiaoming Fu, Dr. Yali Yuan
Assistant: Yachao Shao, MSc

Go Back N/ Selective Repeat Animation

- http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

Chapter 5 outline

- 5.1 Multimedia networking applications
- 5.2 Streaming stored audio and video
- 5.3 Making the best out of best effort service
- 5.4 Protocols for real-time interactive applications
  - RTP,RTCP,SIP

MM Networking Applications

Fundamental characteristics

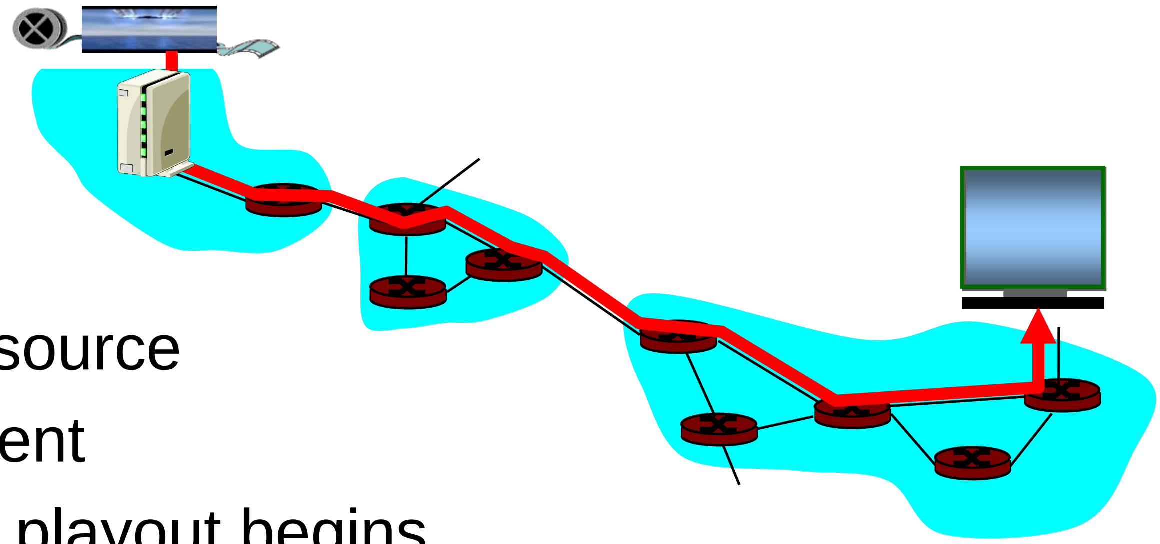
- typically delay sensitive
  - end-to-end delay
  - delay jitter
- loss tolerant: infrequent losses cause minor glitches
- antithesis of data, which are loss intolerant but delay tolerant.

Classes of MM applications

- 1) stored streaming
- 2) live streaming
- 3) interactive, real-time

Jitter is the variability of packet delays within the same packet stream

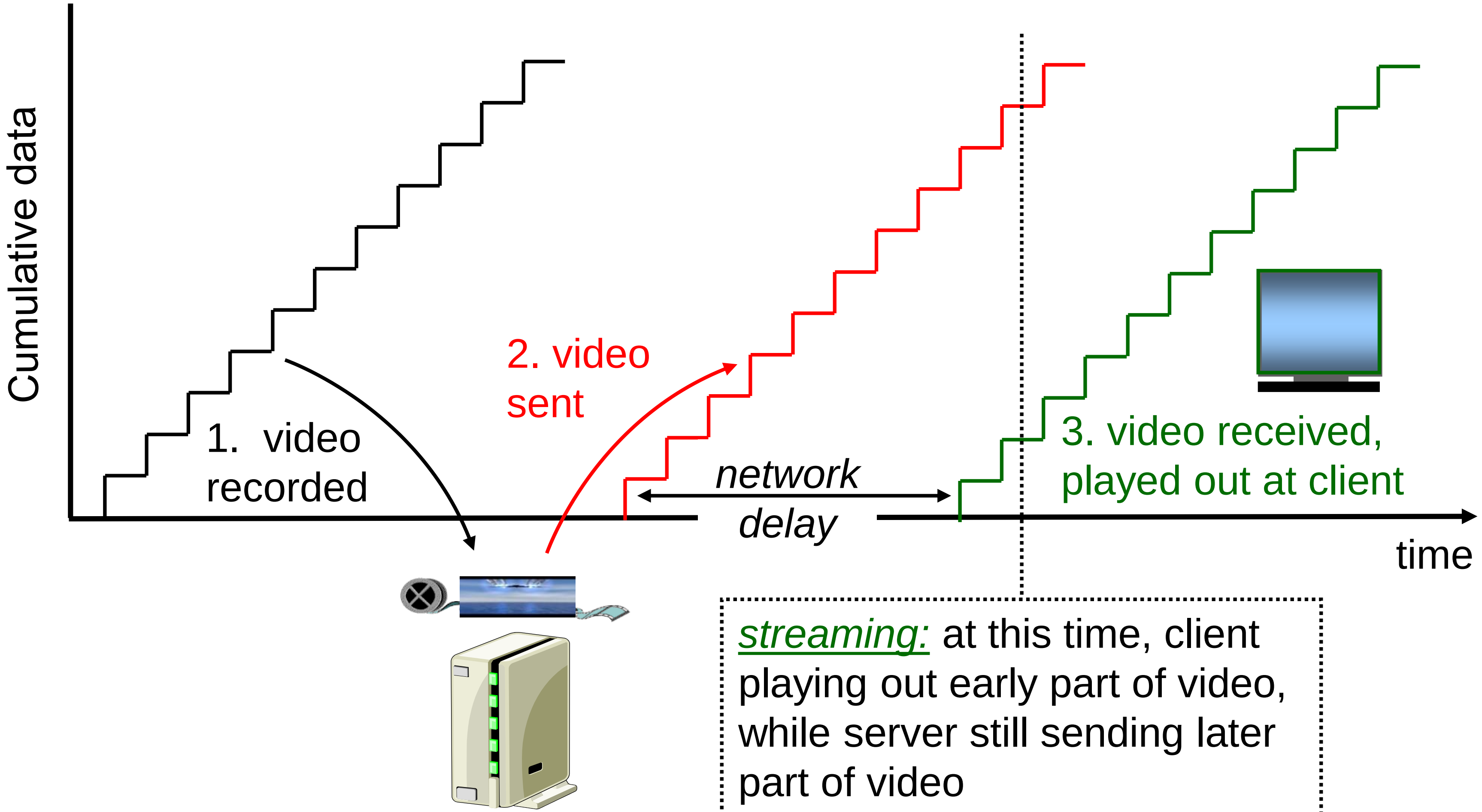
Streaming Stored Multimedia



Stored streaming:

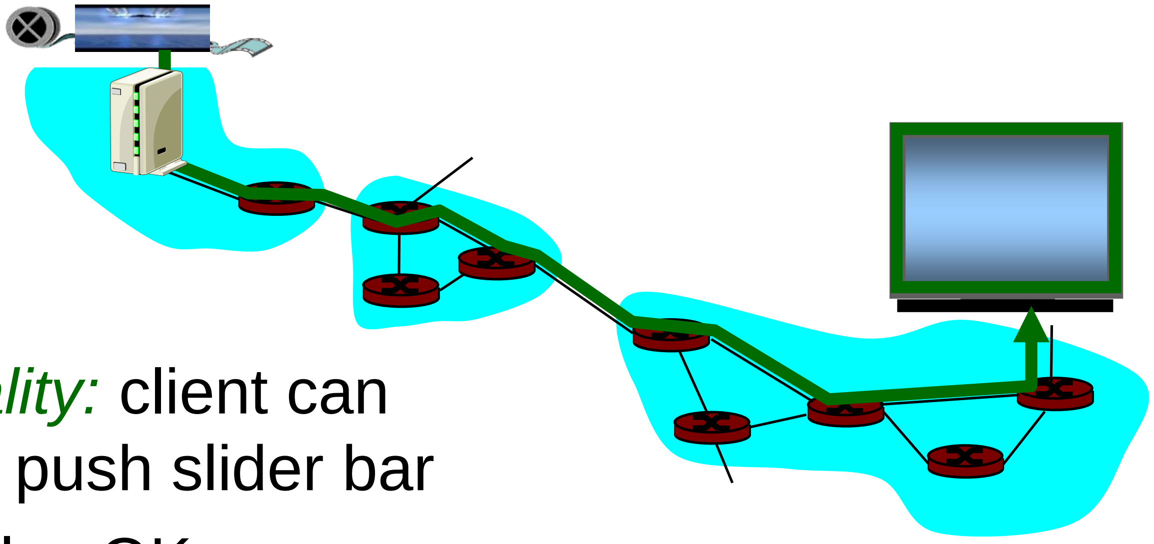
- media stored at source
- transmitted to client
- *streaming:* client playout begins *before* all data has arrived
  - timing constraint for still-to-be transmitted data: in time for playout

Streaming Stored Multimedia: What is it?



streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming *Stored* Multimedia: Interactivity



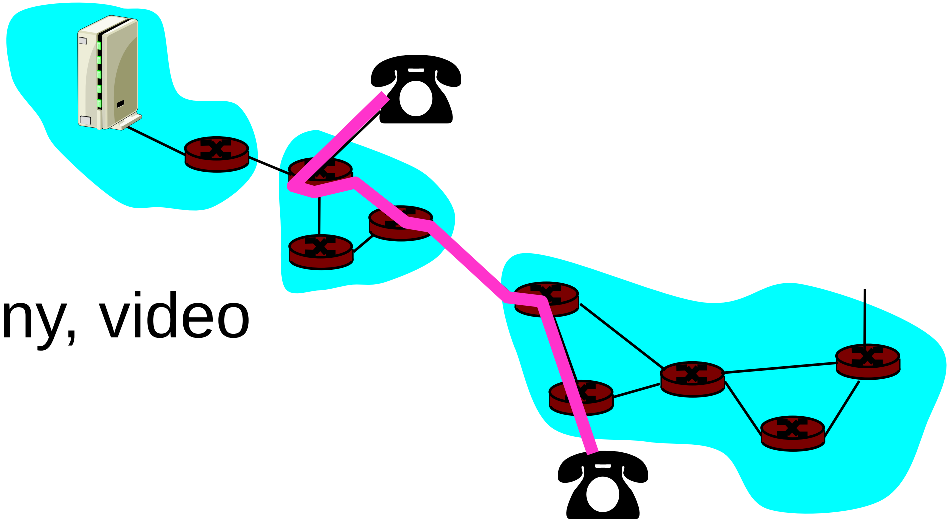
- *VCR-like functionality:* client can pause, rewind, FF, push slider bar
  - 10 sec initial delay OK
  - 1-2 sec until command effect OK

- timing constraint for still-to-be transmitted data: in time for playout

Streaming *Live* Multimedia

- Examples:
  - Internet radio talk show
  - live sporting event
- Streaming (as with streaming stored multimedia)
  - playback buffer
  - playback can lag tens of seconds after transmission
  - still have timing constraint
- Interactivity
  - fast forward impossible
  - rewind, pause possible!

Real-Time Interactive Multimedia



- applications: IP telephony, video conference, distributed interactive worlds
- end-end delay requirements:
  - audio: < 150 msec good, < 400 msec OK
    - includes application-level (packetization) and network delays
    - higher delays noticeable, impair interactivity
- session initialization
  - how does callee advertise its IP address, port number, encoding algorithms?

A few words about audio compression

- analog signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
  - e.g., 2^8=256 possible quantized values
- each quantized value represented by bits
  - 8 bits for 256 values
- example: 8,000 samples/sec, 256 quantized values --> 64,000 bps
- receiver converts bits back to analog signal:
  - some quality reduction

Example rates

- CD: 1.411 Mbps
- MP3: 128, 320, … kbps
- Internet telephony: 5.3 kbps and up

A few words about video compression

- video: sequence of images displayed at constant rate
  - e.g. 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- redundancy
  - spatial (within image)
  - temporal (from one image to next)

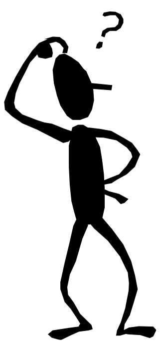
Examples:

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)
- H.265/VP9: Scaling from 240p to 8K (0.2 to >20Mbps; full HD ~ 10Mbps in VP9 on YouTube)

Multimedia Over Today's Internet

TCP/UDP/IP: "best-effort service"

- *no* guarantees on delay, loss



But you said multimedia apps require QoS and level of performance to be effective!

Today's Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

How should the Internet evolve to better support multimedia?

Integrated services philosophy:

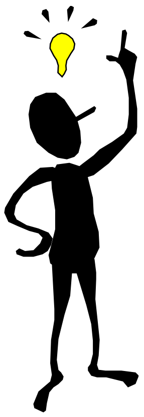
- fundamental changes in Internet so that apps can reserve end-to-end bandwidth
- requires new, complex software in hosts & routers

Laissez-faire

- no major changes
- more bandwidth when needed
- content distribution, application-layer multicast
  - application layer

Differentiated services philosophy:

- fewer changes to Internet infrastructure, yet provide 1st and 2nd class service



What's your opinion?

Chapter 5 outline

- 5.1 Multimedia networking applications
- 5.2 Streaming stored audio and video
- 5.3 Making the best out of best effort service
- 5.4 Protocols for real-time interactive applications
  - RTP,RTCP,SIP

Streaming Stored Multimedia

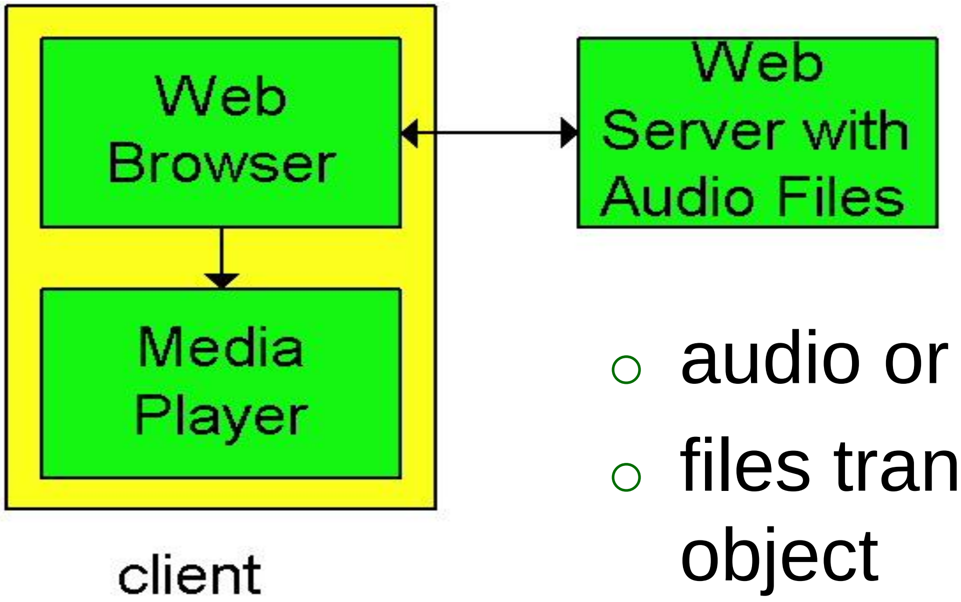
application-level streaming techniques for making the best out of best effort service:

- client-side buffering
- use of UDP versus TCP
- multiple encodings of multimedia

Media Player

- jitter removal
- decompression
- error concealment
- graphical user interface w/ controls for interactivity

Internet multimedia: simplest approach

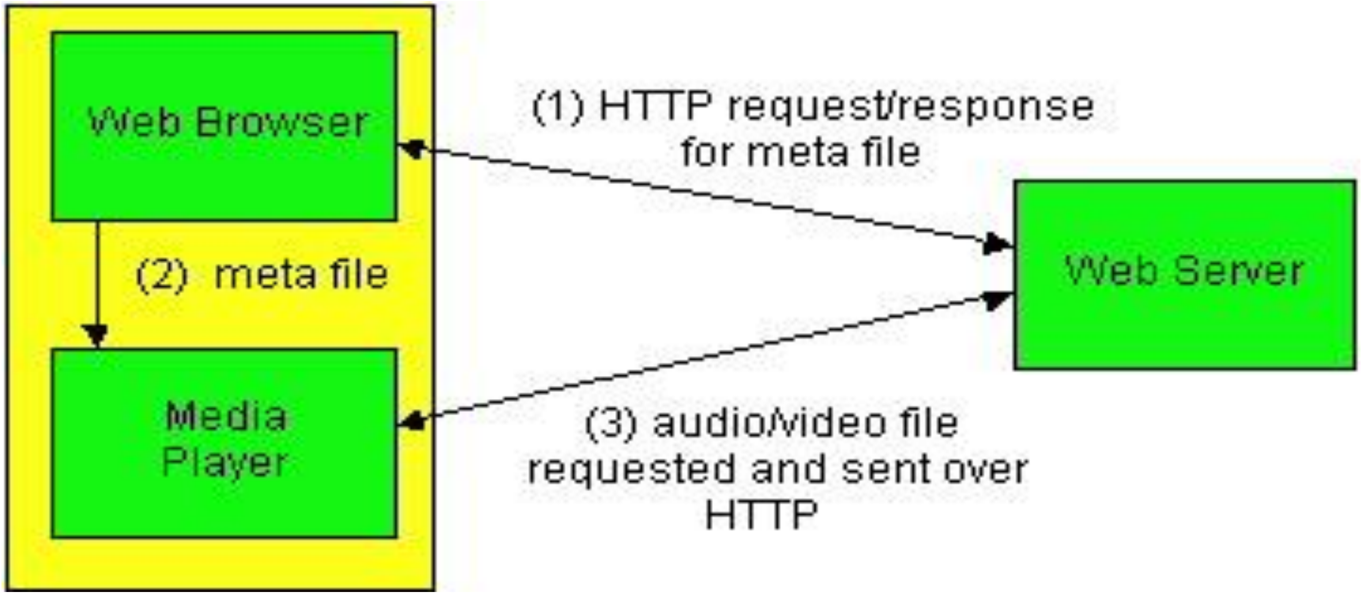


- audio or video stored in file
- files transferred as HTTP object
  - received in entirety at client

audio, video not streamed:

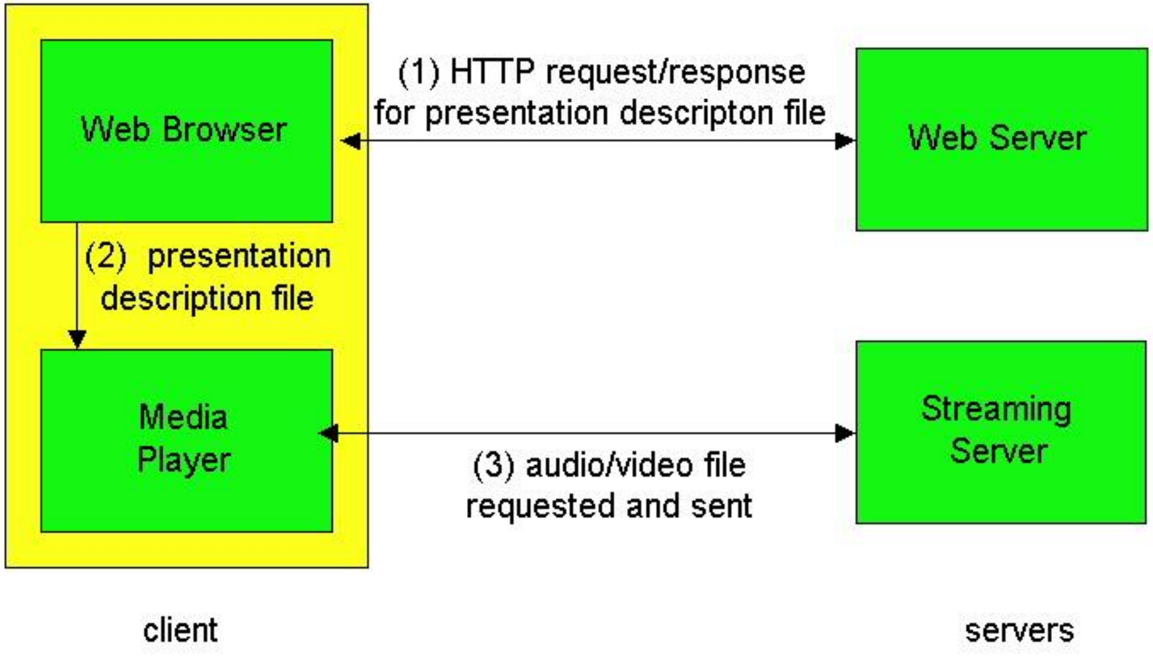
- no “pipelining”: long delays until playout!

Internet multimedia: streaming approach



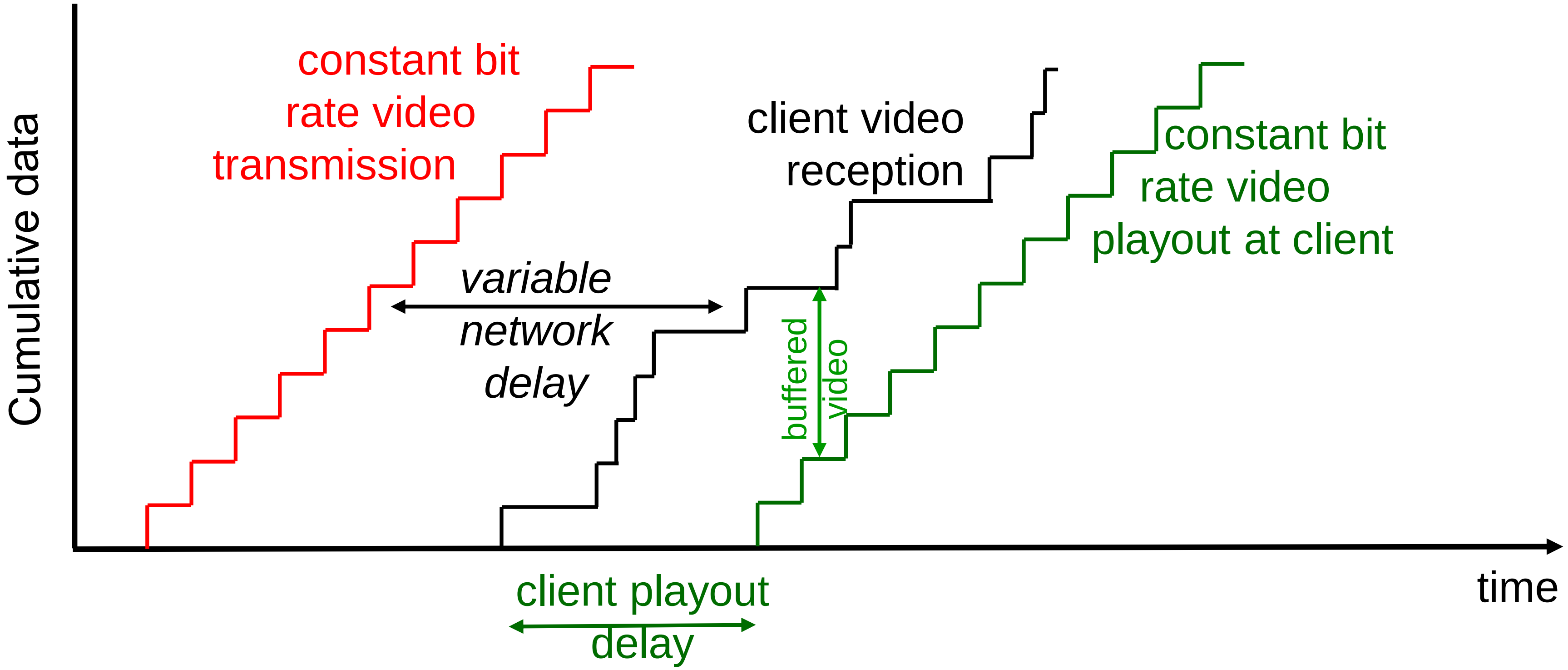
- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server streams audio/video to player

Streaming from a streaming server



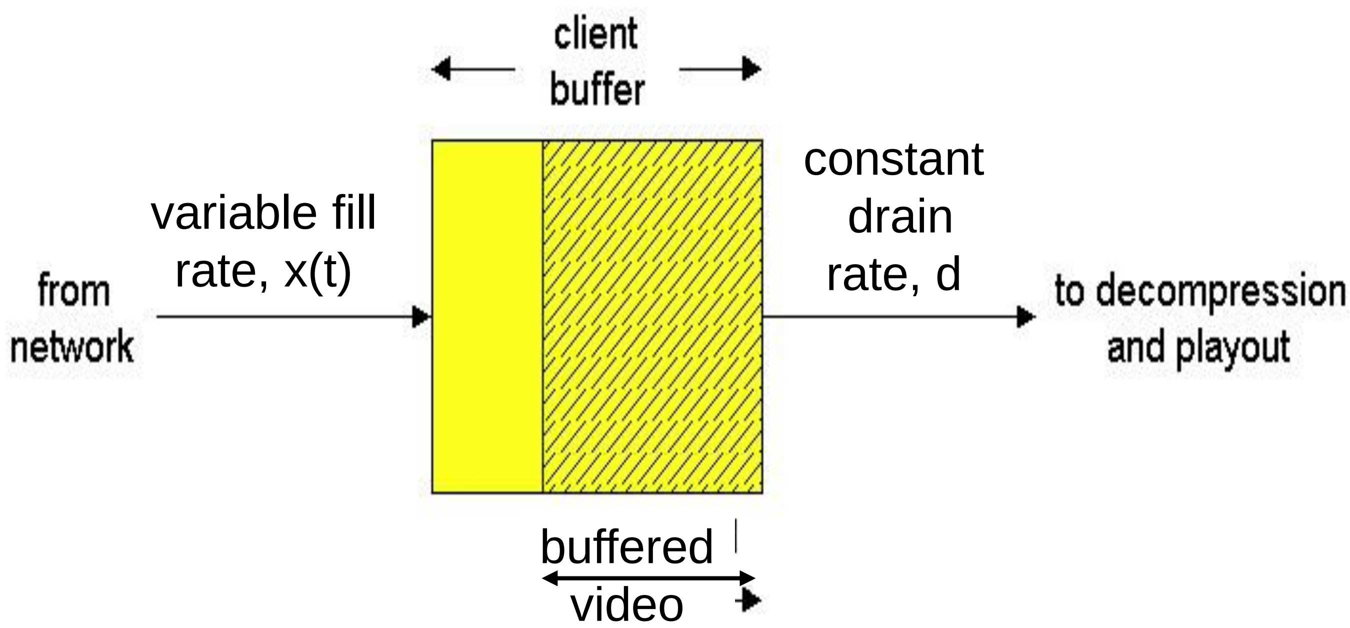
- allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3), more shortly

Streaming Multimedia: Client Buffering



- client-side buffering, playout delay compensate for network-added delay, delay jitter

Streaming Multimedia: Client Buffering



- client-side buffering, playout delay compensate for network-added delay, delay jitter

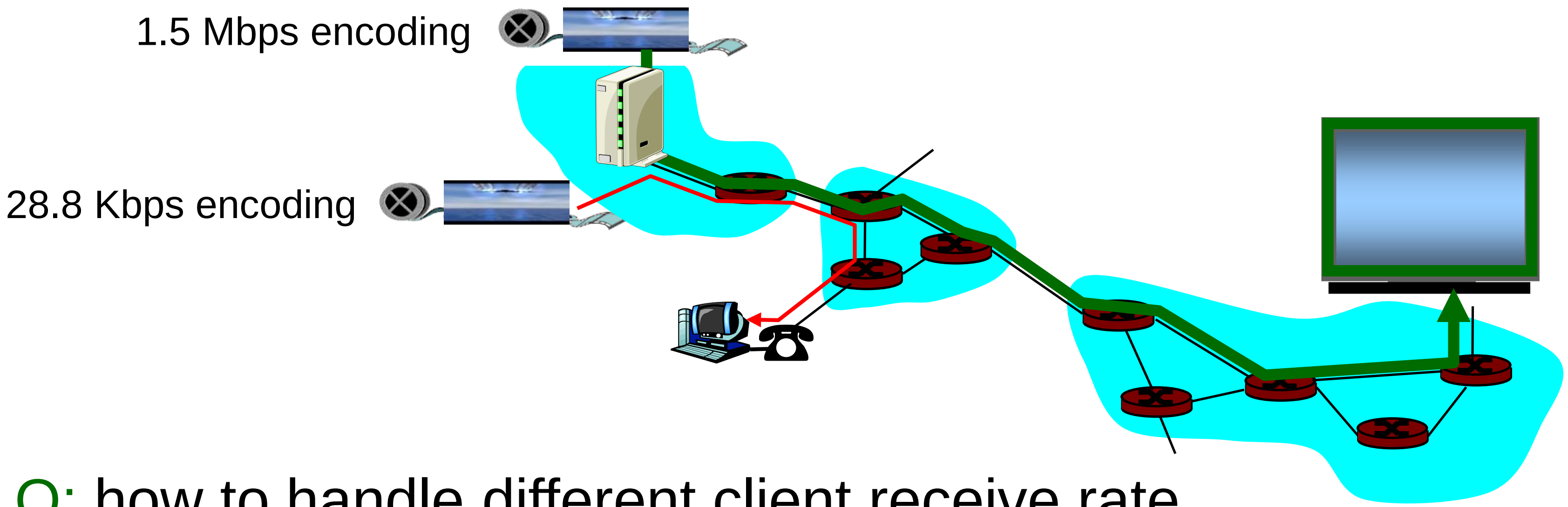
Streaming Multimedia: UDP or TCP?

- UDP
  - server sends at rate appropriate for client (oblivious to network congestion !)
    - often send rate = encoding rate = constant rate
    - then, fill rate = constant rate - packet loss
  - short playout delay (2-5 seconds) to remove network jitter
  - error recovery: time permitting

Streaming Multimedia: UDP or TCP?

- TCP
  - send at maximum possible rate under TCP
  - fill rate fluctuates due to TCP congestion control
  - larger playout delay: smooth TCP delivery rate
  - HTTP/TCP passes more easily through firewalls

Streaming Multimedia: client rate(s)



Q: how to handle different client receive rate capabilities?

- 28.8 Kbps dialup
- 100 Mbps Ethernet

A: server stores, transmits multiple copies of video, encoded at different rates

User Control of Streaming Media: RTSP

HTTP

- does not target multimedia content
- no commands for fast forward, etc.

RTSP: RFC 2326

- client-server application layer protocol
- user control: rewind, fast forward, pause, resume, repositioning, etc…

What it doesn't do:

- doesn't define how audio/video is encapsulated for streaming over network
- doesn't restrict how streamed media is transported (UDP or TCP possible)
- doesn't specify how media player buffers audio/video

RTSP: out of band control

FTP uses an "out-of-band" control channel:

- file transferred over one TCP connection.
- control info (directory changes, file deletion, rename) sent over separate TCP connection
- "out-of-band", "in-band" channels use different port numbers

RTSP messages also sent out-of-band:

- RTSP control messages use different port numbers than media stream: out-of-band.
  - port 554
- media stream is considered "in-band".

RTSP Example

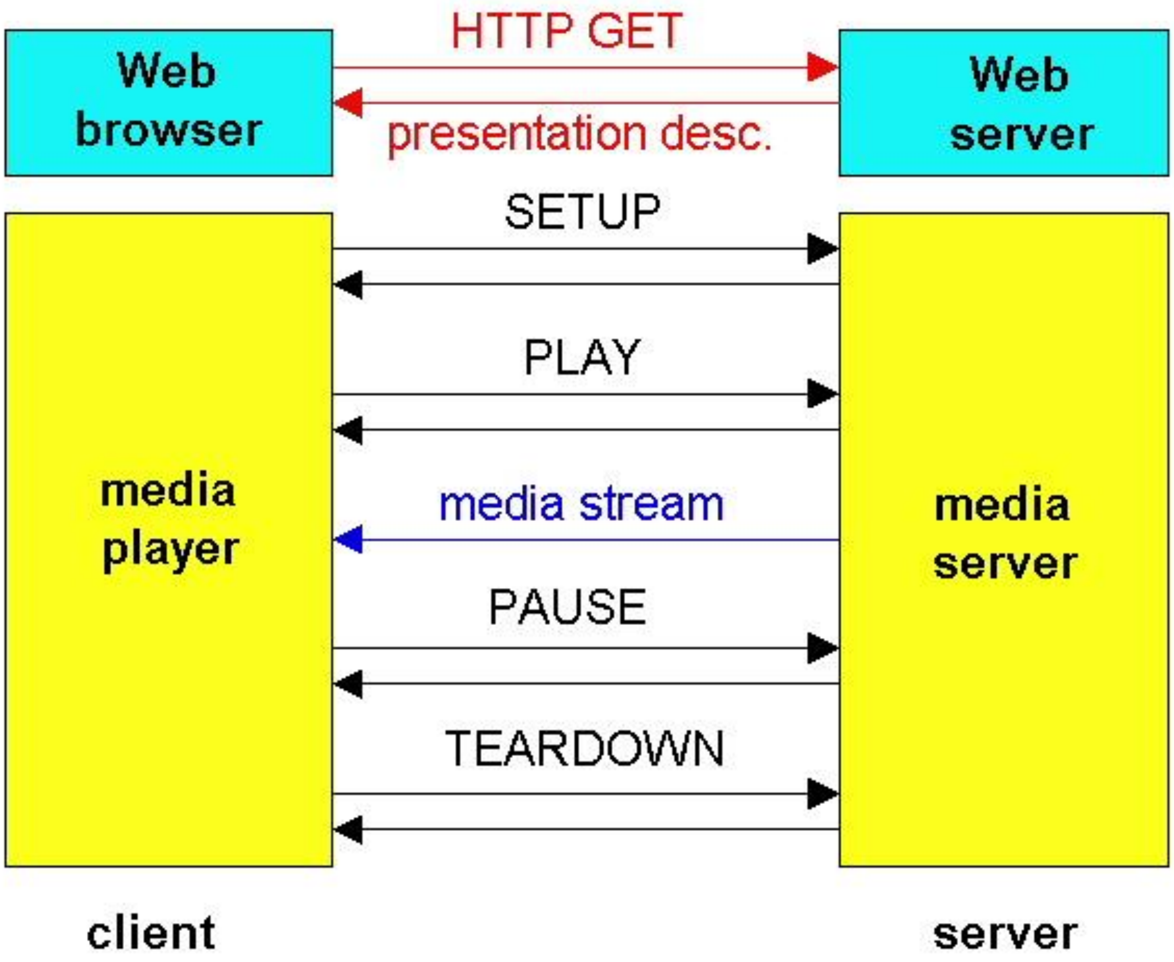
Scenario:

- metafile communicated to web browser
- browser launches player
- player sets up an RTSP control connection, data connection to streaming server

Metafile 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

Chapter 5 outline

- 5.1 Multimedia networking applications
- 5.2 Streaming stored audio and video
- 5.3 Making the best out of best effort service
- 5.4 Protocols for real-time interactive applications
  - RTP,RTCP,SIP

Real-time interactive applications

- PC-2-PC phone
  - Skype
- PC-2-phone
  - Dialpad
  - Net2phone
  - Skype
- videoconference with webcams
  - Skype
  - Polycom

Going to now look at a PC-2-PC Internet phone example in detail

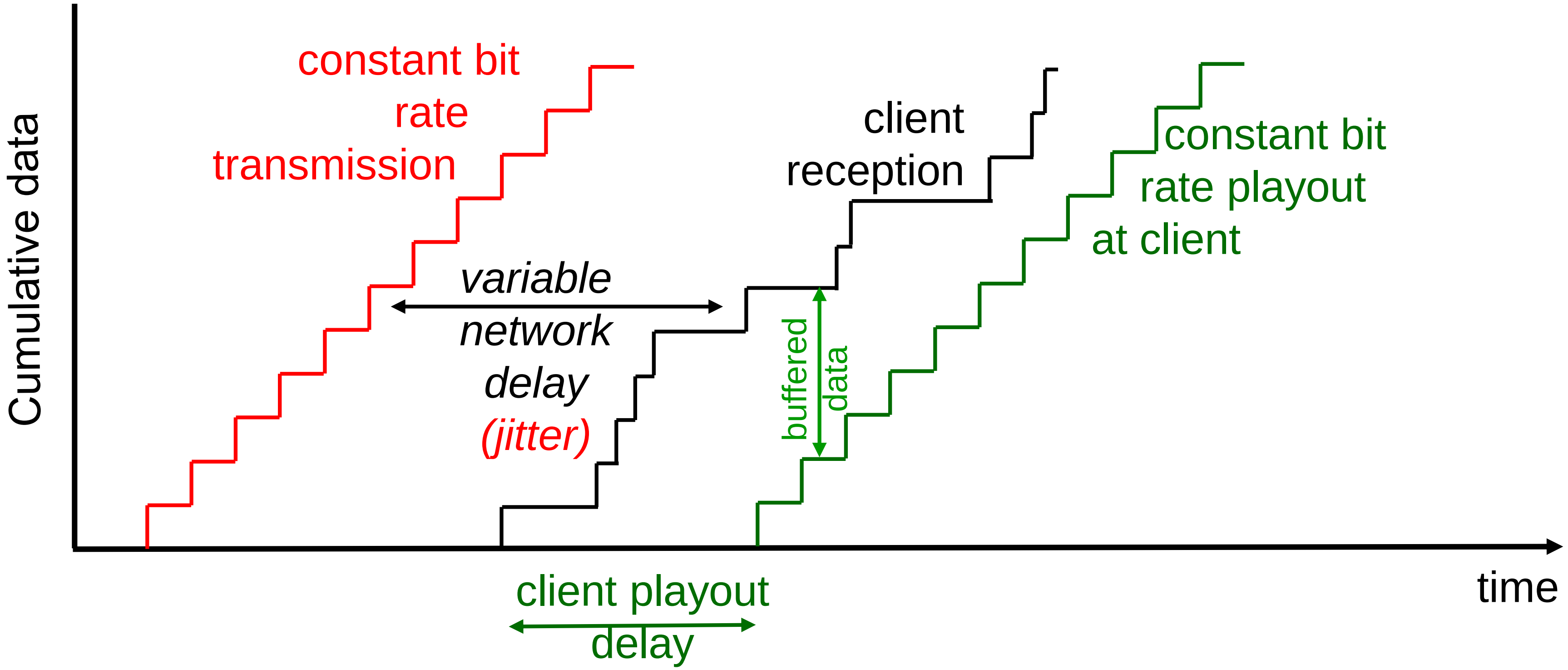
Interactive Multimedia: Internet Phone

- speaker's audio: alternating talk spurts, silent periods.
  - 8000 bytes per sec (64 Kbps) during talk spurt
  - pkts generated only during talk spurts
  - 20 msec chunks at 8 bytes/msec: 160 bytes data
- application-layer header added to each chunk.
- chunk+header encapsulated into UDP segment.
- application sends UDP segment into socket every 20 msec during talkspurt

Internet Phone: Packet Loss and Delay

- network loss: IP datagram lost due to network congestion (router buffer overflow)
- delay loss: IP datagram arrives too late for playout at receiver
  - delays: processing, queueing in network; end-system (sender, receiver) delays
  - typical maximum tolerable delay: 400 ms
- loss tolerance: depending on voice encoding, losses concealed, packet loss rates between 1% and 10% can be tolerated.

Delay Jitter



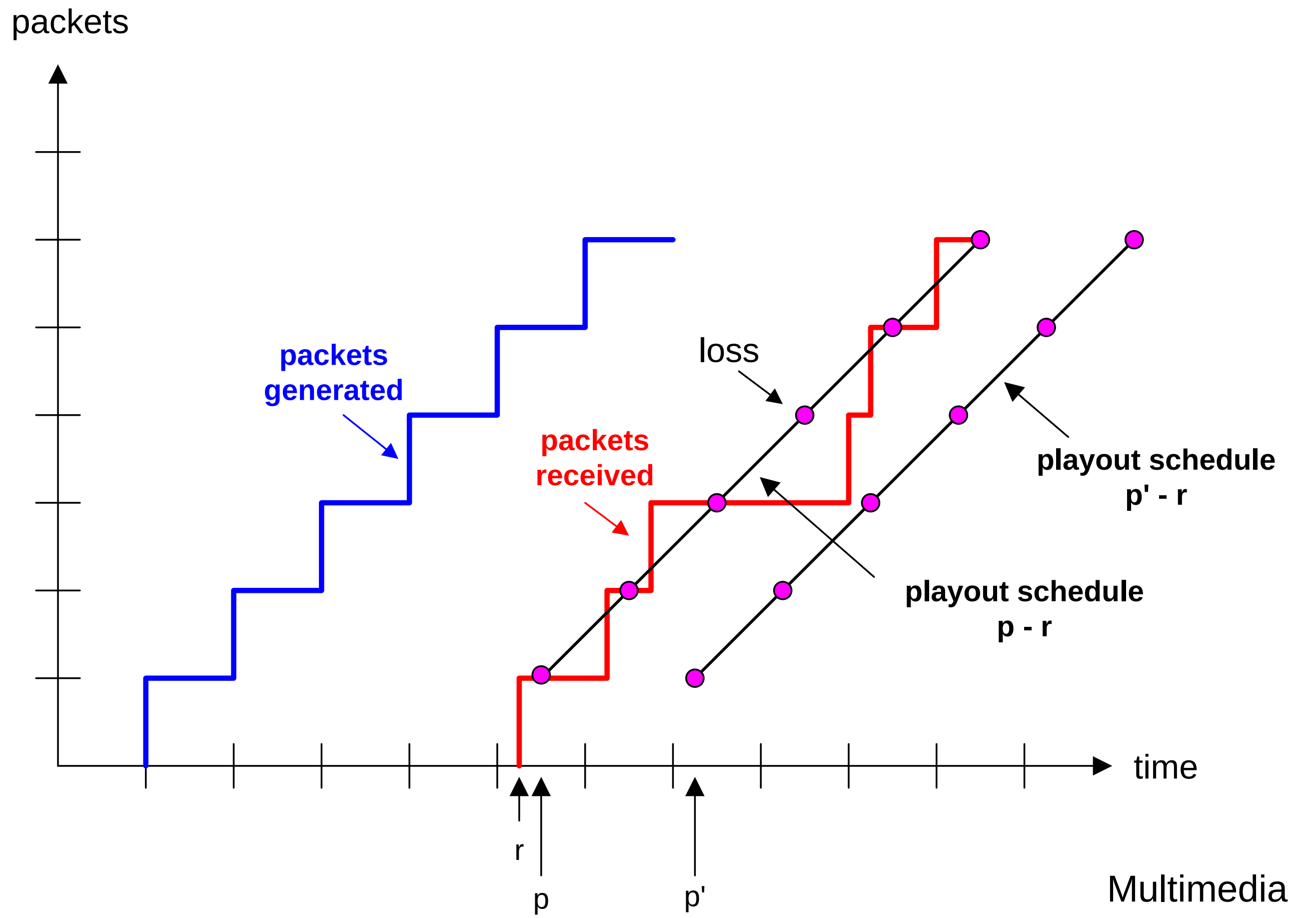
- consider end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

Internet Phone: 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

Fixed Playout Delay

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



Adaptive Playout Delay (1)

- Goal: minimize playout delay, keeping late loss rate low
- 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.

$$\mathrm{t_i} = \text{timestamp of the ith packet}$$

$$\mathrm{r_i} = \text{the time packet i is received by receiver}$$

$$\mathrm{p_i} = \text{the time packet i is played at receiver}$$

$$\mathrm{r_i} - \mathrm{t_i} = \text{network delay for ith packet}$$

$$\mathrm{d_i} = \text{estimate of average network delay after receiving ith packet}$$

dynamic estimate of average delay at receiver:

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

where *u* is a fixed constant (e.g., *u* = .01).

Adaptive playout delay (2)

- also useful to estimate average deviation of delay, v_i :

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

- 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:

$$p_i = t_i + d_i + Kv_i$$

where K is positive constant

- remaining packets in talkspurt are played out periodically

Adaptive Playout (3)

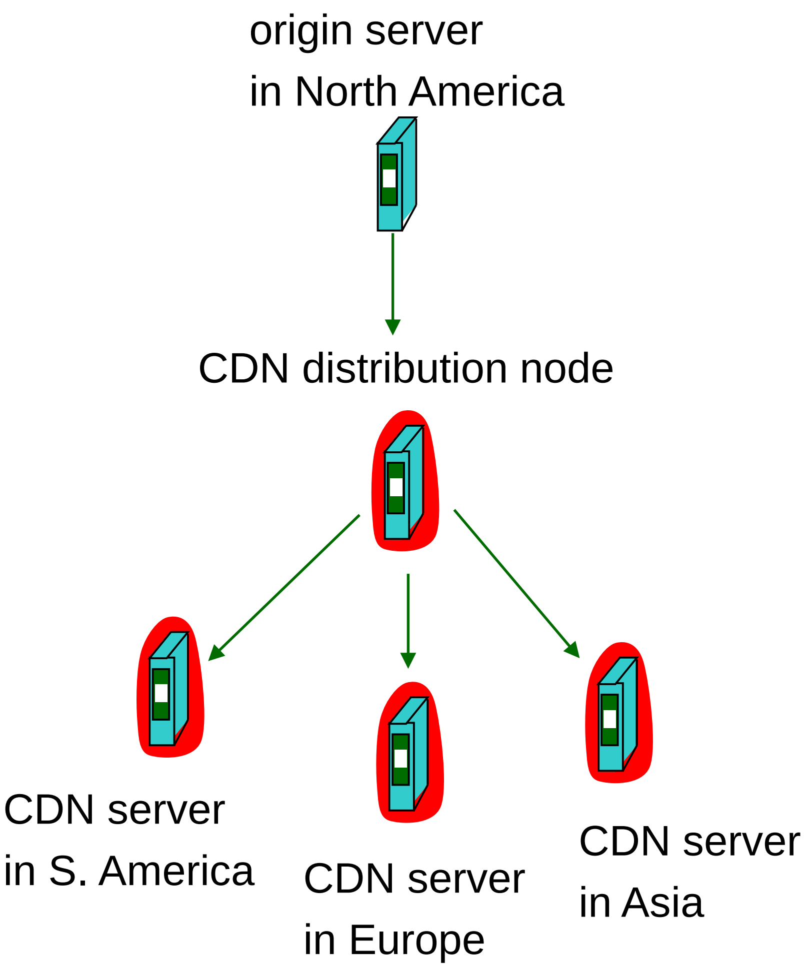
Q: How does receiver determine whether packet is first in a talkspurt?

- 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.

Content distribution networks (CDNs)

Content replication

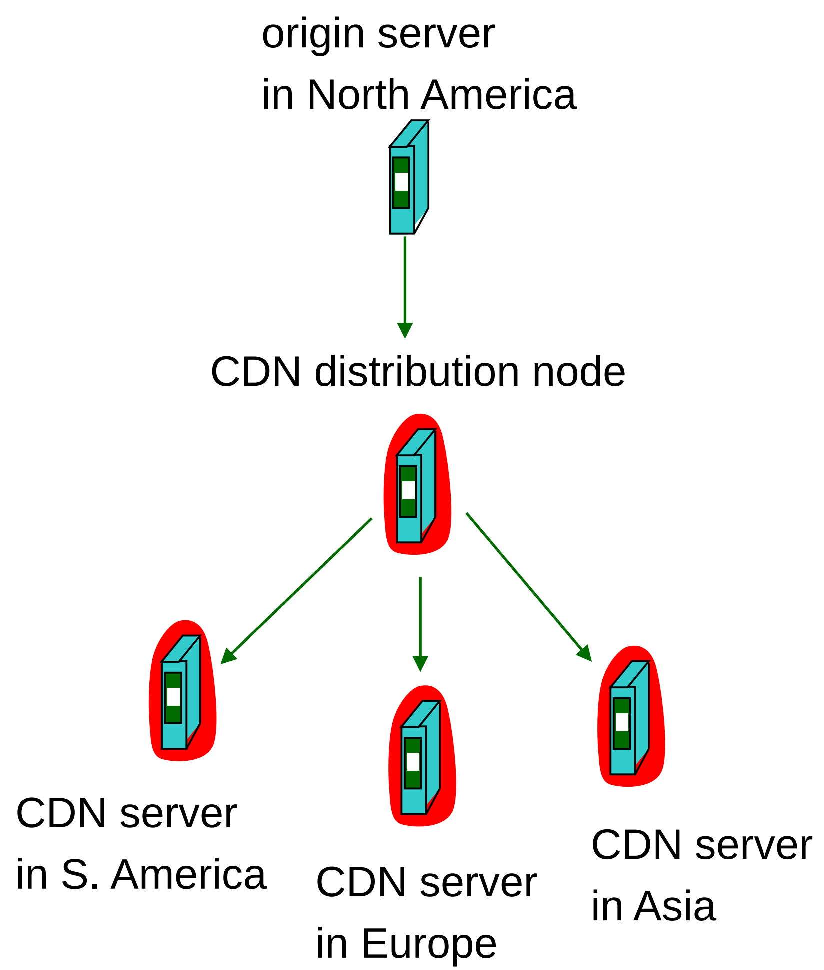
- challenging to stream large files (e.g., video) from single origin server in real time
- *solution:* replicate content at hundreds of servers throughout Internet
  - content downloaded to CDN servers ahead of time
  - *placing content "close" to user avoids impairments (loss, delay) of sending content over long paths*
  - CDN server typically in edge/access network



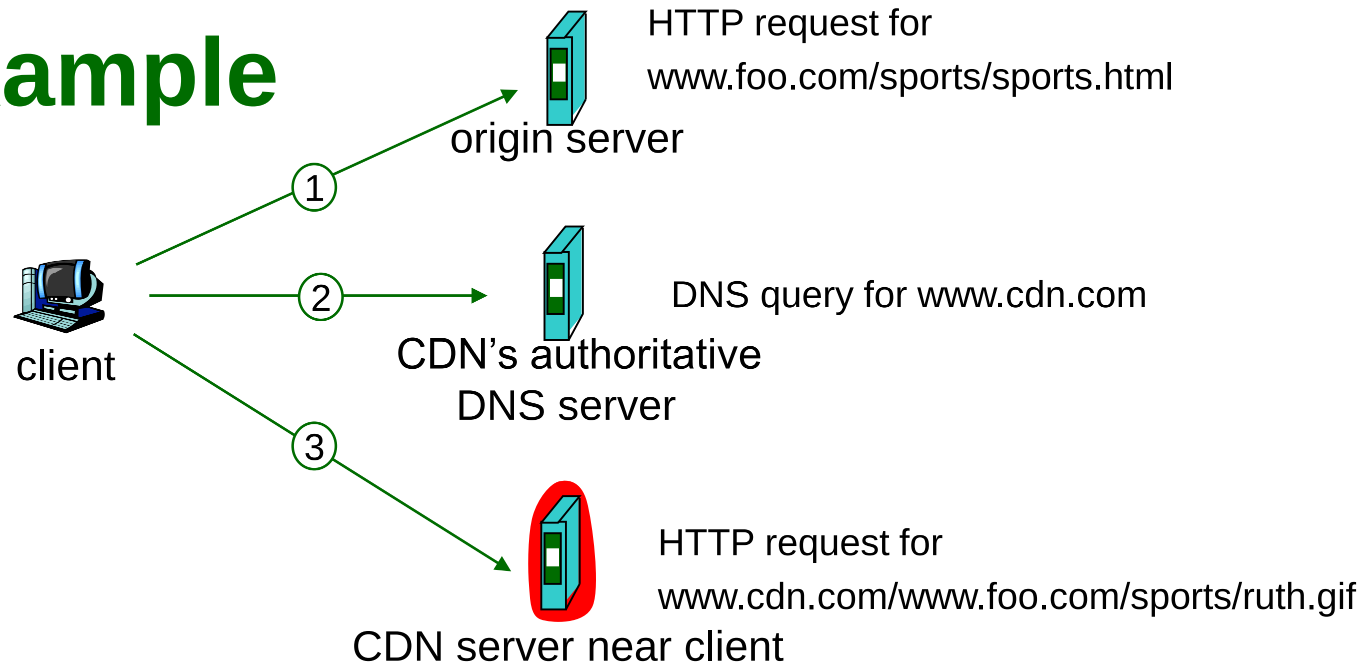
Content distribution networks (CDNs)

<u>Content replication</u>

- CDN (e.g., Akamai) customer is the content provider (e.g., CNN)
- CDN replicates customers' content in CDN servers.
- when provider updates content, CDN updates servers



CDN example



origin server (www.foo.com)

- distributes HTML
- replaces:

  http://www.foo.com/sports.ruth.gif

  with

  http://www.cdn.com/www.foo.com/sports/ruth.gif

CDN company (cdn.com)

- distributes gif files
- uses its authoritative DNS server to route redirect requests

More about CDNs

routing requests

- CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server:
  - server determines ISP from which query originates
  - uses “map” to determine best CDN server
- CDN nodes create application-layer overlay network

Summary: Internet Multimedia: bag of tricks

- use UDP to avoid TCP congestion control (delays) for time-sensitive traffic
- client-side adaptive playout delay: to compensate for delay
- server side matches stream bandwidth to available client-to-server path bandwidth
  - chose among pre-encoded stream rates
  - dynamic server encoding rate
- error recovery (on top of UDP)
  - FEC, interleaving, error concealment
  - retransmissions, time permitting
- CDN: bring content closer to clients

Chapter 5 outline

- 5.1 Multimedia networking applications
- 5.2 Streaming stored audio and video
- 5.3 Making the best out of best effort service
- 5.4 Protocols for real-time interactive applications
    - RTP,RTCP,SIP

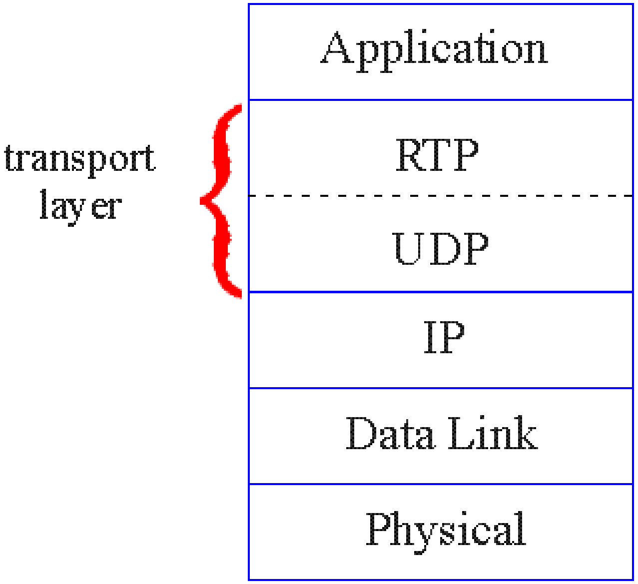
Real-Time Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two Internet phone applications run RTP, then they may be able to work together

RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

- port numbers, IP addresses
- payload type identification
- packet sequence numbering
- time-stamping



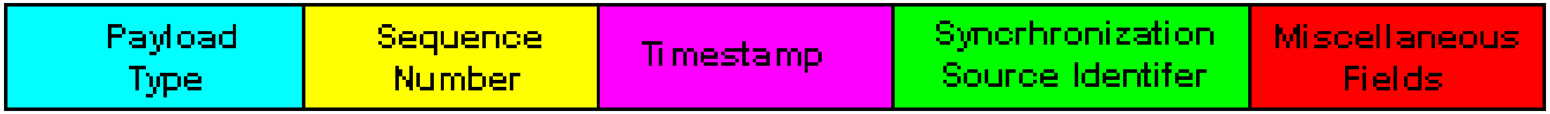
RTP Example

- consider sending 64 kbps PCM-encoded voice over RTP.
- application collects encoded data in chunks, e.g., every 20 msec = 160 bytes in a chunk.
- audio chunk + RTP header form RTP packet, which is encapsulated in UDP segment
- RTP header indicates type of audio encoding in each packet
  - sender can change encoding during conference.
- RTP header also contains sequence numbers, timestamps.

RTP and QoS

- RTP does **not** provide any mechanism to ensure timely data delivery or other QoS guarantees.
- RTP encapsulation is only seen at end systems (not) by intermediate routers.
  - routers providing best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter.

RTP Header



RTP Header

Payload Type (7 bits): Indicates type of encoding currently being used. If sender changes encoding in middle of conference, sender informs receiver via payload type field.

- Payload type 0: PCM mu-law, 64 kbps
- Payload type 3, GSM, 13 kbps
- Payload type 7, LPC, 2.4 kbps
- Payload type 26, Motion JPEG
- Payload type 31. H.261
- Payload type 33, MPEG2 video

Sequence Number (16 bits): Increments by one for each RTP packet sent, and may be used to detect packet loss and to restore packet sequence.

RTP Header (2)

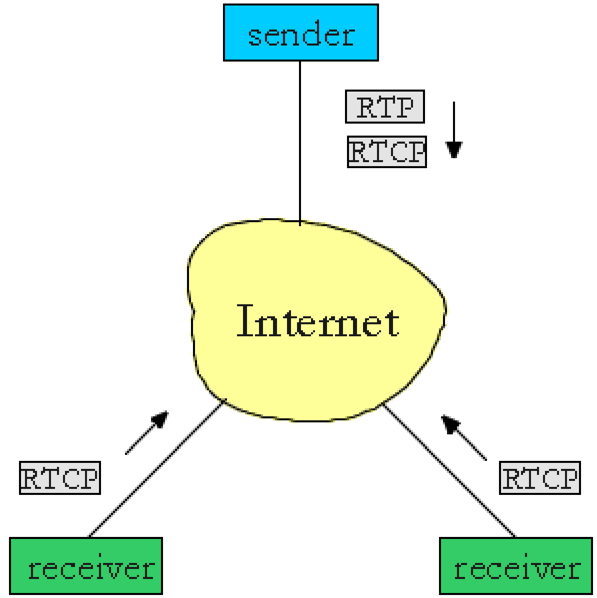
- *Timestamp field (32 bytes long):* sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock typically increments by one for each sampling period (for example, each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, then timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.

- *SSRC field (32 bits long):* identifies source of the RTP stream. Each stream in RTP session should have distinct SSRC.

Real-Time Control Protocol (RTCP)

- works in conjunction with RTP.
- each participant in RTP session periodically transmits RTCP control packets to all other participants.
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter, etc.
- feedback can be used to control performance
  - sender may modify its transmissions based on feedback

RTCP - Continued



- each RTP session: typically a single multicast address; all RTP /RTCP packets belonging to session use multicast address.

- RTP, RTCP packets distinguished from each other via distinct port numbers.

- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases

RTCP Packets

Receiver report packets:

- fraction of packets lost, last sequence number, average interarrival jitter

Sender report packets:

- SSRC of RTP stream, current time, number of packets sent, number of bytes sent

Source description packets:

- e-mail address of sender, sender's name, SSRC of associated RTP stream
- provide mapping between the SSRC and the user/host name

Synchronization of Streams

- RTCP can synchronize different media streams within a RTP session
- consider videoconferencing app for which each sender generates one RTP stream for video, one for audio.
- timestamps in RTP packets tied to the video, audio sampling clocks
  - *not* tied to wall-clock time
- each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created.
- receivers uses association to synchronize playout of audio, video

RTCP Bandwidth Scaling

- RTCP attempts to limit its traffic to 5% of session bandwidth.

Example

- Suppose one sender, sending video at 2 Mbps. Then RTCP attempts to limit its traffic to 100 Kbps.
- RTCP gives 75% of rate to receivers; remaining 25% to sender
- 75 kbps is equally shared among receivers:
  - with R receivers, each receiver gets to send RTCP traffic at 75/R kbps.
- sender gets to send RTCP traffic at 25 kbps.
- participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate

SIP: Session Initiation Protocol

[RFC 3261]

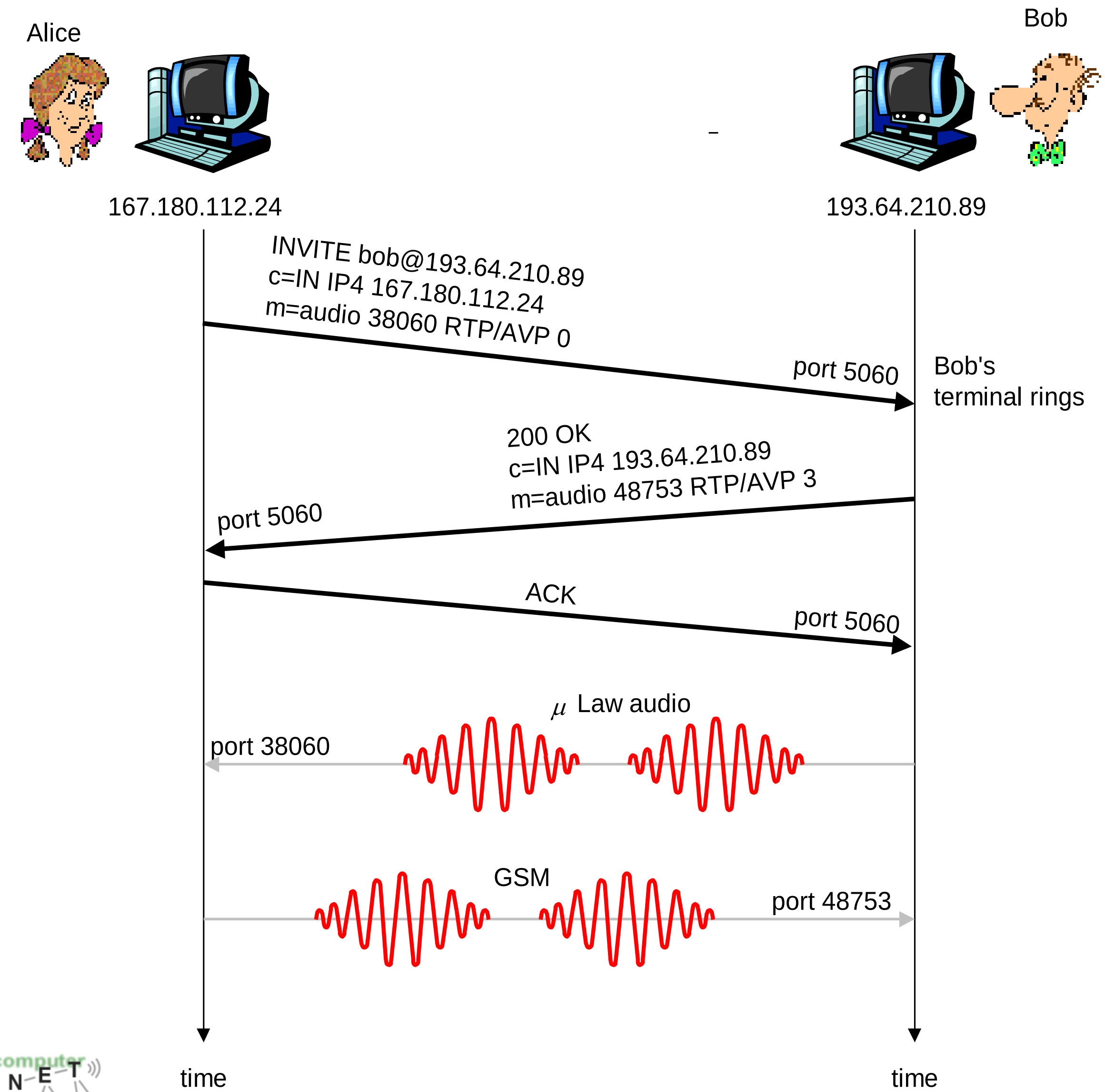
SIP long-term vision:

- all telephone calls, video conference calls take place over Internet
- people are identified by names or e-mail addresses, rather than by phone numbers
- you can reach callee, no matter where callee roams, no matter what IP device callee is currently using

SIP Services

- Setting up a call, SIP provides mechanisms ..
  - for caller to let callee know she wants to establish a call
  - so caller, callee can agree on media type, encoding
  - to end call
- determine current IP address of callee:
  - maps mnemonic identifier to current IP address
- call management:
  - add new media streams during call
  - change encoding during call
  - invite others
  - transfer, hold calls

Setting up a call to known IP address



- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive (PCM ulaw)
- Bob's 200 OK message indicates his port number, IP address, preferred encoding (GSM)
- SIP messages can be sent over TCP or UDP; here sent over RTP/UDP.
- default SIP port number is 5060.

Setting up a call (more)

- codec negotiation:
  - suppose Bob doesn't have PCM ulaw encoder.
  - Bob will instead reply with 606 Not Acceptable Reply, listing his encoders Alice can then send new INVITE message, advertising different encoder
- rejecting a call
  - Bob can reject with replies "busy," "gone," "payment required," "forbidden"
- media can be sent over RTP or some other protocol

Example of SIP message

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

Notes:

- HTTP message syntax
- sdp = session description protocol
- Call-ID is unique for every call.

- Here we don't know Bob's IP address. Intermediate SIP servers needed.
- Alice sends, receives SIP messages using SIP default port 5060
- Alice specifies in Via: header that SIP client sends, receives SIP messages over UDP

Name translation and user location

- caller wants to call callee, but only has callee's name or e-mail address.
- need to get IP address of callee's current host:
  - user moves around
  - DHCP protocol
  - user has different IP devices (PC, PDA, car device)
- result can be based on:
  - time of day (work, home)
  - caller (don't want boss to call you at home)
  - status of callee (calls sent to voicemail when callee is already talking to someone)

Service provided by SIP servers:

- SIP registrar server
- SIP proxy server

SIP Registrar

- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server
  (similar function needed by Instant Messaging)

Register Message:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

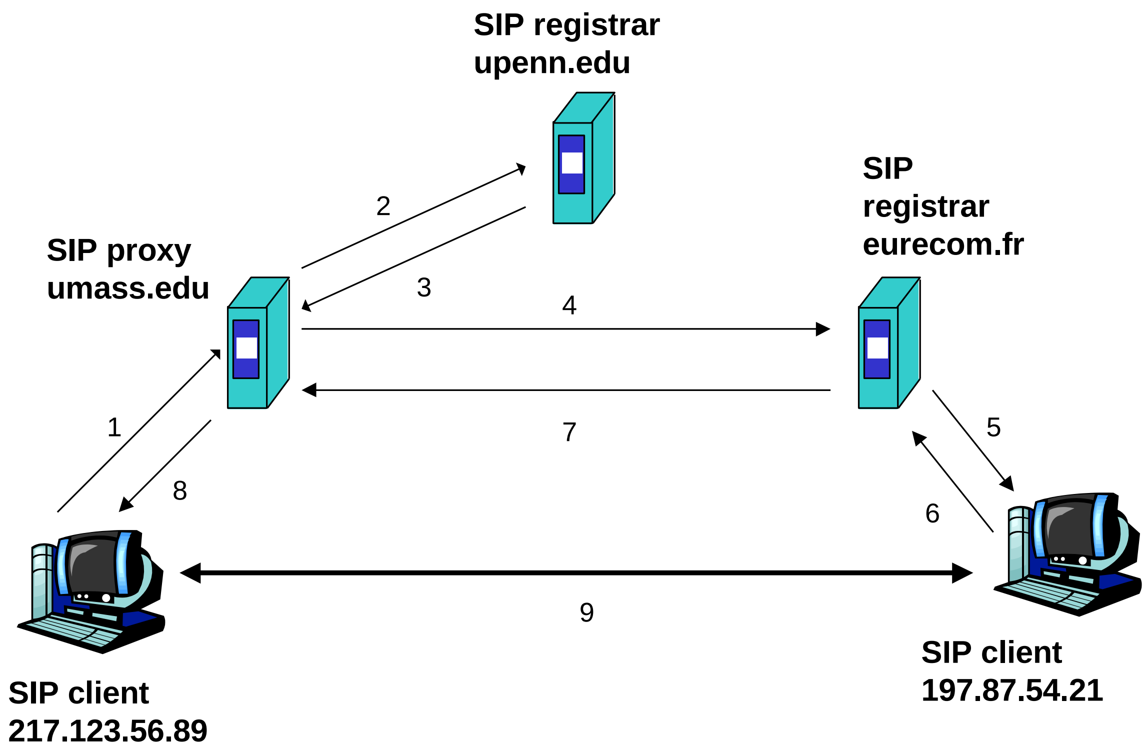
SIP Proxy

- Alice sends invite message to her proxy server
  - contains address sip:bob@domain.com
- proxy responsible for routing SIP messages to callee
  - possibly through multiple proxies.
- callee sends response back through the same set of proxies.
- proxy returns SIP response message to Alice
  - contains Bob's IP address
- proxy analogous to local DNS server

Example

Caller jim@umass.edu places a call to keith@upenn.edu

(1) Jim sends INVITE message to umass SIP proxy. (2) Proxy forwards request to upenn registrar server.
(3) upenn server returns redirect response, indicating that it should try keith@eurecom.fr



(4) umass proxy sends INVITE to eurecom registrar. (5) eurecom registrar forwards INVITE to 197.87.54.21, which is running keith's SIP client. (6-8) SIP response sent back (9) media sent directly between clients.

Note: also a SIP ack message, which is not shown.

Thank you

Any questions?