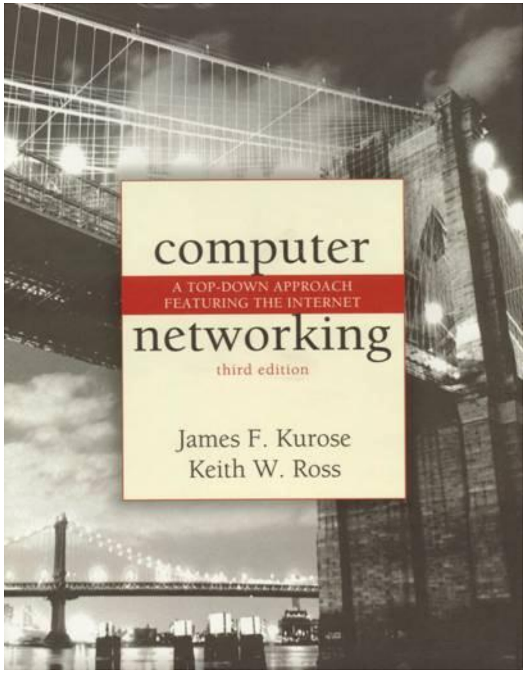
# Chapter 7
# Multimedia Networking



## A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

□ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)

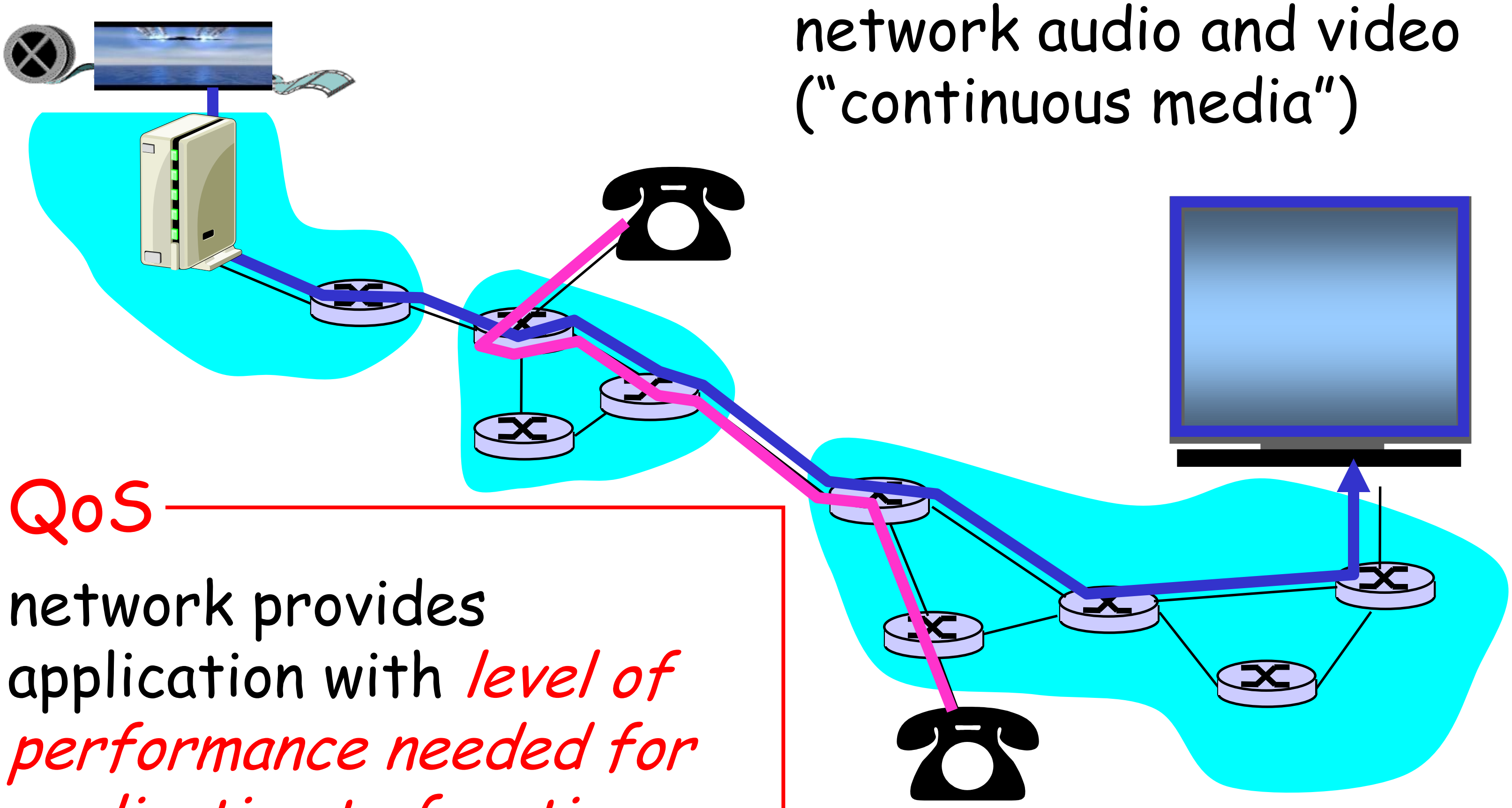
□ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK / KWR

*Computer Networking: A Top Down Approach Featuring the Internet,*
3rd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2004.

# Multimedia, Quality of Service: What is it?

Multimedia applications: network audio and video ("continuous media")



QoS

network provides application with *level of performance needed for application to function.*

# Chapter 7: Goals

## Principles

- Classify multimedia applications
- Identify the network services the apps need
- Making the best of best effort service
- Mechanisms for providing QoS

## Protocols and Architectures

- Specific protocols for best-effort
- Architectures for QoS

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# MM Networking Applications

## Classes of MM applications:

1) Streaming stored audio and video

2) Streaming live audio and video

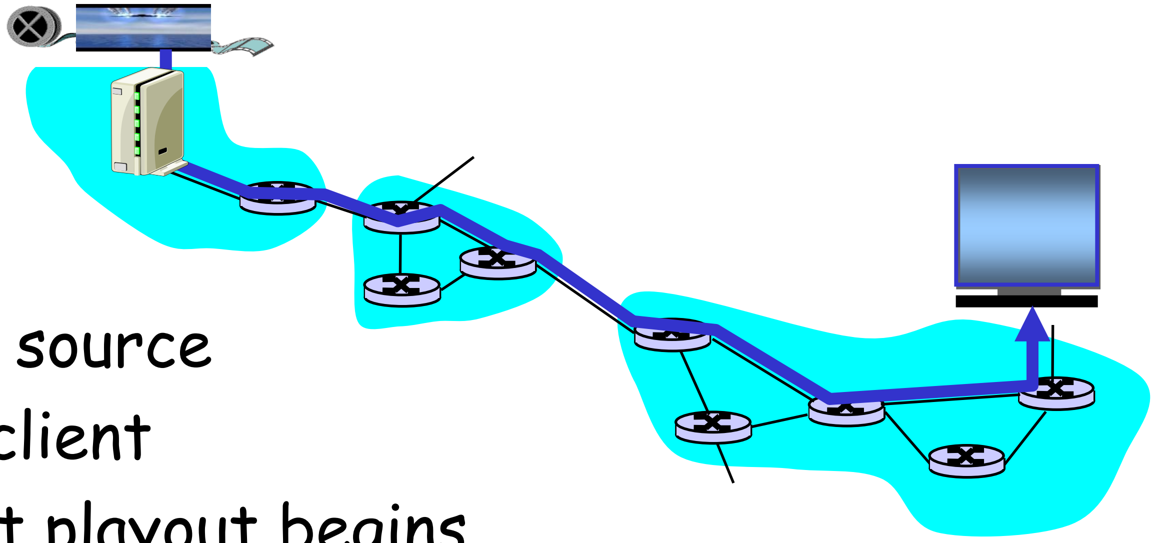
3) Real-time interactive audio and video

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

## Fundamental characteristics:

- Typically **delay sensitive**
  - end-to-end delay
  - delay jitter
- But **loss tolerant**: infrequent losses cause minor glitches
- Antithesis of data, which are loss intolerant but delay tolerant.

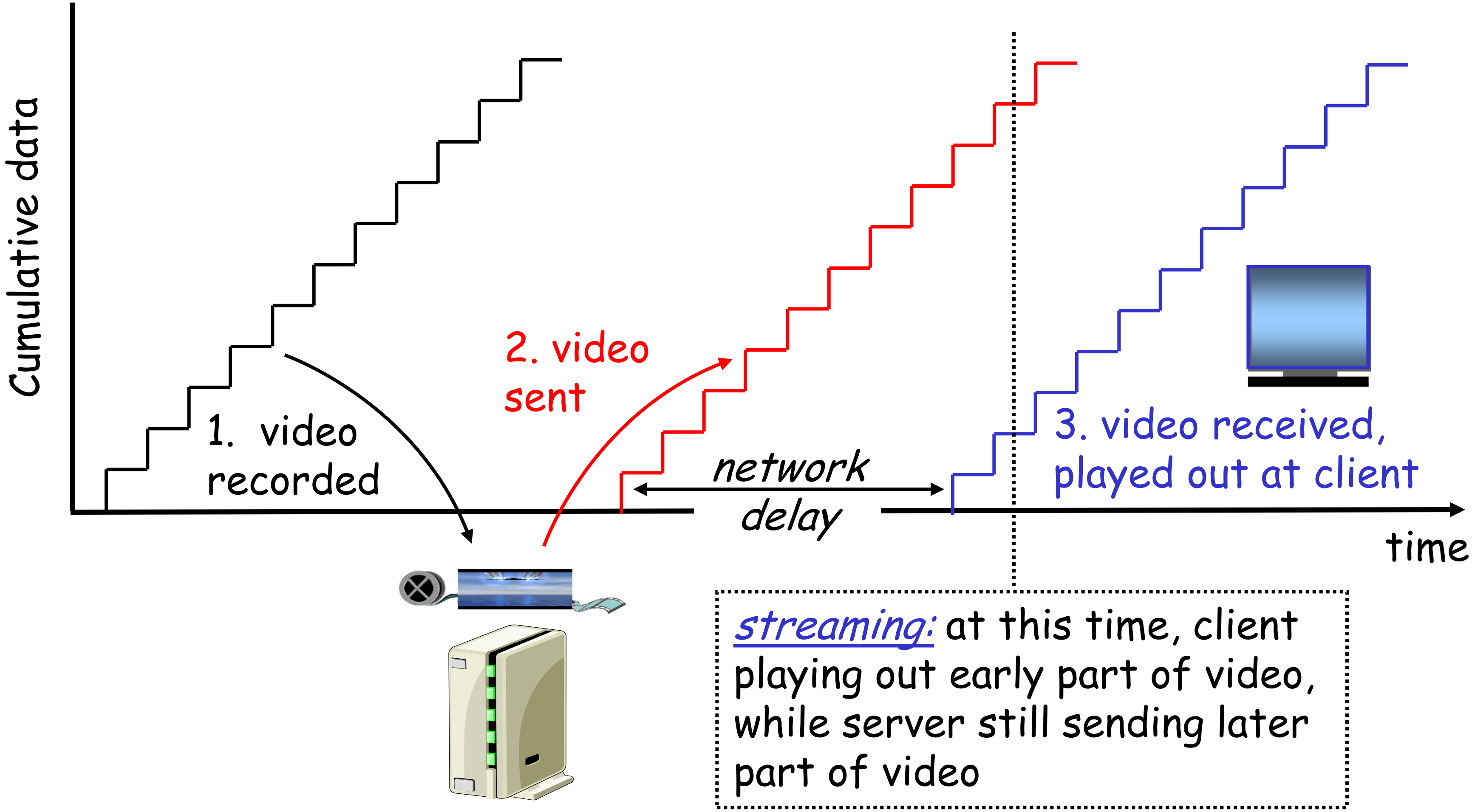
# Streaming Stored Multimedia



Streaming:

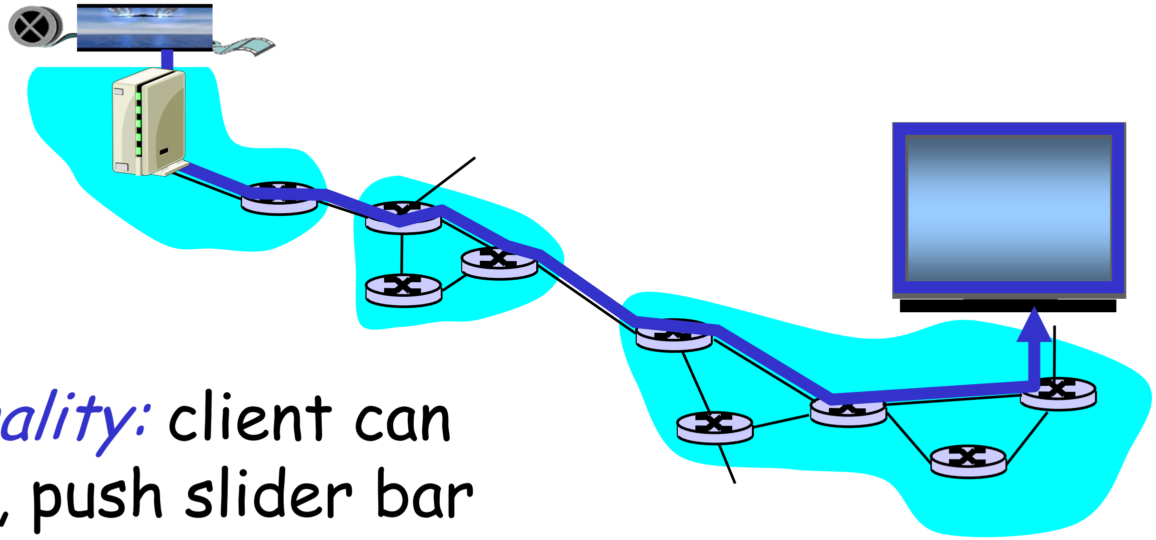
- media stored at source
- transmitted to client
- streaming: client playout begins *before* all data has arrived
  - timing constraint for still-to-be transmitted data: in time for playout

# 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
  - RTSP often used (more later)
- timing constraint for still-to-be transmitted data: in time for playout

# Streaming Live Multimedia

Examples:

- Internet radio talk show
- Live sporting event

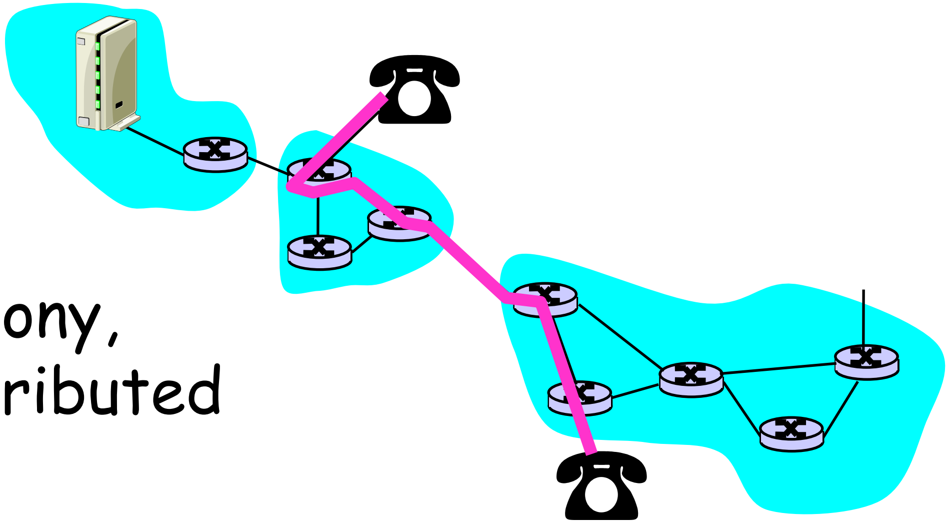
Streaming

- playback buffer
- playback can lag tens of seconds after transmission
- still have timing constraint

Interactivity

- fast forward impossible
- rewind, pause possible!

# Interactive, Real-Time Multimedia

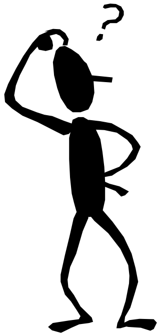


- **applications:** IP telephony, video conference, distributed interactive worlds
- **end-end delay requirements:**
  - audio: < 150 msec good, < 400 msec OK
    - includes application-level (packetization) and network delays
    - higher delays noticeable, impair interactivity
- **session initialization**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

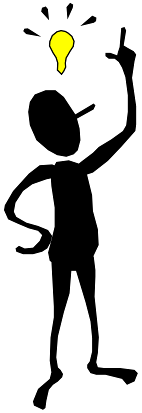
# Multimedia Over Today's Internet

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

- *no* guarantees on delay, loss



But you said multimedia apps requires 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:

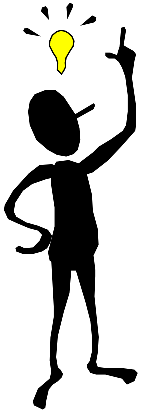
- Fundamental changes in Internet so that apps can reserve end-to-end bandwidth
- Requires new, complex software in hosts & routers

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?

# 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 it back to analog signal:
  - some quality reduction

## Example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet telephony: 5.3 - 13 kbps

# A few words about video compression

- Video is sequence of images displayed at constant rate
  - e.g. 24 images/sec
- Digital image is array of pixels
- Each pixel represented by bits
- Redundancy
  - spatial
  - temporal

Examples:

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)

Research:

- Layered (scalable) video
  - adapt layers to available bandwidth

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# Streaming Stored Multimedia

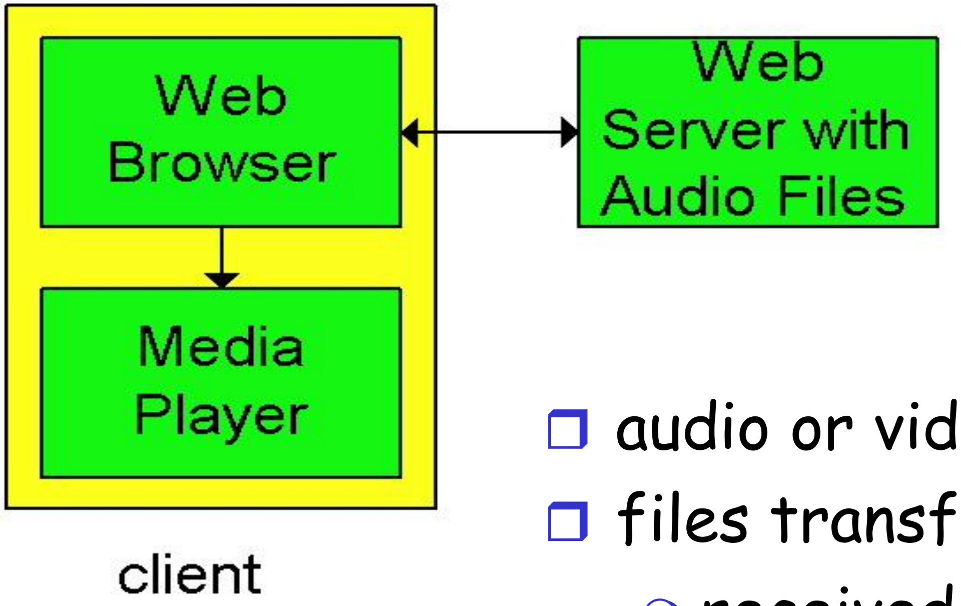
Application-level streaming techniques for making the best out of best effort service:

- 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

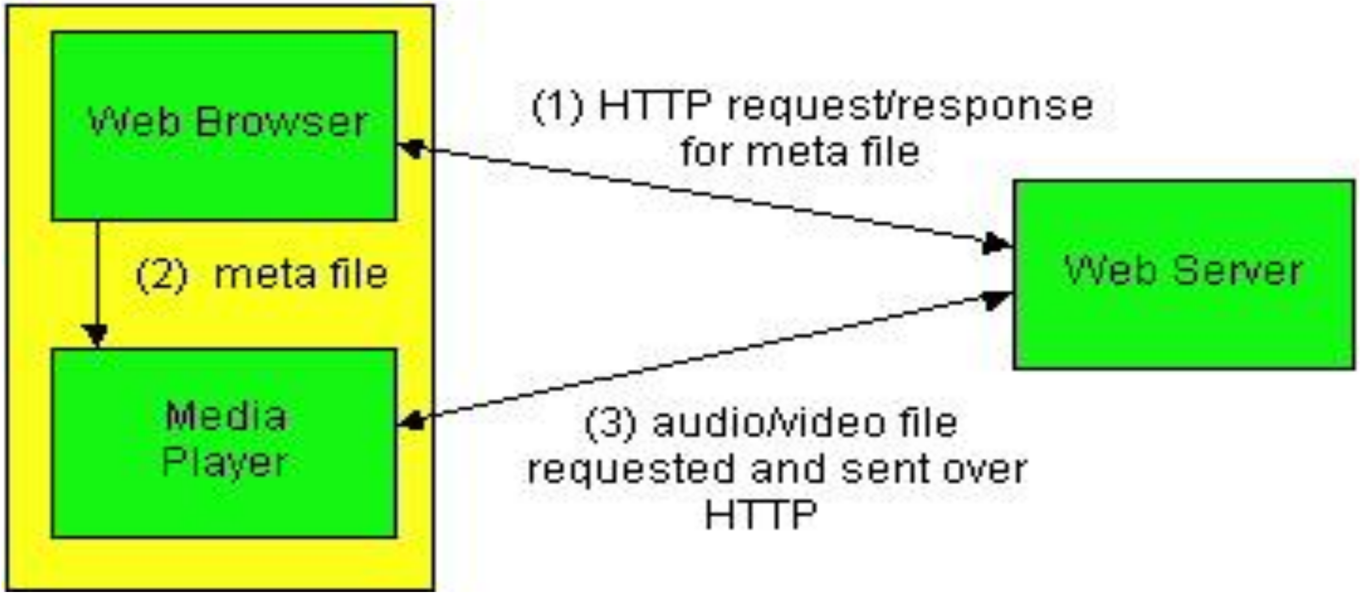


- audio or video stored in file
- files transferred as HTTP object
  - received in entirety at client
  - then passed to player

audio, video not streamed:

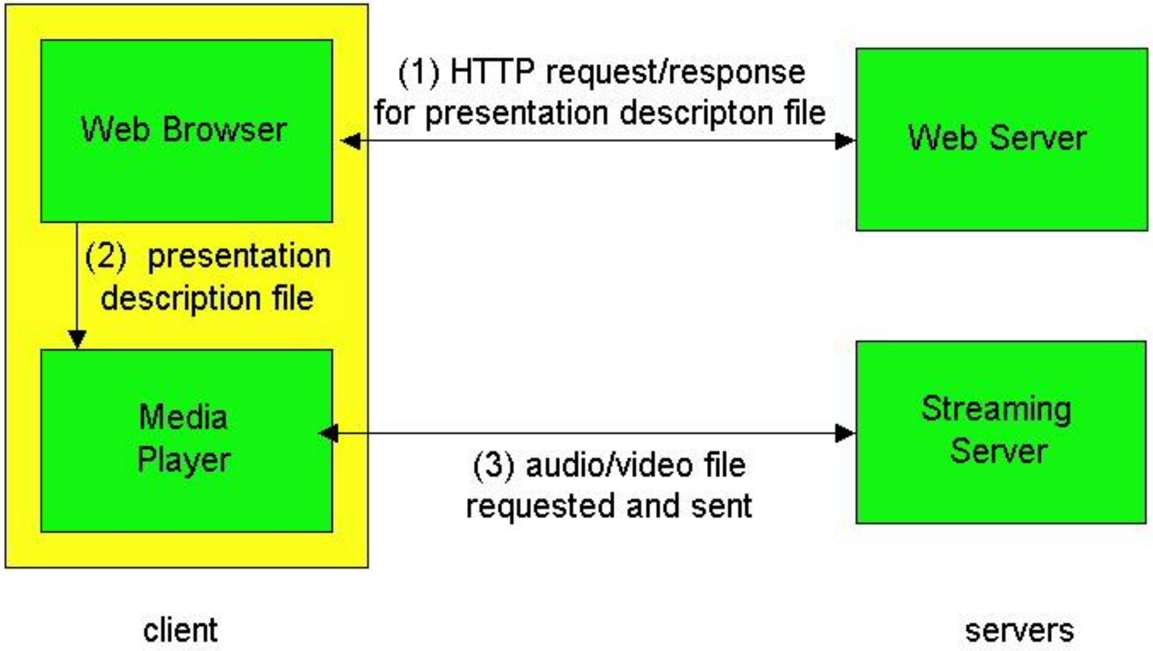
- no, "pipelining," long delays until playout!

# Internet multimedia: streaming approach


Web Browser
(1) HTTP request/response for meta file
(2) meta file
Web Server
Media Player
(3) audio/video file requested and sent over HTTP

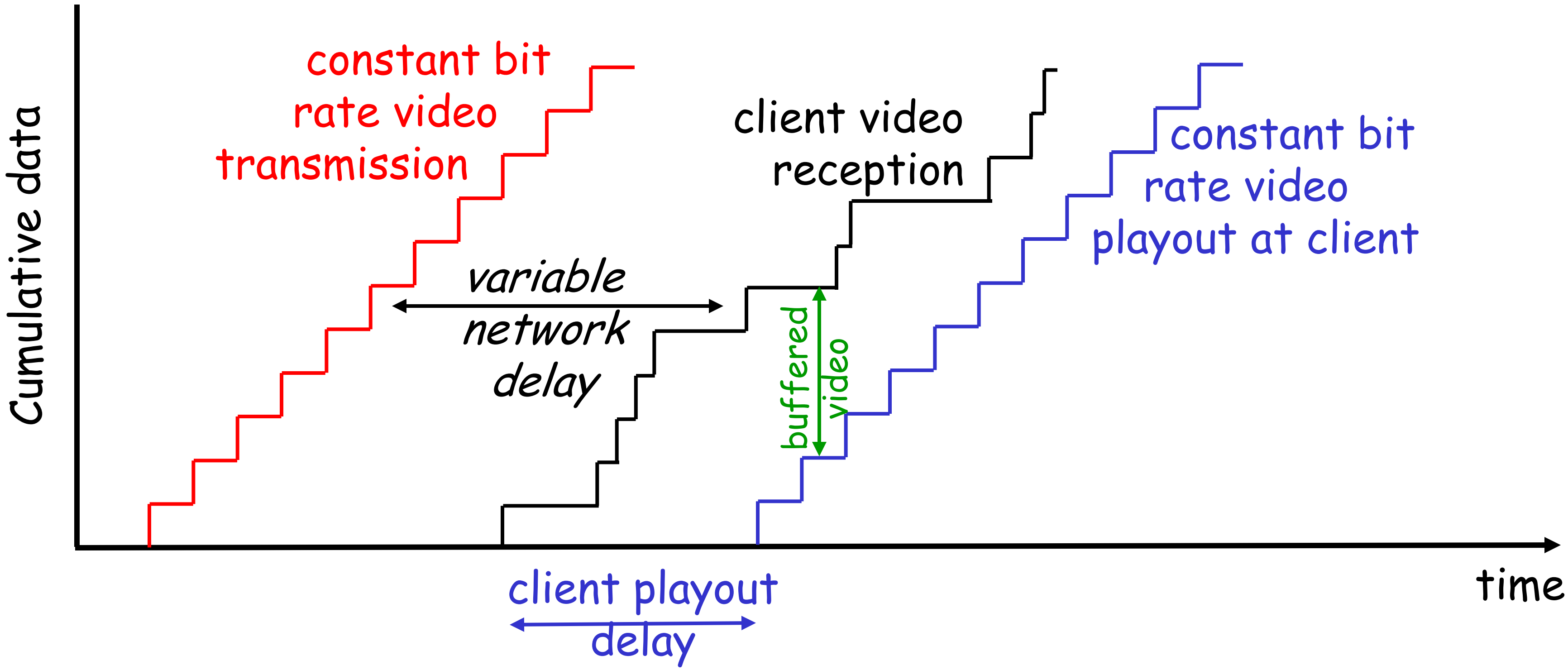
- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server streams audio/video to player

# Streaming from a streaming server



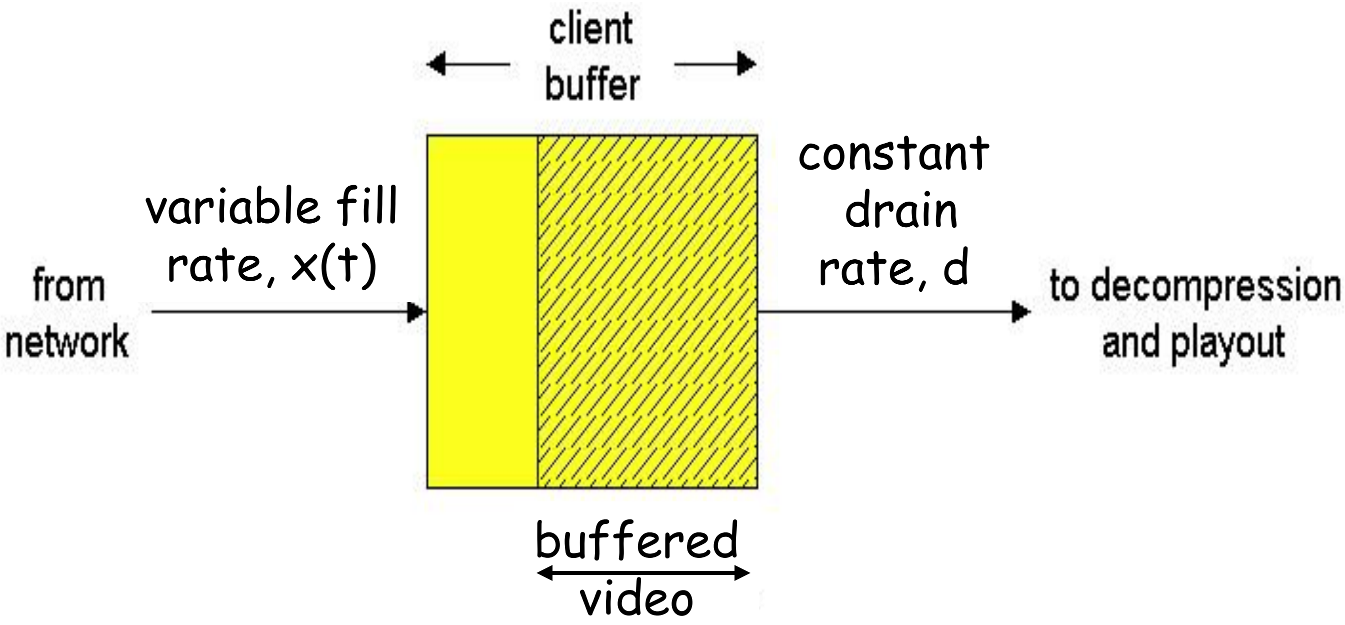
- This architecture allows for non-HTTP protocol between server and media player
- Can also use UDP instead of TCP.

# 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?

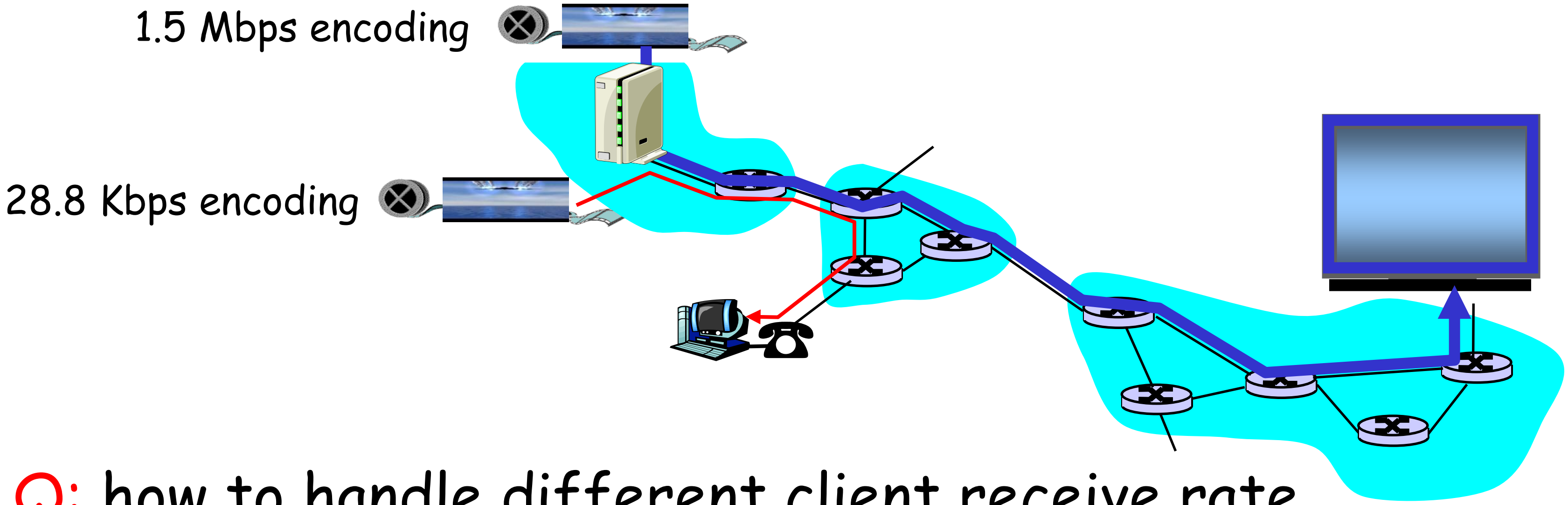
## UDP

- 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 compensate for network delay jitter
- error recover: time permitting

## 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
- 100Mbps 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.
- For user to control display: rewind, fast forward, pause, resume, repositioning, etc…

## What it doesn't do:

- does not define how audio/video is encapsulated for streaming over network
- does not restrict how streamed media is transported; it can be transported over UDP or TCP
- does not specify how the media player buffers audio/video

# RTSP: out of band control

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

- A file is transferred over one TCP connection.
- Control information (directory changes, file deletion, file renaming, etc.) is sent over a separate TCP connection.
- The "out-of-band" and "in-band" channels use different port numbers.

## RTSP messages are also sent out-of-band:

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

# RTSP Example

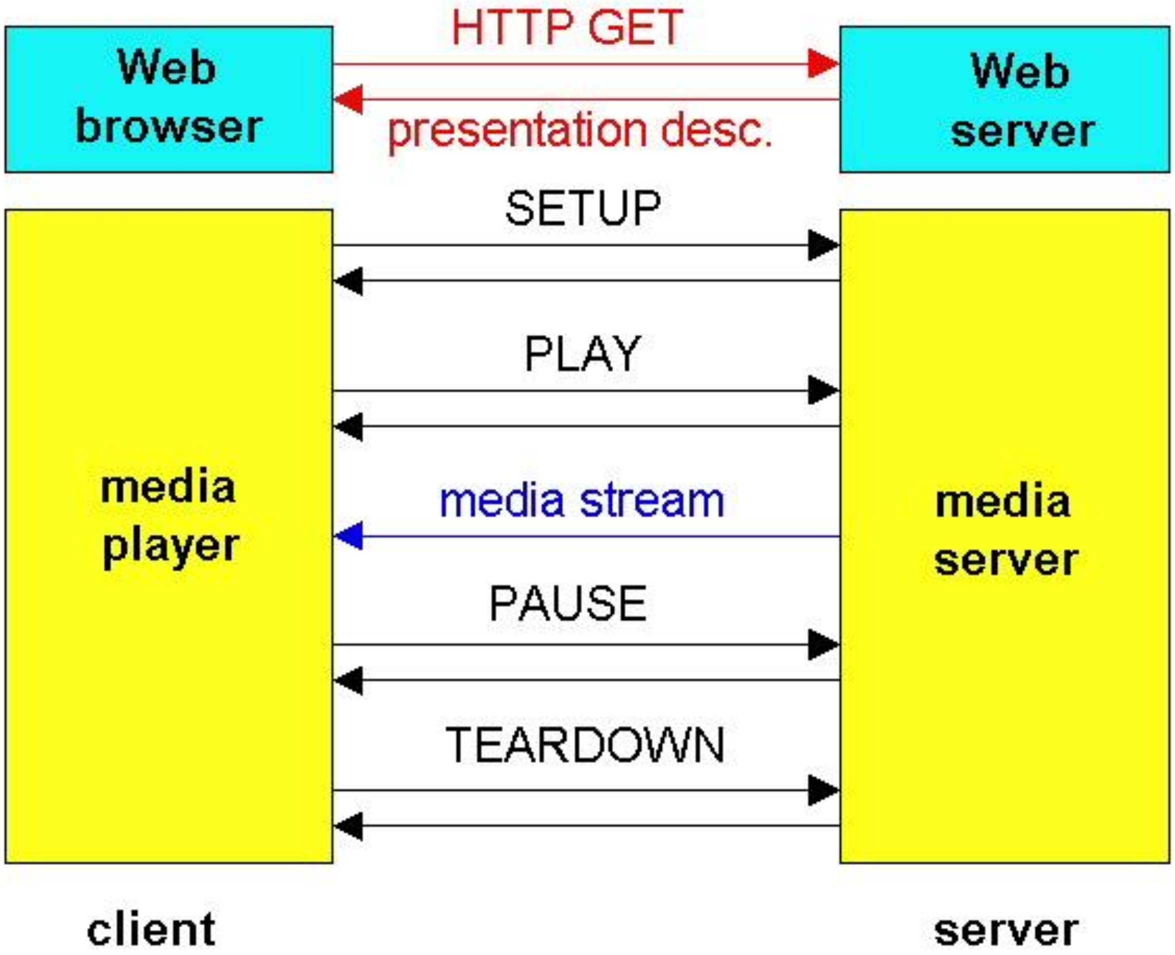
Scenario:

- 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 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone case study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# Real-time interactive applications

- PC-2-PC phone
  - instant messaging services are providing this
- PC-2-phone
  - Dialpad
  - Net2phone
- videoconference with Webcams

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

# *Interactive* Multimedia: Internet Phone

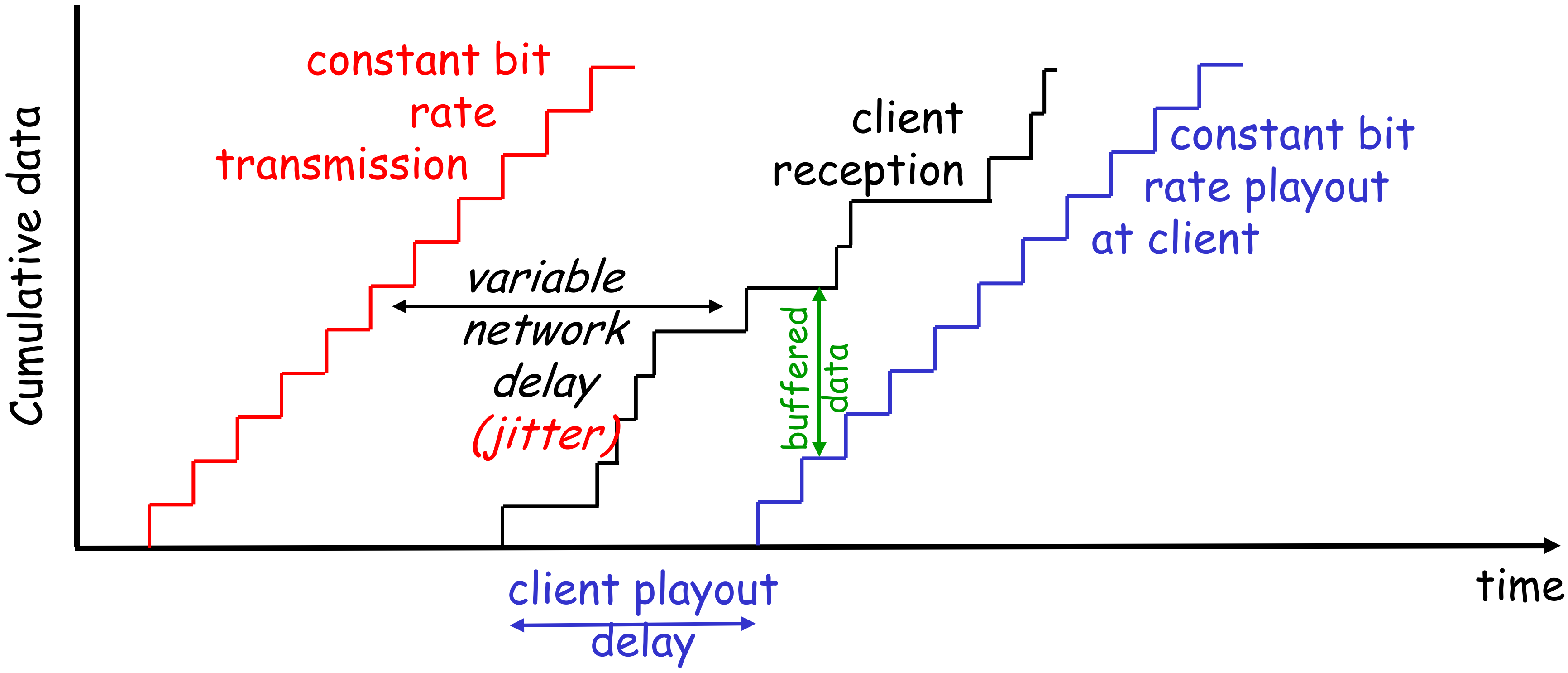
Introduce Internet Phone by way of an example

- speaker's audio: alternating talk spurts, silent periods.
  - 64 kbps during talk spurt
- pkts generated only during talk spurts
  - 20 msec chunks at 8 Kbytes/sec: 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



constant bit rate transmission
variable network delay (jitter)
client reception
buffered data
constant bit rate playout at client
client playout delay
Cumulative data
time

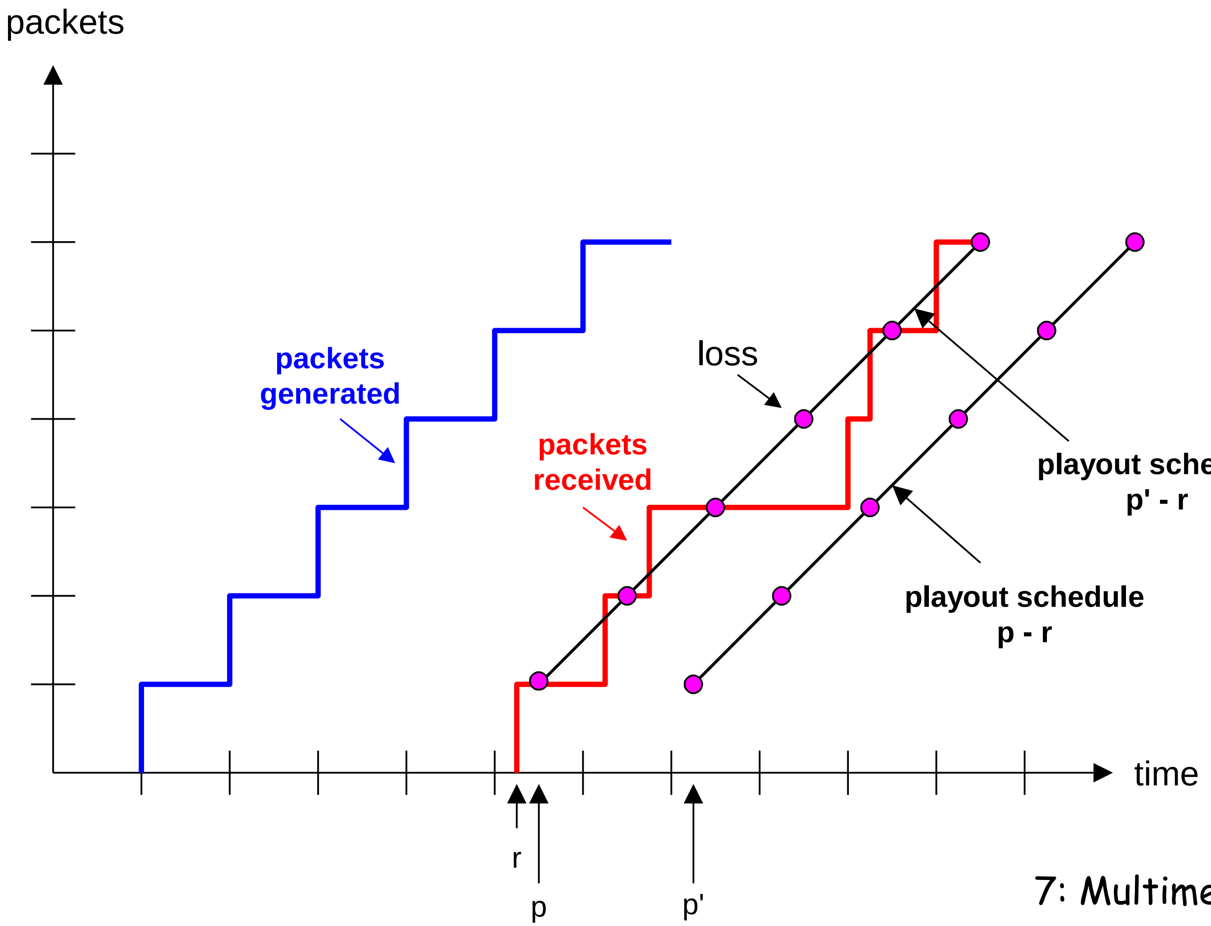
- Consider the end-to-end delays of two consecutive packets: difference can be more or less than 20 msec

# 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 for 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, I

- 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.

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

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

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

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

$\mathrm{d_i}$ = 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 II

Also useful to estimate the average deviation of the delay, $v_i$:

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

The estimates $d_i$ and $v_i$ are calculated for every received packet, although they are only used at the beginning of a talk spurt.

For first packet in talk spurt, playout time is:

$$p_i = t_i + d_i + Kv_i$$

where K is a positive constant.

Remaining packets in talkspurt are played out periodically

# Adaptive Playout, III

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.

# Recovery from packet loss (1)

forward error correction (FEC): simple scheme

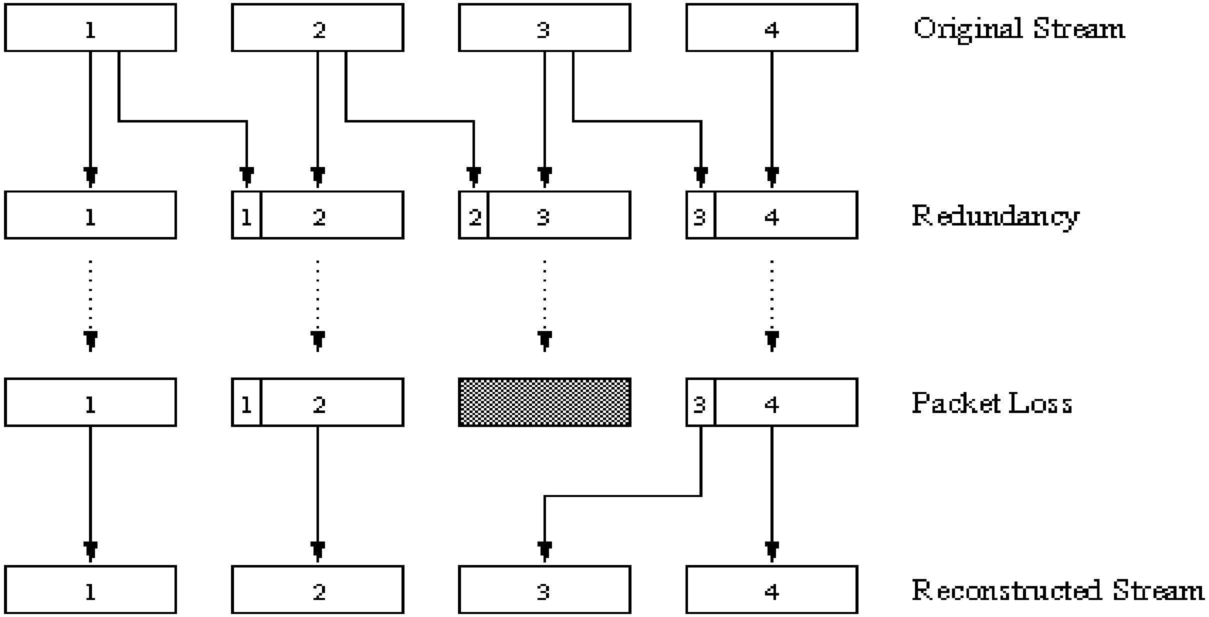
- for every group of n chunks create a redundant chunk by exclusive OR-ing the n original chunks
- send out n+1 chunks, increasing the bandwidth by factor 1/n.
- can reconstruct the original n chunks if there is at most one lost chunk from the n+1 chunks

- Playout delay needs to be fixed to the time to receive all n+1 packets
- Tradeoff:
  - increase n, less bandwidth waste
  - increase n, longer playout delay
  - increase n, higher probability that 2 or more chunks will be lost

# Recovery from packet loss (2)

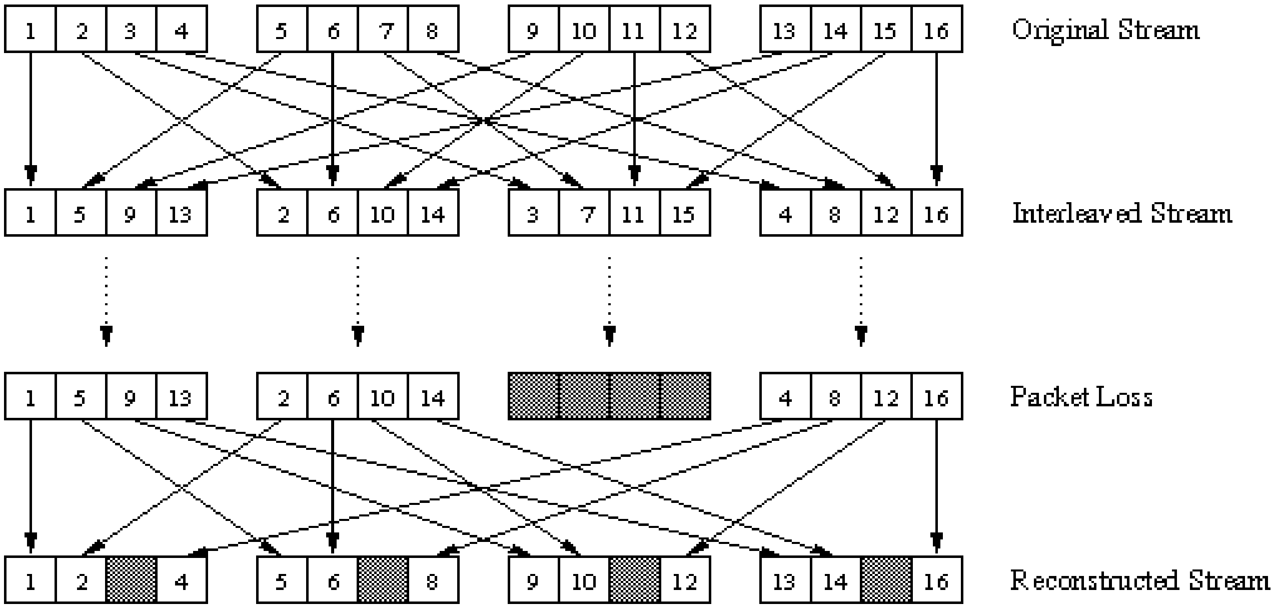
**2nd FEC scheme**

- "piggyback lower quality stream"
- send lower resolution audio stream as the redundant information
- for example, nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps.



- Whenever there is non-consecutive loss, the receiver can conceal the loss.
- Can also append (n-1)st and (n-2)nd low-bit rate chunk

# Recovery from packet loss (3)



## Interleaving

- chunks are broken up into smaller units
- for example, 4 5 msec units per chunk
- Packet contains small units from different chunks

- if packet is lost, still have most of every chunk
- has no redundancy overhead
- but adds to playout delay

# 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
  - retransmissions, time permitting
  - conceal errors: repeat nearby data

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

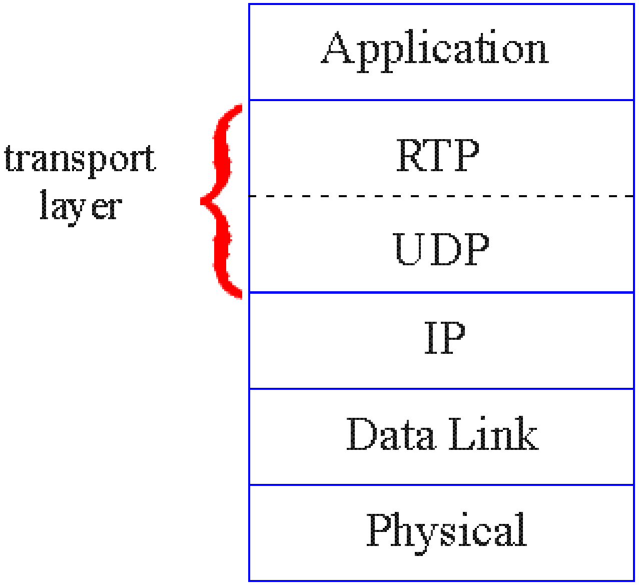
# Real-Time Protocol (RTP)

- RTP specifies a packet structure for packets carrying audio and video data
- RFC 1889.
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - timestamping
- RTP runs in the end systems.
- RTP packets are 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 a transport-layer interface that extend UDP:

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



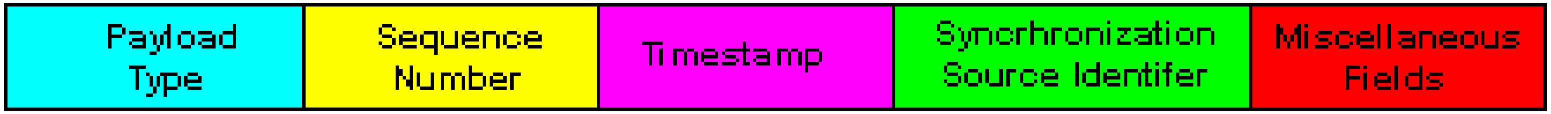
# RTP Example

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

# RTP and QoS

- RTP does **not** provide any mechanism to ensure timely delivery of data or provide other quality of service guarantees.
- RTP encapsulation is only seen at the end systems: it is not seen by intermediate routers.
  - Routers providing best-effort service do not make any special effort to ensure that RTP packets arrive at the destination in a timely matter.

# RTP Header


Payload Type
Sequence Number
Timestamp
Synchrhronization Source Identifer
Miscellaneous Fields

RTP Header

**Payload Type (7 bits):** Indicates type of encoding currently being used. If sender changes encoding in middle of conference, sender informs the receiver through this 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).** Reflects the sampling instant of the first byte in the RTP data packet.
  - For audio, timestamp clock typically increments by one for each sampling period (for example, each 125 usecs for a 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 the source of the RTP stream. Each stream in a RTP session should have a distinct SSRC.

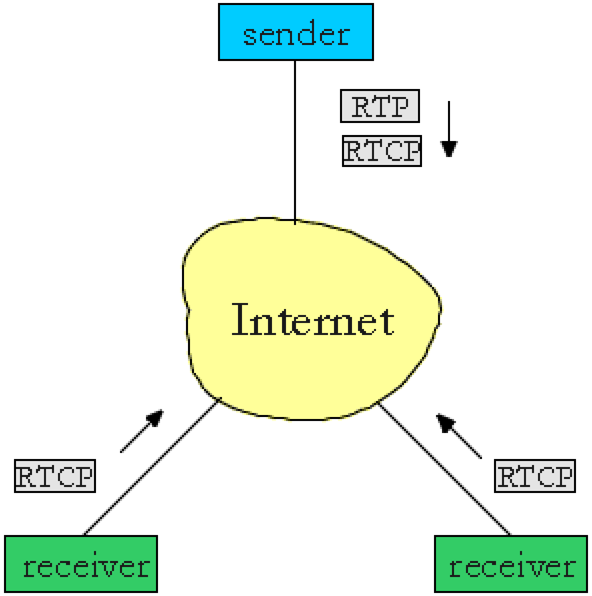
# RTSP/RTP Programming Assignment

- Build a server that encapsulates stored video frames into RTP packets
  - grab video frame, add RTP headers, create UDP segments, send segments to UDP socket
  - include seq numbers and time stamps
  - client RTP provided for you
- Also write the client side of RTSP
  - issue play and pause commands
  - server RTSP provided for you

# 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
- Statistics include number of packets sent, number of packets lost, interarrival jitter, etc.
- Feedback can be used to control performance
  - Sender may modify its transmissions based on feedback

# RTCP - Continued



- For an RTP session there is typically a single multicast address; all RTP and RTCP packets belonging to the session use the multicast address.

- RTP and RTCP packets are distinguished from each other through the use of distinct port numbers.

- To limit traffic, each participant reduces his RTCP traffic as the 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 the RTP stream, the current time, the number of packets sent, and the 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 and one for audio.
- Timestamps in RTP packets tied to the video and audio sampling clocks
  - not tied to the wall-clock time
- Each RTCP sender-report packet contains (for the most recently generated packet in the associated RTP stream):
  - timestamp of the RTP packet
  - wall-clock time for when packet was created.
- Receivers can use this association to synchronize the playout of audio and video.

# RTCP Bandwidth Scaling

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

Example

- Suppose one sender, sending video at a rate of 2 Mbps. Then RTCP attempts to limit its traffic to 100 Kbps.
- RTCP gives 75% of this rate to the receivers; remaining 25% to the sender
- The 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 the entire session) and dividing by allocated rate.

# SIP

- Session Initiation Protocol
- Comes from IETF

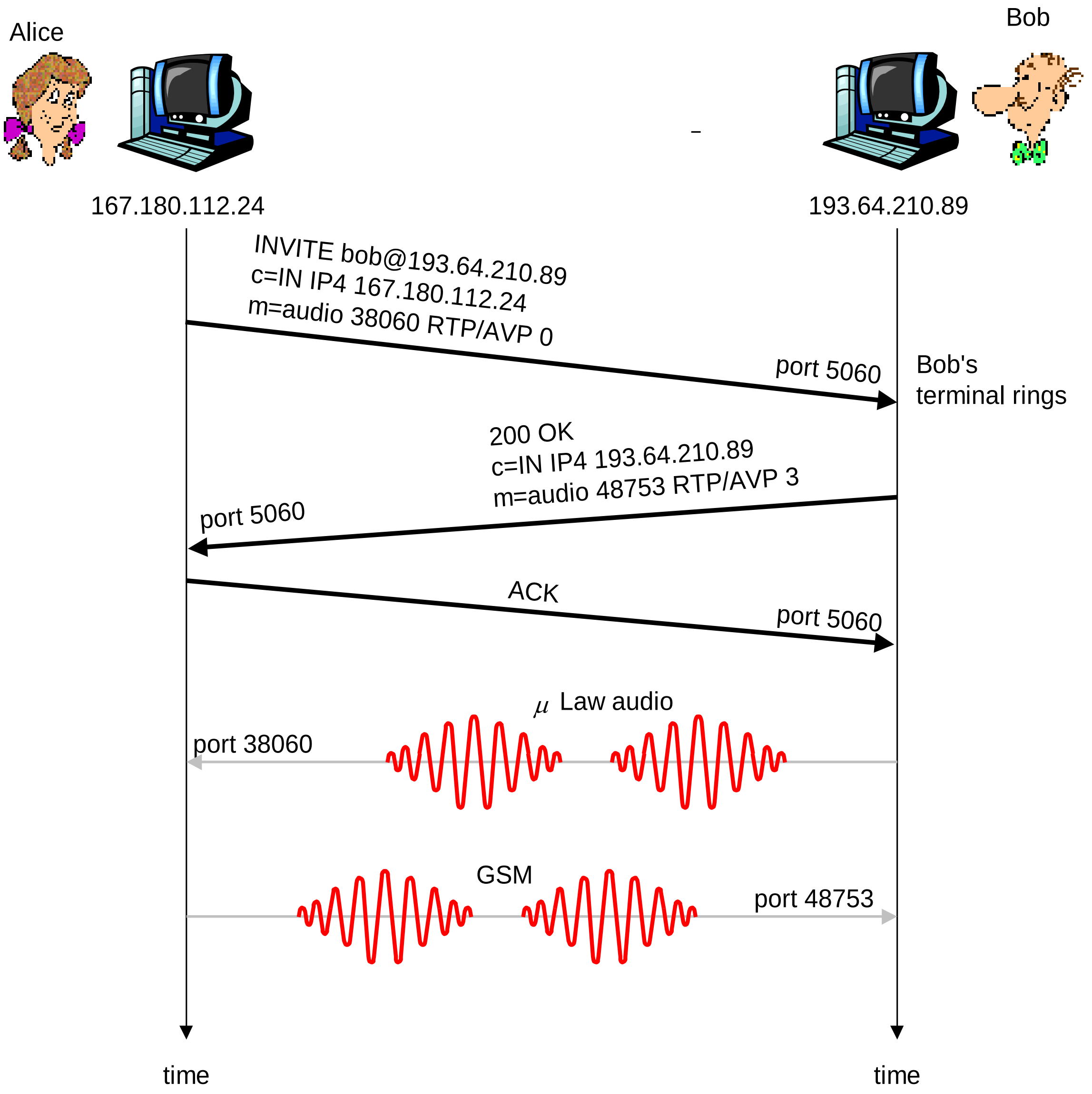
## SIP long-term vision

- All telephone calls and video conference calls take place over the Internet
- People are identified by names or e-mail addresses, rather than by phone numbers.
- You can reach the callee, no matter where the callee roams, no matter what IP device the callee is currently using.

# SIP Services

- Setting up a call
  - Provides mechanisms for caller to let callee know she wants to establish a call
  - Provides mechanisms so that caller and callee can agree on media type and encoding.
  - Provides mechanisms 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 and hold calls

# Setting up a call to a known IP address



- Alice's SIP invite message indicates her port number & IP address. Indicates encoding that Alice 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 and list encoders he can use.
  - Alice can then send a new INVITE message, advertising an appropriate encoder.

- Rejecting the 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 will be necessary.
- Alice sends and receives SIP messages using the SIP default port number 506.
- Alice specifies in Via: header that SIP client sends and receives SIP messages over UDP

# Name translation and user locataion

- 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

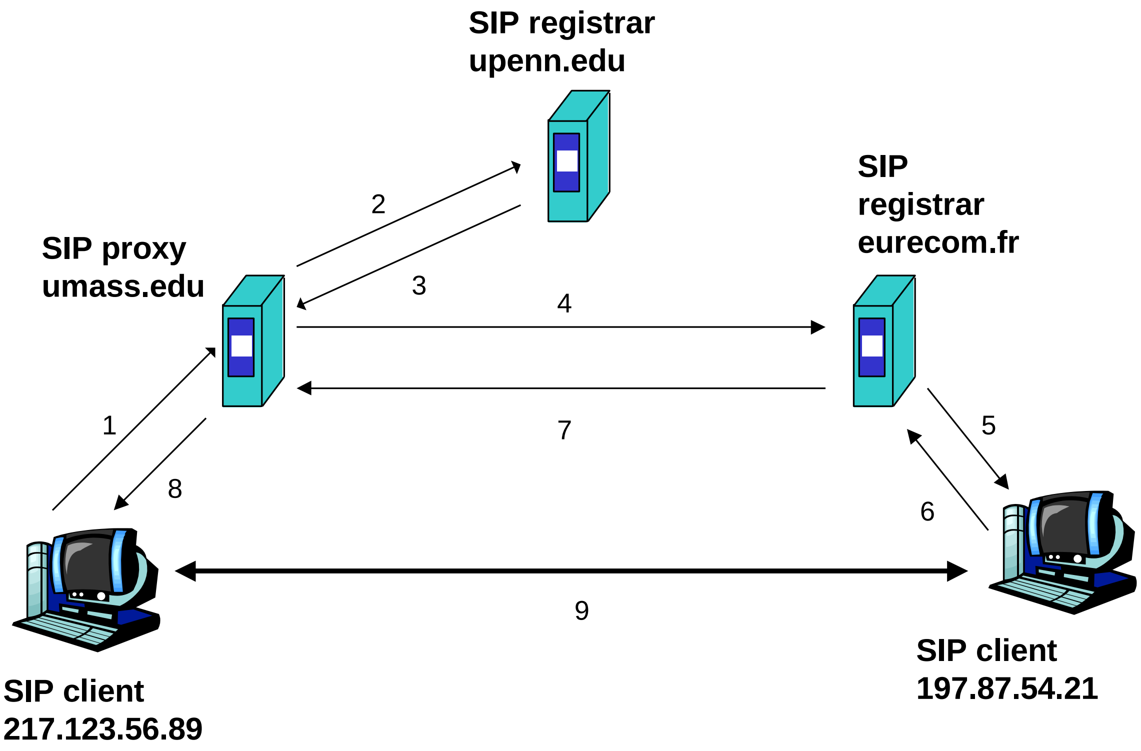
- Alice sends invite message to her proxy server
  - contains address sip:bob@domain.com
- Proxy responsible for routing SIP messages to callee
  - possibly through multiple proxies.
- Callee sends response back through the same set of proxies.
- Proxy returns SIP response message to Alice
  - contains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

- Note: proxy is analogous to local DNS server

# Example

Caller jim@umass.edu
with 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.

# Comparison with H.323

- H.323 is another signaling protocol for real-time, interactive
- H.323 is a complete, vertically integrated suite of protocols for multimedia conferencing: signaling, registration, admission control, transport and codecs.
- SIP is a single component. Works with RTP, but does not mandate it. Can be combined with other protocols and services.
- H.323 comes from the ITU (telephony).
- SIP comes from IETF: Borrows much of its concepts from HTTP. SIP has a Web flavor, whereas H.323 has a telephony flavor.
- SIP uses the KISS principle: Keep it simple stupid.

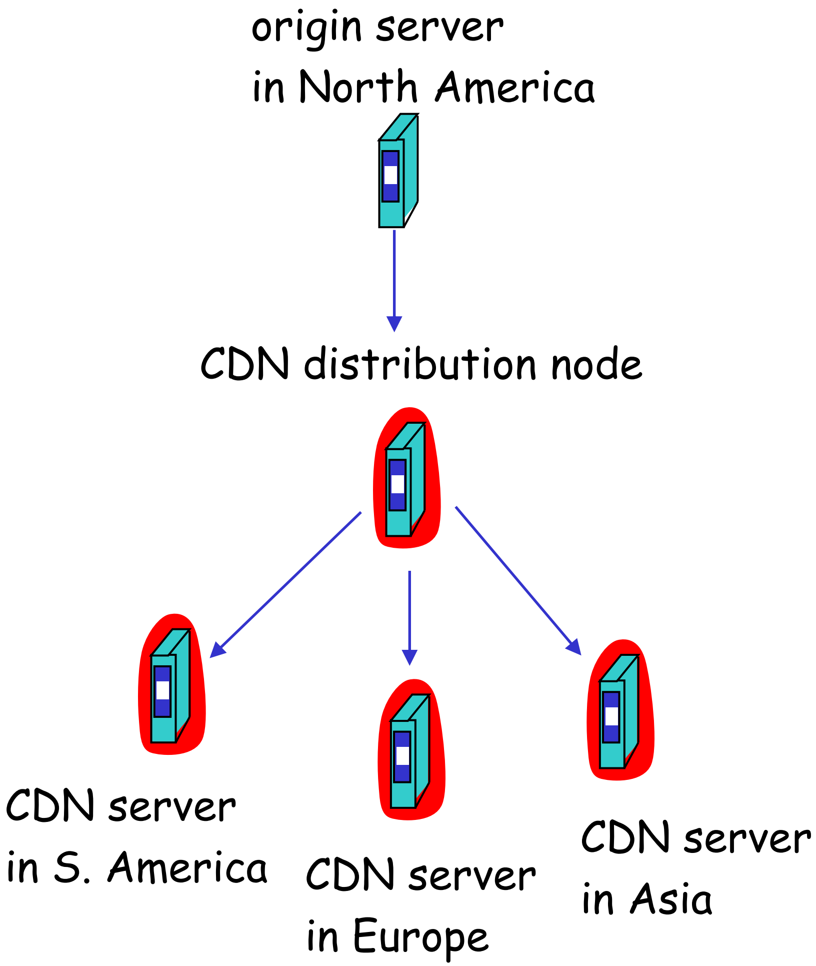
# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# Content distribution networks (CDNs)

## Content replication

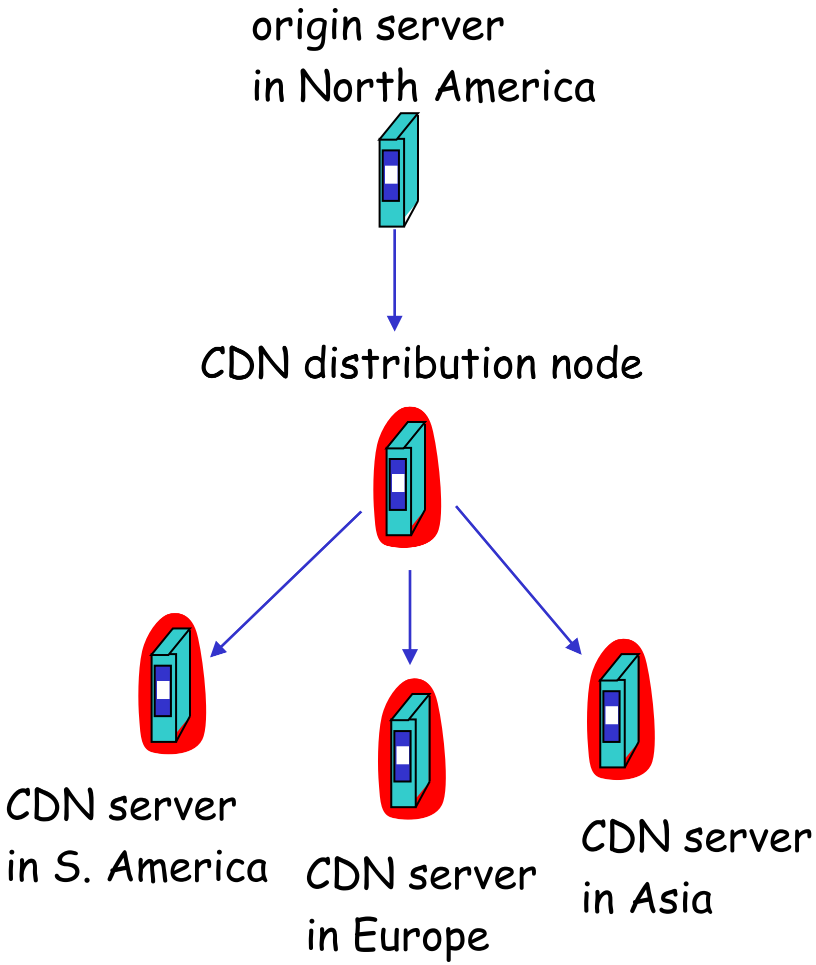
- Challenging to stream large files (e.g., video) from single origin server in real time
- Solution: replicate content at hundreds of servers throughout Internet
  - content downloaded to CDN servers ahead of time
  - placing content "close" to user avoids impairments (loss, delay) of sending content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



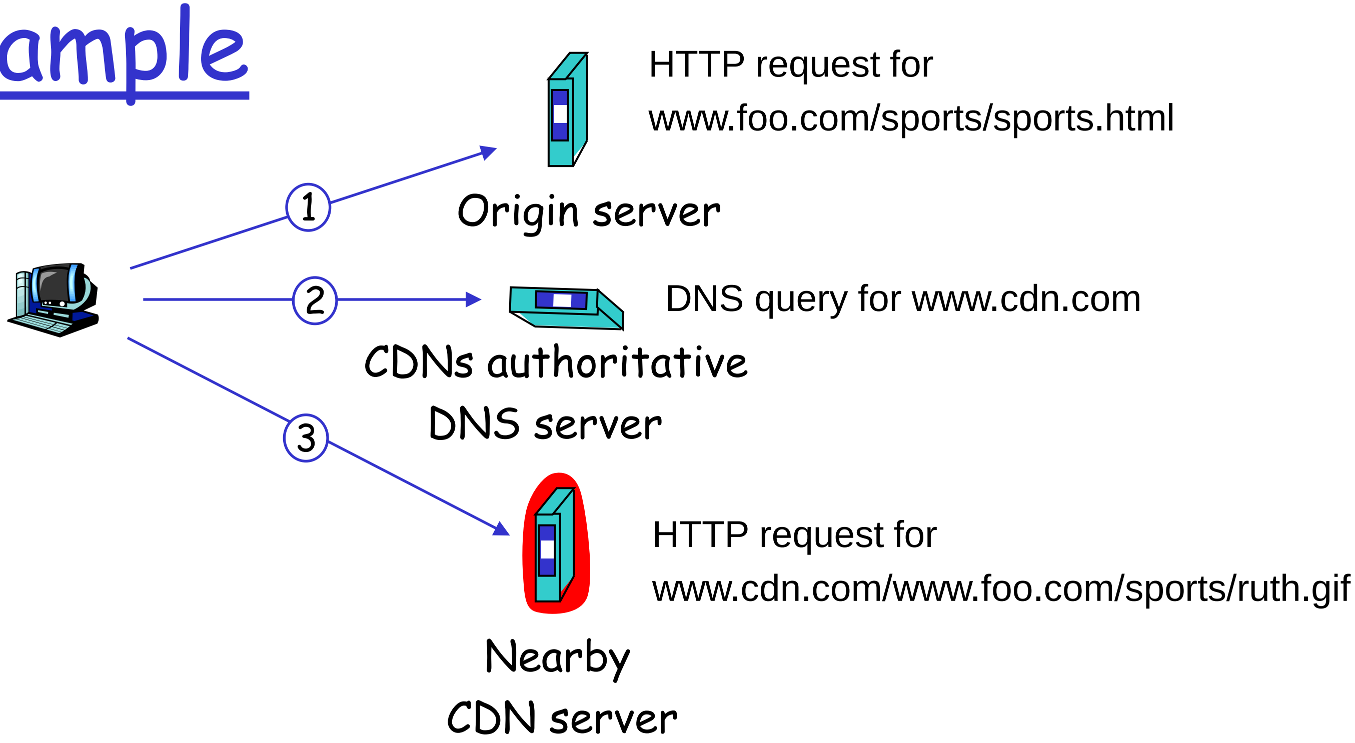
# Content distribution networks (CDNs)

## Content replication

- 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


HTTP request for
www.foo.com/sports/sports.html
1
Origin server
2
DNS query for www.cdn.com
CDNs authoritative
DNS server
3
HTTP request for
www.cdn.com/www.foo.com/sports/ruth.gif
Nearby
CDN server

## 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

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

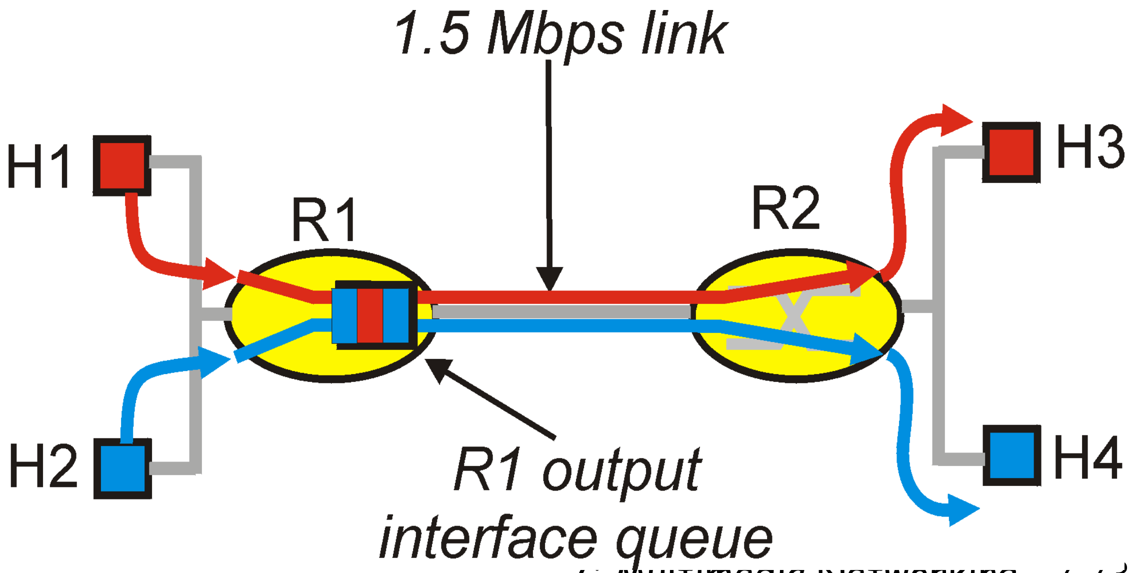
# Improving QOS in IP Networks

Thus far: “making the best of best effort”

Future: next generation Internet with QoS guarantees

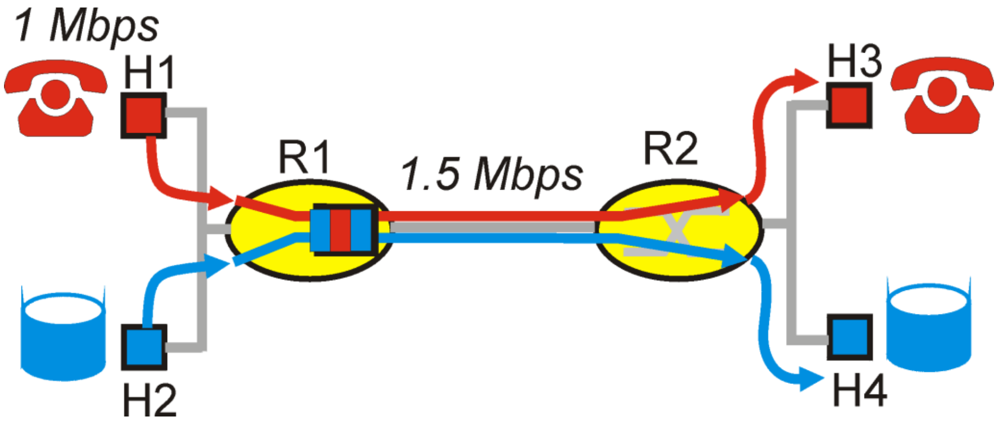
- RSVP: signaling for resource reservations
- Differentiated Services: differential guarantees
- Integrated Services: firm guarantees

- simple model for sharing and congestion studies:



# Principles for QOS Guarantees

- Example: 1MbpsI P phone, FTP share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - want to give priority to audio over FTP

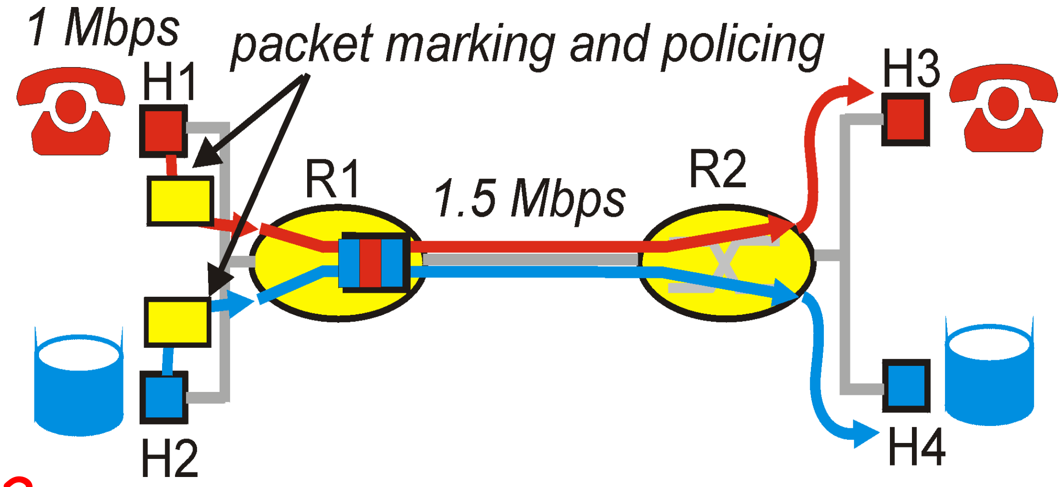


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

# Principles for QOS Guarantees (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

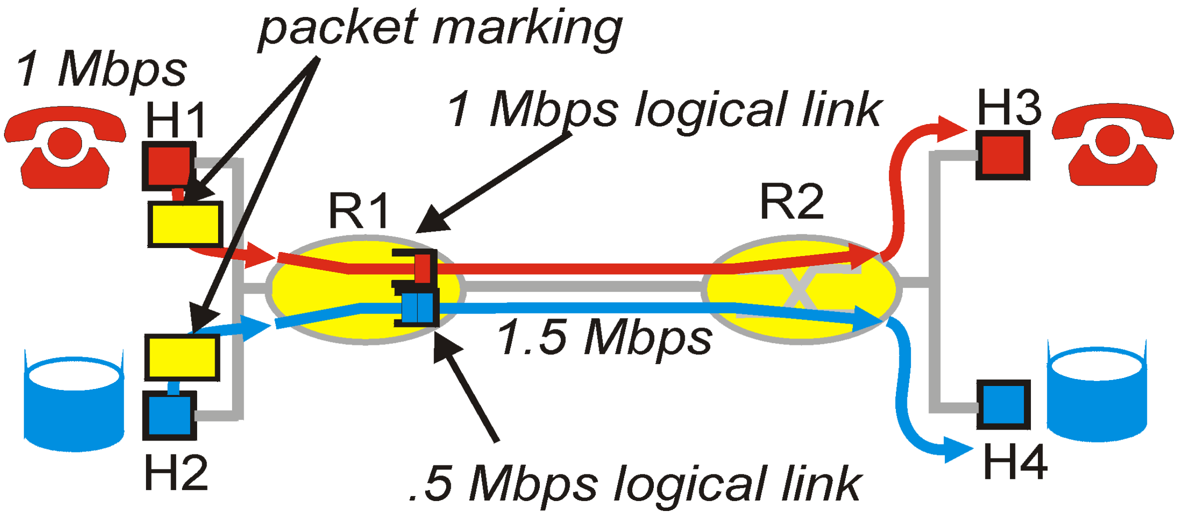


Principle 2

provide protection (*isolation*) for one class from others

# Principles for QOS Guarantees (more)

- Allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

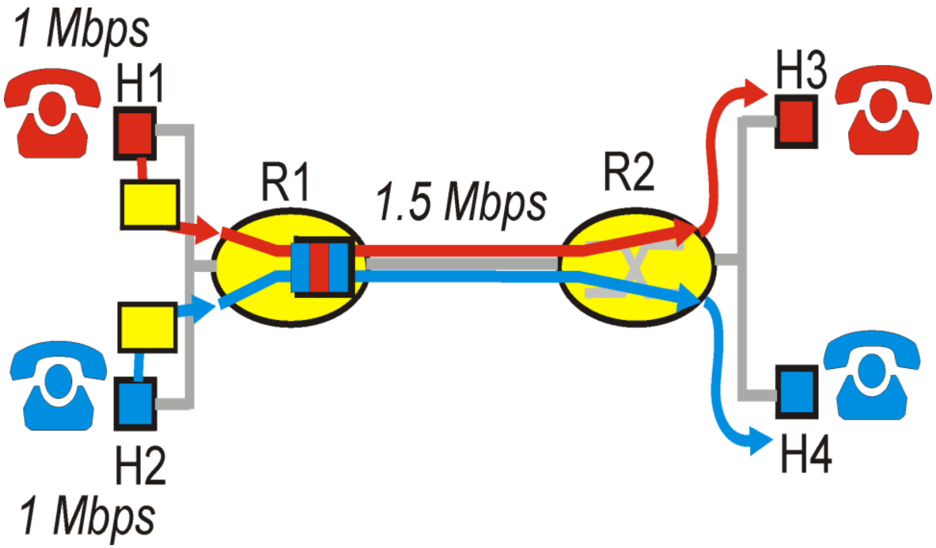


Principle 3

While providing isolation, it is desirable to use resources as efficiently as possible

# Principles for QOS Guarantees (more)

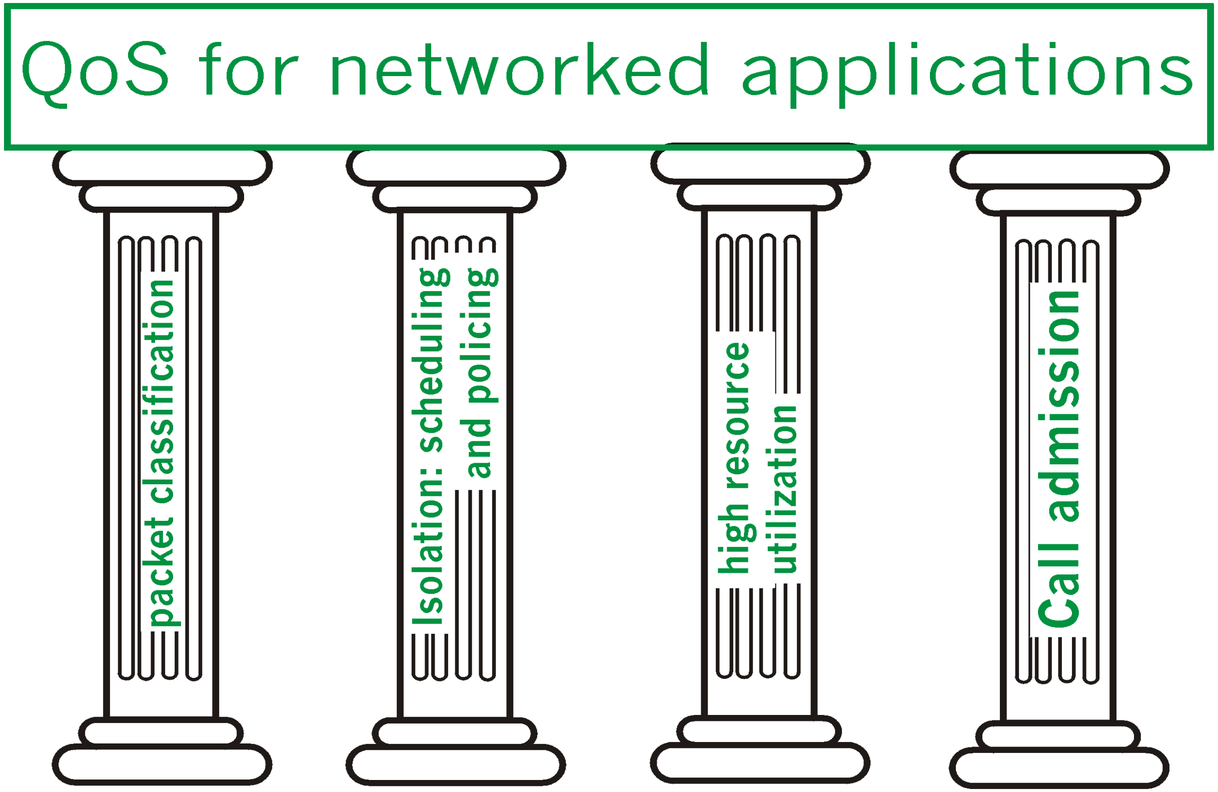
- *Basic fact of life:* can not support traffic demands beyond link capacity



Principle 4

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

# Summary of QoS Principles


QoS for networked applications
packet classification
Isolation: scheduling and policing
high resource utilization
Call admission

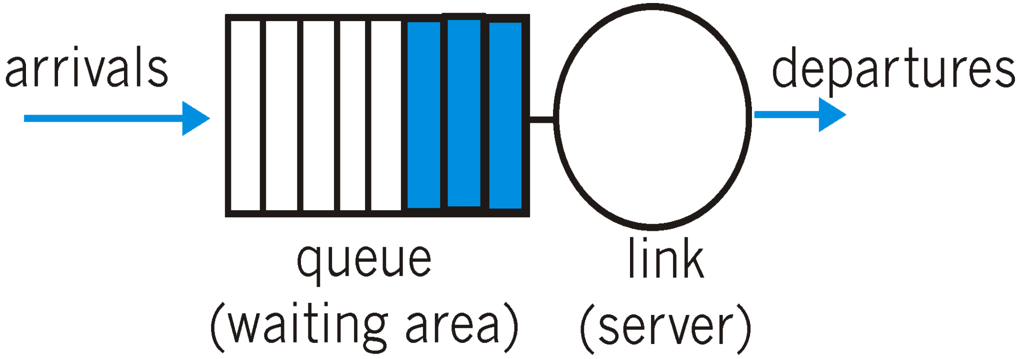
Let's next look at mechanisms for achieving this ....

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# Scheduling And Policing Mechanisms

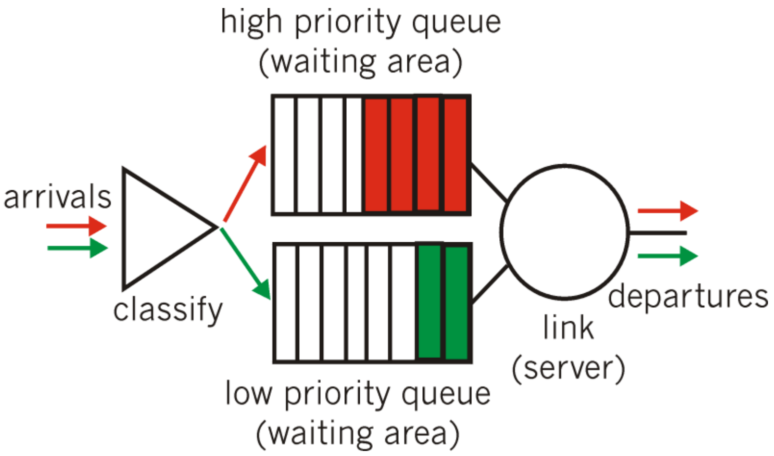
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly

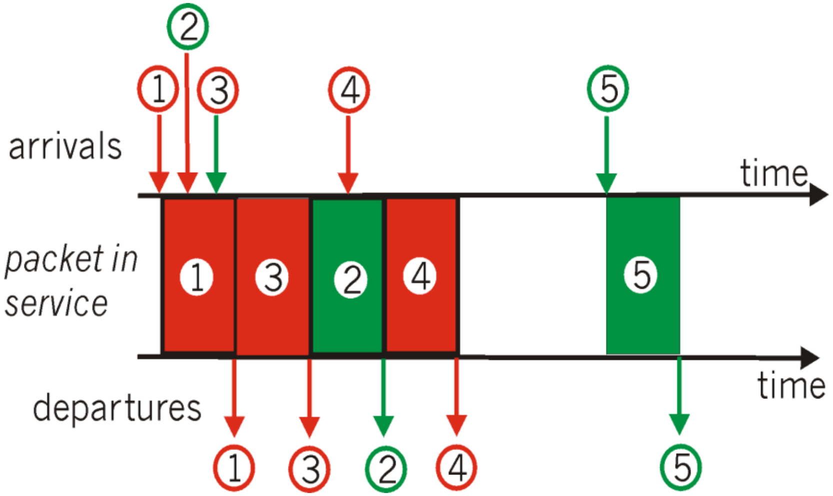


# Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..
  - Real world example?

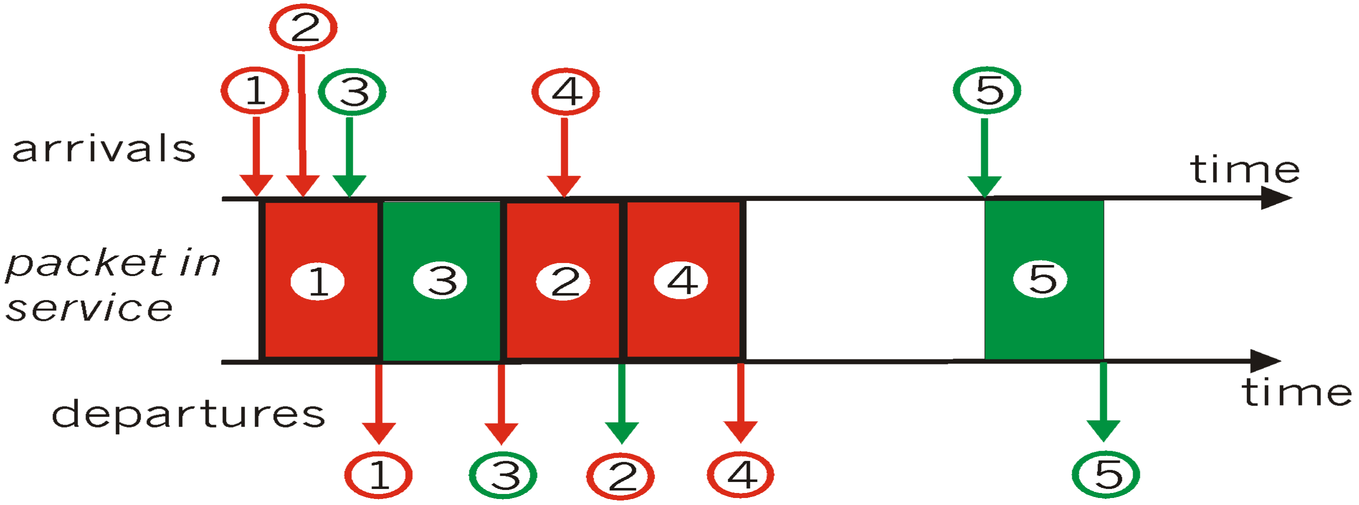




# Scheduling Policies: still more

round robin scheduling:

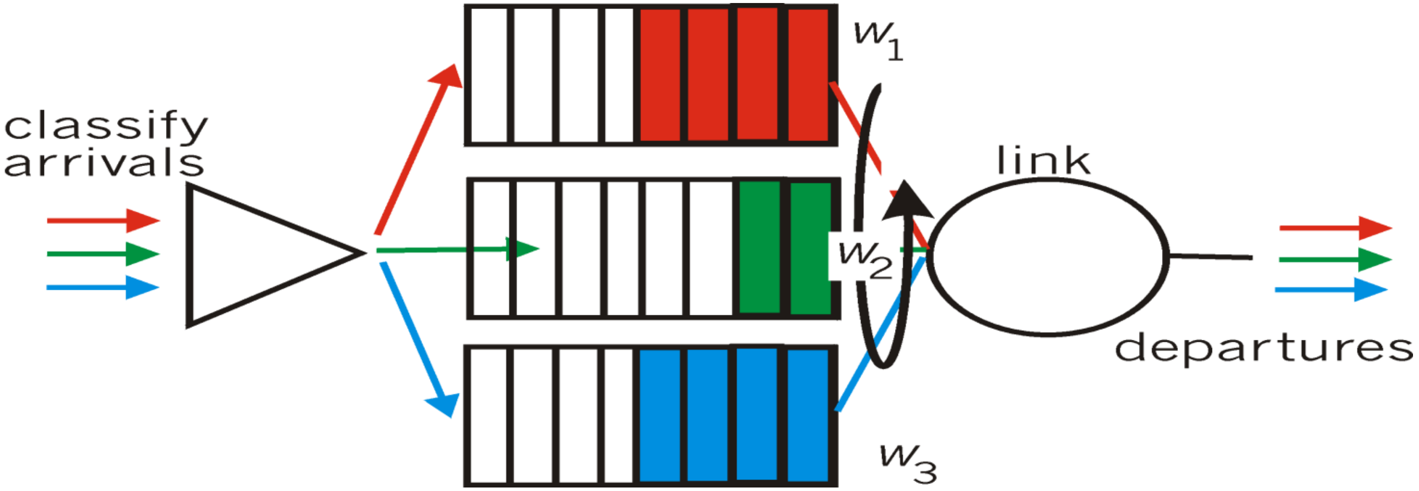
- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- real world example?



# Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?



# Policing Mechanisms

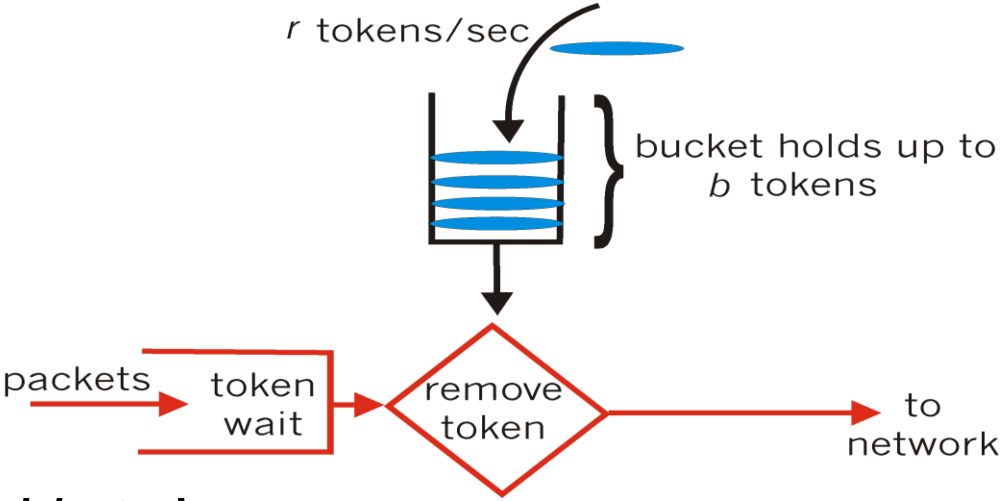
Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

- *(Long term) Average Rate:* how many pkts can be sent per unit time (in the long run)
  - crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!
- *Peak Rate:* e.g., 6000 pkts per min. (ppm) avg.; 1500 ppm peak rate
- *(Max.) Burst Size:* max. number of pkts sent consecutively (with no intervening idle)

# Policing Mechanisms

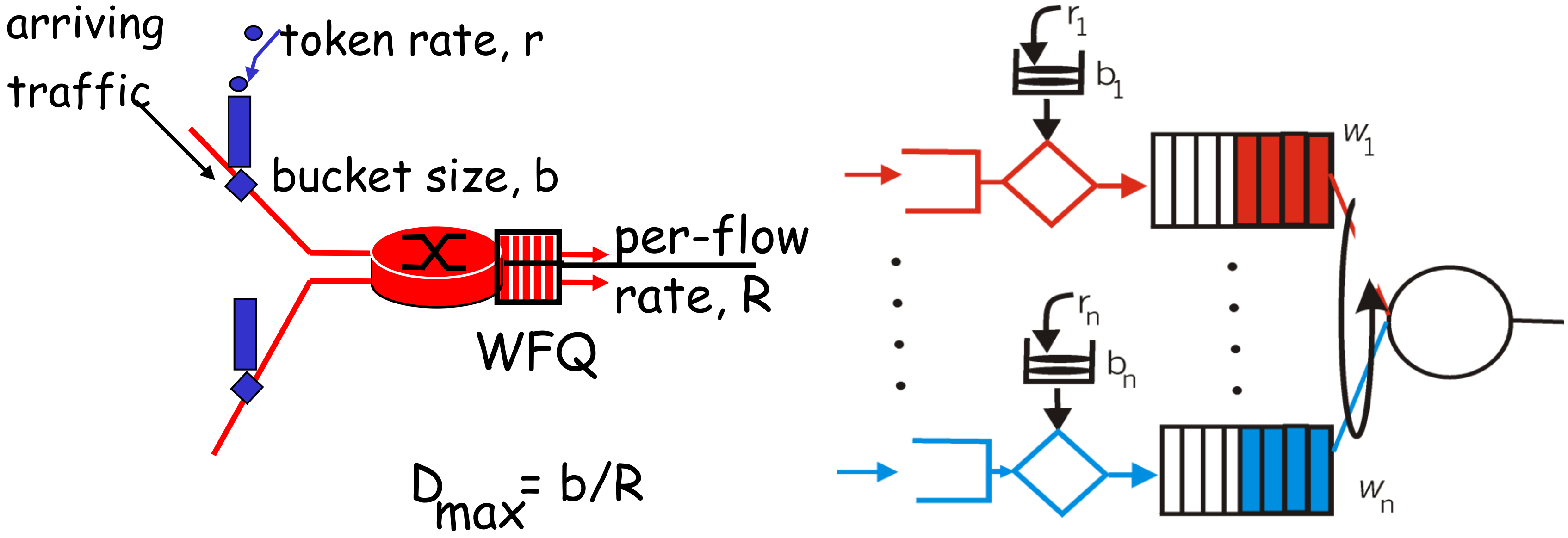
Token Bucket: limit input to specified Burst Size and Average Rate.



- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b).*

# Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., *QoS guarantee*!



# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

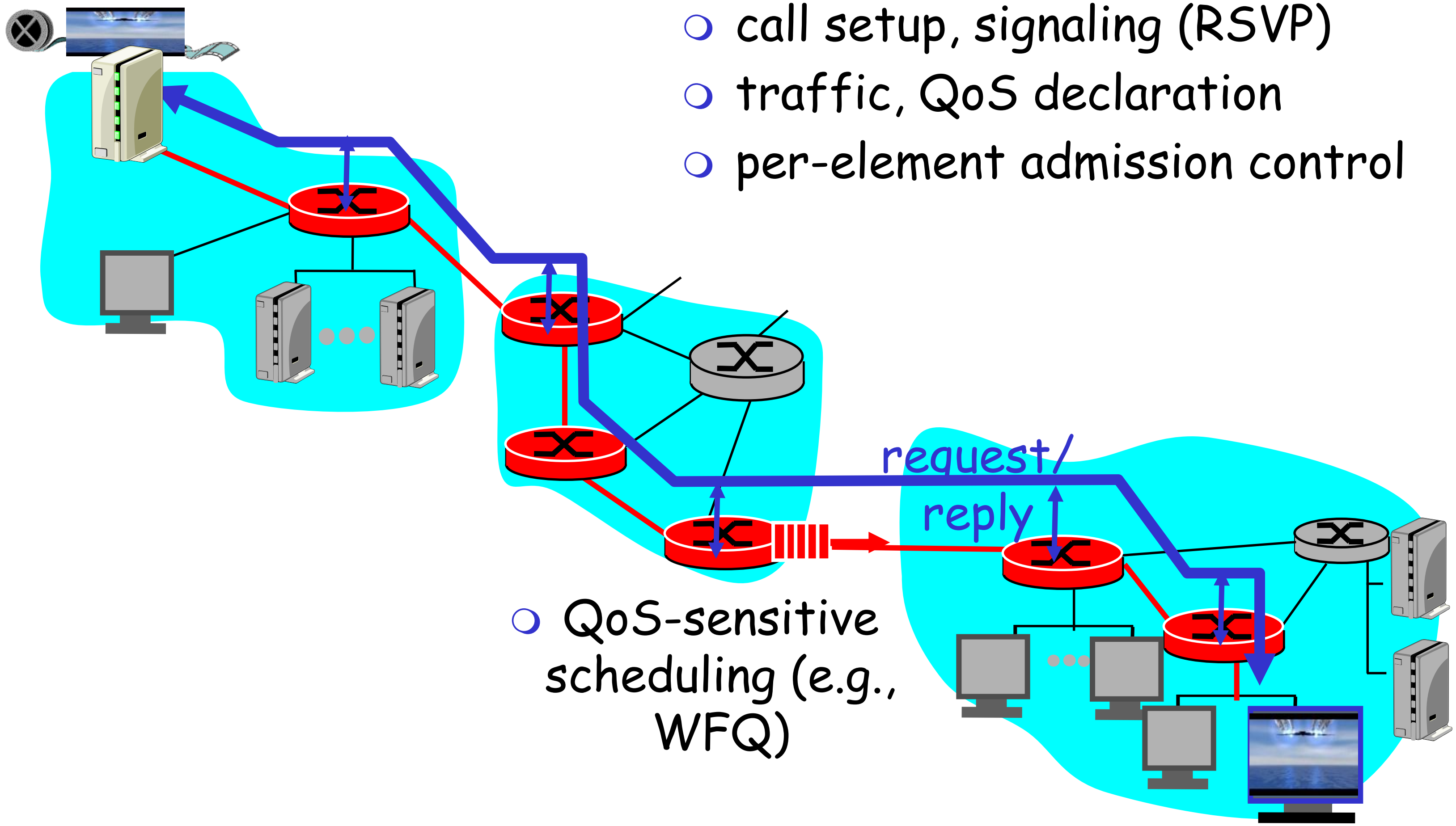
# IETF Integrated Services

- architecture for providing QOS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info (a la VC) of allocated resources, QoS req's
- admit/deny new call setup requests:

Question: can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

# Intserv: QoS guarantee scenario

- Resource reservation
  - call setup, signaling (RSVP)
  - traffic, QoS declaration
  - per-element admission control



- QoS-sensitive scheduling (e.g., WFQ)

# Call Admission

Arriving session must :

- declare its QOS requirement
  - R-spec: defines the QOS being requested
- characterize traffic it will send into network
  - T-spec: defines traffic characteristics
- signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

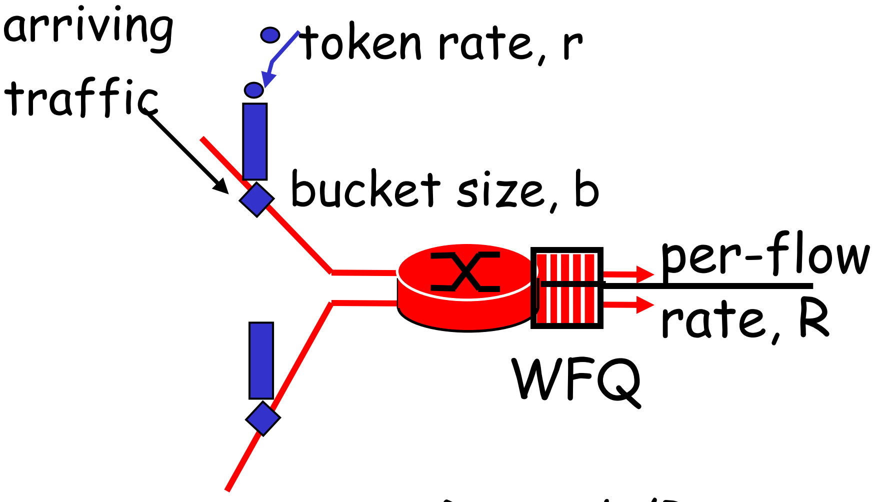
# Intserv QoS: Service models [rfc2211, rfc 2212]

## Guaranteed service:

- worst case traffic arrival: leaky-bucket-policed source
- simple (mathematically provable) *bound* on delay [Parekh 1992, Cruz 1988]

## Controlled load service:

- "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."



$D_{max} = b/R$

# IETF Differentiated Services

## Concerns with Intserv:

- Scalability: signaling, maintaining per-flow router state difficult with large number of flows
- Flexible Service Models: Intserv has only two classes. Also want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver

## Diffserv approach:

- simple functions in network core, relatively complex functions at edge routers (or hosts)
- Do't define define service classes, provide functional components to build service classes

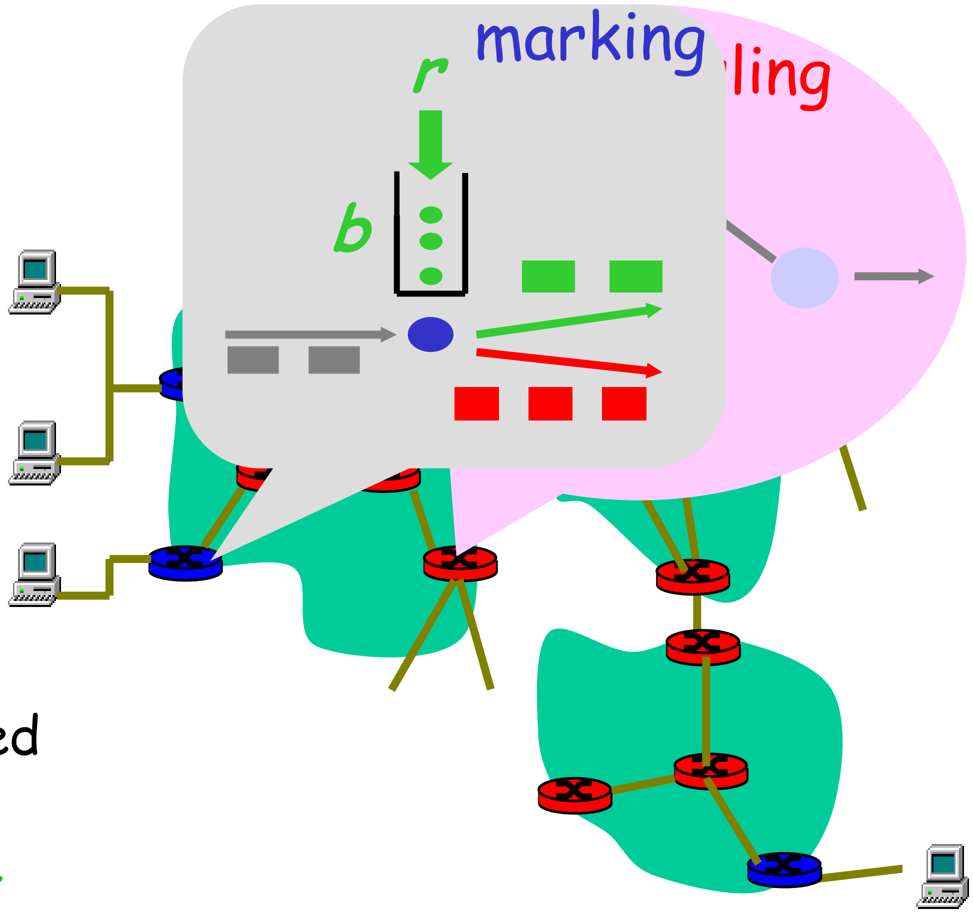
# Diffserv Architecture

## Edge router:

- per-flow traffic management
- marks packets as in-profile and out-profile

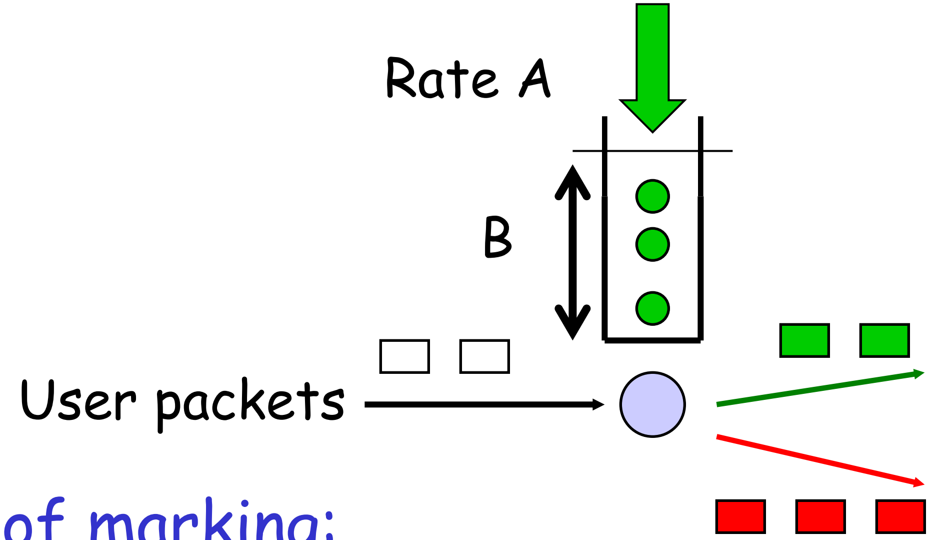
## Core router:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
- Assured Forwarding



# Edge-router Packet Marking

- profile: pre-negotiated rate A, bucket size B
- packet marking at edge based on per-flow profile

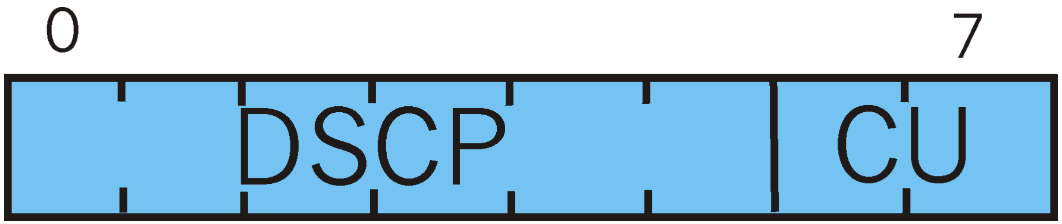


## Possible usage of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

# Classification and Conditioning

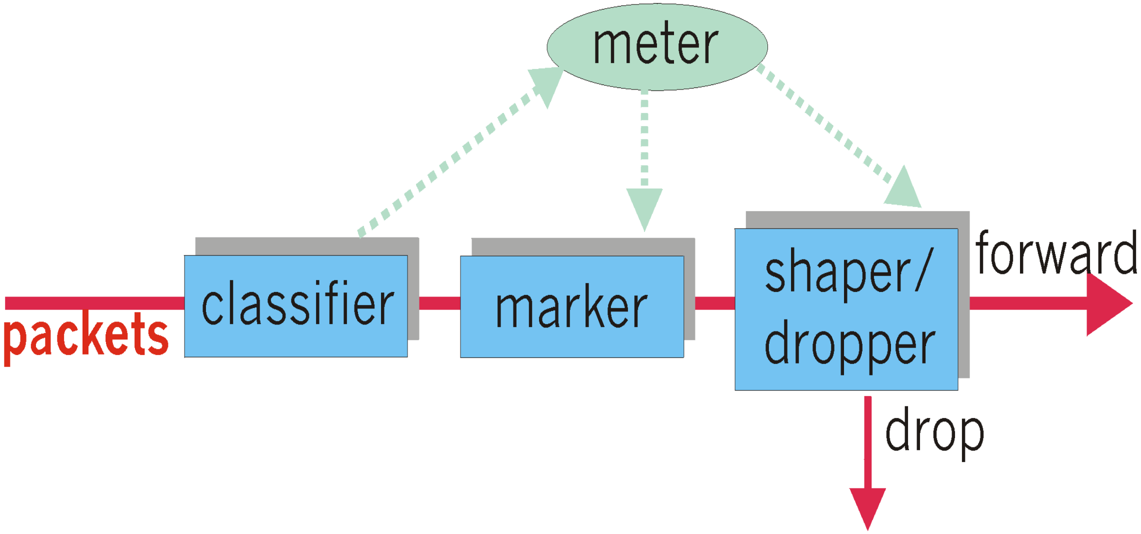
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits are currently unused



# Classification and Conditioning

may be desirable to limit traffic injection rate of some class:

- ❒ user declares traffic profile (e.g., rate, burst size)
- ❒ traffic metered, shaped if non-conforming



# Forwarding (PHB)

- PHB result in a different observable (measurable) forwarding performance behavior
- PHB does not specify what mechanisms to use to ensure required PHB performance behavior
- Examples:
  - Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - Class A packets leave first before packets from class B

# Forwarding (PHB)

PHBs being developed:

- Expedited Forwarding: pkt departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding: 4 classes of traffic
  - each guaranteed minimum amount of bandwidth
  - each with three drop preference partitions

# Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP,RTCP,SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP

# Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- earlier Internet Signaling protocol: ST-II [RFC 1819]

# RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies

# RSVP: does not…

- specify how resources are to be reserved
  - rather: a mechanism for communicating needs
- determine routes packets will take
  - that's the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

# RSVP: overview of operation

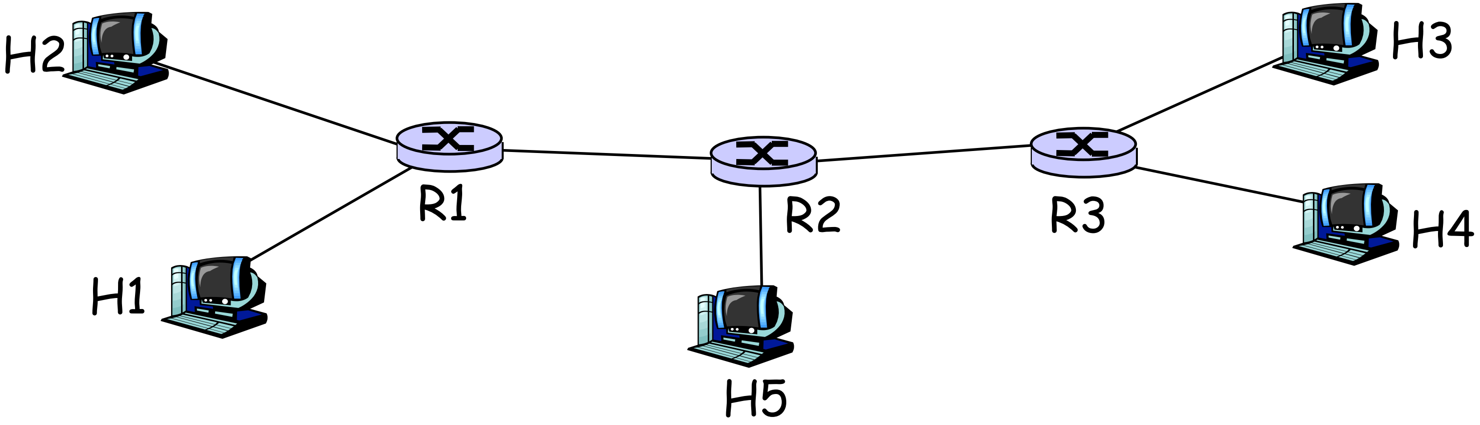
- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

# Path msgs: RSVP *sender-to-network* signaling

- path message contents:
  - *address:* unicast destination, or multicast group
  - *flowspec:* bandwidth requirements spec.
  - *filter flag:* if yes, record identities of upstream senders (to allow packets filtering by source)
  - *previous hop:* upstream router/host ID
  - *refresh time:* time until this info times out
- path message: communicates sender info, and reverse-path-to-sender routing info
  - later upstream forwarding of receiver reservations

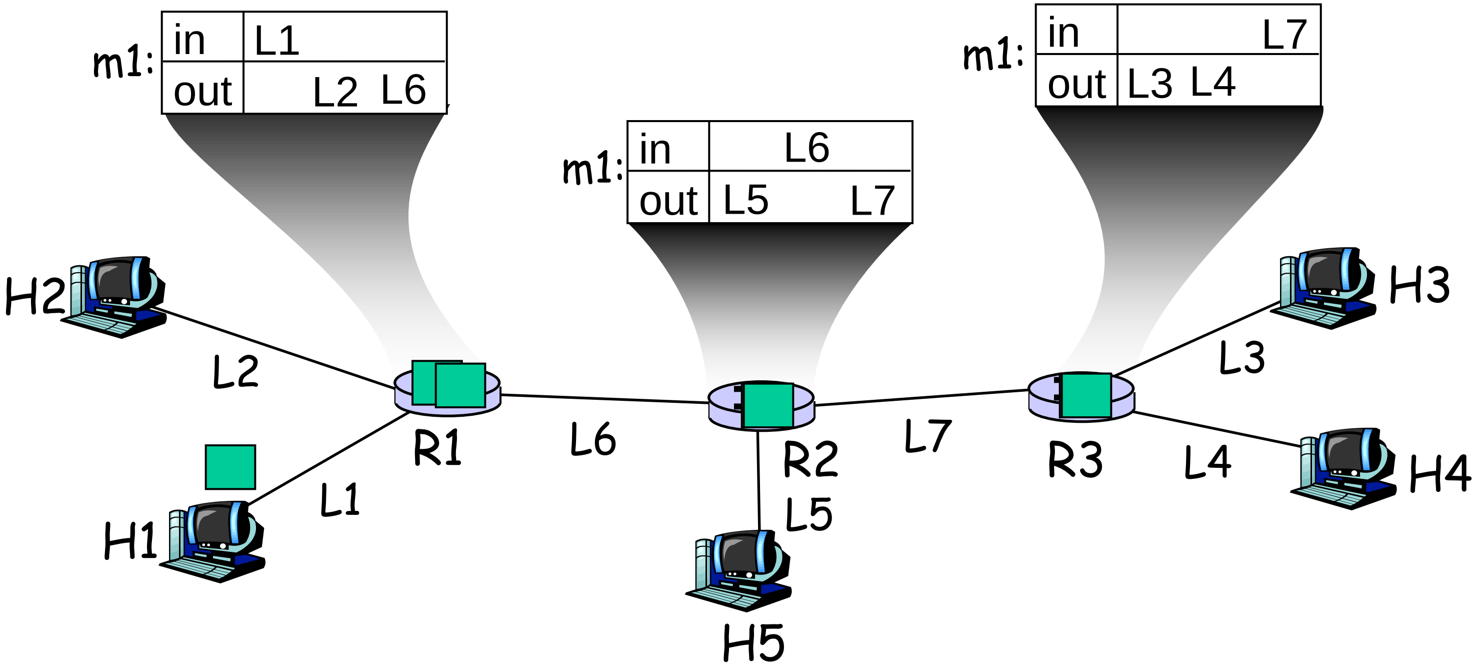
# RSVP: simple audio conference

- H1, H2, H3, H4, H5 both senders and receivers
- multicast group m1
- no filtering: packets from any sender forwarded
- audio rate: *b*
- only one multicast routing tree possible



# RSVP: building up path state

- H1, …, H5 all send path messages on *m1:*
  (address=*m1*, Tspec=*b*, filter-spec=no-filter,refresh=100)
- Suppose H1 sends first path message



m1:

| | | |
|---|---|---|
| in | L1 | |
| out | L2 | L6 |

m1:

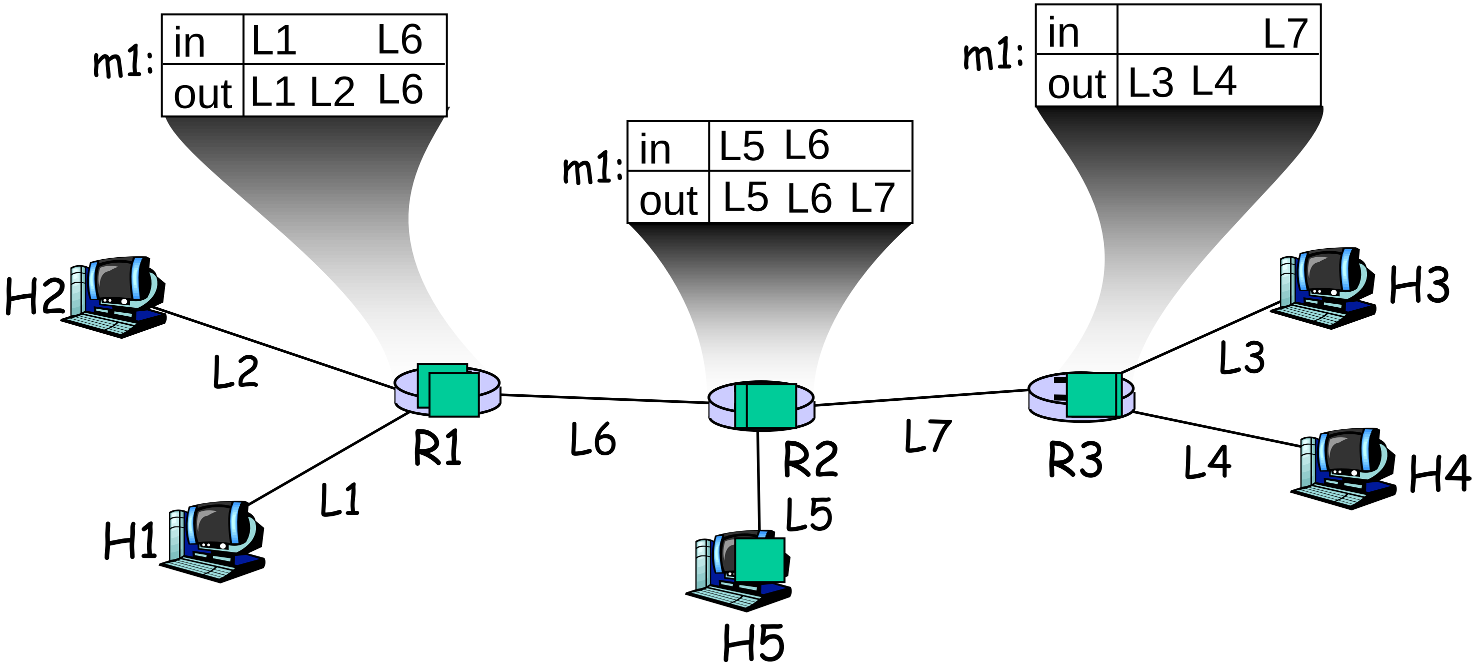
| | | |
|---|---|---|
| in | L6 | |
| out | L5 | L7 |

m1:

| | | |
|---|---|---|
| in | L7 | |
| out | L3 | L4 |

# RSVP: building up path state

- next, H5 sends path message, creating more state in routers



m1:

| | |
|---|---|
| in | L1 L6 |
| out | L1 L2 L6 |

m1:

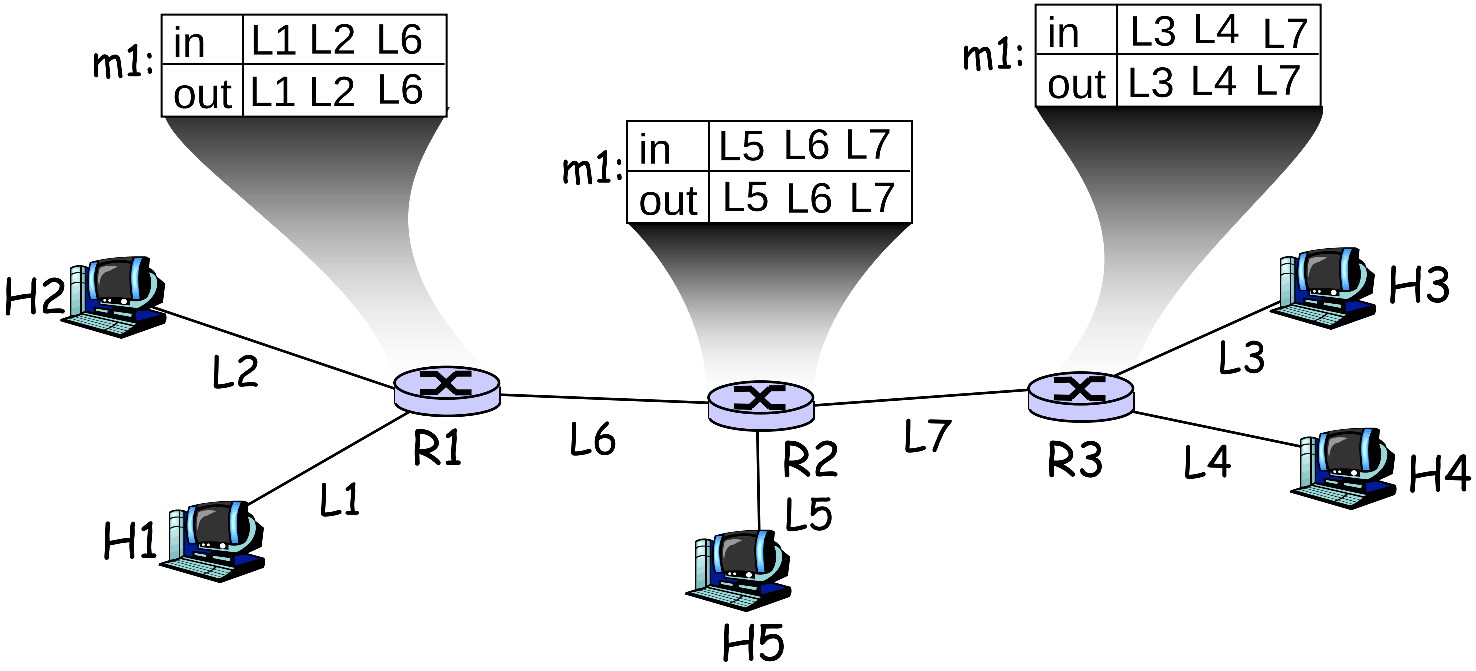
| | |
|---|---|
| in | L5 L6 |
| out | L5 L6 L7 |

m1:

| | |
|---|---|
| in | L7 |
| out | L3 L4 |

# RSVP: building up path state

- H2, H3, H5 send path msgs, completing path state tables



m1:

| in | L1 L2 L6 |
|---|---|
| out | L1 L2 L6 |

m1:

| in | L5 L6 L7 |
|---|---|
| out | L5 L6 L7 |

m1:

| in | L3 L4 L7 |
|---|---|
| out | L3 L4 L7 |

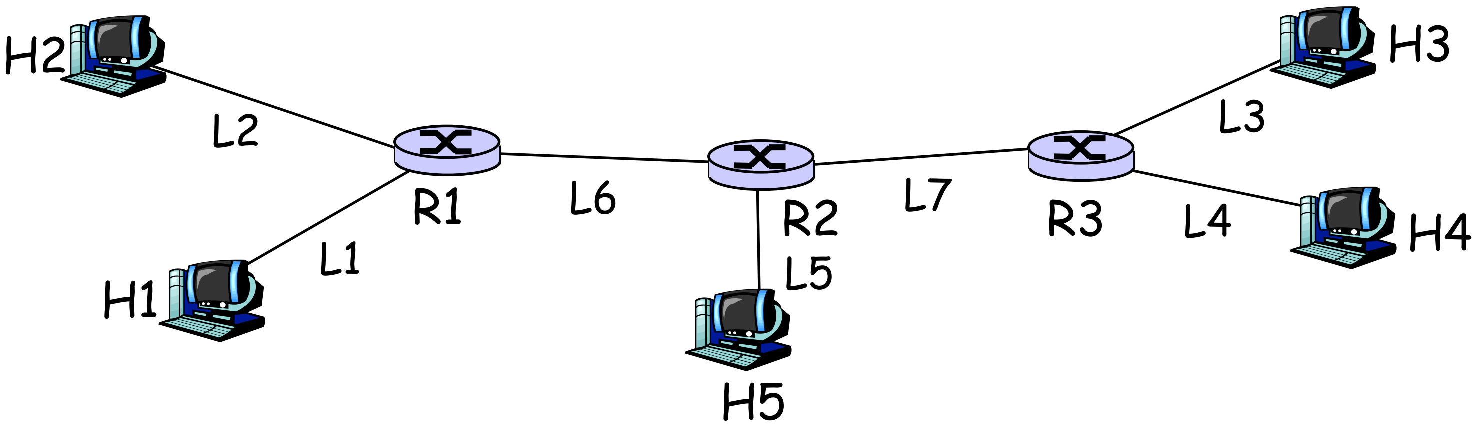
# reservation msgs: *receiver-to-network* signaling

- reservation message contents:
  - *desired bandwidth:*
  - *filter type:*
    - no filter: any packets address to multicast group can use reservation
    - fixed filter: only packets from specific set of senders can use reservation
    - dynamic filter: senders who's p[ackets can be forwarded across link will change (by receiver choce) over time.
  - *filter spec*
- reservations flow upstream from receiver-to-senders, reserving resources, creating additional, *receiver-related* state at routers

# RSVP: *receiver* reservation example 1

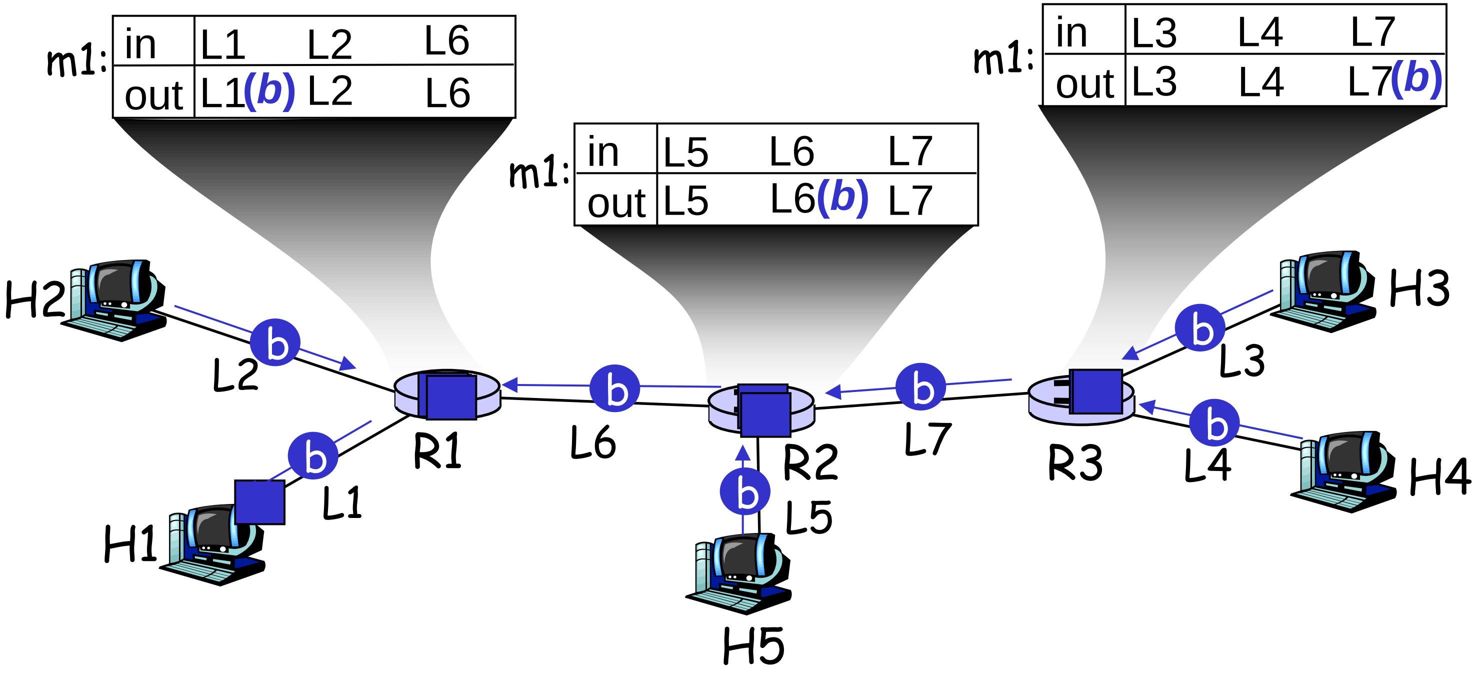
H1 wants to receive audio from all other senders

- H1 reservation msg flows uptree to sources
- H1 only reserves enough bandwidth for 1 audio stream
- reservation is of type "no filter" – any sender can use reserved bandwidth



# RSVP: *receiver* reservation example 1

- H1 reservation msgs flows uptree to sources
- routers, hosts reserve bandwidth b needed on downstream links towards H1



m1:

| in | L1 | L2 | L6 |
|---|---|---|---|
| out | L1(***b***) | L2 | L6 |

m1:

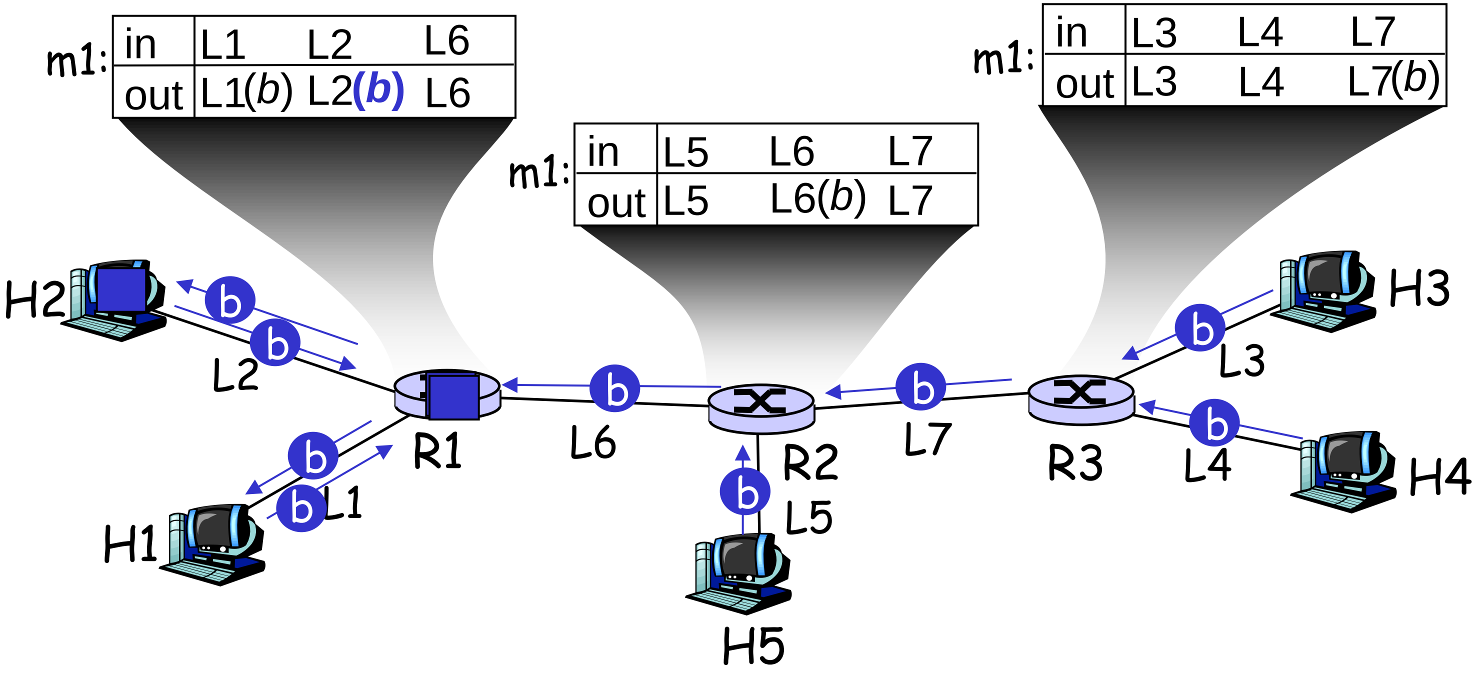
| in | L5 | L6 | L7 |
|---|---|---|---|
| out | L5 | L6(***b***) | L7 |

m1:

| in | L3 | L4 | L7 |
|---|---|---|---|
| out | L3 | L4 | L7(***b***) |

# RSVP: *receiver* reservation example 1 (more)

- next, H2 makes no-filter reservation for bandwidth *b*
- H2 forwards to R1, R1 forwards to H1 and R2 (?)
- R2 takes no action, since *b* already reserved on L6



m1:

| | | | |
|---|---|---|---|
| in | L1 | L2 | L6 |
| out | L1(*b*) | L2(***b***) | L6 |

m1:

| | | | |
|---|---|---|---|
| in | L5 | L6 | L7 |
| out | L5 | L6(*b*) | L7 |

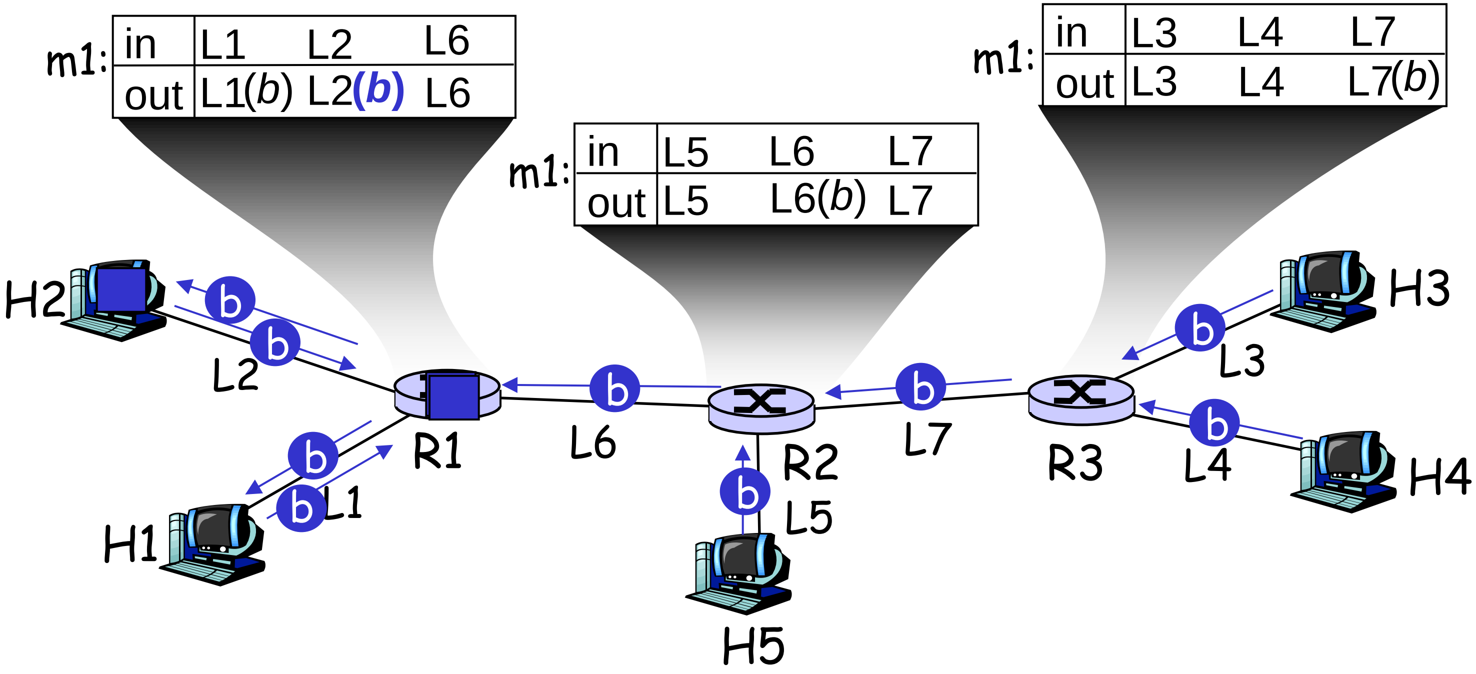
m1:

| | | | |
|---|---|---|---|
| in | L3 | L4 | L7 |
| out | L3 | L4 | L7(*b*) |

# RSVP: *receiver* reservation: issues

What if multiple senders (e.g., H3, H4, H5) over link (e.g., L6)?

- arbitrary interleaving of packets
- L6 flow policed by leaky bucket: if H3+H4+H5 sending rate exceeds b, packet loss will occur



m1:

| | | | |
|---|---|---|---|
| in | L1 | L2 | L6 |
| out | L1(*b*) | L2(***b***) | L6 |

m1:

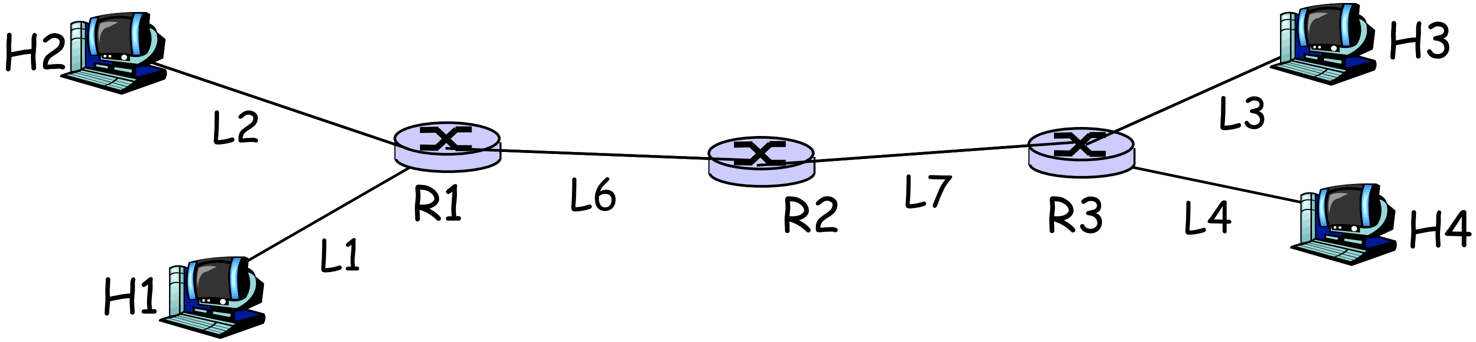
| | | | |
|---|---|---|---|
| in | L5 | L6 | L7 |
| out | L5 | L6(*b*) | L7 |

m1:

| | | | |
|---|---|---|---|
| in | L3 | L4 | L7 |
| out | L3 | L4 | L7(*b*) |

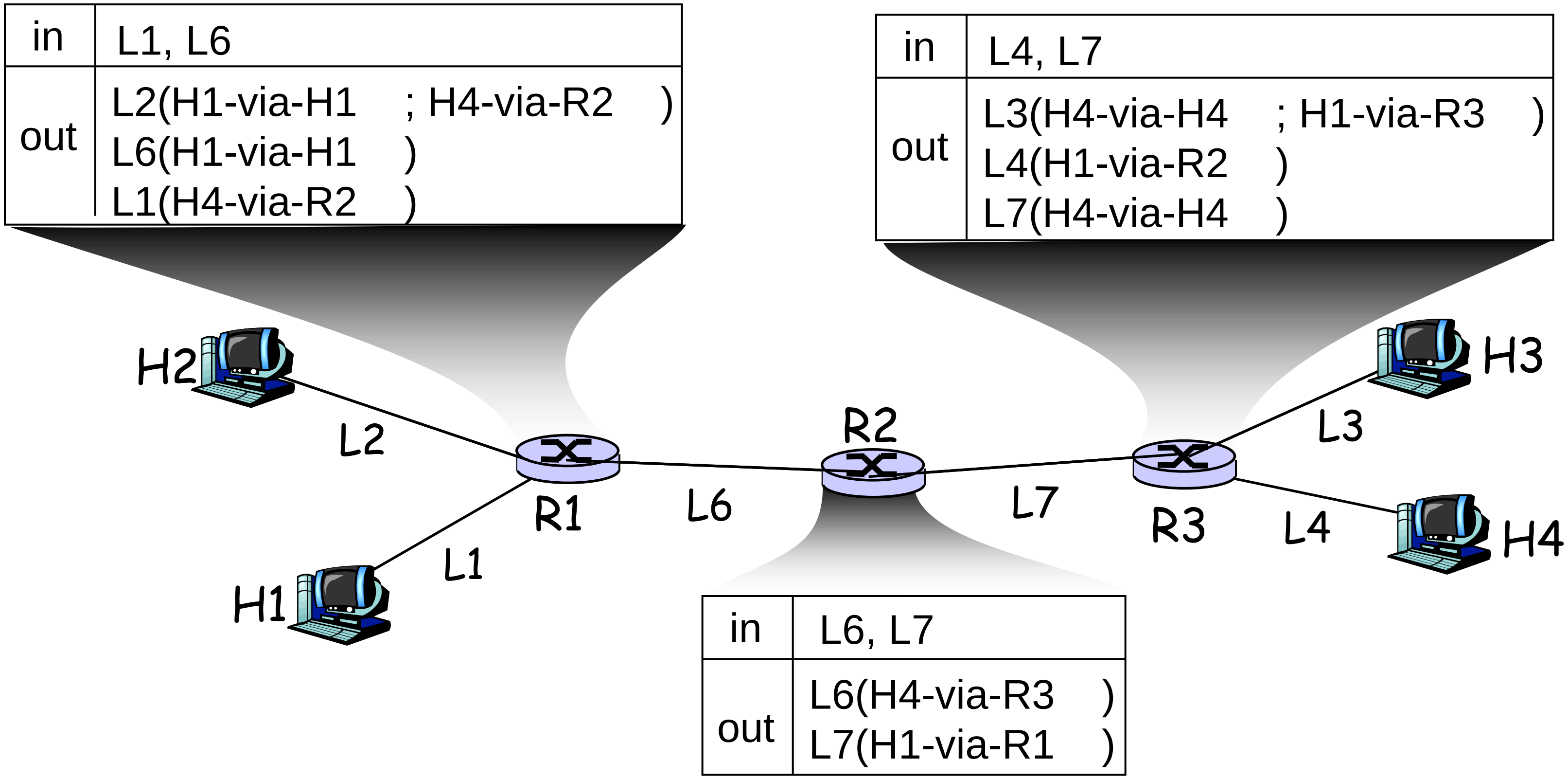
# RSVP: example 2

- H1, H4 are only senders
  - send *path messages* as before, indicating filtered reservation
  - Routers store upstream senders for each upstream link
- H2 will want to receive from H4 (only)



# RSVP: example 2

- H1, H4 are only senders
  - send *path messages* as before, indicating filtered reservation



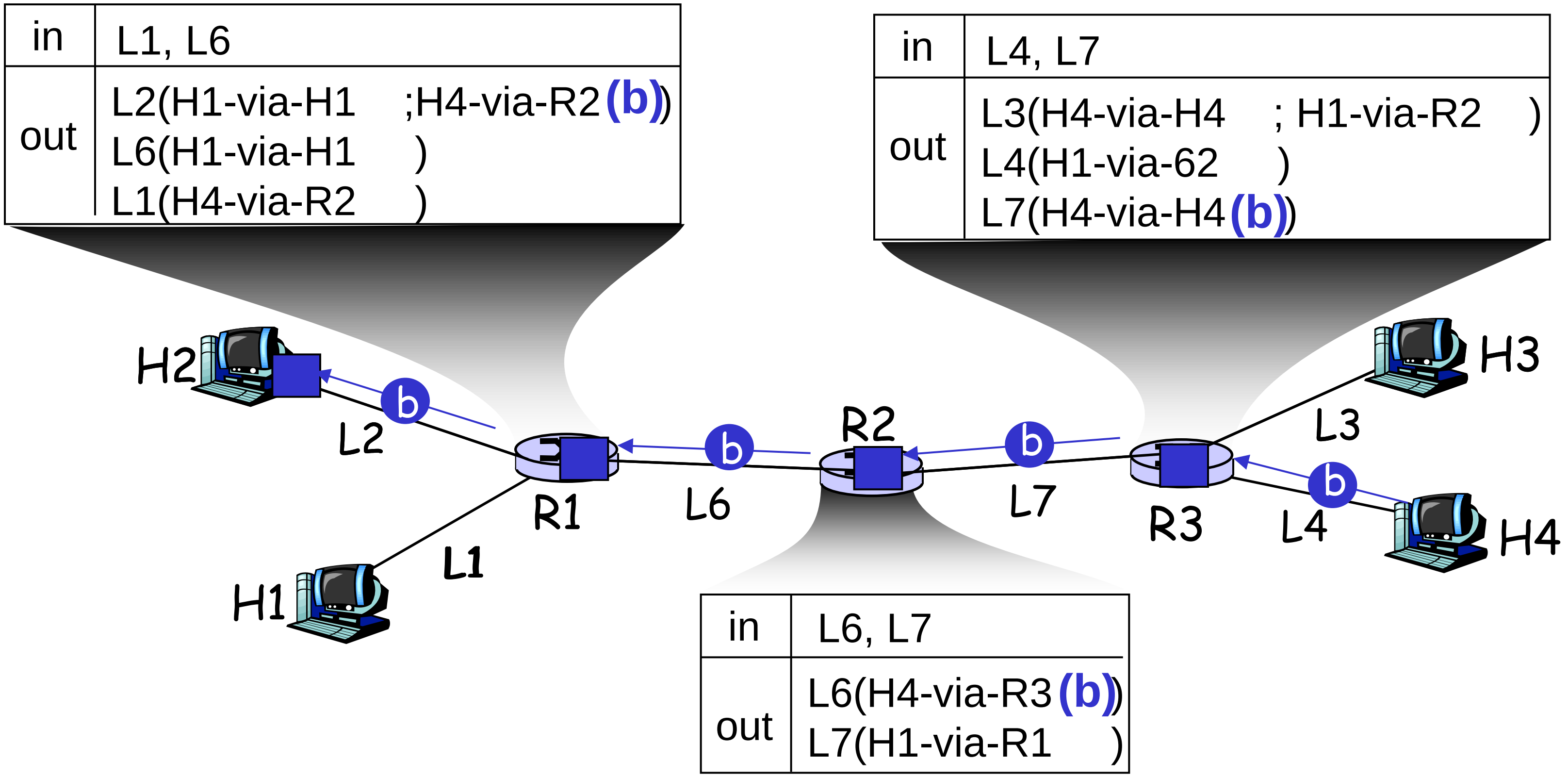
| | |
|---|---|
| in | L1, L6 |
| out | L2(H1-via-H1 ; H4-via-R2 )<br>L6(H1-via-H1 )<br>L1(H4-via-R2 ) |

| | |
|---|---|
| in | L4, L7 |
| out | L3(H4-via-H4 ; H1-via-R3 )<br>L4(H1-via-R2 )<br>L7(H4-via-H4 ) |

| | |
|---|---|
| in | L6, L7 |
| out | L6(H4-via-R3 )<br>L7(H1-via-R1 ) |

# RSVP: example 2

- receiver H2 sends reservation message for source H4 at bandwidth *b*
  - propagated upstream towards H4, reserving *b*



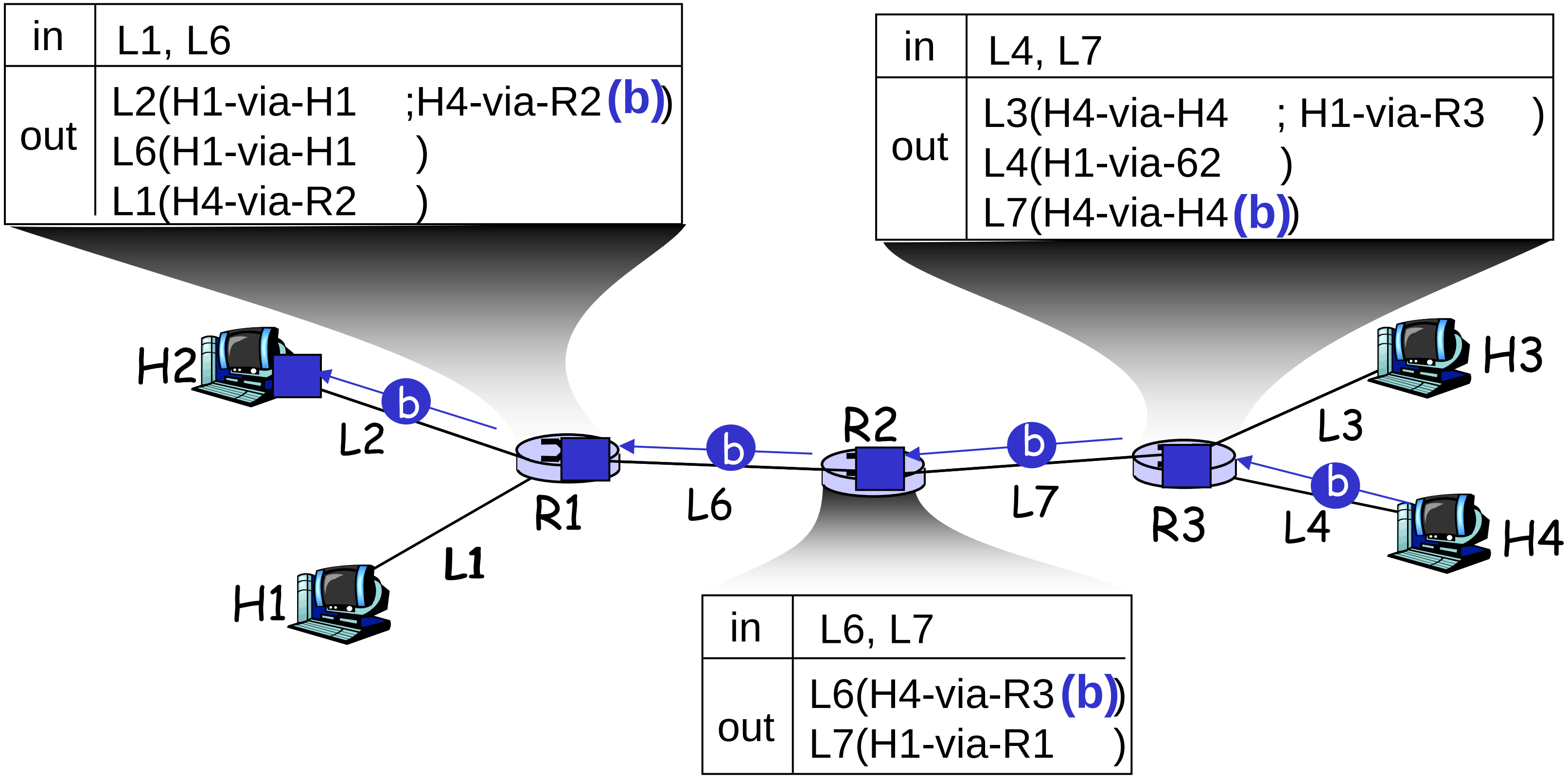
| in | L1, L6 |
|---|---|
| out | L2(H1-via-H1 ;H4-via-R2**(b)**)<br>L6(H1-via-H1 )<br>L1(H4-via-R2 ) |

| in | L4, L7 |
|---|---|
| out | L3(H4-via-H4 ; H1-via-R2 )<br>L4(H1-via-62 )<br>L7(H4-via-H4**(b)**) |

| in | L6, L7 |
|---|---|
| out | L6(H4-via-R3**(b)**)<br>L7(H1-via-R1 ) |

# RSVP: *soft-state*

- senders periodically resend path msgs to refresh (maintain) state
- receivers periodically resend resv msgs to refresh (maintain) state
- path and resv msgs have TTL field, specifying refresh interval



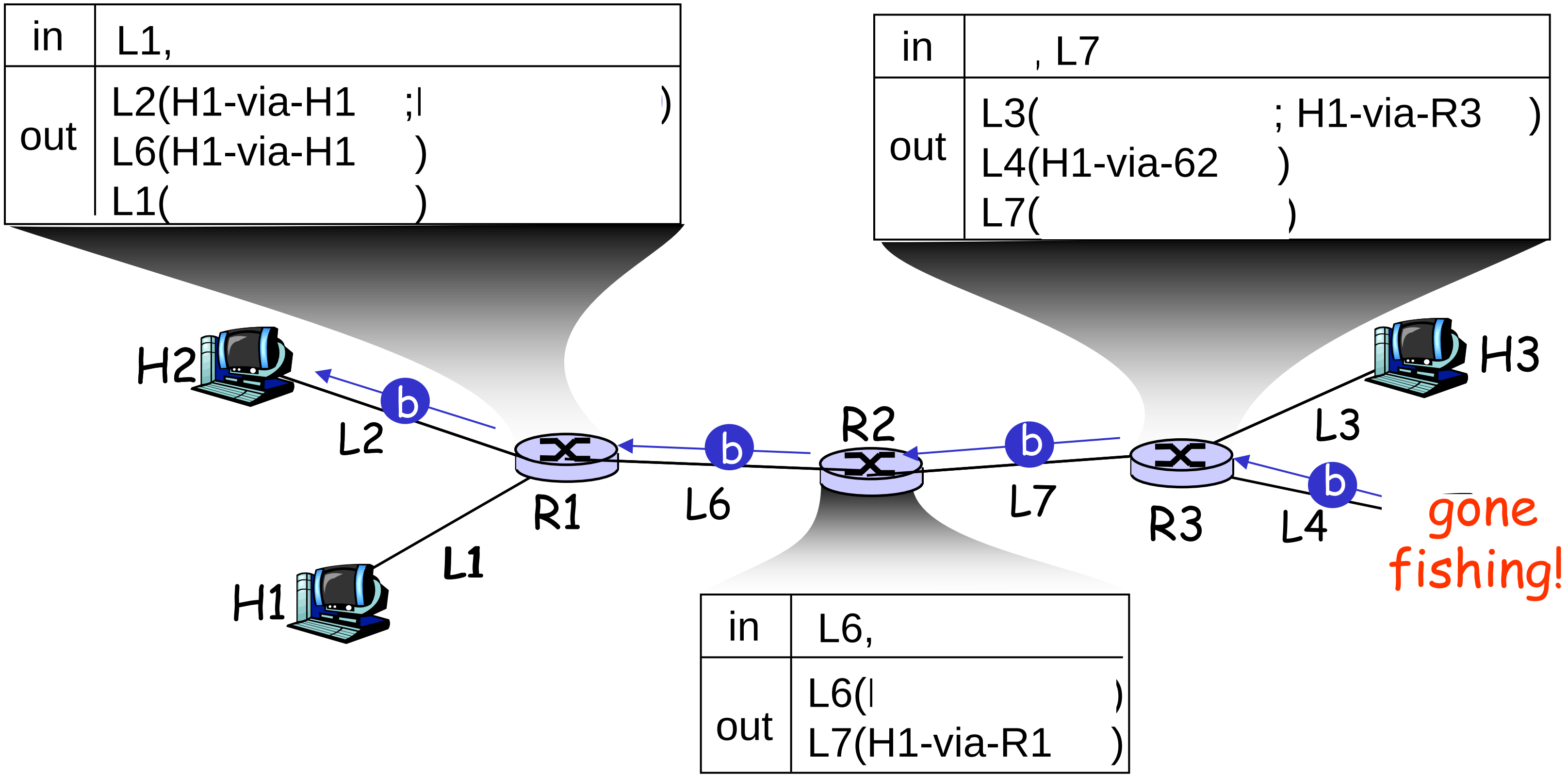
| | |
|---|---|
| in | L1, L6 |
| out | L2(H1-via-H1 ;H4-via-R2 (b))<br>L6(H1-via-H1 )<br>L1(H4-via-R2 ) |

| | |
|---|---|
| in | L4, L7 |
| out | L3(H4-via-H4 ; H1-via-R3 )<br>L4(H1-via-62 )<br>L7(H4-via-H4 (b)) |

| | |
|---|---|
| in | L6, L7 |
| out | L6(H4-via-R3 (b))<br>L7(H1-via-R1 ) |

# RSVP: *soft-state*

- suppose H4 (sender) leaves without performing teardown
- eventually state in routers will timeout and disappear!



| | |
|---|---|
| in | L1, |
| out | L2(H1-via-H1 ;l )<br>L6(H1-via-H1 )<br>L1( ) |

| | |
|---|---|
| in | , L7 |
| out | L3( ; H1-via-R3 )<br>L4(H1-via-62 )<br>L7( ) |

| | |
|---|---|
| in | L6, |
| out | L6(l )<br>L7(H1-via-R1 ) |

# The many uses of reservation/path refresh

- recover from an earlier lost refresh message
  - expected time until refresh received must be longer than timeout interval! (short timer interval desired)
- Handle receiver/sender that goes away without teardown
  - Sender/receiver state will timeout and disappear
- Reservation refreshes will cause new reservations to be made to a receiver from a sender who has joined since receivers last reservation refresh
  - E.g., in previous example, H1 is only receiver, H3 only sender. Path/reservation messages complete, data flows
  - H4 joins as sender, nothing happens until H3 refreshes reservation, causing R3 to forward reservation to H4, which allocates bandwidth

# RSVP: reflections

- multicast as a “first class” service
- receiver-oriented reservations
- use of soft-state

# Multimedia Networking: Summary

- multimedia applications and requirements
- making the best of today's best effort service
- scheduling and policing mechanisms
- next generation Internet: Intserv, RSVP, Diffserv