U.PORTO

U.PORTO FCUP FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

[dcc]

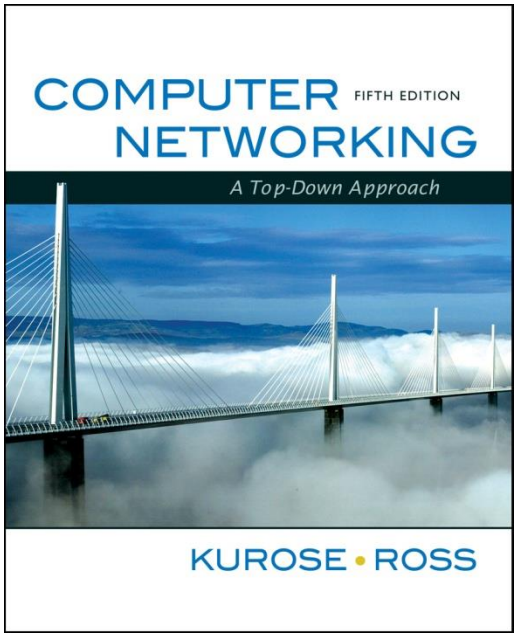
Multimedia Networking

Tópicos Avançados em Redes

2023/24

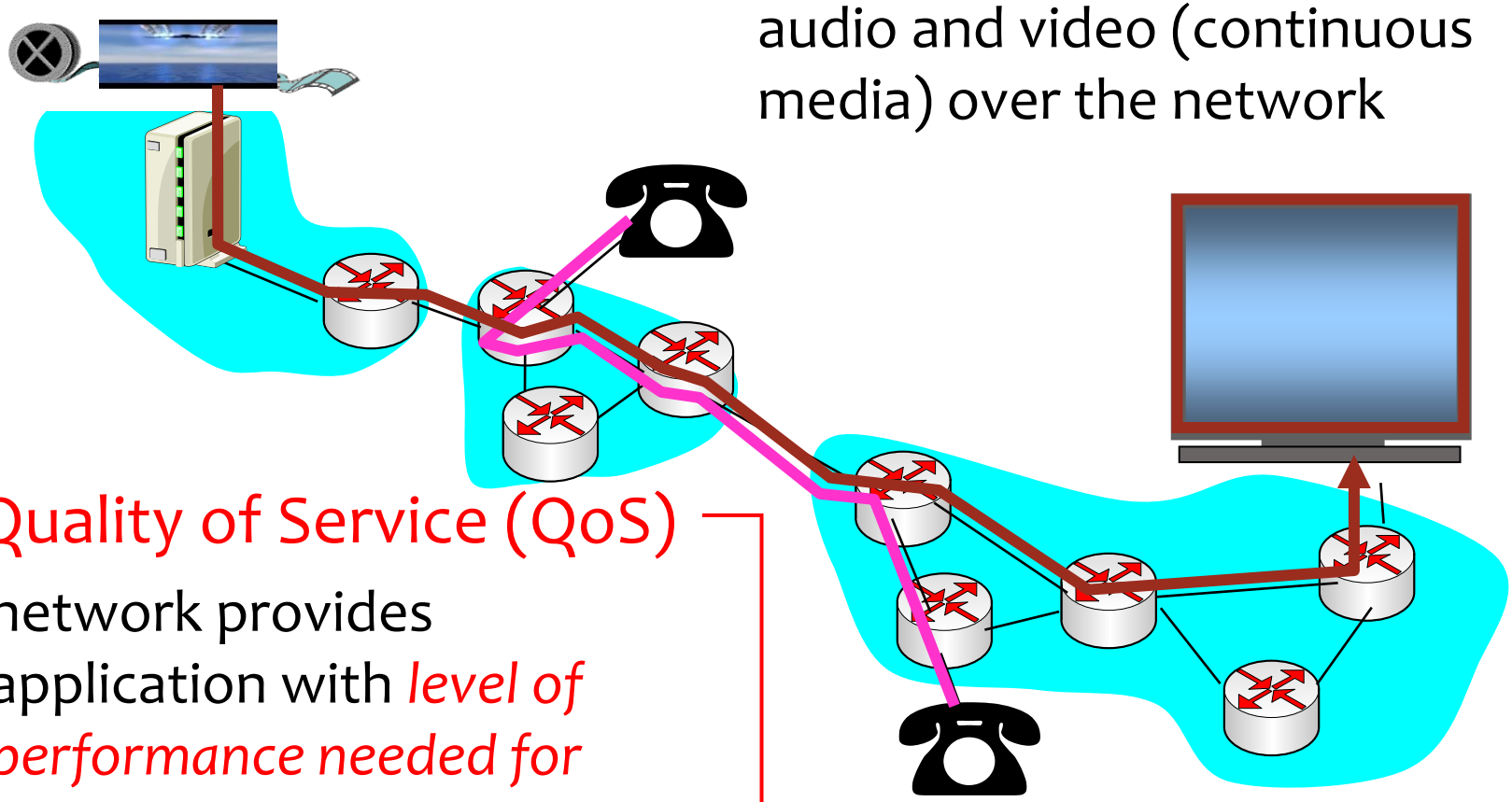
Multimedia Networking

These slides are mostly based on materials provided with the book *Computer Networking – A Top-Down Approach*, 5th ed., by James Kurose & Keith Ross, with updates from Pedro Brandão and Rui Prior



Multimedia and Quality of Service: What is it?

multimedia applications: audio and video (continuous media) over the network



Quality of Service (QoS)

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

Multimedia Application needs

- Multimedia application have requirements on
  - Delay
  - Jitter (delay variation)
  - Bandwidth

- Q: How can we use them in a best-effort Internet?

Goals

Principles

- classify multimedia applications
- identify the network services applications need
- making the best of best effort service

Protocols and Architectures

- specific protocols for best-effort
- mechanisms for providing QoS
- architectures for QoS

Outline

7.1 multimedia networking applications

7.2 streaming stored audio and video

7.3 making the best out of best effort service

7.4 protocols for real-time interactive applications

RTP,RTCP,SIP

7.5 providing multiple classes of service

7.6 providing QoS guarantees

MM Networking Applications

Classes of MM applications:

1) stored streaming

2) live streaming

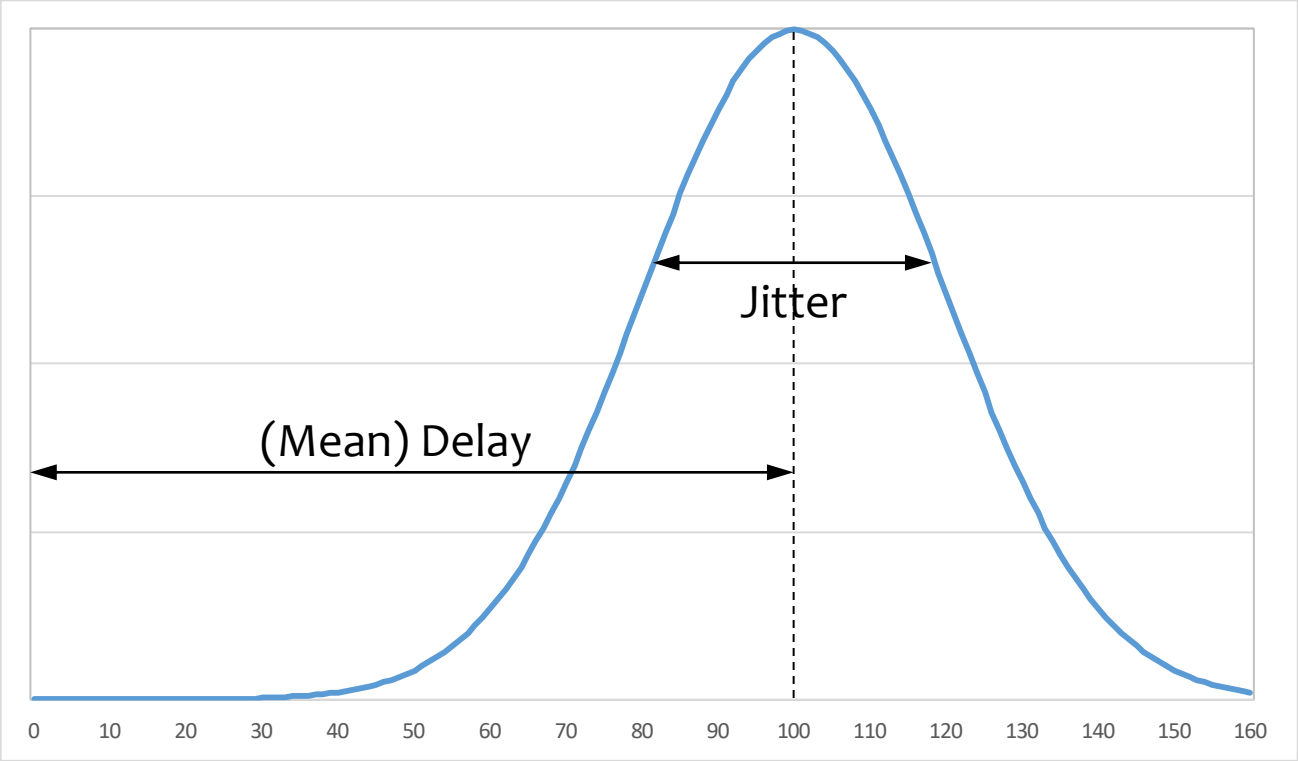
3) interactive, real-time

Fundamental characteristics:

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

What is jitter?

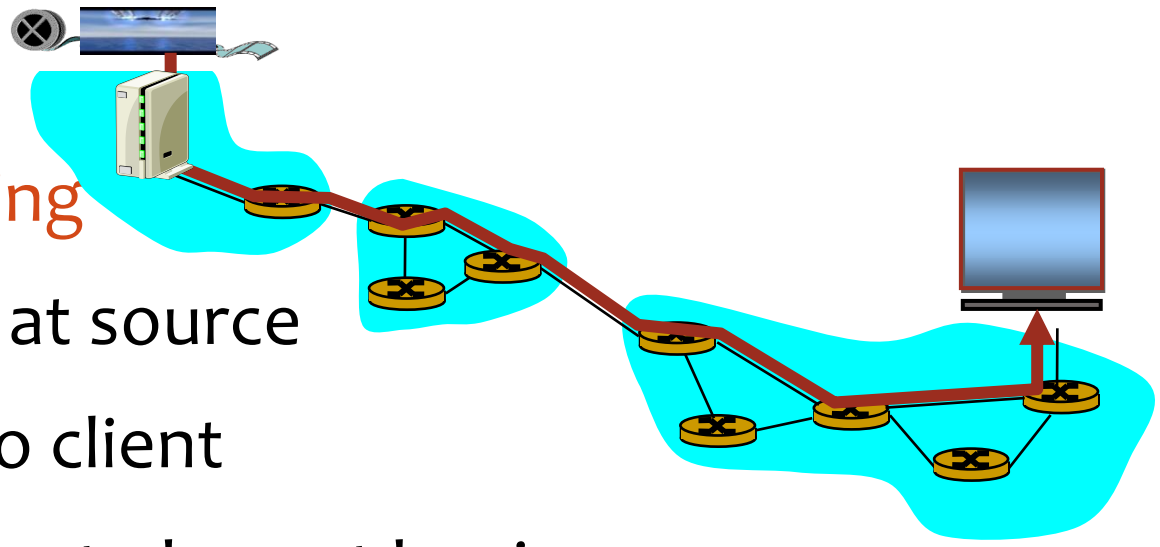
Jitter is the variability of packet delays within the same packet stream



What are the QoS Parameters?

- Delay (latency)
  - Transmission
  - Propagation
  - Queues
  - Node processing
  - Application processing
- Jitter
  - Medium access
  - Traffic aggregation
  - Congestion
  - Route changes
- Rate
  - Path
  - Congestion
  - Temporal Variations
    - Fast
    - Slow
- Reliability
  - Losses
  - Out of order delivery
  - Duplicates

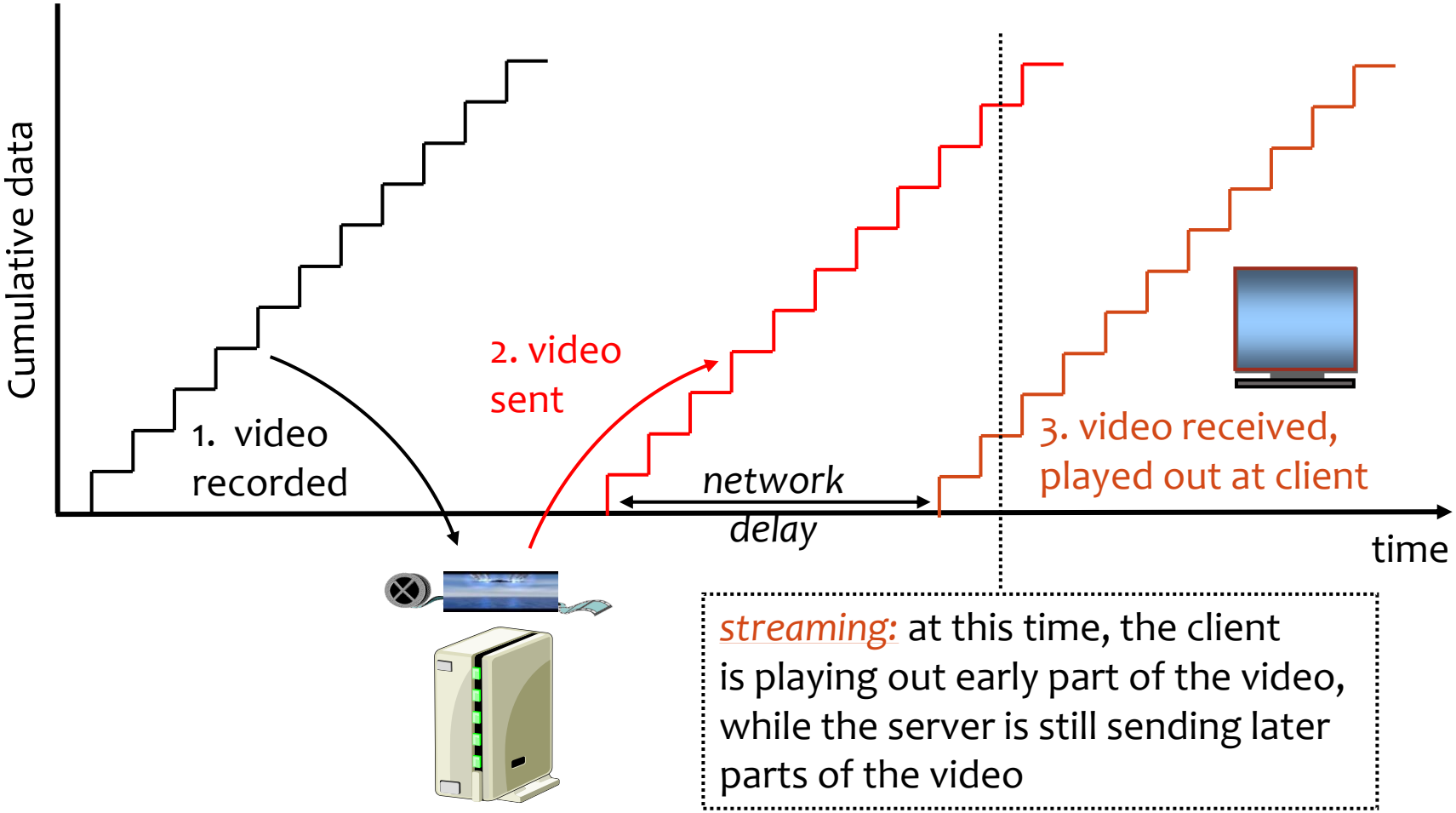
Streaming Stored Multimedia



Stored streaming

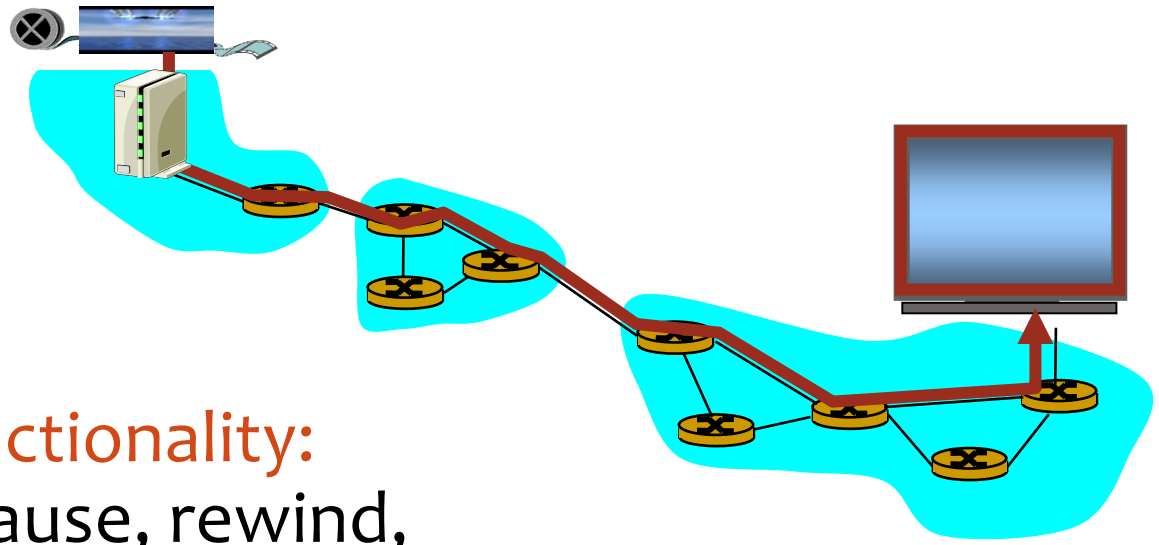
- media stored at source
- transmitted to client
- streaming: client play out begins before all data has arrived
  - instead of transferring the entire file first
- implies timing constraints for still-to-be transmitted data: in time for play out

Streaming Stored Multimedia: What is it?



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

Streaming Stored Multimedia: Interactivity



- VCR-like functionality:
  client can pause, rewind,
  fast forward, 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 play out

Streaming *Live* Multimedia

Examples:

- Internet radio talk show
- Live sports event

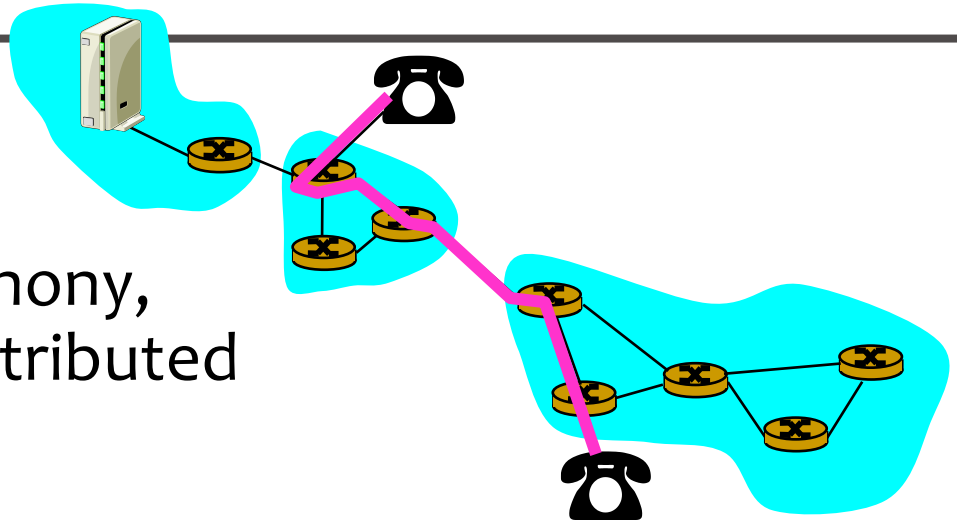
Streaming (as with streaming *stored* multimedia)

- Playback buffer
- Playback can lag tens of seconds after transmission…
- Still have timing constraint

Interactivity

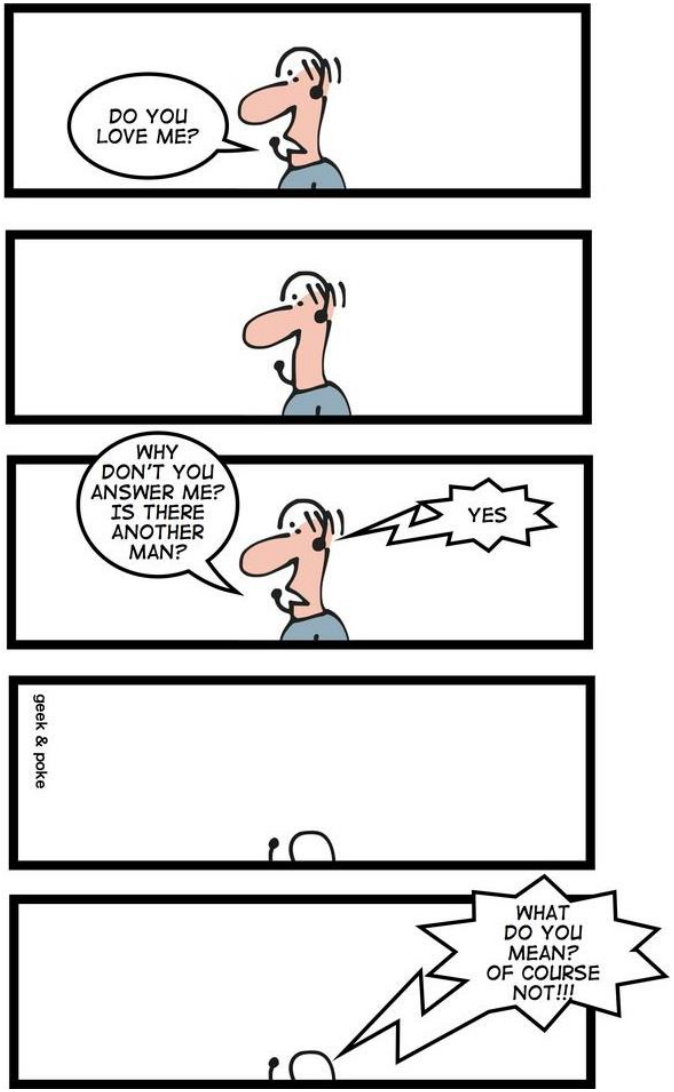
- Fast forward obviously impossible
- Rewind and pause possible!

Real-Time Interactive Multimedia



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

Simply Explained: Latency



Source: https://geek-and-poke.com/

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 as 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

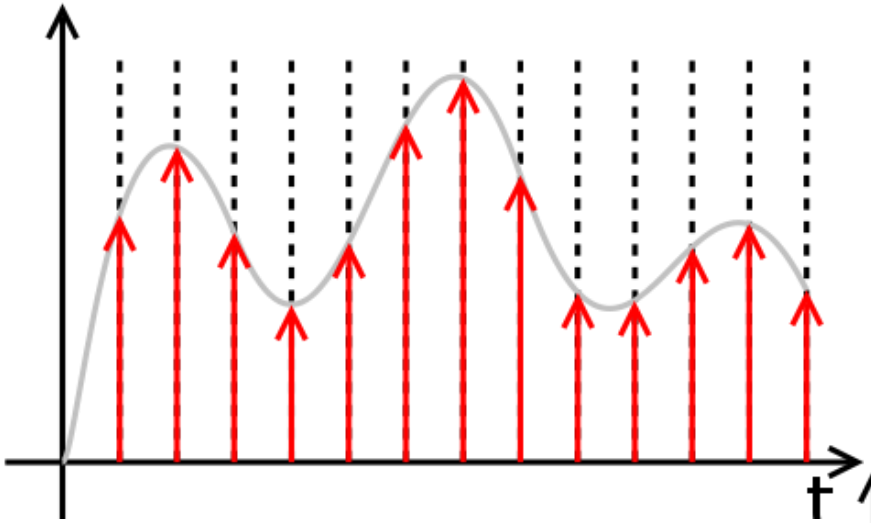
- no major changes
- more bandwidth when needed
- content distribution, application-layer multicast
- application layer techniques

Differentiated services philosophy:

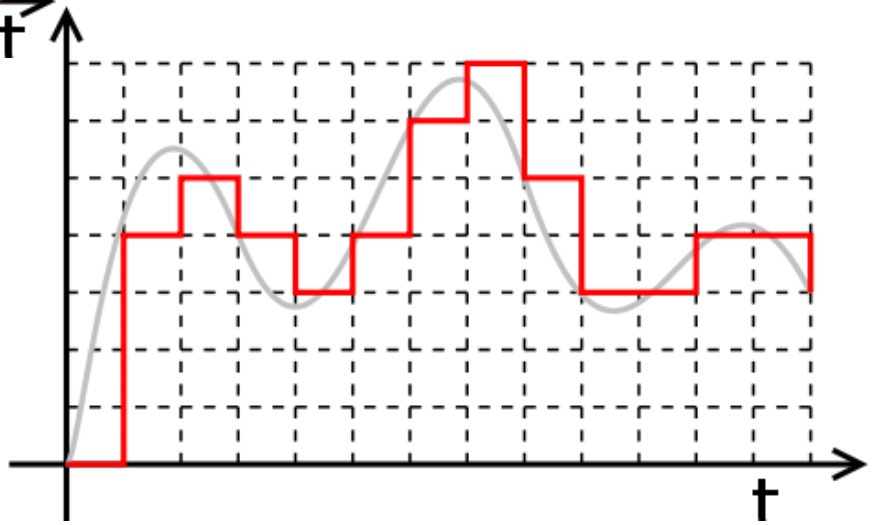
- fewer changes to Internet infrastructure, yet provide 1st and 2nd class service

What's your opinion?

Converting analog signal to digital



Sampling
(discretization of time)



Quantization (discretization of values)

Images from Wikipedia

A few words about audio compression

- analogue signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized
  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 analogue signal
  - possibly with some quality loss

Example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160, … 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 the next)

Examples:

- MPEG1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used on the Internet) < 1 Mbps

Research:

- Layered (scalable) video
  - adapt layers to available bandwidth
  - Annex G of H.264/MPEG-4 AVC
- Layered depth video (LDV)

Chapter 7 outline

7.1 multimedia networking applications

7.2 streaming stored audio and video

7.3 making the best out of best effort service

7.4 protocols for real-time interactive applications

RTP,RTCP,SIP

7.5 providing multiple classes of service

7.6 providing QoS guarantees

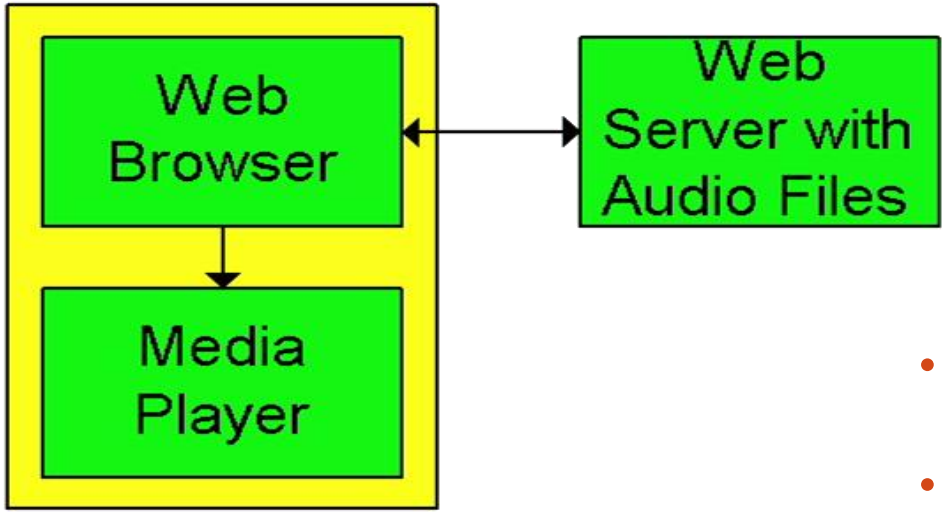
Streaming Stored Multimedia

- 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 with controls for interactivity

Internet multimedia: simplest approach

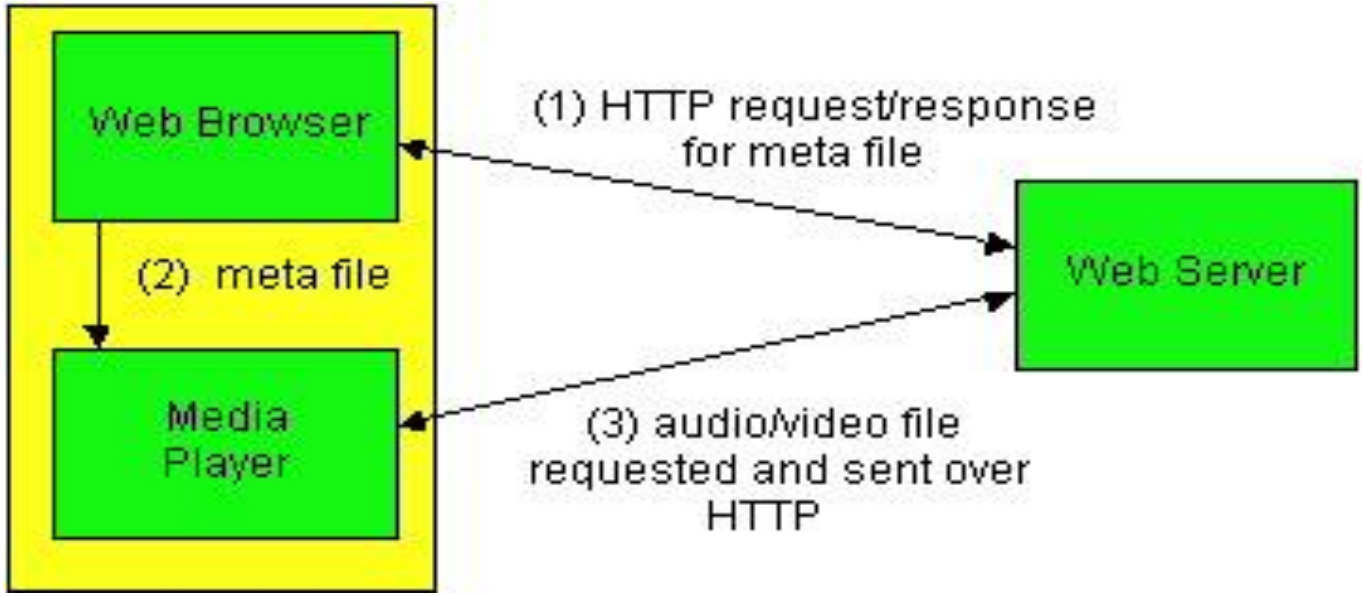


- audio or video stored in file
- files transferred as HTTP object
  - received in entirety at client
  - then passed to player

audio, video **not** streamed:

- no "pipelining" — long delays until play out!

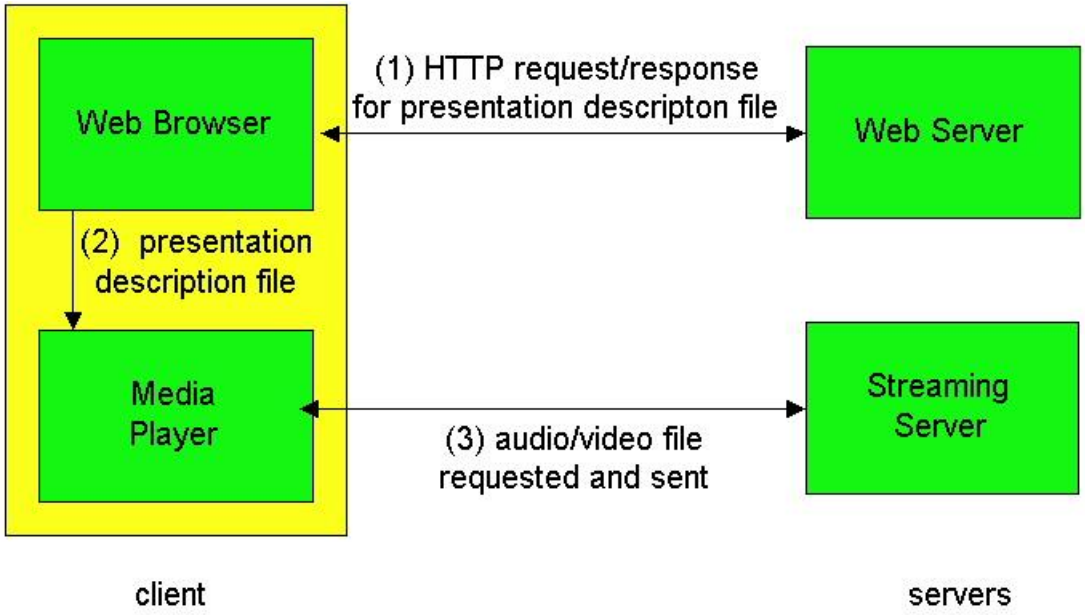
Internet multimedia: streaming approach



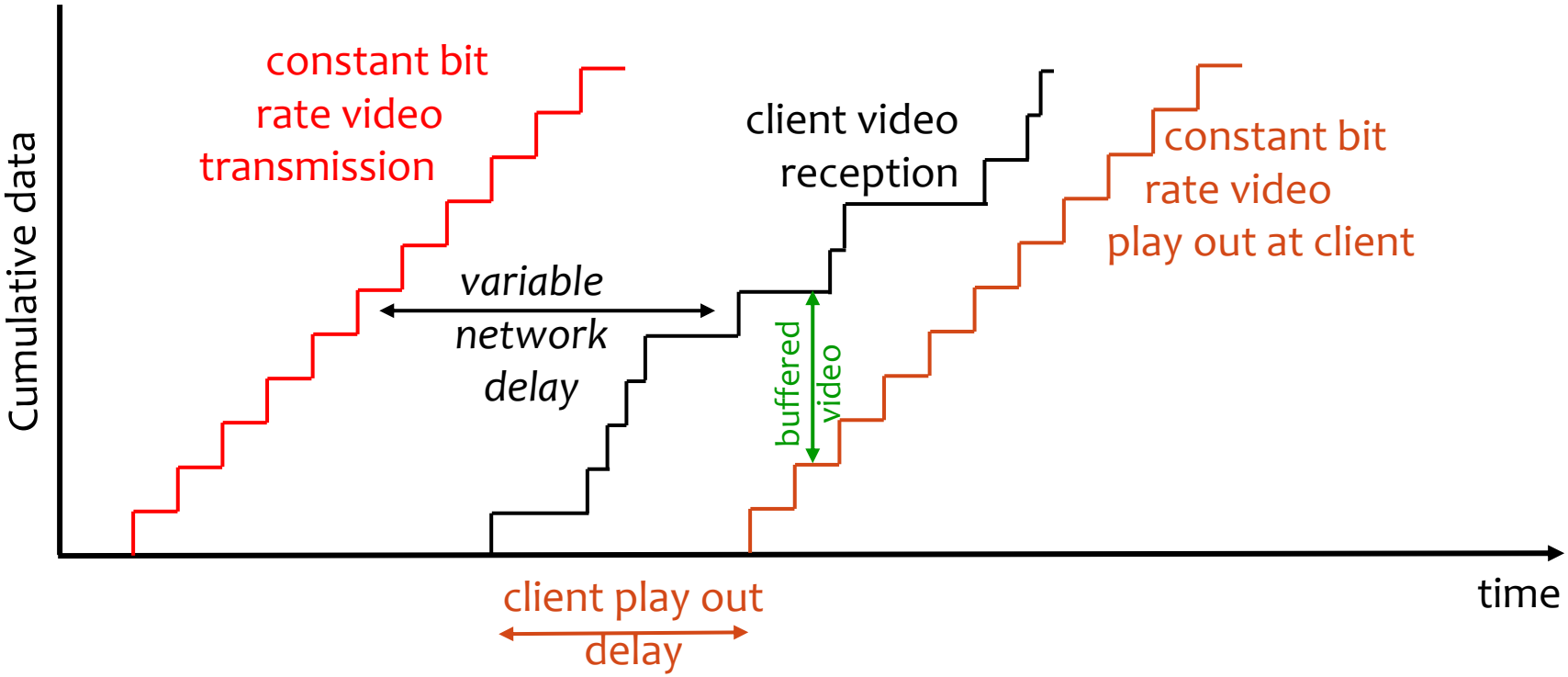
- browser GETs **metafile** referencing media file(s)
  - media can be split across multiple files
- browser launches player, passing metafile
- player contacts server
- server **streams** audio/video to player

Streaming from a streaming server

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

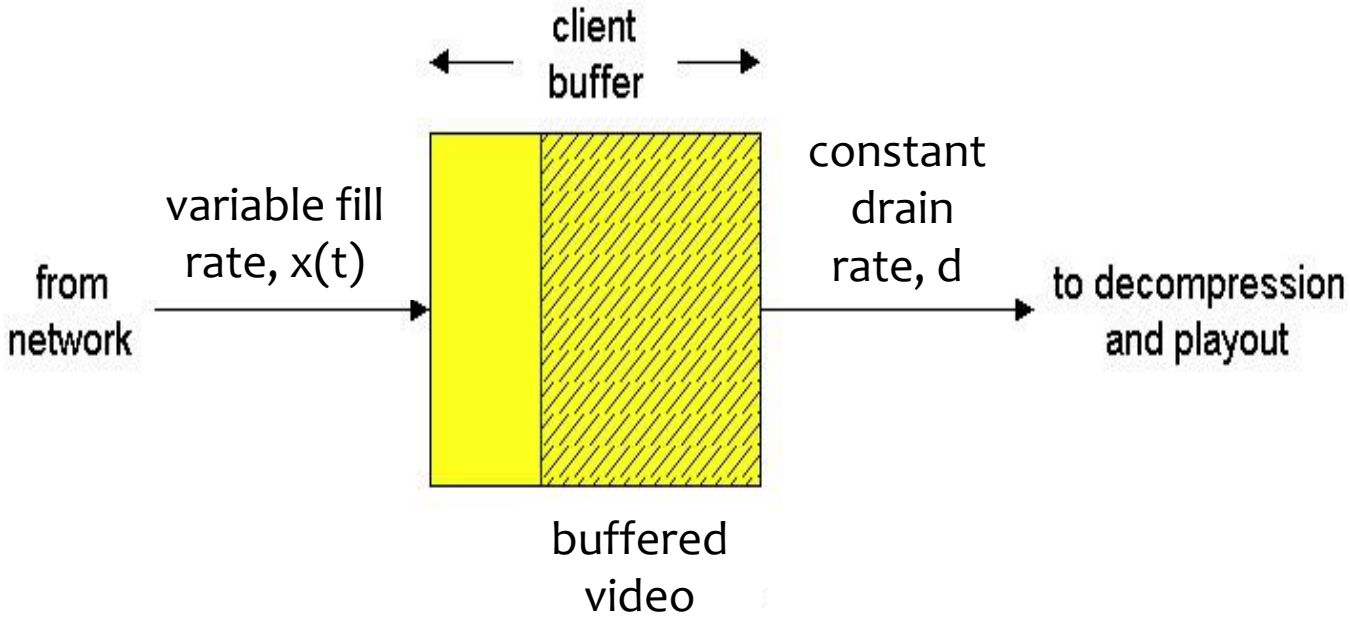


Streaming Multimedia: Client Buffering



- client-side buffering, play out delay compensate for network-added jitter

Streaming Multimedia: Client Buffering



- buffer size limits the amount of jitter we can absorb

Streaming Multimedia: UDP or TCP?

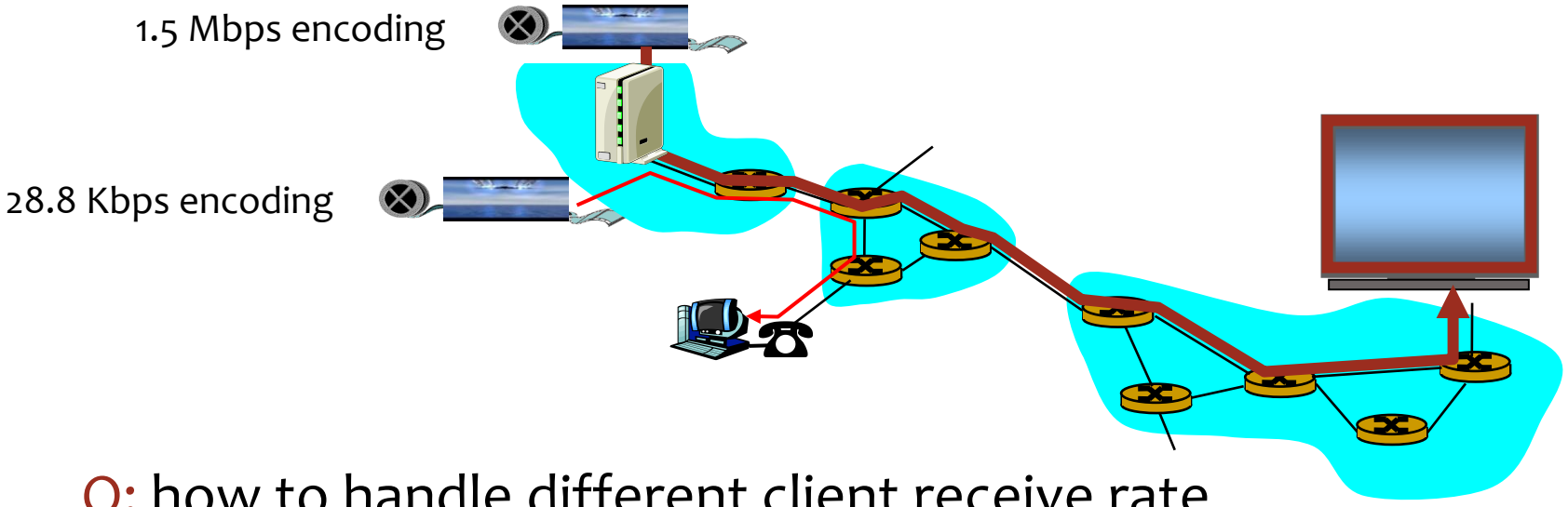
UDP

- 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 play out delay (2-5 seconds) to remove network jitter
- error recovery: time permitting

TCP

- send at maximum possible rate under TCP
  - subject to flow control
- fill rate fluctuates due to TCP congestion control
- larger play out 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?

- 100 kb/s 3G access
- 1 Gb/s 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
  - Though there are mechanisms such as HLS and DASH
- no commands for fast forward, etc.

RTSP: RFC 2326

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

What RTSP 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

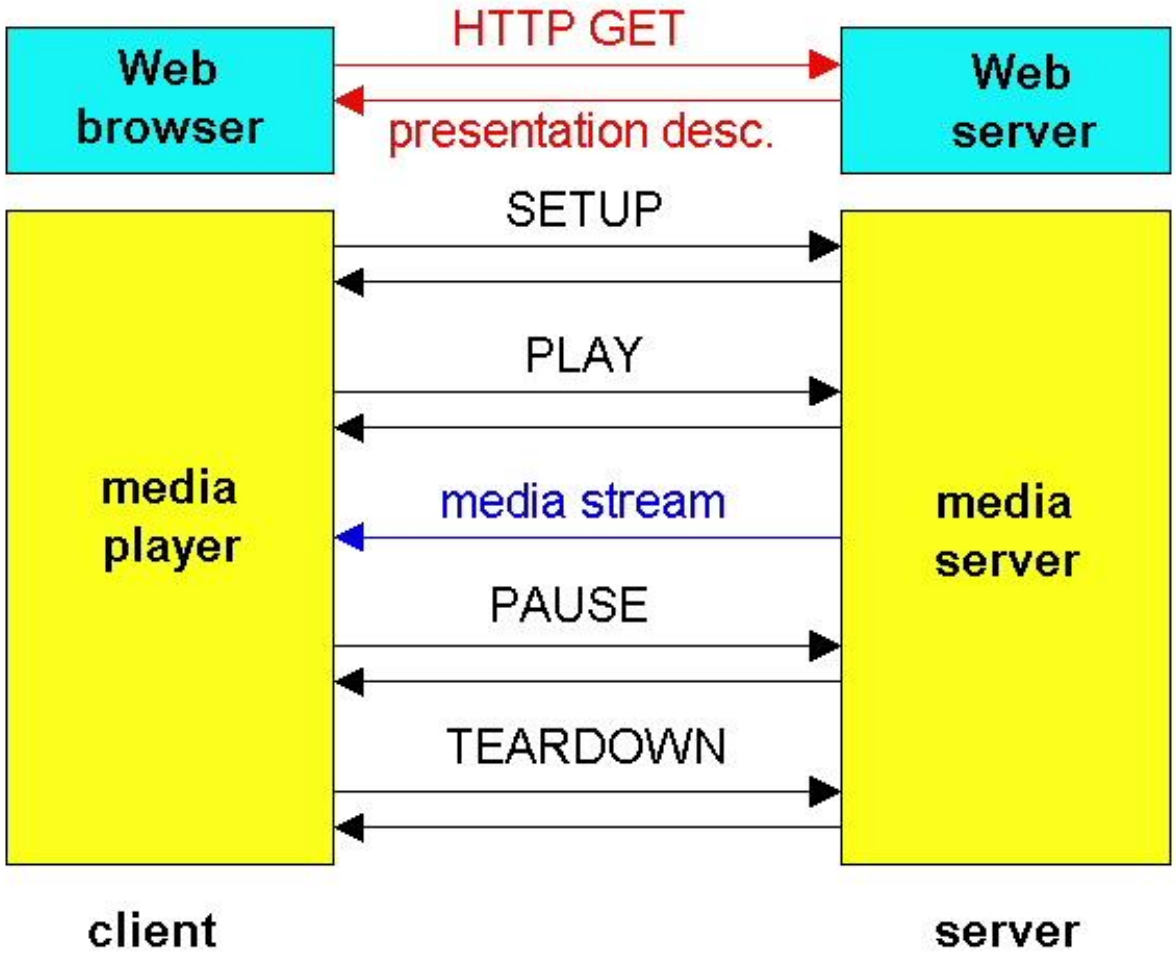
Scenario:

- 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 OK
Session: 4231
C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=0-
S: RTSP/1.0 200 OK
Session: 4231
C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=37
S: RTSP/1.0 200 OK
Session: 4231
C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
S: RTSP/1.0 200 OK
Session: 4231

Sequence numbers omitted for brevity

Chapter 7 outline

7.1 multimedia networking applications

7.2 streaming stored audio and video

7.3 making the best out of best effort service

7.4 protocols for real-time interactive applications

RTP,RTCP,SIP

7.5 providing multiple classes of service

7.6 providing QoS guarantees

Real-time interactive applications

- PC-2-PC phone
  - Skype
  - WebRTC
- PC-2-phone
  - Dialpad
  - Net2phone
  - Skype
- Videoconference with webcams
  - Zoom
  - Skype
  - Polycom
  - WebRTC

Interactive Multimedia: Internet Phone

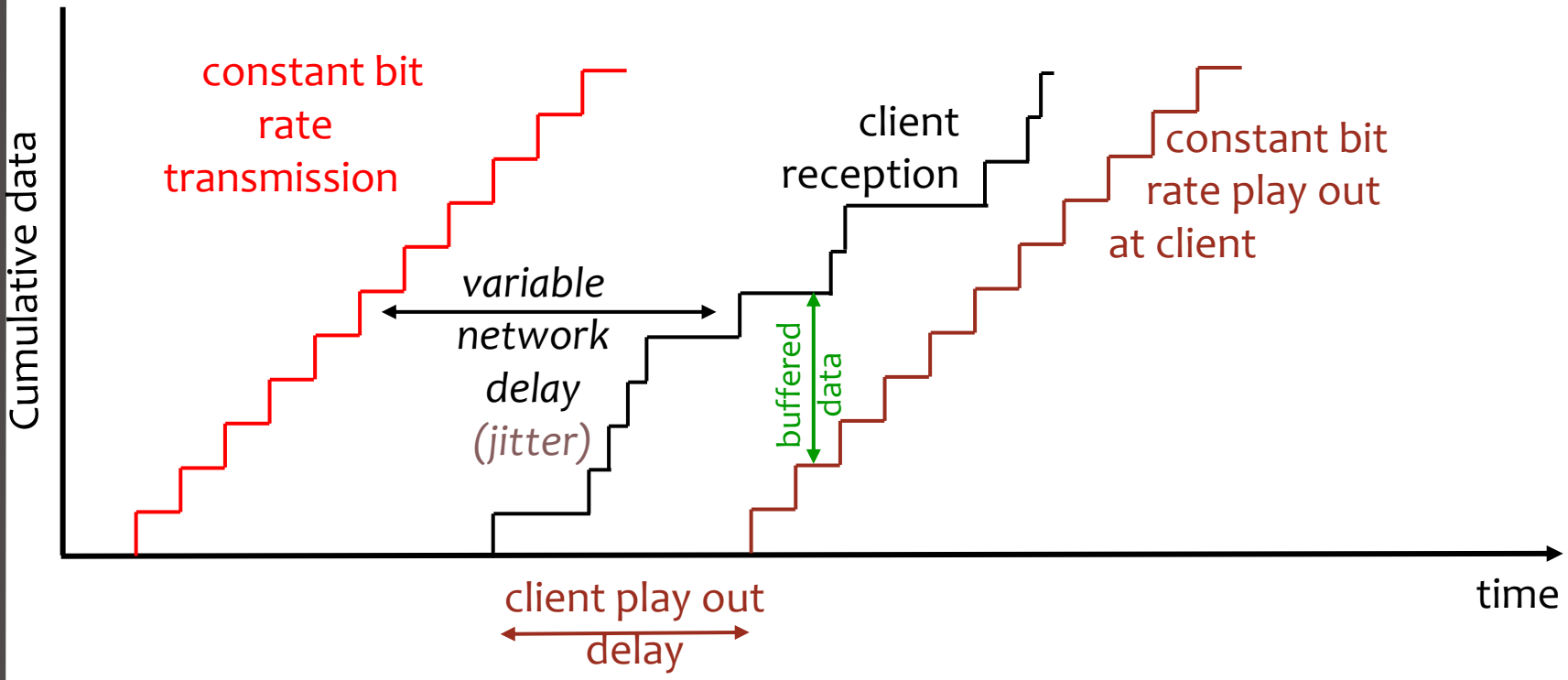
Internet Phone by way of an example

- speaker's audio: alternating talk spurts, silent periods
  - 64 kbps during talk spurt
  - packets 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 talk spurt

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 play out at receiver
  - delays: processing, queuing in network; end-system (sender, receiver) delays
  - typical maximum tolerable delay: 400 ms
- loss tolerance: depending on voice encoding and losses concealed, packet loss rates between 1% and 10% can be tolerated

Delay Jitter

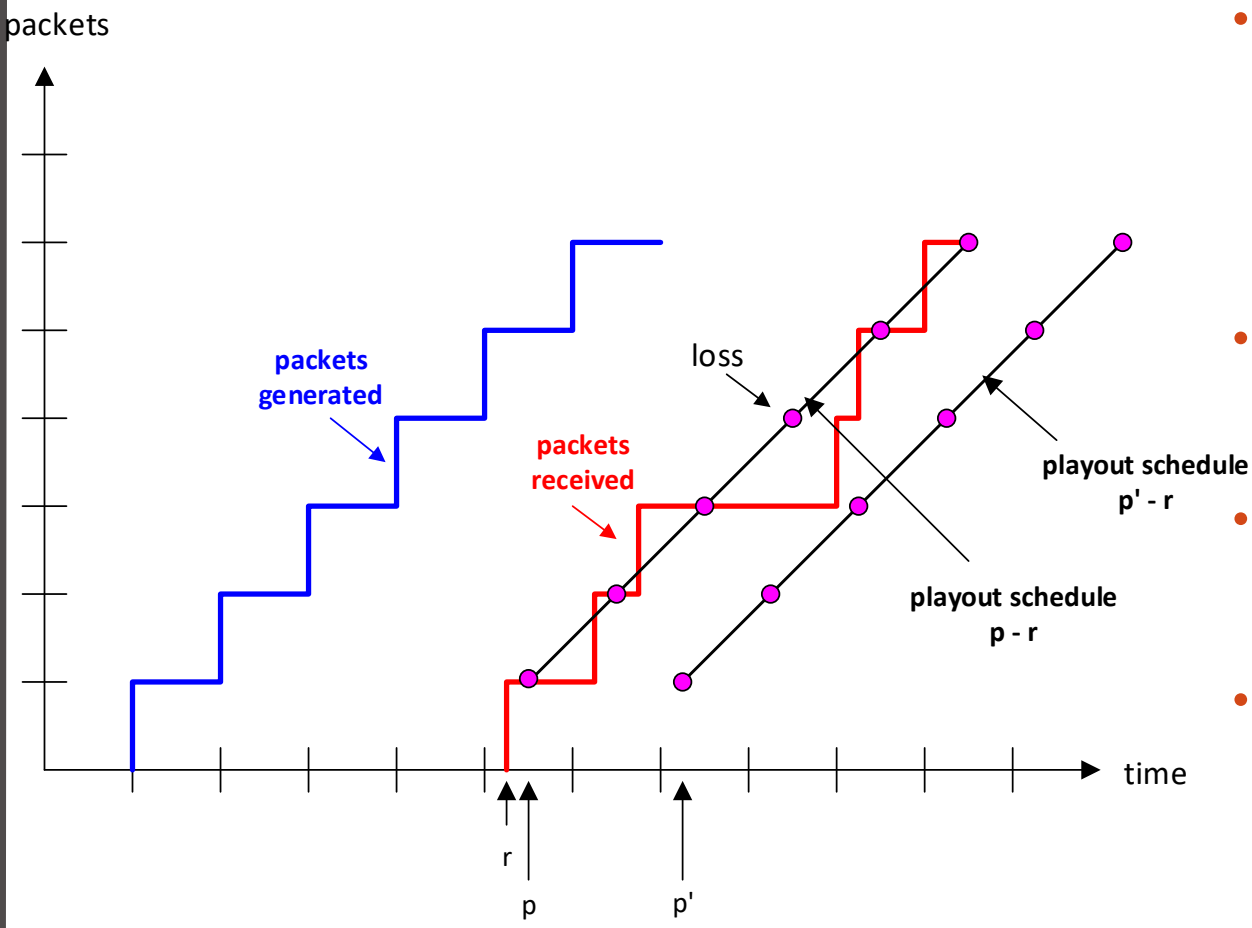


- end-to-end delivery of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)
- Play out must be exactly every 20 msec

Internet Phone: Fixed play out Delay

- receiver attempts to play out 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 play out — data "lost"
- trade-off in choosing q:
  - *large q:* fewer packet losses (delay losses)
  - *small q:* better interactive experience

Fixed play out Delay



- sender generates packets every 20 msec during talk spurt
- first packet received at time r
- play out schedule A: begins at p
- play out schedule B: begins at p'

Adaptive play out Delay (1)

- Goal: minimize play out delay, keeping delay losses low
- Approach: adaptive play out delay adjustment:
  - estimate network delay, adjust play out delay at the beginning of each talk spurt
  - silent periods may be compressed or elongated
  - chunks still played out every 20 msec during talk spurt

t_i = timestamp of the i^{th} packet
r_i = the time the i^{th} packet is received by the player
p_i = the time the i^{th} packet is played by the player
$r_i - t_i$ = network delay for the i^{th} packet
d_i = estimate of average network delay after receiving i^{th} 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 $0<u<1$ (e.g., $u = 0.01$) ➔ EWMA.

Adaptive play out 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, play out time is:

$$p_i = t_i + d_i + Kv_i$$

  - where K is a positive constant (e.g., K= 4)

- remaining packets in talk spurt are played out periodically (every 20 ms in this example)

Adaptive play out (3)

- Q: How does receiver determine whether packet is first in a talk spurt?
- if no loss, receiver looks at timestamps of successive packets
  - 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
- some apps mark explicitly the 1st pkt in a talk spurt

Recovery from packet loss

Challenge: how to recover from packet loss given small tolerable delay between original transmission and playout

- ARQ (retransmission) may be infeasible
  - each ACK/NAK takes one RTT
- Alternative: *Forward Error Correction (FEC)*
  - send redundant bits to allow recovery without retransmission

Recovery from packet loss (1)

Forward Error Correction (FEC): simple scheme

- for every group of *n* chunks create redundant chunk by XORing *n* original chunks
- send out *n*+1 chunks, increasing bandwidth by factor 1/*n*.
- can reconstruct original *n* chunks if at most one lost chunk from *n*+1 chunks

- play out delay: enough time to receive all *n*+1 packets
- Q: What are the tradeoffs?
- A: Larger *n* means
  - Less bandwidth waste
  - Longer play out delay
  - Higher probability that 2 or more chunks will be lost

Recovery from packet loss (1)

- Transmitted packets

Original packet 1:	10100011011011011111011011100101
Original packet 2:	01110101000101010100101100110110
Extra packet (XOR):	11010110011110001011110111010011

- Received packets (#2 was lost)

Original packet 1:	10100011011011011111011011100101
Extra packet:	11010110011110001011110111010011

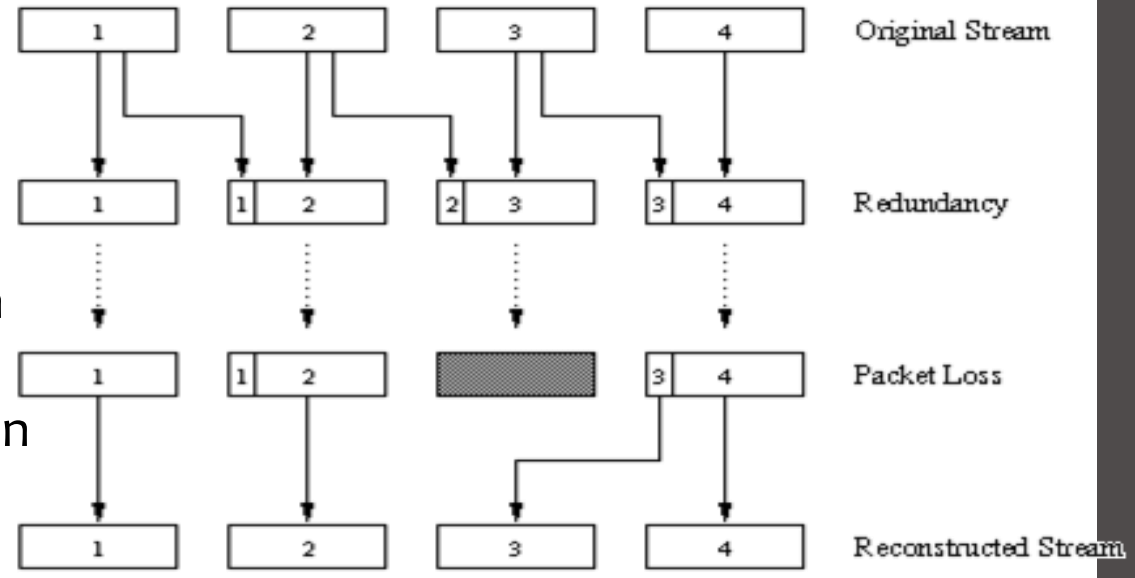
- Packet 2 can be recovered

Original packet 1:	10100011011011011111011011100101
Extra packet:	11010110011110001011110111010011
XOR = packet 2:	01110101000101010100101100110110

Recovery from packet loss (2)

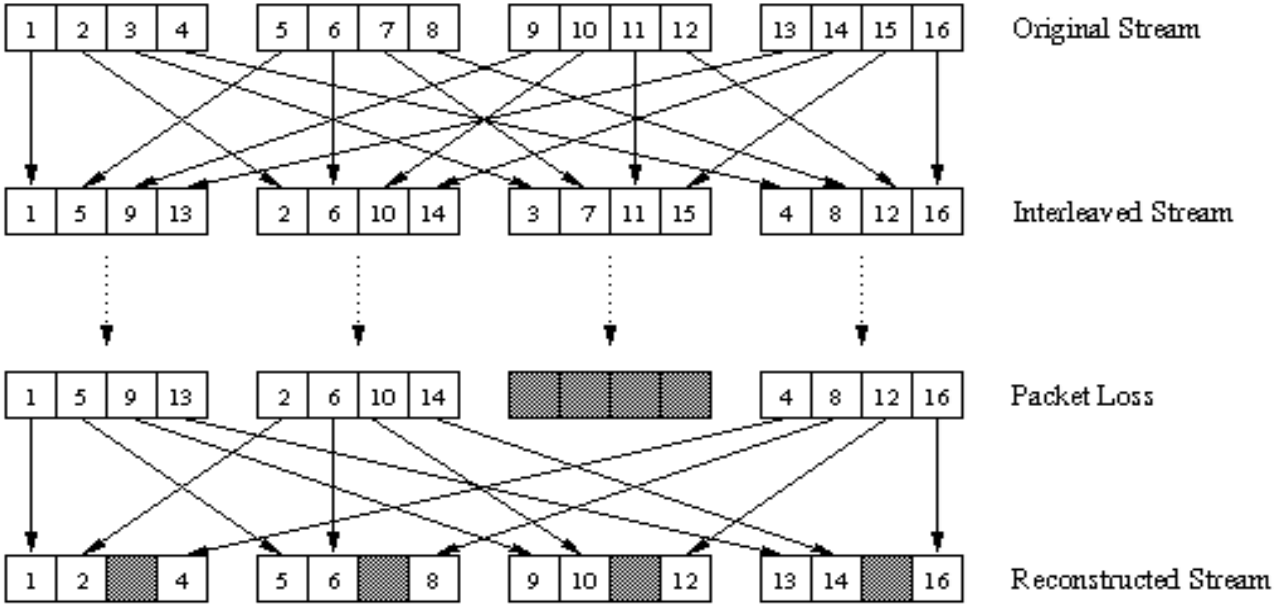
2nd FEC scheme

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



- receiver can conceal non-consecutive losses
- playout only needs to be delayed by one additional packet time
- can also append $(n-1)^{th}$ and $(n-2)^{th}$ low-bit rate chunk

Recovery from packet loss (3)



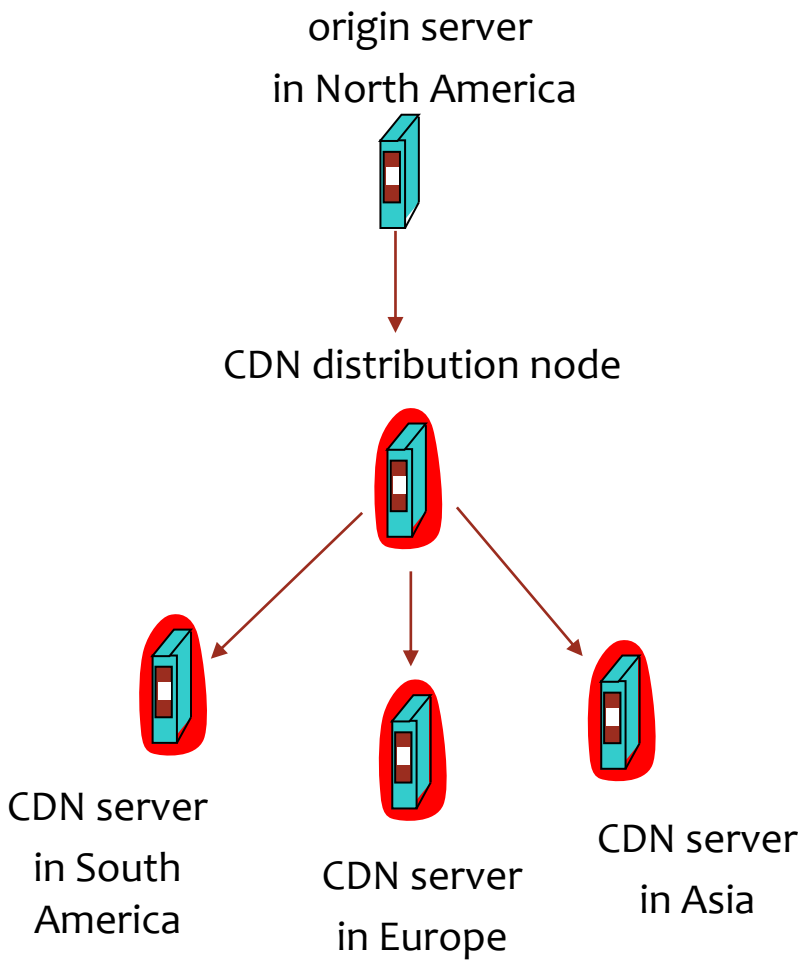
Interleaving

- chunks divided into smaller units
- for example, four 5 msec units per chunk
- packet contains small units from different chunks
- if packet lost, still have most of every chunk
- no redundancy overhead, but increases play out delay

Content delivery 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 the world
  - content uploaded 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



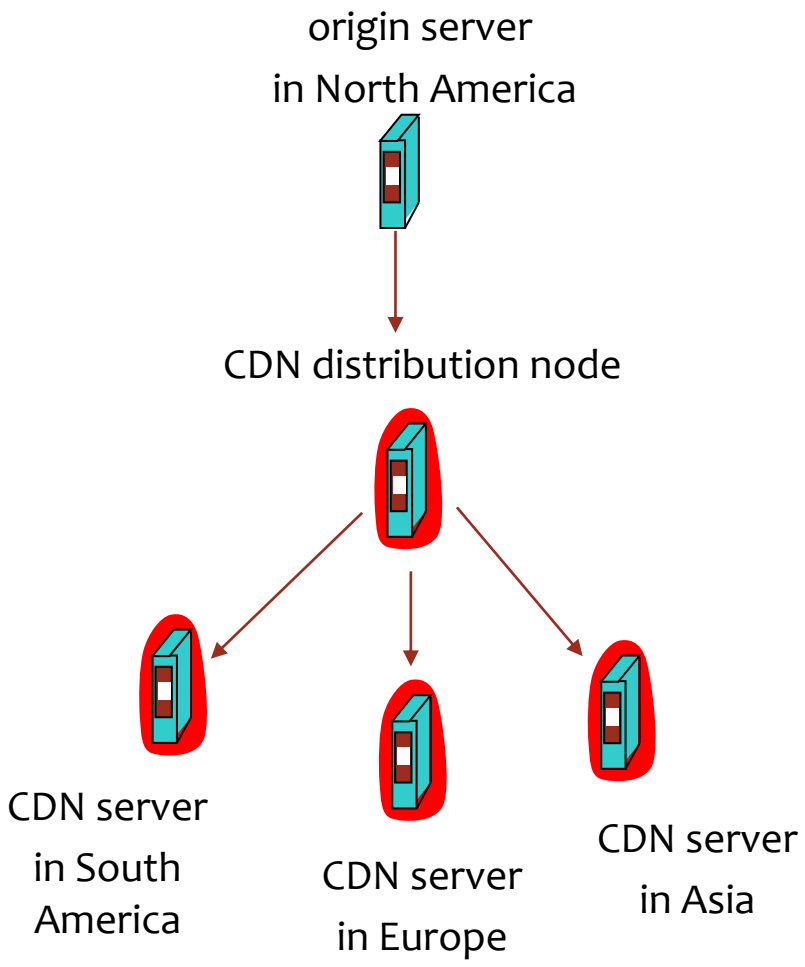
Content delivery networks (CDNs)

Content replication

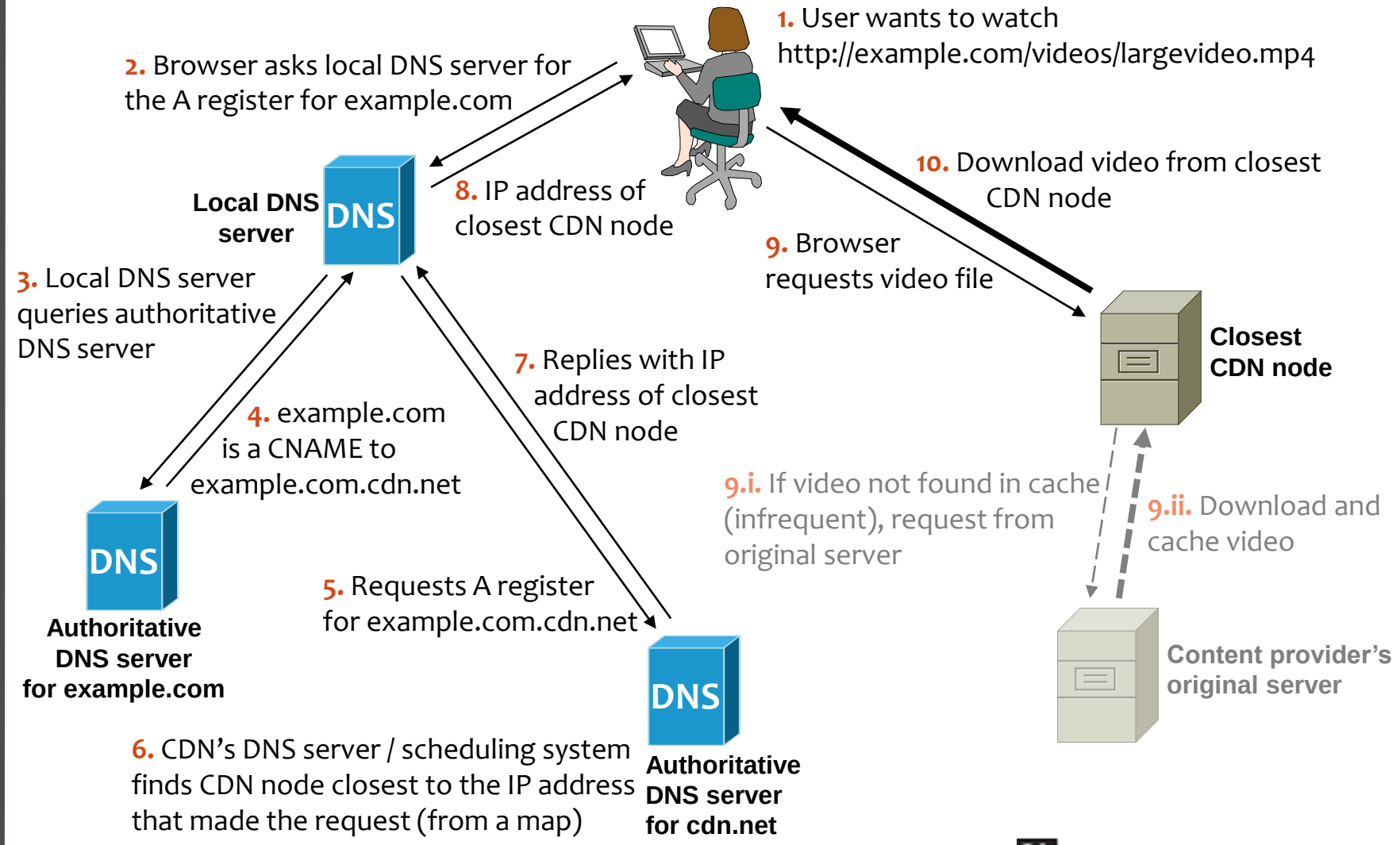
- CDN (e.g., Akamai) customer is the content provider (e.g., BBC)
- CDN replicates customers' content in CDN servers
- When provider updates content, CDN updates servers

Alterative method

- No CDN distribution node
- When first user in region requests a file, local CDN node retrieves it from original server and caches it for future requests



CDN Example


1. User wants to watch http://example.com/videos/largevideo.mp4
2. Browser asks local DNS server for the A register for example.com
Local DNS server
DNS
8. IP address of closest CDN node
10. Download video from closest CDN node
9. Browser requests video file
Closest CDN node
3. Local DNS server queries authoritative DNS server
4. example.com is a CNAME to example.com.cdn.net
7. Replies with IP address of closest CDN node
9.i. If video not found in cache (infrequent), request from original server
9.ii. Download and cache video
DNS
Authoritative DNS server for example.com
5. Requests A register for example.com.cdn.net
Content provider's original server
6. CDN's DNS server / scheduling system finds CDN node closest to the IP address that made the request (from a map)
DNS
Authoritative DNS server for cdn.net

More about CDNs

routing requests

- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server
  - DNS server determines ISP from which query originates (from source IP address)
  - uses "map" to determine best/closest CDN server
  - note: using global DNS servers (Google, OpenDNS, …) may lead to performance issues
    - use of EDNS Client Subnet (ECS) extension (RFC-7871) solves this issue, if supported
- CDN nodes create an 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 play out delay to compensate for jitter
- server side matches stream bandwidth to available client-to-server path bandwidth
  - choose 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 7 outline

7.1 multimedia networking applications

7.2 streaming stored audio and video

7.3 making the best out of best effort service

7.4 protocols for real-time interactive applications
RTP, RTCP, SIP

7.5 providing multiple classes of service

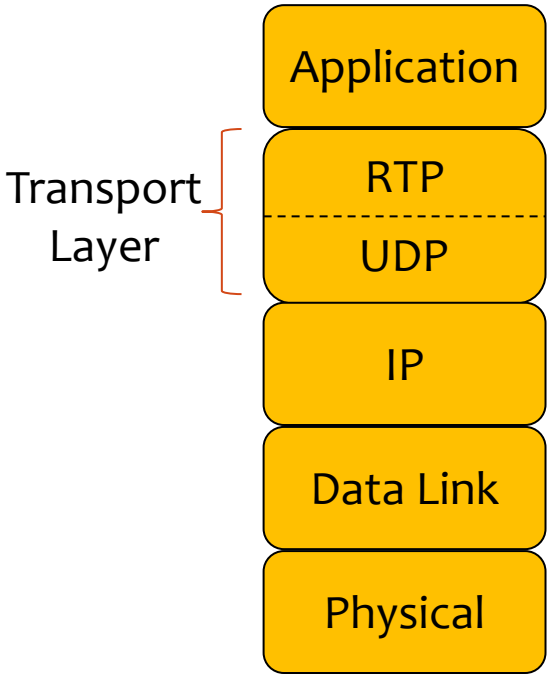
7.6 providing QoS guarantees

Real-time Transport Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type (CODEC) 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 interoperate

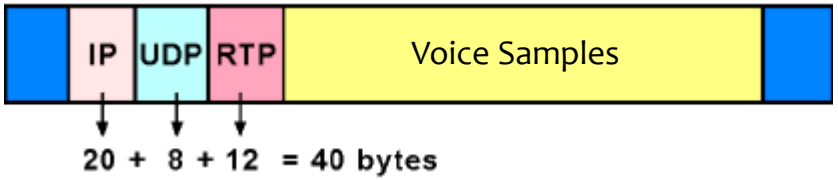
RTP runs on top of UDP

- RTP libraries provide transport-layer interface that extends UDP:
  - port numbers, IP addresses
  - payload type identification
  - packet sequence numbering
  - time-stamping



RTP Example

- 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
  - encapsulated in UDP segment
- RTP header indicates type of audio encoding in each packet
  - sender may change encoding during conference
- RTP header also contains sequence numbers and 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 provide best-effort service, making no special effort to ensure that RTP packets arrive at destination in a timely manner
  - RTP helps end systems deal with issues of best-effort packet delivery

RTP Header

- Payload Type (7 bits): Indicates type of encoding currently being used. If sender changes encoding in middle of conference, receiver is informed via payload type field. (RFC3551)
  - 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
  - may be used to detect packet loss and to restore packet sequence.

RTP Header (2)

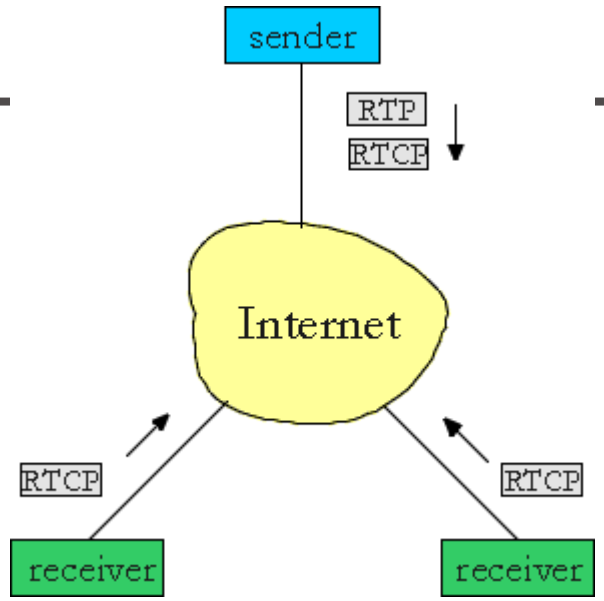
- Timestamp field (32 bits long): sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock typically increments by one for each sampling period (e.g., every 125 μs for 8 kHz sampling clock)
  - if application generates chunks of 160 encoded samples, then timestamp increases 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 RTP stream. Each stream in RTP session should have distinct SSRC. (Synchronization Source Identifier)
- Marker bit: marks special packets, with specific meaning dependent on the payload type, e.g.,
  - First packet in talk spurt for audio
  - Last packet of a frame for video

Real-time Transport Control Protocol (RTCP)

- works in conjunction with RTP
- each participant in RTP session periodically transmits RTCP control packets to all other participants
- each RTCP packets may contain sender or receiver reports
  - statistics useful to application: # packets sent, # packets lost, inter-arrival 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 the session use the multicast address



- RTP and RTCP packets distinguished by their distinct port numbers
- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases (bandwidth scaling)

RTCP Messages

Receiver report:

- fraction of packets lost, last sequence number, average inter-arrival jitter

Sender report:

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

Source description:

- 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 an 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 the most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created
- receivers use association to synchronize play out of audio and video

Synchronization of Streams: Example

- Two streams
  - Audio: 22100 samples/s
  - Video: 25 frames/s
- Last sender reports
  - For audio, matched timestamp 4044300 to 2022-02-22 16:28:03
  - For video, matched timestamp 4625 to 2022-02-22 16:28:05
- Which video frame should be played simultaneously with the audio sample with timestamp 4110600?

- 4110600 - 4044300 = 66300 (samples)
- 66300/22100 = 3 (seconds)
- Current wall clock time (NTP) is 2022-02-22 16:28:03 + 3 = 2022-02-22 16:28:06
- 2022-02-22 16:28:06 - 2022-02-22 16:28:05 = 1 (s)
- 25 × 1 = 25 (frames)
- 4625 + 25 = 4650
- Answer: the frame with timestamp 4650

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 average RTCP packet size (across entire session) and dividing by allocated rate

See RFC 3550: Sec 6.2.1 for # of Session Members

Jitter Calculation

- R_k – arrival time of packet k
- S_k – RTP timestamp of packet k
- 1/16 factor amortizes noise and leads to a good convergence (EWMA)

Delay difference between packets

$$D(i,j) = (R_j - S_j) - (R_i - S_i) = (R_j - R_i) - (S_j - S_i)$$

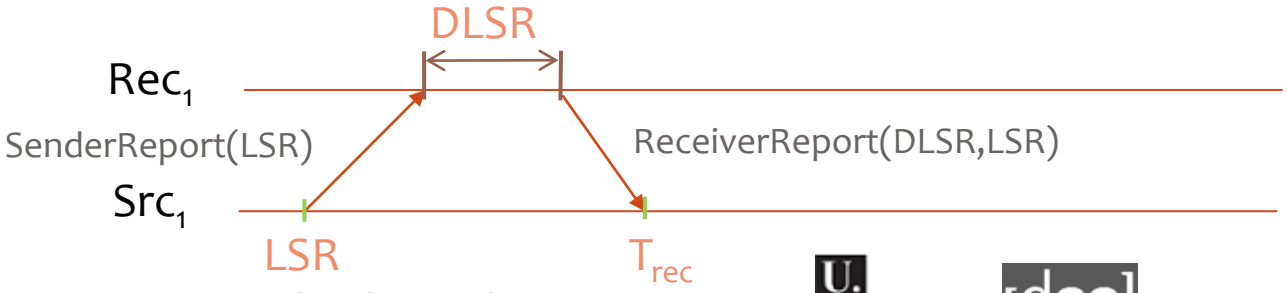
$$J(i) = \left(1 - \frac{1}{16}\right) J(i-1) + \frac{1}{16} |D(i-1,i)|$$

Estimated jitter

RTT Calculation

- If Rec_1 sends a RR (Receiver Report) that Src_1 receives at time T_{rec}
- The report has:
  - LSR of Src_1 – time at which Src_1 sent the last report received by Rec_1
  - DLSR – time since Rec_1 received the last report from Src_1 and sent this RR

$$RTT = T_{rec} - LSR - DLSR$$



Overview of...

RTSP	RTP	RTCP
• controls the playing of a multimedia stream (play, pause, etc.)	• transports multimedia packets (with associated metadata)	• sends reports regarding RTP media flows (from receivers and senders)

SIP: Session Initiation Protocol [RFC 3261]

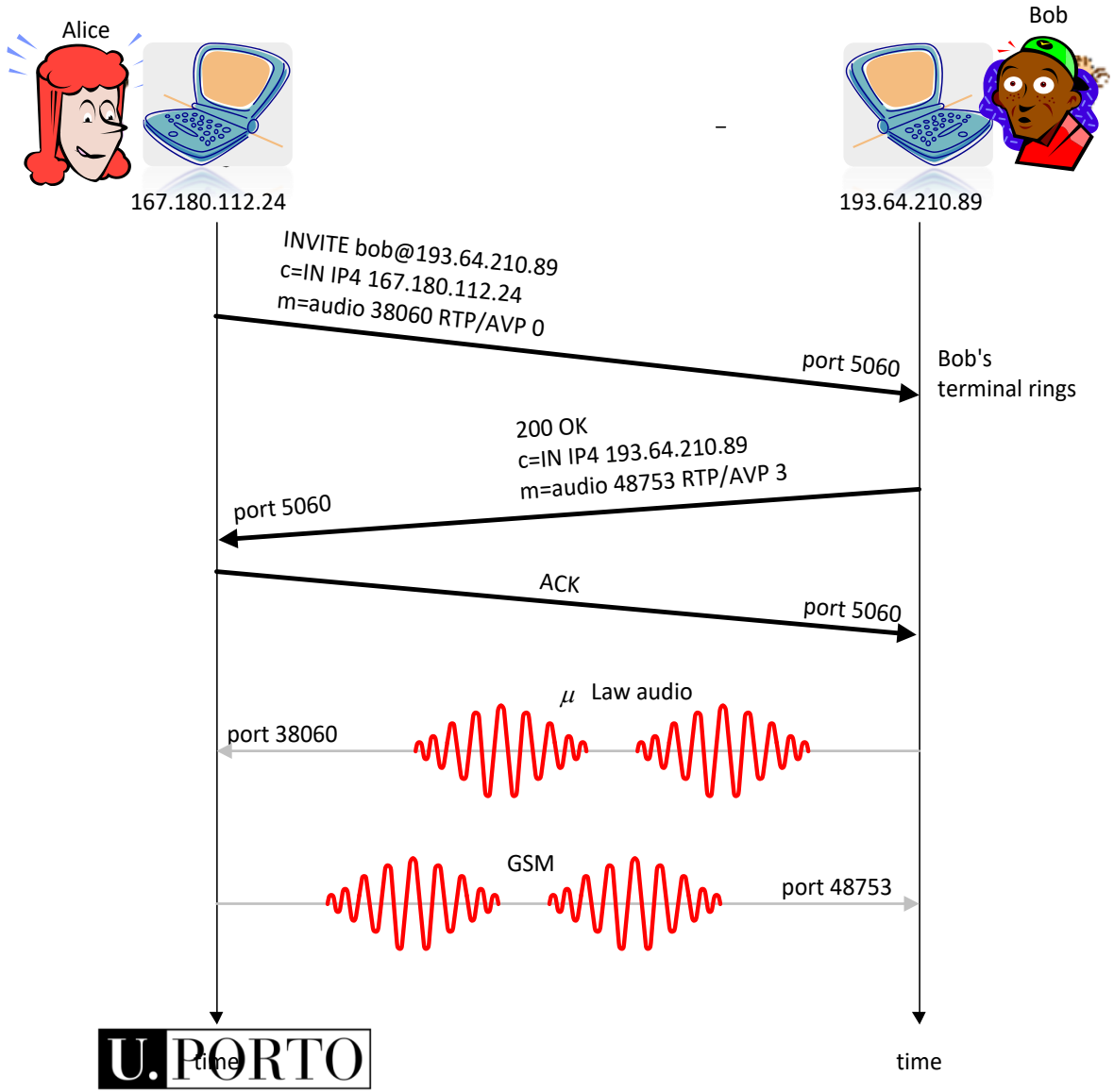
SIP long-term vision:

- 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

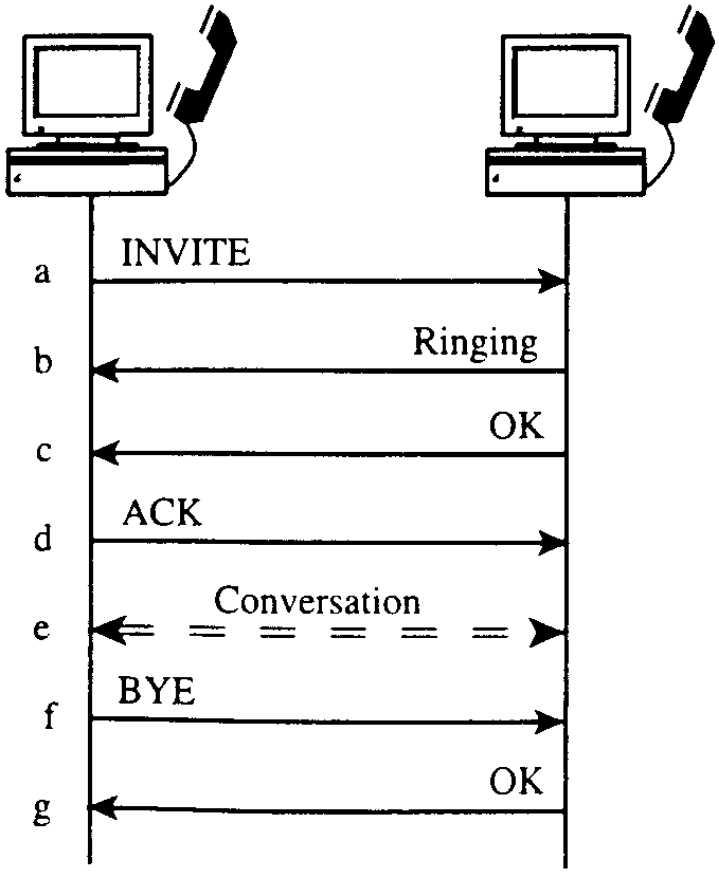
- Setting up a call: mechanisms
  - for caller to let callee know she wants to establish a call
  - so caller and callee can agree on media type and encoding
  - to end the call
- determining current IP address of callee:
  - maps mnemonic identifier to current IP address
- call management:
  - add new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

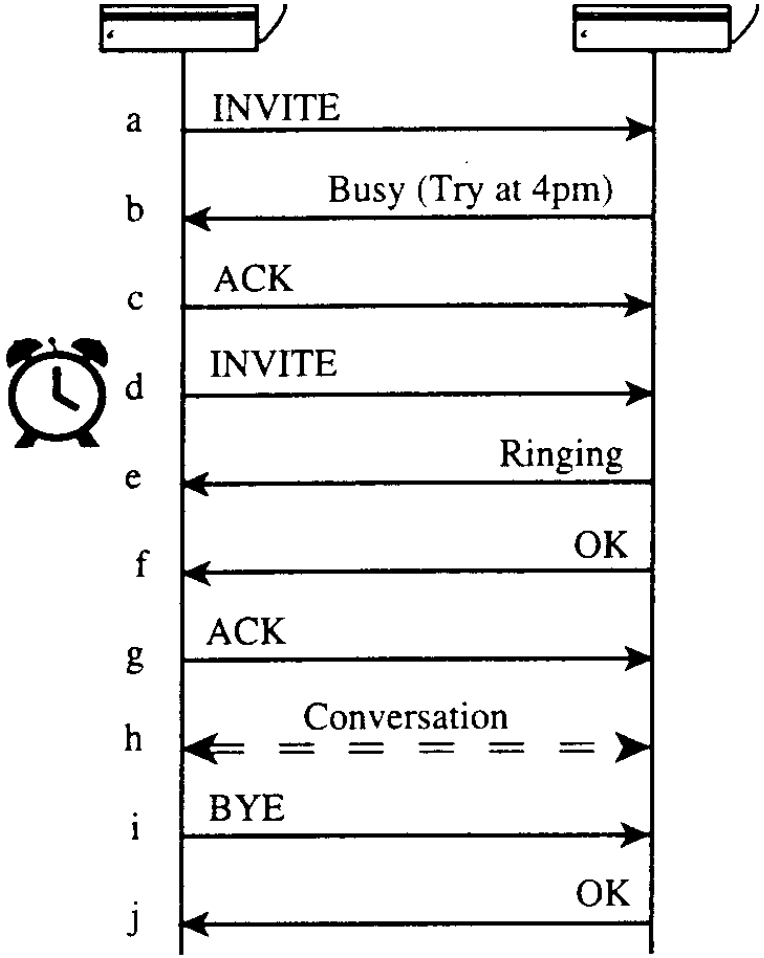
Setting up a call to known IP address



- 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
- default SIP port number is 5060

Setting up a call (cont)





Setting up a call (more)

- codec negotiation:
  - suppose Bob doesn't have PCM ulaw encoder
  - Bob will instead reply with 488 Not Acceptable Here, listing his encoders
  - Alice can then send new INVITE message, asking for different encoder
  - negotiation is made through *Session Description Protocol* messages
    - transported by SIP in the body of the message
- rejecting a call
  - Bob can reject with replies "busy," "gone," "payment required," "forbidden"
- *media can be sent over RTP or some other protocol*
  - SIP doesn't care

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
```

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

Notes:

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

SDP (RFC 4566): information

Session Description Protocol used to negotiate:

- Media streams used
  - May include multiple streams with diverse content
- Destination Addresses
  - May use multicast
- Ports for each stream
- Payload type
- For broadcast session (e.g. TV, radio)
  - Start and finish time
  - Originator

Example

Request

```
INVITE sip:UserB@there.com SIP/2.0
```

```
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>
Call-ID: 12345600@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: BigGuy <sip:UserA@here.com>
Content-Type: application/sdp
Content-Length: 147
```

```
v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 100.101.102.103
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Reply

```
SIP/2.0 200 OK
```

```
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>;tag=65a35
Call-ID: 12345601@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: LittleGuy <sip:UserB@there.com>
Content-Type: application/sdp
Content-Length: 134
```

```
v=0
o=UserB 2890844527 2890844527 IN IP4 there.com
s=Session SDP
c=IN IP4 110.111.112.113
t=0 0
m=audio 3456 RTP/AVP 0 96
a=rtpmap:0 PCMU/8000
a=rtpmap:96 L8/8000
```

Headers

Body

IP address for receiving data

Port for receiving

Media types supported

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, IP address may change
  - user has different IP devices (PC, smartphone, 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)
- support services provided by two entities:
  - SIP registrar server
  - SIP proxy server

SIP Registrar

- When Bob starts SIP client, it sends a SIP REGISTER message to Bob's registrar server

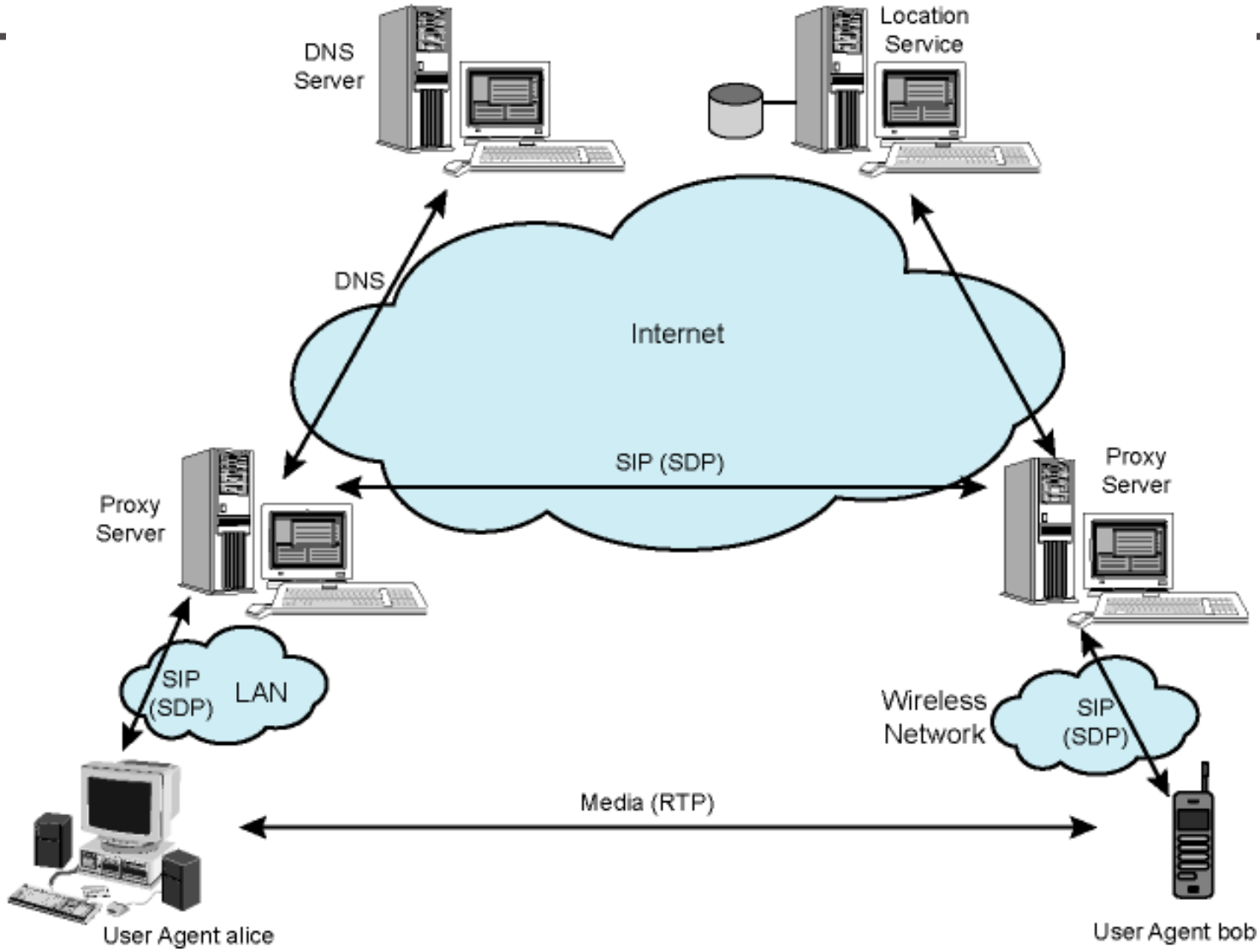
Register Message:

```
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

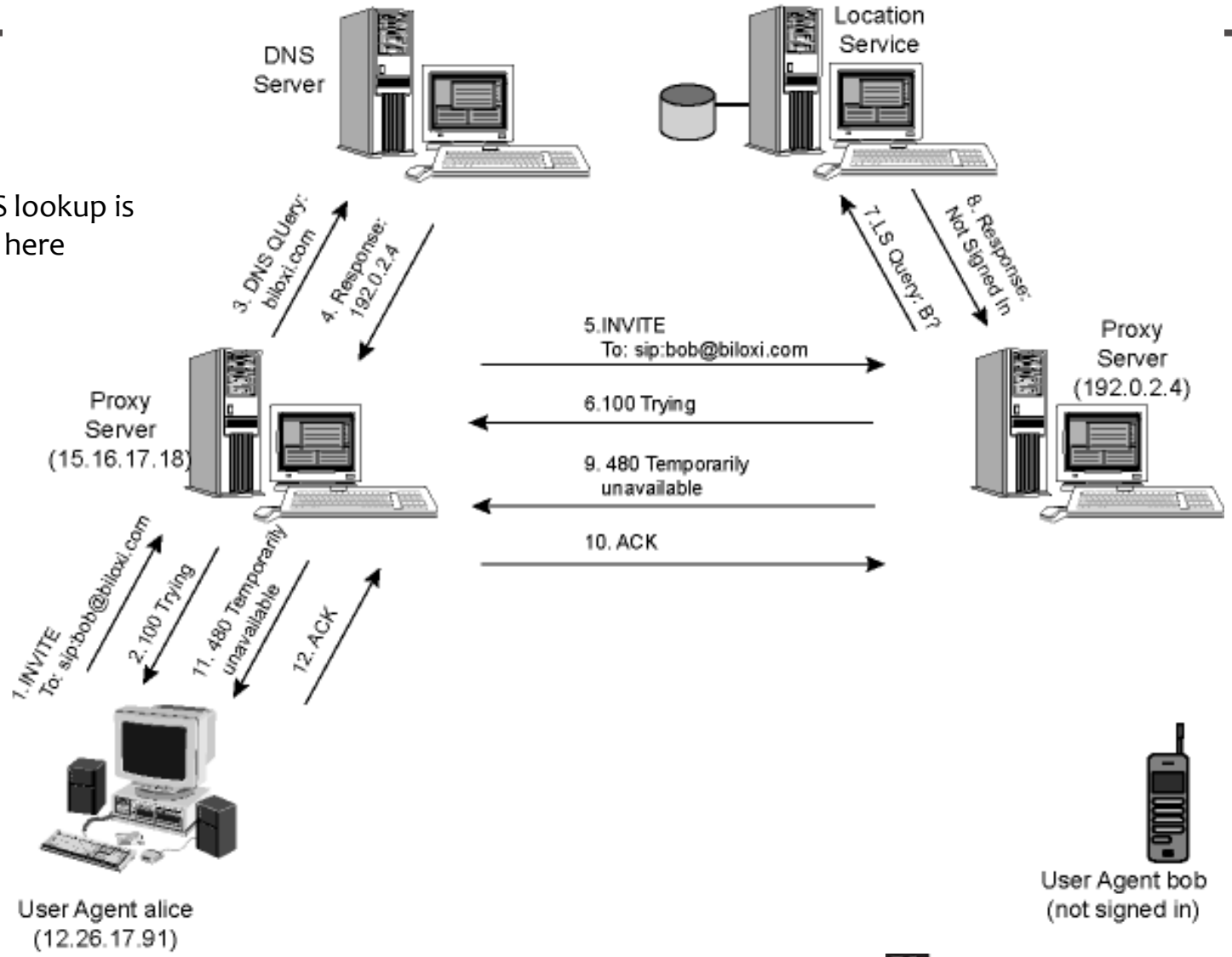
- Alice sends INVITE message to her proxy server
  - contains address sip:bob@domain.com
- proxy responsible for routing SIP messages to callee
  - possibly through multiple proxies
- callee sends response back through the same set of proxies (in reverse order)
  - using *Via:* headers
- proxy returns SIP response message to Alice
  - contains Bob's IP address

SIP Components

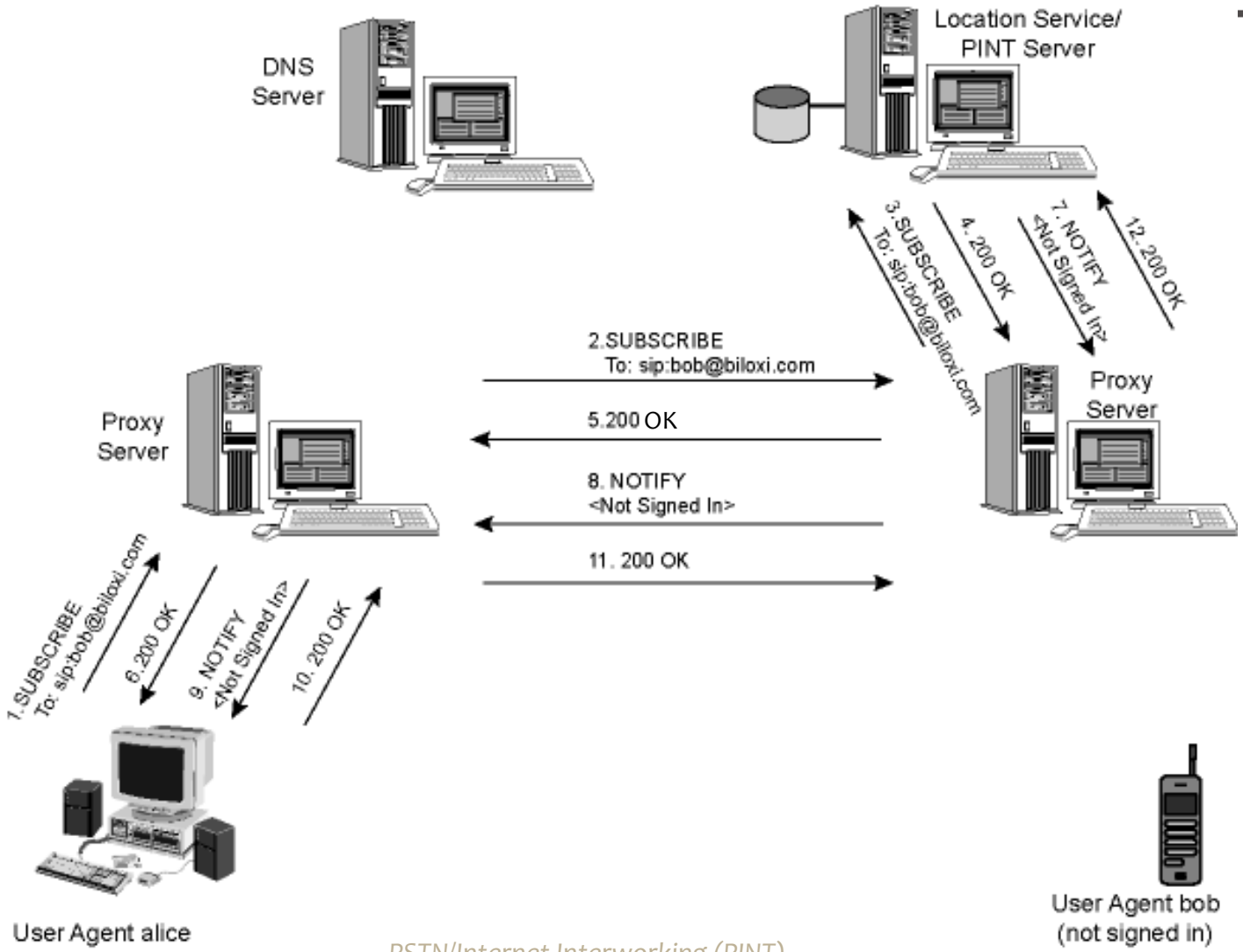


Example: Call Attempt

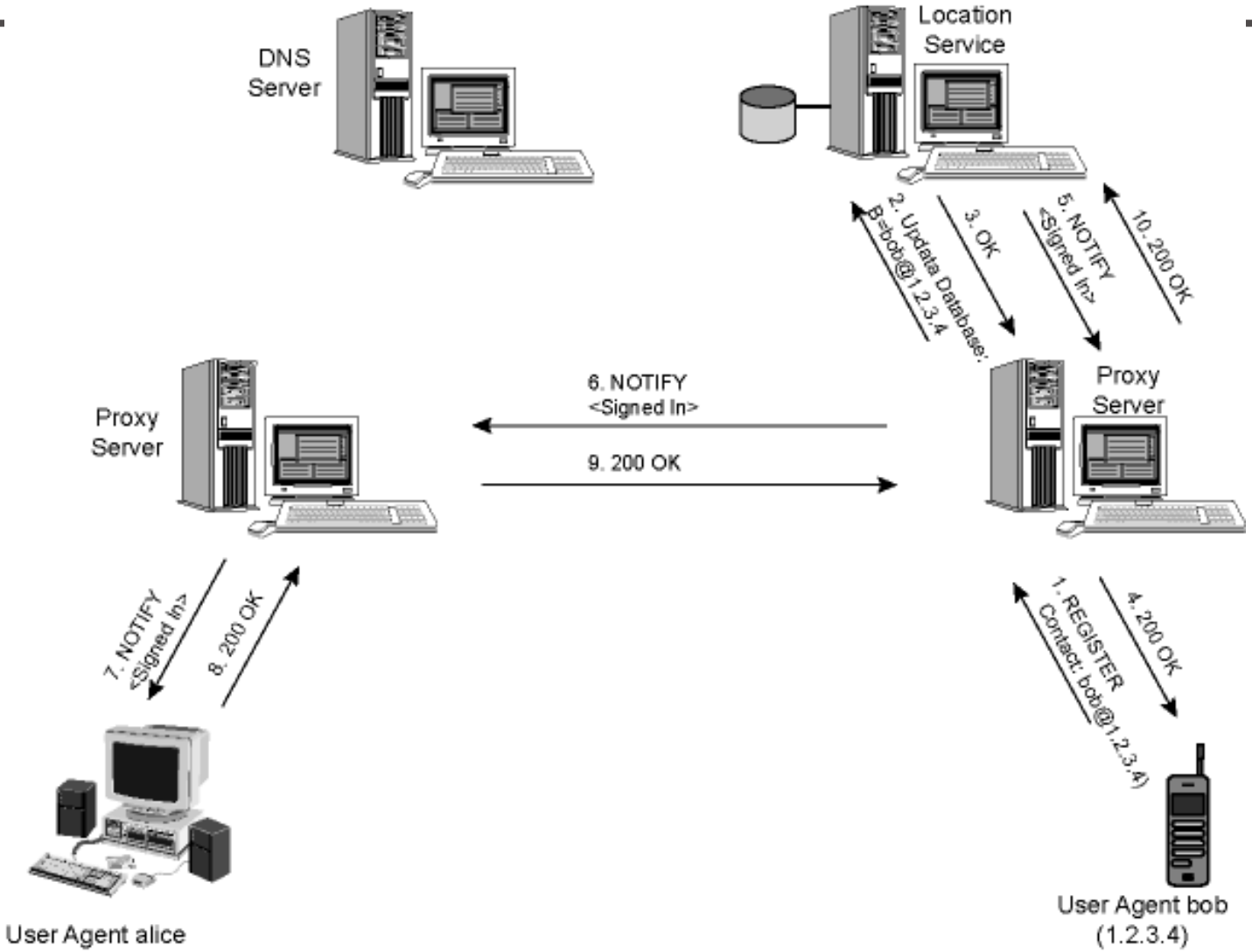
Note: DNS lookup is simplified here



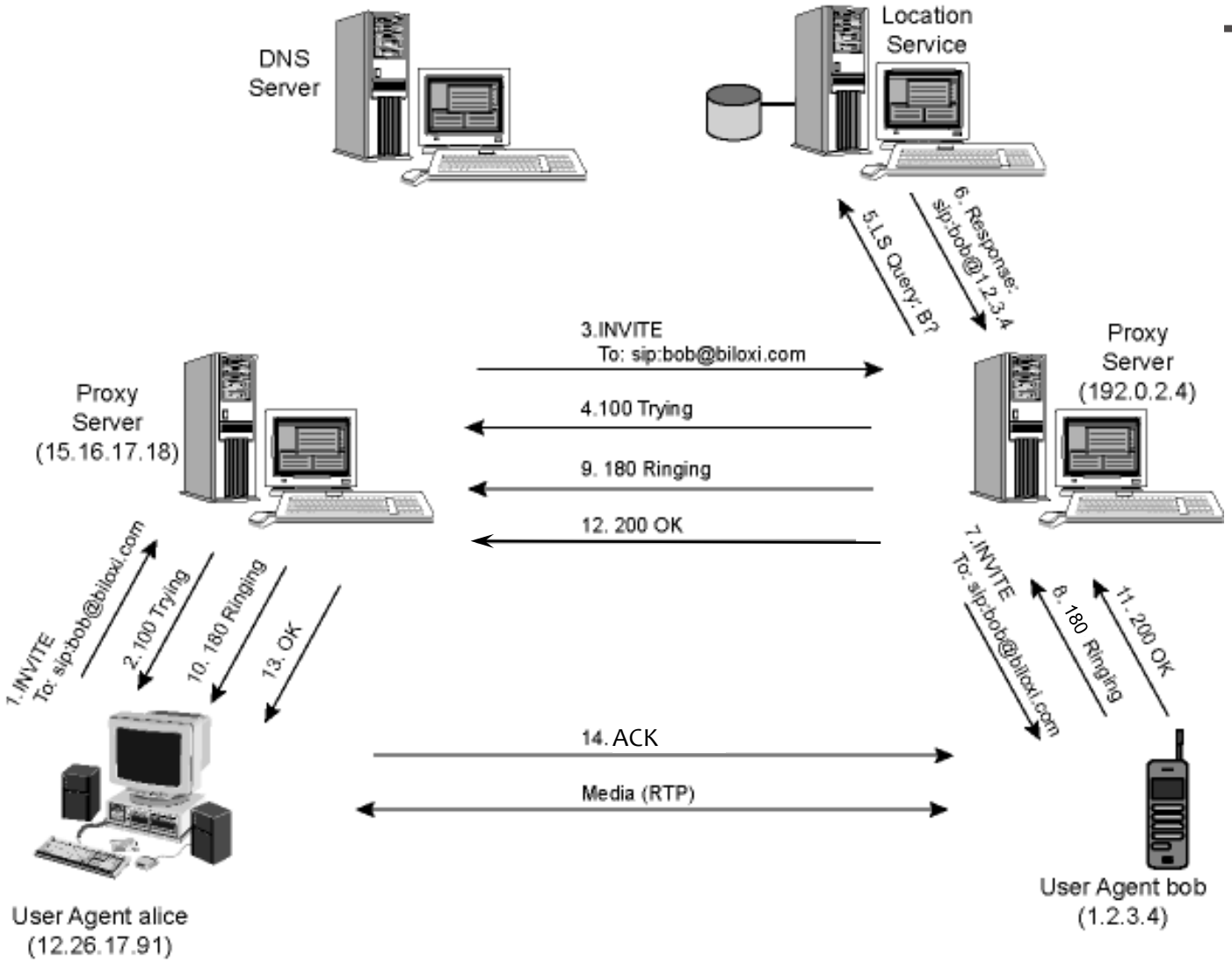
Example: Subscribe to



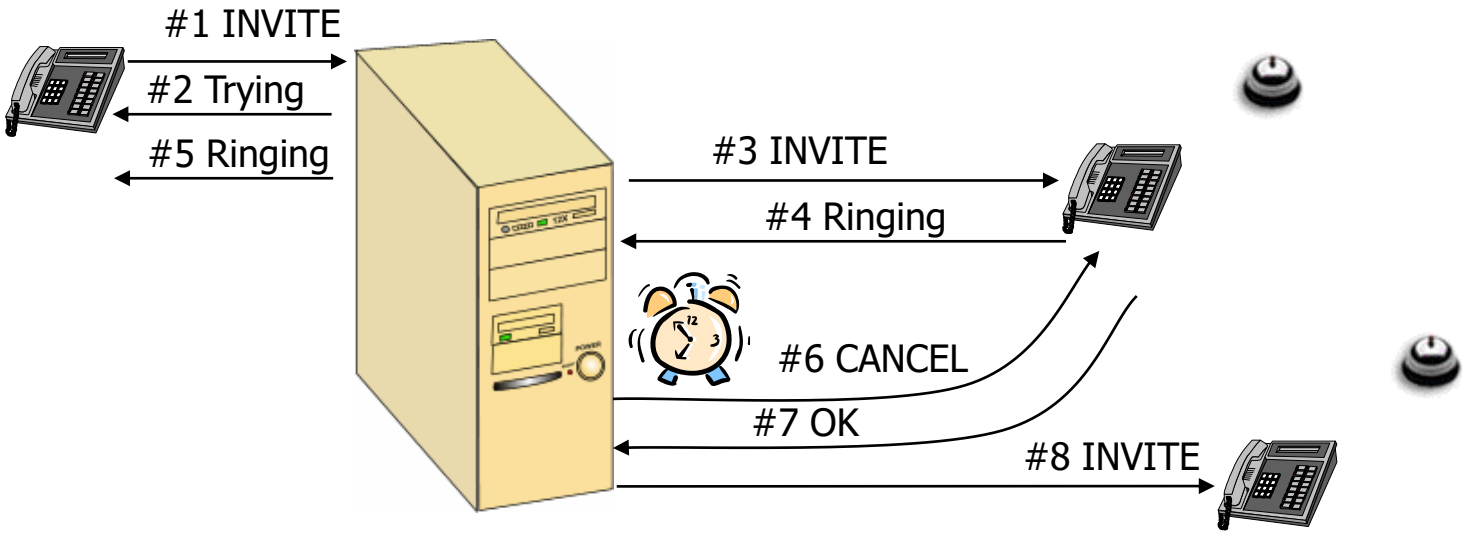
Example: Register and Notification



Example: Call



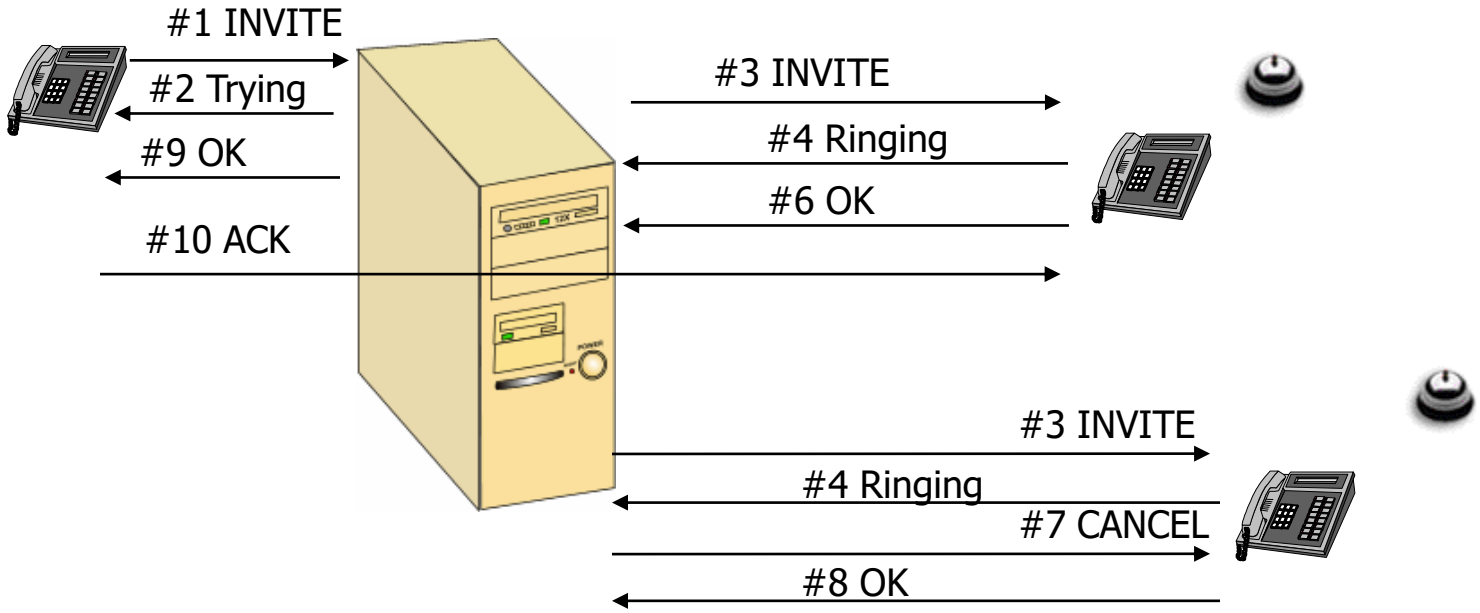
Example: Recursive Forking



- Proxy cancels request after *timeout*.

Example from Dorgham Sisalem@FhG Fokus

Example: Forking



- Proxy rings the 2 terminals simultaneously, accepting 1st responder.
- Proxy cancels all pending requests.

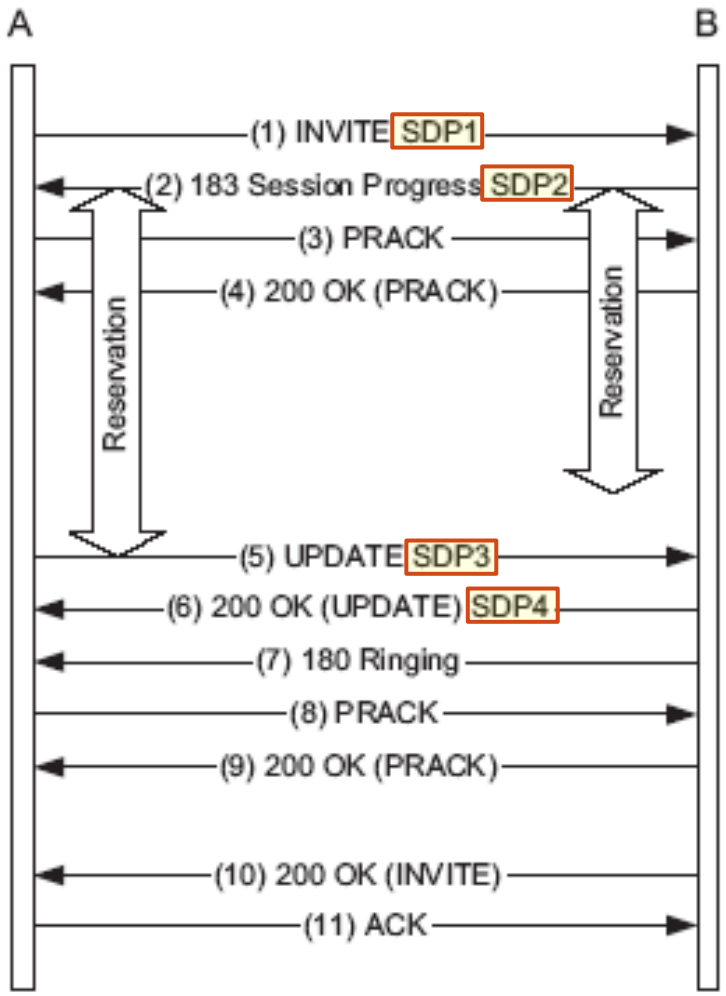
Example from Dorgham Sisalem@FhG Fokus

SIP and QoS

- SIP does **NOT** provide QoS
- However INVITE SDP fields may indicate QoS requirements
  - Resources may be acquired before RING
  - These are *pre-conditions* in SDP for the call

SIP and QoS: example

See RFC3312, section 13.1



SDP1

```
m=audio 20000 RTP/AVP 0
c=IN IP4 192.0.2.1
a=curr:qos e2e none
a=des:qos mandatory e2e sendrecv
```

SDP2

```
m=audio 30000 RTP/AVP 0
c=IN IP4 192.0.2.4
a=curr:qos e2e none
a=des:qos mandatory e2e sendrecv
a=conf:qos e2e recv
```

From the point of view of B

SDP3

```
m=audio 20000 RTP/AVP 0
c=IN IP4 192.0.2.1
a=curr:qos e2e send
a=des:qos mandatory e2e sendrecv
```

From the p.o.v. of A

SDP4

```
m=audio 30000 RTP/AVP 0
c=IN IP4 192.0.2.4
a=curr:qos e2e sendrecv
a=des:qos mandatory e2e sendrecv
```

Chapter 7 outline

7.1 multimedia networking applications

7.2 streaming stored audio and video

7.3 making the best out of best effort service

7.4 protocols for real-time interactive applications

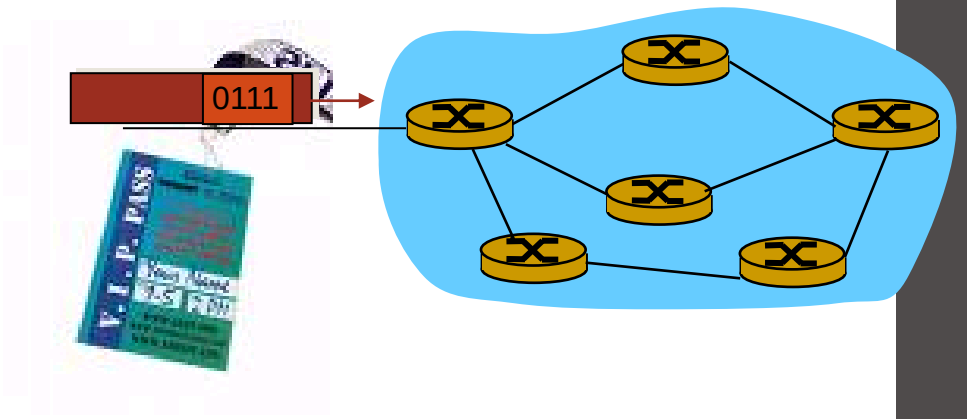
RTP, RTCP, SIP

7.5 providing multiple classes of service

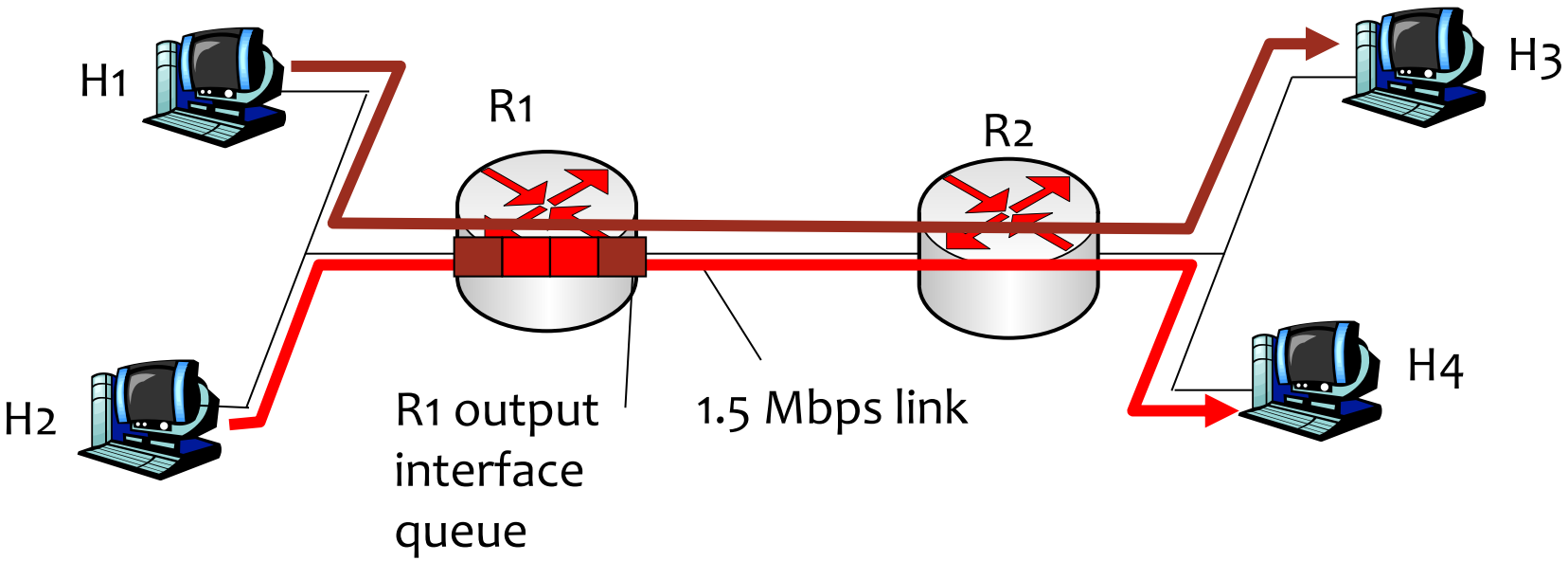
7.6 providing QoS guarantees

Providing Multiple Classes of Service

- thus far: making the best of best effort service
  - one-size fits all service model
- alternative: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service vs. regular service)
- granularity: differential service among multiple classes, not among individual connections
- history: ToS bits

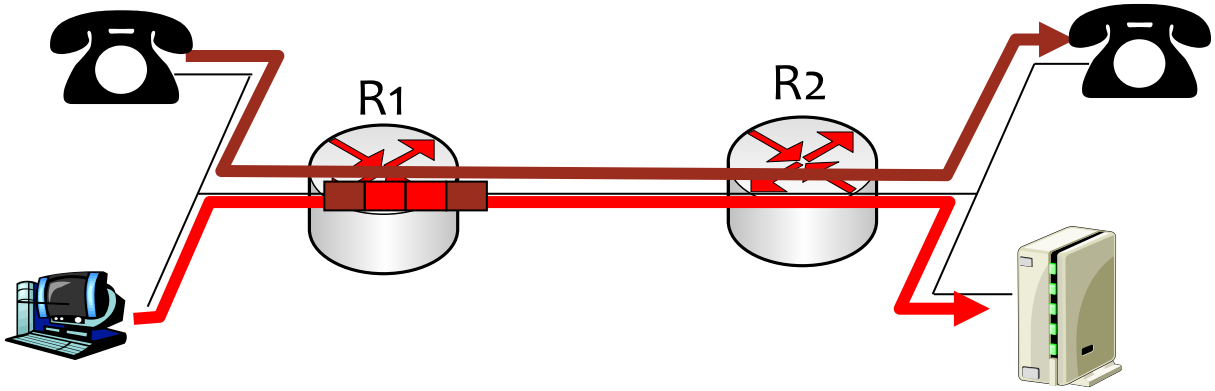


Multiple classes of service: scenario



Scenario 1: mixed FTP and audio

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

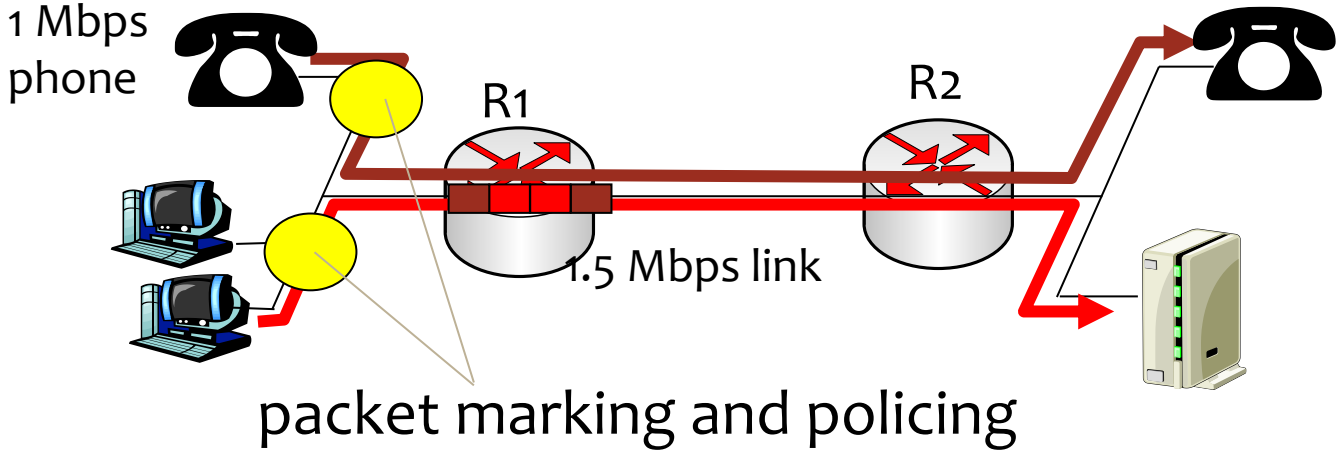


Principle 1

packet marking for router to distinguish between different classes, 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

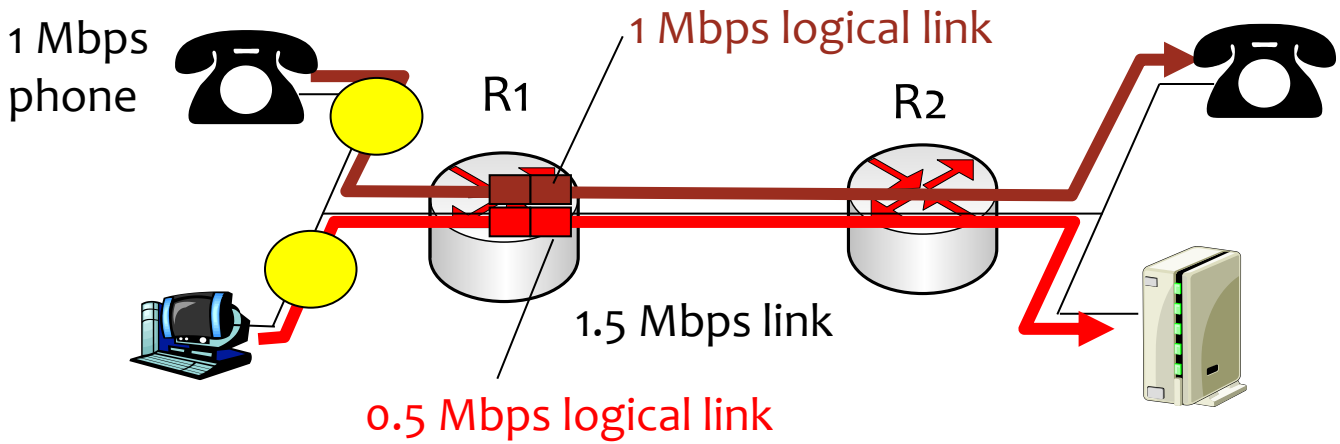


Principle 2

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 flow doesn't use its allocation

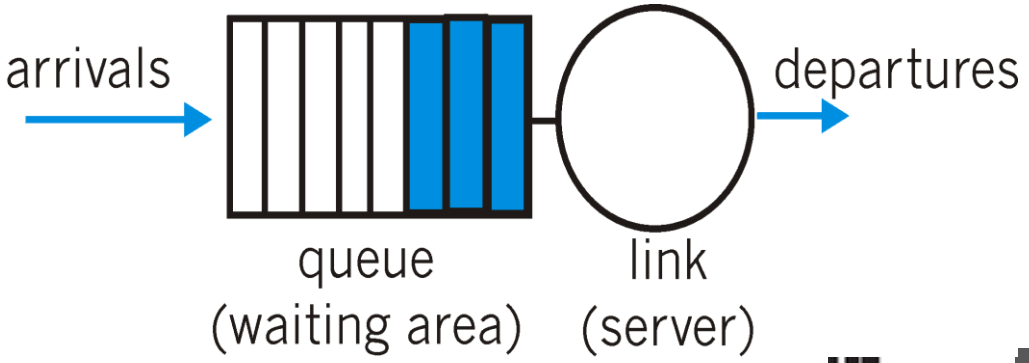


Principle 3

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

Scheduling Policies

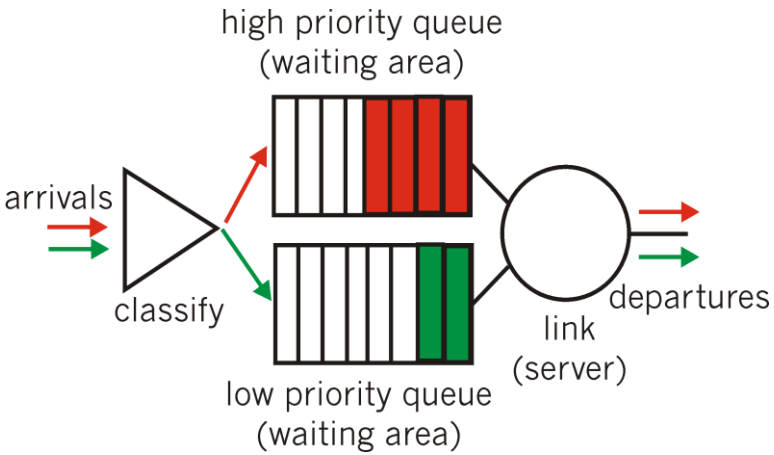
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - discard policy: if packet arrives to full queue, which one to discard?
    - tail drop: drop arriving packet
    - head drop: drop oldest packet in queue
    - priority: drop on priority basis
    - random: drop randomly

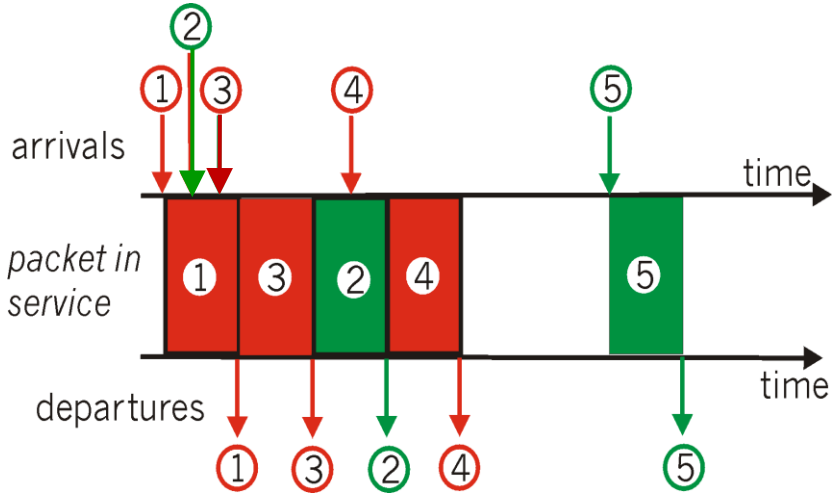


Scheduling Policies: Priority

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..
  - non-preemptive: transmission of a lower priority packet is not interrupted

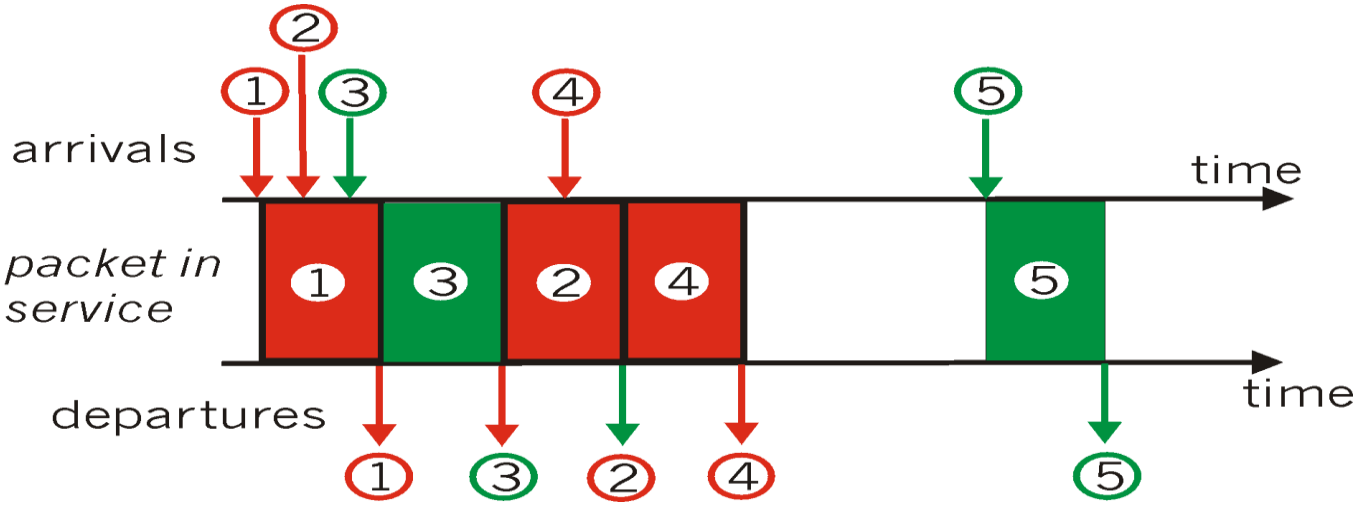




Scheduling Policies: Round Robin

Round Robin scheduling:

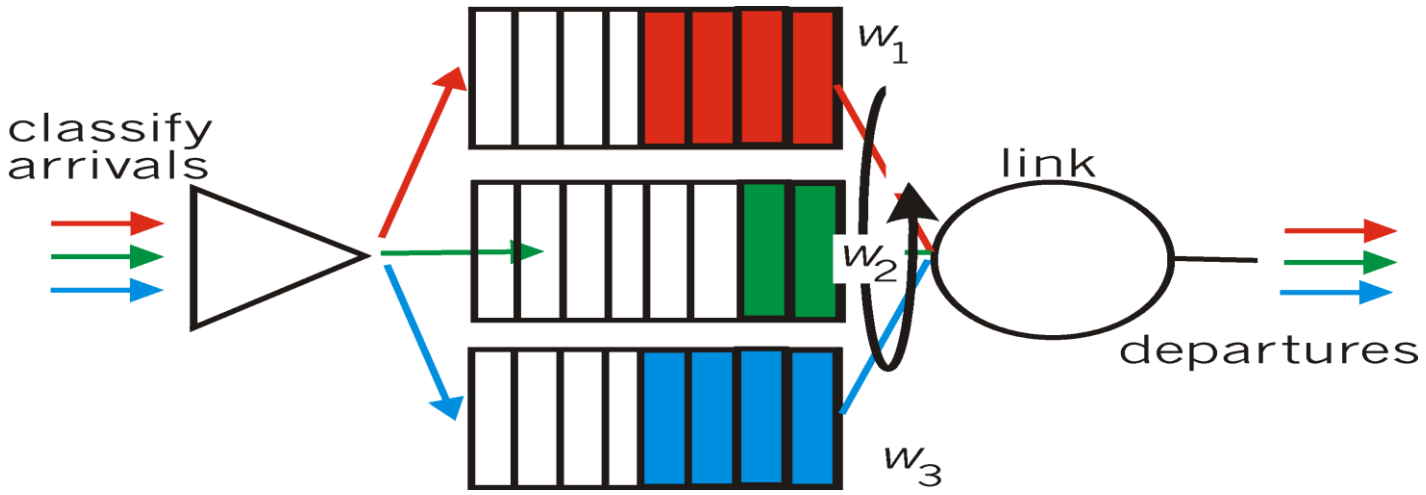
- multiple classes
- cyclically scan class queues, serving one packet from each class (if available)



Scheduling Policies: WFQ

Weighted Fair Queuing:

- an improved variant of Weighted Round Robin
- each class gets *weighted* amount of service in each cycle



Traffic Shaping and Policing

Goal: limit traffic to not exceed declared parameters (traffic profile)

Commonly used criteria:

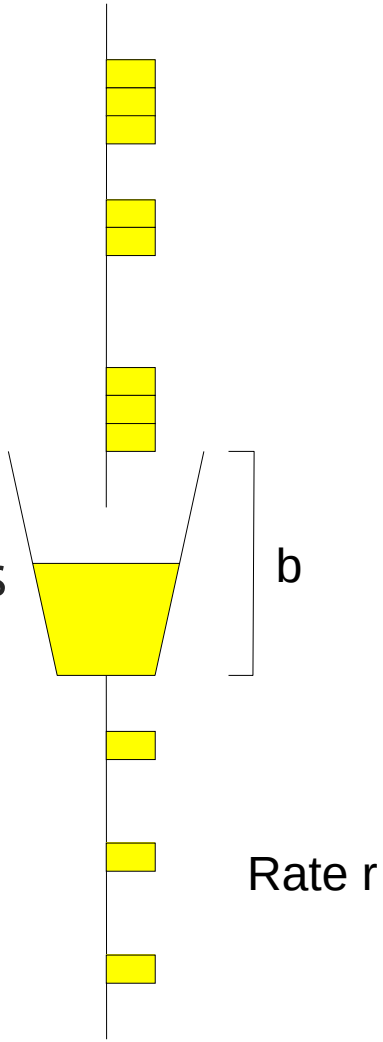
- *(Long term) Average Rate:* how many packets can be sent per time unit (in the long run)
  - crucial question: what is the interval length? 100 pkts/sec or 6000 pkts/min have same average!
- *Peak Rate:* same but over short periods
  - e.g., 6000 pkts/min average rate 30000 pkts/min peak rate
- *(Max.) Burst Size:* max. number of packets sent consecutively (with no intervening idle)

How to deal with out-of-profile packets to enforce the profile?

- Traffic shaping: delay out-of-profile packets
- Traffic policing: drop out-of-profile packets

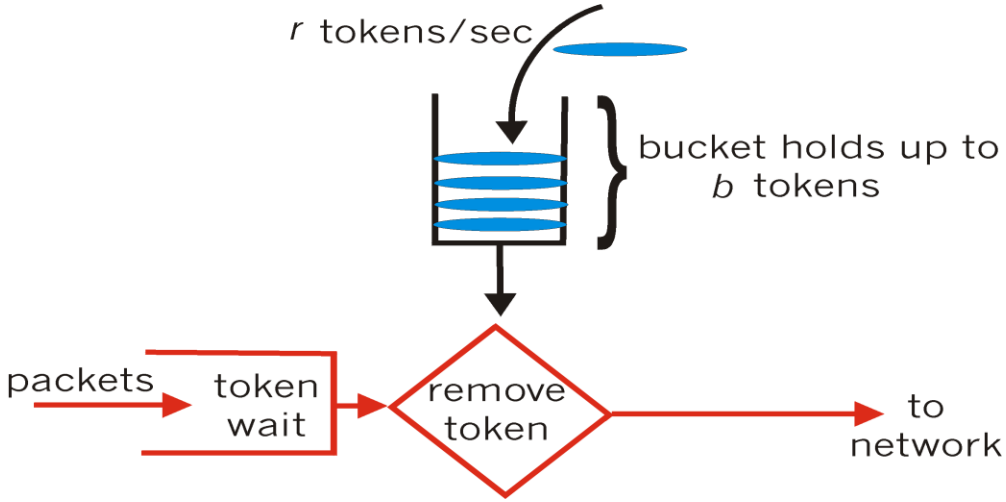
Leaky Bucket

- 2 Parameters
  - b - Bucket size
  - r - Rate
- Packets have limited exit rate of R
  - b/r is the max queuing delay
  - Bursts exceeding b lead to packet loss
  - Inter-departure time of L/r for L sized packets



Token Bucket

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

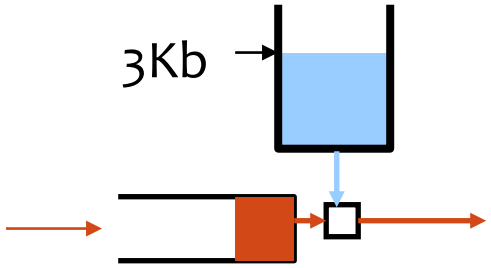


- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over **any** interval of length t, the number of packets admitted is always* $\leq (r \cdot t + b)$

Token Bucket: Example

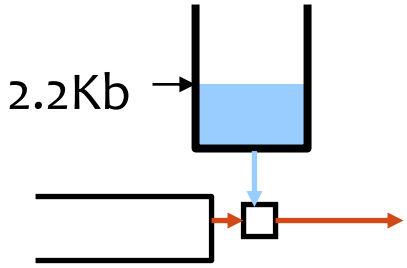
- r = 100 Kbps b = 3 Kb R = 500 Kbps

(a)



T = 0ms : 1Kb packet arrives

(b)

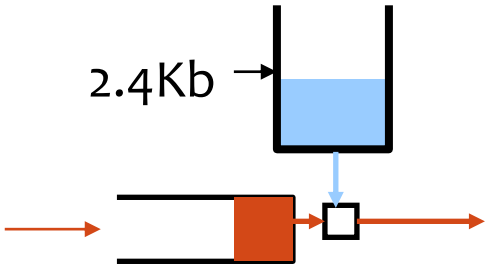


T = 2ms : packet transmitted

b = 3Kb–1Kb+2ms*100Kbps = 2.2Kb

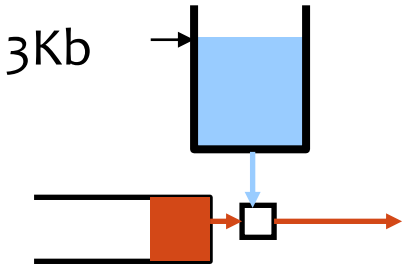
Slide from Stoica Katz

(c)



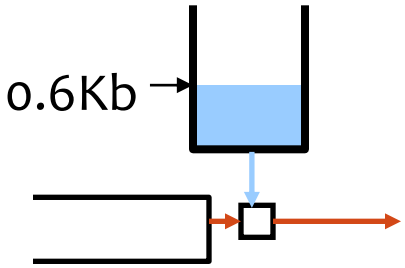
T = 4ms : 3Kb pkt arrives, **must wait until bucket has enough tokens**

(d)



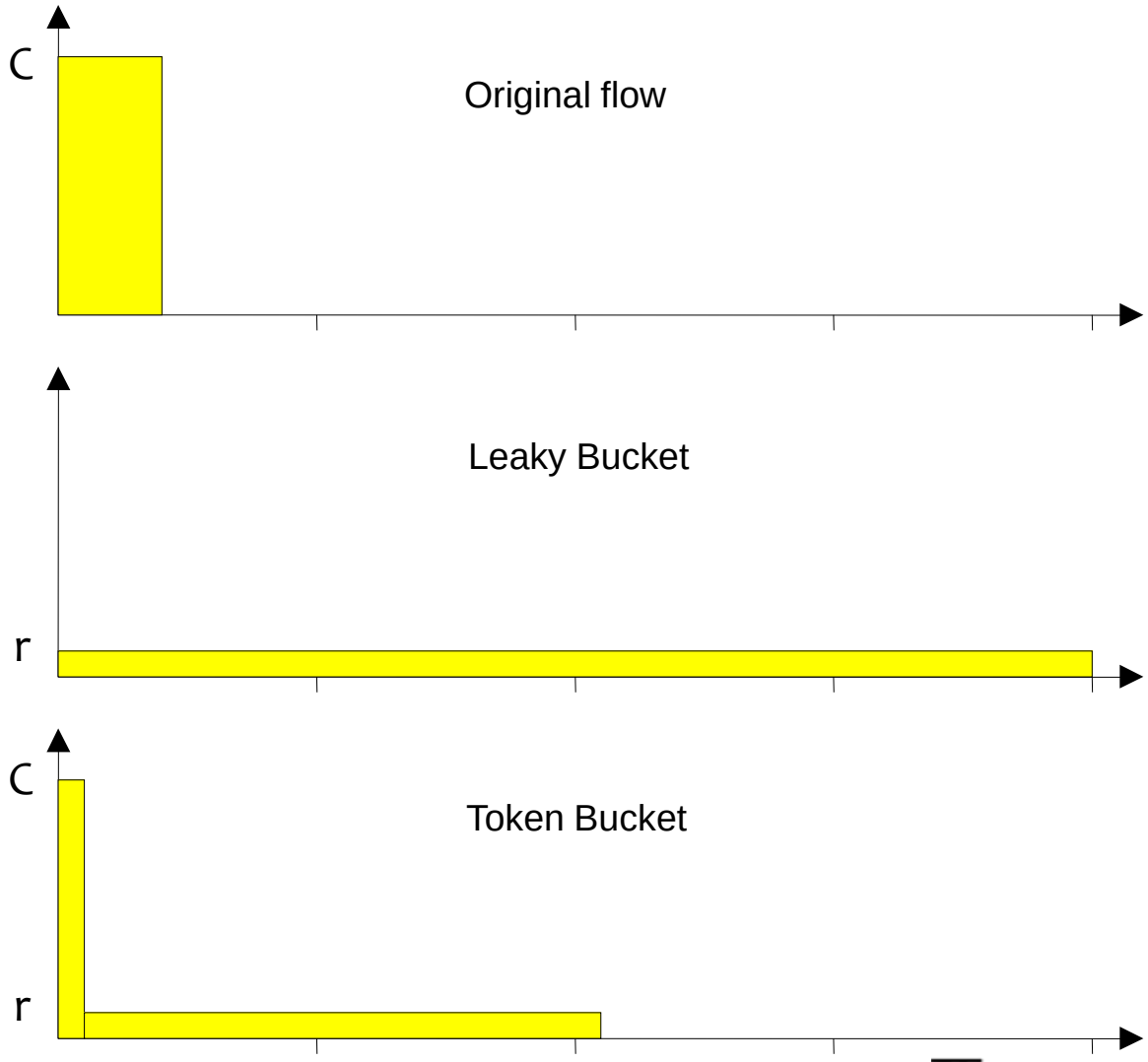
T = 10ms : packet can now be transmitted

(e)

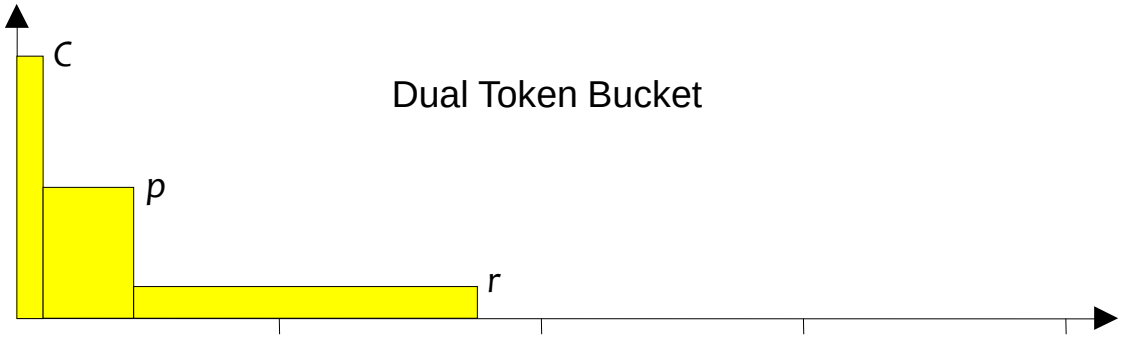


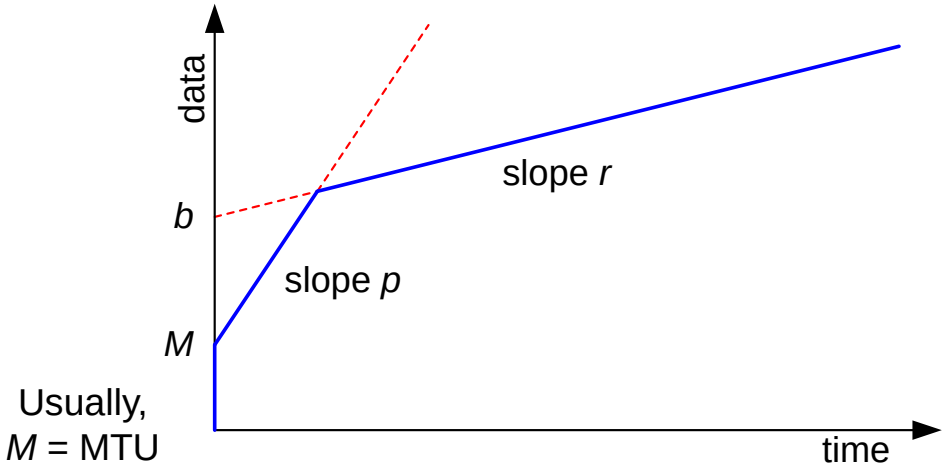
T = 16ms : packet finished transmission

Traffic shape



Traffic shape (contd.)

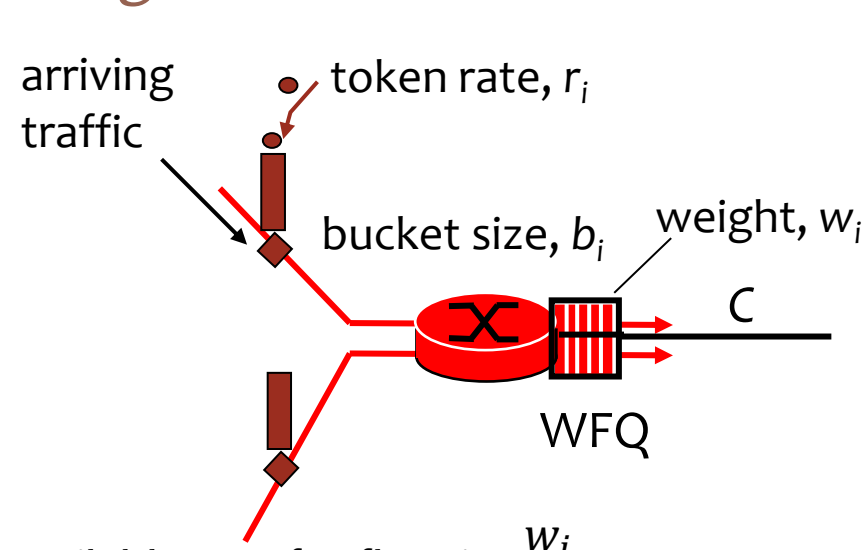


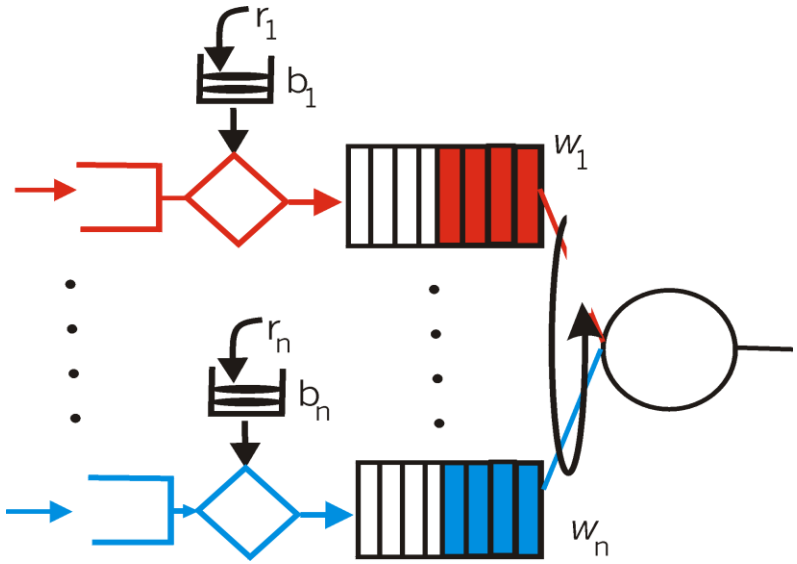


- Equivalent to two token buckets in series: (r,b) and (M,p)
- Blue curve is upper bound starting from **any** instant

Token Bucket + WFQ → QoS Guarantee

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



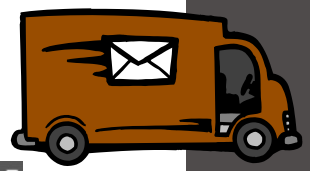


available rate for flow *i*: $\frac{w_i}{\sum_j w_j} C$

max delay for in-profile packets of flow *i*: $\frac{b_i}{C \cdot w_i / \sum_j w_j} + \frac{L_{\max}}{C}$

IETF Differentiated Services [RFC2475]

- want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver
- *scalability:*
  - simple functions in network core
  - more complex functions at edge routers / hosts
    - signaling, maintaining per-flow router state difficult with large number of flows
- does not define service classes, but provides functional components to build service classes



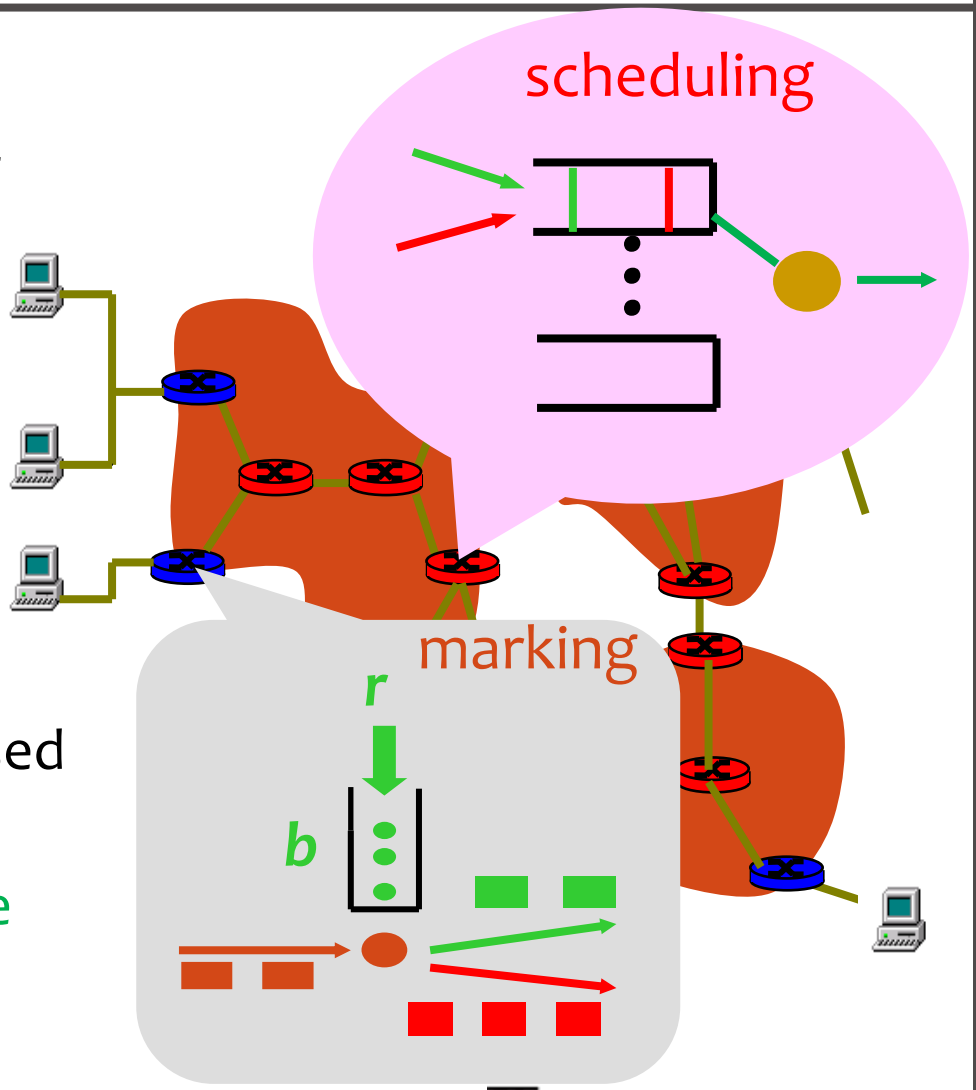
Diffserv Architecture

Edge router:

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

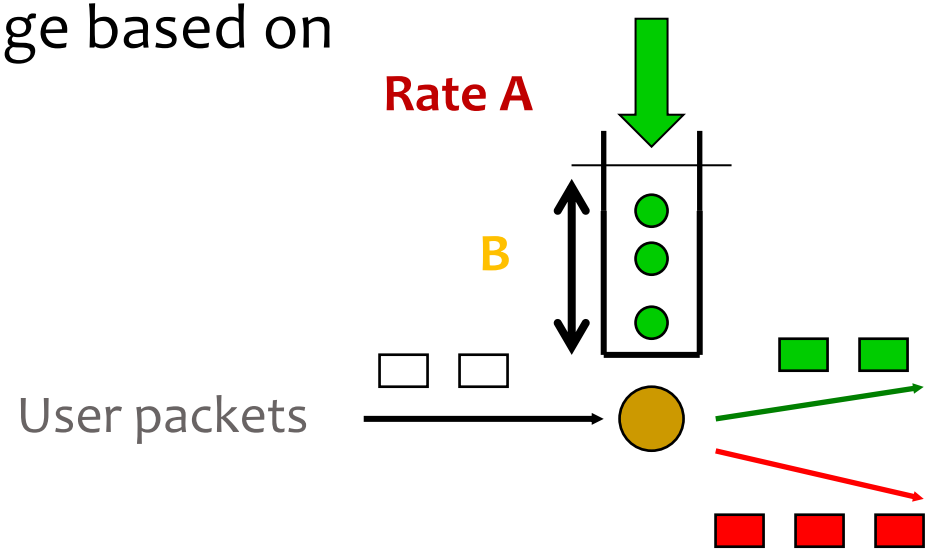
Core router:

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



Edge-router Packet Marking

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



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

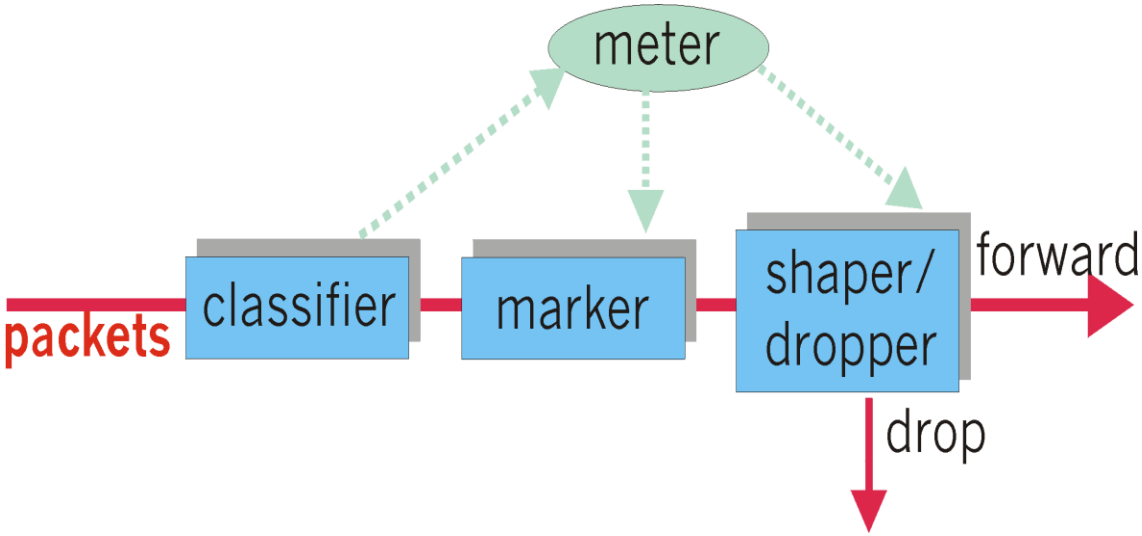
Classification and Conditioning

- Packet is marked in the Type of Service (ToS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Services Code Point (DSCP) and determine Per Hop Behaviour (PHB) that the packet will receive
- 2 bits "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 – Per Hop Behaviour
- PHB results in a different observable (measurable) forwarding performance behaviour
- PHB does not specify what mechanisms to use to ensure required PHB performance behaviour
- 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 developed:

- Expedited Forwarding [RFC3246]: packet departure rate of a class equals or exceeds a specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding [RFC2597]: 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 making the best out of best effort service

7.4 protocols for real-time interactive applications

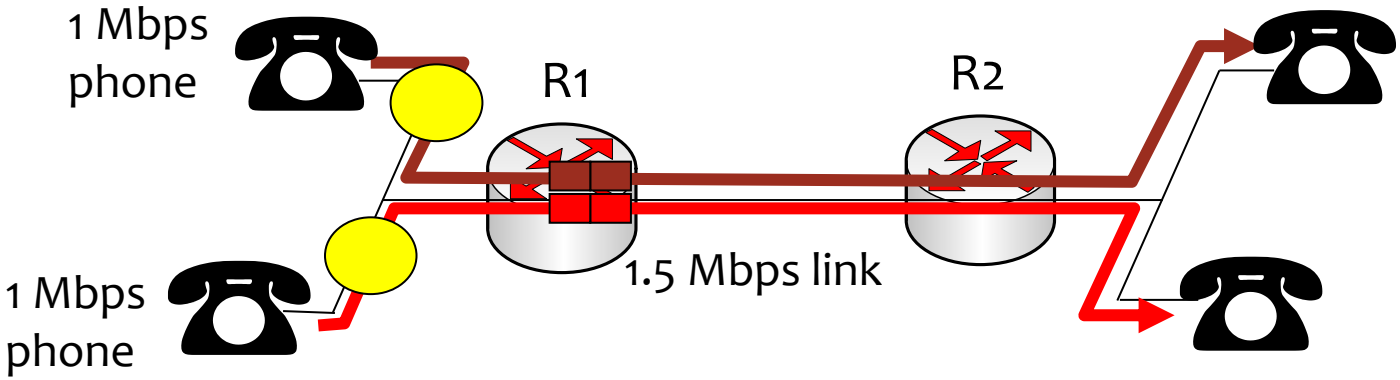
RTP, RTCP, SIP

7.5 providing multiple classes of service

7.6 providing QoS guarantees

Principles for QoS Guarantees (more)

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

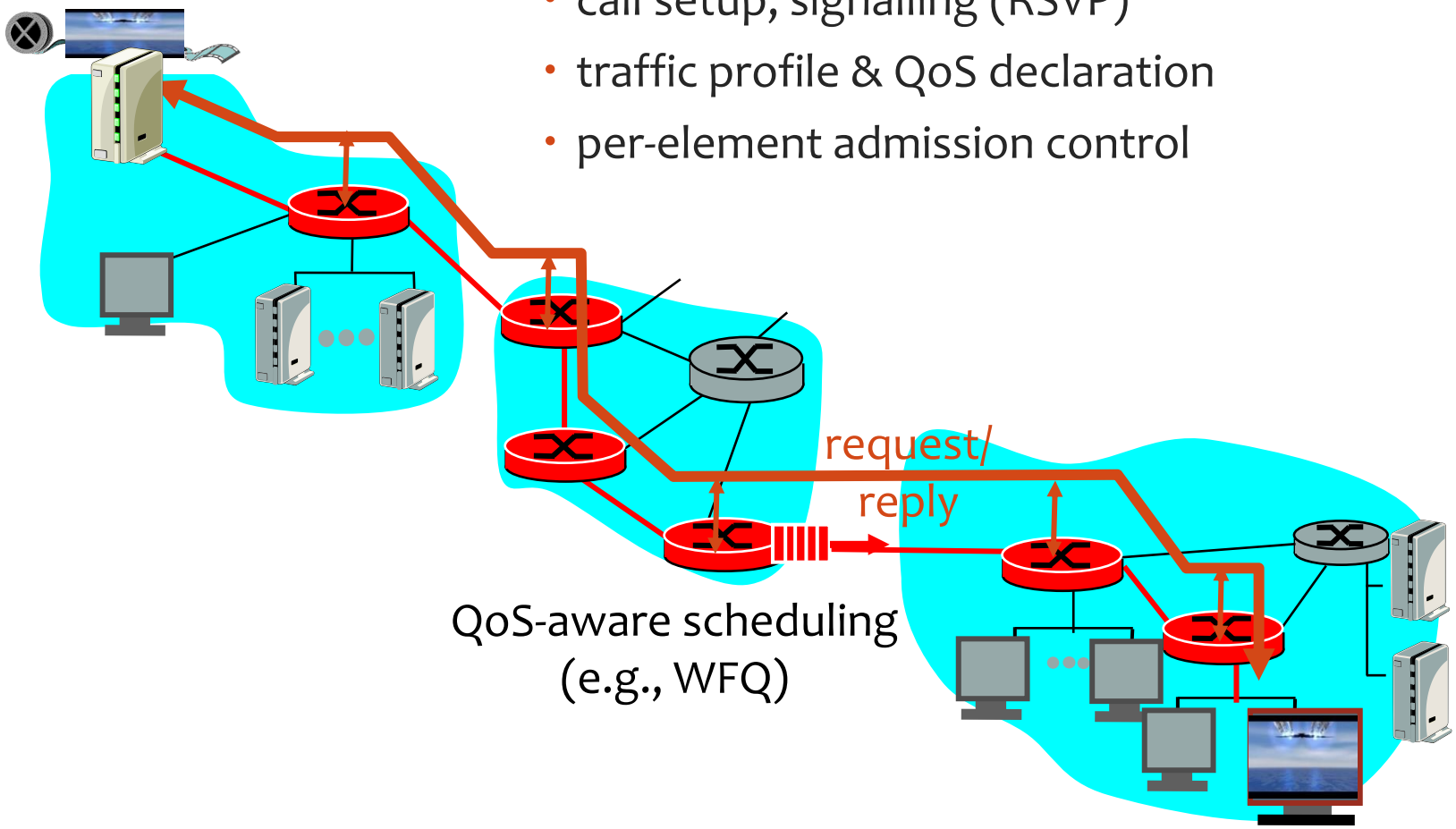


Principle 4

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

QoS guarantee scenario

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



IETF Integrated Services [RFC1633]

- architecture for providing QoS guarantees in IP networks for individual application sessions
- resource reservation — routers maintain state info of allocated resources and QoS requirements
- admit/reject new call setup requests

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

Admission Control

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 profile
- use a signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

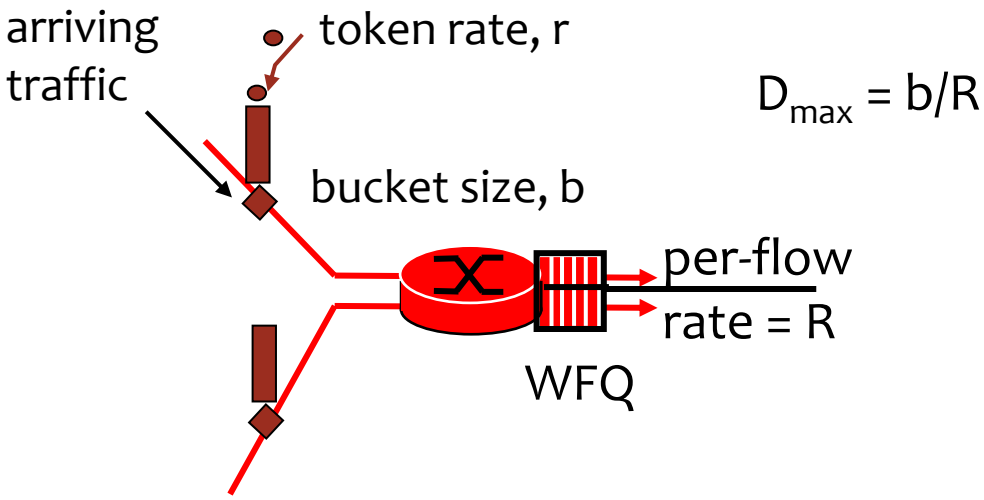
Intserv QoS: Service models

Guaranteed service [RFC2212]:

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

Controlled load service [RFC2211]:

- flows receive a quality of service closely approximating best-effort from that same network element when lightly loaded



Signalling on 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 [RFC2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling!

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, adapting 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

- receivers 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 signalling
  - path error
  - reservation error

IntServ and DiffServ

- Complementary
  - DiffServ: aggregation per client/ user/ user-group/ application, ISP oriented
  - IntServ: per flow, app oriented
- Integration possible
  - IntServ reservation with DiffServ "flows"

	DiffServ	IntServ
Signaling	Network	Application
Granularity	Aggregate (Class)	Flow
Mechanism	Packet Class (other possible)	Dst Addr, protocol, port
Scope	Between networks, E2E	End-to-end

Comments on QoS solutions Usage

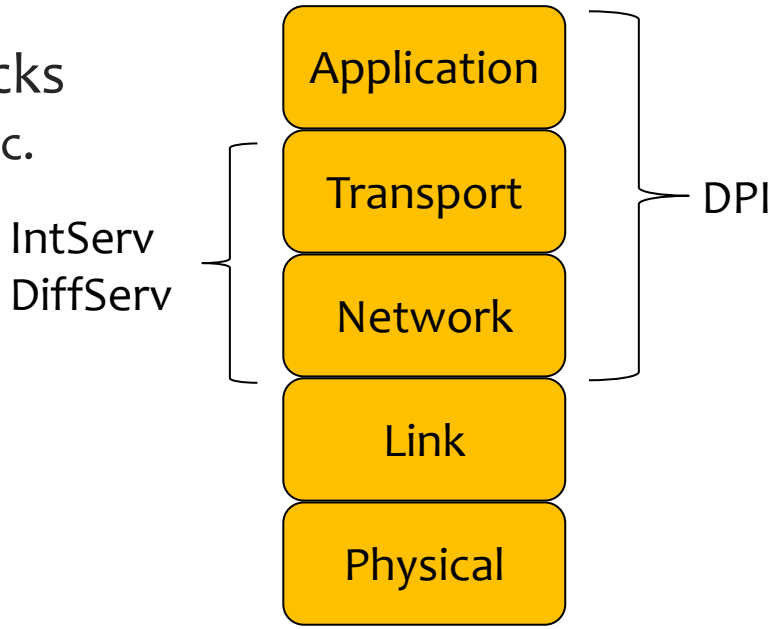
- All ISPs on the packets' path must agree on the E2E commitment
  - So that the selling QoS ISP can provide the paying customer
- Couple with Network dimensioning
  - If there isn't capacity, QoS will not magically work
- Billing procedures
  - Need to bill customers based on traffic volume (and differentiate among the traffic)
- Delay is mostly due to
  - Access rates
  - Router hops

  } Service quality for paying and best-effort clients similar?

Over provisioning and "*bag of tricks*" is winning

Side Note: Deep Packet Inspection

- Uses information up to Application layer
  - Including app data
- Can differentiate based on all information
  - Prioritize, reroute, shape, drop, etc.
- Used by ISPs to:
  - Detect/mitigate security attacks
    - DoS, buffer overflows, virus, etc.
  - Throttle "unwanted" traffic
    - P2P
    - Touches net neutrality
- Hardware implemented
  - Needs to work at wire speed



Chapter 7: Summary

Principles

- classify multimedia applications
- identify network services applications need
- making the best of best effort service

Protocols and Architectures

- specific protocols for best-effort
- mechanisms for providing QoS
- architectures for QoS
  - multiple classes of service
  - QoS guarantees, admission control