SOFTENG 364: Computer Networks

Multimedia Networking

Kurose and Ross, chapter 9

ENGINEERING

Learning Outcomes

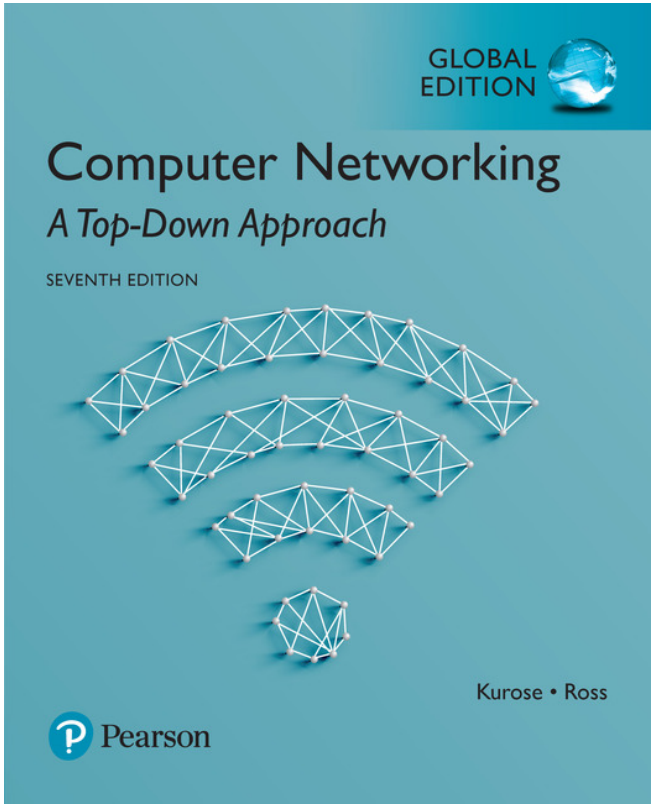
By the end of this module you should be able to:

- Explain how audio and video data is encoded
- List the three multimedia application types
- Explain how streaming video is sent, including mechanisms to reduce jitter
- Describe what Voice-Over-IP is and how it works
- Explain what Real-Time Protocol does
- Explain what Real-Time Control Protocol does
- Explain what Session Initiation Protocol does
- Describe how RTP, RTCP and SIP work together

References

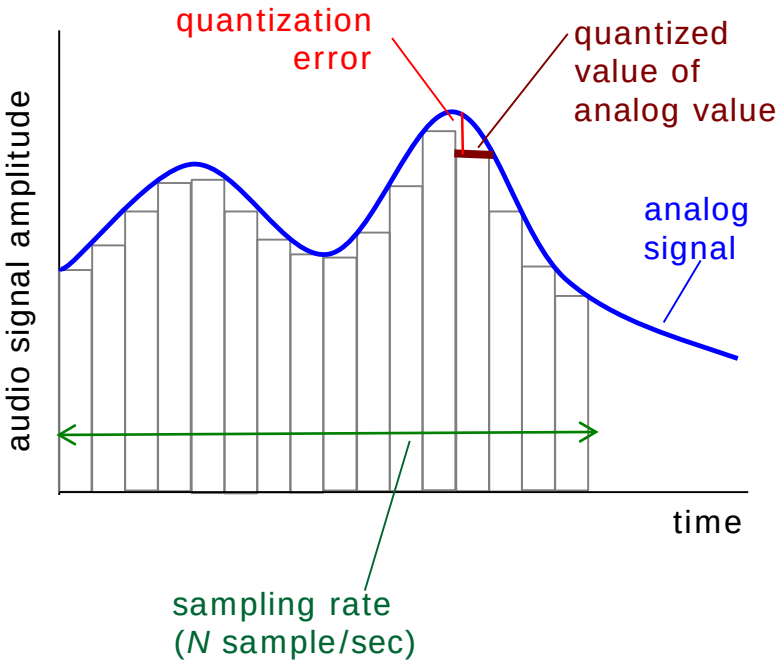
Computer Networking: A Top Down Approach, 7th edition (2016). *By J. Kurose & K. Ross.*

- Chapter 9



Multimedia: Audio

- Analog audio 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, e.g., 8 bits for 256 values

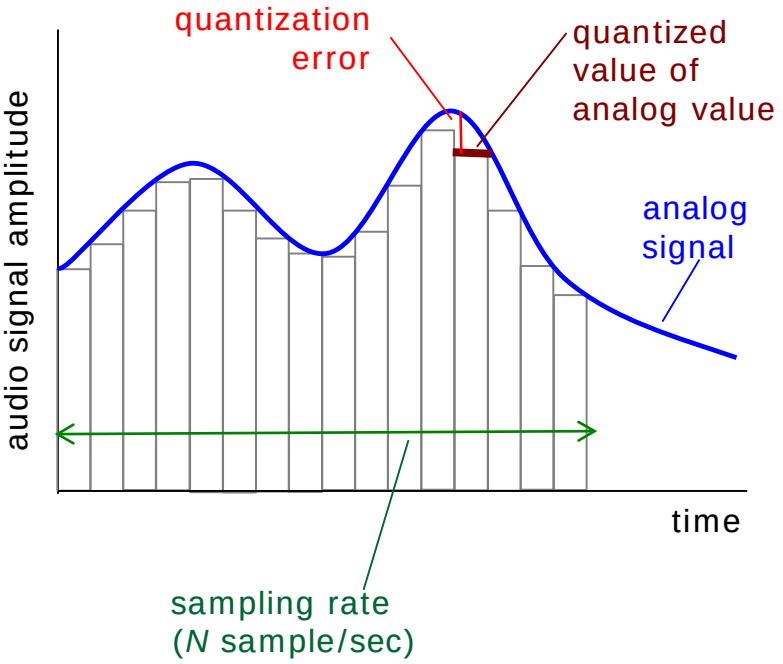


Multimedia: Audio

- Example: 8,000 samples/sec, 256 quantized values: 64,000 bps
- Receiver converts bits back to analog signal:
  - some quality reduction

Example rates

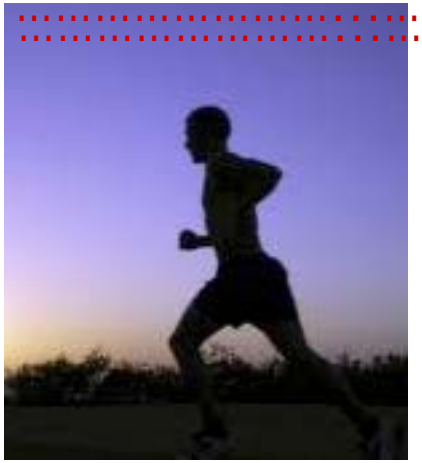
- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet telephony: 5.3 kbps and up



Multimedia: Video

- Video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- Digital image: array of pixels
  - each pixel represented by bits
- Coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - Spatial (within image)
  - Temporal (from one image to next)

Spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple) and number of repeated values (*N)



frame *i*

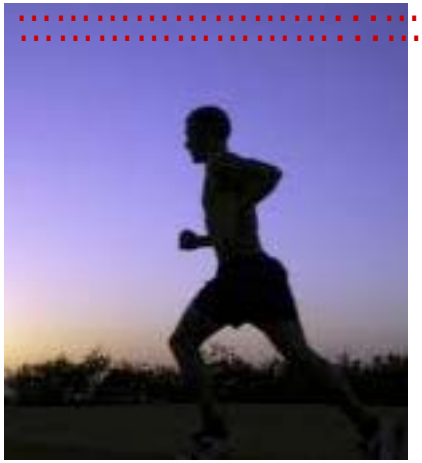
frame *i+1*

Temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Multimedia: Video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

Spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple) and number of repeated values (*N)



frame *i*

Temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

Multimedia networking: 3 application types

Streaming, stored audio, video

- *Streaming:* can begin playout before downloading entire file
- *Stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
- e.g., YouTube, Netflix, Hulu

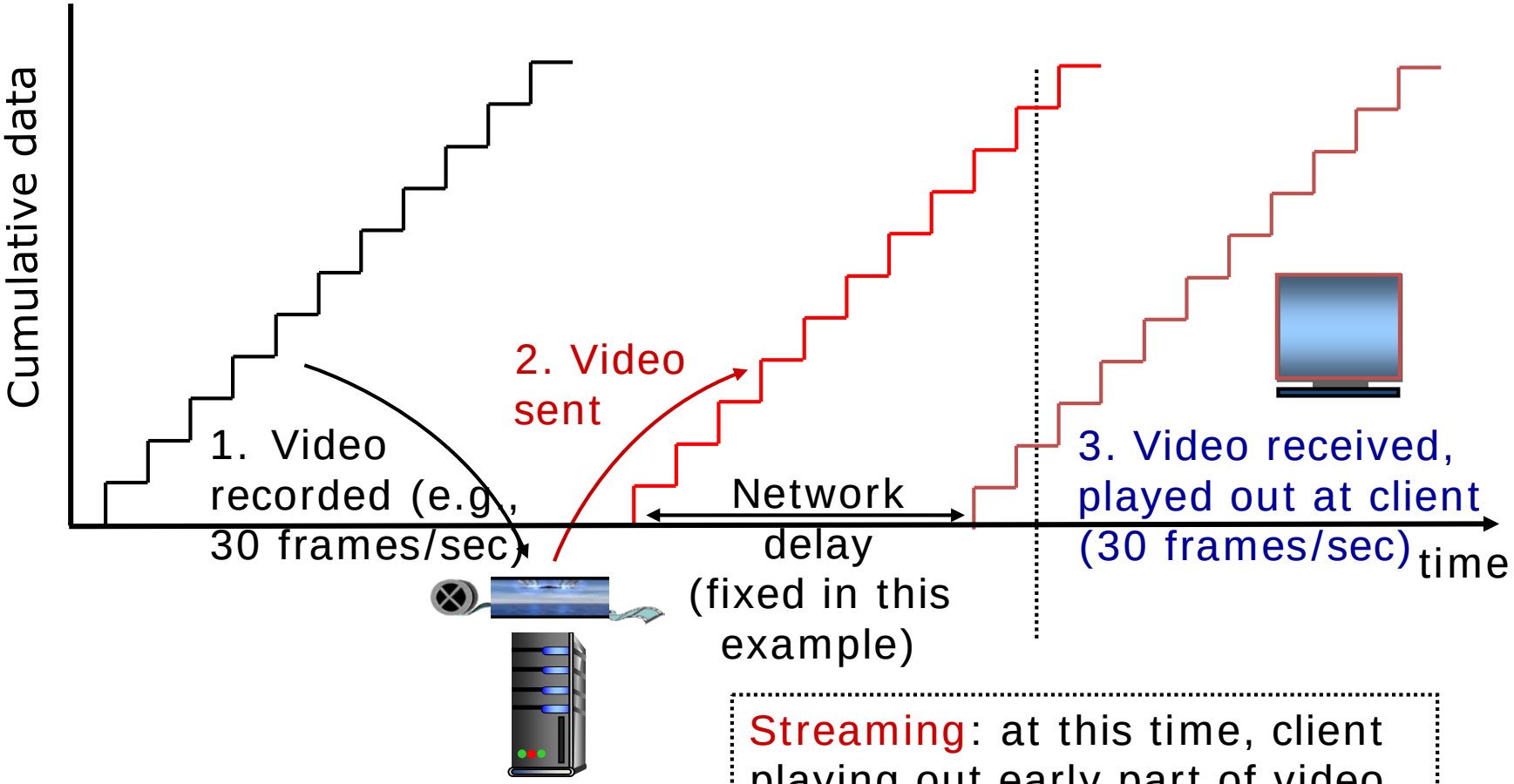
Conversational voice/video over IP

- Interactive nature of human-to-human conversation limits delay tolerance
- E.g., Skype

Streaming live audio, video

- E.g., live sporting event (futbol)

Streaming stored video

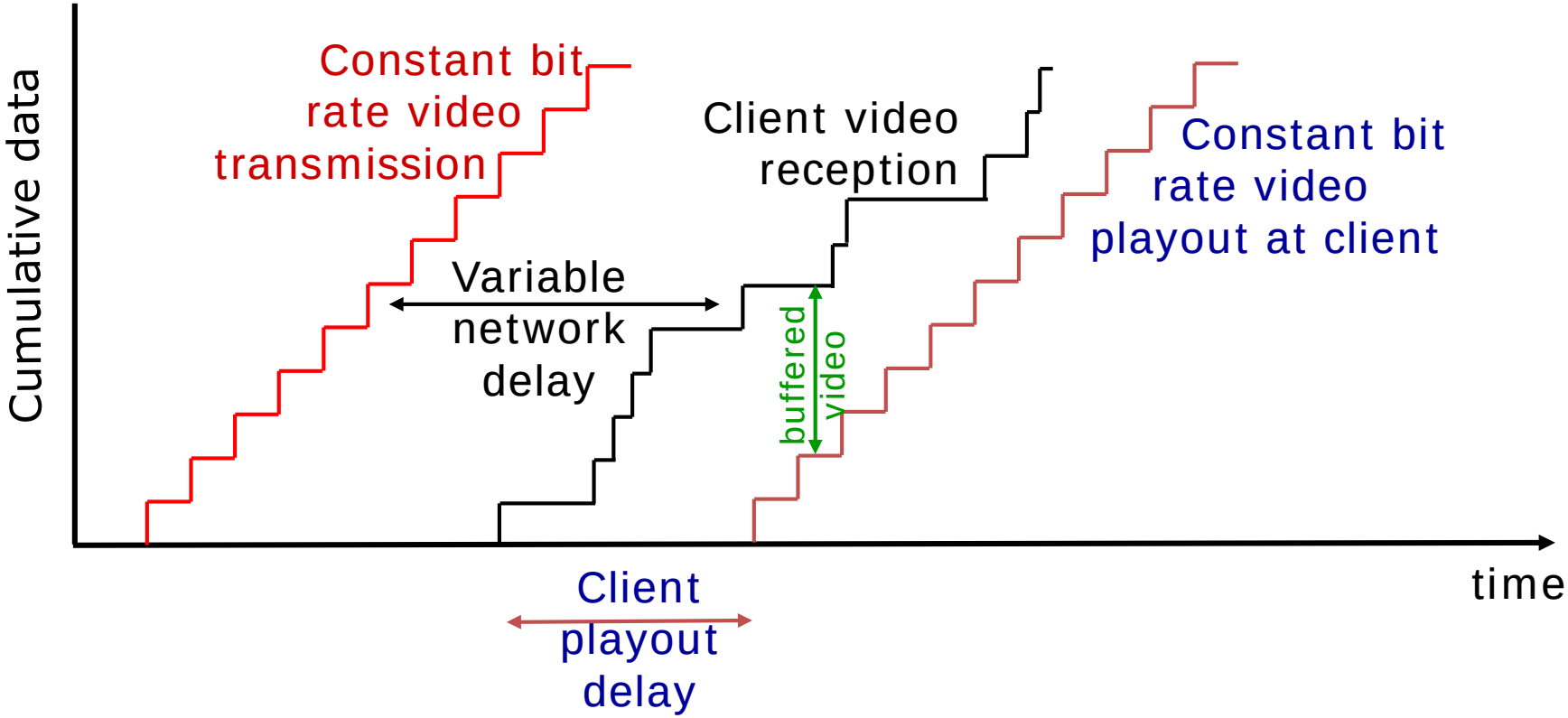


Streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming stored video: Challenges

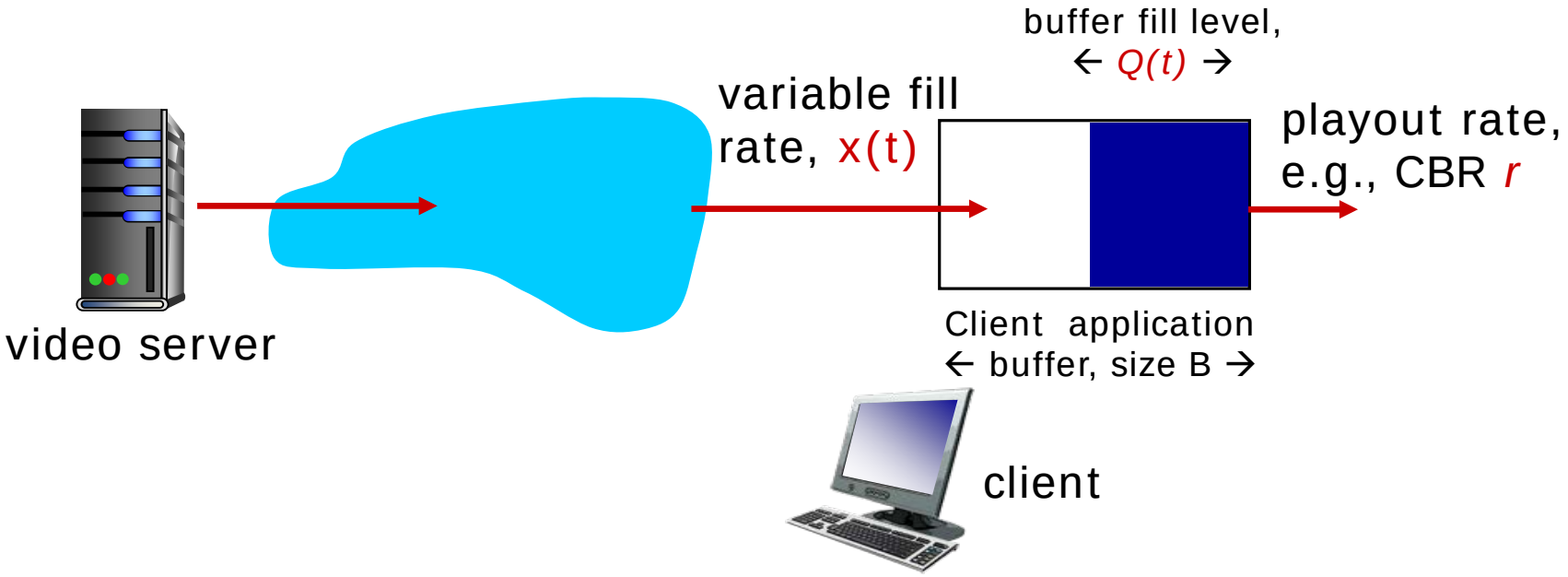
- Continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- Other challenges:
  - Client interactivity: pause, fast-forward, rewind, jump through video
  - Video packets may be lost, retransmitted

Streaming stored video: Revisited

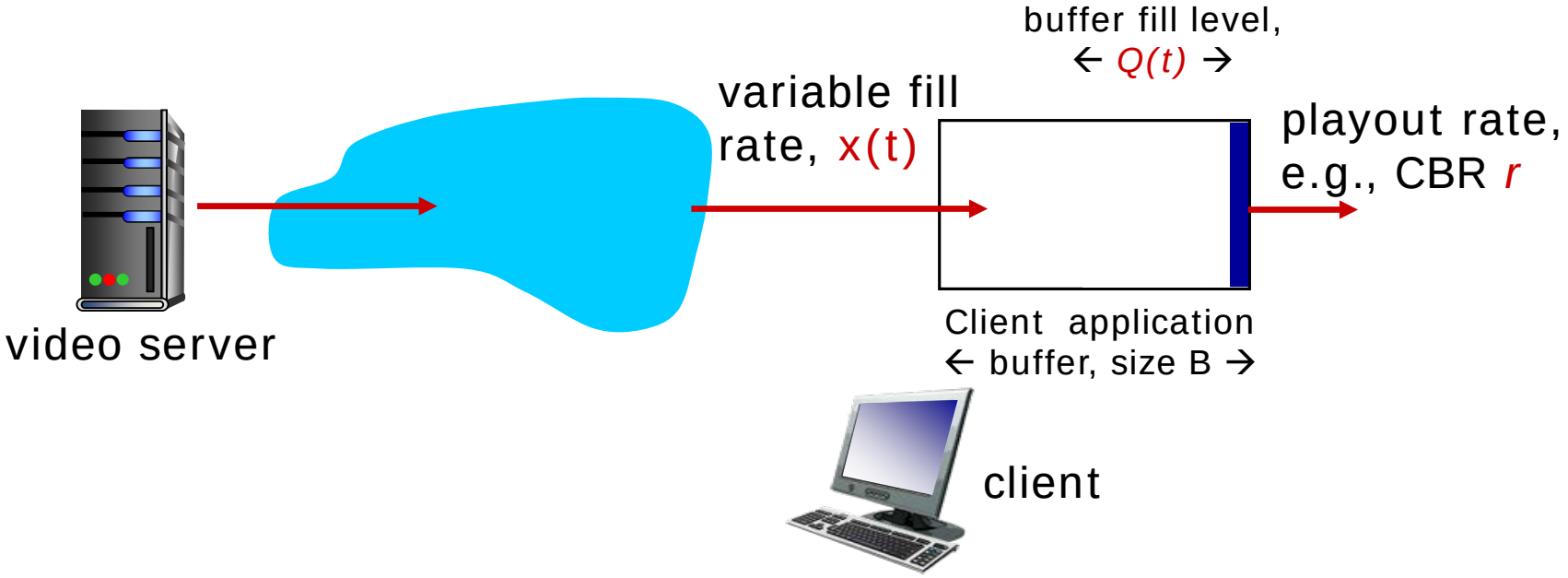


- *Client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Client-side buffering, playout

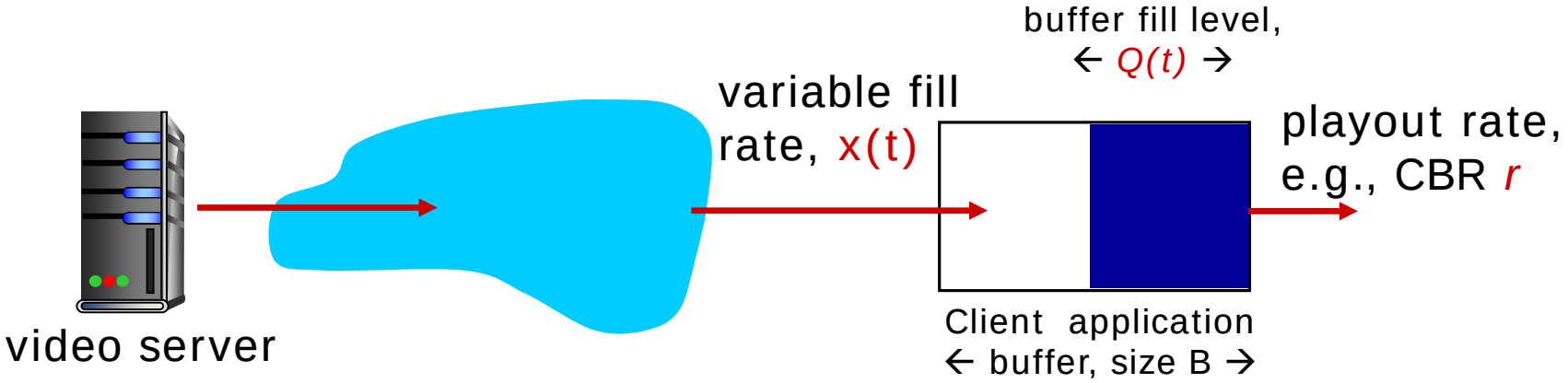


Client-side buffering, playout



1. Initial fill of buffer until playout begins at t_p
2. Playout begins at t_p,
3. Buffer fill level varies over time as fill rate x(t) varies and playout rate r is constant

Client-side buffering, playout



Playout buffering: average fill rate ($\overline{x}$), playout rate (r):

- $\overline{x}$ < r: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\overline{x}$ > r: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *Initial playout delay tradeoff:* buffer starvation less likely with larger delay, but larger delay until user begins watching

Streaming Multimedia: UDP

Server sends at rate appropriate for client

- Often: send rate = encoding rate = constant rate
- Transmission rate can be oblivious to congestion levels

Short playout delay (2-5 seconds) to remove network jitter

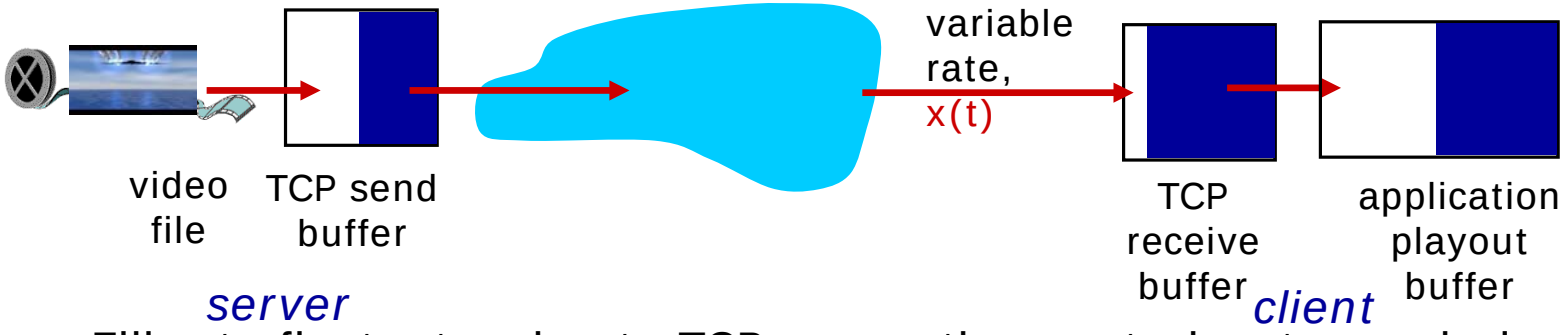
Error recovery: application-level, time permitting

RTP [RFC 2326]: multimedia payload types

UDP may *not* go through firewalls

Streaming Multimedia: HTTP

- Multimedia file retrieved via HTTP GET
- Send at maximum possible rate under TCP



- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- Larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Voice-over-IP (VoIP)

VoIP end-end-delay requirement: needed to maintain "conversational" aspect

- Higher delays noticeable, impair interactivity
- < 150 msec: good
- > 400 msec bad
- Includes application-level (packetization, playout), network delays

Session initialization: how does callee advertise IP address, port number, encoding algorithms?

Value-added services: call forwarding, screening, recording

Emergency services: 911

VoIP Characteristics

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 of data

Application-layer header added to each chunk

Chunk+header encapsulated into UDP or TCP segment

Application sends segment into socket every 20 msec during talkspurt

VoIP: Packet loss, delay

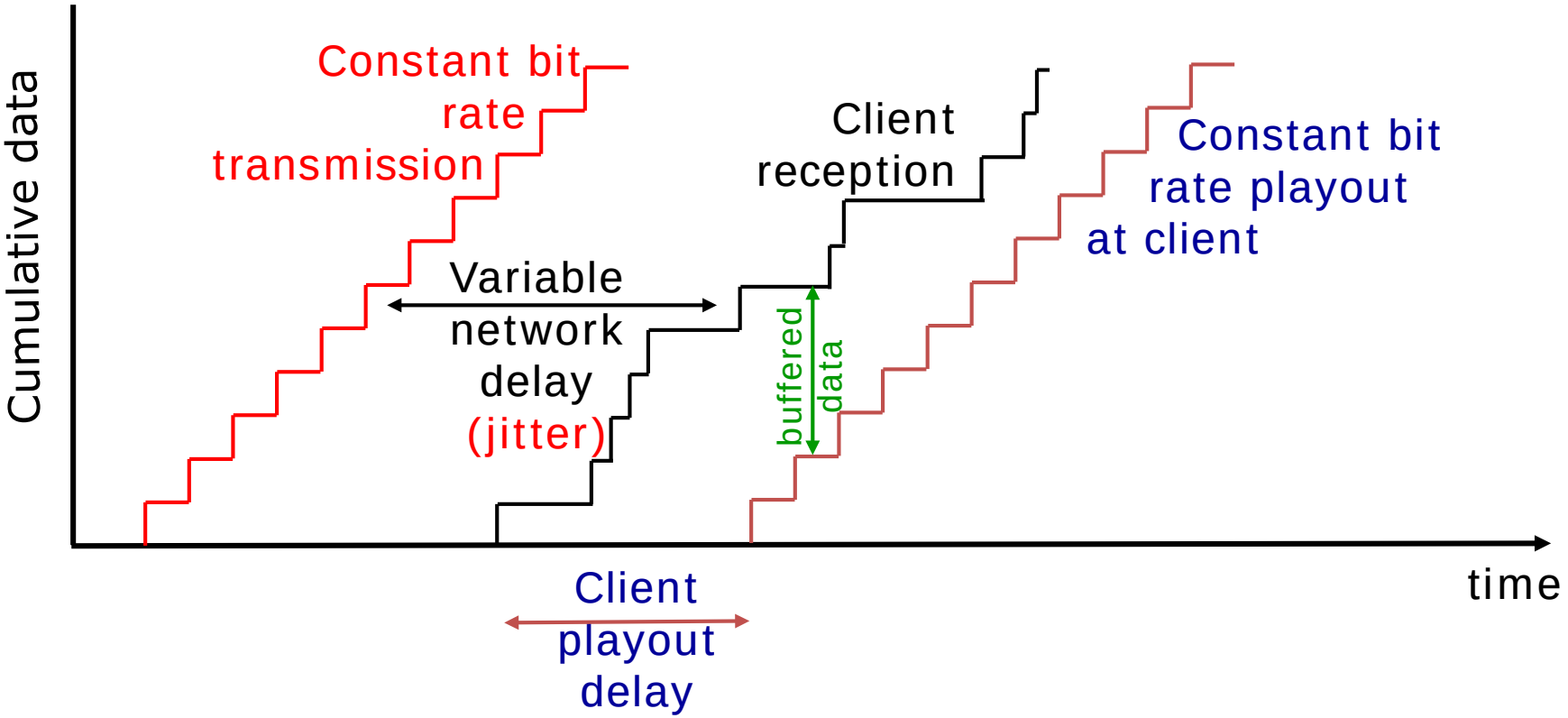
Network loss: IP datagram lost due to network congestion (router buffer overflow)

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, loss concealment, packet loss rates between 1% and 10% can be tolerated

Delay Jitter



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

VoIP: Fixed playout delay

Receiver attempts to playout each chunk exactly *q* msecs after chunk was generated.

- Chunk has time stamp *t:* play out chunk at *t+q*
- Chunk arrives after *t+q*: data arrives too late for playout: data "lost"

Tradeoff in choosing *q*:

- *Large q:* less packet loss
- *Small q:* better interactive experience

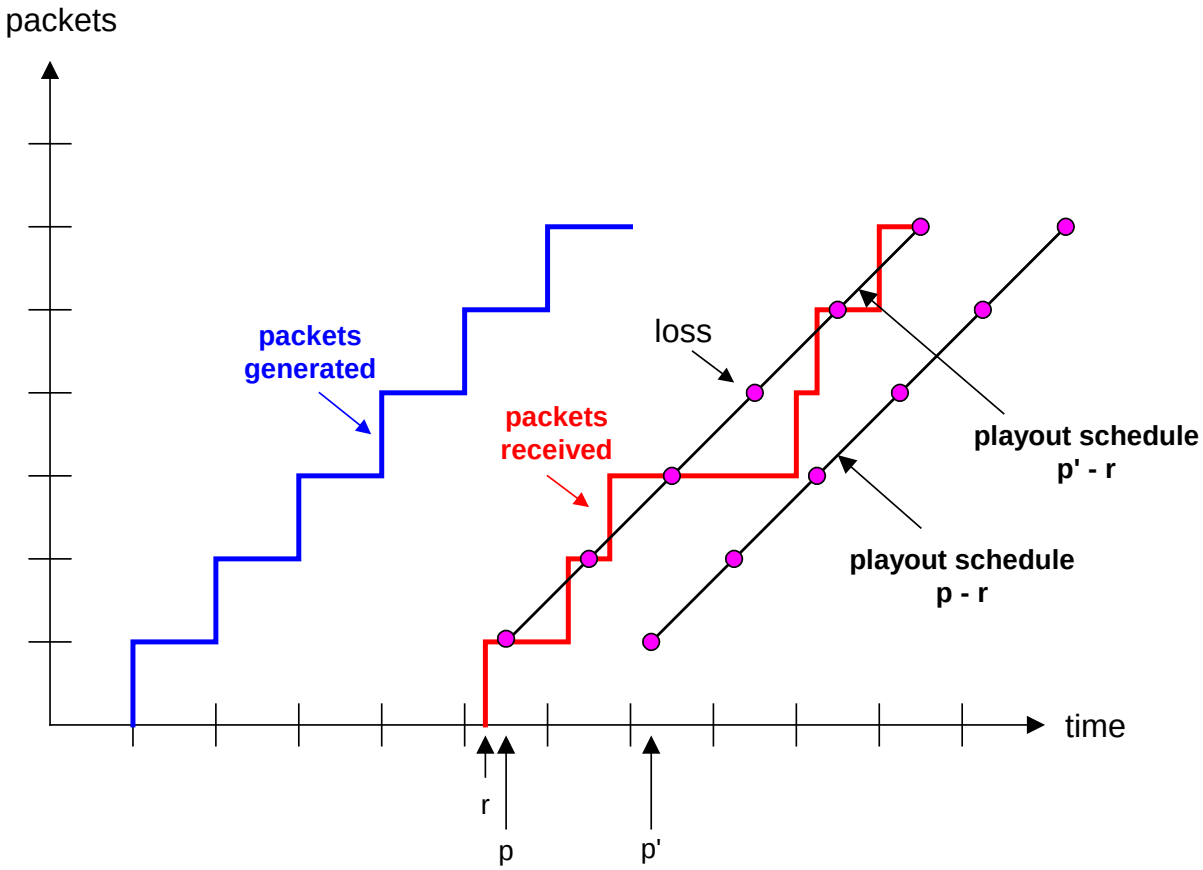
VoIP: Fixed playout delay

Sender generates packets every 20 msec during talk spurt.

First packet received at time r

First playout schedule: begins at p

Second playout schedule: begins at p’



Adaptive playout delay (1)

Goal: low playout delay, low late loss rate

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

Adaptively estimate packet delay: (EWMA - exponentially weighted moving average, recall TCP RTT estimate):

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

- d_i: *delay estimate after ith packet*
- α: *small constant, e.g. 0.1*
- r_i: *time received*
- t_i: *time sent (timestamp)*
- $r_i - t_i$: *measured delay of ith packet*

Adaptive playout delay (2)

Also useful to estimate average deviation of delay, v_i :

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

Estimates d_i, v_i calculated for every received packet, but used only at start of talk spurt

For first packet in talk spurt, playout time is:

$$playout\text{-}time_i = t_i + d_i + Kv_i$$

Remaining packets in talkspurt are played out periodically

Adaptive playout delay (3)

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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.

VoiP: Recovery from packet loss (1)

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

Each ACK/NAK takes ~ one RTT

Alternative: *Forward Error Correction (FEC)*

- send enough bits to allow recovery without retransmission (recall two-dimensional parity in Ch. 5)

Simple FEC

For every group of *n* chunks, create redundant chunk by exclusive OR-ing *n* original chunks

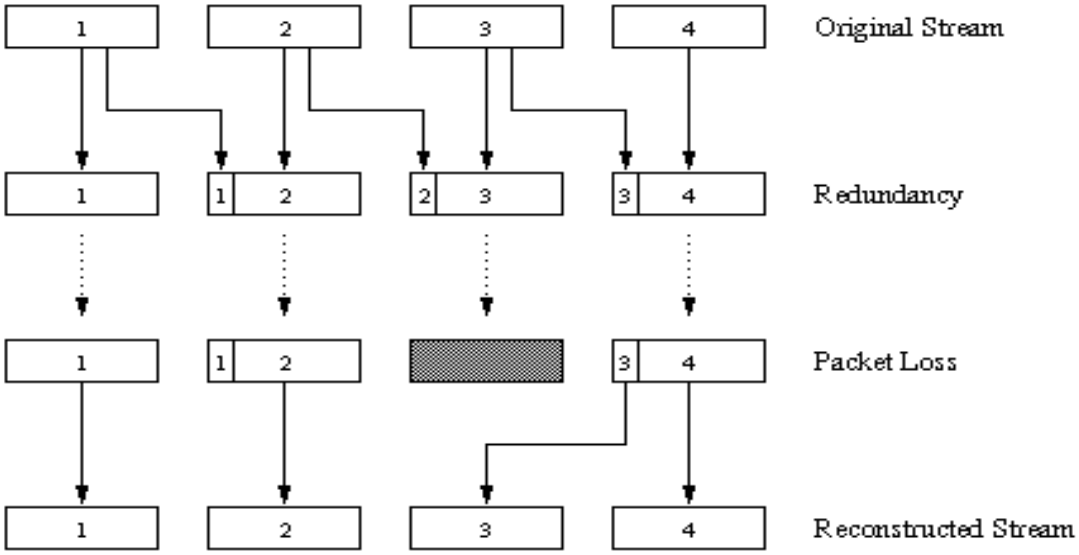
Send *n+1* chunks, increasing bandwidth by factor *1/n*

Can reconstruct original *n* chunks if at most one lost chunk from *n+1* chunks, with playout delay

VoiP: Recovery from packet loss (2)

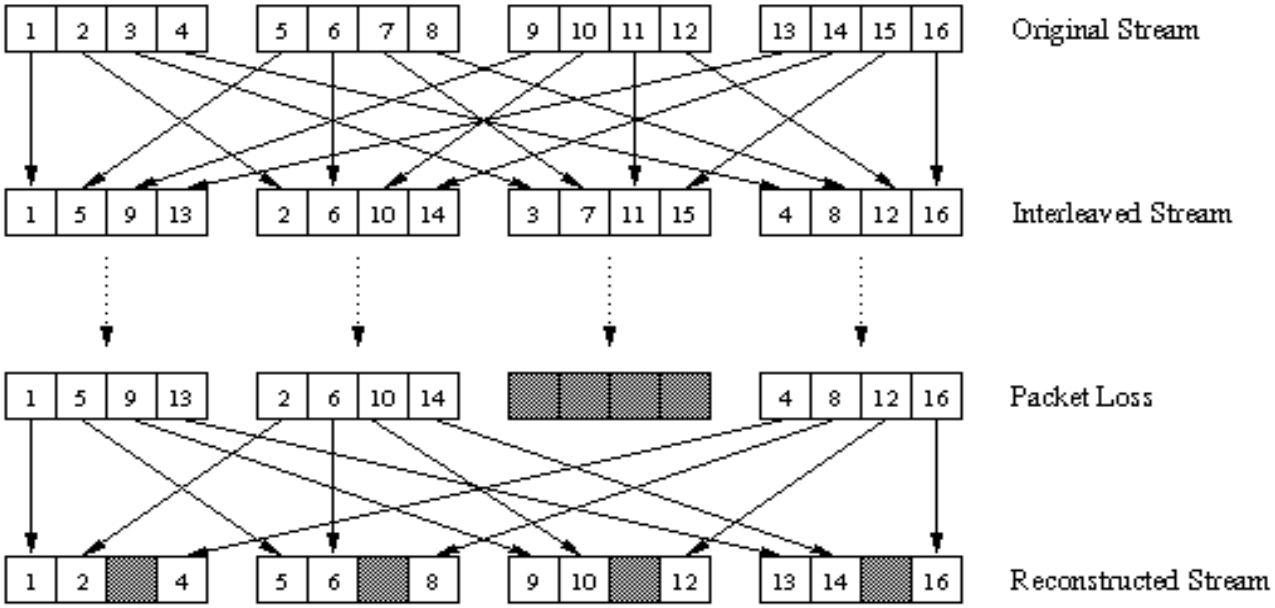
Another FEC scheme:

- "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



- Non-consecutive loss: receiver can conceal loss
- Generalization: can also append (n-1)st and (n-2)nd low-bit rate chunk

VoiP: Recovery from packet loss (3)



Interleaving to conceal loss:

Audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk

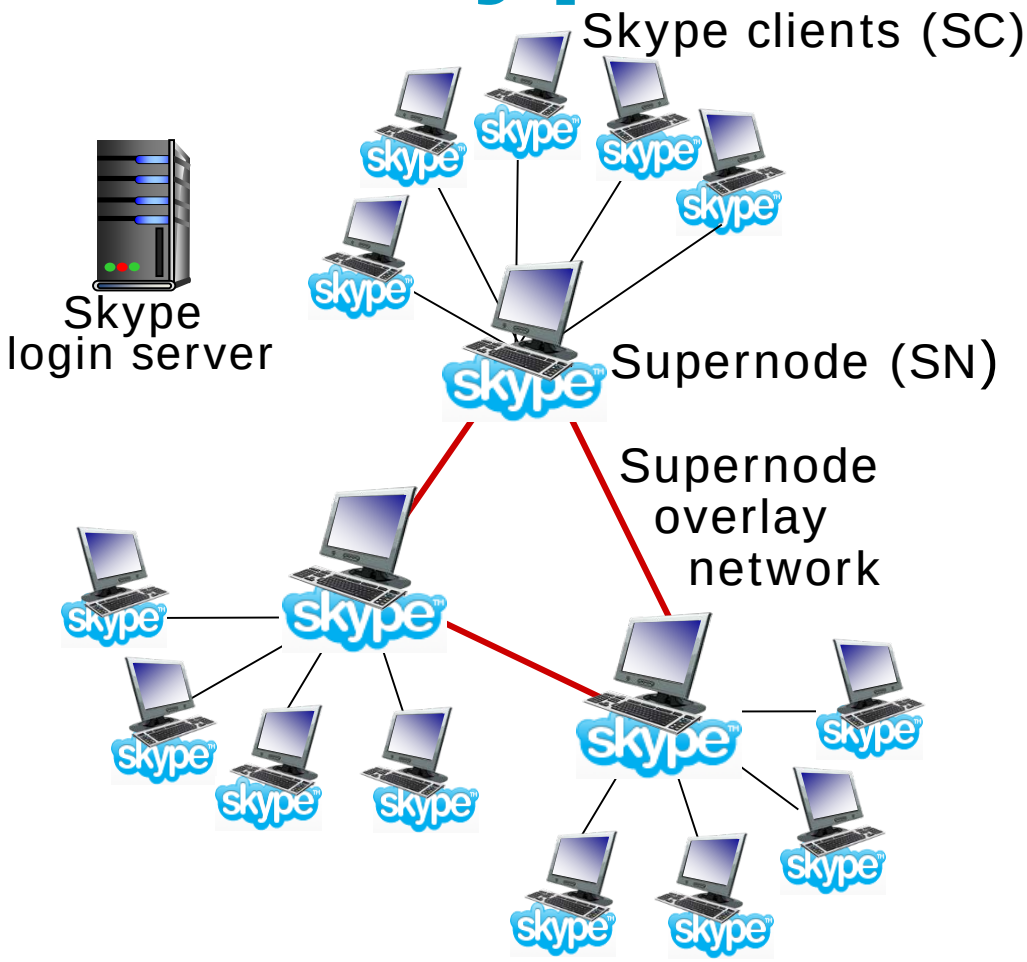
Packet contains small units from different chunks

If packet lost, still have *most* of every original chunk

No redundancy overhead, but increases playout delay

Voice-over-IP: Skype

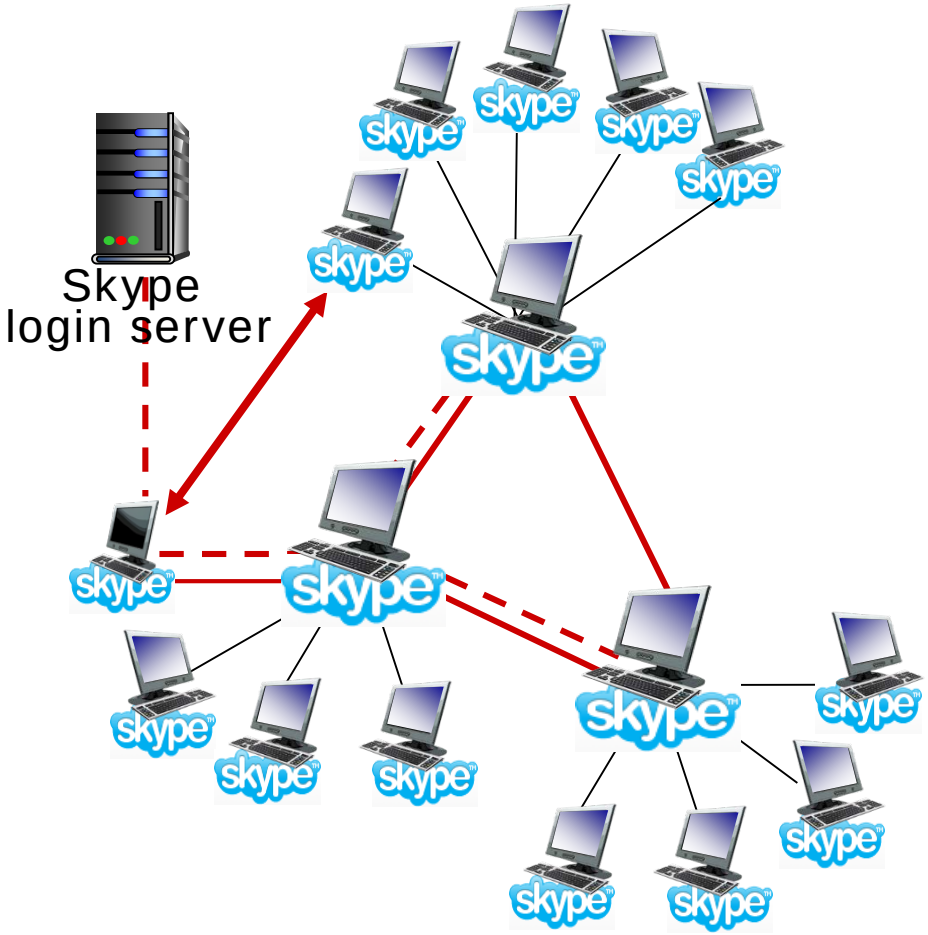
- Proprietary application-layer protocol (inferred via reverse engineering)
  - Encrypted msgs
- P2P components:
  - Clients: Skype peers connect directly to each other for VoIP call
  - Super nodes (SN): Skype peers with special functions
  - Overlay network: among SNs to locate SCs
  - Login server



P2P
Voice-over-ip: Skype

Skype client operation:

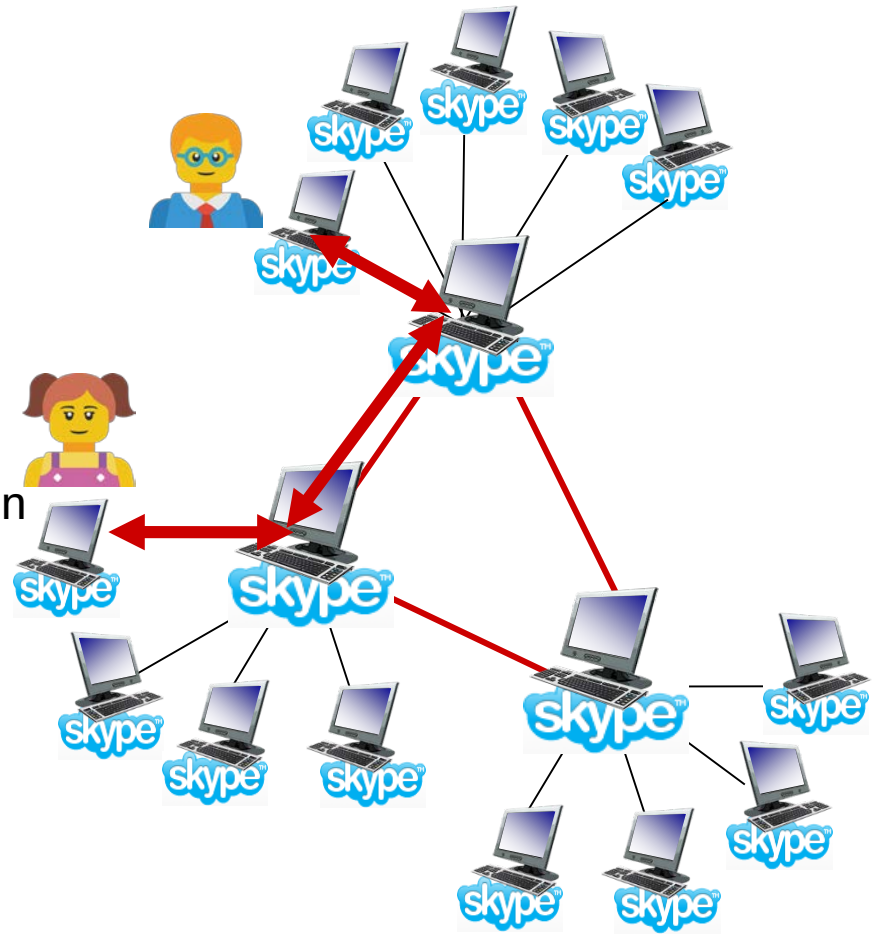
1. joins Skype network by contacting SN (IP address cached) using TCP
2. logs-in (username, password) to centralized Skype login server
3. obtains IP address for callee from SN, SN overlay
   - or client buddy list
4. initiate call directly to callee



Skype: Peers as relays

- *Problem:* both Alice, Bob are behind "NATs"
  - NAT prevents outside peer from initiating connection to insider peer
  - Inside peer *can* initiate connection to outside

- Relay solution: Alice, Bob maintain open connection to their SNs
  - Alice signals her SN to connect to Bob
  - Alice's SN connects to Bob's SN
  - Bob's SN connects to Bob over open connection Bob initially initiated to his SN



Real-Time Protocol (RTP)

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

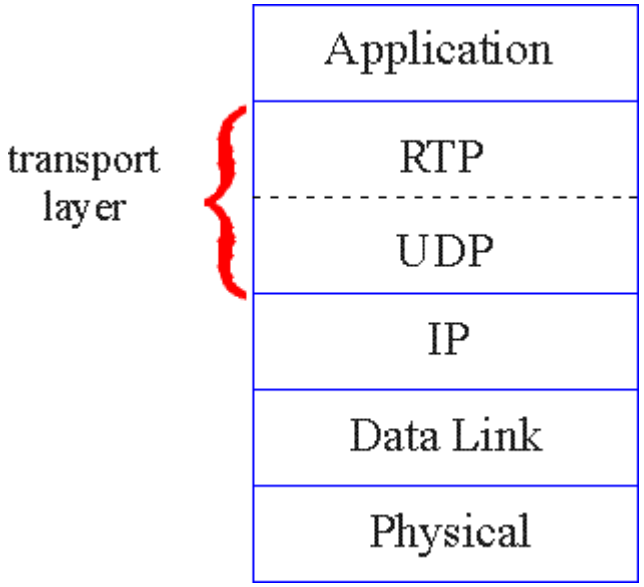
- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping

- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two VoIP applications run RTP, they may be able to work together

RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

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



RTP Example

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, which is encapsulated in UDP segment

- RTP header indicates type of audio encoding in each packet
  - sender can change encoding during conference
- RTP header also contains sequence numbers, timestamps

RTP and QoS

- RTP does *not* provide any mechanism to ensure timely data delivery or other QoS guarantees
- RTP encapsulation 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 timely matter

RTP header

payload type	sequence number type	time stamp	Synchronization Source ID	Miscellaneous fields

payload type (7 bits): indicates type of encoding currently being used. If sender changes encoding during call, sender informs receiver via payload type field

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

sequence # (16 bits): increment by one for each RTP packet sent

- detect packet loss, restore packet sequence

RTP header (cont.)

payload type	sequence number type	time stamp	Synchronization Source ID	Miscellaneous fields

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

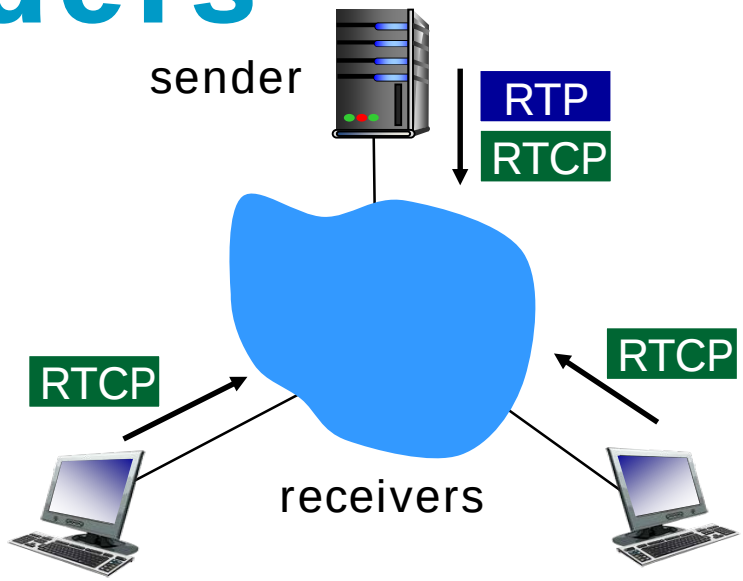
- *SSRC field (32 bits long):* identifies source of RTP stream. Each stream in RTP session has distinct SSRC

Real-Time Control Protocol (RTCP)

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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

RTCP: multiple multicast senders



- each RTP session: typically a single multicast address; all RTP /RTCP packets belonging to session use multicast address
- RTP, RTCP packets distinguished from each other via distinct port numbers
- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases

RTCP: packet types

receiver report packets:

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

sender report packets:

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

source description packets:

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

RTCP: stream synchronization

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

RTCP: bandwidth scaling

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

example : one sender, sending video at 2 Mbps

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

Have you seen this?

Craig Sutherland is inviting you to a scheduled Zoom meeting.

Topic: SOFTENG 364 Lectures
Time: This is a recurring meeting Meet anytime

…

Join by SIP
92879143171@130.216.15.174
92879143171@130.216.15.175

Join by H.323
130.216.15.174
130.216.15.175
Meeting ID: 928 7914 3171
Password: 016299

SIP: Session Initiation Protocol

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Long-term vision:

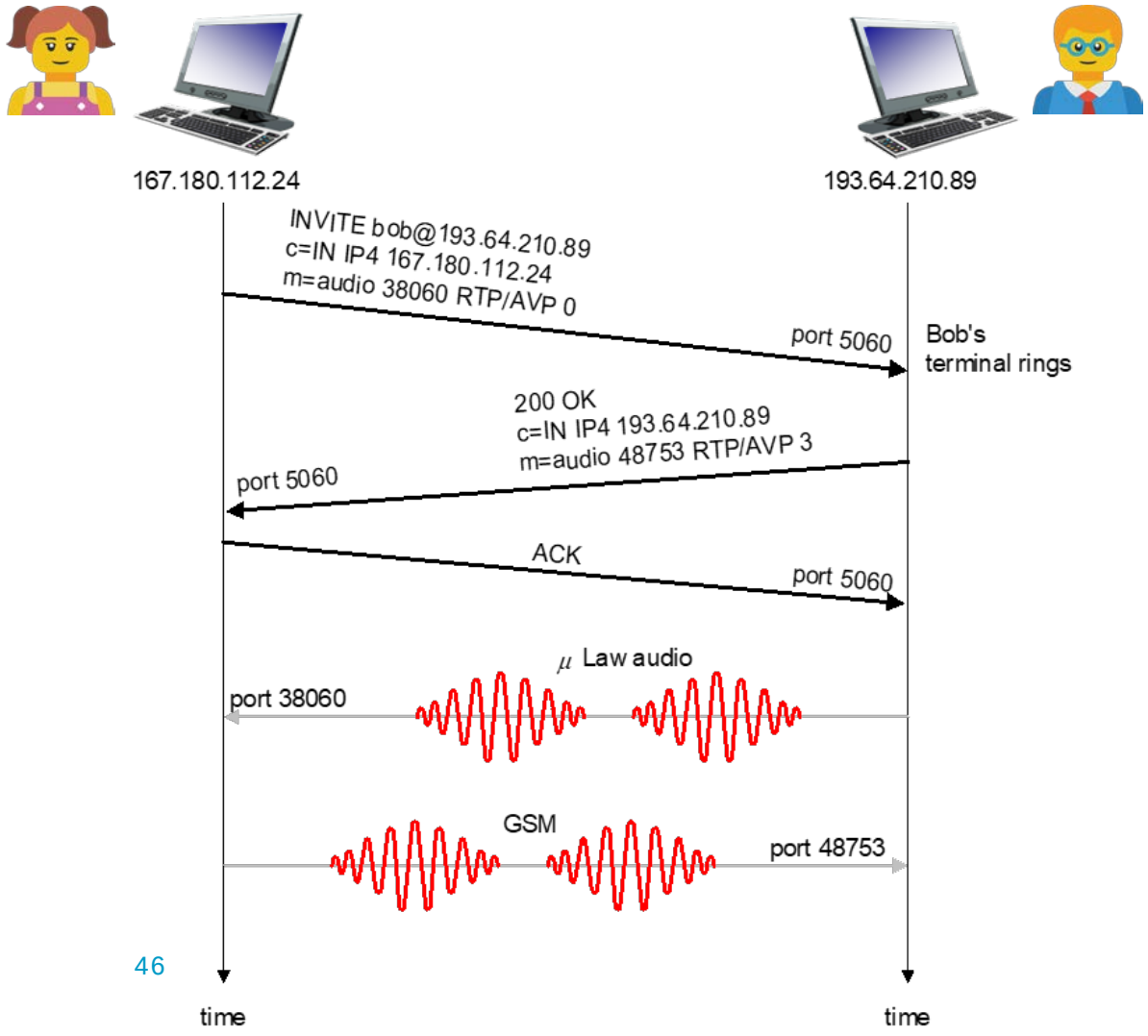
- all telephone calls, video conference calls take place over Internet
- people identified by names or e-mail addresses, rather than by phone numbers
- can reach callee *(if callee so desires)*, no matter where callee roams, no matter what IP device callee is currently using

RFC3261

SIP services

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

Example: setting up call to known IP address



- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive (PCM mlaw)
- 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 mlaw encoder
  - Bob will instead reply with 606 Not Acceptable Reply, listing his encoders. Alice can then send new INVITE message, advertising different encoder
- Rejecting a call
  - Bob can reject with replies "busy," "gone," "payment required," "forbidden"
- Media can be sent over RTP or some other protocol

Example of SIP message

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

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

Notes:

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

- Here we don't know Bob's IP address
  - intermediate SIP servers needed

- Alice sends, receives SIP messages using SIP default port 506
- Alice specifies in header that SIP client sends, receives SIP messages over UDP

Name translation, user location

- caller wants to call callee, but only has callee's name or e-mail address.
- need to get IP address of callee's current host:
  - user moves around
  - DHCP protocol
  - user has different IP devices (PC, 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)

SIP registrar

- one function of SIP server: registrar
- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server

register message:

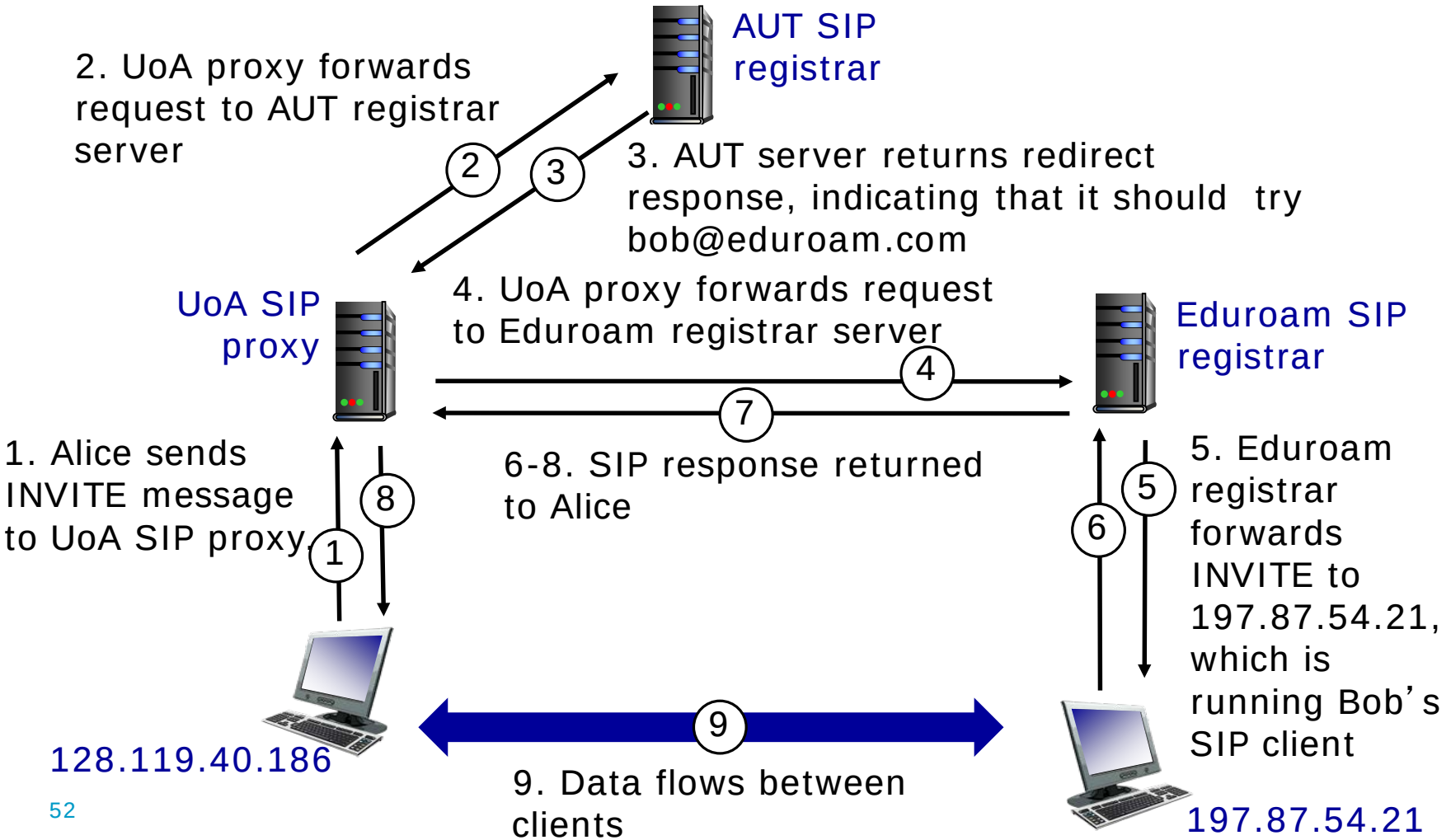
```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

SIP proxy

- another function of SIP server: *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
- Bob sends response back through same set of SIP proxies
- proxy returns Bob's SIP response message to Alice
  - contains Bob's IP address
- SIP proxy analogous to local DNS server plus TCP setup

SIP example

alice@uoa.ac.nz calls bob@aut.ac.nz



Comparison with H.323

- H.323: another signaling protocol for real-time, interactive multimedia
- H.323: complete, vertically integrated suite of protocols for multimedia conferencing: signaling, registration, admission control, transport, codecs
- SIP: single component. Works with RTP, but does not mandate it. Can be combined with other protocols, services

- H.323 comes from the ITU (telephony)
- SIP comes from IETF: borrows much of its concepts from HTTP
  - SIP has Web flavor; H.323 has telephony flavor
- SIP uses KISS principle: Keep It Simple Stupid

Summary

Described how to store audio and video (image) data

Explained three ways to transmit multimedia data

Introduced streaming video data and how client-side buffering reduces jitter

Explored Voice-over-IP and some of the challenges involved:

- Handling delay jitter
- Handling packet loss

Used Skype as a VoIP example

Looked at three more recent protocols for streaming audio and video data:

- Real-Time Protocol: conversational media streaming
- Real-Time Control Protocol: RTP performance control
- Session Initiation Protocol: services to support call setup, address resolution and call management

ENGINEERING