NSCOM01

Other Application Protocols

VOICE OVER IP

❑ VoIP

- End-to-end delay requirement: <150ms; more than 400ms is noticeable
  - 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

VOIP

❑ Characteristics

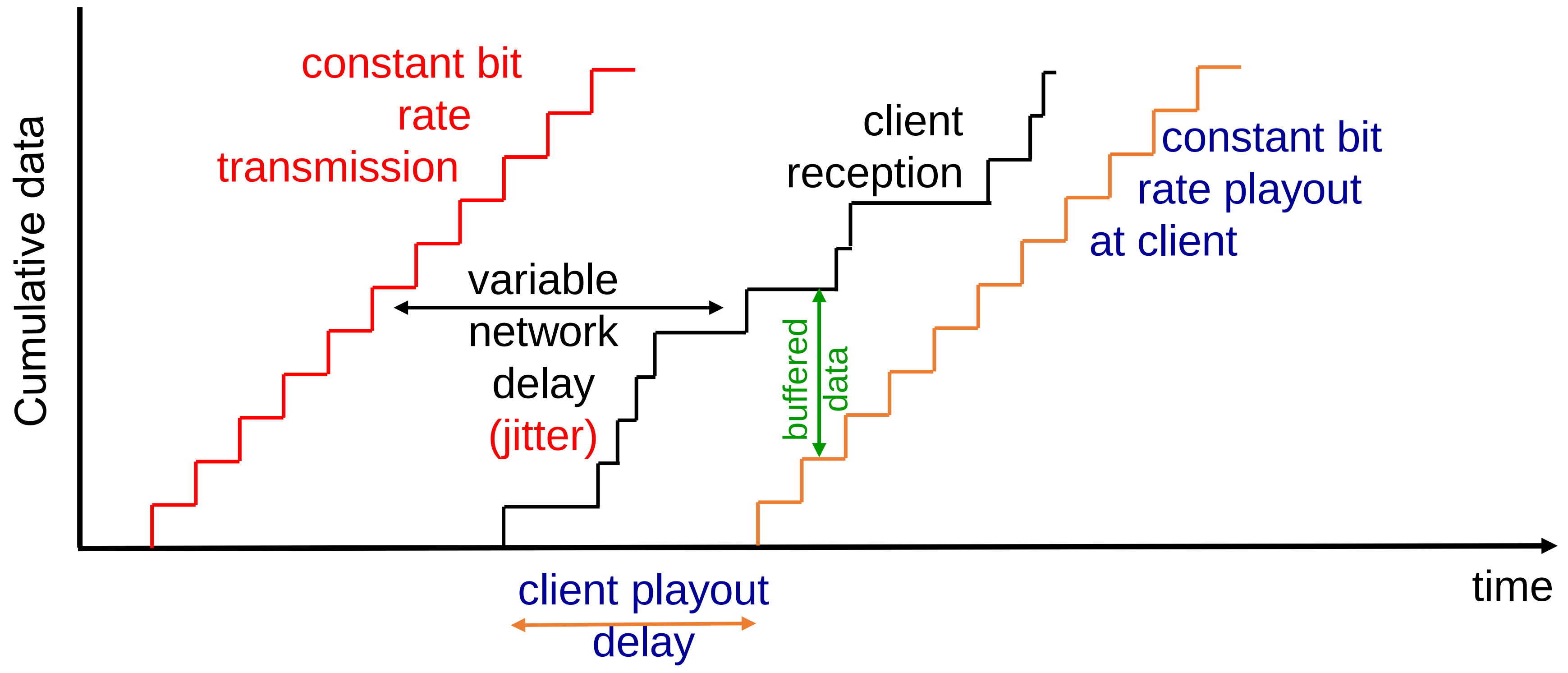
- speaker's audio: alternating talk spurts, silent periods.
  - 64 kbps during talk spurt
  - pkts generated only during talk spurts
  - 20 msec chunks at 8 Kbytes/sec: 160 bytes of data
- application-layer header added to each chunk
- chunk+header encapsulated into UDP or TCP segment
- application sends 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



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

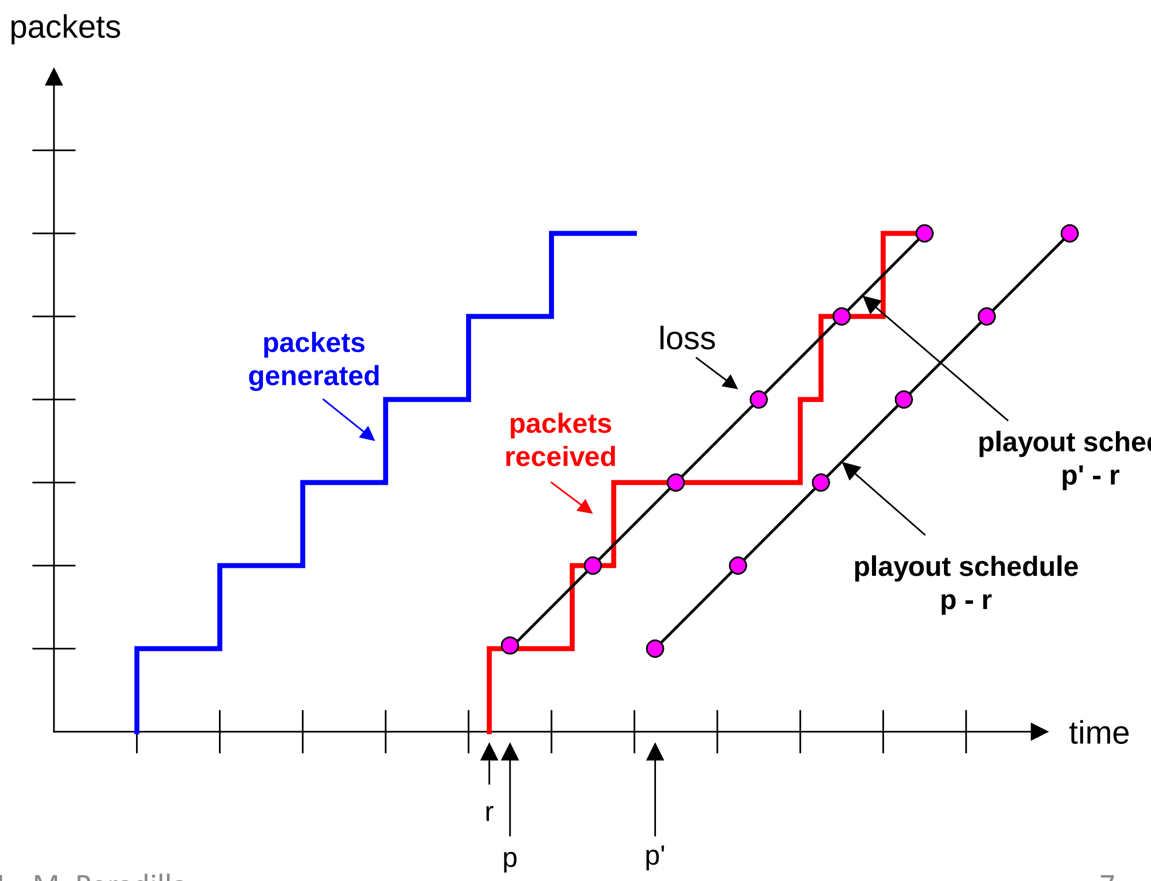
- ❑ **end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)**

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

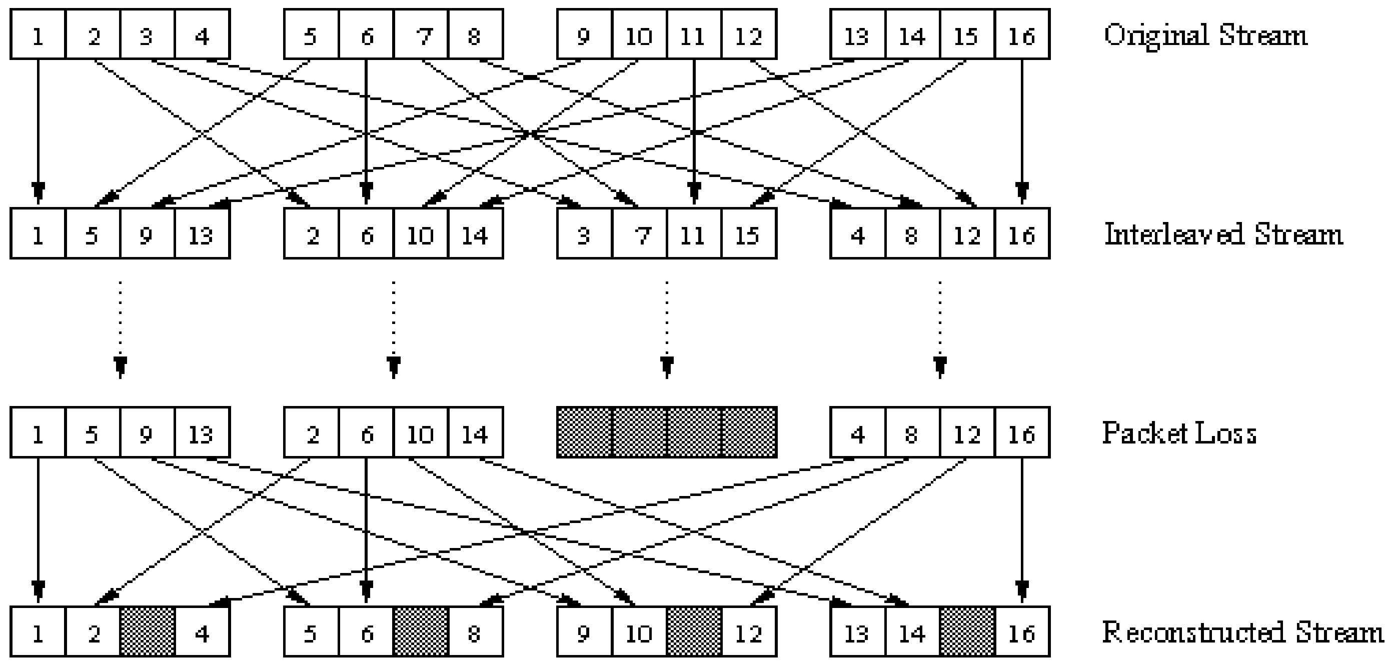
VOIP: FIXED PLAYOUT DELAY

- 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'



VOIP: RECOVERY FROM PACKET LOSS

- **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

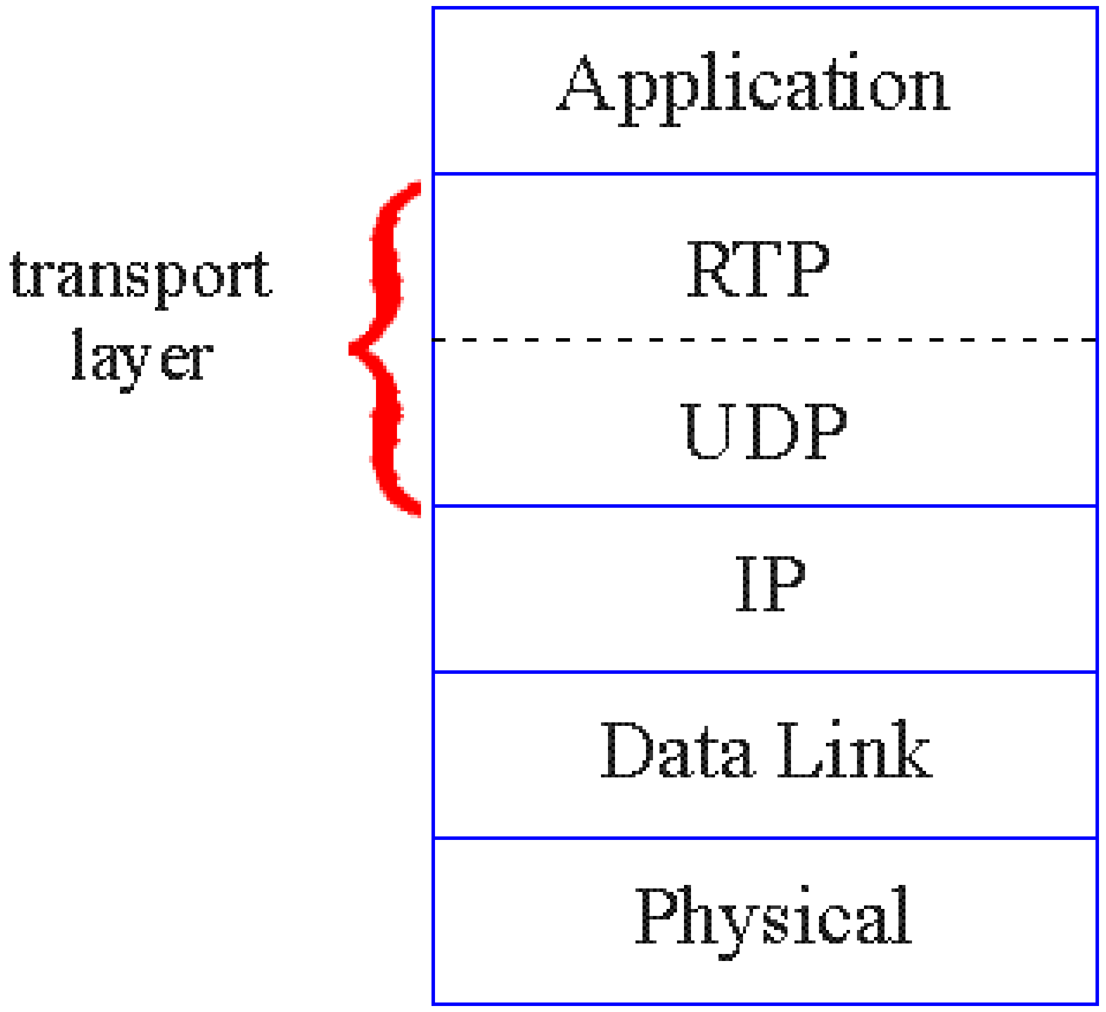


REAL TIME PROTOCOL

- **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

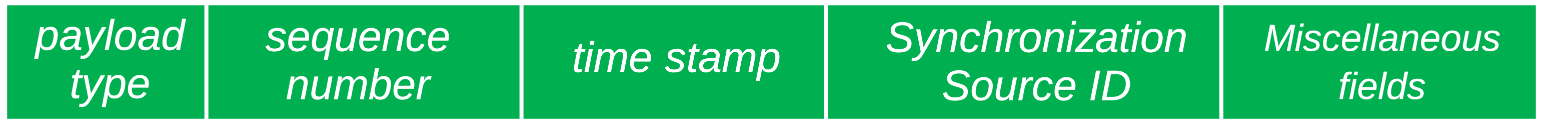
- **RTP does not provide any mechanism to ensure timely data delivery or other QoS guarantees**
- **RTP encapsulation only seen at end systems (not by intermediate routers)**
  - routers provide best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter

RTP HEADER

payload type	*sequence number*	*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



- **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(

- ❑ **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 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 SYNC

- 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.

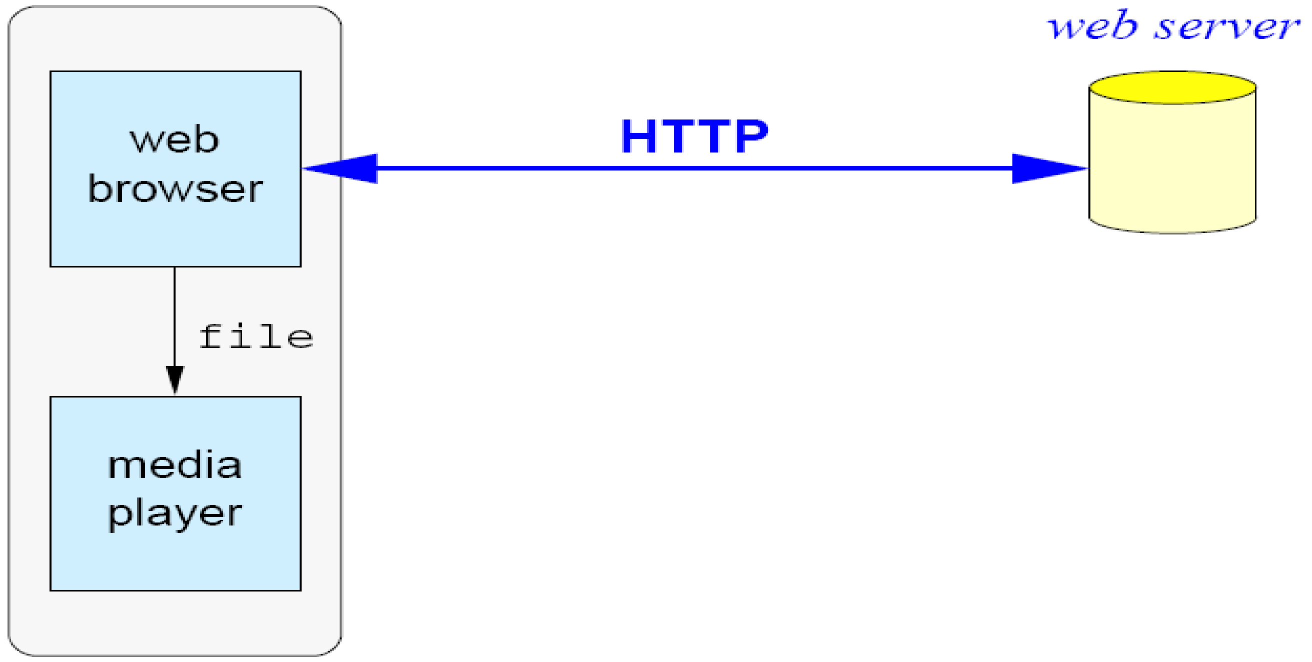
participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate

RTSP

- ❑ **Real Time Streaming Protocol**
- ❑ **Motivation**
  - Internet Video On Demand
    - Internet VCR
      - Requirements like pause, record etc
  - Integration with web architecture
  - A new application level protocol for media files

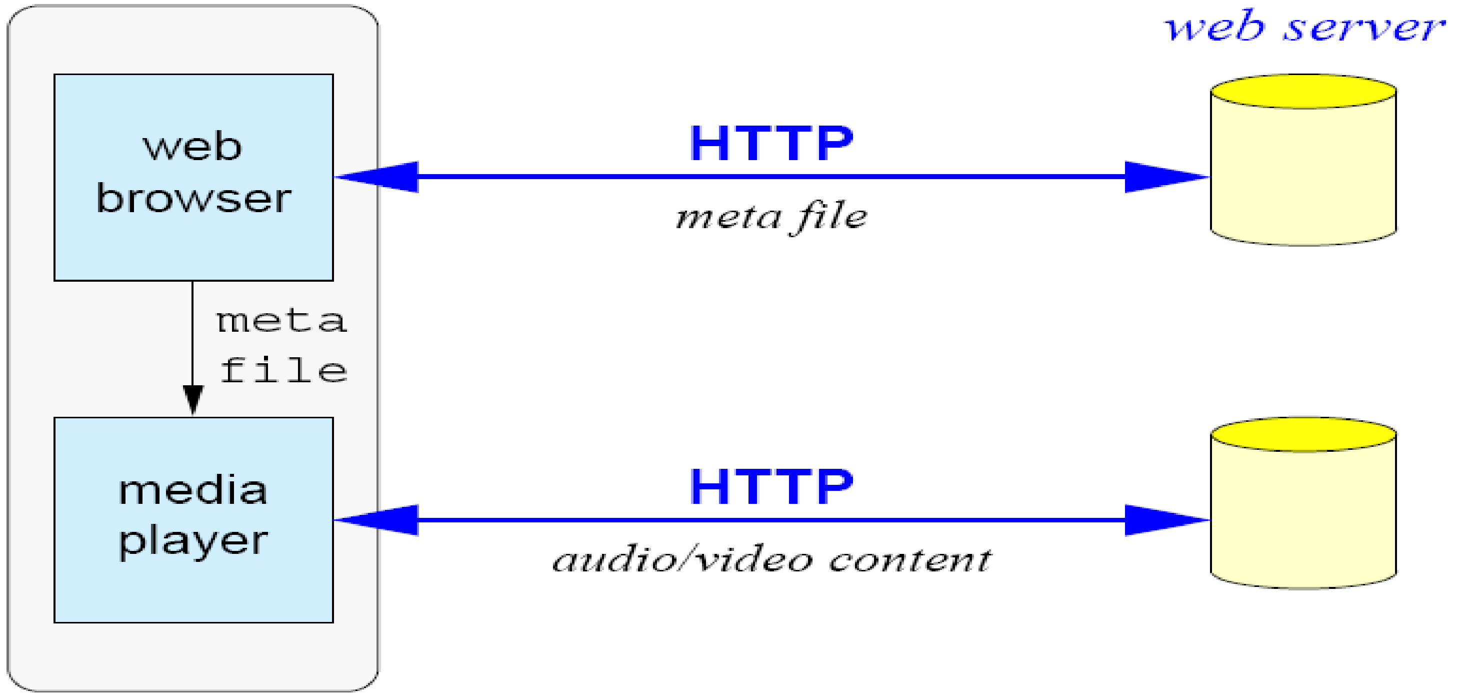
RTSP

❑ Architecture



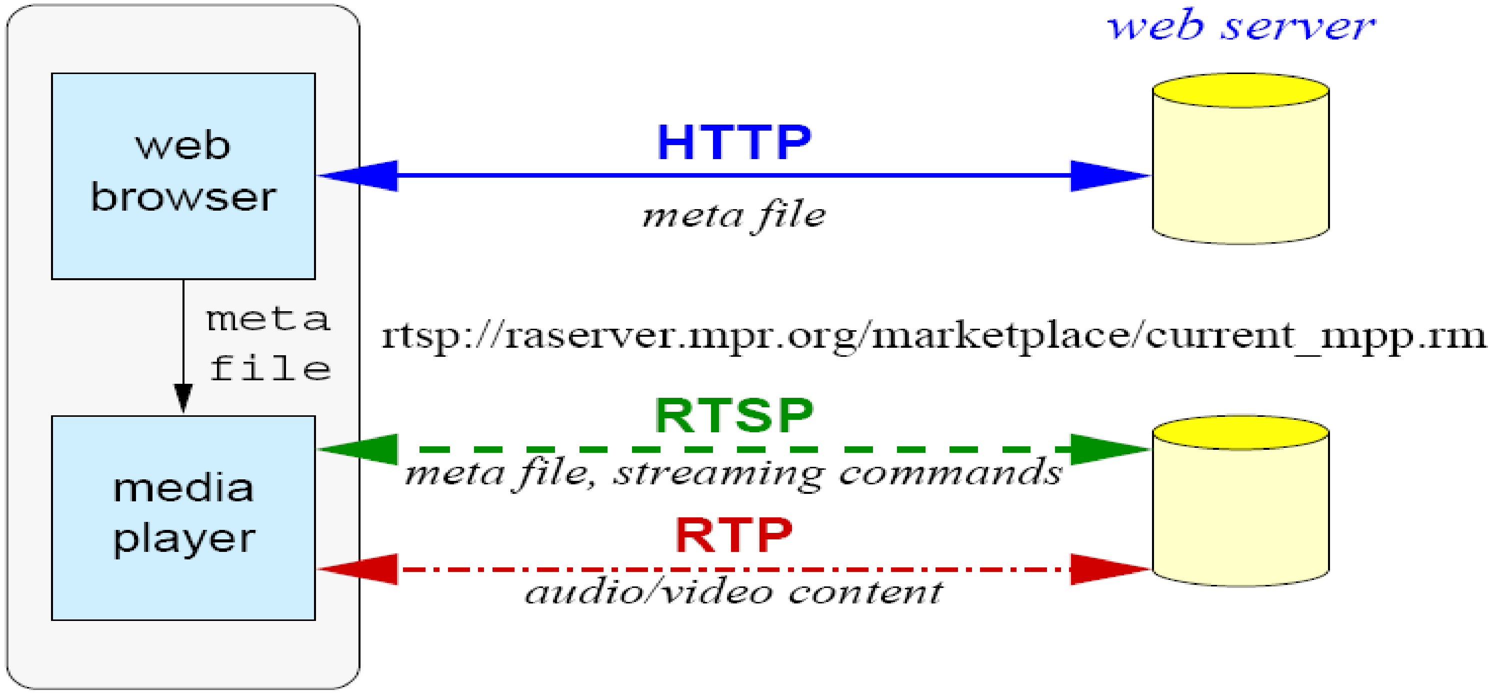
RTSP

❑ Architecture (Meta Files)



RTS

❑ Architecture



RTSP FEATURES

- ❑ “rough” synchronization (fine-grained)
- ❑ virtual presentations = synchronized playback from several servers
- ❑ load balancing using redirection at connect, during stream
- ❑ supports any session description
- ❑ device control → camera pan, zoom, tilt
- ❑ caching: similar to HTTP

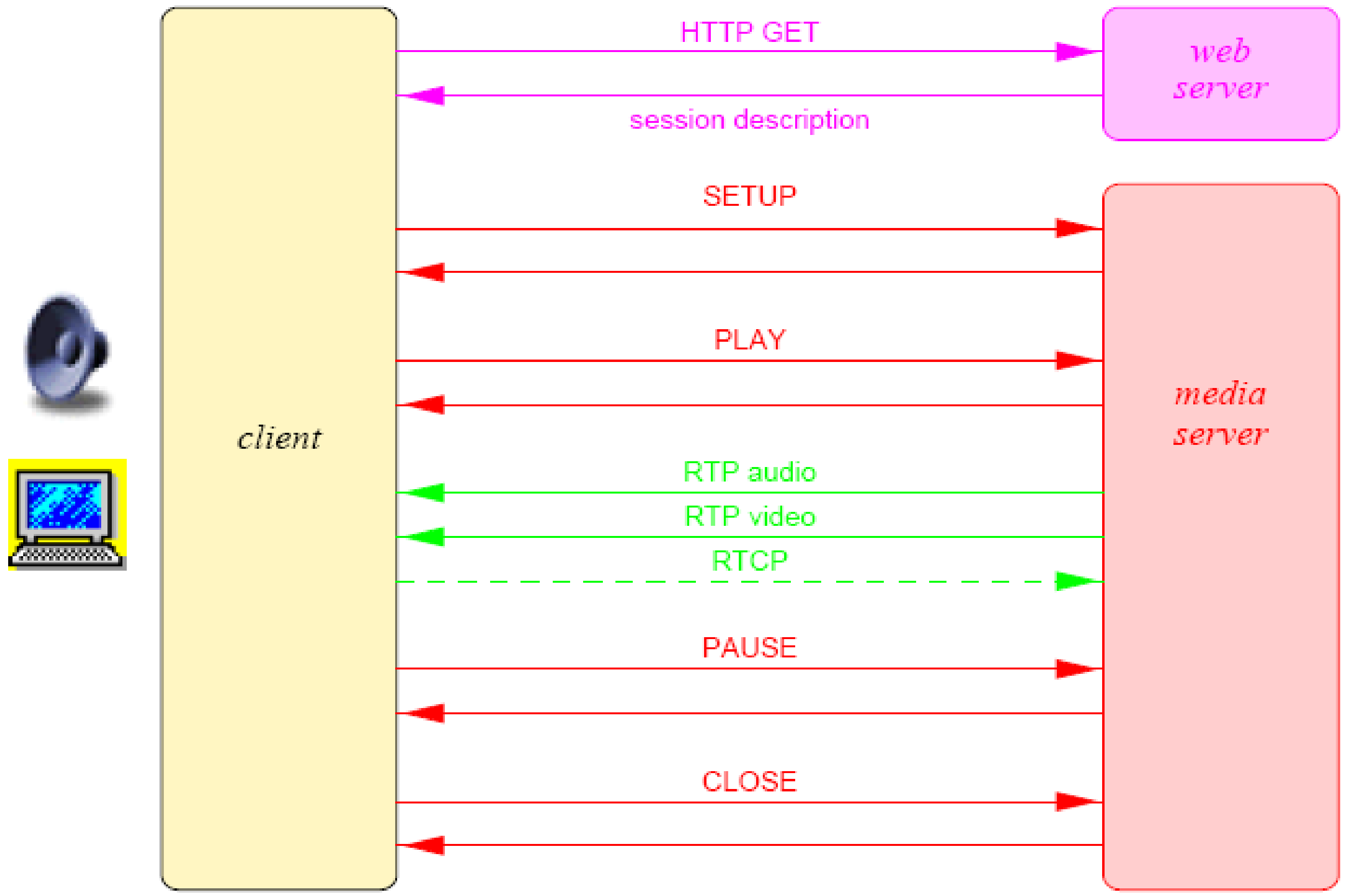
RTSP - FEATURES

- **Retrieval:**
  - first, get presentation description
  - Unicast
  - multicast, client chooses address
  - multicast, server chooses address (NVOD)
  - independent of stream file format

RTSP - FEATURES

- **Methods**
  - OPTIONS
  - SETUP
  - ANNOUNCE
  - DESCRIBE
  - PLAY
  - RECORD
  - REDIRECT
  - PAUSE
  - SET PARAMETER
  - TEARDOWN

RTSP - EXAMPLE



RTSP - TIME

- ❑ **Normal Play Time (NPT) : seconds, microseconds**
- ❑ **SMPTE timestamps: seconds, frames**
- ❑ **Absolute time (for live events)**

SESSION INITIATION PROTOCOL

- **RFC 3261**
- **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

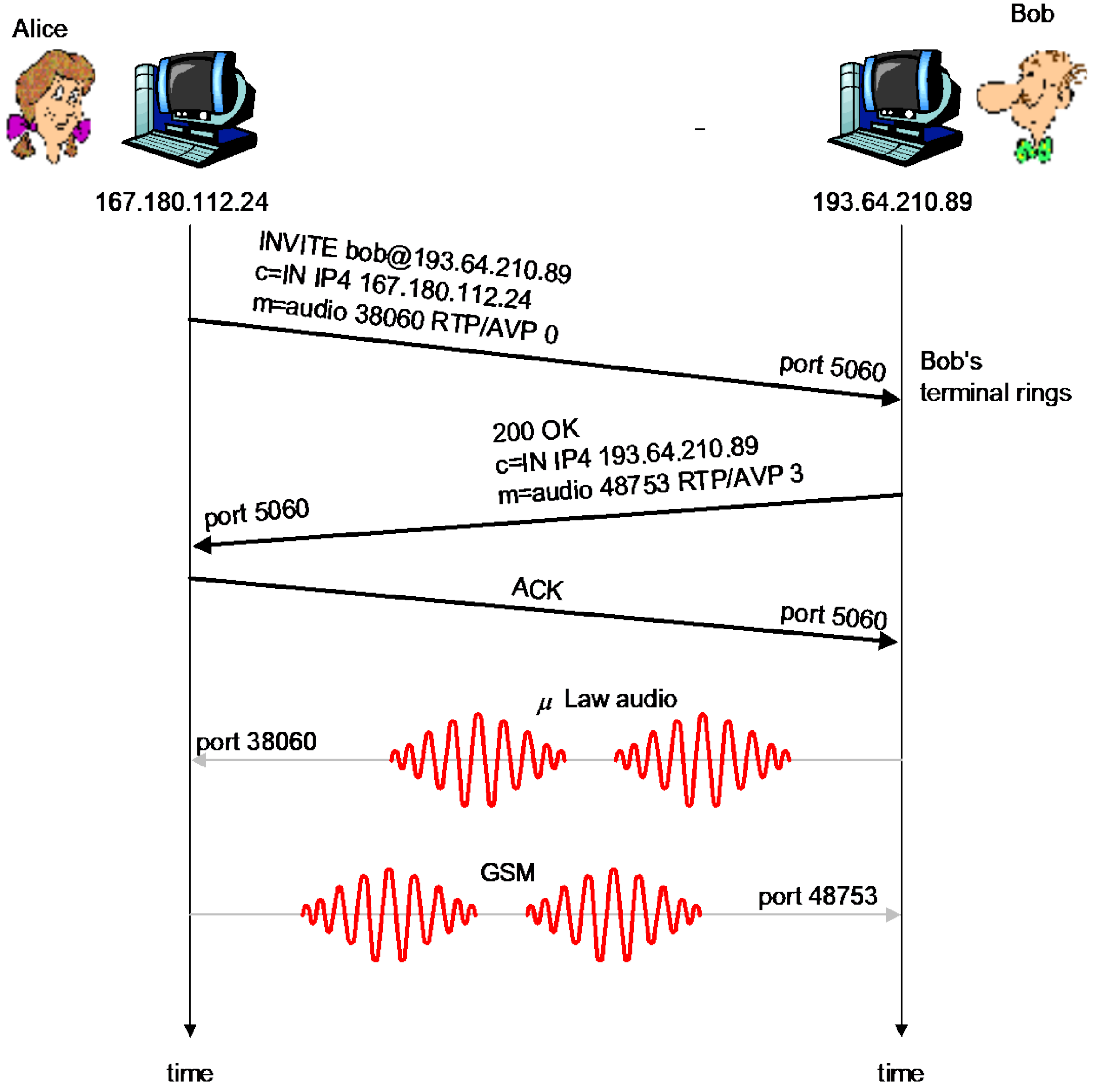
SIP SERVICES

- **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

SIP

❑ 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 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



SIP

❑ 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
```

SIP

❑ 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

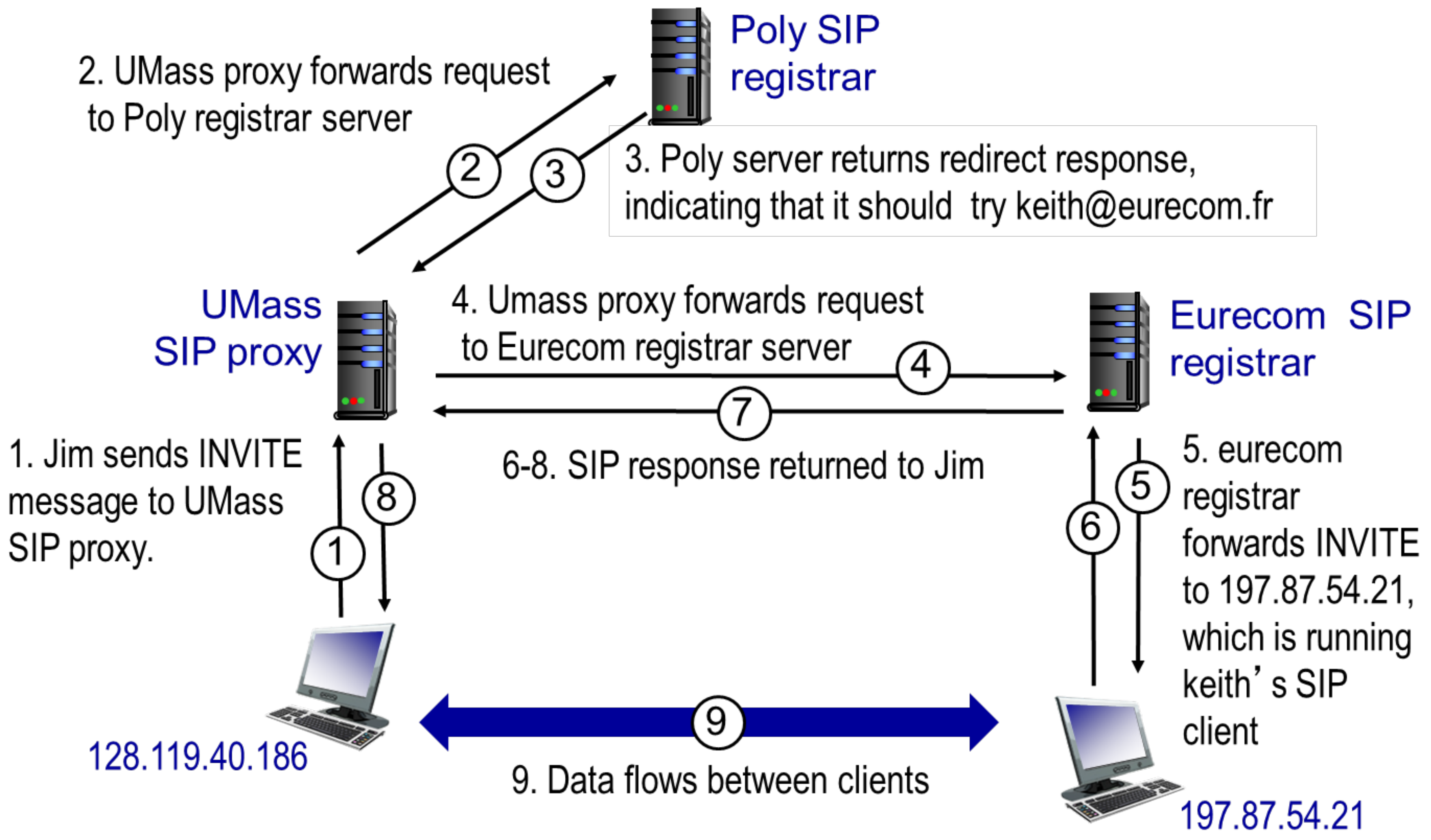
- one function of SIP server: registrar
- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server
- register message:

```
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

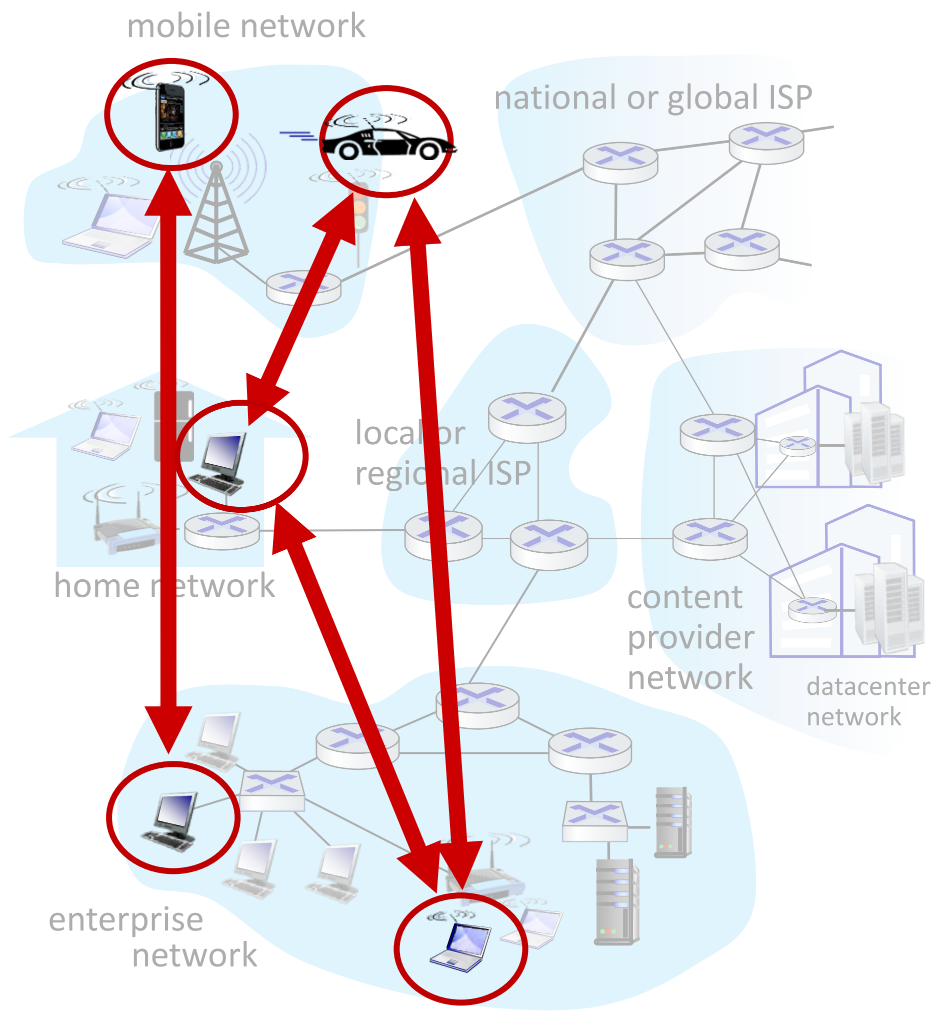
- ❑ **another function of SIP server: proxy**
- ❑ **Alice sends invite message to her proxy server**
  - contains address sip:bob@domain.com
  - proxy responsible for routing SIP messages to callee, possibly through multiple proxies
- ❑ **Bob sends response back through same set of SIP proxies**
- ❑ **proxy returns Bob’s SIP response message to Alice**
  - contains Bob’s IP address
- ❑ **SIP proxy analogous to local DNS server plus TCP setup**

EXAMPLE


Poly SIP registrar
2. UMass proxy forwards request to Poly registrar server
3. Poly server returns redirect response, indicating that it should try keith@eurecom.fr
UMass SIP proxy
4. Umass proxy forwards request to Eurecom registrar server
Eurecom SIP registrar
6-8. SIP response returned to Jim
1. Jim sends INVITE message to UMass SIP proxy.
5. eurecom registrar forwards INVITE to 197.87.54.21, which is running keith's SIP client
1
2
3
4
5
6
7
8
9
128.119.40.186
9. Data flows between clients
197.87.54.21

PEER TO PEER (P2P(ARCHITECTURE

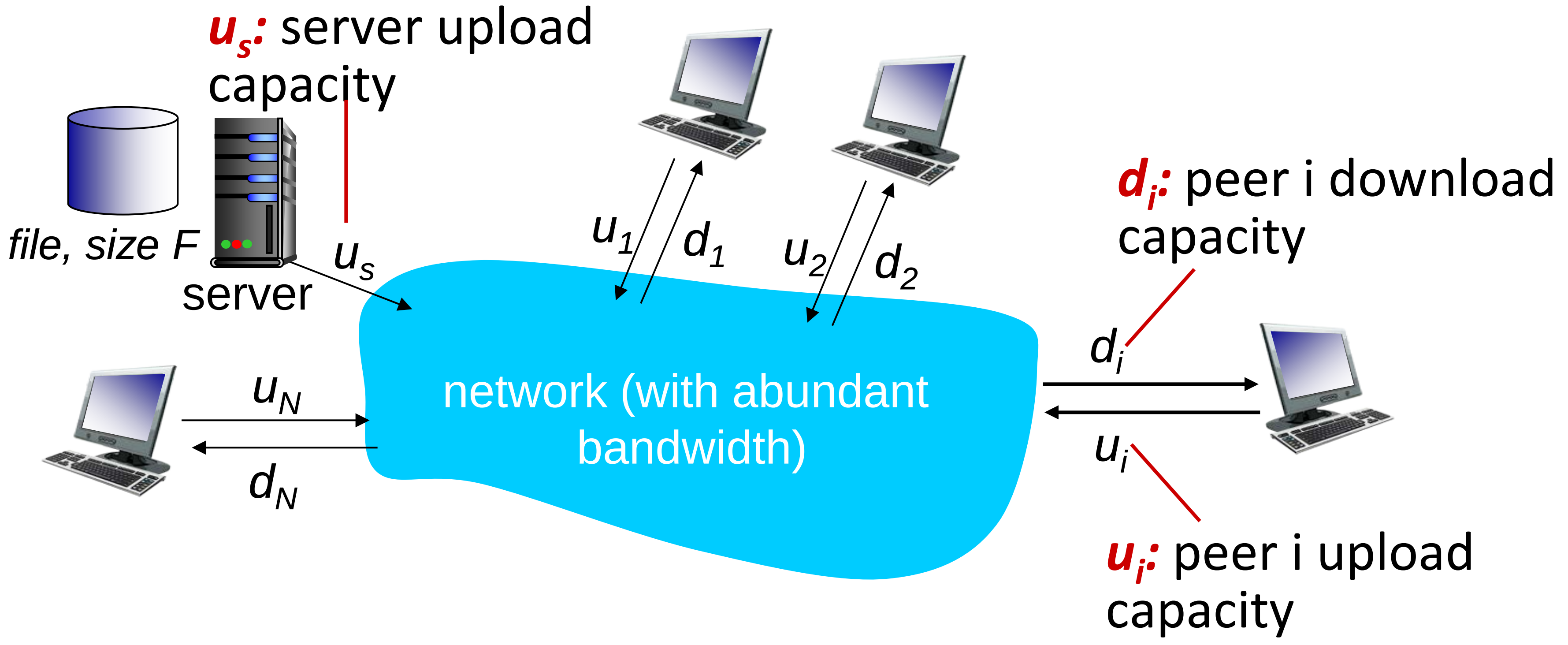
- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, and new service demands
- peers are intermittently connected and change IP addresses
  - complex management
- examples: P2P file sharing (BitTorrent), streaming (KanKan), VoIP (Skype)



FILE DISTRIBUTION(CLIENT SERVER VS P2P

Q: how much time to distribute file (size *F*) from one server to *N peers*?

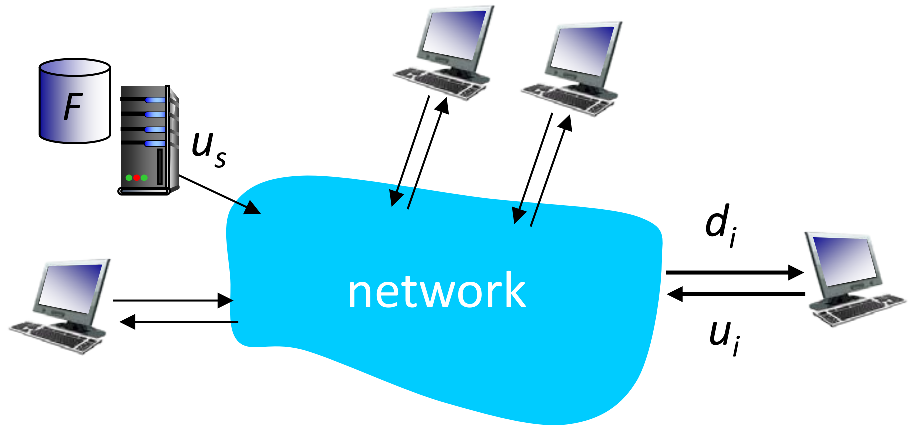
- peer upload/download capacity is limited resource



u_s: server upload capacity
file, size F
server
u_s
u_1
d_1
u_2
d_2
d_i: peer i download capacity
d_i
u_N
d_N
network (with abundant bandwidth)
u_i
u_i: peer i upload capacity

FILE DISTRIBUTION TIME(CLIENT SERVER

- *server transmission:* must sequentially send (upload) N file copies:
  - time to send one copy: F/u_s
  - time to send N copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}



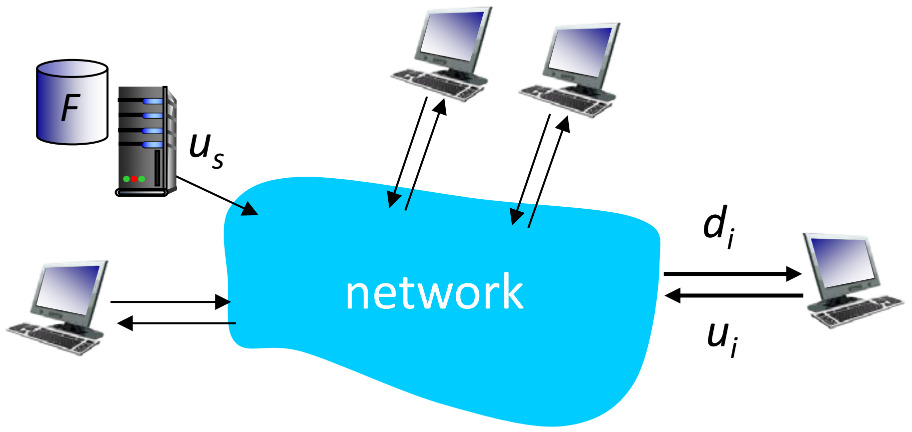
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_{s,},F/d_{min}\}$$

increases linearly in N

FILE DISTRIBUTION TIME(P2P

- *server transmission:* must upload at least one copy:
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download *NF* bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$



time to distribute F to N clients using P2P approach

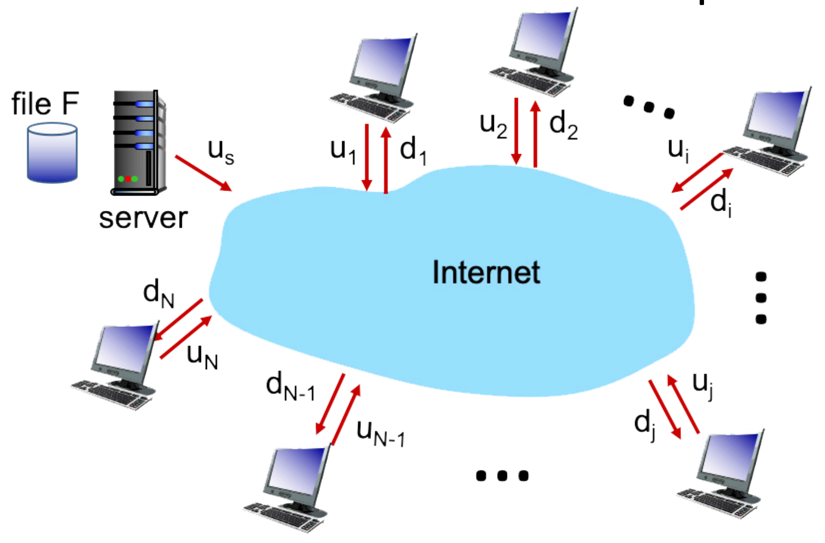
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

SAMPLE PROBLEM

Q: Compare the time needed to distribute a file that is initially located at a server to clients via either client-server download or peer-to-peer download.



The problem is to distribute a file of size **F = 8 Gbits** to each of these **5 peers**. Suppose the server has an upload rate of ***u* = 59 Mbps**.

The **5 peers** have **upload rates** of: **u_1 = 21 Mbps, u_2 = 30 Mbps, u_3 = 15 Mbps, u_4 = 23 Mbps, and u_5 = 13 Mbps**

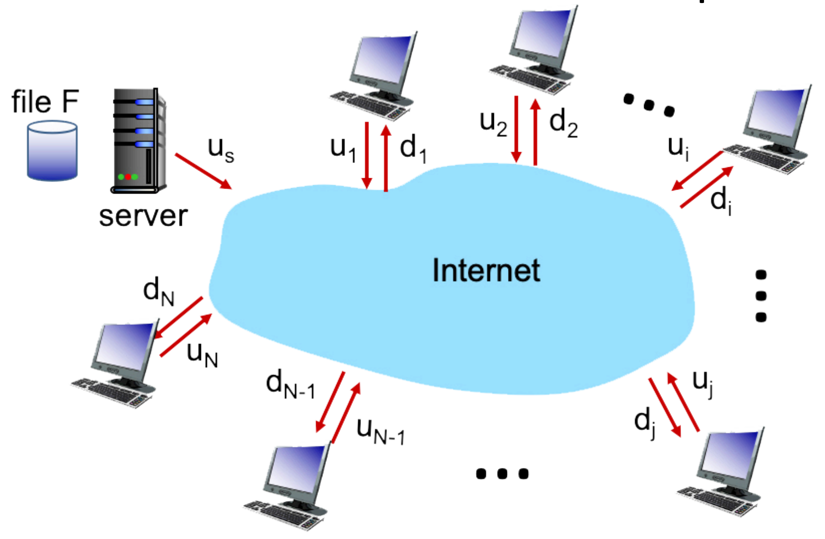
The 5 peers have **download rates** of: **d_1 = 21 Mbps, d_2 = 40 Mbps, d_3 = 23 Mbps, d_4 = 20 Mbps, and d_5 = 16 Mbps**

1. What is the minimum time needed to distribute this file from the central server to the 5 peers using the client-server model?

2. What is the minimum time needed to distribute this file using peer-to-peer download?

SAMPLE PROBLEM

Q: Compare the time needed to distribute a file that is initially located at a server to clients via either client-server download or peer-to-peer download.



The problem is to distribute a file of size **F = 8 Gbits** to each of these **5 peers**. Suppose the server has an upload rate of ***u* = 59 Mbps**.

The **5 peers** have **upload rates** of: **u_1 = 21 Mbps, u_2 = 30 Mbps, u_3 = 15 Mbps, u_4 = 23 Mbps, and u_5 = 13 Mbps**

The 5 peers have **download rates** of: **d_1 = 21 Mbps, d_2 = 40 Mbps, d_3 = 23 Mbps, d_4 = 20 Mbps, and d_5 = 16 Mbps**

What is the minimum time needed to distribute this file from the central server to the 5 peers using the client-server model?

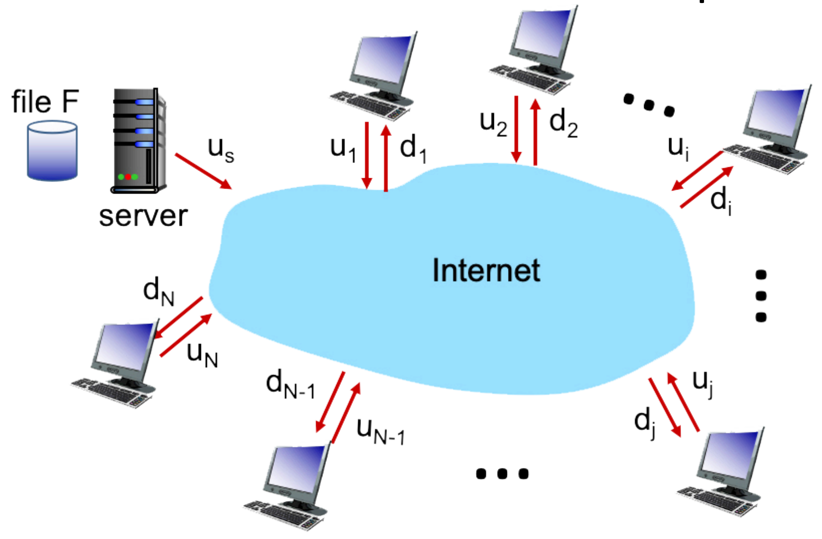
$$D_{c\text{-}s} > max\{NF/u_{s,},F/d_{min}\}$$

$$D_{c\text{-}s} > max\{5*8x10^9 bits/59x10^6 bits/second,\ 8x10^9 bits/16x10^6 bits/second\}$$

$$D_{c\text{-}s} > max\{\underline{677.97\ seconds},\ 500\ seconds\}$$

SAMPLE PROBLEM

Q: Compare the time needed to distribute a file that is initially located at a server to clients via either client-server download or peer-to-peer download.



The problem is to distribute a file of size **F = 8 Gbits** to each of these **5 peers**. Suppose the server has an upload rate of ***u* = 59 Mbps**.

The **5 peers** have **upload rates** of: **u_1 = 21 Mbps, u_2 = 30 Mbps, u_3 = 15 Mbps, u_4 = 23 Mbps, and u_5 = 13 Mbps**

The 5 peers have **download rates** of: **d_1 = 21 Mbps, d_2 = 40 Mbps, d_3 = 23 Mbps, d_4 = 20 Mbps, and d_5 = 16 Mbps**

What is the minimum time needed to distribute this file using peer-to-peer download?

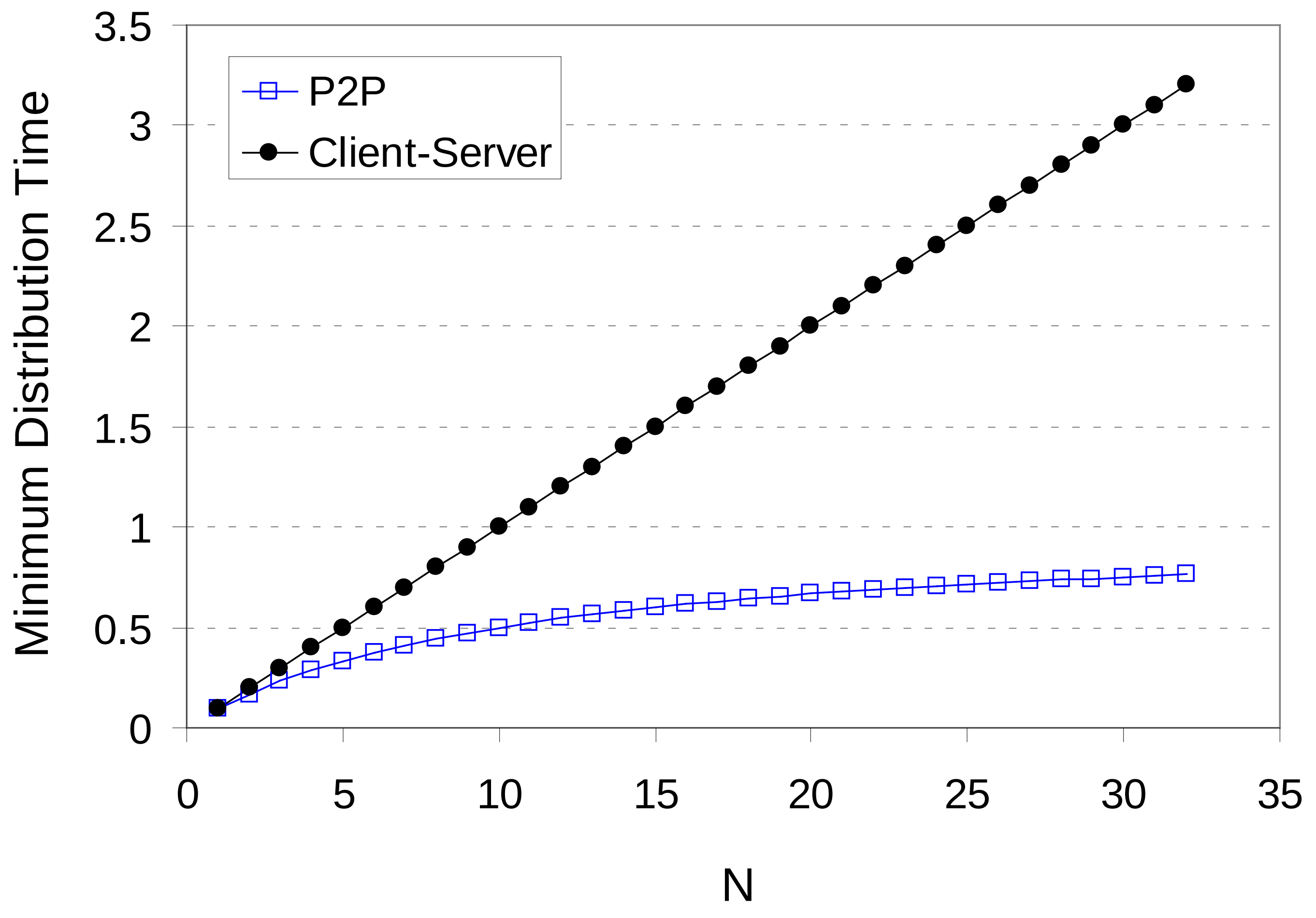
$$D_{P2P} > max\{F/u_{s,}, F/d_{min,}, NF/(u_s + \Sigma u_i)\}$$

$$D_{P2P} > max\{8x10^9 bits/59x10^6 bits/second,\ 8x10^9 bits/16x10^6 bits/second,\ (5*8x10^9 bits)/(59+21+30+15+23+13\ Mbps)\}$$

$$D_{c\text{-}s} > max\{135.59\ seconds,\ \underline{\textbf{500 seconds}},\ 248.45\ seconds\}$$

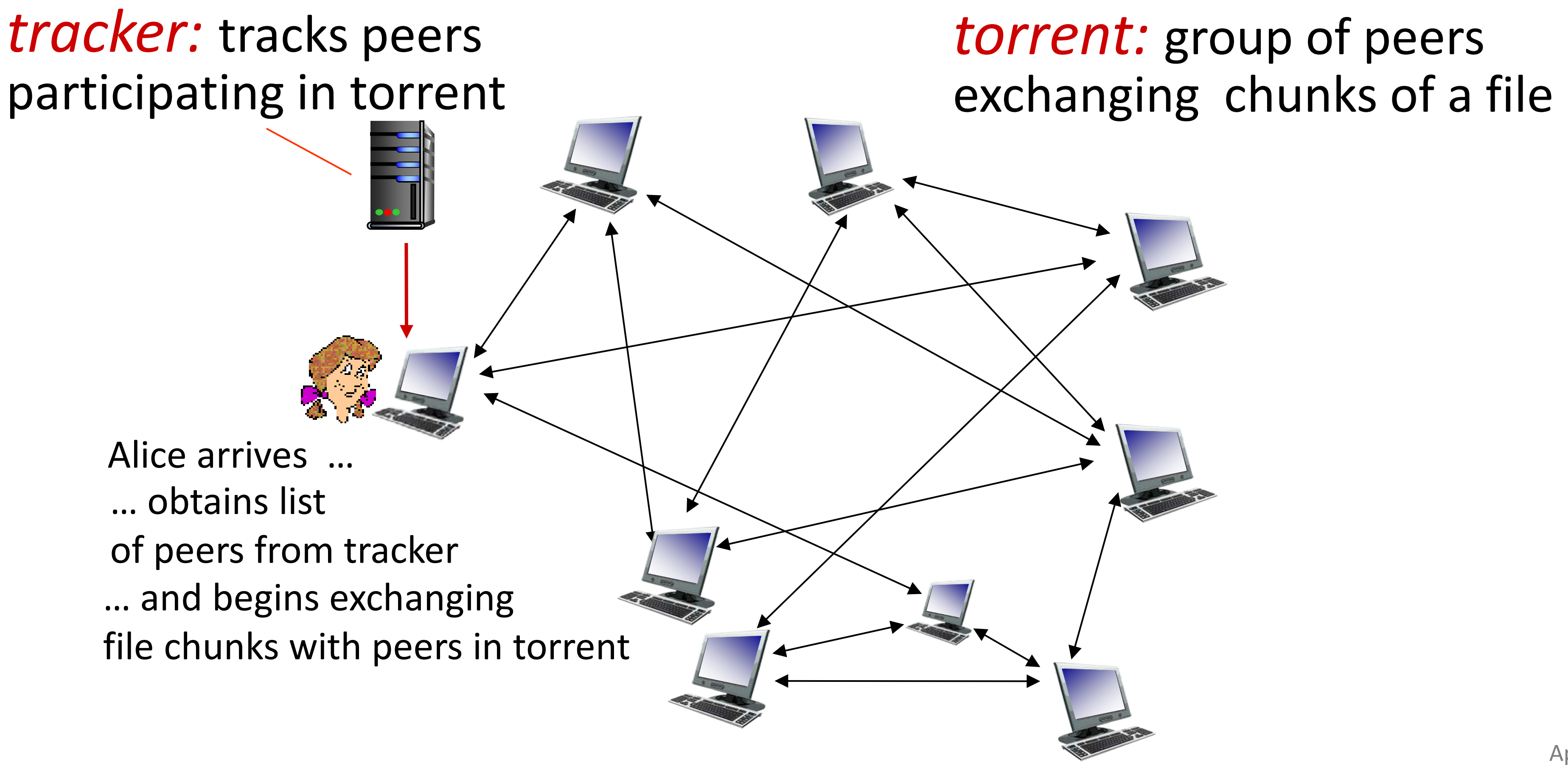
CLIENT SERVER VS(P2P(EXAMPLE

client upload rate = *u*, *F/u* = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



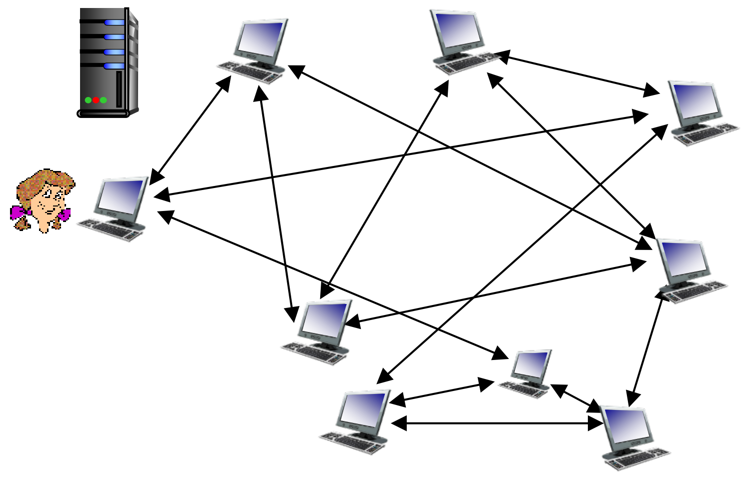
P2P FILE DISTRIBUTION(BITTORRENT

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks


tracker: tracks peers participating in torrent
torrent: group of peers exchanging chunks of a file
Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P FILE DISTRIBUTION(BITTORRENT

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers (“neighbors”)



- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BITTORRENT(REQUESTING(SENDING FILE CHUNKS

Requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Sending chunks: tit-for-tat

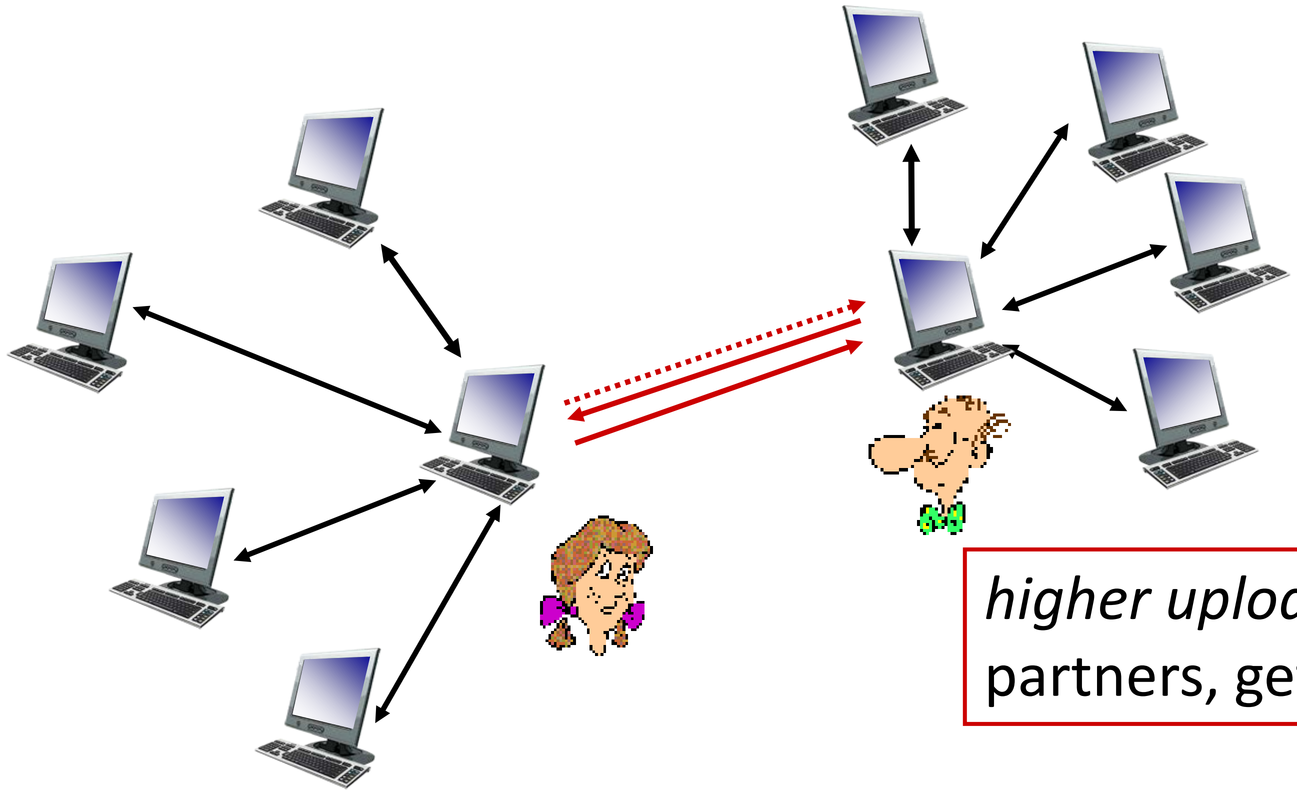
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - “optimistically unchoke” this peer
  - newly chosen peer may join top 4

BITTORRENT(TIT FOR TAT

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

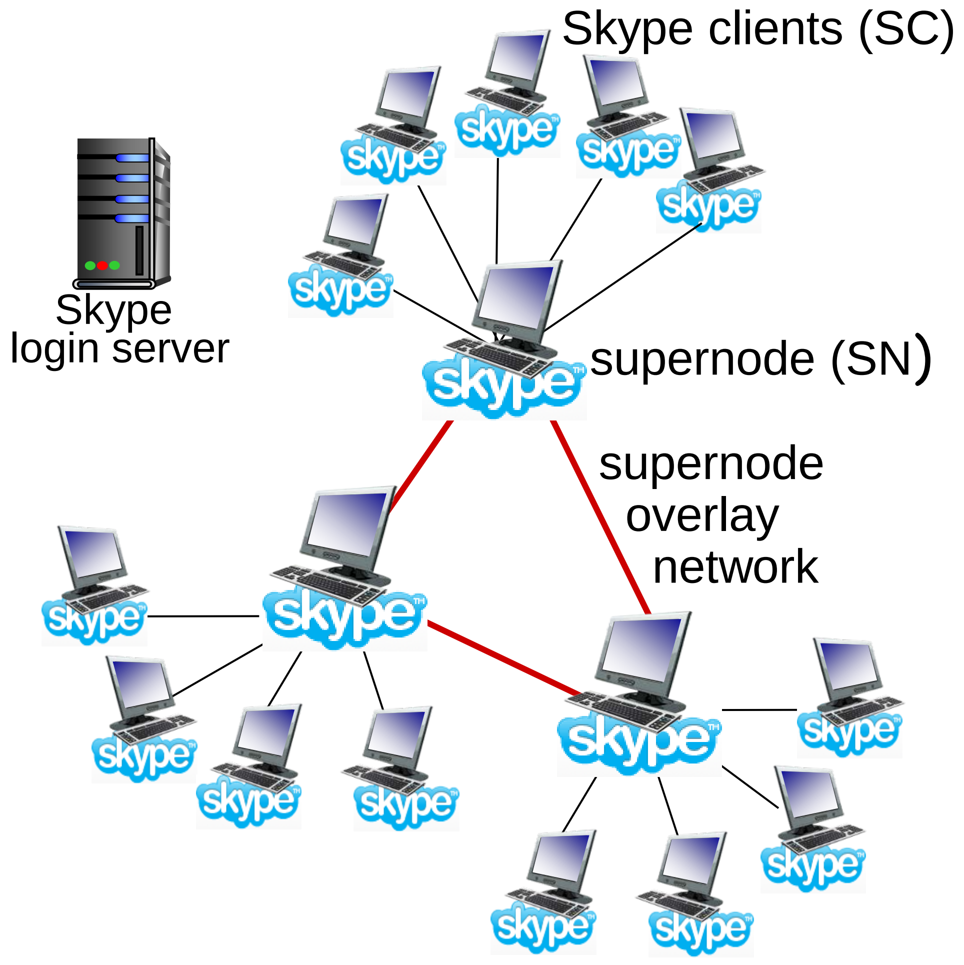
(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

Voice-over-IP: Skype

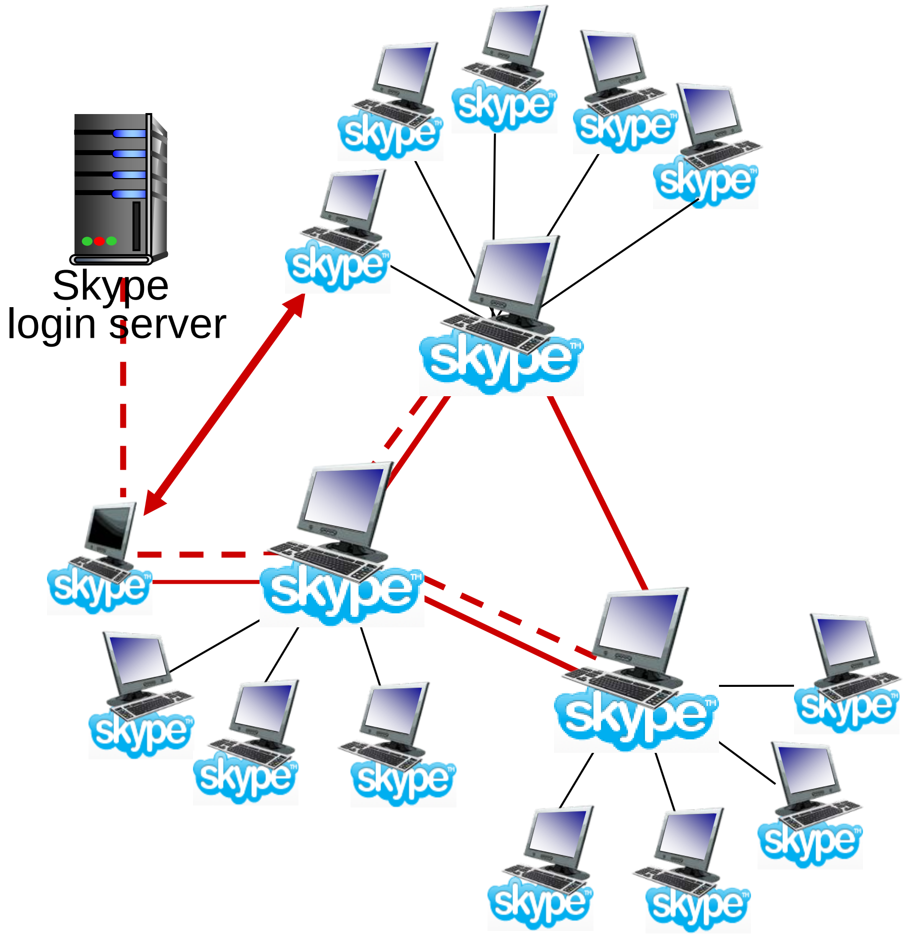
- proprietary application-layer protocol (inferred via reverse engineering)
  - encrypted msgs
- P2P components:
  - clients: Skype peers connect directly to each other for VoIP call
  - super nodes (SN): Skype peers with special functions
  - overlay network: among SNs to locate SCs
  - login server



P2P voice-over-IP: Skype

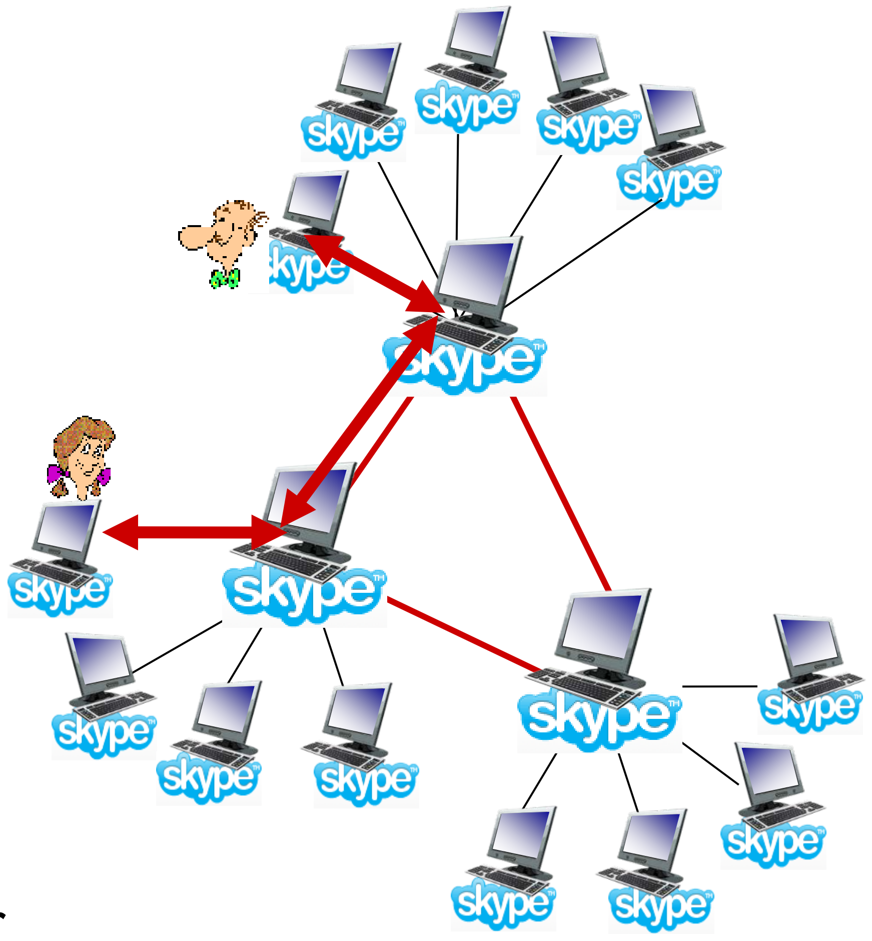
Skype client operation:

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



MESSAGE FROM DPO

"The information and data contained in the online learning modules, such as the content, audio/visual materials or artwork are considered the intellectual property of the author and shall be treated in accordance with the IP Policies of DLSU. They are considered confidential information and intended only for the person/s or entities to which they are addressed. They are not allowed to be disclosed, distributed, lifted, or in any way reproduced without the written consent of the author/owner of the intellectual property."