Voice-over-IP (VoIP)

VoIP

VoIP ***end-end-delay requirement***: Required to maintain *"conversational"* aspect

- Higher delays are noticeable; impair interactivity
- ***< 150 ms***: *good*
- ***> 400 ms***: *bad*
- Includes application-level (packetization, playout) and network-level delays

Session initialization

- How does a *callee* advertise *IP address, port number, and 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***
- Packets generated *only* during talk spurts
- *20 ms* ***chunks*** at *8 Kbytes/s*: *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 ms* during talk spurt

VoIP: Packet loss, delay

Network loss

- IP datagram lost due to network congestion (router buffer overflow)

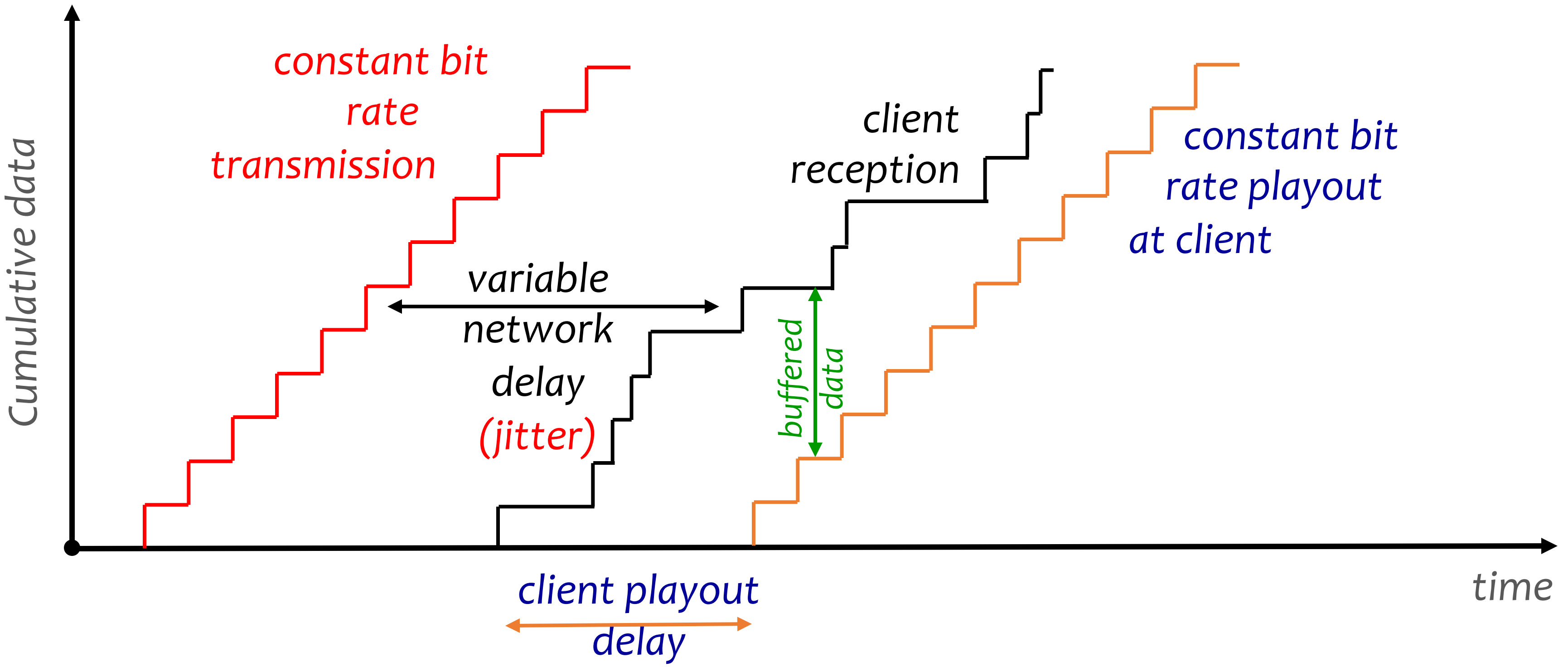
Delay loss

- IP datagram arrives too late for playout at receiver
- 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 ms (transmission time difference)

VoIP: Fixed playout delay

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

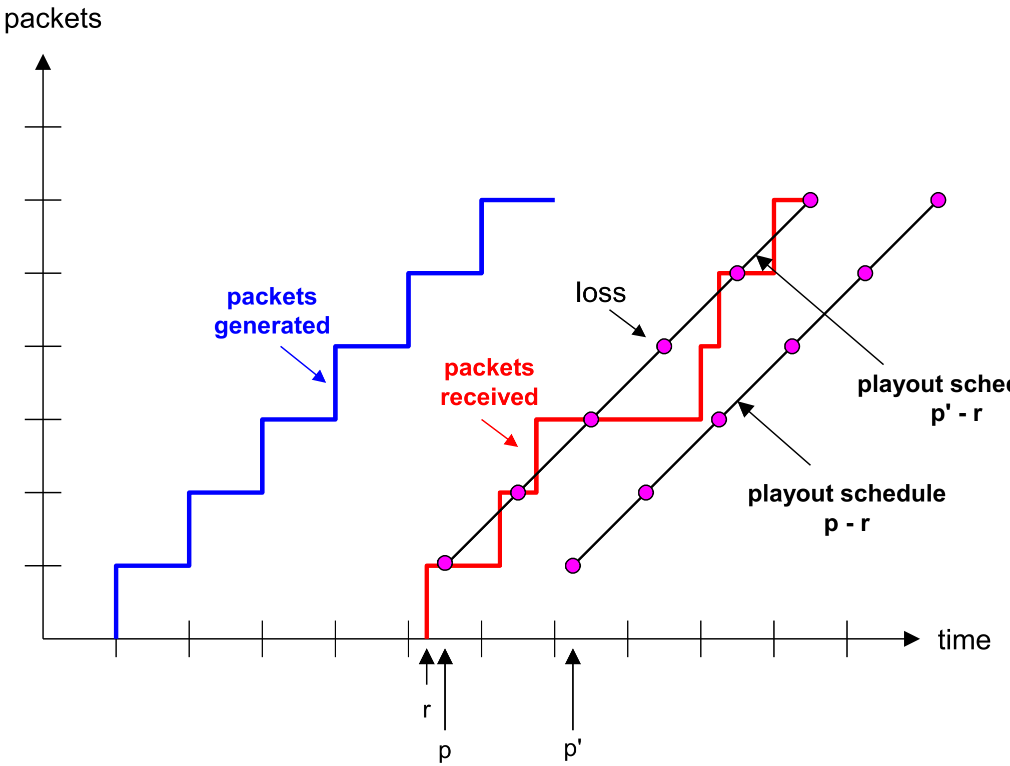
- Chunk has time stamp ***t***: play out chunk at ***t+q***
- Chunk arrives after ***t+q***: data arrives too late for playout; data "lost"

Tradeoff in choosing ***q***:

- ***large q***: *less packet loss*
- ***small q***: *better interactive experience*

VoIP: Fixed playout delay

- Sender generates packets every *20 ms* during talk spurt.
- First packet received at time ***r***
- First playout schedule: begins at ***p***
- Second playout schedule: begins at ***p'***



VoIP: Adaptive playout delay

Goal

- Low playout delay, low late loss rate

Approach

- Adaptive playout delay adjustment

VoIP: Adaptive playout delay

Approach

- Adaptive playout delay adjustment

How?

- Estimate network delay, adjust playout delay at beginning of each talk spurt
- Silent periods compressed and elongated
- Chunks still played out every *20 ms* during talk spurt
- Adaptively estimate packet delay
  - Exponentially weighted moving average (EWMA); recall TCP RTT estimate: $d_i = (1-\alpha)d_{i-1} + \alpha(r_i - t_i)$

VoIP: Adaptive playout delay

- 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 talk spurt are played out periodically

VoIP: adaptive playout delay

How does receiver determine whether packet is first in a talk spurt?

- If no loss, receiver looks at successive timestamps
  - Difference of successive stamps > 20 msec -->talk spurt begins.
- Under loss, 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: Loss recovery

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

VoIP: Loss recovery

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

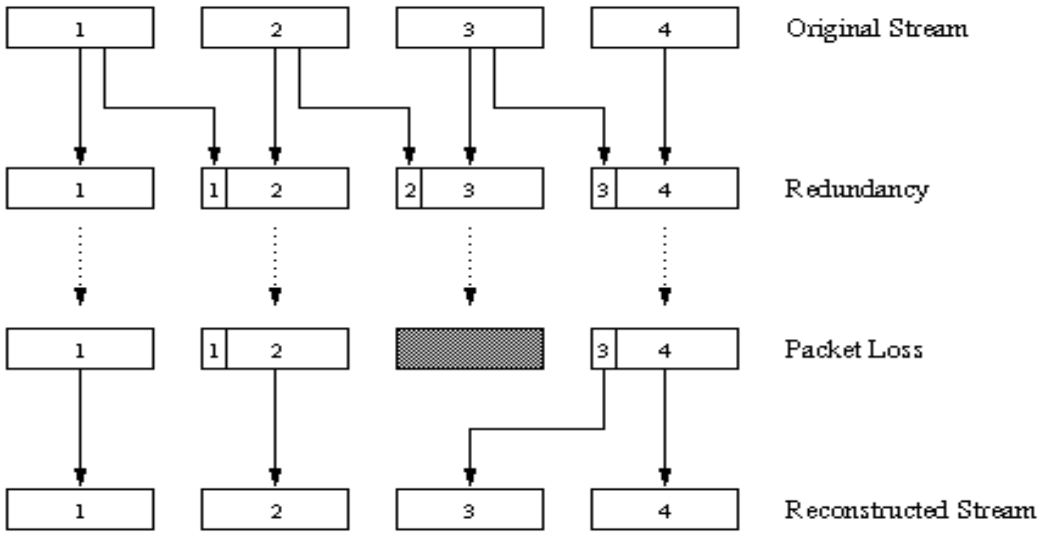
VoIP: Loss recovery

Another FEC ...

- "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*

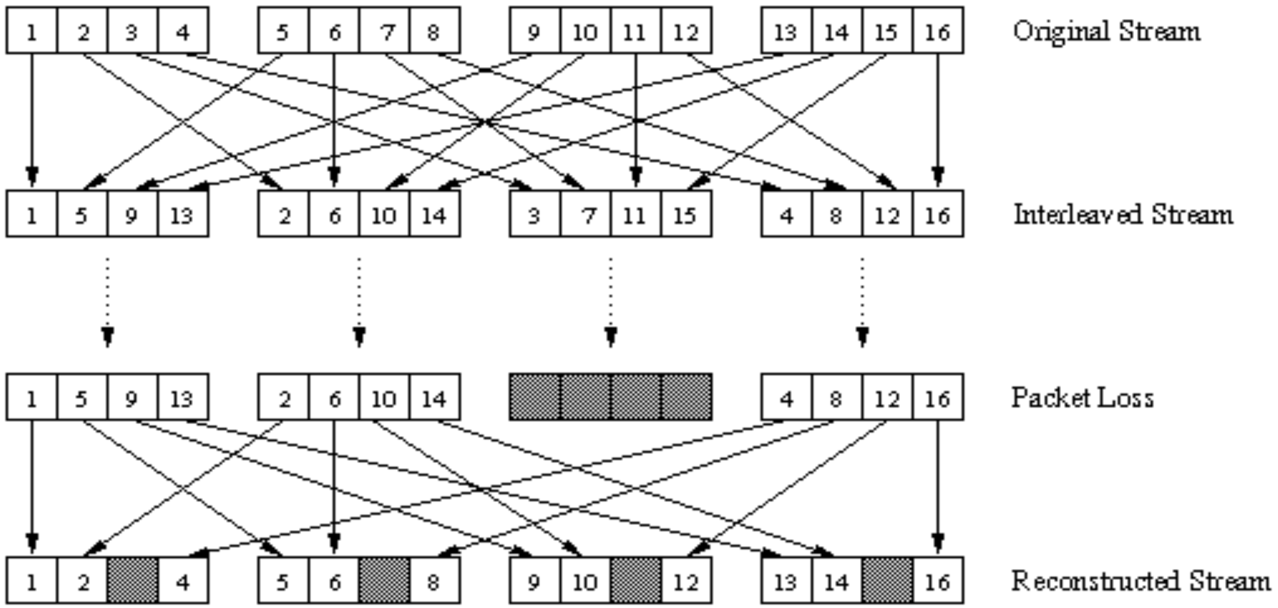
- *Non-consecutive loss*: Receiver can conceal loss
- *Generalization*: Can also append *(n-1)*st and *(n-2)*nd low-bit rate chunk



VoIP: Loss recovery

Interleaving to conceal loss …

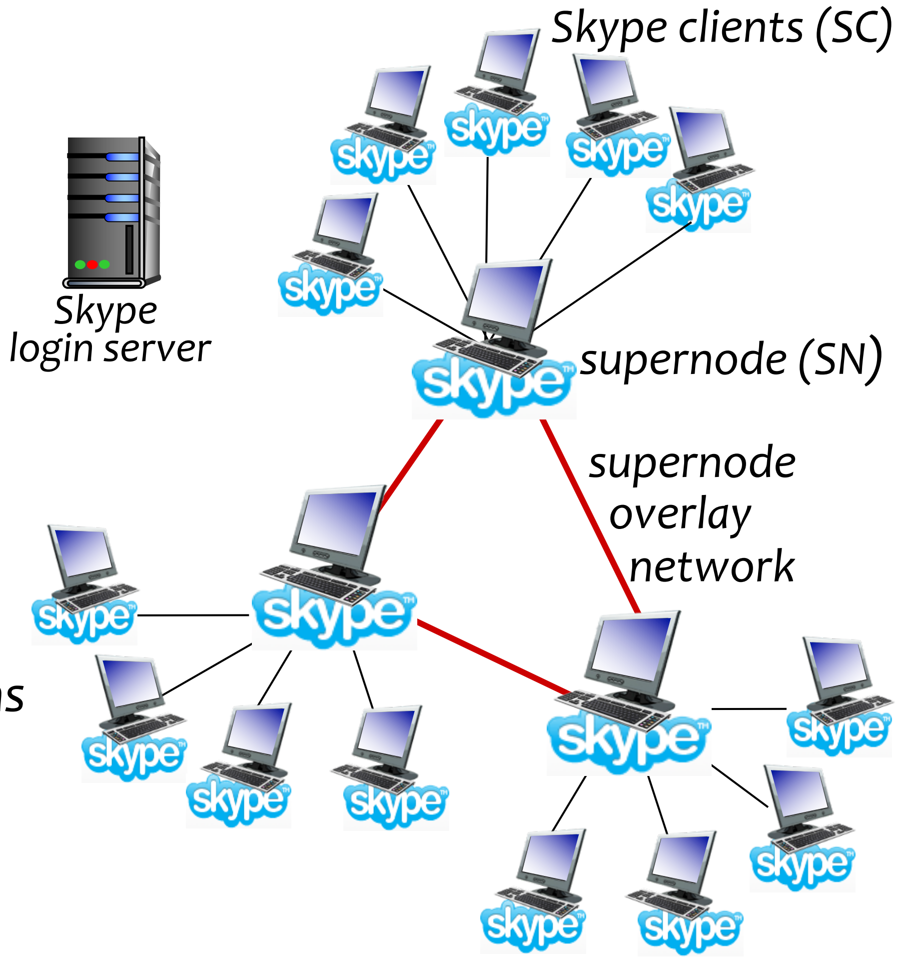
- Audio chunks divided into smaller units, e.g. four *5 ms* units per *20 ms* audio chunk
- Packet contains small units from different chunks
- If packet lost, still have *most* of every original chunk
- No redundancy overhead, *but* increases playout delay



VoIP: Skype

Proprietary application-layer protocol (inferred via *reverse engineering*)

- Encrypted messages
- Older version (prior to Microsoft's acquisition)



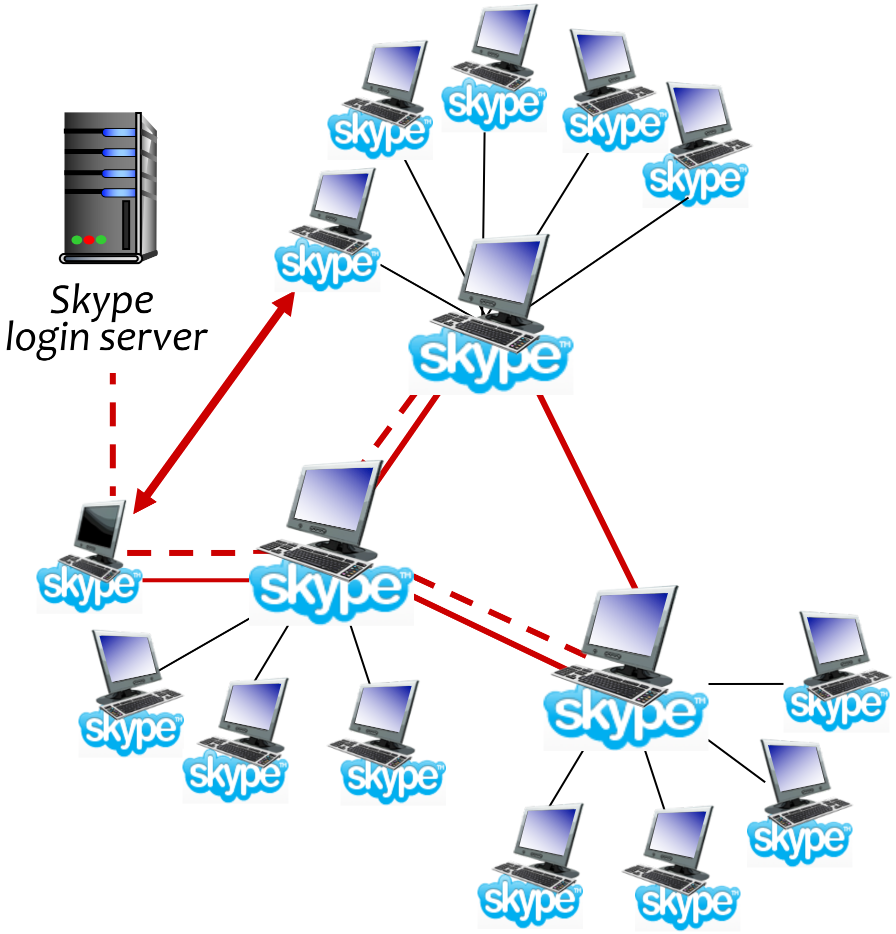
P2P components

- *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 VoIP: Skype

Skype client operation:

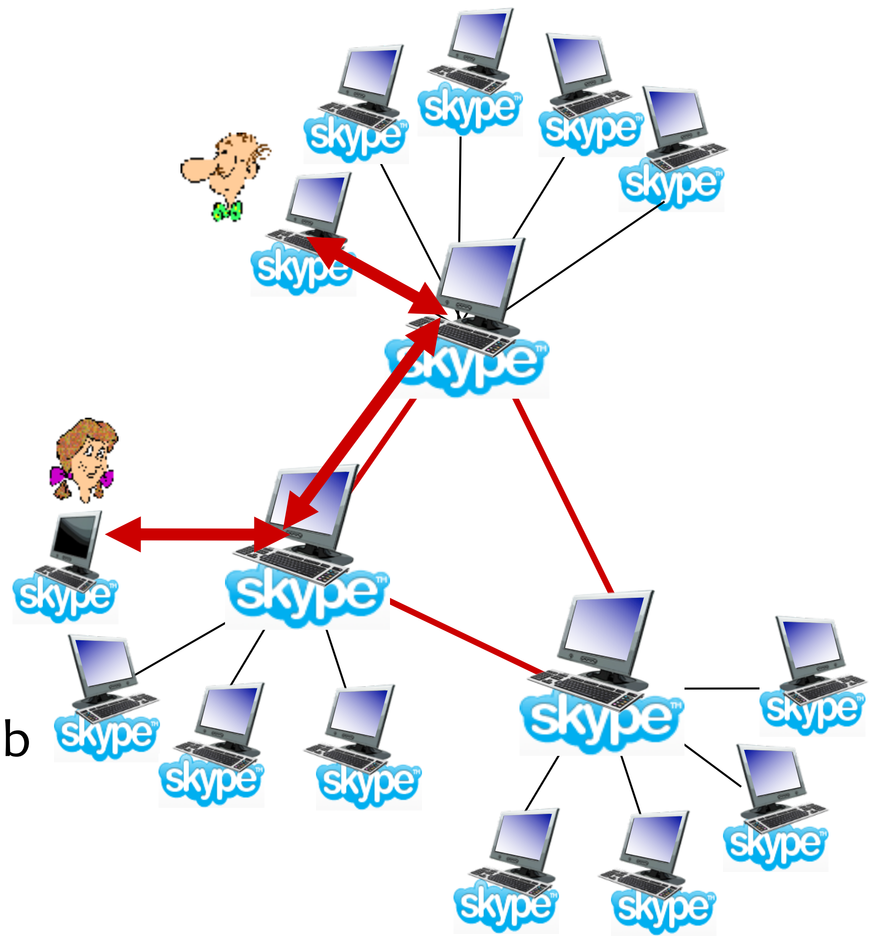
1. Joins Skype network by contacting the SN (IP address cached) using TCP
2. Logs-in (username, password) to centralized Skype login server
3. Obtains IP address for callee from SN, SN overlay, or client buddy list
4. Initiate call directly to callee



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

Summary

- Voice-over-IP
  - Jitter, playout, applications