# Real-time interactive applications

- PC-2-PC phone
- PC-2-phone
  - Dialpad
  - Net2phone
- videoconference
- Webcams

- Now we look at a PC-2-PC Internet phone example in detail

# Internet phone over best-effort (1)

Best effort

- packet delay, loss and jitter

Internet phone example

- now examine how packet delay, loss and jitter are often handled in the context of an IP phone example.
- Internet phone applications generate packets during talk spurts
- bit rate is 64 kbps during talk spurt
- during talk spurt, every 20 msec app generates a chunk of 160 bytes = 8 kbytes/sec * 20 msec
- header is added to chunk; then chunk+header is encapsulated into a UDP packet and sent out
- some packets can be lost and packet delay will fluctuate.
- receiver must determine when to playback a chunk, and determine what do with missing chunk

# Internet phone (2)

## packet loss

- UDP segment is encapsulated in IP datagram
- datagram may overflow a router queue
- TCP can eliminate loss, but
  - retransmissions add delay
  - TCP congestion control limits transmission rate
- Redundant packets can help

## end-to-end delay

- accumulation of transmission, propagation, and queuing delays
- more than 400 msec of end-to-end delay seriously hinders interactivity; the smaller the better

## delay jitter

- consider two consecutive packets in talk spurt
- initial spacing is 20 msec, but spacing at receiver can be more or less than 20 msec

## removing jitter

- sequence numbers
- timestamps
- delaying playout

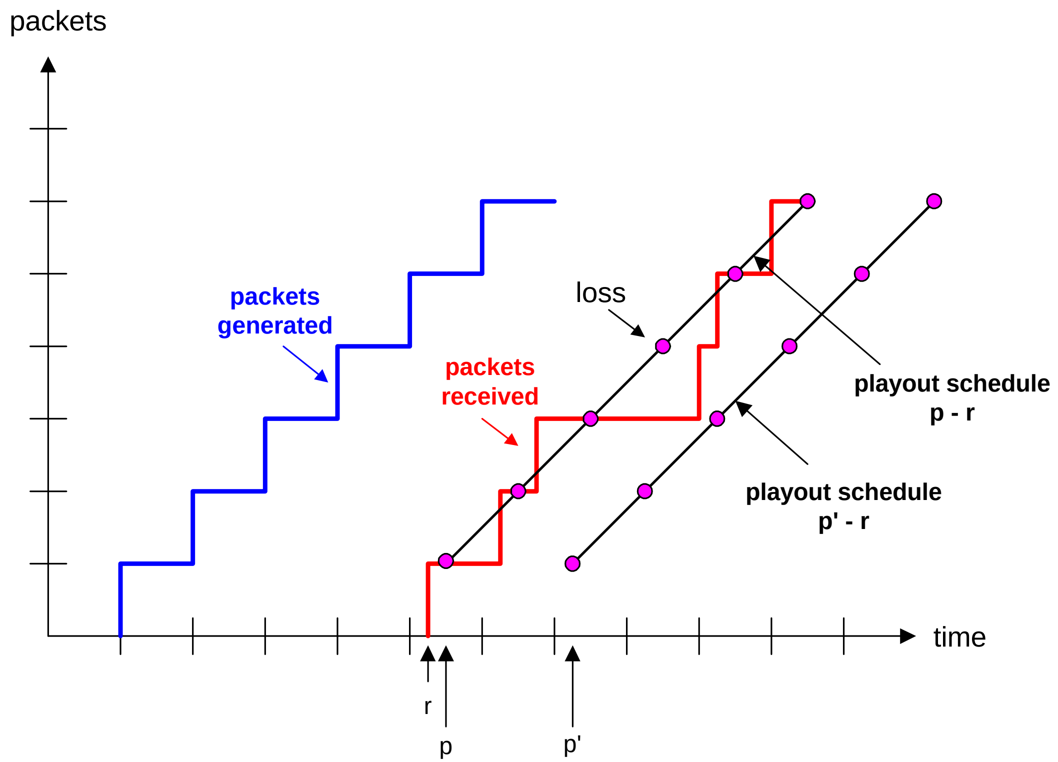
# Internet phone (3): fixed playout delay

- Receiver attempts to playout each chunk at exactly q msecs after the chunk is generated.
  - If chunk is time stamped t, receiver plays out chunk at t+q .
  - If chunk arrives after time t+q, receiver discards it.
- Strategy allows for lost packets.

- Tradeoff for q:
  - large q: less packet loss
  - small q: better interactive experience

# Internet phone (4): 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)

- Estimate network delay and adjust playout delay at the beginning of each talk spurt.

- Silent periods are compressed and elongated.

- Chunks still played out every 20 msec during talk spurt.

$t_i$ = timestamp of the $i$th packet

$r_i$ = the time packet $i$ is received by receiver

$p_i$ = the time packet $i$ is played at receiver

$r_i - t_i$ = network delay for $i$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 (e.g., $u$ = .01).

# Adaptive playout delay (2)

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

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

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

For first packet in talk spurt, playout time is:

$$p_i = t_i + d_i + Kv_i$$

where K is a positive constant. For this same packet, the play out delay is:

$$q_i = p_i - t_i$$

For packet j in the same talk spurt, play packet out at

$$p_j = t_j + q_i$$

# Adaptive playout (3)

How to determine whether a packet is the first in a talkspurt:

- If there were never loss, receiver could simply look at the successive time stamps.
  - Difference of successive stamps > 20 msec, talk spurt begins.
- But because loss is possible, receiver must look at both time stamps and sequence numbers.
  - Difference of successive stamps > 20 msec and sequence numbers without gaps, talk spurt begins.

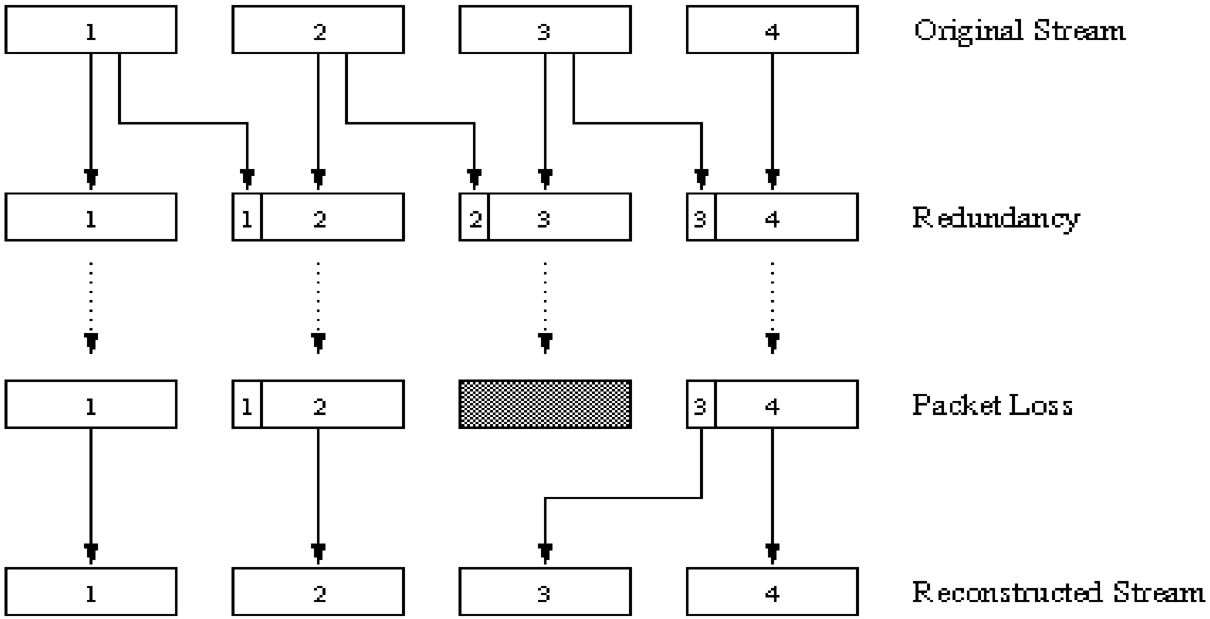
# Recovery from packet loss (1)

- Loss: packet never arrives or arrives later than its scheduled playout time

forward error correction (FEC): simple scheme

- for every group of n chunks create a redundant chunk by exclusive OR-ing the n original chunks
- send out n+1 chunks, increasing the bandwidth by factor 1/n.
- can reconstruct the original n chunks if there is at most one lost chunk from the n+1 chunks
- Playout delay needs to fixed to the time to receive all n+1 packets
- Tradeoff:
  - increase n, less bandwidth waste
  - increase n, longer playout delay
  - increase n, higher probability that 2 or more chunks will be lost

# Recovery from packet loss (2)




2nd FEC scheme

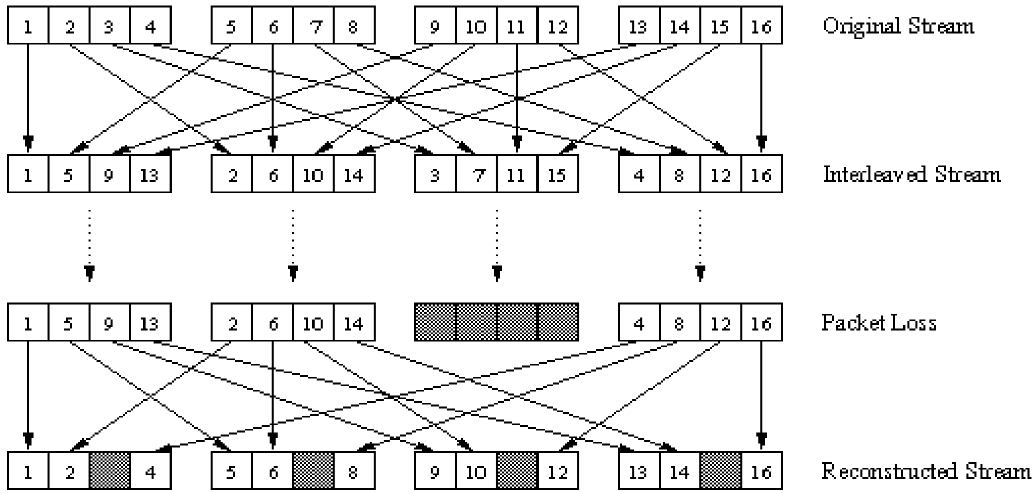
- "piggyback lower quality stream"
- send lower resolution audio stream as the redundant information
- for example, nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps.
- Sender creates packet by taking the nth chunk from nominal stream and appending to it the (n-1)st chunk from redundant stream.

- Whenever there is non-consecutive loss, the receiver can conceal the loss.
- Only one packets need to be received before playback
- Can also append (n-1)st and (n-2)nd low-bit rate chunk

# Recovery from packet loss (3)

Interleaving

- chunks are broken up into smaller units
- for example, 4 5 msec units per chunk
- interleave the chunks as shown in diagram
- packet now contains small units from different chunks




- Reassemble chunks at receiver
- if packet is lost, still have most of every chunk

# Recovery from packet loss (4)

Receiver-based repair of damaged audio streams

- produce a replacement for a lost packet that is similar to the original
- can give good performance for low loss rates and small packets (4-40 msec)
- simplest: repetition
- more complicated: interpolation