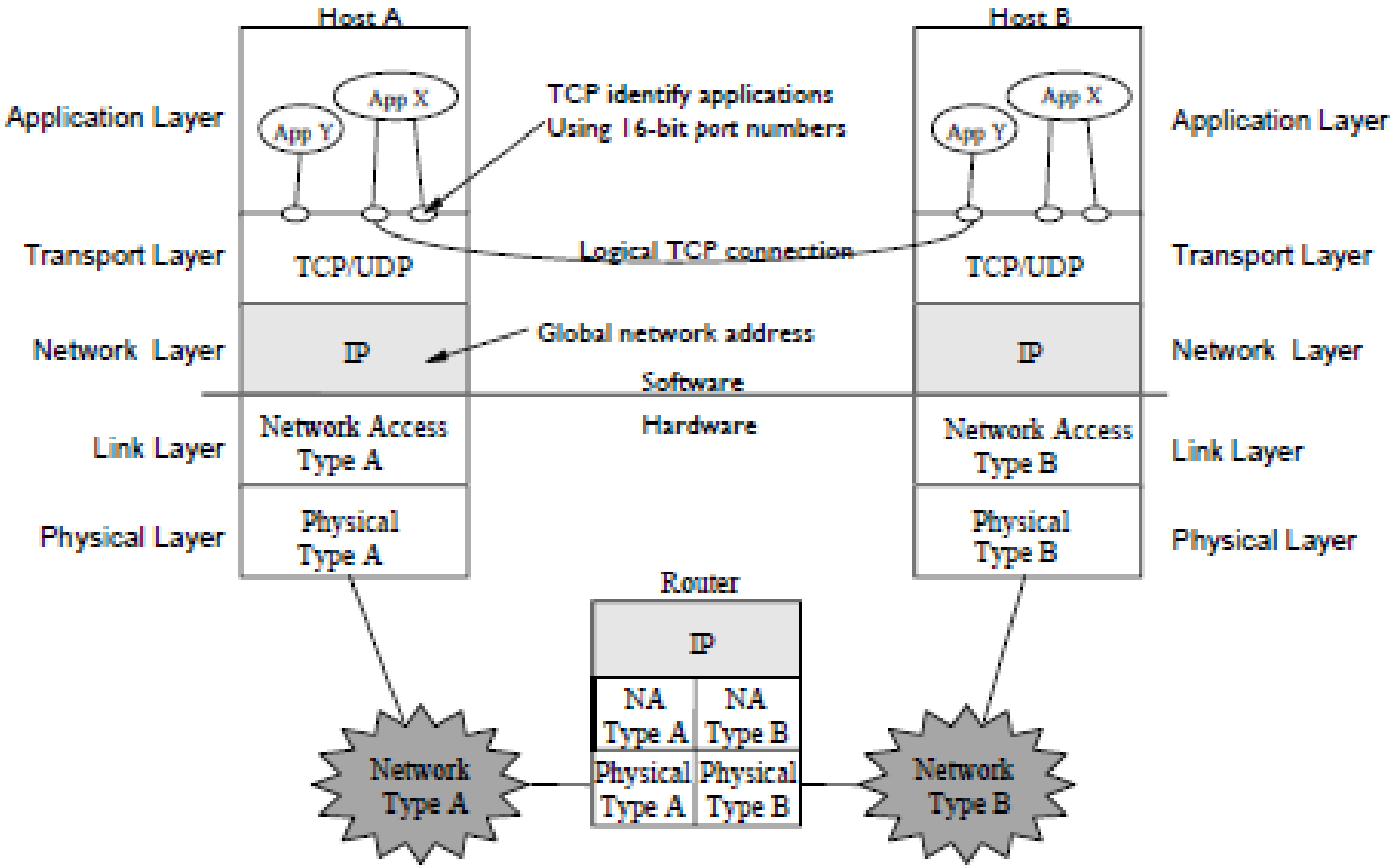
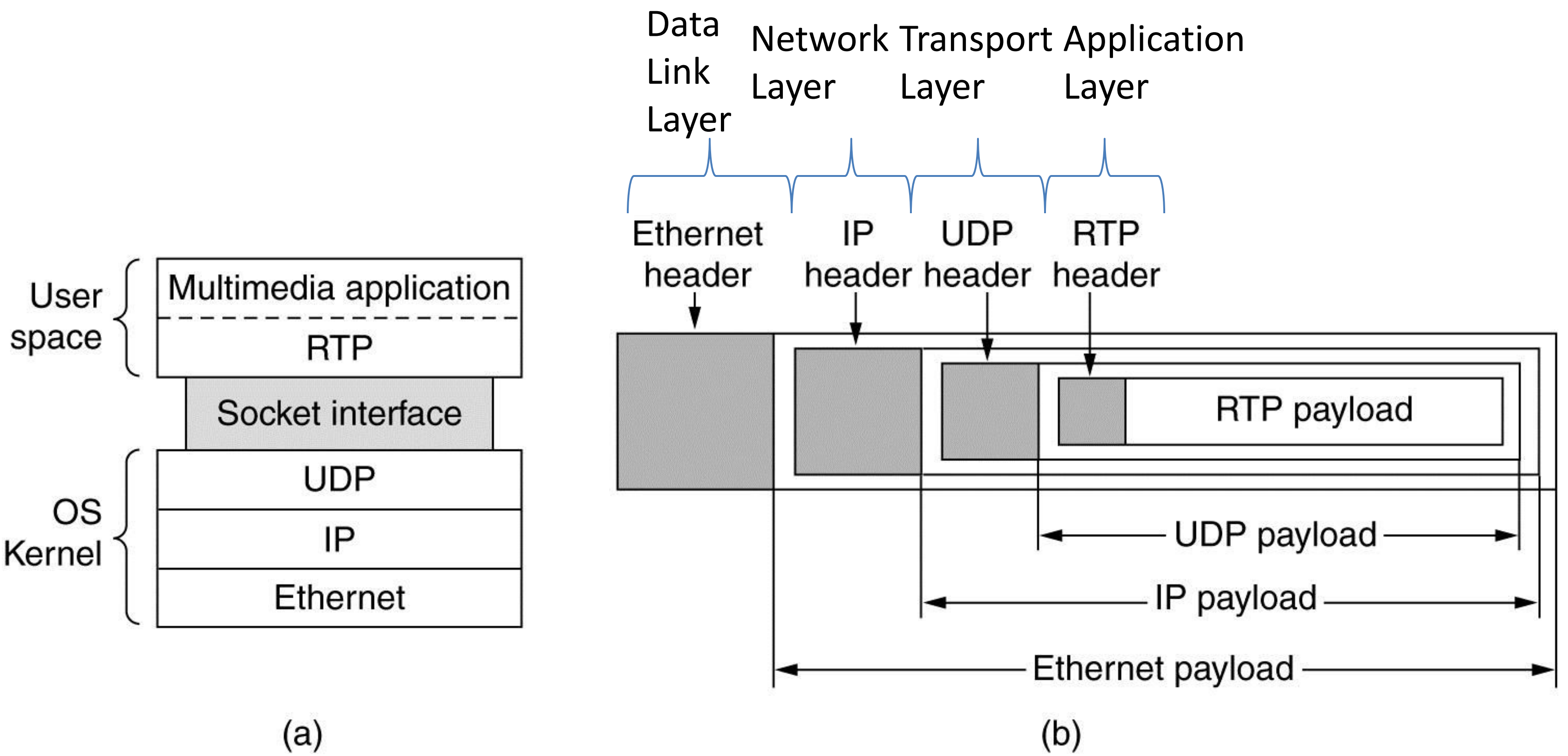
Multimedia Networking

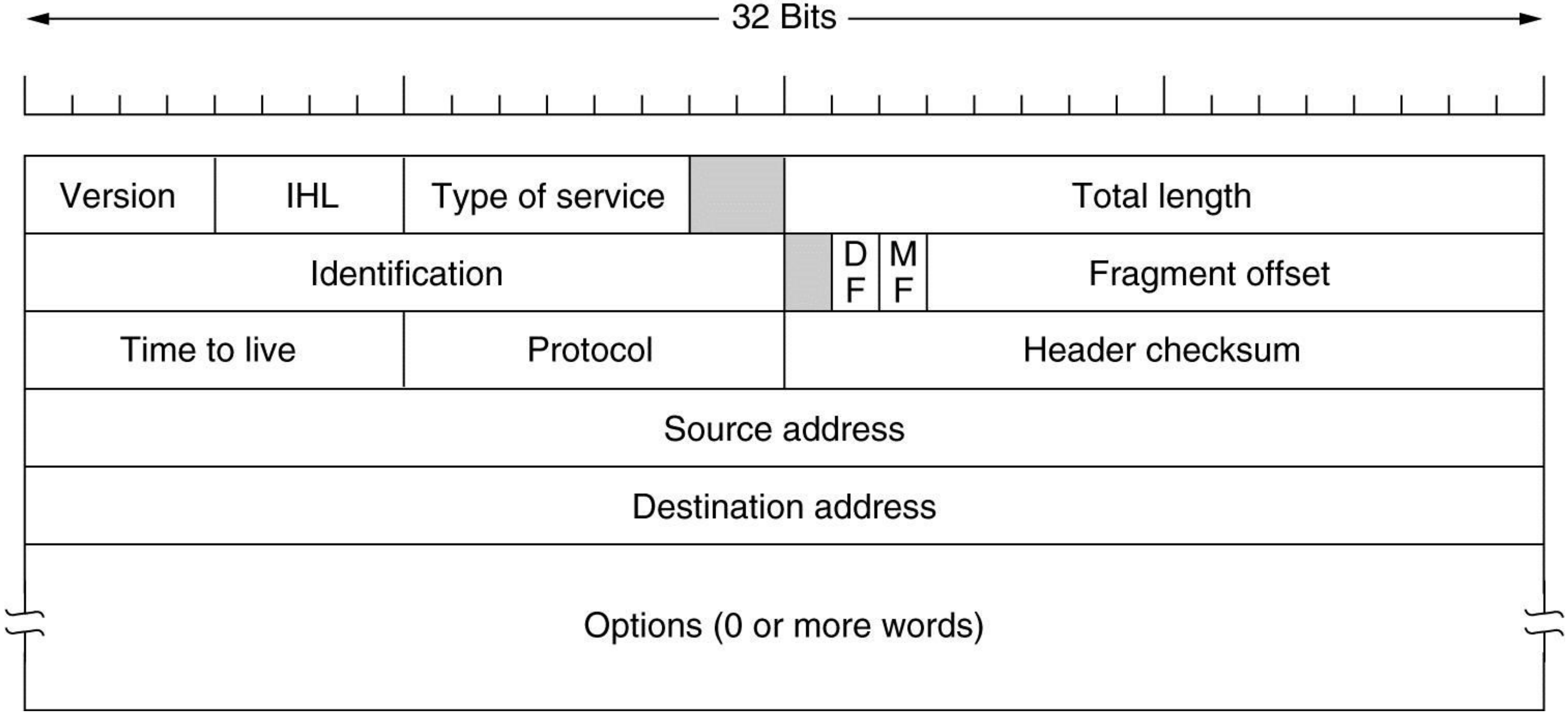
Internet Protocol Architecture



The embedding of protocol data units for multimedia transmission



The IP Protocol



UDP

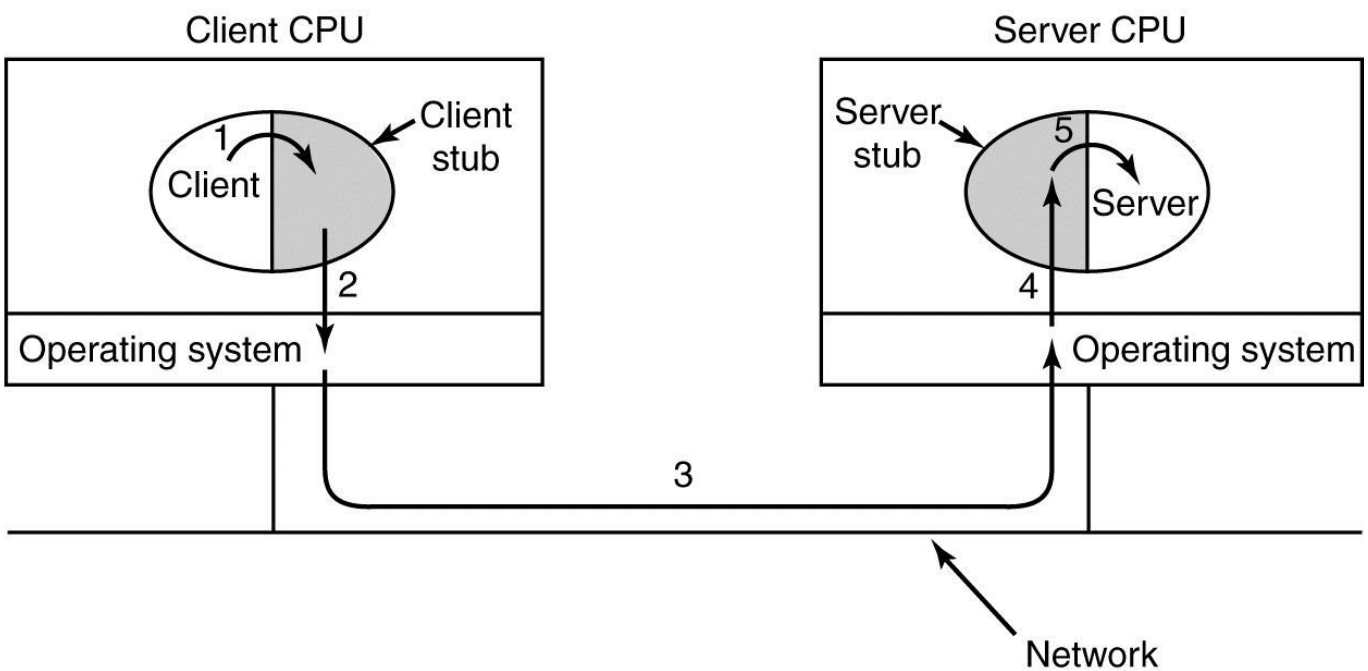
- UDP is connectionless (no flow control, no error control,
no retransmission req.
- Just IP packet with source and destination ports specified
- Demultiplexing multiple processes using ports
- Client-server request reply.

The UDP header.

Source port	Destination port
UDP length	UDP checksum

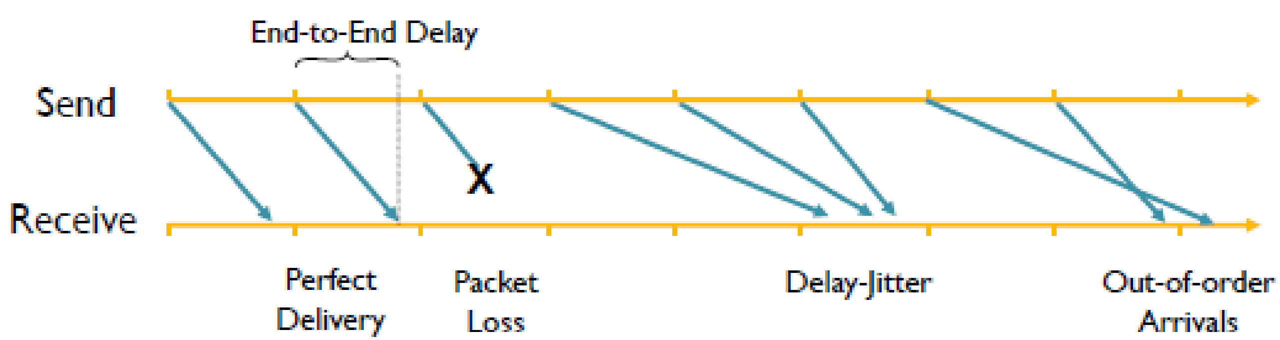
Remote Procedure Call

- Remote procedure call is just like a functional call, but execution takes place on a remote machine.
- Client is talking to server procedure through client stub which is local.
- Pointer passing is difficult. No support for global variables.



Network issues important for multimedia transmission

- Real-time interactive multimedia applications, e.g., Internet phone and real-time video conferencing are very sensitive to
  - Delay
  - Loss
  - Delay-Jitter
  - Out-of-Order Arrivals



Problems with internet

- Latency: Transmission processing delay + Queuing delays (routers) + Propagation delays + End system processing delay.
  - Voice comm.:
    - Roundtrip delay >50msec results in echo
    - One-way delay >250msec results in talker overlap
- Delay Jitters:
  - Variance of frame/packet delays, measure of smoothness of playback
  - Due to random queuing delays in the routers.
  - Might reduce jitter by buffering at the destination, but this increases average delay
- Packet Loss: Packet drop due to overflowed queues in the routers.
  - Also due to TTL expiring (packet not reaching destination for long)
- Out-of-Order Arrivals: No TCP
  - Packets taking different paths.
  - Random queue delays in the routers on different paths.
- Sync skew
  - Measure of multimedia data synchronization, i.e. audio and video
    - Lip synchronization is typically limited to 80msec (voice before video is less)

Current solutions

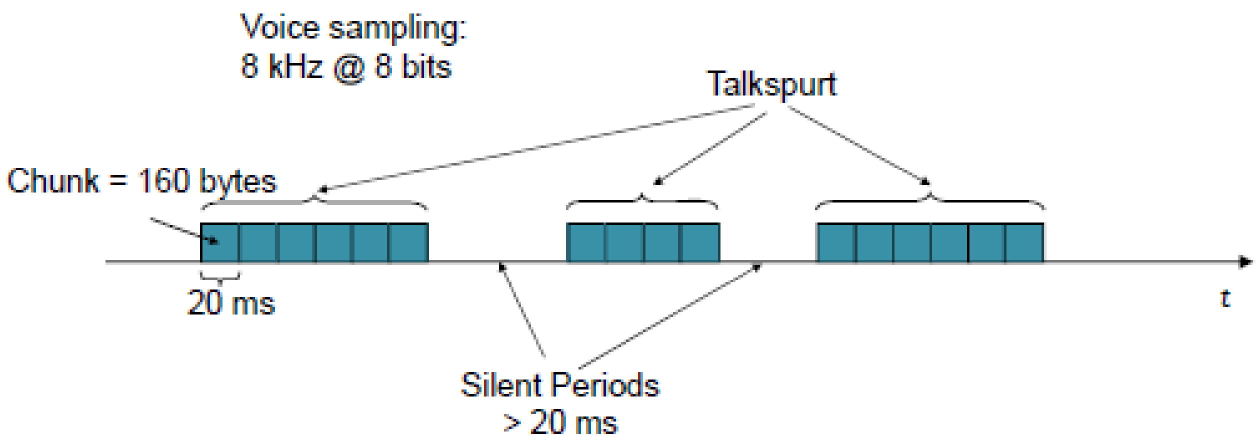
- Unlike TCP, UDP does not worry about congestion
  - No acknowledgements of correctly received frames=>don’t have to wait for acks
  - No transmitter window for frames that increases slowly in size with correct transmissions
    - In TCP slow start
      - Send 1 byte frame receive +ve ack
      - Send 2 byte frame receive +ve ack
      - Send 4 byte frame receive +ve ack, etc.

Current solutions

- Remedies for jitter
  - Content may be buffered at client (increases delay)
  - Data may be prefetched or playback stopped (Interruptions make users unhappy!)
- Remedies for synchronization
  - Timestamp packets at the sender (so receiver knows when the packets should be played back) for each medium
- Remedies for changes in currently available bandwidth
  - Compression ratio (bit rate) dynamically adapted

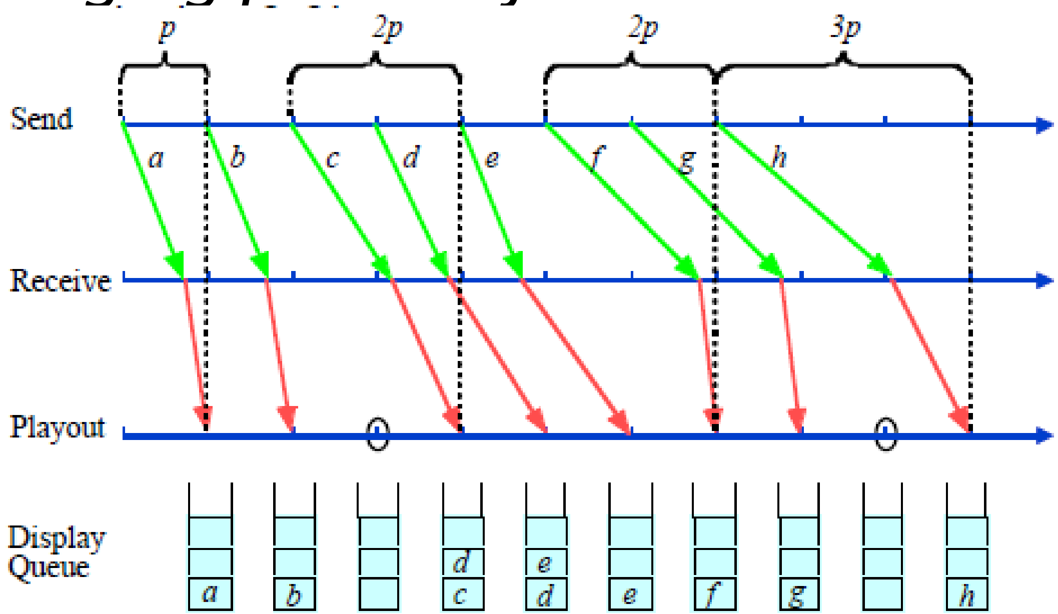
Internet Phone Example

- Application needs seq. no. and timestamp
  - RTP could be used



Reducing jitter

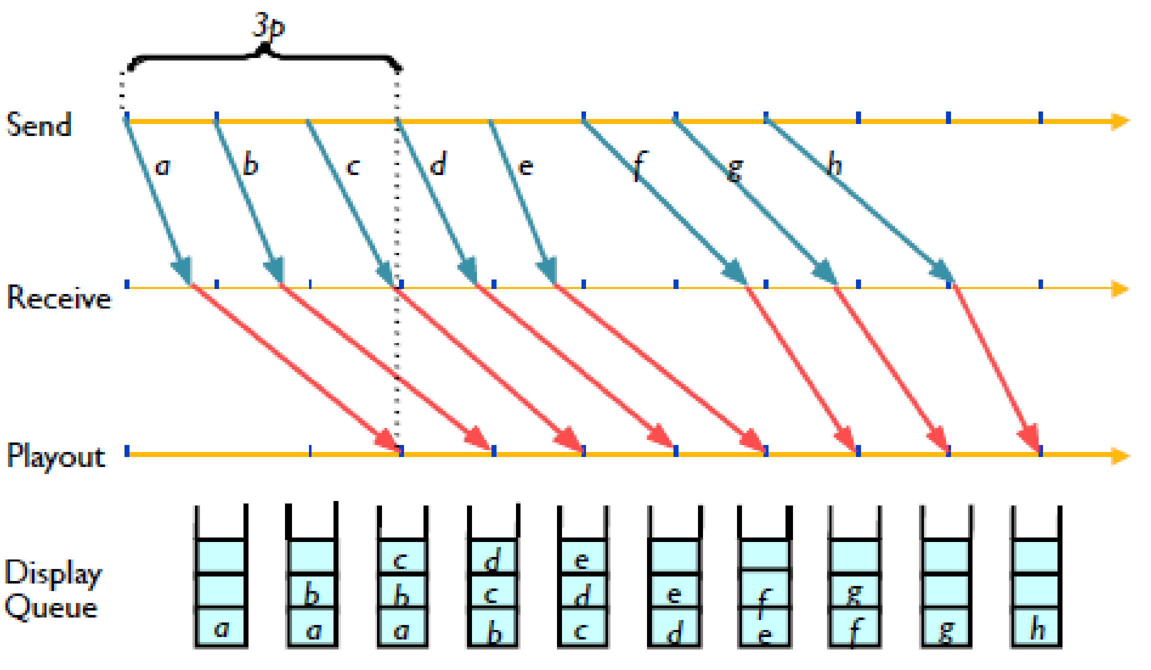
- *p = packaging period of the media at the sender*



- Playing the samples as soon as they arrive
  - ensures minimal end-to-end latency
  - has large gap variation

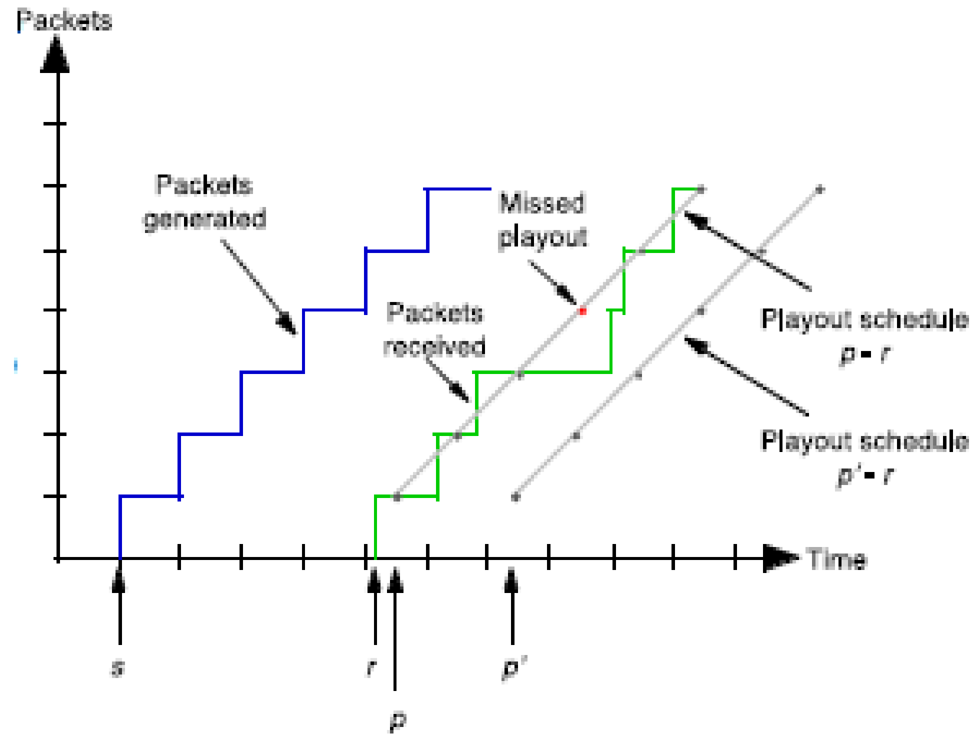
Reducing jitter

- Enqueuing the first sample for some duration better ensures continuous playout.
  - But at the cost of higher playout latency throughout the conference.



Fixed playout delay

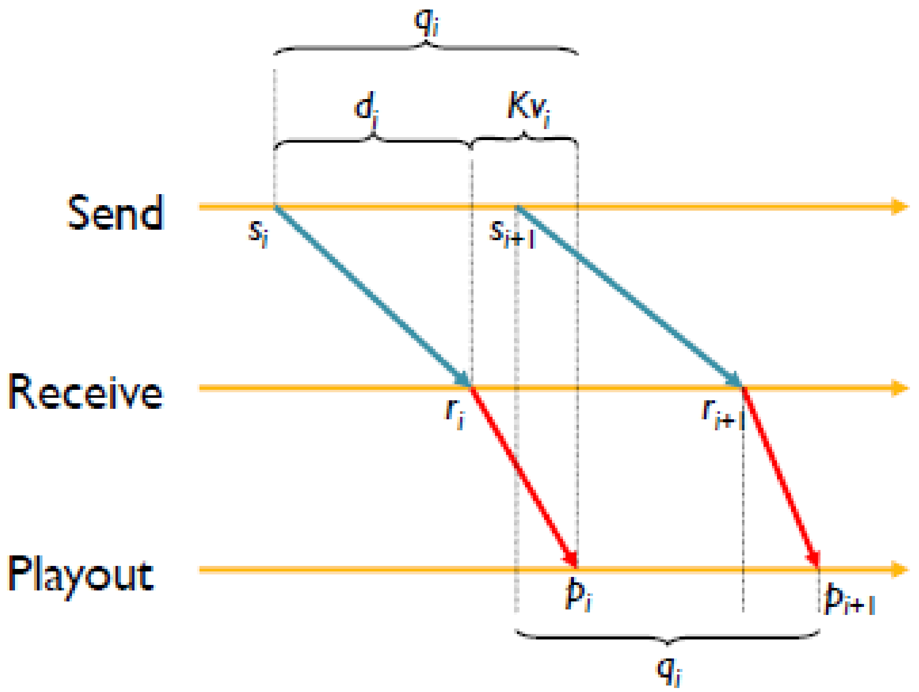
- When should it playout?
  p – r or *p' - r?*
- Playout delay should not exceed some threshold,e.g.,
  - 250 ms for audio
  - 10-15 frames/s for video (66.6 ms - 100 ms)
- Tradeoff between *playout delay vs. packet loss.*
- For interactive services, such as internet phone, long delays can be bothersome if not intolerable



Adaptive playout delay

- Typically, end-to-end delay distribution of packets within a talk spurt
  - is not known.
  - can change over time.
- Estimate the *network delay* and *its variance,* and adjust the playout delay accordingly.
- Determine play out delay on a per-talkspurt basis.

Adaptive playout delay



- *si = timestamp of the ith packet generated by the sender*
- *ri = time packet i received by the receiver*
- *pi= time packet i is played at the receiver*
- *di= average network delay*
- *vi= average deviation of network delay*

Adaptive playout delay

- End-to-end delay for the *ith* packet will be $r_i - s_i$. The average network delay of the *ith* packet, d_i, is estimated by a running average as

  $d_i = (1 - u)d_{i-1} + u(r_i - s_i)$

  where *u is fixed constant (e.g., u = 0.01)*
- The average deviation of delay is

  $v_i = (1 - u)v_{i-1} + u|r_i - s_i - d_i|$
- Playout time allows for exceeding the deviation by Kv_i

  $pi = si + di + Kvi$

  where *K* is called a congestion estimator

Adaptive playout delay

- When should playout delay be changed?
  - For voice, wait for silent periods in between talkspurts
- No packet loss case:
  - receiver can determine that a packet is the first packet of the talk spurt.
  - e.g., $s_i - s_{i-1} > 20\ ms$, then receiver knows ith packet start a new talkspurt
- Packet loss case:
  - then easily, within a talkspurt, two successive packets received may have timestamp that differ by more than 20 ms.
    - Use sequence numbers.

Continuous audio transmission

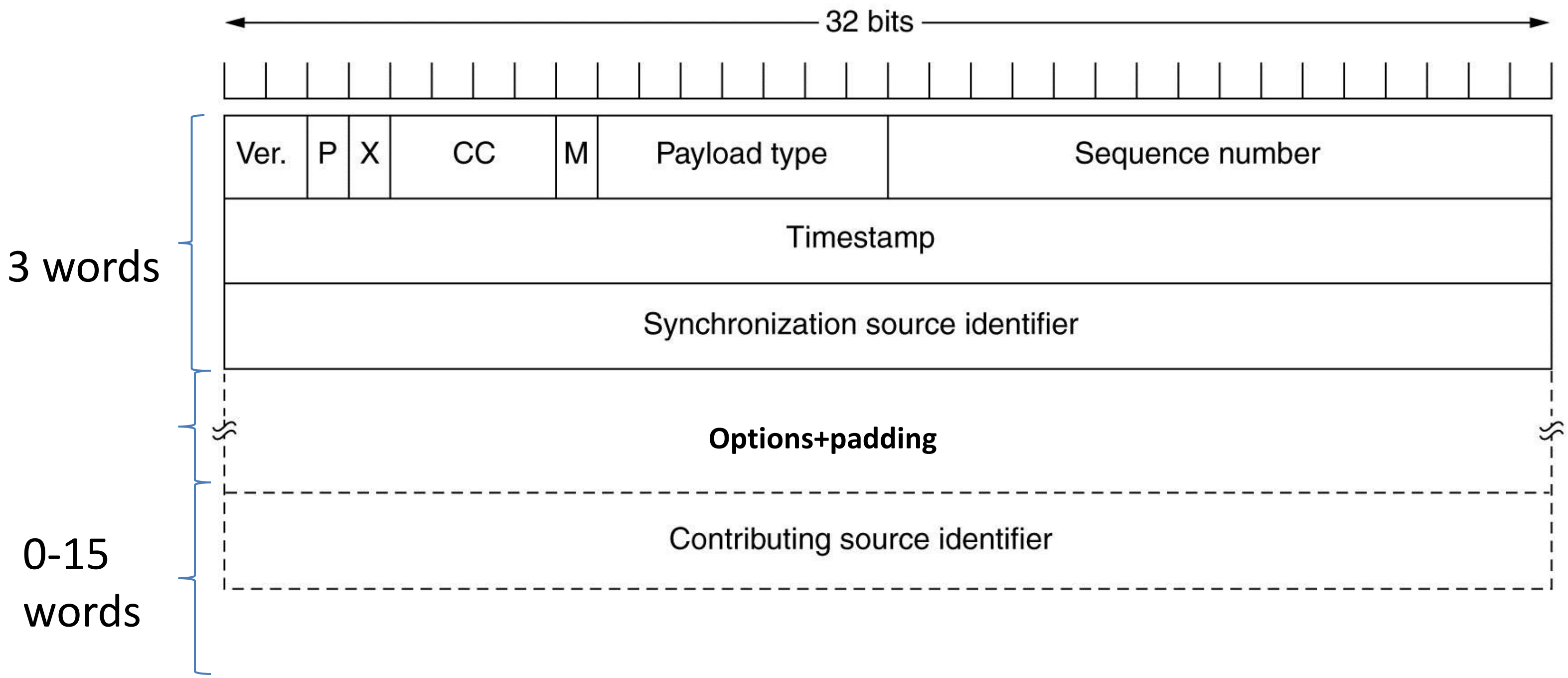
- Scheduling playout based on network delay estimates can give poor results
  - Need *E-to-E measurement* of delay-jitter .
  - Need synchronized clocks at receiver and transmitter
- Continuous multimedia: No silent periods (talkspurts) for many forms of multimedia
  - Music
  - Mixed audio streams
  - Video

Real Time Transport Protocol

- RTP is a transport protocol implemented in the application layer.
  - Runs on top of UDP (in the payload of UDP segments)
- RTP multiplexes several real time multimedia data streams onto a single stram of UDP packets.
- No flow control, error control, retransmissions, acks.
- However, timestamps, sequence numbers and info on how sources are mixed, etc.

RTP Header

- Time stamps: Enable synchronization between multiple streams and reduce jitter.
- Payload type: Encoding algorithm
- P: Padding applied to multiple of 4 bytes. X: Extension header present. CC: # contributing sources. M:application specific bit.
- Synchronization source id: which stream the packet belongs to.
- Contributing source id: the streams being mixed.



Payload Type

- Indicates the type of multimedia data carried by the packet.
  - Audio: PCM μ-law, GSM, MPEG Audio, etc.
  - Video: Motion JPEG, H.261, MPEG-1, MPEG-2, MPEG-4 H.264 AVC etc.
- Defines sampling rate of timestamp.
  - Audio
    - 8 kHz sampling rate (every 125 μs)
    - 20 ms minimum inter-frame time.
    - Marker (M) bit set on the first packet after a silence period in which no packets are sent.

Sequence no.

- Allows the receiver to detect missing and out of-ordered packets.
- 16 bits long.
- Sender increments the sequence number by one for each transmitted packet.
- RTP does not do anything when a packet is detected to be lost
  - no negative acks, timeouts, retransmissions as in a sliding window protocol
  - instead, the application decides what should be done.

Timestamp

- Enables the receiver to playback samples at appropriate intervals and to enable different media streams to be synchronized.
- 32 bits long.
- A counter of "ticks", where time between ticks depends on media format.
- Example:
  - Audio application that samples data every 125 µs use this as its clock resolution.
  - Multiple audio samples can be transmitted together.
  - If packet is generated every 10 ms (each containing 10 ms/125 µs = 80 samples), timestamp between successive packets would have time stamp differ by 80.

Timestamp vs. Sequence No

- Timestamp relates packet to real-time.
  - Timestamp values sampled from a media specific clock.
- Sequence number relates packet to other packets.
- Many packets can have the same timestamp but each should have different sequence number.

MPEG Example

- Out-of-order transmission.
  - Sequence numbers increase monotonically.
  - Timestamps reflect reference relationships.
- Large frames.
  - One video frame likely to be split into parts and packed into multiple RTP packets.
  - Timestamps associate packets together as part of the same frame, while seq. no distinguish packets from each other

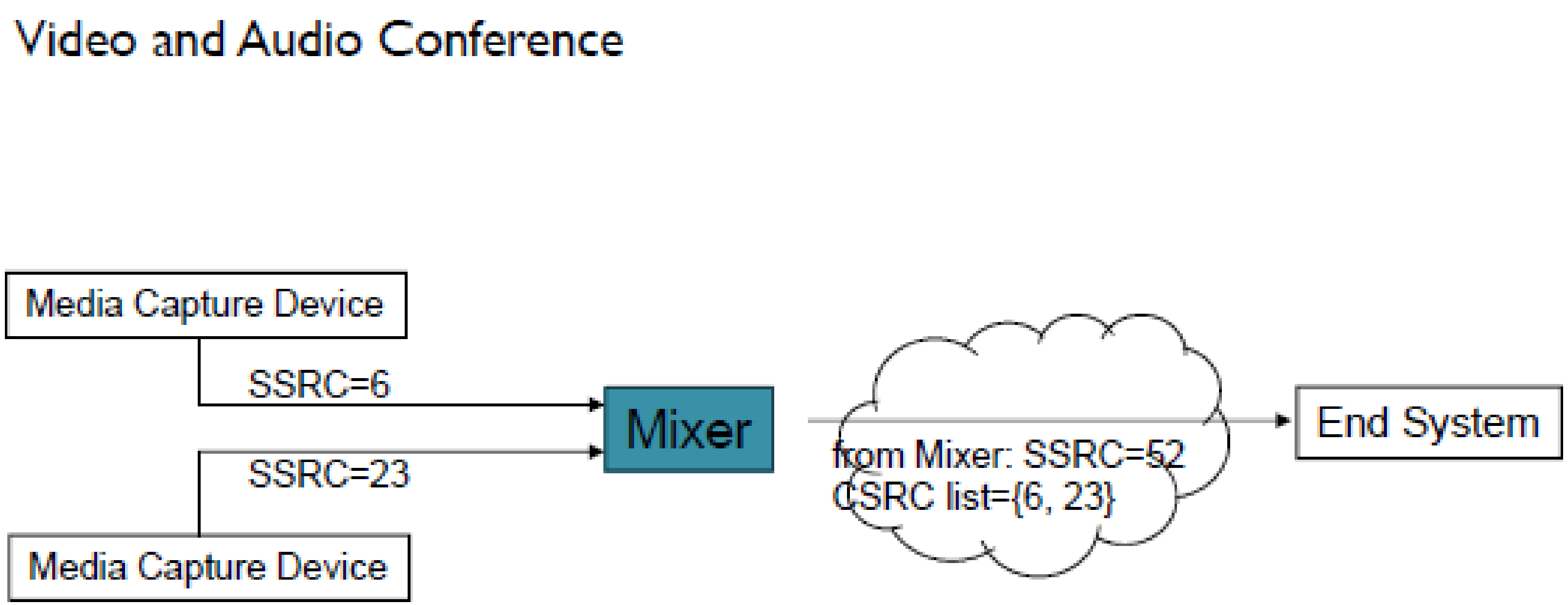
Audio Silence example

- Consider audio data type.
- How can the receiver distinguish between packet loss and silence using the timestamp/seq. no mechanism ?
- A big jump in timestamp occurs after receiving no packets for a while.
  - Lost packet detected: next packet received will not have the awaited next seq. no.
  - Silent period detected: next packet received have the awaited next seq. no.

CSRC

- SSRC value for a contributor.
- Used by a mixer to identify the contributing sources of media.
- Mixers
  - Helps in resynchronizing the incoming audio packets and forward it to the low speed link.
  - Also combines several flows in a single new one.
  - Appears as a new source.

Mixer Example



SSRC

- 32-bit number that uniquely identifies the source of the RTP stream.
  - Not an IP or TCP address.
  - All packets with the same SSRC go together.
- Each sender picks a random SSRC.
- Purpose:
  - Independence from lower-layer protocols.
  - A single node with multiple input sources (e.g.,several cameras) can distinguish those sources

Real Time Transport Control Protocol

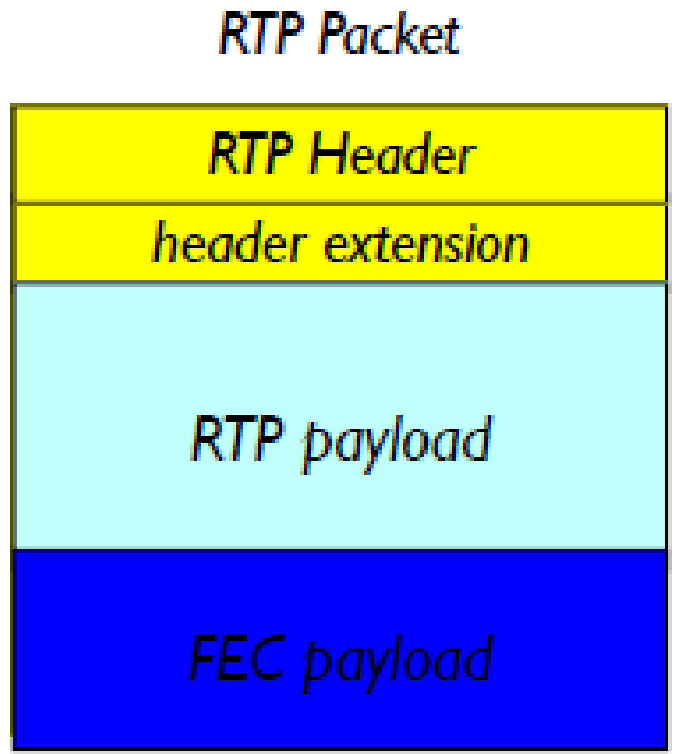
- Hand in hand with RTP.
- Provide feedback on network conditions such as delay, jitter, bandwidth,congestion, packet loss rates.
  - Adaptive encoding.
- Interstream synchronization
  - Different clocks with different granularities, drift rates.

Dealing with Packet Loss

- Packet is considered lost if it never arrives or arrives after its scheduled playout time.
- Sending packets over TCP not viable
  - Retransmission => unacceptable end-to-end delay.
  - Congestion control => slow start: transmission rate is reduced
- Packet loss rate of 1% - 20% is manageable (depending how voice is encoded and transmitted).
  - Forward Error Correction and Interleaving can help conceal loss packets.

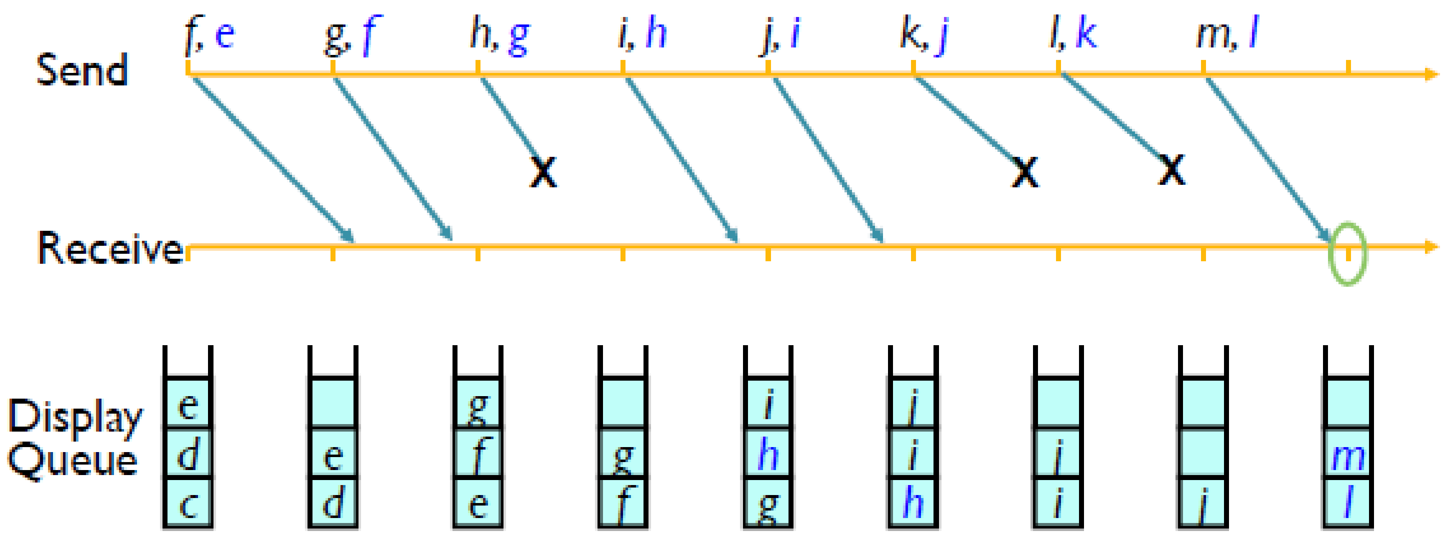
Forward Error Correction

- Redundancy is introduced into the stream to enable the receiver to recover from errors due to loss.
- Forms of redundancy
  - Simple replication and retransmission of original data.
  - *k-way XOR*
  - Replication, recoding, and re-transmission of original data.



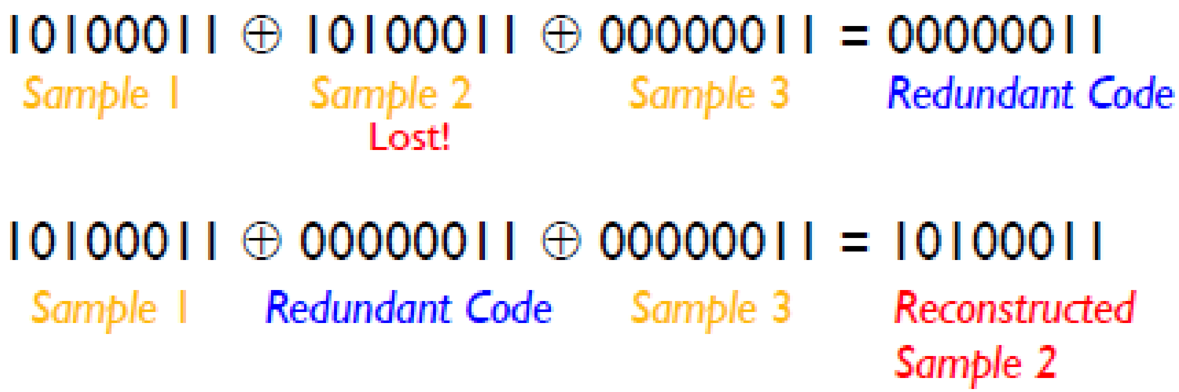
FEC Basic Concept

- Problem: If a sample is lost, how do we ensure that the redundant information necessary for the repair arrives?
  - In time.
    - Bandwidth of FEC?
    - Location of FEC in the stream?



FEC Method 1: n-way XOR

- Send a redundant encoded chunk after every n chunks.
  - Transmit the word-by-word XOR of groups of n chunks.
- A lost packet can be fully recovered at the receiver



- Example: 3-way XOR
- n too large => higher prob. of packet loss.
- n too small => higher transmission rate and delay.

Hamming code to correct burst errors

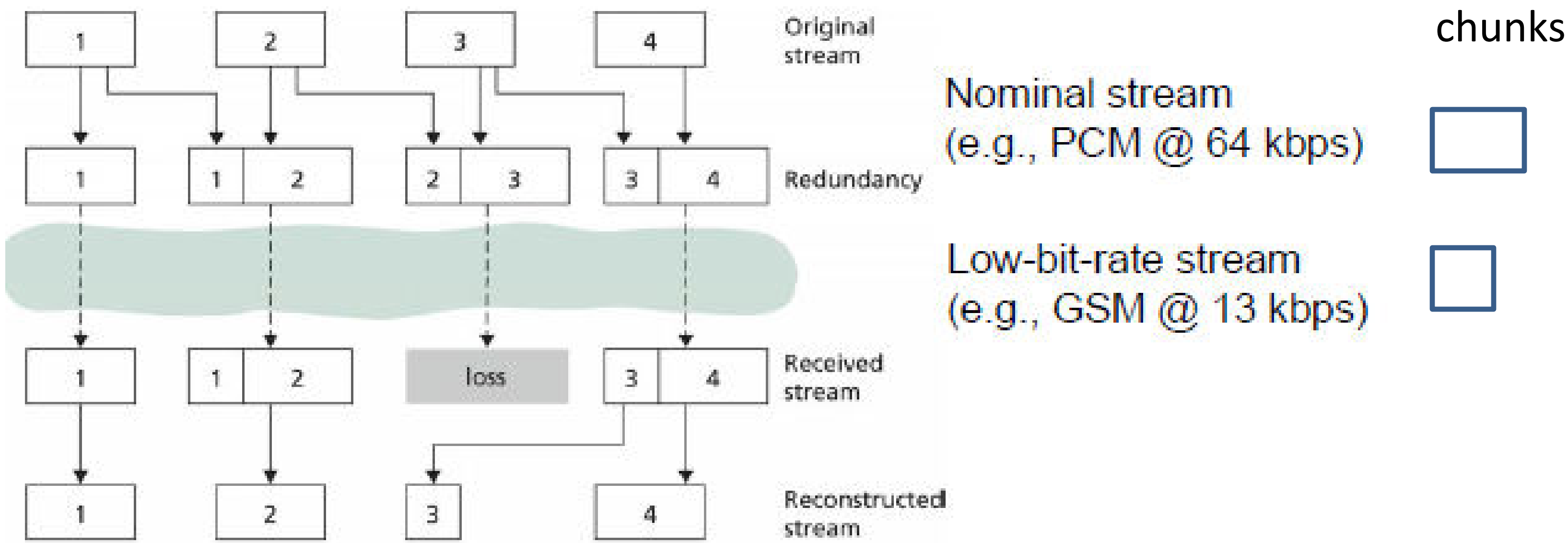
- $\lceil \log_2 m \rceil$ check bits (m=11)
- Each check bit is added to yield zero XOR (even parity)
- A bit is checked by those check bits occurring in its expansion
  - E.g.
    - Data bit in position 7=1+2+4 is checked by check bits in positions 1,2,4
    - Check bits 1,2,4 not satisfying even parity=> data bit in position 7 is in error

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

FEC Method 2: Sending Lowbit-Rate Encoded Chunks

- Occasional low-bit-rate chunks in between high-bit-rate chunks yields higher overall audio quality in the face of packet losses.

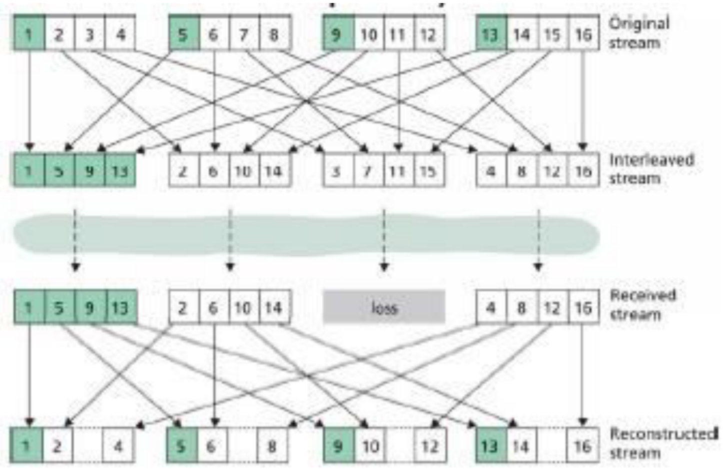


Other FEC issues

- Obvious problem with FEC?
  – Could make congestion (and thus loss) worse.
- If bandwidth is to be held constant, then original media rate is reduced resulting in terms of lower original media quality.

Interleaving

- Interleave parts of consecutive chunks to get new units
- If a unit is lost, chunks are at least partially received.



- Multiple small gaps as opposed to a single large gap!
  - Can interpolate to close small gaps
  - Low overhead, but increase in latency.
  - Not for Internet phone, but for streaming stored video.