INF3190 – Data Communication
Multimedia Protocols

Carsten Griwodz

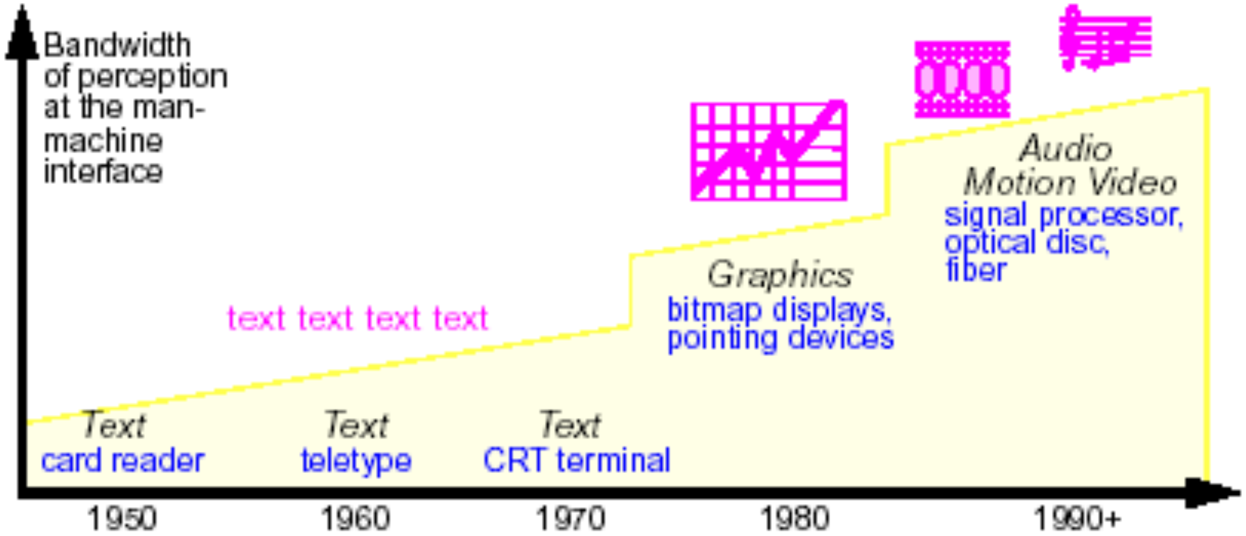
Email: griff@ifi.uio.no

Media

Medium: "Thing in the middle“

- here: means to distribute and present information

Media affect human computer interaction



The mantra of multimedia users

- Speaking is faster than writing
- Listening is easier than reading
- Showing is easier than describing

Dependence of Media

- Time-independent media
  - Text
  - Graphics
  - *Discrete* media
- Time-dependent media
  - Audio
  - Video
  - Animation
  - Multiplayer games
  - *Continuous* media
- Interdependant media
  - *Multi*media

- "Continuous" refers to the user's impression of the data, not necessarily to its representation
- Combined video and audio is multimedia - relations must be specified

Continuous Media

Fundamental characteristics

- Typically **delay sensitive**
- Often **loss tolerant**: infrequent losses cause minor glitches that can be concealed
- Antithesis of discrete media (programs, banking info, etc.), which are loss intolerant but delay tolerant

Classes of MM applications

- Streaming stored audio and video
- Streaming live audio and video
- Interactive real-time audio and video
- Interactive real-time event-driven applications

Multimedia in networks

Streaming stored MM

- Clients request audio/video files from servers and pipeline reception over the network and display
- Interactive: user can control operation (pause, resume, fast forward, rewind, etc.)
- Delay: from client request until display start can be 1 to 10 seconds

Unidirectional Real-Time

- similar to existing TV and radio stations, but delivery over the Internet
- Non-interactive, just listen/view

Interactive Real-Time

Phone or video conference

- More stringent delay requirement than Streaming & Unidirectional because of real-time nature
- Audio: < 150 msec good, < 400 msec acceptable
- Video: < 150 msec acceptable

  [Note: higher delays are feasible, but usage patterns change ***(!)***]

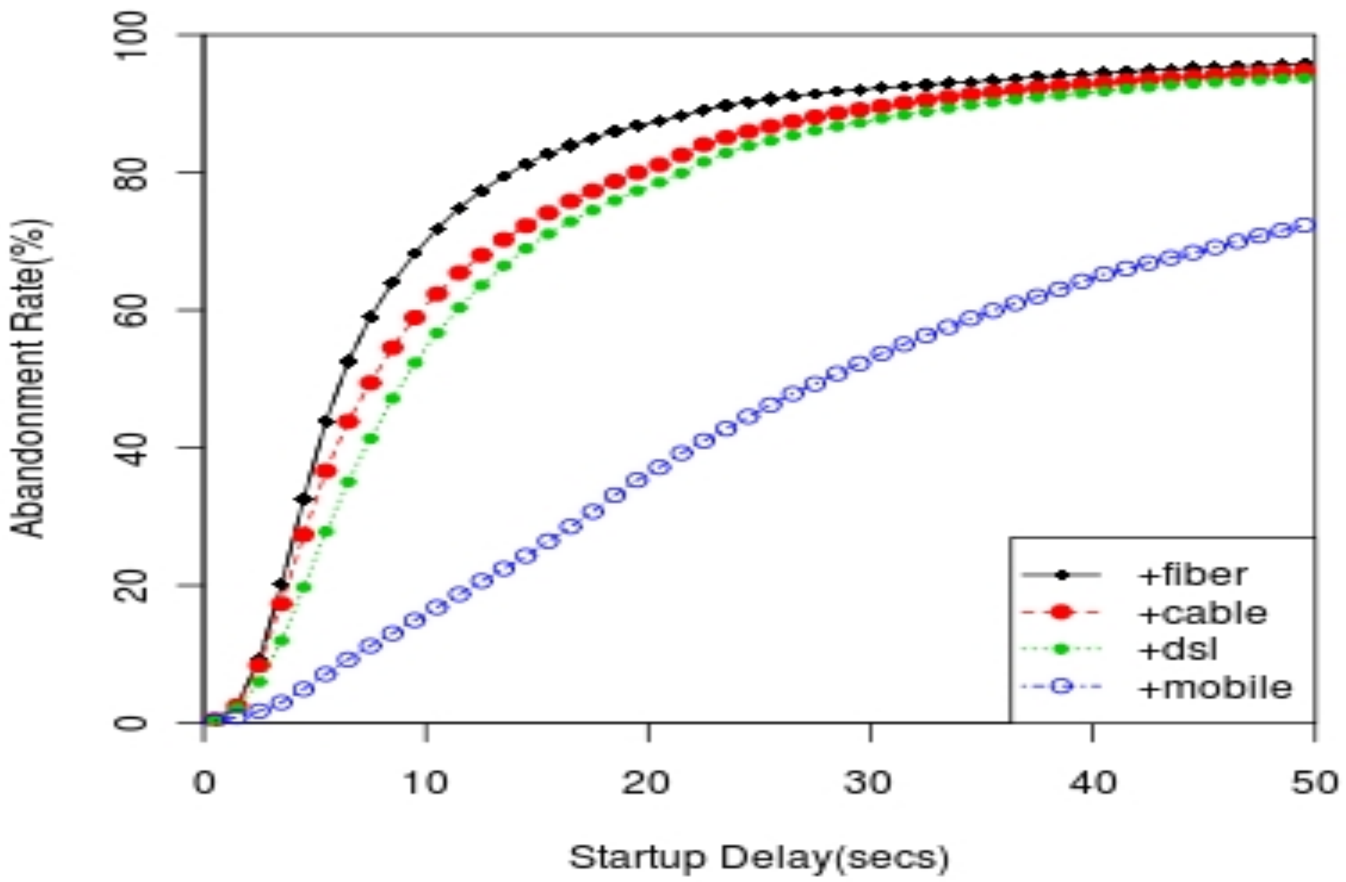
Games *(but also high-speed trading)*

- Role playing games: < 500 msec
- First person shooter (FPS) games: < 100 msec ***(may be too high)***
- Cloud gaming FPS: < 40 msec ***(estimated)***

Slides by Prof. Ramesh Sitaraman, UMass, Amherst (shown with permission)

“Video Stream Quality Impacts Viewer Behavior: Inferring Causality using Quasi-Experimental Designs”, S. S. Krishnan and R. Sitaraman, ACM Internet Measurement Conference (IMC), Boston, MA, Nov 2012

Viewers with better connectivity have less patience for startup delay and abandon sooner.



Quality of service - QoS

A term that is used in all kinds of contexts.

Be careful what it means when you hear it.

In this lecture: 3 ***classical***
parameters of **network QoS**:

- end-to-end delay
- packet loss
- jitter

end-to-end delay

- transmission time
- Σ propagation time on link l
  sum of propagation times over all links l
- Σ queueing time on router r
  sum of queueing times at all routers' queues r

packet loss

- probability of a packet to get lost
- $1 - (\Pi (P($queue at r not full$)))$
  1 – product of probabilities for all r that queue at r is not full

jitter

- variance of end-to-end delay
- estimated for several packets
- reasons
  - link layer retransmissions
  - queue length variation

Multimedia Networking

Internet without network QoS support

- Internet applications must cope with networking problems
  - Application itself or middleware
  - "Cope with" means either "*adapt to*" or "*don't care about*"
  - "Adapt to" must deal with TCP-like service variations
  - "Don't care about" approach is considered "unfair"
  - "Don't care about" approach cannot work with TCP

Internet with network QoS support

- Application must specify their needs
- Internet infrastructure must change – negotiation of QoS parameters
- Routers need more features
  - Keep QoS-related information
  - Identify packets as QoS-worthy or not
  - Treat packets differently keep routing consistent

> - approach seemed "dead" for many years
> - revival with recent Software Defined Networking (SDN) idea
> - not yet mainstream again

Non-QoS Multimedia Networking

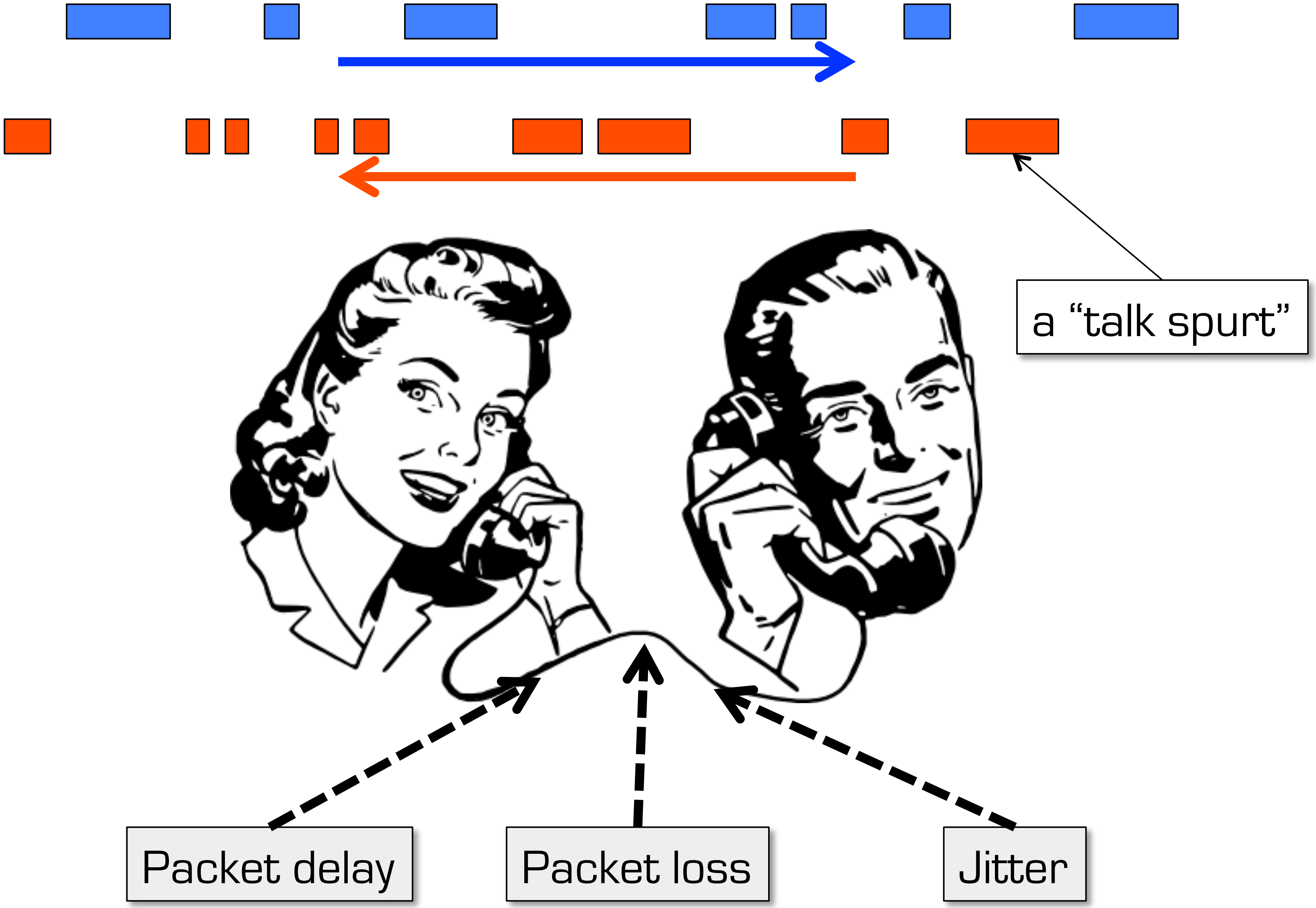
Basics

Making the best of best effort

Mitigating the impact of "best-effort" in the Internet

Use UDP to avoid TCP and its slow-start phase	… but TCP is changing (removing slow start, larger initial windows)
Buffer content at client and control playback to remedy jitter	… but not for event-based multimedia (games)
We can timestamp packets, so that receiver knows when the packets should be played back	… but applications may ignore this and look for timestamps in content
Adapt compression level to available bandwidth	… but not for event-based multimedia
We can send redundant packets to mitigate the effects of packet loss	… but retransmission and TCP may be more efficient

Streaming over best-effort networks: audio conferencing



Streaming over best-effort networks: audio conferencing

end-to-end delay

- end-to-end delay can seriously hinder interactivity; the smaller the better

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

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

Jitter compensation

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

Sequence numbers not necessary

Strategy allows for lost packets

Tradeoff for q:

- large q: less packet drop/loss (better audio quality)
- small q: better interactive experience

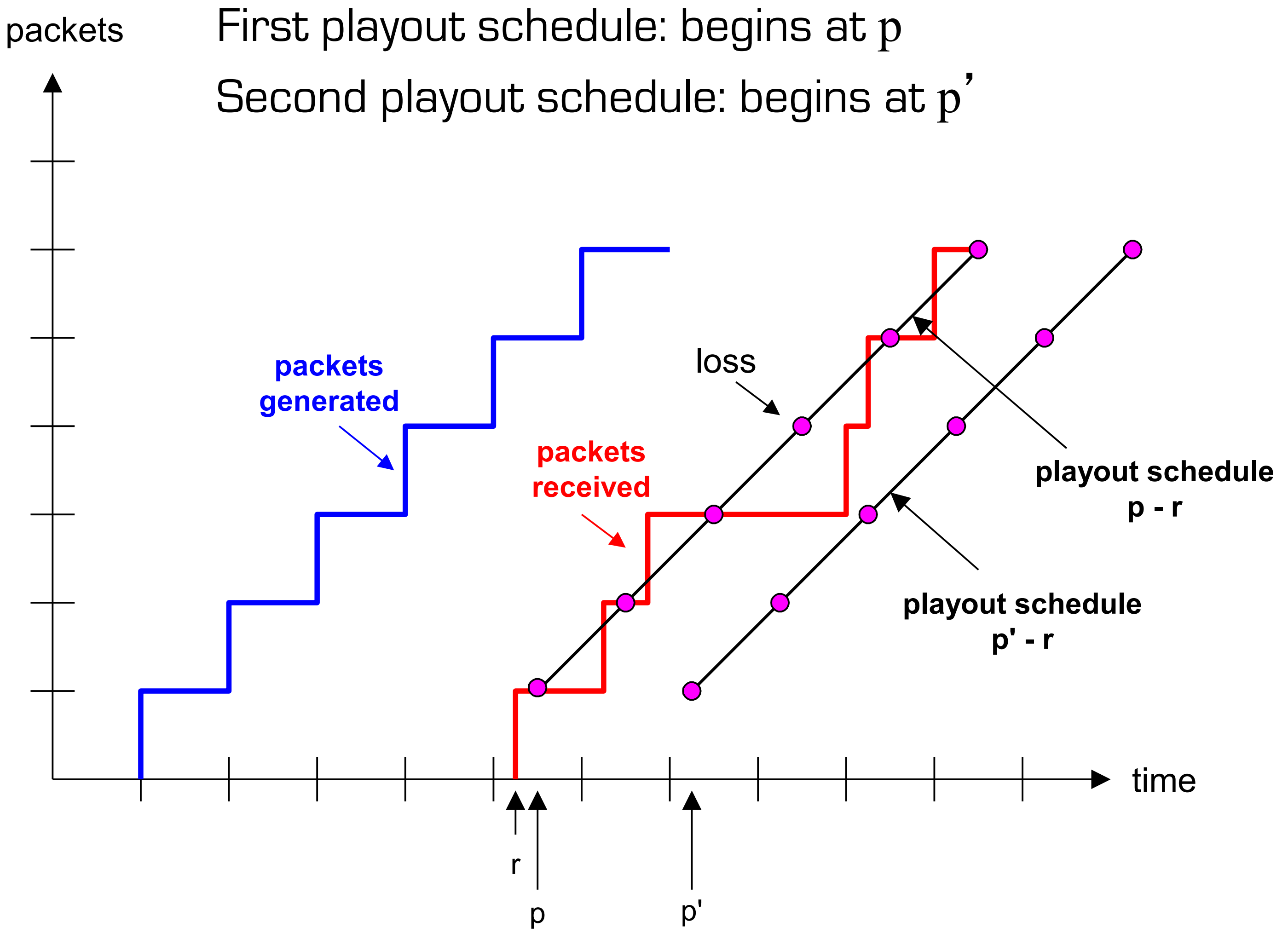
Jitter compensation

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’



Jitter compensation: Adaptive playout delay

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

Silent periods are compressed and elongated as needed

Chunks *still* played out every 20 msec during talk spurt

t_i = timestamp of the ith 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 ith packet

d_i = estimate of average network delay after receiving ith 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)

Jitter compensation: Adaptive playout delay

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

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

Deviation: How strongly does the queue length change?

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:

$$p_i = t_i + d_i + Kv_i$$

application chooses the safety margin Kv_i

where K is a positive constant

Playout delay is $q_i = p_i - t_i = d_i + Kv_i$

for this and ***all other*** packets in ***this*** talk spurt

Jitter compensation: Adaptive playout delay

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

Loss compensation

Basic assumption

- we have very little time to loose in audio conferencing
- every packet carries dozens of samples
- adding several packets delay for complex schemes is not viable

forward error correction (FEC): simple scheme

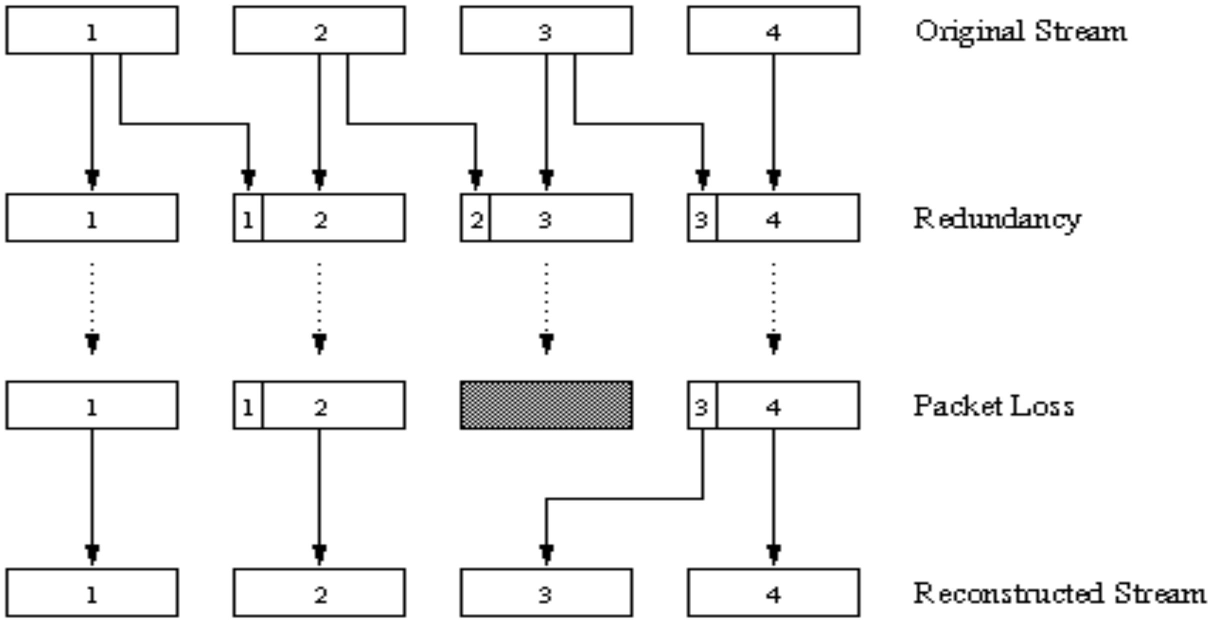
- for every group of n chunks create a redundant chunk by exclusive OR-ing the n original chunks
- send out n+1 chunks, increasing the bandwidth by factor 1/n.
- can reconstruct the original n chunks if there is at most one lost chunk from the n+1 chunks

- Playout delay needs to be 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

Loss compensation

<u>2nd FEC scheme</u>

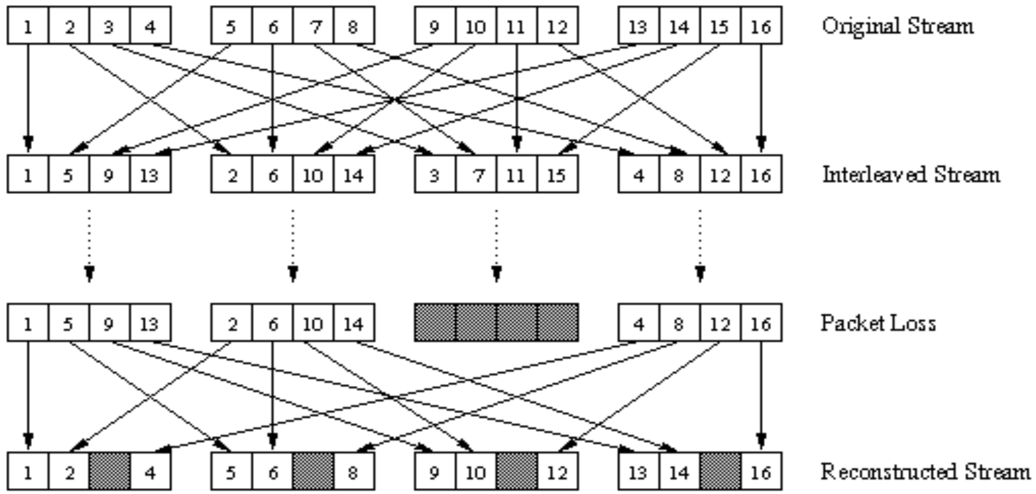
- "piggyback lower quality stream"
- send lower resolution audio stream as the redundant information
- for example, nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps.



- 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 two packets need to be received before playback
- Can also append (n-1)st and (n-2)nd low-bit rate chunk

Loss compensation



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 one packet is lost, still have most of every chunk

Loss compensation

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

Non-QoS
Multimedia Networking

Application Layer Framing &
Integrated Layer Processing

Multimedia Content Processing

- Problem: optimize transport of multimedia content

- It is application-dependent and specific
  - Application-layer processing has high overhead
  - Application processes data as it arrives from the network

- Impact of lost and mis-ordered data
  - Transport layer tries to recover from error
    - Prevents delivery of data to application
    - Prevents immediate processing as data arrives
    - Application must stop processing
  - Transport layer ignores error
    - Application experiences processing failures
    - Application must stop processing

Application Level Framing

[Clark/Tennenhouse 1990]

Give application more control

- Application understands meaning of data
- Application should have the option of dealing with a lost data
  - Reconstitute the lost data (recompute/buffer by applications)
  - Ignore the lost data

Application level framing

- Application breaks the data into suitable aggregates
  - Application Data Units (ADUs)
- Lower layers preserve the ADU frame boundaries
- ADU takes place of packet as the unit of manipulation

ALF: Application Data Units

ADUs become the unit of error recovery

- Should be upper bounded
  - loss of large ADUs is more difficult to fix
- Lower bounded
  - application semantics define smallest sensible unit
  - small ADUs mean larger protocol overhead
- Segmentation/reassembly
  - try to avoid
  - multi-TPDU ADU is wasted because one packet is lost

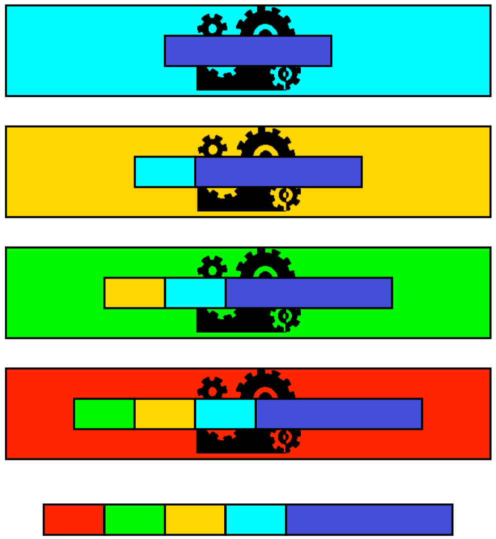
ADU "name"

- Sender computes a name for each ADU (e.g. sequence number)
- Receiver uses name to understand its place in the sequence of ADUs
- Receiver can process ADUs out of order

Integrated Layer Processing

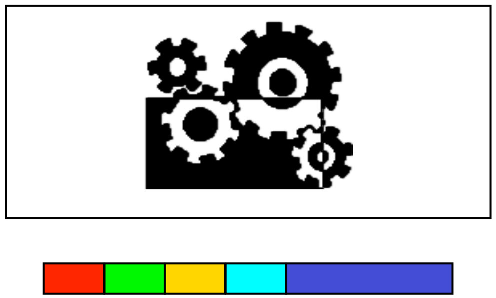
Layered engineering is not fundamental

- Assignment of functions to layers in OSI is not following fundamental principles
- Specific application may work better with different layering of functions or no layering at all
- Sequential processing through each layer
  - → Not an efficient engineering
  - → Processing all functions at once saves computing power



Integrated Layer Processing

- Vertical integration
- Performing all the manipulation steps in one or two integrated processing loops, instead of serially



Integrated Layer Processing

- Ordering constraint
  - Data manipulation can only be done after specific control steps
  - Data manipulation can only be done once the data unit is in order
  - Layered multiplexing (extract the data before it can be demultiplexed)

- Minimize inter-layer ordering constraints imposed on implementors
  - Implementors know best which data must be ordered

- Drawback: complex design due to fully customized implementation

Non-QoS
Multimedia Networking

RTP – Real-Time Transfer Protocol

Real-time Transport Protocol (RTP)

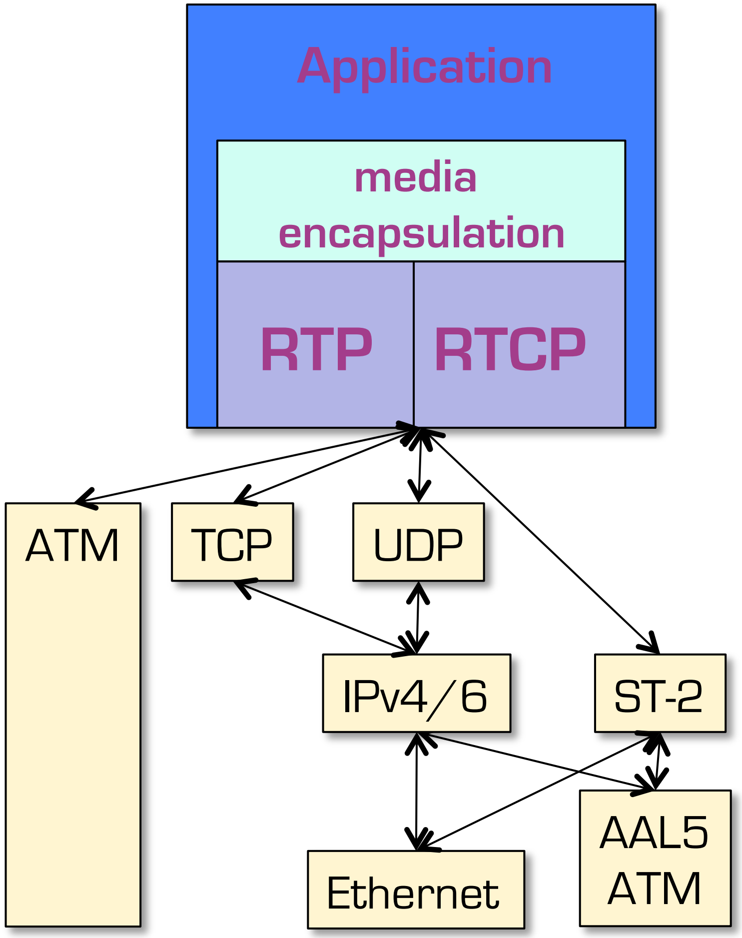
- Real-time Transport Protocol (RTP)
  - RFC 3550 (replaces RFC 1889)
  - Designed for requirements of real-time data transport
  - **NOT** real-time
  - **NOT** a transport protocol
- Two Components
  - Real-Time Transfer Protocol (RTP)
  - RTP Control Protocol (RTCP)
- Provides end-to-end transport functions
  - Scalable in multicast scenarios
  - Media independent
  - Mixer and translator support
  - RTCP for QoS feedback and session information

Real-time Transport Protocol (RTP)

- No premise on underlying resources
  - layered above transport protocol
  - no reservation / guarantees

- Integrated with applications

- RTP follows principles of
  - Application Level Framing and
  - Integrated Layer Processing



WebRTC / rtcweb

In the last 5 years,

RTP was nearly killed by HTTP Adaptive Streaming (HAS)

but Google brought it back

WebRTC

- free, open project
- adopted by Google, later Mozilla Foundation, Opera, …
- Real-Time Communications (RTC) for browsers and mobile devices through HTML5 and JavaScript APIs

rtcweb

- Real Time Collaboration on the World Wide Web
- effort standardize infrastructure for real-time communication in Web browsers
- IETF: formats and protocols
- W3C: APIs for control

RTP

- RTP services are
  - sequencing
  - synchronization
  - payload identification
  - QoS feedback and session information

- RTP supports
  - multicast in a scalable way
  - generic real-time media and changing codecs on the fly
  - mixers and translators to adapt to bandwidth limitations
  - encryption

- RTP is **not** designed for
  - reliable delivery
  - QoS provision or reservation

RTP Functions

- RTP with RTCP provides
  - support for transmission of real-time data
  - over multicast or unicast network services

- Functional basis for this
  - Loss detection – sequence numbering
  - Determination of media encoding
  - Synchronization – timing
  - Framing - “guidelines” in payload format definitions
  - Encryption
  - Unicast and multicast support
  - Support for stream “translation” and “mixing” (SSRC; CSRC)

RTP Packet Format

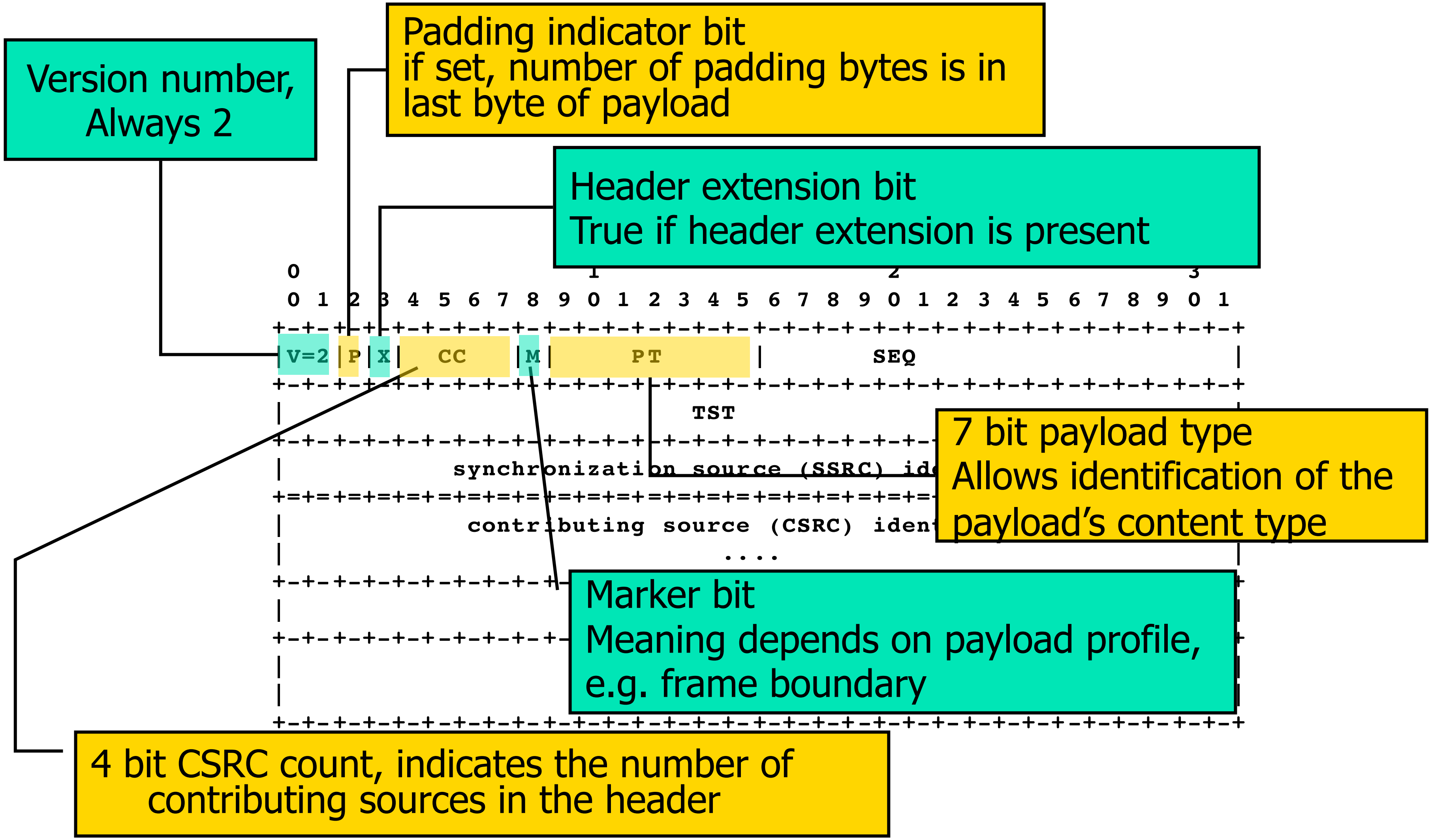
Typical IETF RFC bit-exact representation

a longword (32 bit)

a byte

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |          SEQ                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            TST                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            header extension                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            payload (audio, video, ...)        |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RTP Packet Format



RTP Packet Format

32 bit timestamp

16 bit sequence number

32 bit SSRC
Synchronization source identifier, a random number identifying the sender

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |             SEQ               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             TST                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       header extension                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

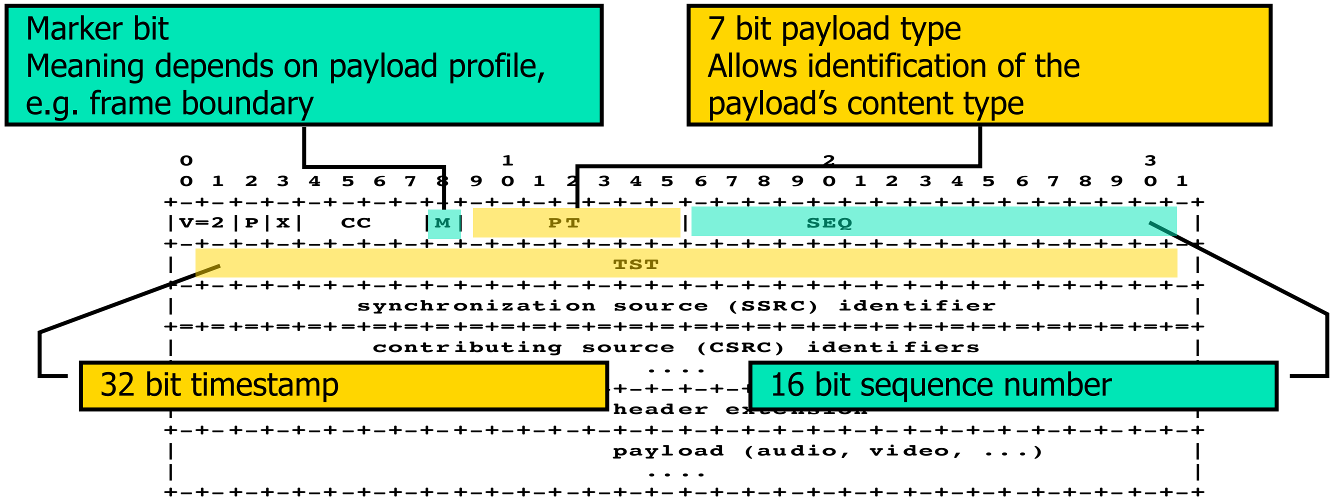
Header extension multiples of 32 bit

Several 32 bit CSRC
Contribution source identifier, the number is indicated by CC
A mixer copies the original sources' SSRCs here

RTP Architecture Concepts

Integrated Layer Processing

- Typical for layered processing
  - Data units sequentially processed by each layer
- Integrated layer processing
  - Adjacent layers tightly coupled
- Therefore, RTP is not complete by itself: requires application-layer functionality/ information in header



RTP Packet Format

- Relatively long header (>40 bytes)
  - overhead carrying possibly small payload
  - header compression
  - other means to reduce bandwidth (e.g. silence suppression)

- No length field
  - Exactly one RTP packet carried in UDP packet
  - When you use RTP with TCP or SCTP or RTSP or ATM AAL5:
    - do-it-yourself packaging

- Header extensions for payload specific fields possible
  - Specific codecs
  - Error recovery mechanisms

RTP Profile (RFC 1890)

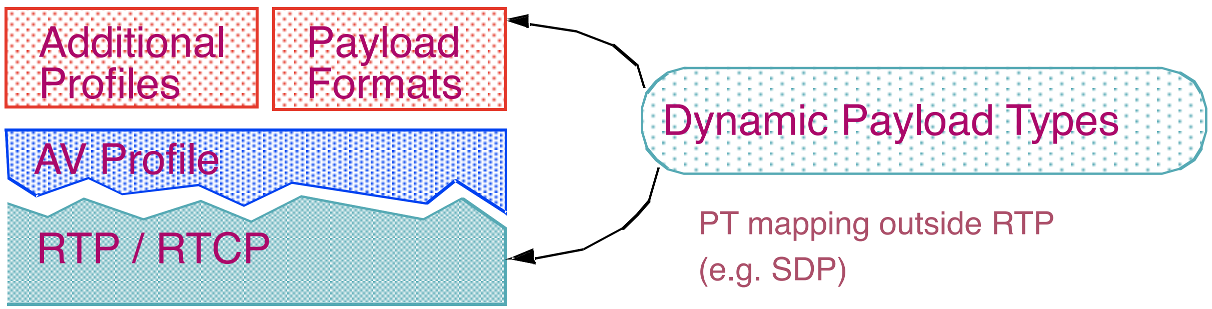
- Set of standard encodings and payload types
  - Audio: e.g. PCM-u, GSM, G.721
    (for WebRTC G.711 and Opus are mandatory)
  - Video: e.g. JPEG, H.261
    (for WebRTC H.264 and VP8 are mandatory)

- Number of samples or frames in RTP packet
  - Sample-based audio: no limit on number of samples
  - Frame-based audio: several frames in RTP packet allowed

- Clock rate for timestamp
  - Packetized audio: default packetization interval 20 ms
  - Video: normally 90 kHz, other rates possible

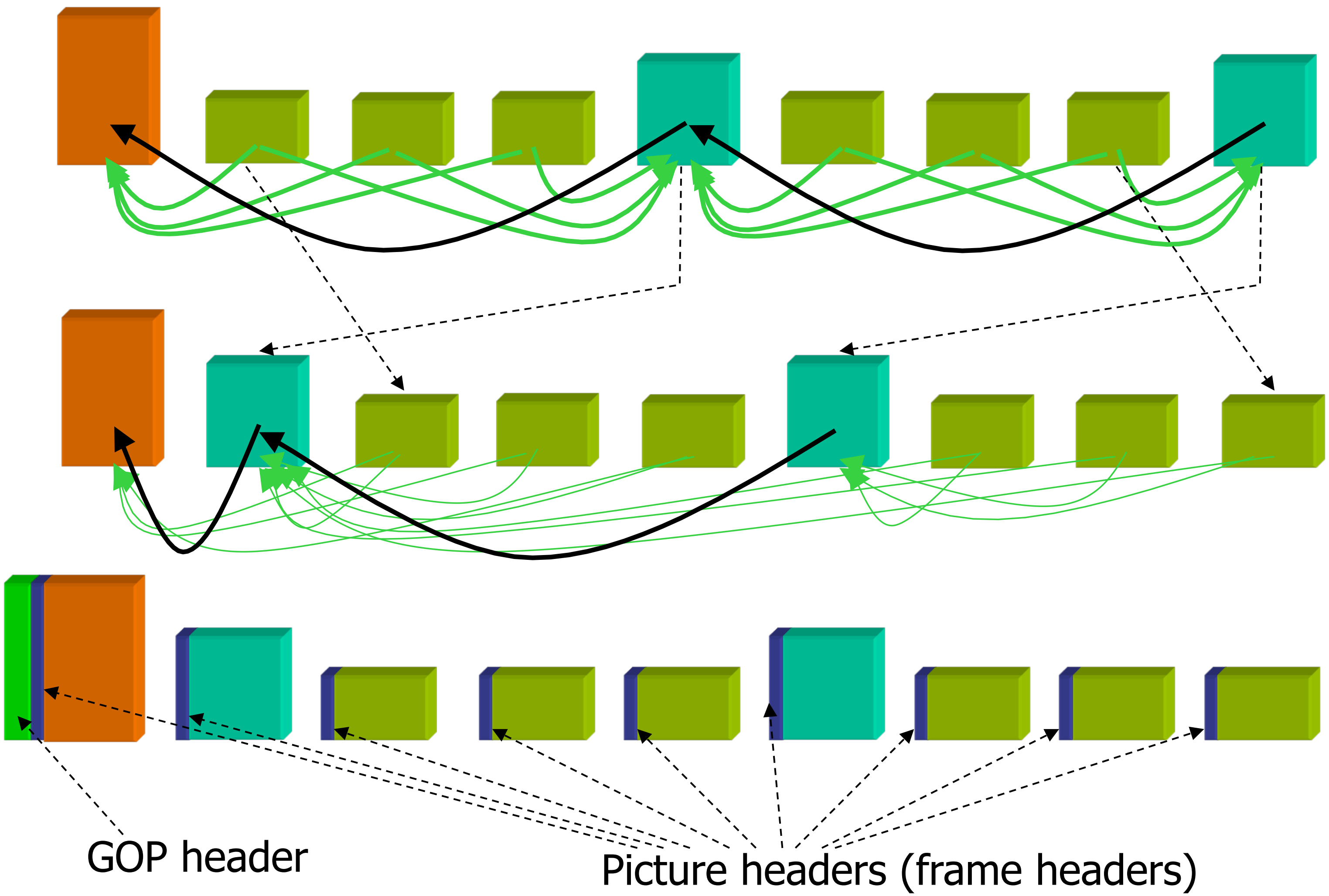
RTP Profiles

- Payload type identification
  - RTP provides services needed for generic A/V transport
    - Particular codecs with additional requirements
    - Payload formats defined for each codec: syntax and semantic of RTP payload
  - Payload types
    - Static: RTP AV profile document
    - Dynamic: agreement on per-session basis
- Profiles and Payload Formats in RTP Framework



RTP Profile for MPEG-1 Video Payload

Note: MPEG-4 profile for RTP exists, but is much more complex due to H.264's 16-way dependencies.



RTP Profile for MPEG-1 Video Payload

- Fragmentation rules
  - Video sequence header
    - if present, starts at the beginning of an RTP packet
  - GOP sequence header
    - Either at beginning of RTP packet
    - Or following video sequence header
  - Picture header
    - Either at beginning of RTP packet
    - Following GOP header
  - No header can span packets

- Marker Bit
  - Set to 1 if packet is end of picture

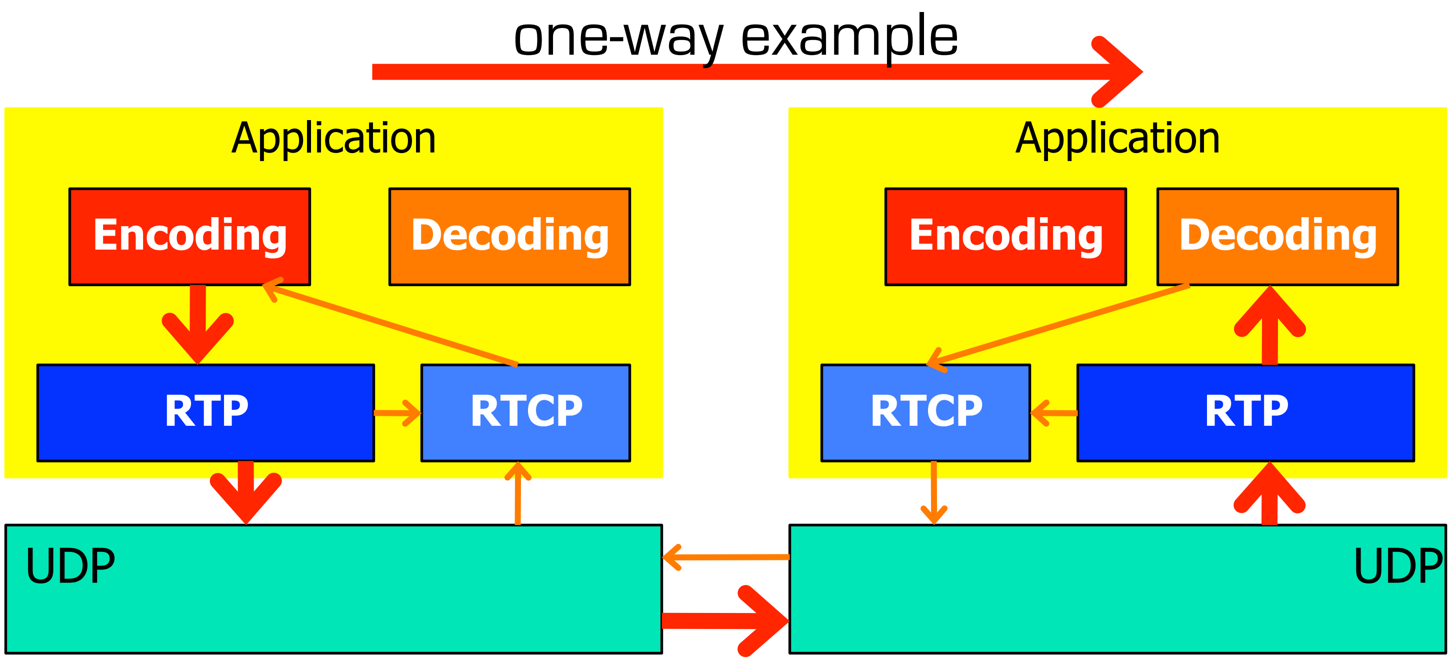
RTP Profile for MPEG-1 Video Payload

MPEG Video Profile

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MBZ    |           TR          |MBZ|S|B|E|  P  | | BFC | | FFC |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                FBV     FFV
```

- MPEG-1 Video specific payload header
- TR
  - Temporal reference
  - The same number for all packets of one frame
  - For ordering inside an MPEG GOP
- MBZ
  - Must be zero
- S
  - 1 if sequence header is in this packet
- B
  - 1 if payload starts with new slice
- E
  - 1 if last byte of payload is end of slice
- P
  - 3 bits that indicate picture type (I, P, B or D)
- FBV, BFC, FFV, FFC
  - Indicate how a P or B frame is related to other I and P frames (copied from last frame header)

RTP-enabled Quality Adaptation



- Component interoperations for control of quality
- Evaluation of sender and receiver reports
- Modification of encoding schemes and parameters
- Adaptation of transmission rates
- Hook for possible retransmissions (outside RTP)

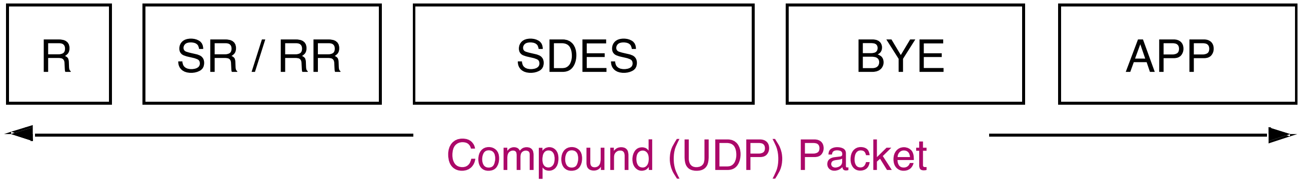
RTP Control Protocol (RTCP)

Companion protocol to RTP (tight integration with RTP)

- Monitoring
  - of QoS
  - of application performance
- Feedback to members of a group about delivery quality, loss, etc.
  - Sources may adjust data rate
  - Receivers can determine if QoS problems are local or network-wide
- Loose session control
  - Convey information about participants
  - Convey information about session relationships
- Automatic adjustment to overhead
  - report frequency based on participant count

Typically, “RTP does …” means “RTP with RTCP does …”

RTCP Packets



- Several RTCP packets carried in one compound packet
- RTCP Packet Structure
  - SR Sender Report (statistics from active senders: bytes sent -> estimate rate)
  - RR Receiver Report (statistics from receivers)
  - SDES Source Descriptions (sources as “chunks” with several items like canonical names, email, location,...)
  - BYE explicit leave
  - APP extensions, application specific

RTP Mixer

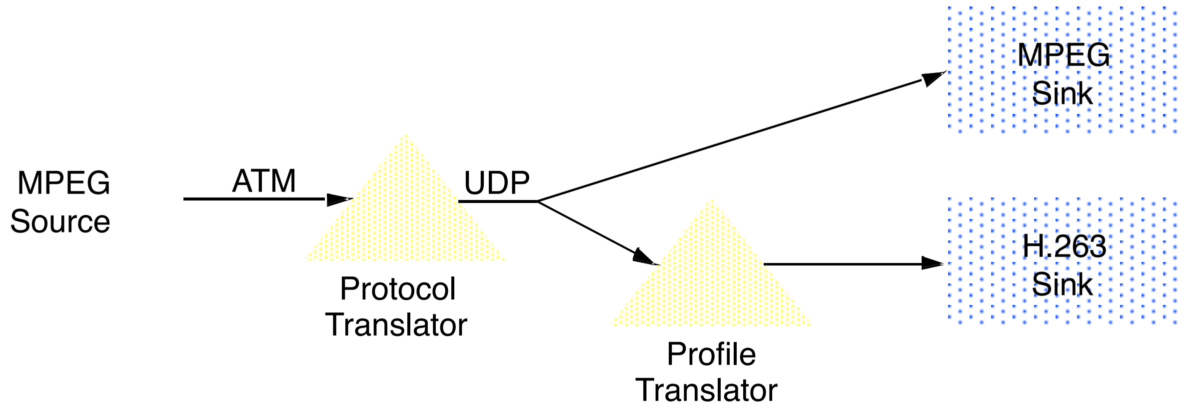
Mixer idea

- If everybody in a large conference talks at the same time, understand it anyway impossible
- Implement in conference bridges
- Reduce bandwidth in large conferences by mixing several speakers into one stream

Mixer tasks

- Reconstruct constant spacing generated by sender (jitter reduction)
- Translate audio encoding to a lower-bandwidth
- Mix reconstructed audio streams into a single stream
- Resynchronize incoming audio packets
  - New synchronization source value (SSRC) stored in packet
  - Incoming SSRCs are copied into the contributing synchronization source list (CSRC)
- Forward the mixed packet stream

RTP Translator



Translation between protocols

- e.g., between IP and ST-2

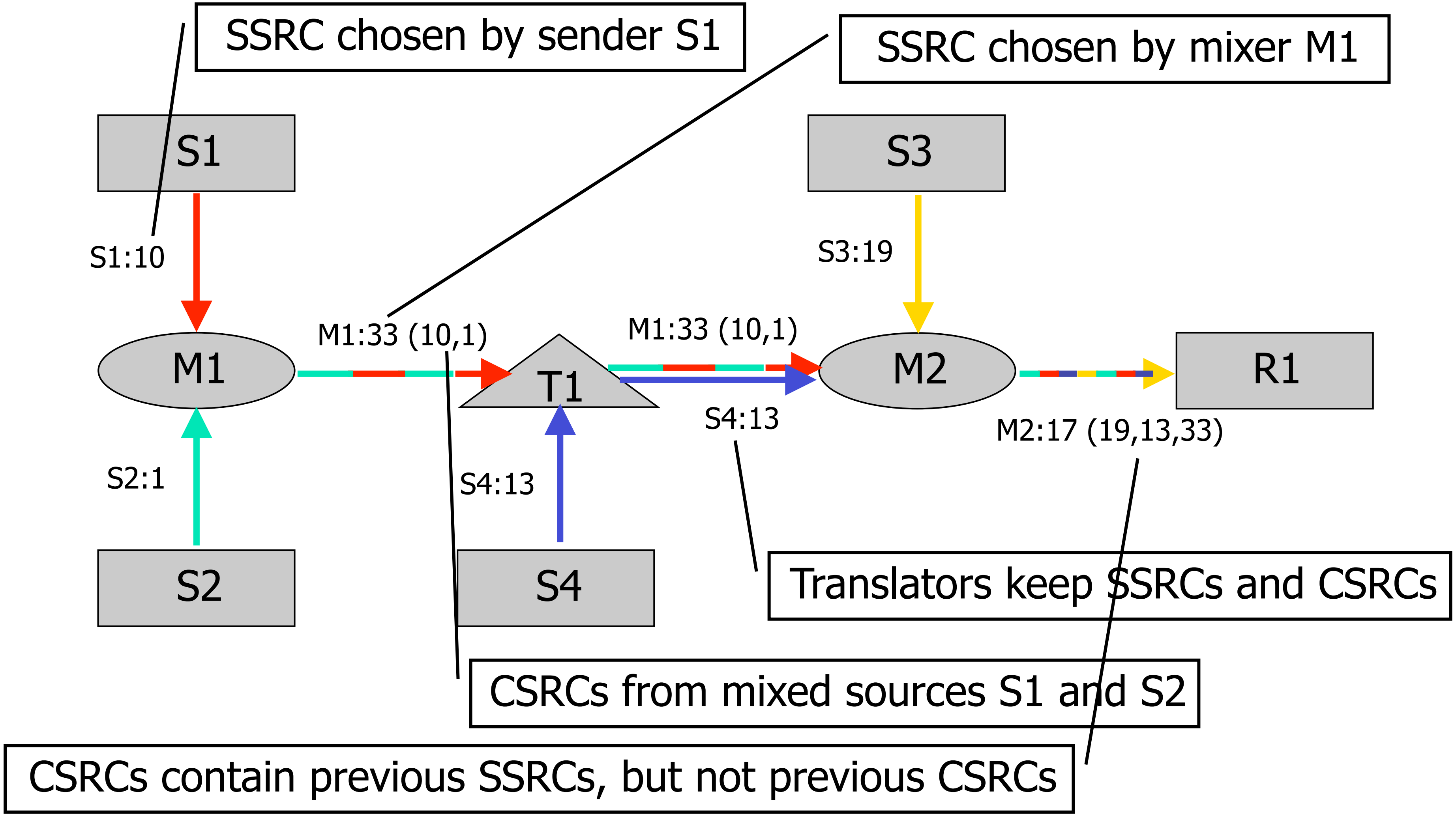
Translation between encoding of data

- e.g. H.265 to H.263
- for reduction of bandwidth without adapting sources

No resynchronization in translators

- SSRC and CSRC remain unchanged

RTP Identifiers



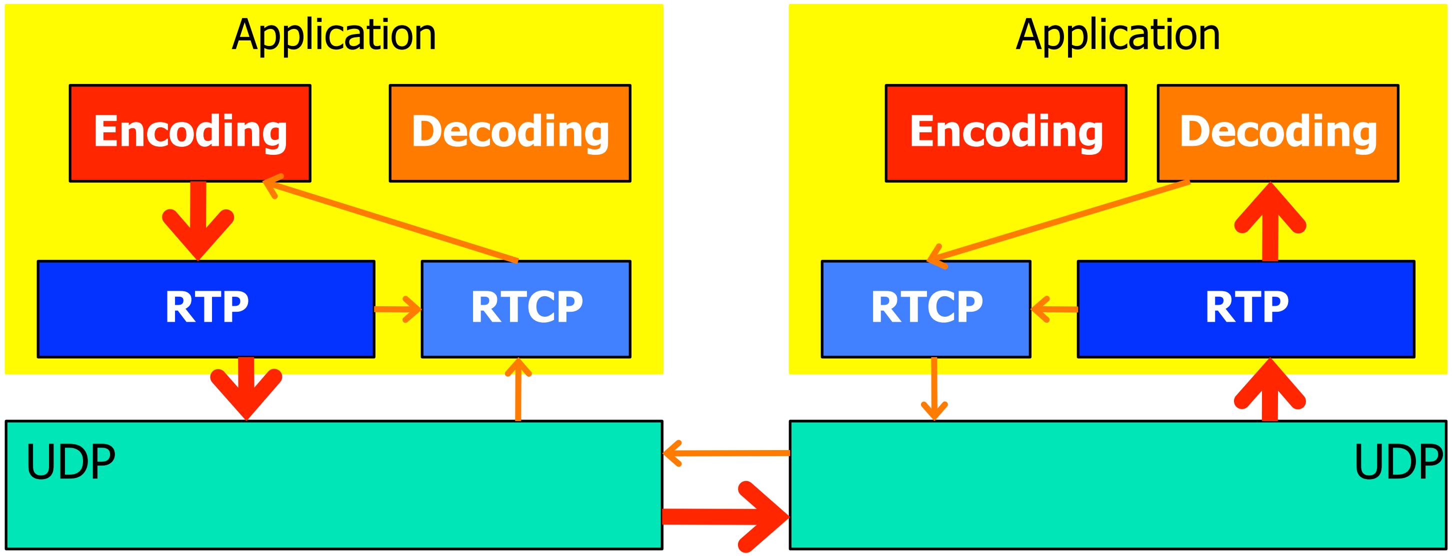
Protocol Development

- Changes and extensions to RTP
  - Scalability to very large multicast groups
  - Congestion Control
  - Algorithms to calculate RTCP packet rate
  - Several profile and payload formats
  - Efficient packetization of Audio / Video
  - Loss / error recovery

Co-existing with TCP

Adapt audiovisual quality to your bandwidth share

RTP Quality Adaptation



Application level framing idea

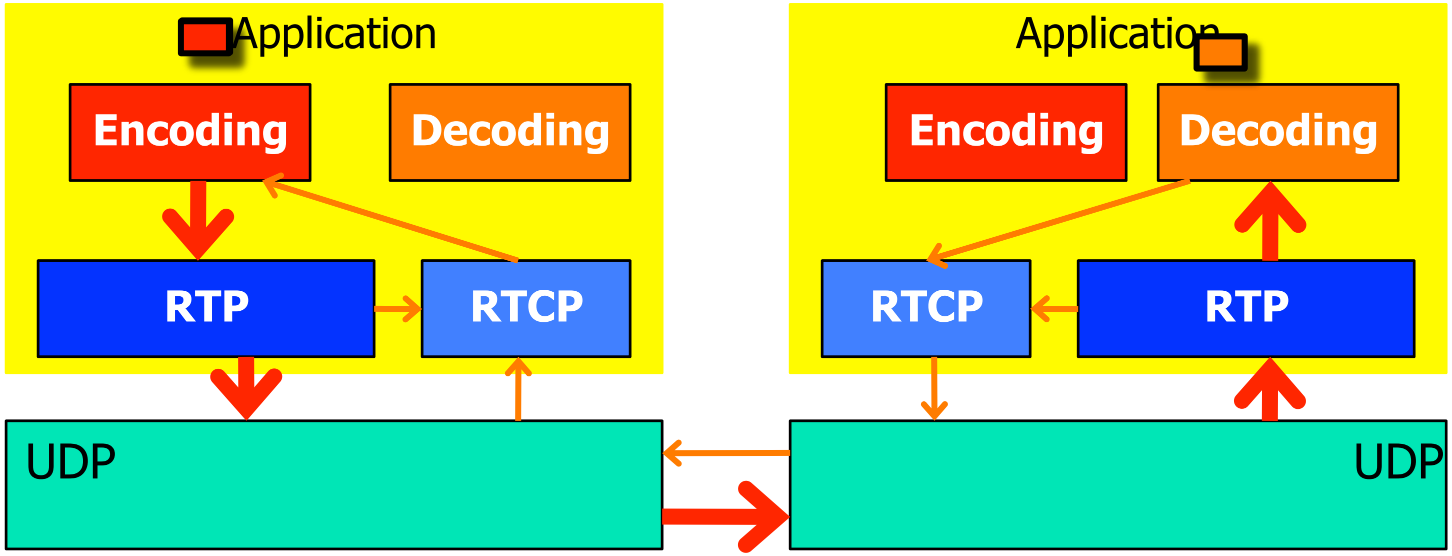
- application knows best how to adapt
- protocol (i.e. RTP) provides information about the network

Application can

- evaluate sender and receiver reports
- modify encoding schemes and parameters
- adapt its transmission rates

Loss-Delay Adjustment Algorithm

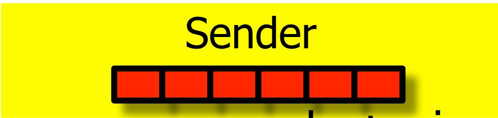
- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically



"The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme ",
D. Sisalem, H. Schulzrinne, NOSSDAV 1998

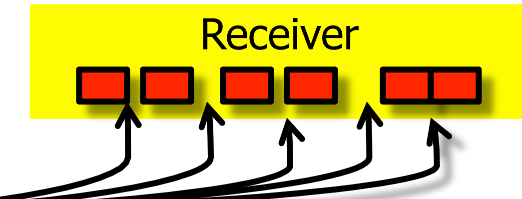
Loss-Delay Adjustment Algorithm

- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically
  - works like TCP's AIMD
    - but RRs are rare
      - max 5% of RTP BW, max ¾ of this RR, equally shared among receivers
    - can't adapt every time
  - step one: estimate the bottleneck bandwidth b



  - use packet size and gap sizes

$$b = \frac{1}{n} \sum_{i=1}^{n} \frac{packetsize(i)}{time(i+1) - time(i)}$$



Loss-Delay Adjustment Algorithm

- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically
  - works like TCP's AIMD
    - but RRs are rare
    - can't adapt every time
  - no loss:
    - use "AIR" – additive increase *rate*
    - but never more than 1 packet/RTT
  - loss:
    - RTCP counts losses, l is *fraction* of lost packets
    - *guess* 3 of those losses in one RTT

current rate

$$AIR_t = AIR * \left(1 - \frac{r_t}{b}\right)$$

$$r_{t+1} = r_t + AIR_t$$

new rate

$$r_{t+1} = r_t * (1 - l * 3)$$