MMS

1 INTRODUCTION

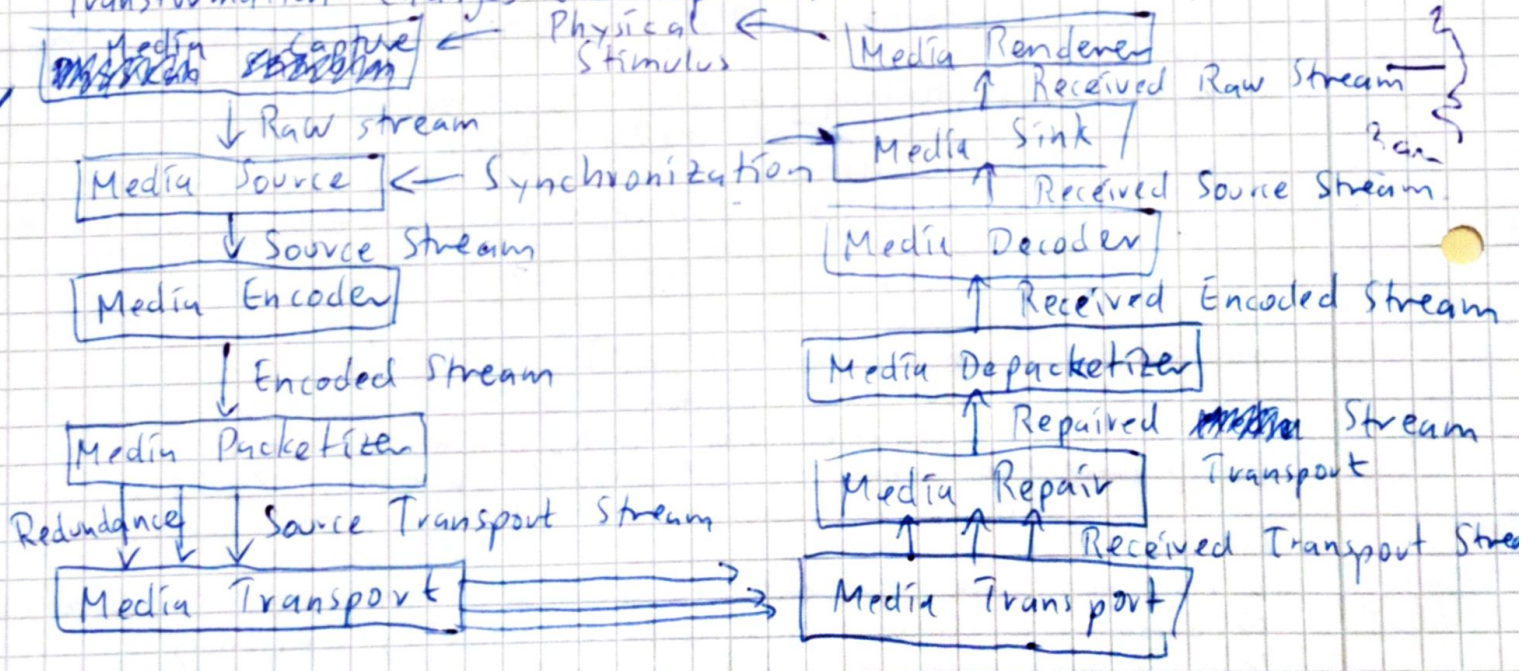
Why? Increasing demand for multimedia services brought over the Internet
Video-on-demand / Streaming / 4K UHD / Live Cams / Youtube / ...
VoIP / Images & AV Bridging / Whatsapp / Snapchat / Video Games / Streaming of Game output screen / Virtual - augmented Reality / Telepresence

Different Media:
Audio: Speech / music / soundtrack } often combined
Video
Images
Time independent media: Image, Text
Time dependent: Video / audio (streams)
MM has different requirements. can have multiple addressies @ same time, can be w/ delay (asynchroneous) or w/out (synchroneous)
Media must be digital (AD -> Media -> DA converter needed)
Client / server or P2P model possible
on top of data stream exists a control stream for stop / pause / play / resume / rewind / fast-forward => signaling
different QoS requirements: delay constraint, loss rate, bandwidth adoption
Transport over the Internet: best-effort, connectionless packet net
Traditionally different networks used for different media:
radio / television / telephony } moving towards Internet
AV Conferencing: needs transport of slides, video, audio
multicast, invitation, media negotiation, synchronization, stream control / QoS
Challenges: Signaling: finding participants
negotiation of media streams
media stream transport (timely delivery, tolerance to data loss, correct, synchronous data playout, rate adoption)
Security
QoS
Streaming of stored data (YT, FB..)
Live data (Twitch, ...)
Interactive data (conversation, telephony)
Groupware: Conferencing w/ shared whiteboard / editor
Group management + session control management
Use of IP Multicast
Participants have different capabilities: bandwidth, supported codecs
=> simulcast: same media, different quality / encoding
QoS: too high delay => packet worthless
loss rate should not be too high
for AV Bridging: often losses tolerable

Error Tolerant	Conversation	Video/voice Messaging	streams	Fax
Error Intolerant	Command Control Games	Transactions Bidding	Downloading Messaging	Usenet, Background
	Delay <<1s	~1s	~10s	

Usually higher bandwidth required than usual applications
Internet has no guarantees
delay, jitter
bufferbloat problem: bigger buffers ⇒ higher delay
packet loss
⇒ provide adaptivie Applications & QoS in the network
VoIP replaces POTS (Plain old telephone System)
PSTN (Public switched telephone network)
VoIP adapters for PSTN devices available
cheaper to operate only one ~~system~~ infrastructure
⇒ telephony also becomes vulnerable when internet goes down
availability & reliability increasingly important
VoIP: - no emergency calls
- no location-dependent service
- no guaranteed quality
- new security issues (eavesdropping, man-in-the-middle)
- (D) DoS flooding attacks
- More unsolicited calls (spam ⇒ SPIT)
+ E2E encryption
+ better codecs ⇒ quality of voice sound
+ better user interfaces
+ adding video or app sharing ⇒ more flexibility
+ reduced management for carrier

2 Multimedia Transport
Media: sequence of synthetic or physical stimulus in digital form
Stream: time based sequences of the physical stimulus in various representations
Transformation: changes the representation in some way



SENDER

RECEIVER

Issues:
Real-Time requirements
→ timelines, continuity

Internet defices:
packet reordering, delay, jitter, packet loss, unknown throughput, no bandwidth guarantees

How to reconstruct the original timed sequence?
One-way-delay: time from first bit sent to last bit received
$\sum$ propagation, transmission, mac (in case of shared medium), processing time, queuing delay

biggest part, reduction difficult

E2E delay important for interactive applications:
games, conversation, control

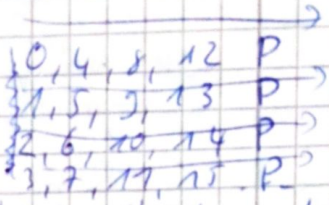
< 150ms: good
150ms - 400ms: perceivable, annoying
400+ ms: useless
Delay Jitter: variation of E2E Delay ⇒ affects continuous media streams
Use play out buffer! Need for timestamps, sequence numbers.
Effectively delays early packets
Fill rate R_F, Playout Rate: R_p
Buffer under run, if $R_p > R_F$ over time
Buffer over flow, if $R_F > R_p$ over time
Idea: play out Packet after g ms after Packet creation
Problem: determine good value for g
too small g: packets miss playout time
too big g: interactive apps impossible
g depends on underlying network induced delay jitter
R_p is known to the receiver due to known codec
t unknown to receiver, would require highly synchronized clocks
Adaptive Play-out
Tradeoff between delay & "loss"
E2E delay for packet i: $r_i - t_i$
t_i: timestamp of packet i
r_i: time of receiving packet i
p_i: time of playout of packet i
Average delay $d_i = (1-u) \cdot d_{i-1} + u \cdot (r_i - t_i)$
Average deviation $v_i = (1-u) \cdot v_{i-1} + u \cdot |r_i - t_i - d_i|$
d_i: smoothed average
u: weight (usually ~0,01)
Calculate d_i & v_i for every packet
Calculate packet playout time $p_i = t_i + d_i + K v_i$
K constant, e.g. 4?K
Assume that receiver can detect begin of a talk spurt
How? if timestamps $|t_{i-1} - t_i| > 20ms$ (⇒ encoder active)
use timestamps & sequence numbers
Loss could be compensated w/ reliable transport (TCP)
but retransmission takes at least 1 more RTT (in LAN, ok)
Delayed ACKs & socket buffers add delay
congestion signals drop sending rate drastically
Flow control does not allow for packet drop ⇒ receiver can't catchup
TCP still often used because of use of HTTP / HTTPS
not practical for highly interactive apps

FEC (Forward Error Correction)

Use redundant information in other packets
cannot achieve 100% reliable transport
adds permanent overhead to transmission

Generic: add after n packets one parity packet
if one of n packets is lost ⇒ good
more than one in n: bad

1-D Interleaved (to deal w/ burst errors)
send parity information for packets



Redundant Layer: send lower bitrate in between high bitrate packets

Interleaving: Reorder packets to one bigger packet. If the big packet is lost, more smaller gaps will occur instead of one big, ~~undetectable~~ noticeable gap.
No overhead, but increased delay

Receiver based repair
provide a ~~repair~~ replacement for lost packets
Packet repetition: low implementation complexity, works well for audio signals & speech
Interpolation: suitable for music, higher computational overhead

Streams can't be sent faster than they are recorded unlike download!
packet loss too high ⇒ stream unusable
Transmission capacity may change frequently, esp. in LTE / WLAN
Congestion based packet loss
usually burst loss due to tail drop queuing strategy
Sender needs to lower sending rate, done by switching the codec settings to lower ones, only discrete steps possible
Receiver needs to tell the sender to lower bitrate / switch codec
What can be switched: Frame size (Resolution), FPS, colors
Need for QoS Parameter @ receiver site
Negotiate set of supported codecs beforehand

Synchronization
Relationship between different media objects in the time domain
eg between an audio & video stream during recording
relationship must be restored during playout
intra / inter-stream relationships
Lip consecutive representation of frames in a video stream
determined by source, depends on sampling clock
for synchronization between multiple media streams

Lip Synchronization for inter stream synchronization

Logical Data Units (LDUs) for inter-stream synchronization
structure for presenting information units of a medium
Closed: duration fixed
open: duration must be determined @ runtime

Standardized Solutions

Session Instantiation Protocol (SIP)
Real Time Transport Protocol (RTP)
RTP Audio/Video Profiles (RTP/AVP)
Session Description Protocol (SDP)
Real Time Streaming Protocol (RTSP)
Synchronized Multimedia Integration Language (SMIL)
Ethernet AVB / Time sensitive Networking

Wiring speakers & mics in music studios / movie studios & live concerts / airports / ...
Replace huge amounts of analog wiring w/ digital network
Link layer approach: Ethernet AVB

Ethernet AVB (TSN)
Complex solution, limited to link layer (LAN)
1 µs accuracy, ~100 ns jitter
Traffic shaping for media streams
avoid bursts → smaller buffers in switches
Admission control, resource reservation → avoid overhead
Identification of non-participating devices

RTP
MM application data transport
Use of sequence numbers & timestamps
Transport Protocols: [illegible], UDP, UDP-Lite, DCCP, MIME

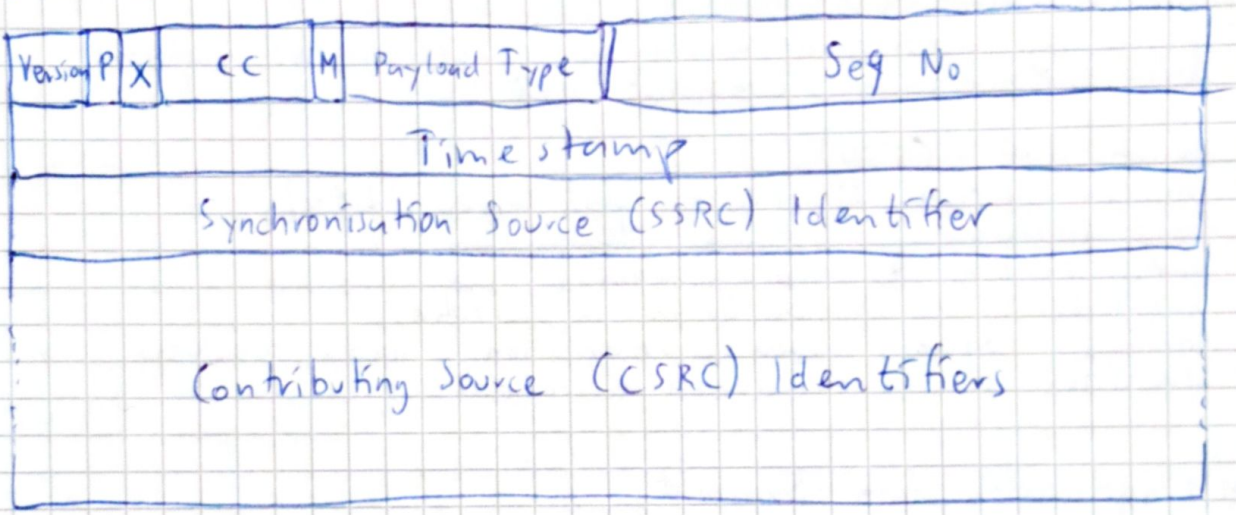

→ Samples → Packetizing to 20ms chunks → RTP encapsulation
→ UDP encapsulation → IP encapsulation
RTCP (RT Control Protocol) for monitor QoS & reporting
RTP allows Translators: change media or transport network
Mixers: combine media streams from different sources

Goals:
Content flexibility: not only A/V! remote data acquisition,
extensibility: incorporate additional distributed simulations, ... mechanisms, allow experimental apps w/ breaking interoperability
independance of transport / routing / link protocols
bridge compatibility: allow for translators & Mixers
bandwidth efficient: low overhead for small packets
international: different codings or char sets
processing efficient: avoid multiplications / divisions
RTP uses the following transport protocol services:
E2E delivery
framing
demultiplexing
multicast
Network can be:
non-reliable
corrupt
loose
reordering packets

RTP does not
provide any mechanisms to ensure timely service
guarantee delivery
prevent out-of-order delivery
assume a reliable lower-layer

RTP is an application level framing protocol
fragmentation & special requirements are built into application instead of any other layer (Application level framing (ALF))
=> Allow for fragmented frame playout. IP fragmentation would discard the whole frame

RTP Header



P: Padding present?
X: Extension?
CC: CSRC Count
M: Marker, interpretation defined by profile
Payload Type: identifies format of payload
Sequence Number: incremented by 1 w/ every packet
Timestamp: sampling instant of the first octet in RTP packet
SSRC: identifies synchronization source (random)
CSRC list (max 15): identifies contributing sources, inserted by mixers

Marker may be redefined w/ PT Field by certain profile:
eg as beginning of Logical Data Unit

PT determines interpretation by application
profile may specify static mapping (PT code → payload format)
source may change payload format during session
should not be used for multiplexing different media streams
receiver must ignore packets w/ code it doesn't understand

Timestamp: The sampling instant of the first octet in the RTP data packet must be derived from a clock that increments linearly & monotonically
=> allows to synchronization & jitter compensation
clock resolution must be sufficient
clock frequency depends on data format
determined from sampling clock, not from system clock
initial value should be random
synchronization of different streams requires pairing w/ reference clock (wall clock)

SSRC: identifies synchronization source
should be chosen randomly
no two sources in the same session should have the same SSRC

collisions in SSRC need to be detected & resolved
detect RTP forwarding loops
on receiving a packet w/ same SSRC but different source transport address

CSRC List: identifies contributing sources for packet payload
inserted by mixers
in case of cascaded mixers: CSRC of already mixed packet should be used

# of MUX should be minimized
RTP MUXing is provided by destination port, usually different for each RTP session
Each medium should be carried in a different RTP session
Using different payload types & different SSRC will not be sufficient

Translator: modifies part of the data
leaves SSRC intact => transparent
passes data stream through from different sources separately

Mixer: receives data stream from ≥1 sources and combines them into a new stream (bandwidth saved!)
input streams usually not synchronized => timing needs to be adjusted in mixer
receivers usually not have total control over mixer

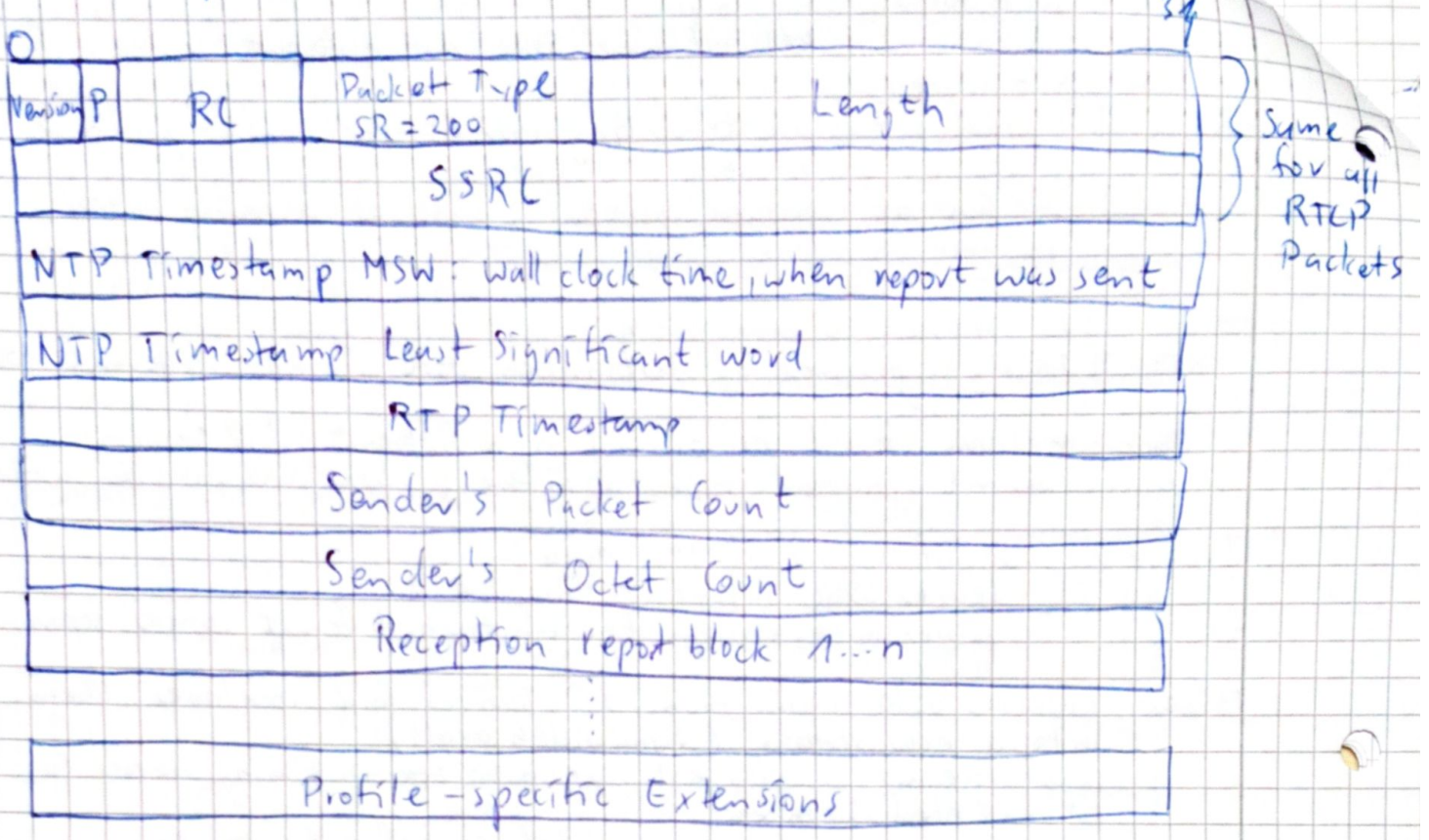
RTCP is a companion protocol to RTP
provides feedback → congestion control for RTP required
control separate from data
provides feedback on quality of data distribution for and from all participants
carrying persistent RTP identifier
scaling to a large # of participants
optionally conveying minimal session control information

Sender tasks: Retransmissions, media scaling, codec choice, FEC, interleaving, congestion control
Receiver tasks: Jitter compensation, synchronization, playout, monitoring, reporting, local error concealment

RTCP packet types
Sender Report (SR)
Receiver Report (RR)
Source Description Items (SDES)
Indication: End of participation (BYE)
Application specific functions (APP)

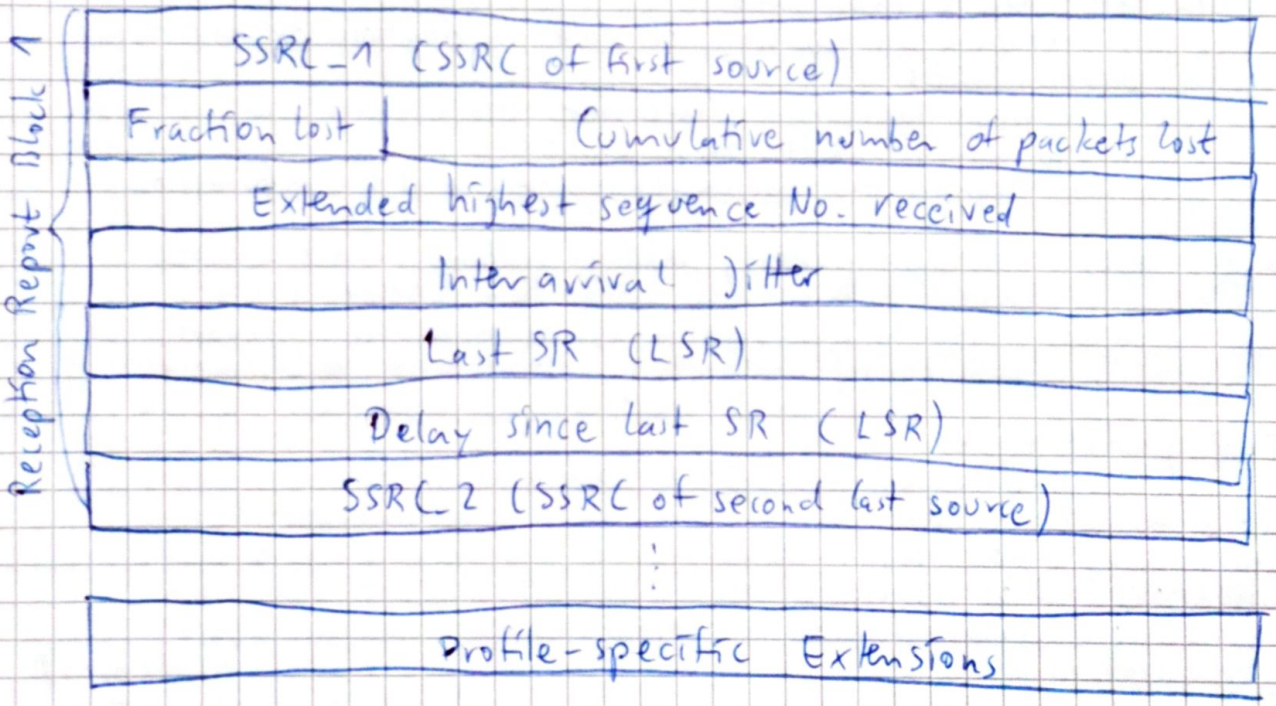
RTCP Packets
fixed header
32-bit alignment
can be concatenated
mixers & translators should compine individual RTCP packets

Sender Report



Receiver Report

<header as above>



Fraction lost: fraction of RTP data packets from SSRC_n lost since LSR or RR sent (fixed point, binary point left edge)
Cumulative number of packet lost: packets expected - packets received
Extended Highest Seq. No: most significant 16 bits: count of sequence number cycles. Low 16 bit: highest seq no received from source SSRC_n
Expected packets: Extended highest Seq NO - initial Seq No

Interarrival jitter: Estimate of statistical variance of RTP packet arrival time: mean deviation of difference D in packet spacing @ receiver compared to sender for a pair of packets
Let S_i be the timestamp from packet i, R_i the time of arrival then:

$$D(i,j) = (R_j - R_i) - (S_j - S_i)$$
$$= (R_j - S_j) - (R_i - S_i) \quad \text{and}$$
$$\text{Jitter } J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16$$

when a RR is issued, current value of J is sampled

Last Sender Report: middle 32 bit out of 64 in NTP timestamps of most recent SR
Delay since LSR: Delay in units of 1/65535 seconds between receiving last SR packet from SSRC_n and sending this reception report block
Round-trip propagation delay = A - LSR - DLSR, A := time when reception report block was received

Source description (SDES)

Source Count in header
Following Chunks: 32 bit: SSRC/CSRC_1
followed by SDES items (NAME, EMAIL, PHONE, LOC, TOOL, NOTE, ...)
SDES Item: 8bit type, 8bit length, UTF-8 encoded text

Goodbye (BYE)
sent, if participator wants to leave session & notify other participants
optional reason
Mixers should bypass BYE unchanged
should send BYE with list of CSRCs it handles and own SSRC

RTCP Scaling
automatic scaling of RTCP session size
control traffic would grow linearly w/ # participants
rate must be scaled down: target 5% of RTP traffic
25% of that should be dedicated to sender reports (newly joining participants should quickly receive CNAME of sender)
fixed minimum interval of 5s
Must estimate # participants
periodical timeout check for stale SSRCs
if too many participants, use SSRC sampling

RTP Compression
packet overhead due to UDP/IP: 40B
Compression for RTP/UDP/IP scheme:
on link-by-link basis
down to 2B (4B w/ UDP checksum)
implementation simplicity
half of the bytes in IP/UDP headers remain constant for a "connection" => differential coding on changing fields
maintain uncompressed header & first order differences in session state between compressor/decompressor
although several fields change often, difference from packet to packet is constant => second order difference is 0

RTP/AVP
RTP Profile for Audio / Video Conferences w/ minimal control
no negotiation of parameters or membership control
defines a set of encodings & payload formats
assigns short names to each encoding
supports also dynamic types

RTP/AVPF
defines early Feedback RTCP messages (FB)
Different modes depending on group size
immidiate / early / regular
Different feedback control information
Transport layer (Generic NACK → lost RTP Packet)
Payload specific (Picture / Slice loss indication
Application layer

RTP Generic FEC
Independent of media (audio / video / ...)
Parity over RTP Payload & header → FEC RTP Packet
Sent in separate stream => compatibility w/ non-FEC hosts
24 bit-mask specifies which media packets are XORed. Bit i specifies RTP Packet N+i, N is seq no
Out-of-band scheme adjustment possible between:
a⊕b, b⊕c, c⊕d, ... allows two consecutive losses
a⊕b, a⊕c, a⊕b⊕c, ... recovers all single loses & some double loses
a⊕b⊕c, a⊕c⊕d, a⊕b⊕d, ... recover one, two, or three losses

SRTP / SRTCP
provide: confidentiality for RTP / RTCP payload
integrity & replay protection
low bandwidth cost, preserve header compression
upgrades to new & cryptographic transforms
small footprint
low computational cost
limited packet expansion
high tolerance to packet loss & re-ordering
defined as profile, extension for AVP
resides between application and RTP transport layer (4,5)
does not provide message authentication in every case: in group communication, there is no data Origin authentication
Header & Payload is authenticated
Payload w/ Padcount & RTP Padding is encrypted
Packet enlonged by optional MKI (Master Key Identifier) & recommended authentication tag

RTP Transport
UDP suitable for multicast-distribution
Port-multiplexing
Framing
bundling
checksum service
} crucial for RTP. RTP can use any transport protocol that supports these two features

unreliable
fast, connection setup omitted
not dealing w/ congestion
provides optional checksums

Bandwidth requirement w/ PCM codec

PCM: 160 B / 20 ms => 64 kb/s
+ Packet overhead: RTP: 12B
UDP: 8B
IP: 20B => Per packet: 200B
=> per minute & direction: 600.000 B
= 80 kB/s

UDP-Lite (Protocol ID 136)
deliver damaged data instead of discarding them
codecs cope w/ errors better than w/ missing data
Link-layer checksum needs to be switched off, or allow forwarding
Packet format:
UDP length field replaced by Checksum Coverage (length obtained from IP header)
contains # octets that are covered (0 => all)
checksum includes pseudo-header, length from IP header
own protocol ID allows support detection
if UDP-Lite had the same ID as UDP, the UDP-Lite packets would be discarded due to checksum errors

DCCP (Datagram Congestion Control Protocol)
Increase of long-lived stream applications expected
=> congestion collapse?
Standardized possibility for feedback & notification of ECN usage is lacking
Choice of TCP-friendly congestion control
3-Way handshake (reliable)
negotiation of protocol features
choice of congestion control mechanism (TCP-like / friendly / VoIP)
low overhead
no flow control
DCCP is connection-oriented
Endpoint initiates actively connection (=> client)
Server is passive
DCCP connections are bidirectional, logically 2 half-connections: data/ack
Seq no count packets, not Bytes
Init-Cookie option to prevent DDoS attacks
Use of ECN
Detection of different reasons for congestion / packet loss
immediate ACK
no flow window / receiving window

Header: Source Port (16 b)
Dest Port (16 b)
Data offset (8 b)
CCVal (4 b) Uses CC mechanism of the sender
SCov (4 b) Checksum Coverage
Checksum (16 b)
Res (3 b) Reserved
Type (4 b) DCCP Type
X (1 b) ¬X: Seq no: 24 bit X: seqno: 48 bit
Sequence No (24/48 b)

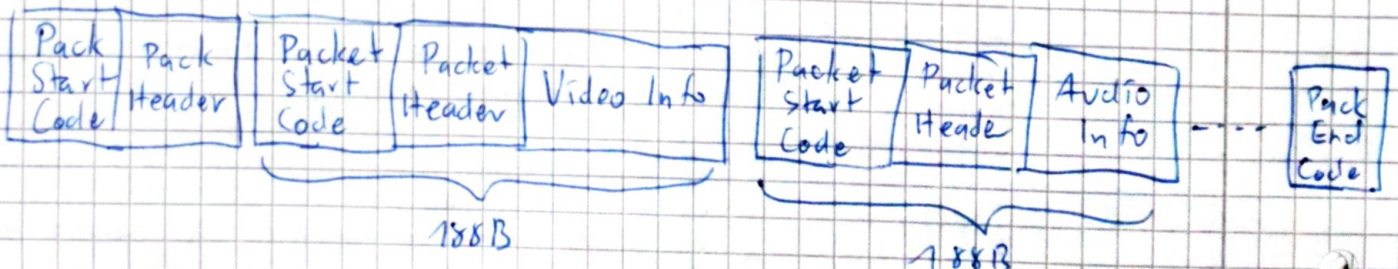
RTP over DCCP
→ duplicated ACKs & duplicated Seq no
overhead add slightly more than 4 B per packet compared to RTP-UDP
DCCP Seq. No can't be inferred from RTP Sequence number,
it increments on non-data & data packets
(⇒ can't infer RTP seq no from DCCP seq no)
Removing RTP seq no would not save anything due to the
32 bit alignment

Congestion: persistent high packet ~~rate~~ drop rates
RTP can monitor drop rates using RTCP
DCCP needs media scaling, but some codecs can't drop below
a certain bitrate ⇒ packet loss. ~20% of packet loss ⇒ worthless data

TCP friendly rate control (TFRC-PS)
be fair w/ TCP
avoid abrupt sending rate changes
intended for applications, where a smooth rate is desired (streaming, ...)
determines sending rate in RTT intervals
fair only if the network limitation was in packets/s not in b/s
TFRC-Small packet (TFRC-SP)
variant for VoIP
rough fairness in b/s in a scenario where each packet receives roughly
the same probability of being dropped
restricted to apps that send no more often than 100/s
nominal packet size: 1460 B
Takes packet headers into account:
Transmit Rate $X = X * s_{true} / (s_{true} + H)$
H = header size
s_{true} = average size

MPEG-2 Systems layer
Program Stream
multiplexing streams of variable length A/V packets
One single stream w/ one timing information
Transport Stream
composed of fixed 188 B packets
reliable transport over (lossy (or noisy) channels
various programs w/ one or more time bases
Access Unit
I/P/B Frame of variable size plus timestamp
Sequence of these is an elementary stream

Transport Pack
Sent to decoder



Elementary Streams = sequence of access units
are multiplexed into a pack of related PES
Pack is smaller than 0,7s
accompanied by header, containing peak data rate, # of streams & timing information

188 B ≙ 4 ATC cells, AAL-1

Header + adaptation (if present) + Information

4B Header: payload unit start indicator: 1 ⇒ first payload byte is first byte of new PES packet
transportation priority: set, if first byte is first byte of PES
Packet ID: identifies elementary stream that the PES was created from
two bit adaptation-field control (payload and/or adaptation field)
Adaptation header may include time & media synch, random access & stream join flags

Fragmentation

MIME (Multi-Purpose Internet Mail Extensions)
Extension for legacy Mail format for embedding mm-objects into E-Mails
6 new Headers: Type of content, Version, Content transfer encoding, Content ID, content description, content length

CODECS
encoding: filtering (low-pass) all frequencies > f_{max}
Sampling: measuring the actual signal value (hold)
quantization: relate value to interval
encoding: assign binary code

Acoustic pressure measured in Bel (deciBel)
Sound Preassure Level (SPL):

$$L_p(p_1)\,[dB] = 10 \log_{10}\left(\frac{p_1^2}{p_0^2}\right) = 20 \log_{10}\left(\frac{p_1}{p_0}\right) = 10\,\mu Pa$$

Subjective perception measured in sone / phon
particular sensitive response between 1kHz - 6 kHz
Quantization levels more dense in low amplitude area, logarithmic placement to higher amplitudes

Pulse-Code Modulation (PCM)
A-Law / μ-Law
G.711 for phone, 8kHz w/ 8 bit Quantization
Differential PCM
only differences between consecutive samples are transmitted
Adaptive DPCM
Sender & receiver use a predictor model that predicts the steps from values in the past
only differences to that prediction model are transmitted
G.721 w/ only 4 bit Quantization

Reduction Tricks
Muting: only send if threshold exceeded
receiver plays silence in the meantimes
Reconstruction: some phonems like M, L, V have a periodic waveform
lost units can be reconstructed from adjacent ones
transition from voiced - to non-voiced important!
Masking: quiet sounds are masked by louder ones of same frequency ⇒
don't need to be transmitted

Compression Delay
PCM / DPCM / ADPCM deal w/ samples, individually
MPS / GSM consider blocks that need to be sampled & processed
completely ⇒ additional delay

Visual Perception
Human Eye: cones / rods
Perceives single images
Spatial resolution depends on Image size & viewing distance
Perception of brightness higher than perception of color
different perception of different colors
Angle & distance influence perception

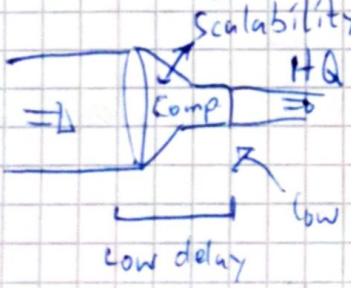
Picture looks smooth @ 16+ FPS
Refresh rate of screen should be 50/s + especially on bright large area
Color models: additive: RGB (screen)
subtractive: YMCB (paper)
Luminance / chrominance: } HLS
Brightness & Hue, saturation }
YUV: $Y = 0,3R + 0,59G + 0,11B$ ← how well humans perceive R/G/B
$U = (B - Y) * 0,493$
$V = (R - Y) * 0,877$

Y has highest resolution (#bits), because it is perceived best by a human. Double resolution for luminance: 4:2:2 or 4:2:0

Y_{11}	Y_{12}	Y_{13}	Y_{14}	U_{11}	U_{12}	V_{11}	V_{12}
Y_{21}	Y_{22}	Y_{23}	Y_{24}	U_{21}	U_{22}	V_{21}	V_{22}

MPEG / JPEG

Compression general requirement



Synchronization of speech & picture
Independence of frame size / frame rate
Dialogue mode: (De-)Compression in real time
25+ FPS
E2E delay < 150ms
Symmetric: same duration for compression & decompression
Retrieval mode: Fast forward & backward retrieval
Random access within 0,5s
Asymmetric: compression takes longer than decompression

Entropy Coding

ignore semantics
Lossless
usable for arbitrary media

Run length Coding
Huffman coding
Arithmetic coding

Idea: eliminate redundancy / decorrelation

Hybrid Coding

MPEG
GSM
MP3
H.263
JPEG

decorrelation
reduction
entropy coding
quantization

Source Coding

Based on data semantics
often lossy
specialized for each media type

Prediction: DPCM
ADPCM
Transformation: FFT
DCT
Layered: Bit Position
Sub-Sampling
Sub-band Coding
Vector Quantization

Idea: smart deletion, eliminate parts w/ low relevance

JPG Idea: human perception is more sensitive to luminance
photos often have smooth transitions
1. Transform RGB to YUV (lossy)
2. DCT in frequency space
⇒ coarse structures are seperated from finer ones
3. Adapted Quantization
4. Entropy coding

Modes: Baseline, lossy sequential DCT
Lossless
Expanded lossy DCT, progressive image display
Hierarchical
Image scaling
Differential coding
Lowest Resolution first, higher resolution as differences to lower resolution

Image preparation: Every 16x16x block of the image is split into 4 8x8 Y blocks
1 8x8 U block
1 8x8 V block

DCT (Discrete cosine transformation)

DCT coefficients make up spectral frequency domain
Pixels refer to one of the frequencies
The spectral energy of a block (= highest values of DCT coefficients) is usually clustered in the coefficients of lower frequencies
ie in the upper left corner
These coefficients are thus very important
The coefficient in the upper left corner of a block (= "frequency zero") determines its mean value, called DC coefficient
others are called AC coefficients
DCT does not reduce the amount of data, but serves for a good base quantization and later entropy coding

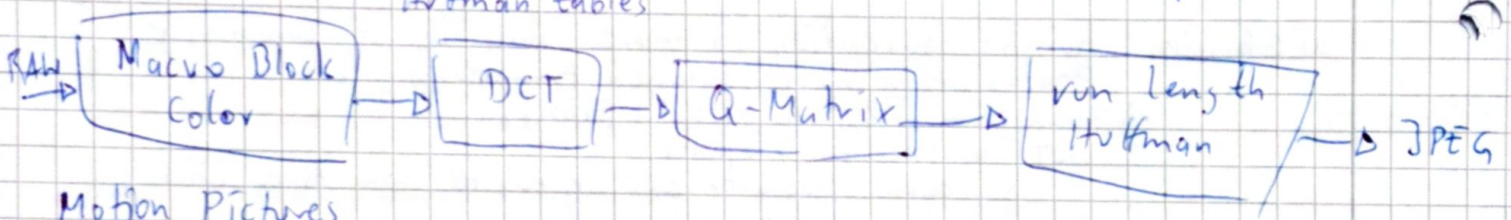
Quantization: Forward $SQ(u,v) = \text{round}\frac{S(u,v)}{Q(u,v)}$
Backward: $R(u,v) = SQ(u,v) * Q(u,v)$

DCT coefficients are divided by quantization factors, that differ in dependence on the position of the coefficient. The less important coefficients represent a higher frequencies get a courser quantization
This step usually introduces loss

Only differences of DC Values are stored
Zig-zag ordering of the 63 AC coefficients ⇒ later ones likely to be 0
Huffman coding all coefficients
Subsequent run-length coding of zeroes

⇒ JPEG Coding 1. Image split in macro blocks: 4 Luminance 8x8
2 chrominance 8x8
RGB → YUV

2. DCT orders structures according to their frequency
3. Different frequencies get different quantizations and are put in sequence from course to fine structures
4. DC & AC coefficients are encoded w/ separate Huffman tables



Motion Pictures
Idea use redundancy of moved (not new!) information of the preceeding frame
Usually a frame is hardly different to the preceeding one.
Differential coding done different: store movement of same pixels (further: blocks)
Search for 16x16 blocks in the previous frame that best match the current 16x16 block. If found, store the new position & do differential coding on it, to get the exact new picture information in place ⇒ shift & remaining difference is encoded
Shifts are determined in half-pixel
Only changed macro blocks get transmitted

Frame types: I Intra Frame: no motion information, new frame, standalone
P Predictive Frame: are referring to previous I or P frames
B Bidirectional Frame: contain two motion vectors and are referring to previous & subsequent frames

In MPEG-2 I: Anchor for random access, no reference to another frame
P: coding & decoding using I & P frames, reference picture for B-Frames
B: Bidirectional predictive coded pictures
Coding based on previous & later (pre-computed) I and P frames
D: DC-Coded Pictures
Intra-Frame coding of DC parameters of a frame
AC parameters are not encoded
Used for fast-forward

Reordering, so that reference frames are transmitted first

Video stream
6 Layers: Sequence Layer = controls temporary store of video data
Group of Pictures layer: I B B P B B P B B ...
Picture layer: data for a single picture
Slice layer: variable number of macro blocks
generate the picture in slices
Macroblock layer: 4 blocks w/ 8x8 pixels Luminance Y
2, 4 or 8 blocks chrominance UV
Block layer: DC- or AC coefficients

Scaling
Spatial (resolution W x H)
Temporal (commiting frames)
SNR scaling
coarser quantization ⇒ coding of macro blocks requires less data
Need no macro blocks w/ small changes

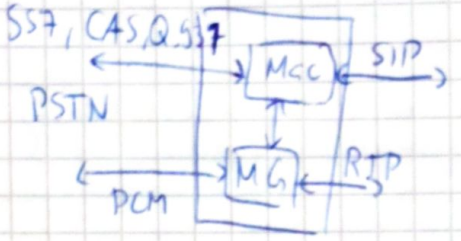
MPEG-4 = H.264 + AAC
or MPEG-4 AVC + AAC
Object oriented video coding
support of interactive applications by audio visual objects (AVO)
eg person, background, text, graphics, animated body
Interaction w/ content
change of objects
navigation in scenes
start of animations
selection of languages
Creation of synthetical scenes from AVO
Coding of different views
Content based coding
Improvements to MPEG-2:
precise segmentation of moving regions
multi picture inter picture prediction
enhanced intra prediction (spatial)
content adaptive in-loop deblocking filter (no artefacts)
enhanced entropy coding
⇒ significant bit rate savings (≈ 50%) @ same visual quality

MPEG-4 SVC (Scalable Video Coding)
Scalability along 3 dimensions: Temporal, Spatial, Quality based on lower quality
AVC: major improvement:
flexible coding tree units for chrominance & luminance
⇒ more precise vectors
more flexible coding schemes

Internet Telephony
Packet switched network carry telephony instead of PSTN
Call setup & control requires signaling:
find & signal participants
negotiate codecs
establish, maintain & end a conversation
SIP / Skype / WebRTC
Data Stream requires encapsulation protocol for transport of voice data
Timestamps, seq no, payload types, etc... Standard: RTP

H.323
for multimedia conferencing w/ non-guaranteed bandwidth packet switched networks
more features than SIP => complex implementation, less popular

Components
Maste/Slave protocol controls devices for announcements, number input, ~
decomposition of early IP telephony gateway into Media Gateway Controller (MGC) & Media Gateways (MG)
Protocol in between: MGCP



MGCP alternatives: MEGACO & H.248
Maste/Slave approach
control of dumb ~~dito~~ slaves
stimulus protocol: events must be relayed to the ~~slave~~ controller
Controller knows every detailed feature of the device
drawbacks:
- protocols limit information that can be exchanged
  => limits new features & services
- MGC single point of failure
- proprietary service logic
- heavy control traffic between maste & slave
- detail-dependency in implementation

H.225.0 Session Signaling
Registration, admission & status (RAS)
Gatekeeper discovery
User Registration
Admission control
Call termination
} uses UDP w/ dynamic ports (Firewall issues)
Call signaling via TCP

H.245 Bearer Control
Negotiates MM data & codecs
Master/Slave determination to avoid conflicts
TCP w/ dynamic ports
Media transport via RTP/UDP/IP

De facto VoIP standard: SIP + RTP solution from IETF
SIP for call signaling, uses DNS for call routing
RTP over UDP for data transport

Advantages of VOIP:
+ multiple voice calls
+ variable bandwidth usage compared to PSTN
+ mobility: user can register (w/ multiple identities) @ current location
  use ~~th~~ ones flatrate from anywhere on earth if sufficient internet access given
  user can be registered @ different locations

+ Low cost for direct calls (only data packets)
+ Better quality (up to HD codecs)
+ easy conferencing w/o special hardware

Security
Signaling
User authentication via HTTP Message Digest Authentication
Encryption for SIP transport (to next device, end device or SIP Proxy) via SIPS (SIP over TLS)
E2E via S/MIME w/ SIP Payload
Payload
IPsec (but larger overhead)
sRTP (smaller overhead)
separate key exchange via DTLS-SRTP

Problems
currently no security assured
no bandwidth guarantees
Usability: too many features for some SIP Clients
⇒ difficult configuration for SIP Server, STUN Server, SIP Port, RTP Port, Codec List, SIP Identity, Login & password, Registration timeout, etc...
NATs are major obstacle
Some attacks are easy
Some PSTN features still not supported

VoLTE
Voice over IP using LTE data channel
+ Faster connection setup
+ Better voice quality
+ less energy consumption
+ support for more voice calls w/ same frequency bandwidth
+ simpler support for rich media services

Skype
VoIP Client using E2E network (w/ E2E security)
Global decentralized user directory
simple user interface
intelligent routing
firewall & NAT traversal
instant messaging
Voice calls (even to POTS and from POTS)
Buddy List
Audio conferencing

Network structure
Overlay P2P network
hosts & super-nodes
Skype login server for authentication of users

Login Process
Client tries to connect to super-node
UDP first, then TCP on specified port, then on port 80/443
else: connect to login server
SN gives out addresses of other Skype nodes

User search
all clients logged in in the past 72 hours can be found
SN returns address to query
Client sends UDP to these addresses
if unsuccessful, client informs SN via TCP
SN then returns twice as many addresses ...
until it search terminates
In case of NAT/FW issues, SN performs search

Call setup
if addresse in buddy list or found by search } : send TCP signaling
if NAT issues occur, use skype server as proxy
if FW filters UDP packets, use TCP instead via skype node
Skype nodes serve as proxy in case of FW/NAT issues } lot of traffic
Serve as conference mixer

Media Transfer
UDP, payload 67 B
bitrate: 5 kB/s up to 16 kB/s, at least 2 kB/s
no silence suppression to keep UDP ports open & avoid drop in TCP congestion window size
even on call hold packets are sent

Google Talk
Instant Messaging
Presence information
group chat
voice mail / calls
file transfer
} using open standards XMPP

"Jingle": XMPP Extensions for E2E call routing using ICE, open source
TLS secured
Now integrated in Google Hangouts

WEB RTC
enable real time communication between browsers
No plugins required, no relays required
Interactive, E2E delay < 50ms
Media: audio, video, "other stuff"
Provide standard set of MM transport protocols
Java script mediates between clients that don't know each other using a server
Media flow then P2P using SRTP (A/V) or DTLS/SCTP (data)
Common functions that all browsers might use:
RTP/sRTP transport
Codec negotiation
NAT traversal
symmetric clients → easy & fast innovation & deployment

Voice Codecs:
G.711 PCM (PCMA / PCMU)
not very efficient
simple or no PLC (Packet Loss Concealment)
} POTS use this ⇒ no transformation needed
G.726
40, 32, 24, 16 kb/s ADPCM
Used in DECT Phones
32 kb/s quality like G.711 but better
coping w/ error rates above 1/1000

Vocoders (optimized encoders / decoders for voice)
G.728 LD-CELP ILBC
G.729 CS-ACELP

AMR (Adaptive MULTI RATE)
defined for use in GSM, UMTS (3GPP)
Multi-rate encoding & mode adaptation
decoder needs to feedback quality to encoder
⇒ Codec Mode Request (CMR)
piggy backed over speech frames in reverse direction
⇒ no out-of-band signaling
VAD (Voice Activity Detection)
CN (Comfort Noise) generation during silence
Multi-Channel audio content separately encoded / decoded
collected in frame block
Perceptually most intensive bits are protected and errors detected
speech classes A, B & C, A is protected
Bit errors in B & C accepted
Link layer support!
3 Checksums! Link layer, transport layer, AMR frame
use UDP-Lite or DCCP

AMR partial checksum
options for use of partial checksums @ transport layer:
1. cover header & at least class A speech data
problem: complete payload discarded if A bits erroneous
⇒ non-affected frames discarded, good B & C speech class discarded
2. cover only header, AMR checksum covers class A bits
⇒ overhead due to frame-wise CRC
AMR supports generic FEC
Payload can be sorted in sensitivity order
Multi-rate capability allows to send copies (or in another mode)

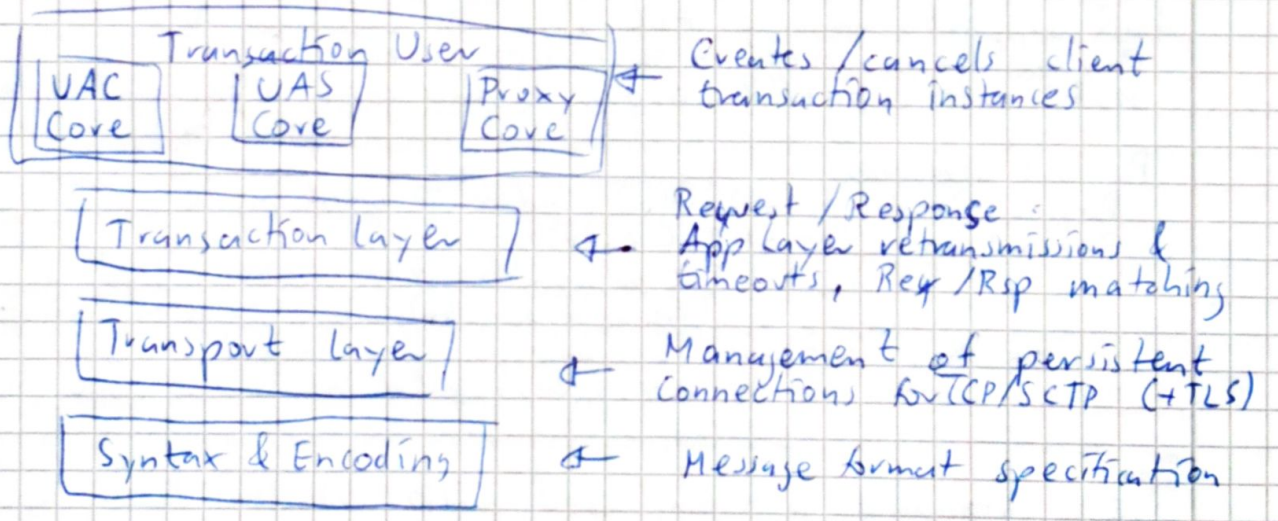
Payload structure
Payload header: CMR (4b), Index to speech mode
Payload ToC: List of ToC entries
ToC entry (5b, Follow on frame (1b),
Frame type (4b), frame quality indicator (1b)
List of CRCs (optional): one byte per speech frame
Speech data: one or more speech frames or comfort noise

Opus
IETF standard providing good audio quality, freely implementable
PLC
wide range of operating conditions & applications:
interactive audio applications, VoIP, in game chat, live music
up to 510 kb/s for high quality stereo music
VBR & CBR support
algorithmic delay of 5ms - 66.5ms
use of linear prediction (LP) & modified DCT

SIP
Core protocol for session establishment in the internet
Application layer signaling protocol
use of proxy services as infrastructural elements
Flexible service creation: dumb network, intelligent devices
E2E principle, SIP servers can be located anywhere in the network
Problems to solve:
Addressing of users : who is to call
User availability : registration & presence
User location : determine address of end device for communication
Session setup : ringing & altering users, determine media codecs & parameters
Session management : termination & invoking services
Session: data exchange between an association of participants
Users may move
be addressed by multiple names
communicate in several different media
e-mail style addresses
names: sip:foo@bar.com permanent
addresses: sip:foo@1.3.3.7 temporary => allows personal basic mobility

SIP Endpoints
User agent client (UAC) creates SIP requests
User agent server (UAS) generates responses to SIP requests
Proxies, used for routing requests or enforcing policies
can be stateless or stateful ("transaction")
SIP Registrars
stores mapping of names to contact addresses
allows to locate an user

Protocol Structure



SIP Location server
stores bindings

SIP Redirect server
returns new location for request

SIP Proxy Server:
routes call requests

SIP Back-to-Back user agent (B2BUA)
triggers calls for incoming calls => maintains dialogue state

SIP Application Server

SIP Entity Roles

Core
functions specific to a particular type of SIP Entity
All cores are stateful except for a stateless proxy

Client
Server
Proxy
Redirector
} all states last from first to last message of a session

Direct call Scenario

Alice sends INVITE w/ SDP packet & AVP port
Bob UA locates Bob, sends 100 TRYING
Bob UA found Bob, starts alert, sends 180 RINGING
Bob answers call, Bob UA sends 200 OK w/ SDP packet & AVP port
Alice sends ACK for chosen codec
<Voice connection w/ chosen codec> using transmitted ports
Bob hangs up, Bob UA sends BYE
Alice sends 200 OK

Address Resolution

finding a callee
Direct calls require knowledge of callee's address
SIP URI provides abstract naming scheme only
Mapping from name to real location required
Address of record (AOR): SIP(S) URI pointing to domain w/ location service
Location service
maps URI to URI, where user might be available
provides list of bindings: AOR → contact address (es)
List sets populated by UAs that register the current location at location server
used by SIP redirect or proxy server
Caller sends INVITE to all SIP servers knowing the location of callee
Server may refuse or redirect call

Finding next hop

UAC may use configured outbound SIP proxy
can be manually configured / learned by registration or DHCP
If URI contains IP & port, call directly
Otherwise use DNS to find next hop (query AAAA & A record)
Problems: determine IP address, port & transport protocol before contacting next hop
client may try another server if one fails
DDDS (Dynamic Delegation Discovery System) or NAPTR (Naming Authority Pointer) allow crude level of capacity-based load balancing

Locating SIP Servers

URI is either target (from Request-URI) or intermediate hop (from Route Header)
not modified or rewritten
resolved to IP address, port number & transport protocol
Client selects transport protocol, port & address
numeric IP address or non-numeric target + port no. given:
sip → UDP sips → TCP + TLS
otherwise: 2-step approach

1. NAPTR query to obtain supported protocols as SRV RR
2. Query for SRV RR: get ports & server name
3. Query for AAAA or A RR to resolve name to IP address

Overall DNS Resolution Process

DNS Query → NAPTR RR → non-numeric addresses & transport protocols known

DNS Query → SRV RR → List of servers for desired protocol known

DNS Query → AAAA RR → numeric IP address of desired server known

NAPTR
maps name to URL
ask for SIP+D2X or SIPS+D2X ⇒ maps to SRV RR
(X = wildcard)

SRV RR
specifies location of server (port & non-numeric address)
if empty, perform AAAA or A lookup on original name, no backup or load balancing services assumed

SIP Registration
SIP REGISTER request by Bob @ SIP Server of his own domain
⇒ establish binding between SIP (S) URI & machine
Registrar puts binding in DB, location service
soft-state: Bob needs to keep binding alive by periodically re-registering
Bob can have multiple registrations
for different SIP (S) URIs
from different devices & locations
Registrar is often co-located w/ SIP Proxy
Registration may require user authentification

SIP Messages
Text based, similar to HTTP
Structure:
- Start line // Request Line / Response Line
  - ↳ Method Request + SIP Version
  - ↳ INVITE, ACK, BYE, OPTIONS, ...
- *message header
- CRLF
- [message body]

SIP Requests
first digit: class of response
1xx provisional
2xx success
3xx Redirection
4xx client error
5xx Server error
6xx Global error

SIP Methods

INVITE, ACK, CANCEL, BYE, OPTIONS, REGISTER
↳ ask server about supported extensions a/o pings

Important header fields

Via: Contains routing information
identifies location where the response is to be sent
branch parameter value must be unique
To: Logical receipent of SIP request
From: Logical ID of originator of the request
should not contain IP address or FQDN (fully qualified domain name)
Call-ID: GUID for call
groups together a series of messages
must be the same for all requests & responses along dialogue
Dialogue ID completed @ first response, built together w/ from-tag
two sided dialogue ID required due to forking: multiple dialogues can be established from a single request, must disambiguate
C-Seq (Command Sequence)
Integer & method name (200 INVITE)
Must match name of request
Incremented for each new request within a dialogue
Allows to detect retransmissions and finding a matching RESPONE
Max Forwards
default: 70
prevents infinite looping
Supported / Require
UAC should list extensions in supported
UAC can request support by listing them in require
Contact:
allows to contact other end system

SIP Dialogue
P2P SIP relationship between 2 UAs
established by 200-Response to INVITE
Identified by (Call ID, local tag, remote tag)
SIP Proxies are dialogue-transparent, B2BUA: not!
↳ do not maintain dialogue state
stateful proxies maintain transaction state
B2BUAs maintain dialogue state & must participate in all requests sent on the dialogues it has established

SIP Transaction
comprises all messages between client & server from first request to non-1xx response
If request is INVITE & final response is non-2XX, transaction includes ACK to response
distinction made due to guaranteed delivery of 2xx responses
2xx responses are transmitted from UAS from E2E
UAC manages retransmission of ACK, not transport layer
⇒ ACK is considered as separate transaction
Identified by (branch in Via; CSeq)
Stateless Proxies insert Via headers, but don't have a transaction layer ⇒ transparent for transaction

Session Setup
3 Way-handshake INVITE / 200 / ACK
provisorial responses indicate progess & don't belong to the 3WH
≥0 provisorial responses can be sent before final response
ACK signals that UAC still available in case of ringing takes long
allows media negotiation
Media session continues indefinitely w/o SIP interaction requirement

Session Modification
RE-INVITE
new INVITE / 200 / ACK (3WH) after initial one
↳ no retransmission!
may change any session types & characteristics, IP source address / port
Thus modifies session state within a dialogue

Termination
BYE

Cancellation
UA ends call before media stream establishment
used by proxie's parallel search when one end point accepted telephony

Call routing
Requests are routed via SIP URI
Route is recorded in Via
Record Route & Route can be used to force request paths
Route
force requests to go through listed set of proxies
provide information for requests
strict & loose routing ("lr" -parameter)
Record Route
inserted by proxies in a request to force future requests in the dialogue to be routed through the proxy

Reliability of SIP Messages
INVITE
not retransmitted over reliable transports (TCP, SCTP)
otherwise, it is retransmitted until RESPONSE received
retransmission interval starts @ T1 (RTT estimate, default: 500 ms)
doubles w/ every retransmission (= exp. backoff)
200 for invite and other final responses are also retransmitted by UAS
Final responses are ACKed → to quench retransmissions
For any transport transaction timeout (64 · T1) is used
For NOIV-INVITE methods
retransmission timeout doubles up to 4s
1XX is sent by UAS only if no final answer expected w/in 200 ms

Additional Methods
INFO: mid-call information
UPDATE: modify state w/o changing dialogue state
allows for setting up QoS mechanisms or "early media"
SUBSCRIBE:
NOTIFY
REFER
PRACK: Provisional ACK
1XX codes sent unreliably → need ACK (often w/ PSTN)
sent, if request was "100rel" containing
PRACK echoes numbers in RSeq & CSeq of the response in RAck hand.
acknoledgements receipt of provisional response 1XX
is not applied to 100 TRYING

Media Negotiation: SDP
not really a protocol, rather description syntax
SDP field list
v = version number
o = owner/creator & session ID { username, session ID, version, network-type, address type, address
s = session name
i = session information
u = URI of description
e = e-mail address
p = phone number
c = connection information { network-type, address-type, connection address
b = bandwidth information
<one + time descriptors>
t = time zone adjustments
k = encryption key
a = zero + session attribute lines
<zero + media descriptors> { m = media port transport format list
media (audio, video, app., data, control)
port no., transport (RTP/AVP or UDP),
format list (media payload types, codec list).

Attributes can be: recvonly, sendrecv, sendonly, format transport, inactive

Offer/answer model
calling party lists capabilities
called party lists supported media capabilities in 200 OK
Offer
must contain exactly one session description. 0 ⇒ added later
if offer is sendonly, the port is used only for RTCP packets
otherwise it is the receiving port for media stream
media format for each stream contains
set of formats (codecs & parameters)
RTP Payload type numbers to identify those formats
if multiple formats are supported: order of presence ≙ priority
Answer
for each m-line, there is a m-line in the answer
rejection by setting port to 0
Answer must contain rtpmap attributes for dynamic payload types
formats should be in same order as in offer

Integration of resource management
if a session requires resource reservation
no session should be established in case of unavailable resources
mutual dependencies
phone should not ring unless reservation was successful
Use of UPDATE Method
Type of QoS Protocol (RSVP, NSIS) negotiated via SDP

SIP Security
VoIP brings new vulnerabilities compared to PSTN
Internet brings " , all must be considered
also downtime of internet
Confidentiality possible, when SIP & media streams are encrypted
→ DTLS-SRTP
VoIP telephones need certificates
Aims: caller & callee authentication / authorization
confidentiality (but proxies & co. must be able to read header)
Signaling is interesting for attacker too

detect communication partners
find information about interesting media streams
record streams
SRTP requires key exchange
SIP message transport via SIP over TLS : SIPS

SIP Authentication
stateless; challenge based, HTTP basing
User-to-User authentication
if no credentials (authentication header) provided, 401 returned
UAS challenges UAC by sending him WWW-Authenticate header
UAC will re-originate the request w/ proper credentials
Proxy-to-User authentication
Proxy may authenticate originating UAC before request procession
unauthenticated requests are responded w/ 407 & Proxy-authenticate header field
UAC should re-originate request w/ Proxy Authorization header field

S/MIME
allows SIP UAs to encrypt MIME Bodies within SIP
not affecting message headers
provide confidentiality & integrity for message bodies, as well as mutual authentication
requires foreknowledge of keys @ sender's side
sign w/ own private key, encrypt w/ public key of peer
S/MIME can encapsulate whole SIP messages
=> SIP header privacy & integrity via SIP over S/MIME tunneling

SIP transport layer
TCP: fragmentation & congestion control
- setup of connection between every hop
- should be kept open & alive
UDP + no connection setup,
+ simple
- unreliable ⇒ use retransmission on higher layer
if request ≤ 200 B or > 1300 B use TCP, UDP otherwise
What Protocol is used, is signaled in Via: header
TCP/UDP support mandatory
SCTP also possible
used between proxies
use of multi-homing feature in case proxy dies
no head of line blocking
TCP + SCTP possess connection setup overhead
can be combined w/ TLS

Problems solved?
Addressing of users → SIP URI
User availability: registration & presence
user location: SIP call routing using SIP URIs, DNS & SIP Proxies
user capabilities: SDP in SIP body
session setup & management: SIP methods / responses / transactions / dialogues

NAT Traversal
SIP signaling may not be easyily possible behind NATs due to unknown bindings, rport extension helps for UDP
Media streams may use arbitrary ports
privat addresses in SIP messages and SDP

STUN (Session Traversal Utilities for NAT)
Client/Server protocol enables to detect
presence of a NAT gateway
NAT configuration for UDP
the used address mappings

Enables incoming UDP packets for certain NAT variants
- not for TCP
- not for symmetric NAT
- requires STUN server in public network (default port 3478)

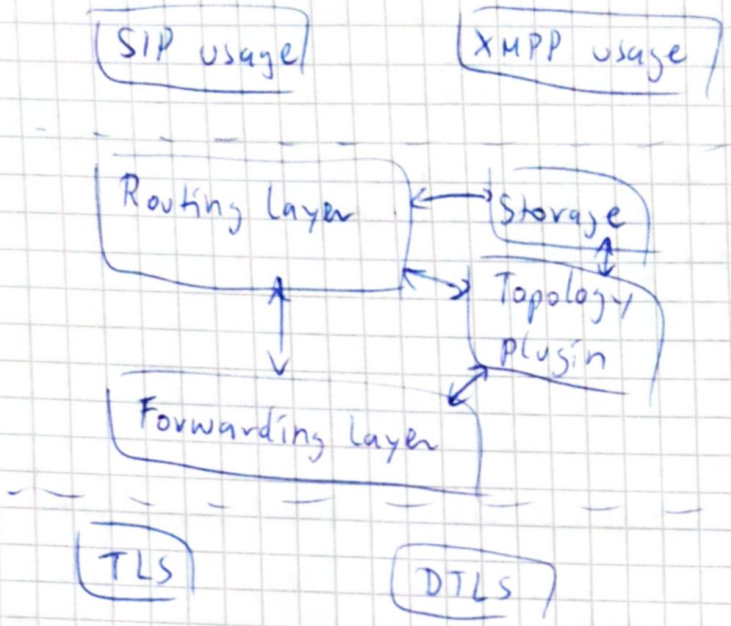
TURN (Traversal using relays around NAT)
traversal for clients behind symmetric NATs using a relay server
designed to be used w/ ICE
RTP streams are (UDP) relayed to TURN server (=> high load!)

ICE (Interactive Connection establishment)
tries to determine address candidates
host candidates, server reflexive candidates, relayed candidates)
make sure the phone doesn't ring unless the media connectivity exists
works through nearly any type of NAT & FW
discovers shortest path for media between endpoints
does not require endpoints to discover NATs
uses relay only in worst case: symmetric NAT
uses & requires TURN & STUN to discover candidates
uses SDP for negotiation of candidates

P2P SIP
using a P2P network for SIP user registration & location lookup
+ cost reduction
+ scalability
+ Reliability
+ Failover for server-based SIP Networks
Use of distributed hash table for registration
DHT
store efficiently (key, value)-pairs items on participating peers
SIP URI → key, Node-ID → value
every peer is responsible for a range of ID-space
data items are assigned to peers according to the value of the key

RELOAD
Central enrollment server provides certificates to authenticate nodes, signed messages & objects
supports other Applications besides P2PSIP
works in NAT environments by using ICE for NAT traversal
HPR (Routing)
Abstract interfaces support pluggable overlay algorithms

Architecture



SIP Services & applications

call transfer
REFER Method
used by UA to request another UA to access an URI or URL resource
can be sent either inside or outside an existing dialogue
Refer-To header contains target URI
If Refer-To contains SIP URI, INVITE is originated
Replaces, if UA receives INVITE w/ "replaces-header field" & matching dialogue. It will transfer SIP session to requesting device
Call hold
Call forwarding
DTMF carriage (dual-tone multi frequency signaling)

SIP Notification Framework
entities in network can subscribe to various resources or calls in the network, which can send a notification to their subscribers on event

Presence: willingness and ability of an user to communicate w/ another
allow SIP to be used for subscription & presence notifications
Pasing
"Presentity" ≠ Presence entity
provides presence information to the presence service
Watcher receives presence information
Fetcher simply requests the current value of some Presentity's value from the presence service
Poller: fetcher on regular period basis
Subscriber requests notification from the presence service of (future) in some presentity's presence information
Watcher (subscriber) wants to learn about presence information from some user (presentity), he creates a SUBSCRIBE request
identifies SIP URI or presence URI (pres:)
is carried along proxies as normal request
arrives @ presence server
NOTIFY
indicates status change of presentity
⇒ mechanism for buddy list

Instant Messaging (IM)
Extension by MESSAGE Method
Body usually of text type, but can be any
im: user@example.com may resolve to SIP URI
is composing defined (interaction)

IP Multimedia Subsystem (IMS)
Trend towards "All-IP"-networks
→ easier, cheaper to maintain only one network
operators only need to provide bit pipe
providers should offer added-value services
be open towards new services
offer any kind of service
offer single sign-in & unified billing

IMS is part of NGN (Next Generation Network)
packet-based
QoS enabled
service-related functions
separation between service & transport
supports generalization mobility
IMS Model
network operator & service provider control access to the network & services
customers are billed for these services
built on SIP services
Functional entities
User Equipment (UE)
Application Server (AS)
Breakout Gateway Control function (BGCF) (for PSTN breakout)
Call Session Control Functions (CSCF)
Serving (S-CSCF)
SIP Registrar
retrieves subscriber profile from HSS (Home Subscriber Server)
performs SIP Routing
forwards SIP Messages to AS if required
Interrogating (I-CSCF)
contact to home network
performs stateless SIP Proxy function
Proxy (P-CSCF)
initial point of contact in IMS
stateful SIP Proxy
HSS (Home Subscriber Server)
contains subscription database for IMS
Media Gateway Control Function (MGCF)
SIP to ISUP conversion, etc.
Media Gateway (MGW)
Terminates bearer channels from PSNT & media streams from packet switched networks
Performs media conversion
Media Resource Function Controller (MRFC)
interprets information from AS or SIP endpoint
controls MRFP to support media services
Media Resource Function Processor (MRFP)
media stream mixing, transcoding, tone & announcement generation
Service Locator Function
front end for distributed HSS systems

ENUM
Telephone number mapping +49 150 33420 1337 → sip: foo@bar...
connect systems that rely on telephone numbers to those who use URI to route transactions
ENUM is a DDDS (Dynamic Delegation Discovery System) app
uses DNS as database, NAPTR records
uses .e164.arpa - zone
divided into sub domains for each country
different or private dialing plans may use the same mechanisms but:
must be called other than ENUM
should be without leading "+"
enum: +49 7216084 46987 → all further addressing information:
SIP URI, email, fax, web, vCard...
"+": anchor
reverse numbers
intersperse "."
add e164.arpa ⇒ domain name for the NAPTR records

Public ENUM
⇒ public phone book
Private ENUM
allows service providers to exchange phone number to URI data
data is non-accessible via general internet (VPN-protected)
Infrastructure ENUM
used & exchanged by service providers in order to find points of interconnection
"Carrier-of"-record ⇒ service provider to which e.164 number was allocated/ported

Various DNS problems / security issues ⇒ use DNSSEC
Application service should still authenticate the peers
Must also consider URI resolution security
Privacy concerns: which / how much contact info should be visible

STREAMING & CACHING TECHNIQUES FOR CONTINUOUS MEDIA

Stream: continuous media flow, playback starting on arrival of first few seconds of media content, no high delay @ start
continuous playout (from play out buffer)
Private usage: YT, ~~Spotify~~ Twitch, Instagram
Commercial: Apple music, Netflix, Amazon, Spotify
TVs & other equipment now have internet access
Classification
Stored / Live / Interactive
A/V Media on demand
Broadcast: traditional, no interaction / control
Pay-per-view: limited interactivity
Near video-on demand
same video distributed in regular time intervals
simulated backward / forward
True video-on-demand
like VCR
requires bi-directional connection

Streams are
expected to be continuous & steady
lengthy (up to several hours)
potentially starting & stopping @ any time
flowing @ some expected mean rate

Problems
Guaranteeing a continuous flow despite packet loss
Varying delay jitter requires compensation (⇒ play-out buffer)
Varying available bandwidth (best-effort network)
non-continuous transmission/receipt
need to restore intra-stream synchronization
Stream control, navigation required

UDP-based streaming
UDP + RTP as transport
RTCP - RTP for control
RTSP for data transport of streams

Application Software
Server side
HTTP + TCP
HTTP Live Streaming (Apple)
Smooth Streaming (Microsoft IIS Media Server)
HTTP Dynamic Streaming (Adobe)
Dynamic Adaptive Streaming over HTTP (DASH, ISO)
Various Clients
Adobe Flashplayer
Microsoft Media player, Silverlight, Net Meeting
Quicktime
Real Networks' Real Player

HTML 5
defines new audio & video elements

Server load increases linearly w/ # clients
unicast transport ⇒ congestion
still no QoS guarantees
⇒ clients experience unpredictable playback quality and/or high start-up latency
Solution: IP multicast: replicate packets @ point of branching
not yet widely supported
lack of business model, management issues, doesn't work well across domains
Only useful in case of synchronous receiving ⇒ broadcast
does not address issues w/ high start-up latency
Pushed by IP-TV
Large providers use IP multicast w/in their domain

Caching of streaming media
Local caches store recently requested stream data
hit: client receives media from cache, fast
miss: client needs to receive media from original server: slow
widely used in CDN (Content delivery networks)

Common CDN architecture
1. Ask server for content
2. Server redirects to one of his CDN servers (⇒ load balancing) & passes a token to the client
3. client sends token to redirected server
4. redirected server verifies token & starts stream

Progressive download
playback while downloading whole file
in case of bandwidth issues: stall playback
HTTP-based Streaming
fetch chunks (4s+)
allow for rate adoption in case of bandwidth issues

Streaming media protocols
RTP: transport for real-time application data
RTSP: application-level control protocol for streaming (like a remote control
RTCP: control protocol for RTP, enables feedback on session status
Widely used:
control: HTTP instead of RTSP
Data transport: HTTP+TCP instead of UDP w/ RTP

RTSP is a control protocol
application-level protocol for control over the delivery of data w/ real time properties ⇒ controls transmission of streaming objects
Mission: client-controlled real-time delivery of media
bidirectional request/response protocol
establishes context including content resources (media) & controls
single or several time-synchronized streams of continuous media
suitable for live data feeds & stored media
controls multiple data delivery sessions
does not deliver the media stream itself
Features
Methods: SETUP, PLAY, PAUSE, DESCRIBE, ...)
data carried out-of-band via RTP
RTSP server needs to maintain state
Request URI always contains absolute URI ⇒ allows easier virtual hosting
RTSP inherits from HTTP
⇒ Reuse of extensions for authentication & cache control
Extensible: easy to add new parameters & methods
Easy to parse: can use independent HTTP Parser
Secure: Re-uses web security mechanisms
Transport independant
Multi-server capable: media stream can reside on different servers
RTSP can use multiple concurrent control connections
Control of playback devices
Separation of stream control & conference initiation
Supports professional applications: frame level accuracy w/ SMPTE timestamps
Presentation description neutral
Proxy & firewall friendly
HTTP friendly (reuse existing infrastructure)
Capability & transport negotiation

RTSP Session
no notion of a connection, only a session
comprises a complete transaction (eg. viewing a movie)
typically consists of phases:
1. setup of transport
2. starting a stream
3. closing the stream
not tied to transport-level connection (client may open & close many TCP connections)

RTSP Session state
client needs to prove liveliness via RTCP feedback or anything else
RTSP requests for a single media stream may be issued sequentially on different TCP connections
Server needs to maintain session state to correlate RTSP requests w/ a stream
Session ID is generated in response to SETUP request
RTSP Methods use session header field

RTSP Methods
define actions/allocations/usage of stream resources on the server
SETUP: resource allocation & session instantiation
PLAY: starts data transmission via allocated stream resources
PAUSE: temporarily halts transmission, resources are not freed
TEARDOWN: frees allocated resources & terminates session
PLAY_NOTIFY, DESCRIBE, OPTIONS, REDIRECT, ...

Requests contain: Request line, general header, request header, entity header, message body
Request line: <Method> <Request URI> <RTSP-Version>
Responses start w/ status line: <RTSP-Version> <Status Code> <Reason Phrase>
Pipelining is allowed
send multiple requests w/o waiting for responses
response order ~~not~~ must be same as request order

RTSP Presentation
set of streams to be controlled
can be in different formats (also SDP)
usually requested by a web server (program guide etc)
can be transmitted via HTTP or email or...
doesn't need to be on the media server
includes encodings, language, addresses, URLs
should be able to express static & temporal properties of a presentation
Some Methods can only be applied to streams, not presentations or vice versa

RTSP Functions
Retrieval of media from media server
client can request presentation description
support for on-demand media & live media w/ recording / time shift
seeking (w/ in given media range)
Speeded up / slowed down playback (scale header) / transport (speed header)
addition of media to existing presentation
tell client about additional media becoming available
especially useful for live presentations

RTSP timestamps
Use of SMPTE relative timestamps
accuracy @ frame-level relative to clip start time
hours : minutes : seconds : frames : subframes
widely used for editing
Normal Play Time ≙ absolute position in stream
Like VCR display

Layering of stream protocols

Application	HTTP	RTSP	RTP	RTCP
Transport		TCP	UDP	DCCP
Network Routing			IP	

DASH (Dynamic Adaptive Streaming over HTTP)
DASH - ISO IEC : set of commands to deliver media content via HTTP
HTTP used for transport
client chooses / changes bitrate
Segments of different qualities stored in different files
Index & media description files (XML format)

Hierarchical data Model:
Period: one (+) media content components (audio in different languages, different views, subtitles, ...)
Media Stream: several encoded versions of media content
Representation: deliverable encoded version, usually different quality, clients request different segments via URLs

Dynamic Adaption
Segments (m4s media) requested via HTTP GET
allows for selection of proper format while playback
bad interaction w/ TCP congestion control w/ rate adaption
playout buffer full → next get-request delayed for more than 200ms
congestion window shrinks ⇒ slow start ⇒ 1s/2s on-off behavior

Caching Techniques
Differences to conventional web traffic
1. worthless if too late
2. Different signaling protocols (HTTP vs RTSP)
3. object size: caching whole stream objects not feasible
⇒ classical caching methods not suitable for streams
Innovative streaming solutions will adapt to user behavior & will scale w/ increasing demand for streaming data
⇒ Fast prefix transfer reduces access delay
Scalability is achieved by segmentation of streaming objects
Dynamic caching increases throughput through request aggregation
Playout Buffer filling delays zapping or hopping
Time to playback = connection delay + buffer delay
⇒ Fast Prefix transfer: transfer packets faster @ beginning of buffer filling ⇒ decreased buffer delay
knowledge of buffer size of client useful
Port prefixes to cache ⇒ reduced connection delay
Remainder is then streamed from original server while playback

Segmentation of streaming objects
Keep the first few segments of a stream in cache, replace the later parts of a stream first
⇒ maximize # prefixes in cache
allows for coordination of different caches
Timing relation among segments must be controlled
⇒ intelligent fetching of segments
" cache replacement to avoid gaps
Goal: fine grained cache replacement, while avoiding data gaps
1. Determine the victim segment
2. determine the chunk to which the victim segment belongs
3. Eject the last segment of the victim chunk
⇒ preserves prefix as much & as long as possible

Dynamic caching
Asynchronous request for the same object characterized by temporal distance Δ
Dynamic cache bridges Δ
⇒ enables multicast even for asynchronous requests
size of dynamic cache independent from size of media object
⇒ depends on Δ-estimation
Dynamic caching requires data patching
Patch closes gap of missing initial part
Patch can be obtained from original server or prefix store in cache

Advanced TV
Why analog TV is gone:
Bad quality if signal is weak
Inefficient bandwidth usage (bandwidth is narrow, too)
Expensive transmission hardware for good coverage
Bad mobile reception, requires stational big antenna
Adding of new services nearly impossible
⇒ Replaced by DVB (Digital Video Broadcasting)
Satellite: DVB-S
Cable: DVB-C
Terrestrial: DVB-T
handheld: DVB-H
} HQ, new services addable, mobile reception, FEC
+ Enabling Multimedia Home Platform (MHP)
extended teletext, games, interactive services, DVB-Java / HTML
E-Commerce / E-Banking
For interactive services, separate feedback channel required
+ More efficient bandwidth usage: multiple DVB programs transmitted instead of one analog TV channel
+ Encryption possible
- new Hardware required
- no interoperability between HW-Devices
- Fails abrupty if signal too weak
- 1-2s delay
- slow zapping
Encoding: MPEG-2: 2 Mb/s - 8 Mb/s
Packetized Elementary stream (PES)
different data streams of each single TV channel
Max 64 kB/s Packet

DVB-S
Satellites relay signals received from the ground (uplink) using so called transponders (downlink)
Solar energy used
Quadrature Phase Shift Keying (QPSK) used for modulation
DVB-S2
since 2005 using 64-QAM, up to 30% more efficient
using H.264, enabling HDTV

DVB-C
Television transmitted using broadband coaxial cable
47-470 MHz
incorporating feedback channel capability (10-46 MHz, can be disturbed by modems)
Also using QAM, data rates identical to DVB-S, injection possible

DVB-T / DVB-T2
transmitted via ground transmitters
should be receivable anywhere, good coverage
stationary or portable
depending on position & orientation ⇒ small antennas sufficient
8 MHz channels (old)
limits available quality, using statistical multiplexing
prone to multipath propagation
↳ signal travelles via different paths, echoes multiple times
use this as signal strengthening by making signals longer
⇒ stronger, but slower signals

DVB-T Modulation
high quality requires fast signals
Multipath propagation leads to signal speed limitation
⇒ transmit symbols in parallel instead of sequential
Guard Interval T_G
use coded orthogonal frequency division multiplex (COFDM)

Single Frequency Network (SFN)
different senders @ different locations transmit the same signal on the same frequency
carefully synchronized, must send isochronously
⇒ larger reception area
more efficient use of bandwidth spectrum
better quality due to amplification / overlapping of signals
Playout center controls multiple sending towers
Towers send @ exact same time T
can't be built arbitrarily large
as soon as signals from other senders arrive out of G_T, disturbances occur
Trade-off G_T: small: good bit rate, small SFN
big: low bit rate, big SFN
Mobile reception: up to 500km/h possible receiver speed in 2k mode
8k mode: up to ~100km/h - 200km/h
8k allows cheap & good coverage & indoor reception
DVB-T can exhibit visually perceivable quality drops due to lower bandwidth
DVB-T2 uses more efficient codec: MPEG-4 ⇒ HDTV

DVB-H
issued for handheld devices
big antenna, large batteries
similar to DVB-T but especially developed to suit handheld receivers back
IP-based ⇒ "future proof"