Self-Study Exercise

Block lecture

Multimedia Coding

- Methods and Applications -

Part 6: Transmission of Media Signals

Dr. Uwe Rauschenbach

Multimedia Coding

Part 6: Transmission of Media Signals

6.1 Overview

6.2 MPEG-2 transport streams

6.3 The protocol family for IP-based media delivery

6.4 Further information

Overview
Transmission of media signals

- **„Transmission“ linked to the physical transport of a medium – like CD – “sneakers network”**
- **„Transmission“ over networks**
  - Broadcast centric (the oldest form, analogue → digital)
  - Data networks (IP centric)
  - Mobile networks (mobile broadcast, mobile telecom → also using IP)
- **Fundamentals**
  - MPEG-2 TS – all-in-one
  - IP / RTP / FLUTE – family of protocols which are designed to work together

Overview
Taxonomy of multimedia transport concepts

- **Content Delivery: Dimensions**
  - Real-Time (Streaming) vs. Non-real-time (File transfer)
  - On demand (Unicast) vs. Push (Multicast)

	On Demand (Unicast, 1:1)	Push (Multicast, 1:N)
Real time	Audio/Video **Streaming on Demand** → RTSP+RTP	Audio/Video **Live Streaming** → MPEG-2 TS → SDP+RTP
Non-realtime	**Download** Services → HTTP	**Carousel** Services → MPEG-2 DSM/CC → FLUTE + ALC

- **Needed additionally: Signalling**

Multimedia Coding
Part 6: Transmission of Media Signals

6.1 Overview

6.2 MPEG-2 transport streams

6.3 The protocol family for IP-based media delivery

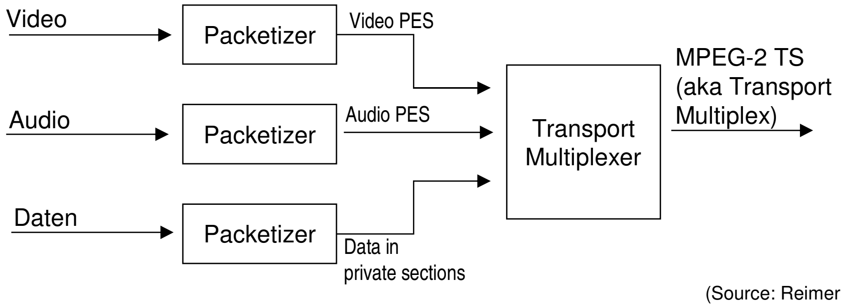
6.4 Further information

MPEG-2 Transport streams
Introduction (1)

- **Standardized as part 1 of the MPEG-2 standard (ISO/IEC 13818-1)**
  - MPEG-2 TS defines a data format for multiplexing, NO video coding format!
- **MPEG-2 TS allow the synchronized and non-synchronized transmission of**
  - Media streams (Video, Audio) → Packetized Elementary Streams (PES)
    - ❍ Video encoding using e.g. MPEG-1, MPEG-2, MPEG-4, H.264
  - Signalling information (e.g. Elektronic Program Guides, EPG) → PSI/SI tables
  - Additional data
    - ❍ Files → Object carousels
    - ❍ Data packets (e.g. IP packets) → Multi Protocol Encapsulation (MPE)
    - ❍ Application data → Private Sections
  - Some of these functionalities are defined on top of MPEG-2 TS by other standards (DSM-CC, DVB)

MPEG-2 Transport streams
Introduction (2)

- **Multiplexing: basic principle**
  - Video, audio and data are packetized (Packetized Elementary Streams, PES)
  - Each stream is assigned a unique Program ID (PID)
  - The packet headers may contain time stamps to synchronize the data streams
  - The packets from different PIDs are inserted into the stream in an alternating manner



(Source: Reimers, p. 92)

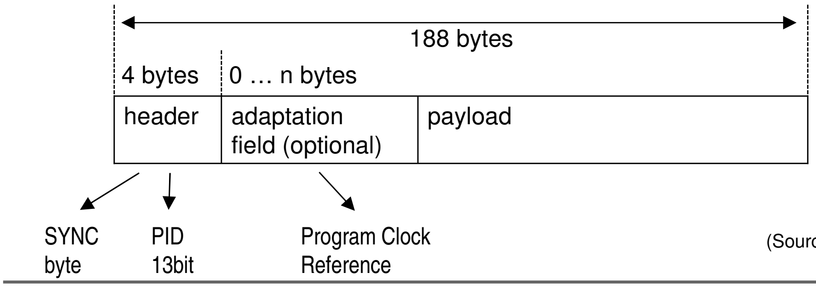
MPEG-2 Transport streams
Transport stream vs. Program stream

- **Program stream**
  - all elementary streams have the same time basis → one program per multiplex
  - packets have variable length
  - suitable for error-free channels of high bitrate – e.g. hard disk and DVD

- **Transport stream**
  - multiple time bases are possible → multiple programs per multiplex
  - packets of fixed length (188 bytes)
  - suitable for the transmission in error-prone channels
  - the foundation of digital TV (DVB-T/S/C/H)

MPEG-2 Transport streams
Transport stream packets

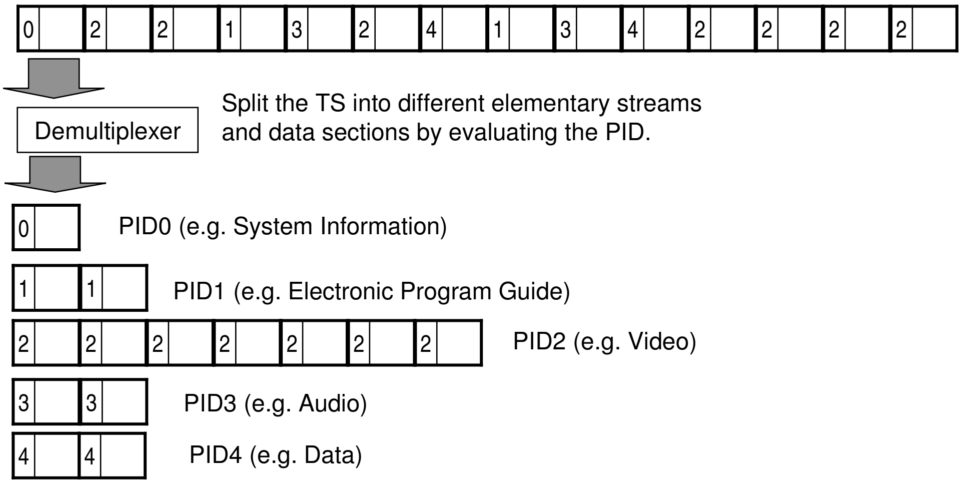
- **An MPEG-2 TS consists of packets of 188 bytes.**
  - each packet starts with a marker (SYNC Byte) of value 0x47, allowing to detect packet start.
  - packet header contains the PID, to identify the PES to which the packet contributes and to allow demultiplexing.
  - optionally, packet contains the value of the System Time Clock (STC), sampled at the time when the packet was sent, as Program Clock Reference (PCR).
  - all data in the MPEG-2 TS are carried in such TS packets.



(Source: Reimers, p. 99)

MPEG-2 Transport streams
Principle of demultiplexing

MPEG-2 TS (multiplex) as sequence of packets. Each elemantary stream is identified by a PID.



Split the TS into different elementary streams and data sections by evaluating the PID.

PID0 (e.g. System Information)

PID1 (e.g. Electronic Program Guide)

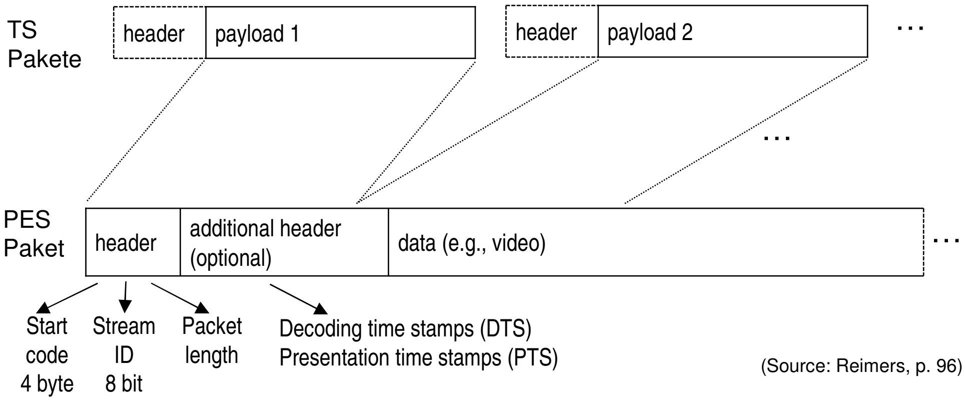
PID2 (e.g. Video)

PID3 (e.g. Audio)

PID4 (e.g. Data)

MPEG-2 Transport streams
PES: Packetized Elementary Streams

- Each elementary video/audio stream is transmitted as PES in the TS, by packaging PES packets into transport stream packets.
- PES may carry synchronization information (DTS and PTS)



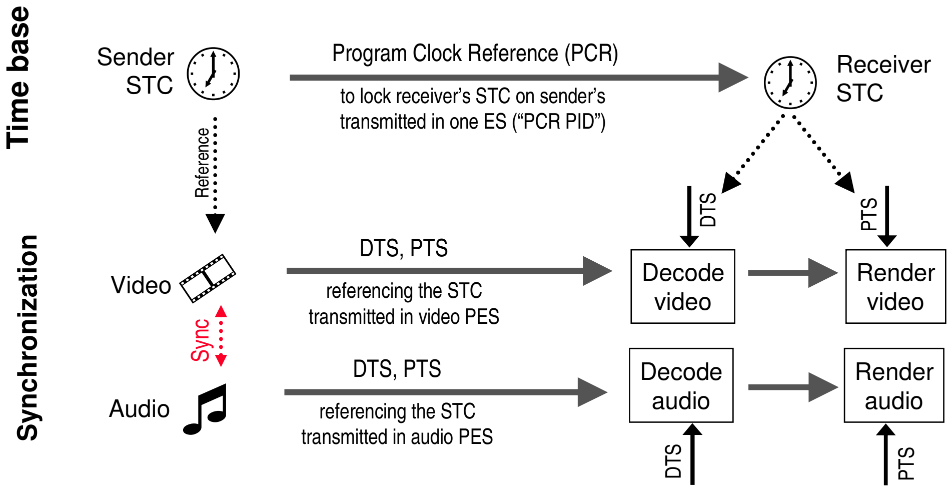
(Source: Reimers, p. 96)

MPEG-2 Transport streams
Important fields in the PES header

- **Start code: Bit pattern that signals the start of a packet**

  binary 0000 0000 0000 0000 0000 0001 (0x000001)
- **Stream ID: Signals the type of the data in the PES (e.g. MPEG-2 video)**
- **Packet length**
  - Packets may have variable length!
- **DTS: Decoding Time Stamp**
  - Time of decoding the first frame in this PES packet
- **PTS: Presentation Time Stamp**
  - Time of rendering of the first frame in this PES packet
  - Allows synchronization of streams

MPEG-2 Transport streams
Timing and Synchronization (1)


Time base
Synchronization
Sender STC
Reference
Program Clock Reference (PCR)
to lock receiver's STC on sender's transmitted in one ES ("PCR PID")
Receiver STC
DTS
PTS
Video
Sync
Audio
DTS, PTS
referencing the STC transmitted in video PES
DTS, PTS
referencing the STC transmitted in audio PES
Decode video
Render video
Decode audio
Render audio
DTS
PTS

MPEG-2 Transport streams
Timing and Synchronization (2)

- **Time base**
  - to synchronize audio and video, a common time base is needed.
  - it is realized by a System Time Clock (STC) of frequency 27MHz in sender and receiver
  - the clock in sender and receiver are aligned at least every 100 ms.
  - this alignment is facilitated by sampling the current value of the sender STC and transmitting it as PCR (Program Clock Reference). Using the value of the PCR, the STC in the receiver is continually re-adjusted and such locked with the clock in the sender.
  - the PCRs are transmitted in the transport stream packets of selected PIDs.
- **Synchronization**
  - for each video and audio packet, the point in time can be determined at which the packet must be provided as input to the decoder (Decoding Time Stamp, DTS), and the point in time at which the decoded frame must be rendered (Presentation Time Stamp, PTS).
  - DTS and PTS refer to the current value of the STC.
  - they are transmitted in PES packets.

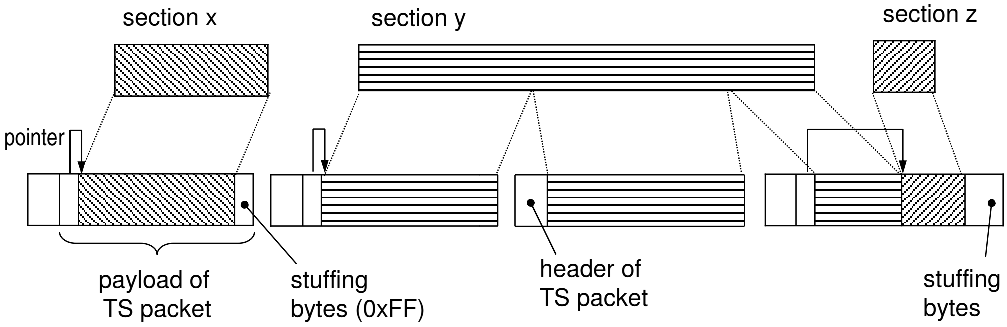
MPEG-2 Transport streams
Sections (1)

- **Sections are used for transmitting data not needing synchronization.**
  - Signaling information (PSI/SI tables)
  - Files and data packets (the extensions *Object carousel* and *MPE* are based on sections
  - Applications specific data are transmitted in **Private Sections**
    - extension of Sections
    - minimal structure defined, allowing to parse a stream even if not knowing the application-specific section structure
    - maximum 4096 Bytes per section

- **Sections are packaged into TS packets**
  - one section packed into one or more TS packets
  - one TS packet may contain one or two (partial) sections

MPEG-2 Transport streams
Sections (2)

Packaging of sections into TS packets



(Source: Reimers, p. 102)

- first byte in packet payload: pointer
- presence of pointer is signaled by a byte in the section header

MPEG-2 Transport streams
Signaling information - overview

- **Metadata are needed to inform the receiver about which components are contained in the signal and about the services**
- **Basically, they describe what is transmitted in the PIDs**
- **Two kinds of metadata tables**
  - PSI: Program Specific Information → signal the structure of the MPEG-2 TS
  - SI: Service Information → used by Services (e.g. Digital TV) → see Service Enabler part of the script.

MPEG-2 Transport streams
Signaling information – PSI tables

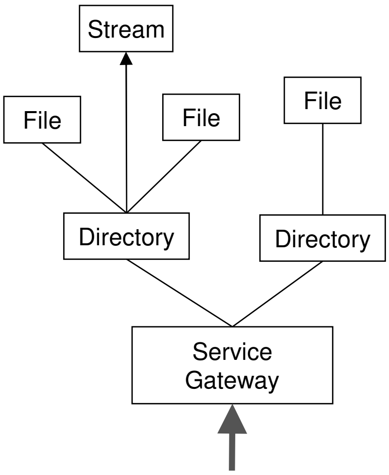
- **PAT:** Program Association Table (once per TS)
  - list of all services in the TS;
  - per service a pointer to PID of Program Map Table (PMT) plus pointer to PID with Network Information Table (NIT)
  - always at PID=0
- **PMT:** Program Map Table (once per service)
  - list of all PIDs of a service
  - pointer to PID with time basis (PCR)
  - mandatory
- **NIT:** Network Information Table (at least once per TS)
  - signals network parameters (e.g. ID, orbital position, transponder number, frequency)
  - mandatory for parameters of the current network (NIT_actual), optional also for parameters of neighboring / related networks (NIT_other)

MPEG-2 Transport streams Object carousels (1)

- **Digital-TV is (potentially) more than Video and Audio!**
  - transmission of data objects (e.g. files) and data structures (directories, links) enables „Rich Media TV“ (e.g. HTML pages or Java applets accompanying a program)
  - these data must be transmitted in the TS
- **TV is a broadcast service!**
  - receivers may join / leave a session at any time
  - head end does not know the receivers
  - → data must be repeated periodically → **carousel!**
  - → (old) example: Videotext
- **Carousels are based on the DSM/CC Specification (Digital Storage Media – Command and Control)**
  - part 6 of MPEG-2 standard (ISO/IEC 13818-6)
  - carousels are transmitted in sections in the MPEG-2 TS

MPEG-2 Transport streams Object carousels (2)

- **… contain structure elements**
  - directories
  - files
  - links to a/v streams
  - Service Gateway (special directory - root)
- **… are based on BIOP messages (Broadcast Inter-ORB Protocol, a subset and variation of CORBA)**
- **… are mapped to sections**
  - one or more BIOP messages are mapped into one module
  - each module is split into blocks
  - each block fits into one section



Stream

File

File

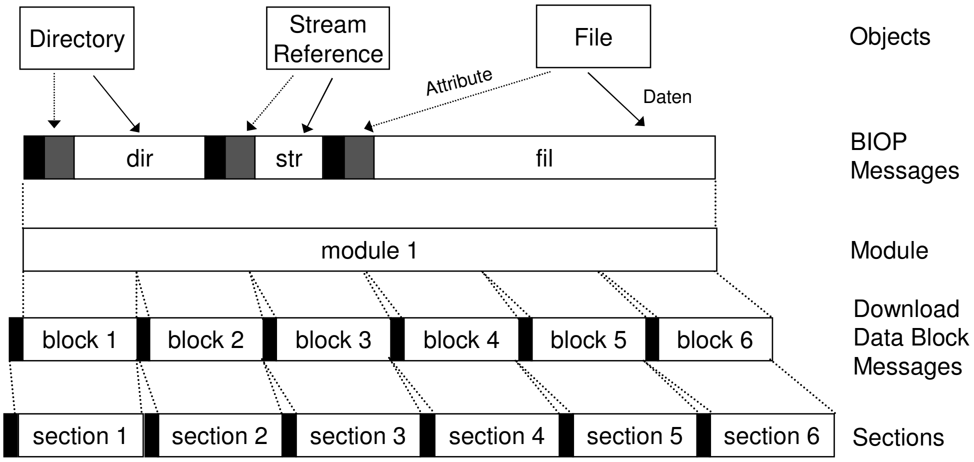
File

Directory

Directory

Service Gateway

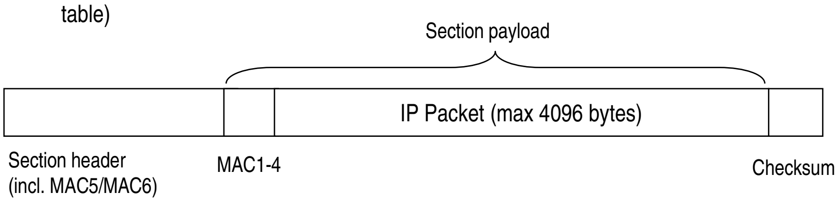
MPEG-2 Transport streams
Object carousels (3)



(Nach Reimers, 294)

MPEG-2 Transport streams
Multi Protocol Encapsulation

- **Multi Protocol Encapsulation (MPE) enables the transmission of packetized data in sections**
  - allows the tunneling of network protocols through MPEG-2 TS
  - optimized for the transport of IP packets
- **Transport of IP Packets in MPE**
  - transmission mode: Multicast
  - two bytes of the MAC address are transmitted in section header (pre-filtering in hardware possible!), other 4 bytes in section payload
  - max. packet size restricted to 4096 Bytes → no fragmentation into multiple sections needed
  - signaling of the mapping between IP and MAC addresses in SI table → INT (IP/MAC notification table)

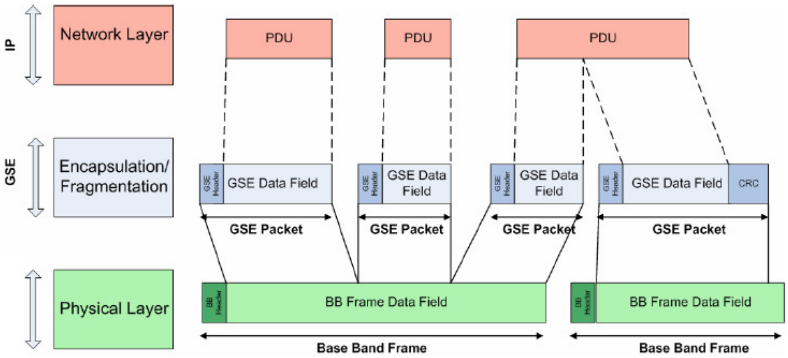


MPEG-2 Transport streams
Multi Protocol Encapsulation: IP platforms

- **Problem:**
  - set of IP addresses in one network (here: TS) must be free of conflicts
  - but: traffic from multiple providers which do not harmonize their IP address spaces is often multiplexed into a single TS
- **Solution: IP platform concept**
  - one *“IP platform”* contains all IP traffic and device population of one operator. A platform must be free of IP and MAC address conflicts.
  - an additional “IP platform id” is introduced which acts as a filter: a device will only see the IP traffic from the platform of the currently selected provider

New Encapsulation Approach
GSE – Generic Stream Encapsulation

- **The new DVB Standards family (DVB-S2 and upcoming T2/C2) will allow to map IP Packets and other Protocol Data Units directly onto base band packets of the Bearer layer *without using MPEG-2 TS and MPE***
- **This decreases the packet overhead from 10% (MPE) to 2-3%**

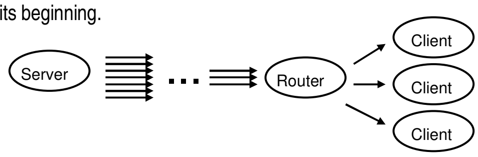


Multimedia Coding
Part 6: Transmission of Media Signals

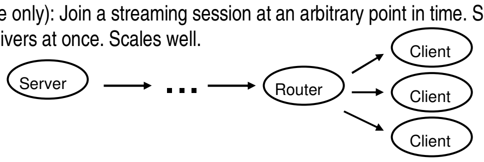
6.1 Overview
6.2 MPEG-2 transport streams
6.3 The protocol family for IP-based media delivery
6.4 Further information

IP Streaming
Overview (1)

- **IP Streaming:**
  - Video and/or Audio are transmitted in real-time over an IP network.
  - Data streams are intended for immediate use: receive, display, forget
- **2 Modes:**
  - ❍ Unicast (live or non-live): One stream per client. Scales poorly. In non-live mode, each client gets the stream from its beginning.



  - ❍ Multicast (live only): Join a streaming session at an arbitrary point in time. Streams are distributed to multiple receivers at once. Scales well.

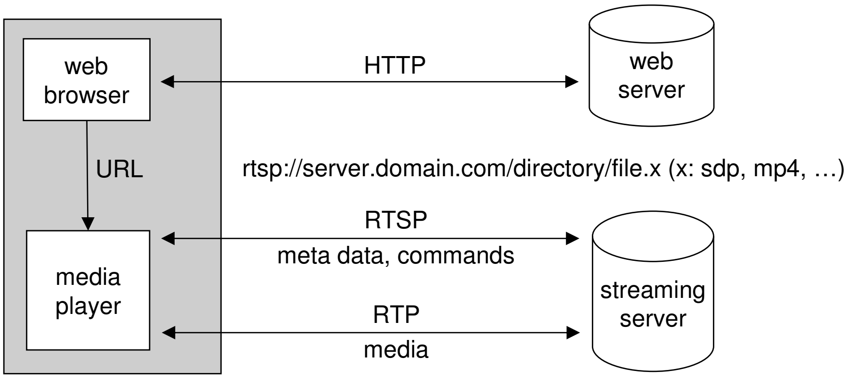


IP Streaming Overview (2)

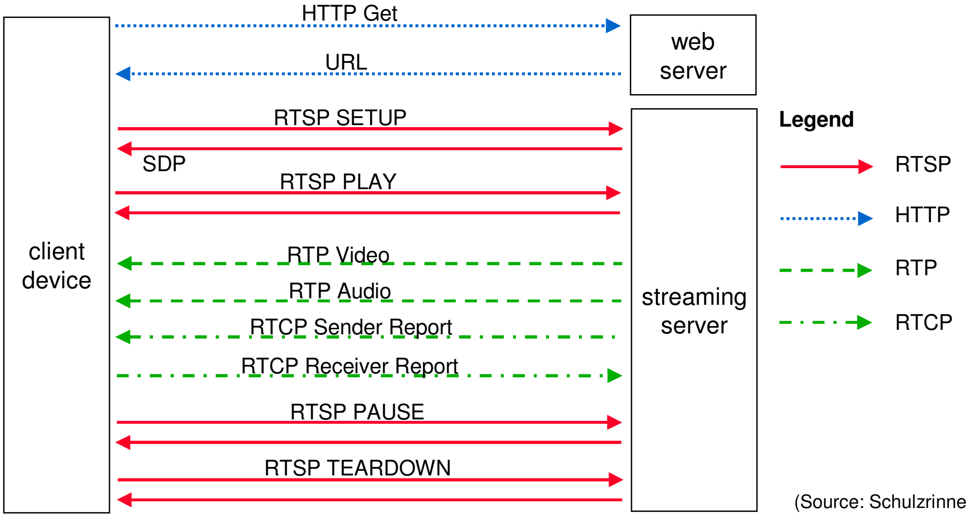
- **Addressing**
  - **Unicast:** Server and client know each other
    - ❍ Client informs server about its IP address
    - ❍ Server sends data packets to this address
  - **Multicast:** Client knows the server. A path for the data packets through the network is switched, upon request ("multicast join") from the client
    - ❍ Server sends Multicast Packets into the network
    - ❍ Client knows/is told the Multicast address
    - ❍ Client requests from his router a multicast join, i.e. the forwarding of the packets (IGMP: Internet Group Management Protocol, RFC 2111)
    - ❍ Client receives the packets
- **UDP is the foundation of all protocols for IP streaming**
- **Family of protocols for IP streaming**
  - RTSP (Real Time Streaming Protocol, RFC 2326): Managing non-live streaming sessions
  - RTP (Real Time Protocol, RFC 3550): Transmission of the media streams
  - RTCP (Real Time Control Protocol, RFC 3550): Transmission of control information for RTP

IP Streaming: RTSP Protocol overview

- **RTSP is a protocol to control media streaming sessions.**
- **stateful (client and server maintain session status, reference: Session ID)**
- **System environment (source: Schulzrinne)**



IP Streaming: RTSP
Steps of a streaming session (1)



(Source: Schulzrinne)

IP Streaming: RTSP
Steps of a streaming session (2)

- **1. Transmit the parameters of the media session (as SDP, see later)**
  - SETUP
  - DESCRIBE
- **2. Start the media session**
  - PLAY
    - ❍ Parameters
      - range: Segment (time interval) from media stream. Time format: NPT, SMPTE Time Code, wallclock
      - scale: Play back rate
      - transport: IP multicast group
    - ❍ several PLAY commands can be in sequence
  - Server reaction to a PLAY command
    - ❍ Unicast: Server starts the stream and sends it to the client
    - ❍ Multicast:
      - Server starts the stream if it is not already running
      - (Client must join the multicast group in order to receive the stream)

IP Streaming: RTSP
Steps of a streaming session (3)

3. Transmit media streams via RTP. In parallel, control commands are transmitted via RTCP.

4. PAUSE pauses the transmission. Use PLAY to continue.

5. TEARDOWN closes the session and deletes the associated status in the server.

IP Streaming: RTSP
RTSP commands („methods“)

OPTIONS	Query available options
SETUP	Create a session
ANNOUNCE	Change the description of media objects
DESCRIBE	Query the description of a media object
PLAY	Start a stream
RECORD	Start recording (implemented rarely!)
REDIRECT	Redirect to another server
PAUSE	Pause streaming, keep session state
SET PARAMETER	Control the player resp. encoder
TEARDOWN	Close session, delete state in the server

IP Streaming: SDP
Session description using SDP (1)

- **SDP – “Session Description Protocol”**
  - strictly speaking, no protocol but a format for signaling session information
  - syntax: <field>=<value>CRLF
  - RFC 4566
- **what information is contained?**
  - name and lifetime of the session
  - description of media streams (audio, video, data)
    - ❍ IP address, port number, payload type (i.e. codec and packetization format) of each stream
    - ❍ format-specific parameters may be signaled
    - ❍ bandwidth requirements
    - ❍ attributes to signal application-specific information

IP Streaming: SDP
Session description using SDP (2)

- **SDP structure** (* → optional)

```
v= (protocol version)
o= (owner/creator and session identifier).
s= (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information - not required if included in all media)
b=* (bandwidth information)
One or more time descriptions (see below)
z=* (time zone adjustments)
k=* (encryption key)
a=* (zero or more session attribute lines)
Zero or more media descriptions (see below)
```

IP Streaming: SDP
Session description using SDP (3)

- **Time description**

```
t= (time the session is active)
r=* (zero or more repeat times)
```

- **Media description**

```
m= (media name and transport address)
i=* (media title)
c=* (connection information – optional if included at session-level)
b=* (bandwidth information)
k=* (encryption key)
a=* (zero or more media attribute lines)
```

IP Streaming: SDP
Some fields detailed (1)

- **“a=” field („Attribute“)**
  - the place to put application-specific extensions
  - a=name[:value]
  - Example: a=myExtension:255
- **“m=” field („media announcement“)**
  - m=<media> <port> <transport> <fmt list>
  - Examples:

```
m=video 49232 RTP/AVP 0
m=audio 49230 RTP/AVP 96
```

  - Parameters
    - media: „audio“ | „video“ | „text” | „application“ | „message“
    - transport: „RTP/AVP“ | „udp“
      - RTP/AVP: RTP with Audio/Video profile
    - fmt list: List of 7bit format numbers (Payload-types) - see RFC 1890
      - new payload numbers can be registered with IANA
      - dynamic payload types: 96 … 127
        → actual payload type is defined by attribute: a=rtpmap<encoding name>/<clock rate> [/<params>]
        → parameters for these types are defined by attribute: a=fmtp <fmtNum> <param list>

IP Streaming: SDP
Some fields detailed (2)

- **"c=" field („Connectivity")**
  - c=<network type> <address type> <connection address>
  - Example: `c=IN IP4 224.2.17.12/127`
  - Parameters:
    - ❍ network type: IN for Internet
    - ❍ address type: IP4 or IP6
    - ❍ connection address
      - Unicast: <Hostname> or <IP Address>
      - Multicast: <IP Address>[[/<TTL>]/<Number of addresses for scalable codecs>]

IP Streaming: SDP
Example

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5         → src IP address
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127                   → IPv4 multicast address, TTL=127
t=2873397496 2873404696                           → session lifetime as NTP
a=recvonly
m=audio 49170 RTP/AVP 14           → audio: port 49170, payload type MPEG Audio
m=video 51372 RTP/AVP 99     → video: port 51372, dynamic payload type – defined below
a=rtpmap:99 h264/90000          → H264 video according to RFC 3984, 90kHz clock rate
a=fmtp:99 profile-level-id=42c00d; packetization-
 mode=1;sprop-parameter-sets=Z0LADZtAoPiA,aN4liA==;
                      → H264 specific parameters, see payload definition at RFC 3984
```

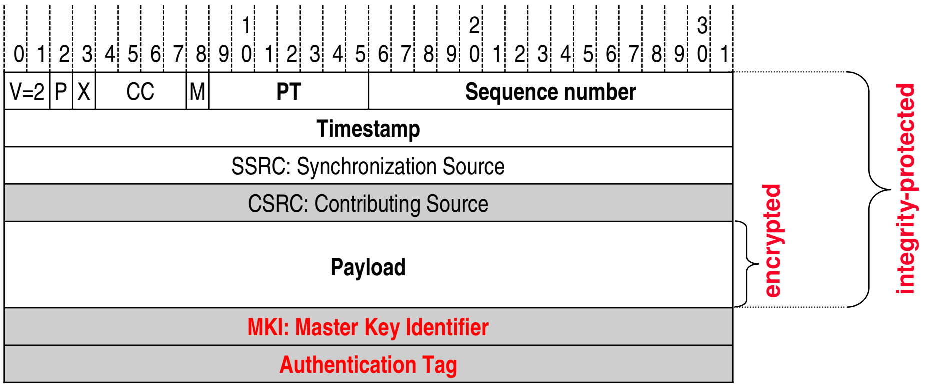
IP Streaming: RTP
Protocol overview

- **RTP is a protocol to transmit media streams in real-time over IP networks**
  - usually, UDP is used as transport layer below RTP (Unicast or Multicast) → unreliable but fast transport
  - supports synchronization of different media streams by time stamping
  - video/audio streaming is just one of the possible applications – a large part of the complexity of RTP results from the support of audio/video conferencing. Other use case: VoIP.
  - usually, audio and video streams are running in different RTP sessions
- **RTP needs profiling**
  - RTP defines just the transport protocol for RTP packets.
  - The actual packetization is specified as payload type that depends on the used codec
- **RTP works in tandem with RTCP for control and synchronization**
  - by convention, RTP uses an even port and RTCP uses the next-higher (odd) port
- **Secure variant: SRTP (Secure RTP)**
  - supports encryption and authentication (i.e. digital signature) of the packet

IP Streaming: RTP
How it works

- **Server sends out a continuous stream of RTP packets**
  - packets may be delayed, swapped or lost during transmission
  - RTP header contains information which allows the client to detect/fix these problems
- **Important fields in RTP packet header**
  - Payload type – to signal what is contained (cf. SDP!)
  - Sequence number – to detect re-ordering and packet loss
  - Time stamp – “sampling time” of the first frame contained in packet.
    - ❍ Format: NTP timestamp (reduced precision)
    - ❍ starting with random value to hinder attacks on encrypted RTP streams
    - ❍ monotonically increasing unless the data are encoded in another order than sampled (e.g. B frames)

IP Streaming: SRTP
Encrypted variant of RTP



0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1

V=2	P	X	CC	M	**PT**	**Sequence number**

Timestamp

SSRC: Synchronization Source

CSRC: Contributing Source

Payload

MKI: Master Key Identifier

Authentication Tag

red: new in SRTP

MKI: Index of the master key, from which keys are derived to encrypt and/or authenticate the current packet

Authentication Tag: Optional integrity protection of the packet.

NB: the actual keys are transmitted by other means.

IP Streaming: RTCP
Protocol overview

- **RTCP is used to control RTP sessions**
  - uses UDP as transport layer (Unicast or Multicast)
  - transmits timeline information for synchronization
  - transmits content descriptions
  - feeds back information about the Quality of Service
  - transmits Retransmission Requests
  - identifies the participants in an audio and video conference
- **Types of RTCP Packets**
  - **SR - Sender Reports: Server sends control information to clients**
    → Rate information, Synchronization information
  - **RR - Receiver Report: Client sends control information to server(s)**
    → Packet loss, delay jitter, round trip times
  - SDES - Source description: for conferences
  - BYE - Explicit leave (for conferences)
  - APP – Application specific

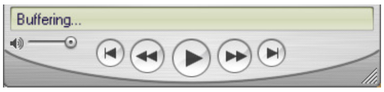
IP Streaming
Synchronization with RTP and RTCP

- **Principle**
  - each *RTP* packet contains a timestamp, which signals the relative time of this packet in the sequence
  - to map this to wallclock time, a pair (wallclock, RTP timestamp) is sent in each *RTCP* sender report
    - as SRs are sent infrequently, client must maintain its own clock and adjust it according to updates in SR
  - the client can compute absolute time to sync audio and video by using this mapping
- **Timing inaccuracy**
  - RTP has been designed to work in non-constant delay environments
  - to compensate delay, it is up to each implementation to chose a buffer size
  - consequence: in contrast to MPEG-2 TS, no tight timing control is possible but delay depends on the implementation
- **Example**
  - Audio: RTP timestamp=10.8, last audio SR contains (27.4, 10) → t=28.2
  - Video: RTP timestamp=309.7s, last video SR contains (23.1, 304.3) → t=28.5
  - Client must play out audio sample at its "local wallclock time" of 28.2 s and video frame at 28.5 s

IP Streaming
Quality of Service (1)

- **IP networks usually just offer „*best effort*" *QoS*, meaning no control of QoS is possible. This has the following implications:**
  - the routing algorithms in IP networks (queues, multi-path transmission) lead to *delays* whose duration may vary (*delay jitter*).
  - Erroneous transmissions and queue overflows lead to *packet loss*.
  - Packets, which arrive after their playout deadline due to long delay are discarded also (conditional packet loss)
- **Excursus: besides *best effort,* there are two more QoS control methods (which, however, must be implemented in all routers on a packet's path in order to work). Here's the list:**
  - **Best effort**: No control.
  - **DiffServ** (Differentiated Services): Data packets are classified into traffic classes (using priorities). Packets with higher priority (e.g. VoIP) are forwarded with precedence.
  - **IntServ** (Integrated Services): Using the RSVP protocol, a virtual connection with guaranteed QoS can be reserved end-to-end for a particular IP flow.

IP Streaming
Quality of Service (2)



Buffering...

- **Compensation of Delay Jitter:**
  - Buffering of data packets, at the expense of a longer start time.
  - Buffer dimensioning
    - ❍ Compromise between short startup time and low conditional packet loss.
    - ❍ If upper bound of delay jitter is known, recommended buffer size is a bit more than the product of maximum delay jitter and playout bit rate
  - Shortening the start delay without changing the buffer size: *Progressive Streaming*
    - ❍ At startup, server sends data at a higher rate than the playout rate. This fills the buffer more quickly.
    - ❍ Client and Server must agree on speed and duration of the accelerated transmission.
    - ❍ Progressive Streaming only works if spare bitrate is available.
- **Compensation of packet loss:**
  - Request a retransmission of the packet(s)
  - Sending of redundant information (Forward error correction, FEC)
  - Error Concealment at application layer

IP Streaming
Function of a streaming server

- **Streaming Server**
  - accepts requests from clients to start streams (Unicast Streaming) or starts streams by control command / playlist which are then distributed by multicast
  - reads media file with additional information („Hints“) and packetizes it according to the hints OR receives data from a live encoder
  - sends the packets controlled by a timer
- **Streaming servers should be able to deliver a large number of streams simultaneously.**
  - → Parsing of the files to be streamed must be simple
  - → Idea: Use external application to pre-process the media data (e.g. MPEG-4), adding a hint track per media track (Audio, Video). This hint track is either appended to the media file or stored separately.
  - → Streaming server only needs logic to parse the hint tracks; the actual streaming is done by just copying byte ranges from the media file to the output packets, controlled by the hint tracks

IP Streaming
Hinting (1)

- **Hint track contains all control information that is necessary for streaming**
  - Header information
  - Timing information
  - SDP fields for the referenced track (e.g. m=audio 0 RTP/AVP 96)
  - RTP packetization information
  - Table of links to the actual data blocks to be sent („Hint Track Samples“)
- **Hint Track Sample**
  - a Hint Track Sample controls the generation of a sequence of RTP packets with the same time stamp
  - Fields
    - ❍ RTP Header, Timing & Sequence Information
    - ❍ Data table with 3 different categories of building blocks for the RTP stream
      - Inline data
      - Links to other hint track samples (Metadata)
      - Links to media tracks

IP Streaming
Hinting (2)

Hinting with the MPEG4IP tools

The tool mpeg4creator allows adding hint tracks to MPEG-4 files.

List the tracks

```
> mp4creator -list file.mp4
Track Type Info
1 od Object Descriptors
2 scene BIFS
201 video MPEG-4, 166.633 secs, 262 kbps, 320x240 @ 14.16 fps
101 audio MPEG-4, 166.528 secs, 49 kbps, 32000 Hz
```

Add a hint track per audio and video track

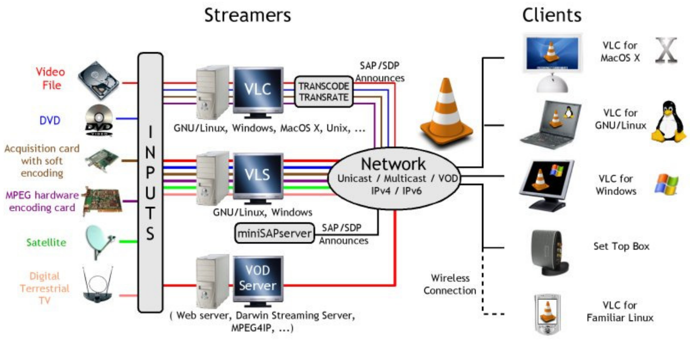
```
> mp4creator -hint=201 file.mp4
> mp4creator -hint=101 file.mp4
```

IP Streaming
Selected Software Tools (1)

- **MPEG4IP**
  - multiplexer, hinting tool
  - streaming client
  - “grown” Software, i.e. difficult to read
- **ffmpeg**
  - powerful (and peculiar) command line tool for transcoding a lot of media formats
- **Darwin Streaming Server**
  - Streaming Server implementation under Apple’s Community Source License
  - stable, scales well
  - sparsely documented
- **Helix Streaming Server**
  - Pendant to the Darwin Server from Real Networks

IP Streaming
Selected Software Tools (2)

- **VideoLAN**
  - Streaming Server for Windows and UNIX/Linux available, Client for many platforms (Windows, Linux and embedded Linux)
  - Large set of codecs
  - well-structured code
  - both graphical and cmd line UI available



Data carousels over IP multicast
Overview

- **Data carousels for the unidirectional file transfer**
  - cyclic traversal and transmission of a (possibly structured) set of objects
  - no session setup needed, *„Tune in at any time“*
  - already known: DSM/CC object carousels in MPEG-2 TS
- **Protocol family for the object/file transfer via IP multicast**
  - Building blocks to solve the problems in uni-directional data transport
    - ❍ **LCT:** Layered Coding Transport (RFC 3451) – baseline protocol for unicast object transfer
    - ❍ **FEC:** Forward Error Correction (RFC 3452 and 3695) – allows error correction by proactively transmitting redundant information. Framework for different methods.
    - ❍ **ALC:** Asynchronous Layered Coding (RFC 3450) – Binds FEC and Session concept to LCT
  - Underspecified protocol framework; Instantiation needed!
- **File Transfer**
  - **FLUTE:** File deLivery over Unicast transport (RFC 3126) – Special instantiation of ALC for the transmission of files, adding headers and metadata to LCT/ALC
  - In the following, with “FLUTE” we mean the combination of LCT/ALC/FEC/FLUTE.

Data carousels over IP multicast
ALC protocol description (1)

- **ALC allows the multicast transmission of *objects***
  - object: set of packets that belong together
  - identified using a unique TOI (Transport Object Identifier) in packet header
  - transmitted in a session which is identified by TSI (Transport Session Identifier) in packet header
- **FEC and multichannel**
  - most simple FEC (“Null-FEC”) defined to add sequence numbers to ALC packet header
    - ❍ packet loss can be detected and packet reordering can be detected and corrected
  - allows to use multiple channels for the transmission
    - ❍ one channel is sufficient to receive all data
    - ❍ receiving multiple channels increases the data rate
    - ❍ additional channels may contain redundant data for FEC
    - ❍ upon network congestion, channels may be dropped

Data carousels over IP multicast
ALC protocol description (2)

- **Session management**
  - 2 flags in the header are used for session management
  - **A-Flag (Close Session flag):** Usually 0. Signals the upcoming end of a session when set to 1 in the last packets of a session.
  - **B-Flag (Close Object flag): Usually 0. Signals the** upcoming end of a the transmission of an object when set to 1 in the last packets belonging to that object.

Data carousels over IP multicast
FLUTE protocol description

- **FLUTE has been designed to transmit files (including directory trees) based on ALC**
  - for that, FLUTE re-uses the TOI concept from ALC
  - a TOI in ALC corresponds to a version of a file in FLUTE, i.e. new version of a file ➔ new TOI!
- **To transmit files/directories, metadata are required to describe the file that's carried by a transport object. They include:**
  - file and directory name
  - file type (MIME)
  - transport encoding (e.g. gzip)
  - filesize (uncoded, coded)
- **Those metadata are carried in a FDT (File Description Table) on TOI=0**
  - thus, the FDT is an object by itself, always carried on the same (reserved) TOI
  - each file update requires a new version of the FDT
  - as TOI changes can not be used for versioning of the FDT, a header extension (EXT_FDT) has been defined for that, carrying the FDT Instance ID. This extension is only used on TOI=0.

Data carousels over IP multicast
FLUTE: File Description Table (FDT)

The FDT is an XML file. Example:

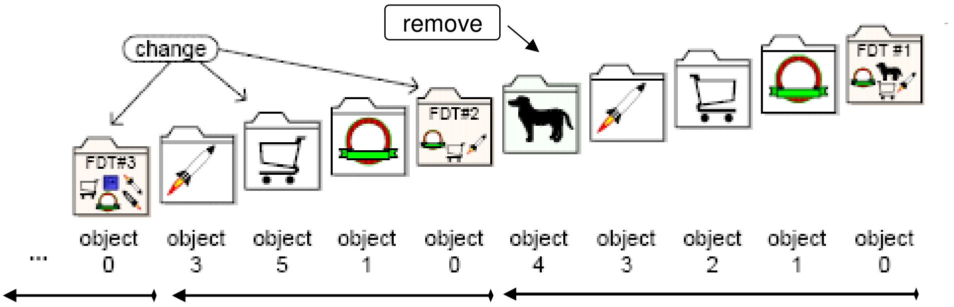
```
<FDT-Instance Expires="2890842807">
     <File Content-Location="menu/tracklist.html"
           TOI="1"
           Content-Type="text/html"/>
     <File Content-Location="tracks/track1.mp3"
           TOI="2"
           Content-Length="6100"
           Content-Type="audio/mp3"
           Content-Encoding="gzip"
           Content-MD5="+VP5IrWploFkZWc11iLDdA=="/>
</FDT-Instance>
```

Data carousels over IP multicast
Dynamic FLUTE sessions

- **Each update of a file requires according signaling in FLUTE.**
- **Sender's algorithm**

1) Update the file
2) Stop sending data packets on “old” TOI,
   announce this by setting the B-Flag in the last packets
3) Assign a new TOI to the updated file
4) Update the FDT by inserting the new TOI
   (and possibly updating other metadata as needed)
5) Transmit updated FDT on TOI=0
   (with incremented FDT Instance ID in EXT_FDT header field)
6) Transmit updated file on “new” TOI

Data carousels over IP multicast: Example
Dynamic data carousel using FLUTE in DVB-IPDC



FDT inst 2
File 1 (toi=1)
File 2 v2 (toi=5)
File 3 (toi=3)
File 5 (toi=6)
File 6 (toi=7)

FDT inst 1
File 1 (toi=1)
File 2 v2 (toi=5)
File 3 (toi=3)

(File 4 removed)

FDT inst 0
File 1 (toi=1)
File 2 (toi=2)
File 3 (toi=3)
File 4 (toi=4)

Source: ETSI TS 102472

Data carousels over IP multicast
Who needs them?

- **Mobile Multimedia is more than Streaming!**
  - Future services will also use Filecast (Carousells).
- **Filecast service scenarios**
  - Distribution of DRM-protected content for which the user can buy access
  - Clipcast/Podcast: the cache of a terminals is filled with content by filecasting. This content can then be consumed offline.
  - Today’s standards for Mobile TV use Filecast to distribute
    - ❍ Electronic Service Guide
    - ❍ Interaktive content in addition to Live TV
  - Distribution of Software
  - …

Data carousels over IP multicast
Selected software tools

- **MAD**
  - FLUTE implementation of the University of Tampere
  - License: GPL
- **MCL**
  - FLUTE implementation of INRIA
  - License: GPL (for academic/private use) and commercial license
- **Ethereal / Wireshark**
  - Toll to capture and analyze network traffic, with many analysis modules, e.g. for LCT/ALC
  - Ethereal has reched end-of-live in 2006. Successor: Wireshark
  - License: GPL

Multimedia Coding
Part 6: Transmission of Media Signals

6.1 Overview
6.2 MPEG-2 transport streams
6.3 The protocol family for IP-based media delivery
6.4 Further information

Further information (1)

- **Books regarding DVB, Video Streaming and MPEG-2**
  - Ulrich Reimers. *DVB: The Family of International Standards for Digital Video Broadcasting.* Springer, Berlin, 2004.
  - The MPEG-2 Book. Edited by Fernando Pereira. Prentice Hall, 2002.
  - Content Networking in the Mobile Internet, Edited by Sudhir Dixit and Tao Wu, ISBN 0-471-46618-2 John Wiley & Sons, 2004.
- **RTP, RTCP, RTSP**
  - Homepage of H. Schulzrinne. http://www.cs.columbia.edu/~hgs/
- **DVB and MPEG-2 – Specifications**
  - INFORMATION TECHNOLOGY - GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO: SYSTEMS ITU Recommendation H.222.0, ISO/IEC 13818-1, International Standard
  - DVB Specifications
    - ○ some are freely available as “blue books” from http://www.dvb.org
    - ○ DVB specifications published as ETSI standards can be downloaded from www.etsi.org

Further information (2)

- **RFCs**
  - RFC3550 – RTP + RTCP http://www.ietf.org/rfc/rfc3550.txt
  - RFC2326 – RTSP http://www.ietf.org/rfc/rfc2326.txt
  - RFC4566 – SDP http://www.ietf.org/rfc/rfc4566.txt (replaces RFC 2327)
  - RFC1890 – RTP payload types http://www.ietf.org/rfc/rfc1890.txt
  - RFC3450 – ALC http://www.ietf.org/rfc/rfc3450.txt
  - RFC3451 – LCT http://www.ietf.org/rfc/rfc3451.txt
  - RFC3926 – FLUTE http://www.ietf.org/rfc/rfc3926.txt
  - RFC3695 – Compact FEC Schemes http://www.ietf.org/rfc/rfc3695.txt
  - RFC3452 – FEC http://www.ietf.org/rfc/rfc3452.txt

Further information (3)

- **Software URLs**
  - Darwin Streaming Server: http://developer.apple.com/darwin/projects/streaming/
  - Helix DNA Server: https://helixcommunity.org/
  - FFMPEG: http://sourceforge.net/projects/ffmpeg/
  - VideoLAN: http://www.videolan.org/
  - MPEG4IP: http://mpeg4ip.sourceforge.net/
  - MAD: University of Tampere FLUTE Implementation: http://atm.tut.fi/mad/
  - MCL: INRIA FLUTE Implementation: http://planete-bcast.inrialpes.fr/
  - Ethereal: http://www.ethereal.com/
  - Wireshark: http://www.wireshark.org/