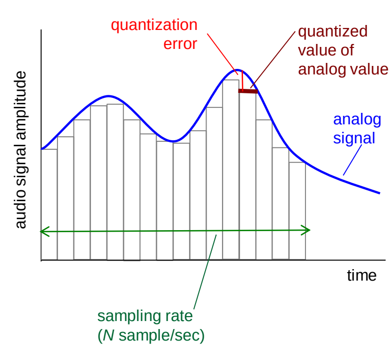
Streaming Multimedia

Dr. T. Venkatesh
Dept of CSE, IIT Guwahati

Multimedia: audio

- analog audio signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
  - e.g., 2^8=256 possible quantized values
  - each quantized value represented by bits, e.g., 8 bits for 256 values

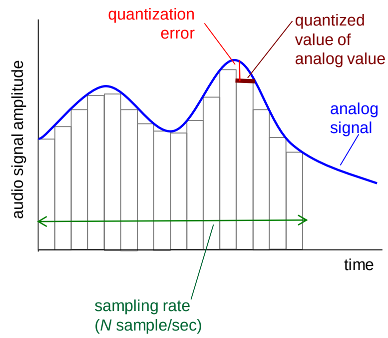


Multimedia: audio

- example: 8,000 samples/sec, 256 quantized values: 64,000 bps
- receiver converts bits back to analog signal:
  - some quality reduction

example rates

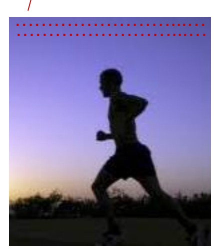
- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet telephony: 5.3 kbps and up



Multimedia: video

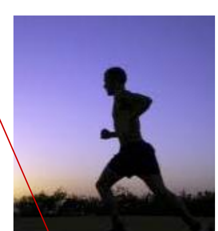
- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

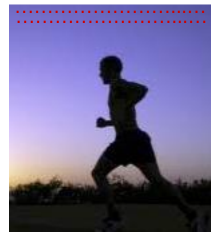


frame *i+1*

Multimedia: video

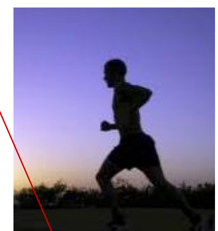
- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i



frame *i+1*

Multimedia networking: 3 application types

- *streaming, stored* audio, video
  - *streaming:* can begin playout before downloading entire file
  - *stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - e.g., YouTube, Netflix, Hulu
  - Delay: from client request until display start can be 1 to 10 seconds
- *conversational* voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance
  - e.g., Skype
  - Stringent delay requirements, 150-400 msec
- *streaming live* audio, video
  - Non-interactive, similar to TV broadcast but on Internet
  - e.g., live sporting event (futbol)
  - Delay sensitive, typically 1-2 seconds, handled with deferred playback

Requirements for Data Transport

- Delay
  - Some small delay at the beginning is acceptable
  - E.g., start-up delays of a few seconds are okay
- Jitter
  - Variability of delay within the same packet stream
  - Client cannot tolerate high variation if buffer starves
  - Buffer starvation causes interruptions in playback
- Loss
  - Small amount of missing data is not disruptive
  - Retransmitting lost packet may take too long anyway

Challenges

- TCP/UDP/IP suite provides best-effort, no guarantees on expectation or variance of packet delay
- Streaming applications delay of 5 to 10 seconds is typical and has been acceptable, but performance deteriorate if links are congested (transoceanic)
- Real-Time Interactive requirements on delay and its jitter have been satisfied by over-provisioning (providing plenty of bandwidth), what will happen when the load increases?...

8

Challenges (more)

- Most router implementations use only First-Come-First-Serve (FCFS) packet processing and transmission scheduling
- To mitigate impact of "best-effort" protocols, we can:
  - Use UDP to avoid TCP and its slow-start phase…
  - Buffer content at client and control playback to remedy jitter
  - Adapt compression level to available bandwidth
  - Over-provision bandwidth, CDN, etc.
- Alternatively, we can change the network:
  - Resource reservations and guarantees and/or
  - Different classes of packets and services
  - Sufficient resources to meet promises

9

Streaming

- Important and growing application due to reduction of storage costs, increase in high speed net access from homes, enhancements to caching
- Interactive control by user (play, pause, seek)
  (but often with long response time)
- Ubiquitous on the web
  - YouTube, Netflix, Vimeo
  - Radio & TV stations
  - News websites
  - Social Networks

10

Streaming Stored Audio and Video

- Client-server system
  - Server stores the audio and video files
  - Clients request files, play them as they download
  - *streaming:* client playout begins *before* all data has arrived
  - Interactive: user can control operation (similar to VCR: pause, resume, fast forward, rewind, etc.)
- Playing data at the right time
  - Server divides the data into segments
  - … and labels each segment with frame id
- Avoiding starvation at the client
  - The data must arrive quickly enough
  - Delay: from client request until display start can be 1 to 10 seconds
  - Timing constraint for still-to-be transmitted data: in time for playout

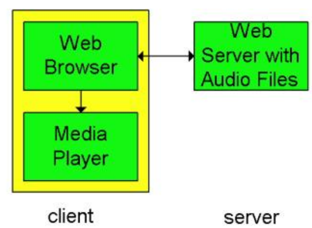
Helper Application

- Displays content, which is typically requested via a Web browser; typical functions:
  - Decompression
  - Jitter removal
  - Error correction: use redundant packets to be used for reconstruction of original stream
  - GUI for user control
- Examples:
  - Windows Media Player
  - QuickTime
  - HTML 5 Player

12

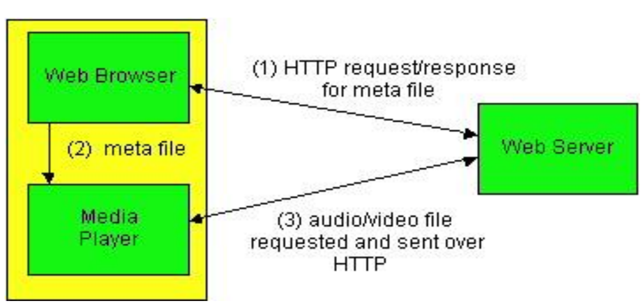
Streaming From Web Servers

- Data stored in a file
  - Audio: an audio file
  - Video: interleaving of audio and images in a file
- HTTP request-response
  - TCP connection between client and server
  - Client HTTP request and server HTTP response
- Client invokes the media player
  - Content-type indicates encoding
  - Browser launches media player
  - Media player renders file



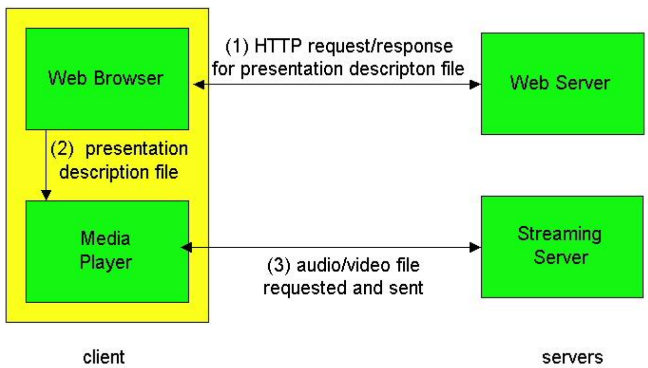
Initiating Streams from Web Servers

- Avoid passing all data through the Web browser
  - Web server returns a meta file describing the object
  - Browser launches media player and passes meta file
  - Player sets up its own connection to the Web server

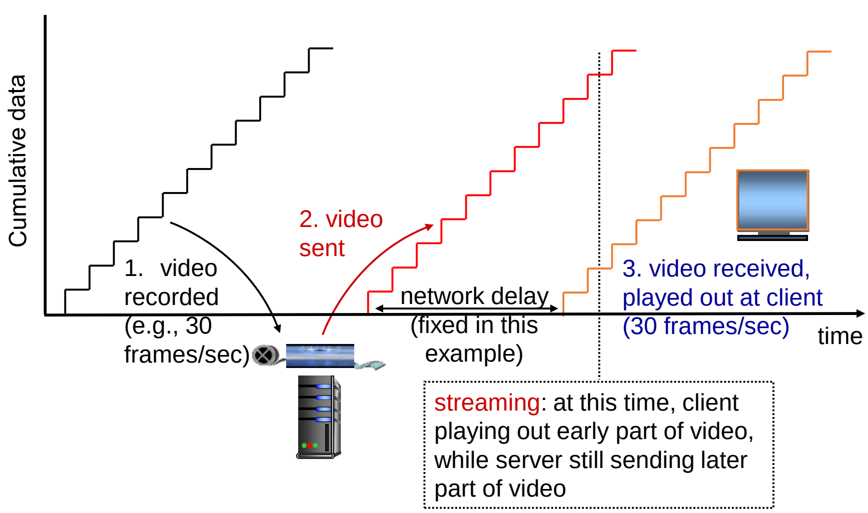


Using a Streaming Server

- Avoiding the use of HTTP (and perhaps TCP, too)
  - Web server returns a meta file describing the object
  - Player requests the data using a different protocol



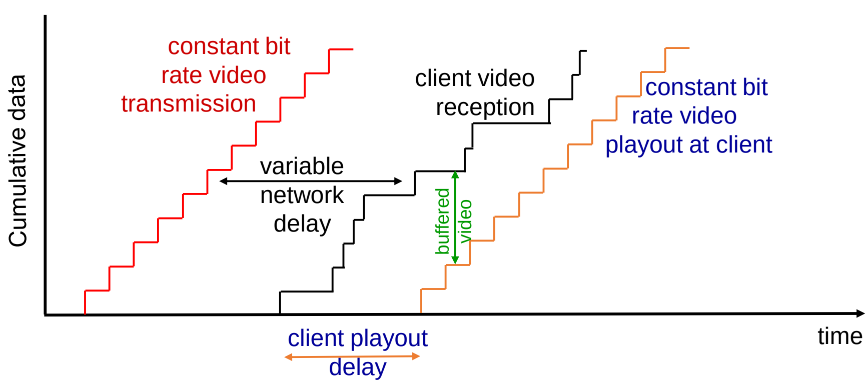
Streaming stored video:



Streaming stored video: challenges

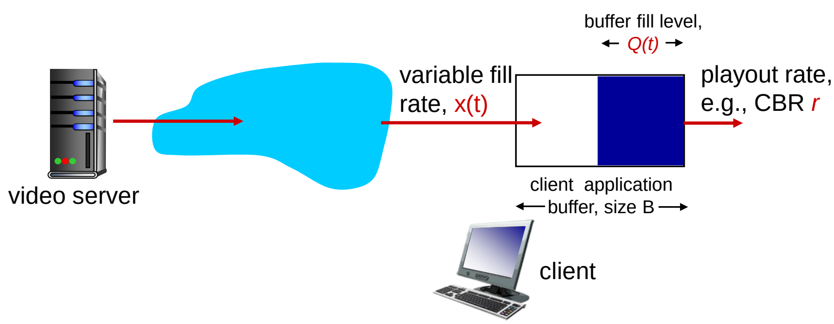
- continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted

Streaming stored video: revisited

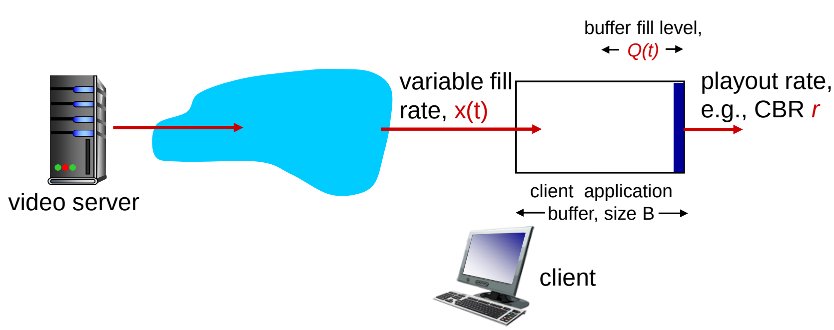


- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Client-side buffering, playout

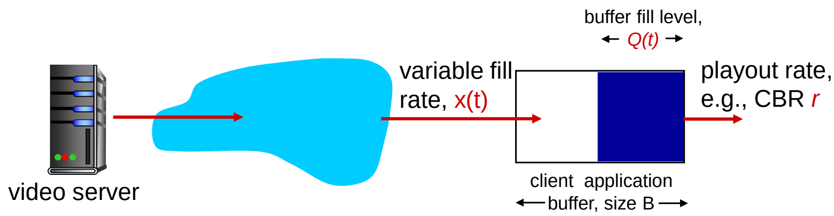


Client-side buffering, playout



1. Initial fill of buffer until playout begins at t_p
2. playout begins at $t_{p,}$
3. buffer fill level varies over time as fill rate x(t) varies and playout rate r is constant

Client-side buffering, playout



playout buffering: average fill rate ($\bar{x}$), playout rate (r):

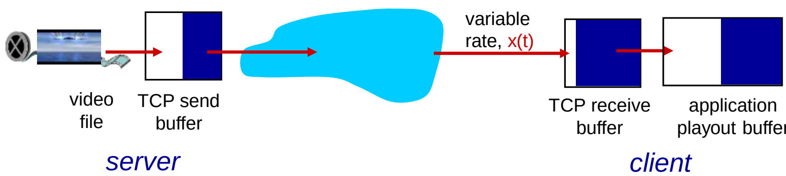
- $\bar{x} < r$: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *initial playout delay tradeoff:* buffer starvation less likely with larger delay, but larger delay until user begins watching

Streaming multimedia: UDP

- server sends at rate appropriate for client
  - often: send rate = encoding rate = constant rate
  - transmission rate can be oblivious to congestion levels
- short playout delay (2-5 seconds) to remove network jitter
- error recovery: application-level, time permitting
- RTP [RFC 2326]: multimedia payload types
- UDP may *not* go through firewalls

Streaming multimedia: HTTP

- multimedia file retrieved via HTTP GET
- send at maximum possible rate under TCP



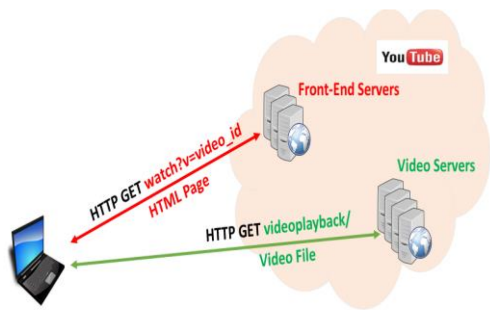
- fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Streaming multimedia: HTTP

- Can manage without a media control server (like RSTP), thus scalable
- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- However keep sending bits at maximum possible rate that TCP allows
- A form of prefetching from client perspective to handle jitter.
- Additionally using a larger playout delay to smooth TCP delivery rate
- Early Termination and Repositioning of the video (interactivity)
- HTTP byte range header (HTTP get message)
  - Server forgets about earlier request and start sending bytes from the point specified in the header.

HTTP Progressive Download

- With helper application doing the download, playback can start immediately...
- Or after sufficient bytes are buffered
- Sender sends at maximum possible rate under TCP; retransmit when error is encountered; Player uses a much larger buffer to smooth delivery rate of TCP



HTTP Progressive Download (2)

- HTTP connection keeps data flowing as fast as possible to user's local buffer
- May download lots of extra data if you do not watch the video
- TCP file transfer can use more bandwidth than necessary
- Mismatch between whole file transfer and stop/start/seek playback controls.
  - However: use file range requests to seek to video position

HTTP Adaptive Streaming (HAS)

- Other terms for similar concepts: Adaptive Streaming, Smooth Streaming, HTTP Chunking
- Actually a series of small progressive downloads of chunks
- No standard protocol. Typically HTTP to download series of small files.
  - Apple HLS, Microsoft IIS Smooth Streaming (Silverlight), Adobe Flash Dynamic Streaming, DASH: Dynamic Adaptive Streaming over HTTP
- Chunks are independent of each other (created at encoding time)
- Playing chunks in sequence gives seamless video
- Hybrid of streaming and progressive download:
  - Stream-like: sequence of small chunks requested/delivered as needed
  - Progressive download-like: HTTP transfer mechanism, stateless servers

27

Adaptive Streaming Concept

- Adaptive Streaming technologies enable
  - Optimal streaming video viewing experience for diverse range of devices over broad set of connection speeds
  - E.g., DASH, HLS
- Adaptive streaming technologies share
  - **Production of multiple files** from the same source file to distribute to viewers watching on different powered devices via different connection speeds
  - **Distribution of files adaptively**, changing stream that is delivered to adapt to changes in effective throughput and available CPU cycles on playback stations
  - **Transparent operation** to the user so that the viewer clicks one button and all streams switch/adapt behind the scenes.

Adaptive Streaming

First Approach of Adaptive Streaming (MPEG-based)
- Server sends first the high important video information (e.g., I frames)
- Then, lower importance video information follows (e.g., P and B frames) if bandwidth and time allows

Second Approach of Adaptive Streaming (HLS)
- Server sends with high quality part of the frame and only progressively ,if

bandwidth and time allow, it sends the rest of the frame information

Third Approach of Adaptive Streaming (DASH)
1. At server video is encoded in multiple bitrates and depending on the device bandwidth, it adjusts at what rate it requests chunks

HAS – Pros and Cons

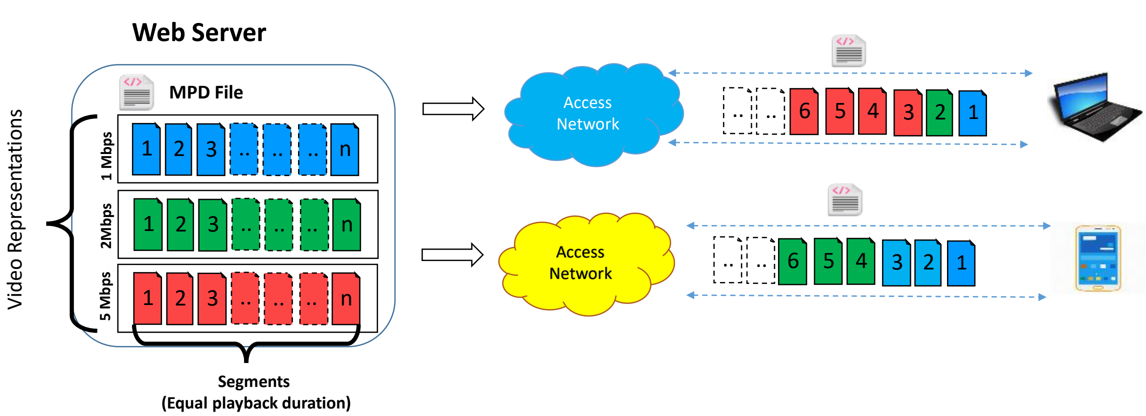
- Adaptation
  - Encode video at different levels of quality/bandwidth
  - Client can adapt by requesting different sized chunks
  - Chunks of different bit rates must be synchronized: All encodings have the same chunk boundaries, so you can make smooth transition to higher or lower bit rates
- Pros and Cons
  - + Easy to deploy: it's just HTTP, caches/proxies/CDN all work
  - + Fast startup by downloading lowest quality/smallest chunk
  - + Bitrate switching is seamless
  - - Many small files
- Chunks can be
  - Independent files -- many files to manage for one movie
  - Stored in single file container -- client or server must be able to access chunks, e.g. using range requests from client.

30

DASH – Dynamic Adaptive Streaming over HTTP

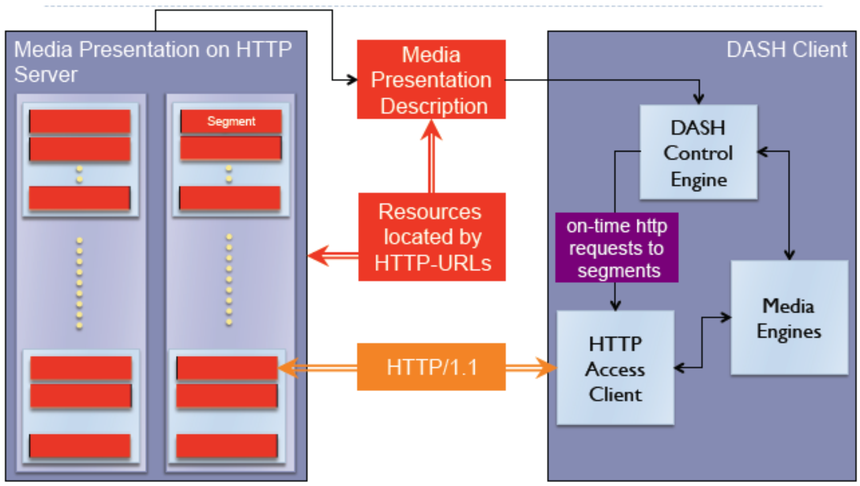
- What is DASH
  - Enabler which **provides formats** to enable efficient and high-quality delivery of streaming services over the Internet
  - Enabler to **reuse existing technologies** (containers, DRM (Digital Rights Management), codecs)
  - Enabler for deployment on top of **HTTP-CDNs**
  - Enabler for **very high user experience** (low start-up, no re-buffering)
  - Provides **simple inter-operability** points (profiles)

Dash Overview



32

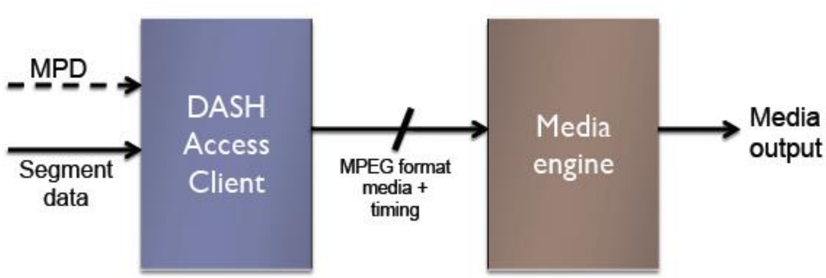
DASH Client



Thomas Stockhammer, Qualcomm, "DASH – Design Principles and Standards , Presentation at MMSys 2011

Information Classification

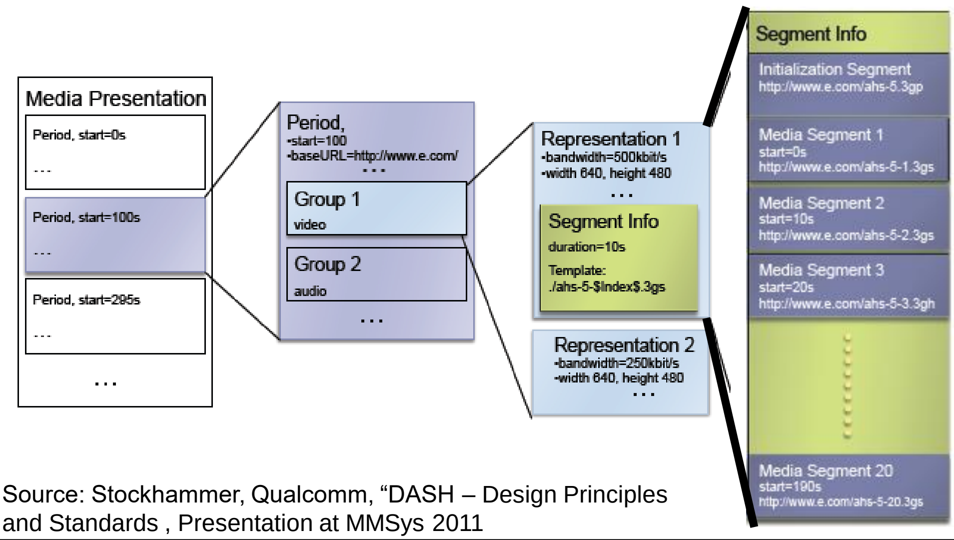
- DASH uses **MPD (Media Presentation Descriptor)** and **Index Information** as metadata for DASH Access Client
- Initialization and Media Segments for Media Engine
  - Reuse of existing container format



Source: MMSys'11

Media Presentation Data Model

MDP - description of accessible segments and corresponding timing



Source: Stockhammer, Qualcomm, “DASH – Design Principles and Standards , Presentation at MMSys 2011

MPD Information

- Includes redundant information of media streams to initially select or reject groups or representations
- Includes access and timing information
  - Content addressing via HTTP-URLs
  - Byte range for each accessible segment
  - Segment availability start and end time in wall-clock time
  - Approximate media start time and duration
  - Instructions on starting playout (for live service)
- Includes switching relations across representations

Segment Indexing

- Provides information in ISO box structure on
  - Accessible units of data (e.g., frames) in media segment
  - Byte range in segments (easy access through HTTP GET)
  - Accurate presentation duration (seamless switching)
  - Presence of representation access positions
- Provides compact bitrate-over-time to client
  - Can be used for intelligent request schedule
- Generic data structure
- Hierarchical structuring for efficient access

Quality of Experience (QoE)

- Quality-of-Service (QoS): Traditional metrics
  - QoS metrics: packet loss, delay, jitter
  - QoS metrics are not understood by the users
- Quality-of-Experience (QoE): User-centric meric
  - Defined by ITU as *"Metric that captures the overall acceptability of the service and included end-to-end factors"*
- Types of QoE Metrics for online video streaming services
  - Subjective Metrics
    - Mean Opinion Score (MOS)
  - Objective Metrics
    - Playback Start Time
    - Interruptions in playback (frequency & duration)
    - User Engagement
    - Video Quality
- Influence Factors
  - Device: screen size, resolution, memory, battery, etc.
  - Content: genre, length, quality etc.
  - Human: emotion, context, intent, socio-psychological conditions

38