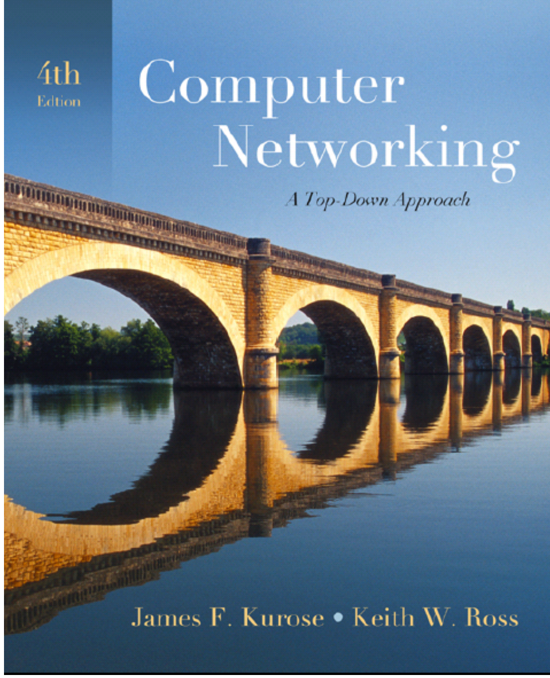
Multimedia Networking and Streaming Protocols

Prof. Wing C. Lau

Acknowledgements

Many of the slides in this lecture are based on:

- Chapter 7 of the Kurose/Ross (L.H.S)
- Greg Thompson (Cisco),"IPTV", a talk to SMPTE, Jan 2008
- Mark Handley, Lecture Notes on Audio and Video Technologies
- Aljoscha Smolic, "The emerging H.264/AVC Video Coding Standard".
- Dinesh Kumar et al, "Overview of the H.264/AVC".
- Lecture Notes from Prof. Jack Y.B.Lee



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

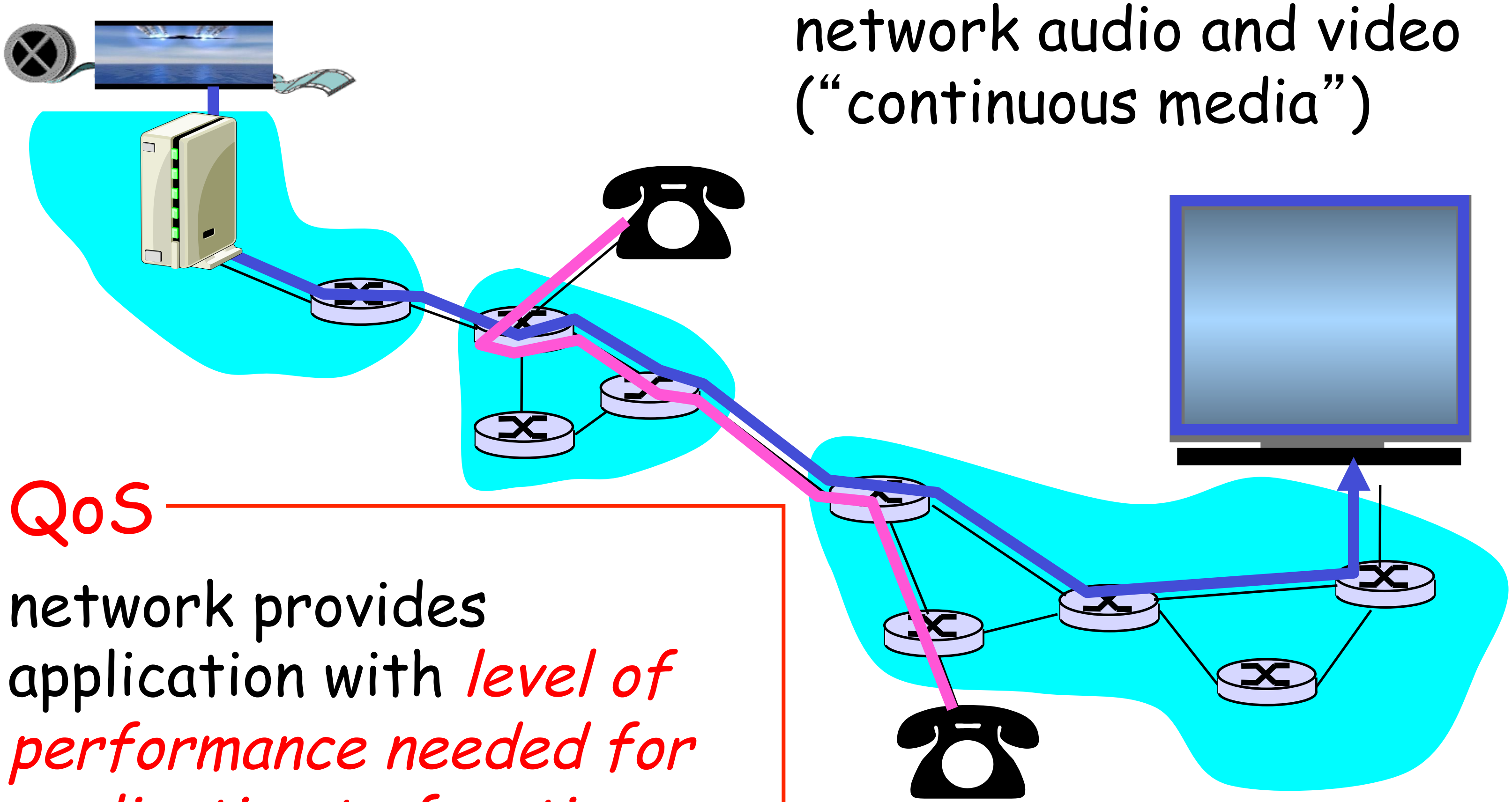
- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK / KWR

Outline

- Multimedia networking applications
- A digression to Audio and Video compression standards
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
  - RTP, RTCP, MPEG-DASH, and more…
- An Example:
  - Commercial IPTV service Architecture

Multimedia and Quality of Service: What is it?



multimedia applications: network audio and video ("continuous media")

QoS

network provides application with *level of performance needed for application to function.*

MM Networking Applications

Classes of MM applications:

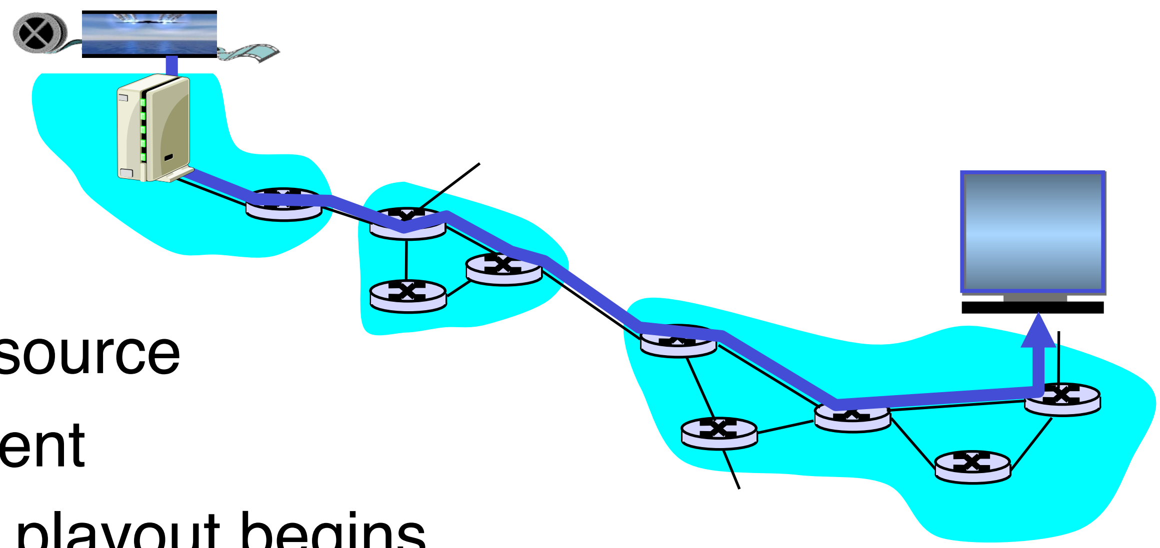
1) stored streaming
2) live streaming
3) interactive, real-time

Jitter is the variability of packet delays within the same packet stream

Fundamental characteristics:

- typically **delay sensitive**
  - end-to-end delay
  - delay jitter
- **loss tolerant**: infrequent losses cause minor glitches
- antithesis of data, which are loss *intolerant* but delay *tolerant*.

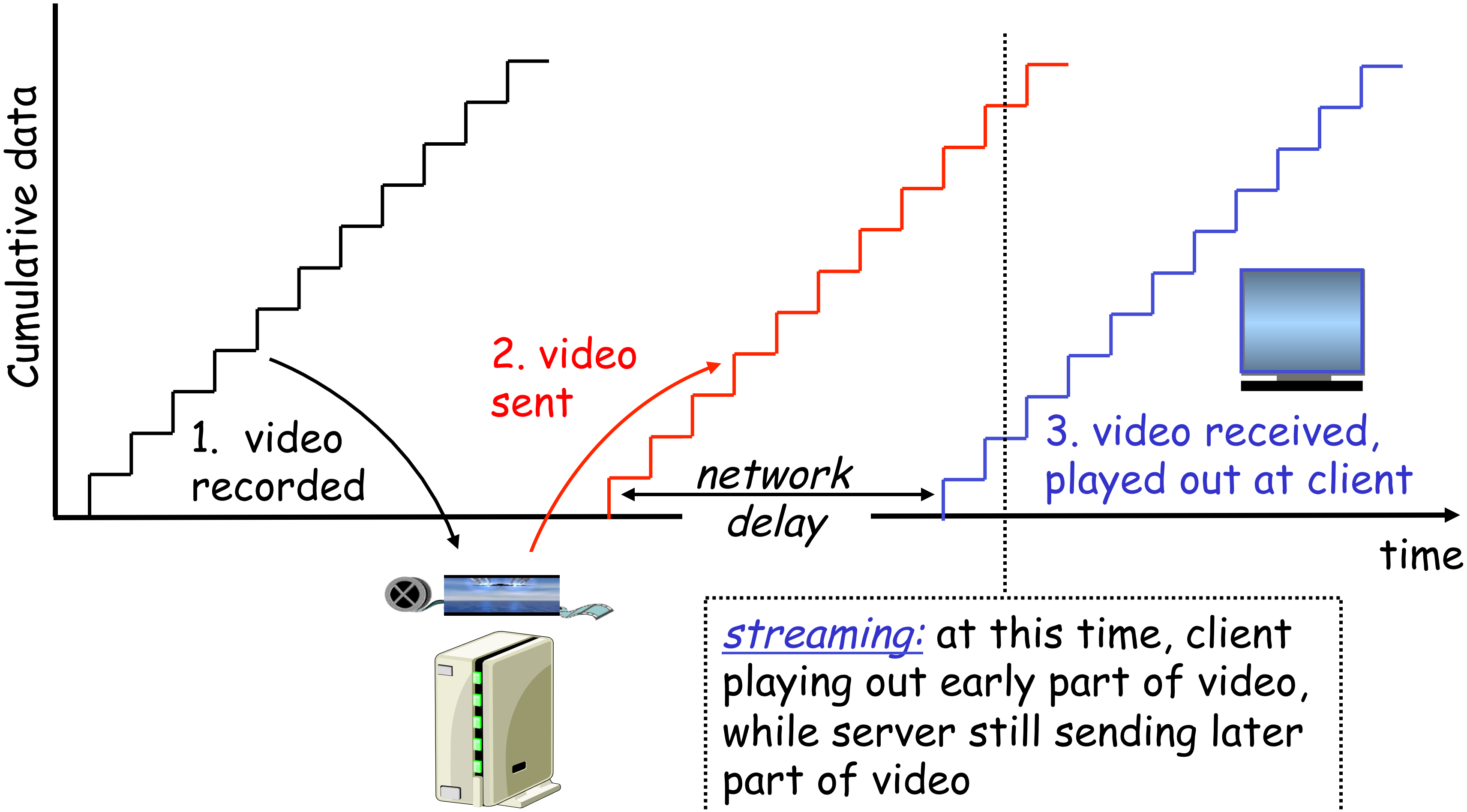
Streaming Stored Multimedia



Stored streaming:

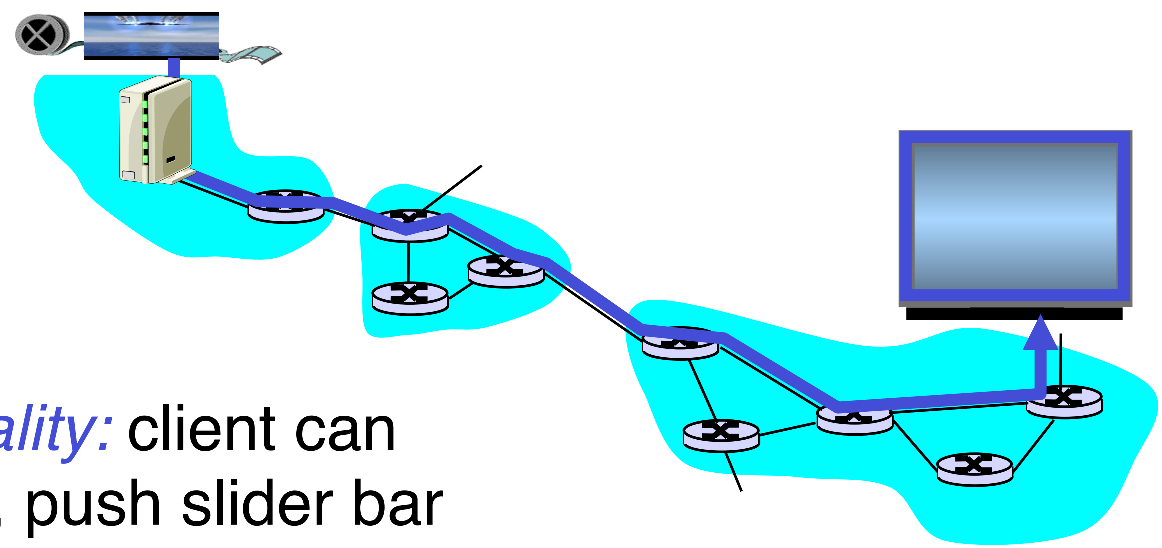
- media stored at source
- transmitted to client
- *streaming:* client playout begins *before* all data has arrived
  - timing constraint for still-to-be transmitted data: in time for playout

Streaming Stored Multimedia: What is it?



streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming *Stored* Multimedia: Interactivity



- *VCR-like functionality:* client can pause, rewind, FF, push slider bar
  - 10 sec initial delay OK
  - 1-2 sec until command effect OK
- timing constraint for still-to-be transmitted data: in time for playout

Streaming *Live* Multimedia

Examples:

- Internet radio talk show
- live sporting event

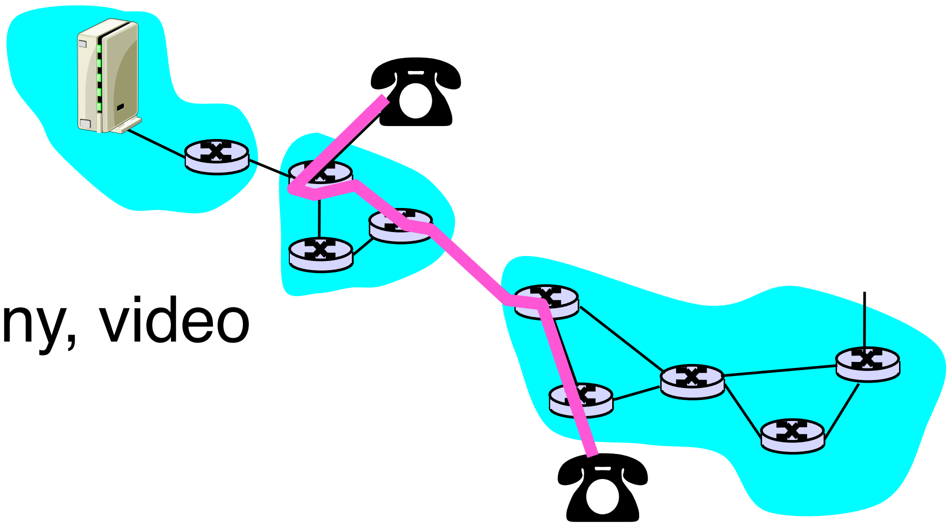
Streaming (as with streaming *stored* multimedia)

- playback buffer
- playback can lag tens of seconds after transmission
- still have timing constraint

Interactivity

- fast forward impossible
- rewind, pause possible!

Real-Time Interactive Multimedia

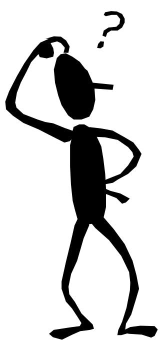


- applications: IP telephony, video conference, distributed interactive worlds
- end-end delay requirements:
  - audio: < 150 msec good, < 400 msec OK
    - includes application-level (packetization) and network delays
    - higher delays noticeable, impair interactivity
- session initialization
  - how does callee advertise its IP address, port number, encoding algorithms?
    - To be addressed by SIP (later in the course)

Multimedia Over Today's Internet

TCP/UDP/IP: "best-effort service"

- *no* guarantees on delay, loss



But you said multimedia apps requires QoS and level of performance to be effective!

Today's Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

How should the Internet evolve to better support multimedia?

Integrated services philosophy: (The IETF Intserv model)

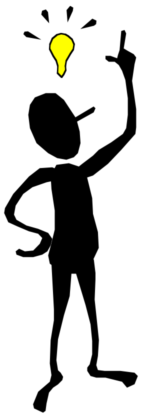
- fundamental changes in Internet so that apps can reserve end-to-end bandwidth
- requires new, complex software in hosts & routers

Laissez-faire

- no major changes
- more bandwidth when needed
- content distribution, application-layer multicast
  - application layer

Differentiated services philosophy: (The IETF Diffserv model)

- fewer changes to Internet infrastructure, yet provide 1st and 2nd class service



What's your opinion?

Outline

- Multimedia networking applications
- A digression to Audio and Video compression standards
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
  - RTP, RTCP, MPEG-DASH, and more…

A few words about audio compression

- analog 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
  - 8 bits for 256 values

- 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 (MPEG-1 Audio Layer 3) : 96, 128, 160 kbps
- Internet telephony: 5.3 kbps and up

http://en.wikipedia.org/wiki/Adaptive_Multi-Rate

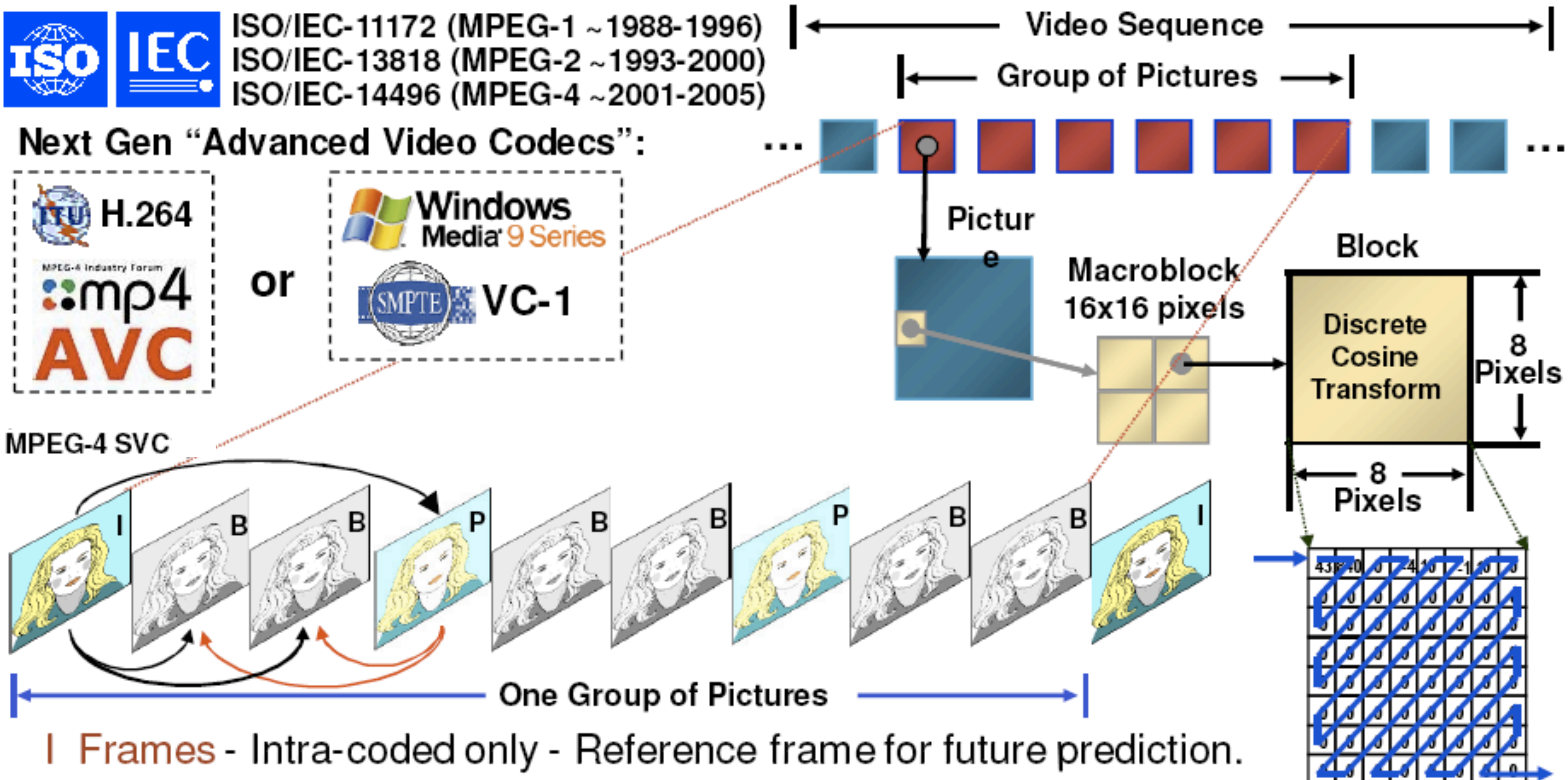
A few words about video compression

- video: sequence of images displayed at constant rate
  - e.g. 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- redundancy
  - spatial (within image)
  - temporal (from one image to next)

Examples:

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)
- H.264/AVC (50kbps - 8+ Mbps)
- H.264/MPEG4 SVC (Scalable Video Coding
  - layered (scalable) video, adapt layers to
    - Available bandwidth (data rate)
    - Screen resolution/size
    - Frame rate

MPEG Video Summary



I Frames - Intra-coded only - Reference frame for future prediction.
P Frames - Forward prediction from either previous I or P frames.
B Frames - Bi-directional interpolated prediction from two sources.

DCT Co-efficients
Zig-Zag extraction

Digital Video Bandwidth Requirements

Uncompressed Digital Video	
SDTV (480i CCIR 601 over SD-SDI SMPTE 259M)	165.9 – 270 Mbps
EDTV (480p or 576p via SMPTE 344M)	540 Mbps
HDTV (1080i or 720p over HD-SDI SMPTE 292M)	1.485 Gbps
HDTV (1080p over Dual link HD-SDI SMPTE 372M)	2.970 Gbps
MPEG-2 Compressed Video	
SDTV Broadcast (3.75 Mbps for cable VOD)	3 – 6 Mbps
HDTV Broadcast (19.3 Mbps for ATSC DTV)	12 – 20 Mbps
SDTV Production (Contribution – 4:2:2 I-frame only)	18 – 50 Mbps
HDTV Production (Contribution – 4:4:4 I-frame 10-bit)	140 – 500 Mbps
MPEG-4 AVC / H.264 Compressed Video	
SDTV Broadcast (about 50% less than MPEG-2)	1.5 – 3 Mbps
HDTV Broadcast (1080i about 4x SDTV)	6 – 9 Mbps

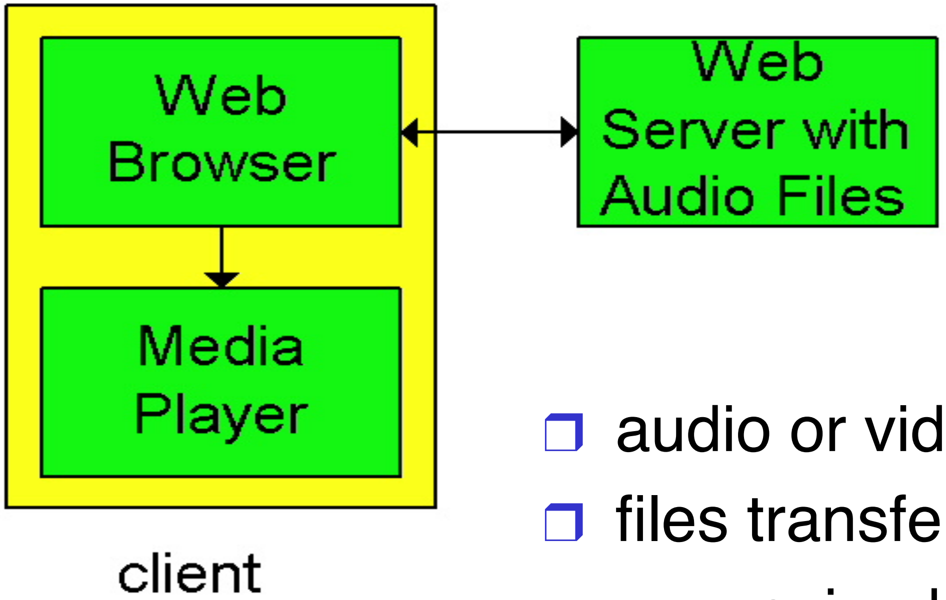
Enable a 200:1 compression ratio for HDTV

Source: Cisco

Outline

- Multimedia networking applications
- A digression to Audio and Video compression standards
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
  - RTP, RTCP
- Providing multiple classes of service
  - Traffic Characterization and Shaping
  - Scheduling
  - The IETF Diffserv model
- Providing QoS guarantees
  - The IETF Intserv model
  - RSVP

Internet multimedia: simplest approach

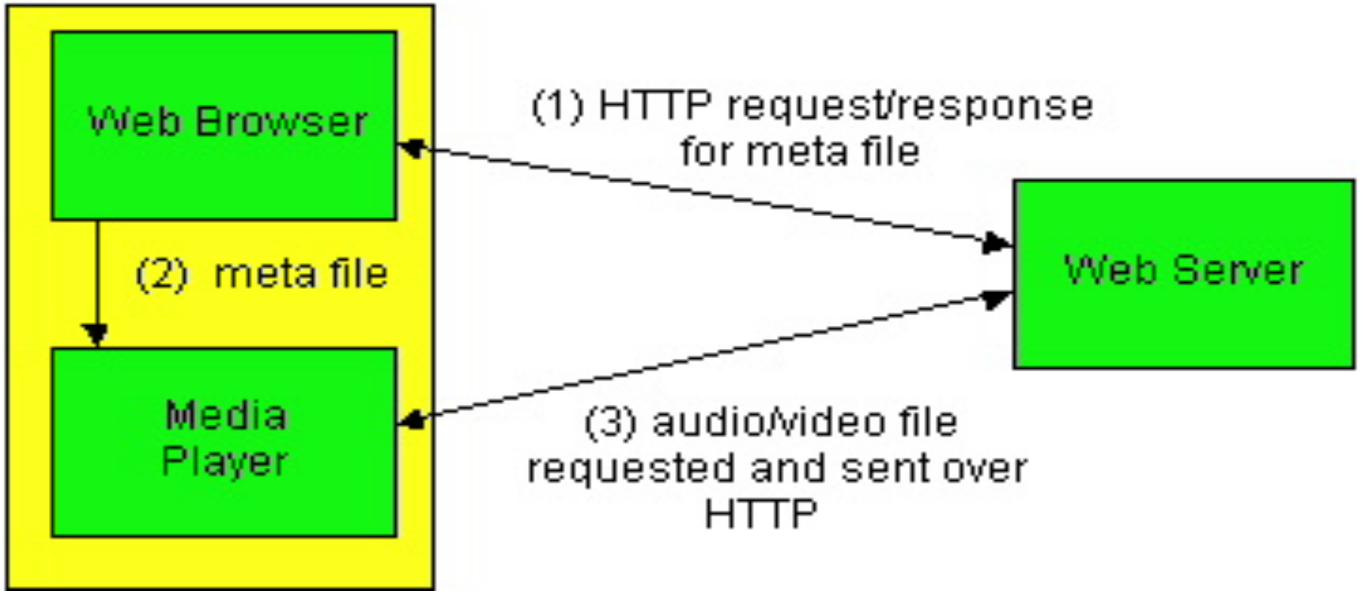


- audio or video stored in file
- files transferred as HTTP object
  - received in entirety at client
  - then passed to player

audio, video not streamed:

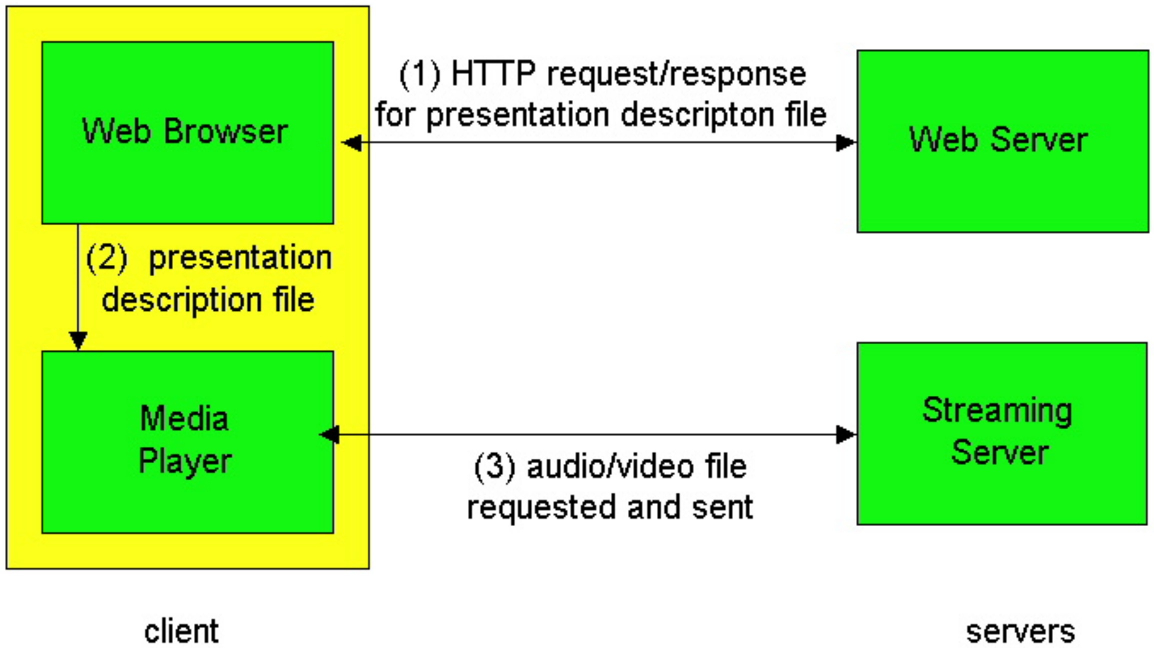
- no, "pipelining," long delays until playout!

Internet multimedia: streaming approach



- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server streams audio/video to player

Streaming from a streaming server



- allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3), more shortly

Streaming Stored Multimedia

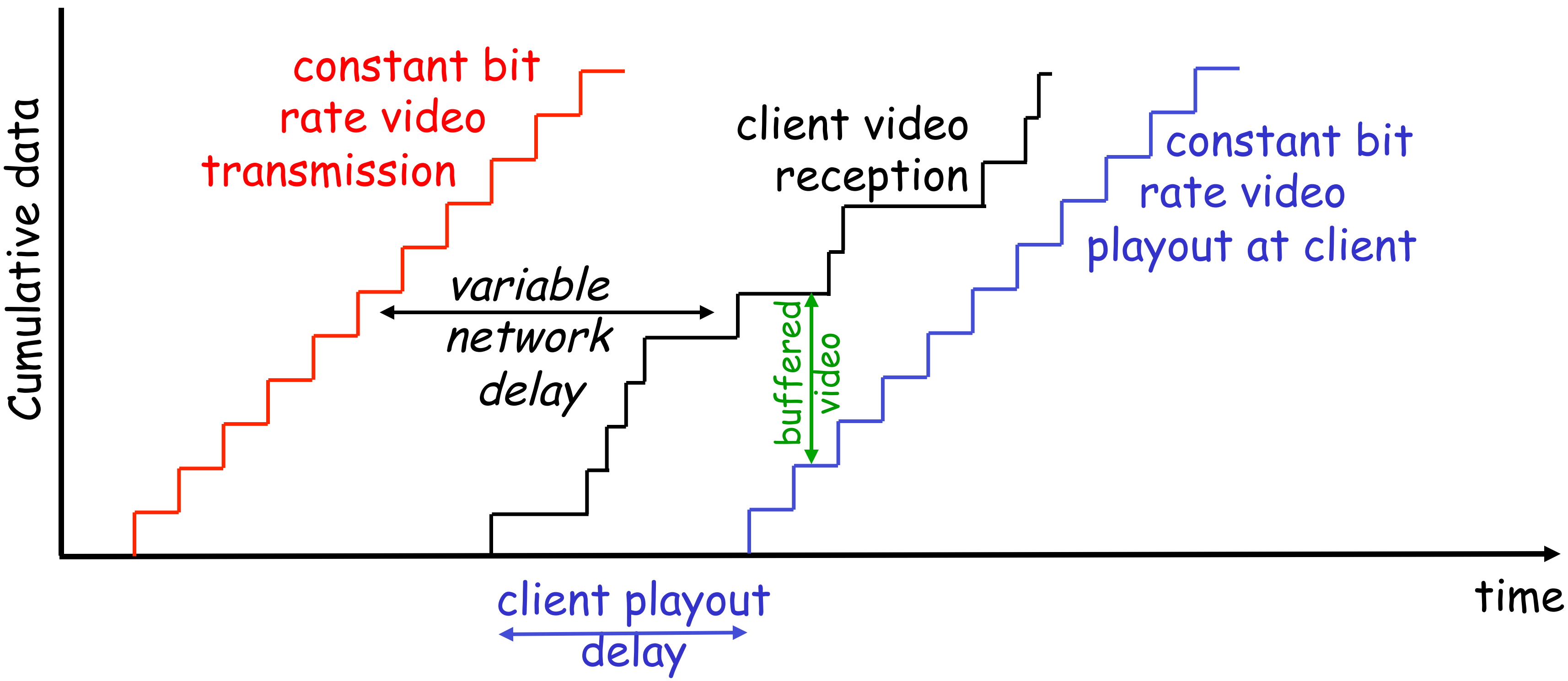
application-level streaming techniques for making the best out of best effort service:

- client-side buffering
- use of UDP versus TCP
- multiple encodings of multimedia

Media Player

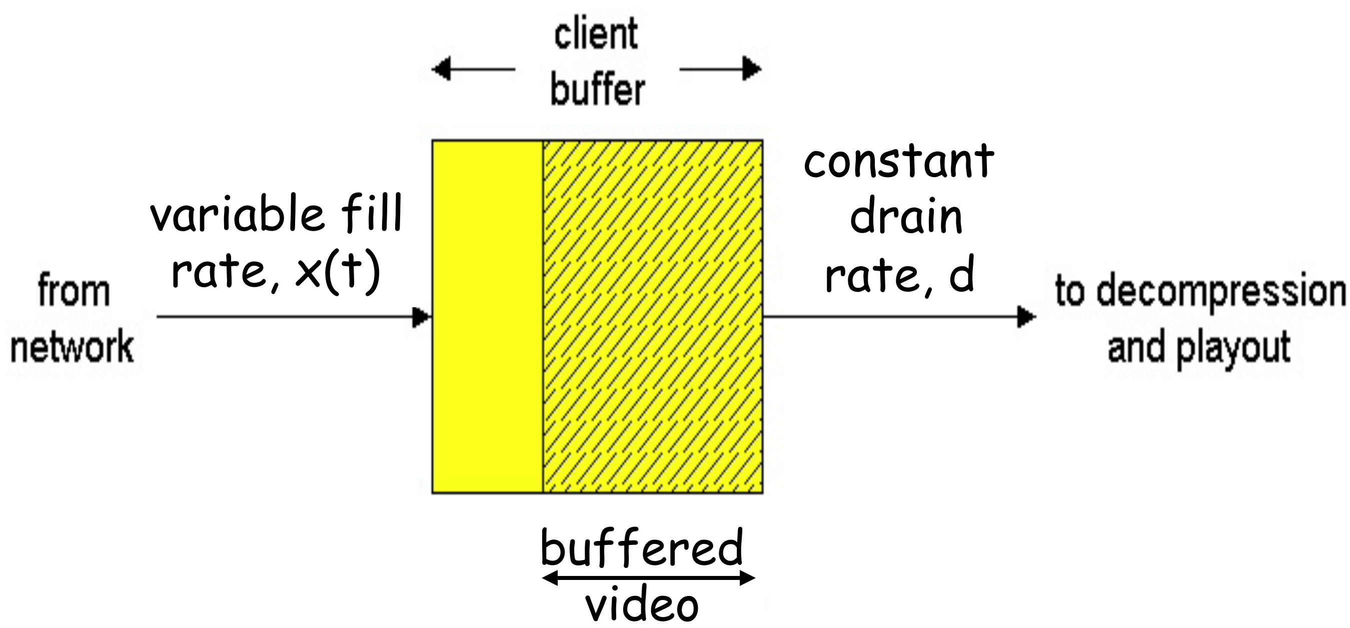
- jitter removal
- decompression
- error concealment
- graphical user interface w/ controls for interactivity

Streaming Multimedia: Client Buffering



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

Streaming Multimedia: Client Buffering



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

Streaming Multimedia: UDP or TCP?

UDP

- server sends at rate appropriate for client (oblivious to network congestion !)
  - often send rate = encoding rate = constant rate
  - then, fill rate = constant rate - packet loss
- short playout delay (2-5 seconds) to remove network jitter
- error recover: time permitting

TCP

- send at maximum possible rate under TCP
- fill rate fluctuates due to TCP congestion control
- larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

User Control of Streaming Media: RTSP

HTTP

- does not target multimedia content
- no commands for fast forward, etc.

RTSP: RFC 2326

- client-server application layer protocol
- user control: rewind, fast forward, pause, resume, repositioning, etc…

What it doesn't do:

- doesn't define how audio/video is encapsulated for streaming over network
- doesn't restrict how streamed media is transported (UDP or TCP possible)
- doesn't specify how media player buffers audio/video

RTSP: out of band control

FTP uses an "out-of-band" control channel:

- file transferred over one TCP connection.
- control info (directory changes, file deletion, rename) sent over separate TCP connection
- "out-of-band", "in-band" channels use different port numbers

RTSP messages also sent out-of-band:

- RTSP control messages use different port numbers than media stream: out-of-band.
  - port 554
- media stream is considered "in-band".

RTSP Example

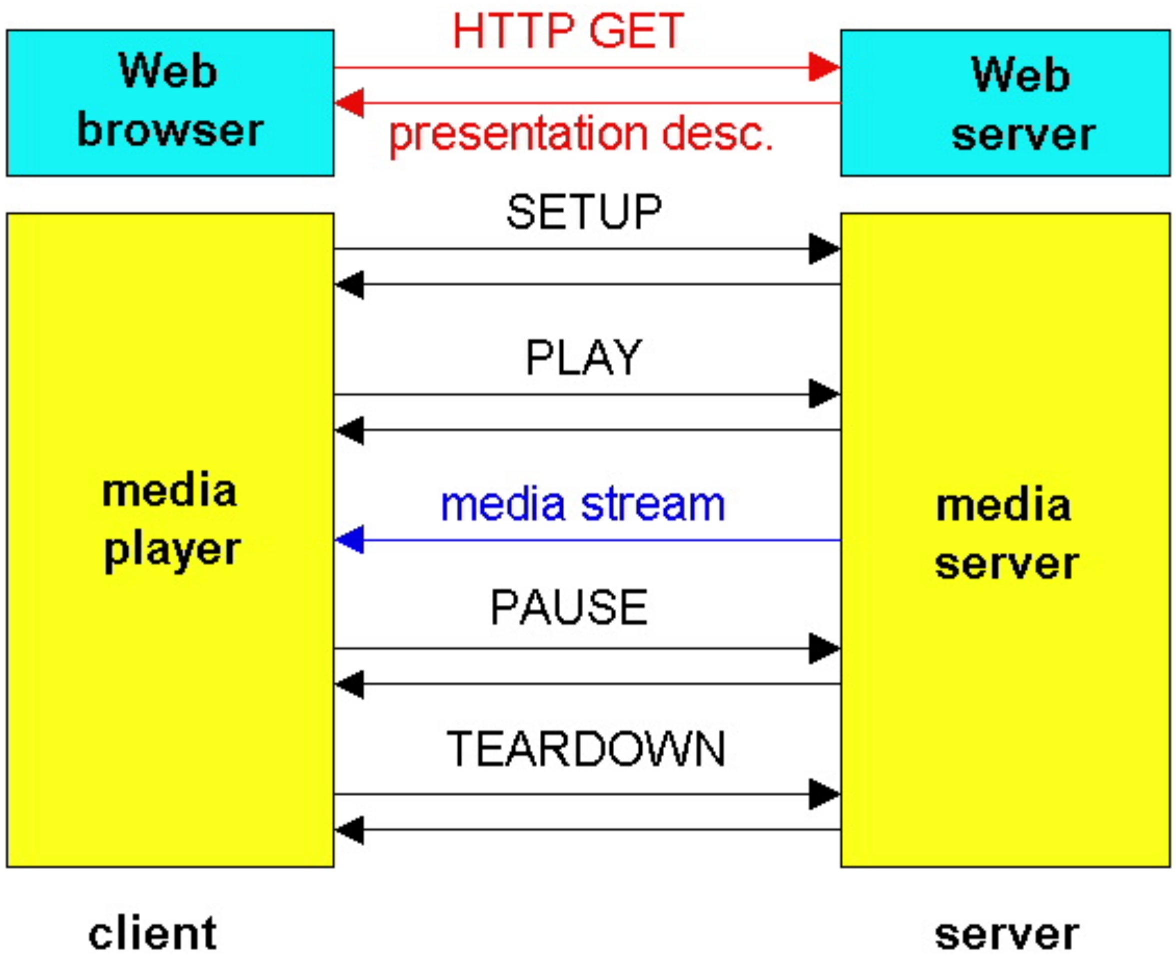
Scenario:

- metafile communicated to web browser
- browser launches player
- player sets up an RTSP control connection, data connection to streaming server

Metafile Example

```
<title>Twister</title>
<session>
        <group language=en lipsync>
                <switch>
                    <track type=audio
                        e="PCMU/8000/1"
                        src = "rtsp://audio.example.com/twister/audio.en/lofi">
                    <track type=audio
                        e="DVI4/16000/2" pt="90 DVI4/8000/1"
                        src="rtsp://audio.example.com/twister/audio.en/hifi">
                 </switch>
              <track type="video/jpeg"
                        src="rtsp://video.example.com/twister/video">
         </group>
</session>
```

RTSP Operation



RTSP Exchange Example

C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK
Session 4231

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=0-

C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=37

C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231

S: 200 3 OK

Outline

- Multimedia networking applications
- A digression to Audio and Video compression standards
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
  - RTP, RTCP, MPEG-DASH, and more…

Interactive Multimedia: Internet Phone

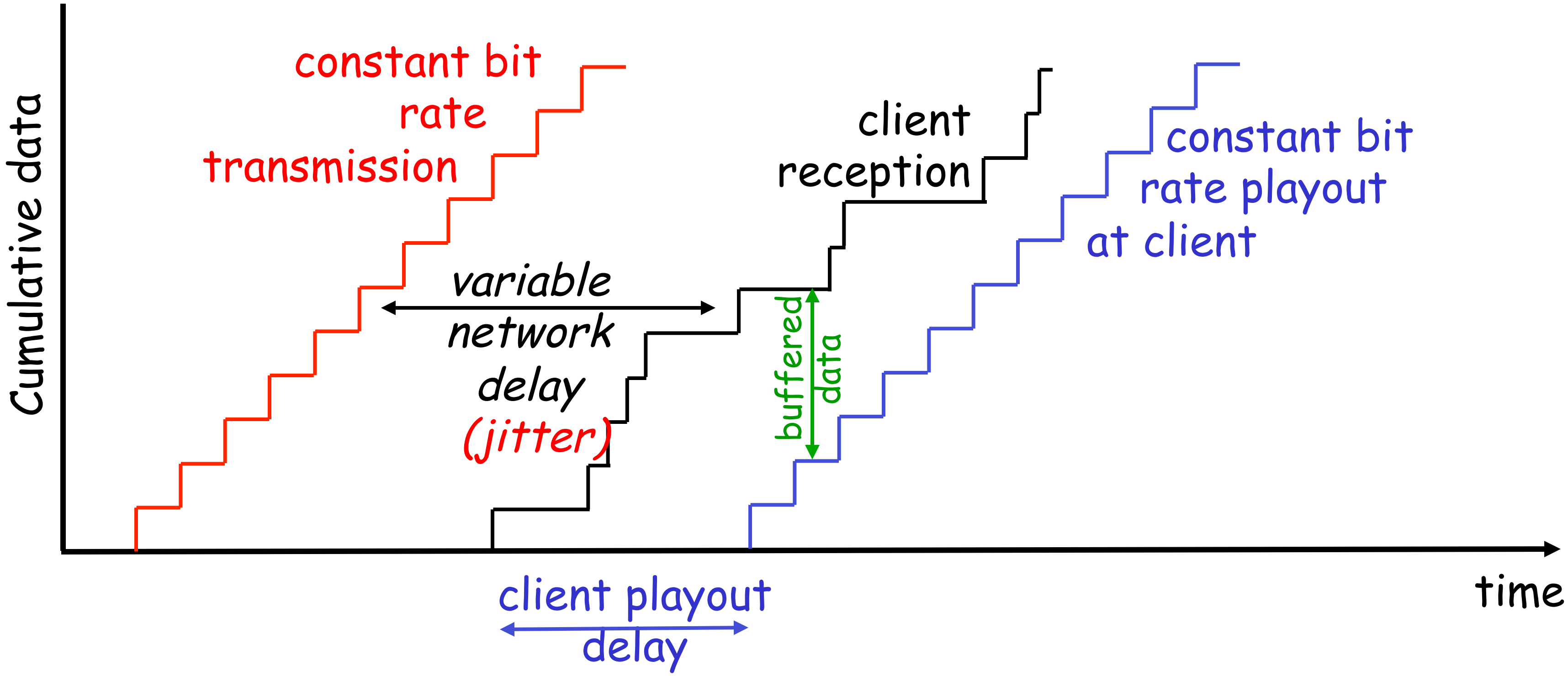
Introduce Internet Phone by way of an example

- speaker's audio: alternating talk spurts, silent periods.
  - 64 kbps during talk spurt
  - pkts generated only during talk spurts
  - 20 msec chunks at 8 Kbytes/sec: 160 bytes data
- application-layer header added to each chunk.
- chunk+header encapsulated into UDP segment.
- application sends UDP segment into socket every 20 msec during talkspurt

Internet Phone: Packet Loss and Delay

- network loss: IP datagram lost due to network congestion (router buffer overflow)
- delay loss: IP datagram arrives too late for playout at receiver
  - delays: processing, queueing in network; end-system (sender, receiver) delays
  - typical maximum tolerable delay: 400 ms
- loss tolerance: depending on voice encoding, losses concealed, packet loss rates between 1% and 10% can be tolerated.

Delay Jitter



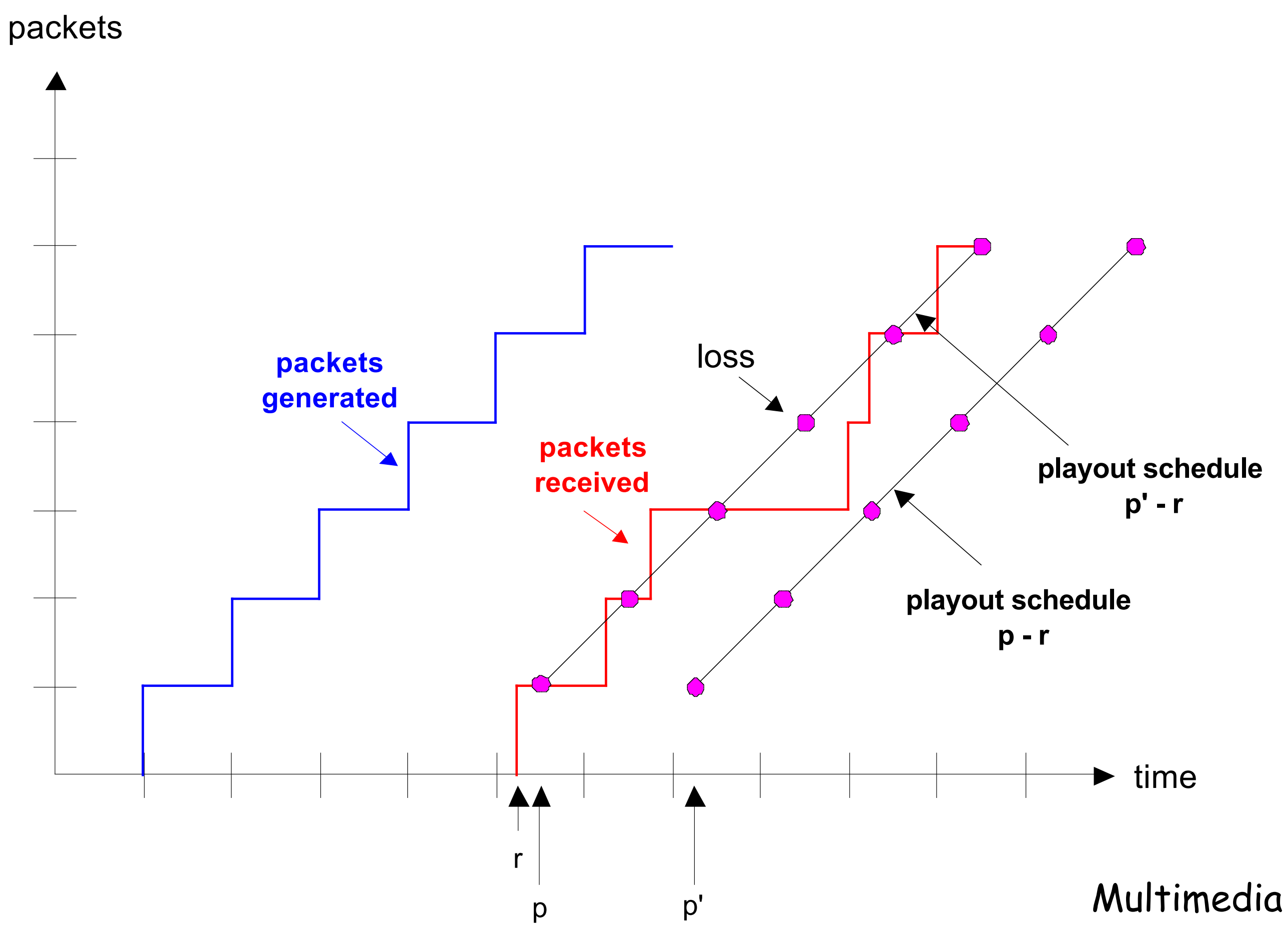
- consider end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

Internet Phone: Fixed Playout Delay

- receiver attempts to playout each chunk exactly q msecs after chunk was generated.
  - chunk has time stamp t: play out chunk at t+q .
  - chunk arrives after t+q: data arrives too late for playout, data "lost"
- tradeoff in choosing q:
  - *large q:* less packet loss
  - *small q:* better interactive experience

Fixed Playout Delay

- sender generates packets every 20 msec during talk spurt.
- first packet received at time r
- first playout schedule: begins at p
- second playout schedule: begins at p'



Recovery from packet loss (1)

Forward Error Correction (FEC): simple scheme

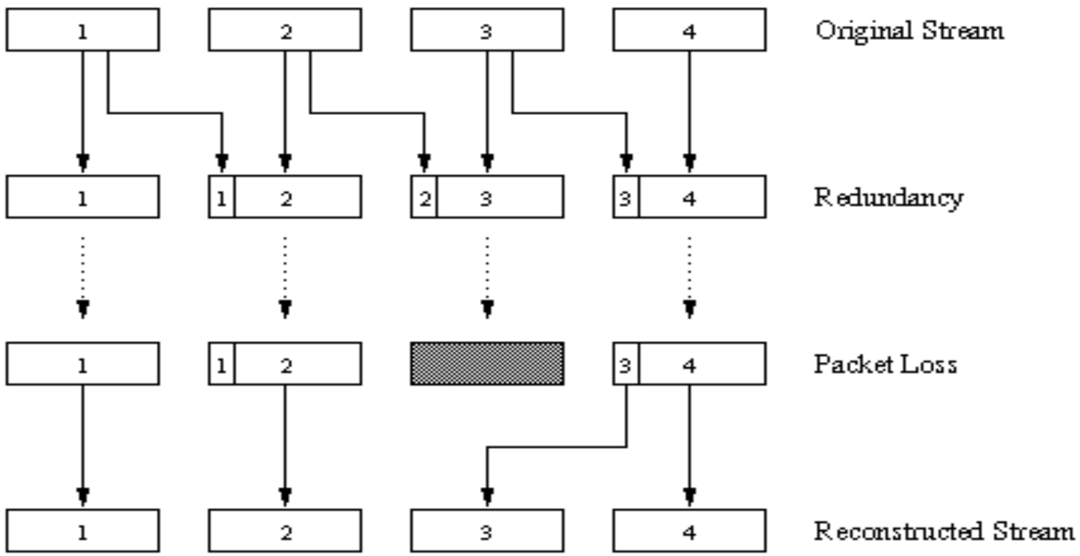
- for every group of *n* chunks create redundant chunk by exclusive OR-ing *n* original chunks
- send out *n+1* chunks, increasing bandwidth by factor *1/n*.
- can reconstruct original *n* chunks if at most one lost chunk from *n+1* chunks

- playout delay: enough time to receive all n+1 packets
- tradeoff:
    - increase n, less bandwidth waste
    - increase n, longer playout delay
    - increase n, higher probability that 2 or more chunks will be lost

Recovery from packet loss (2)

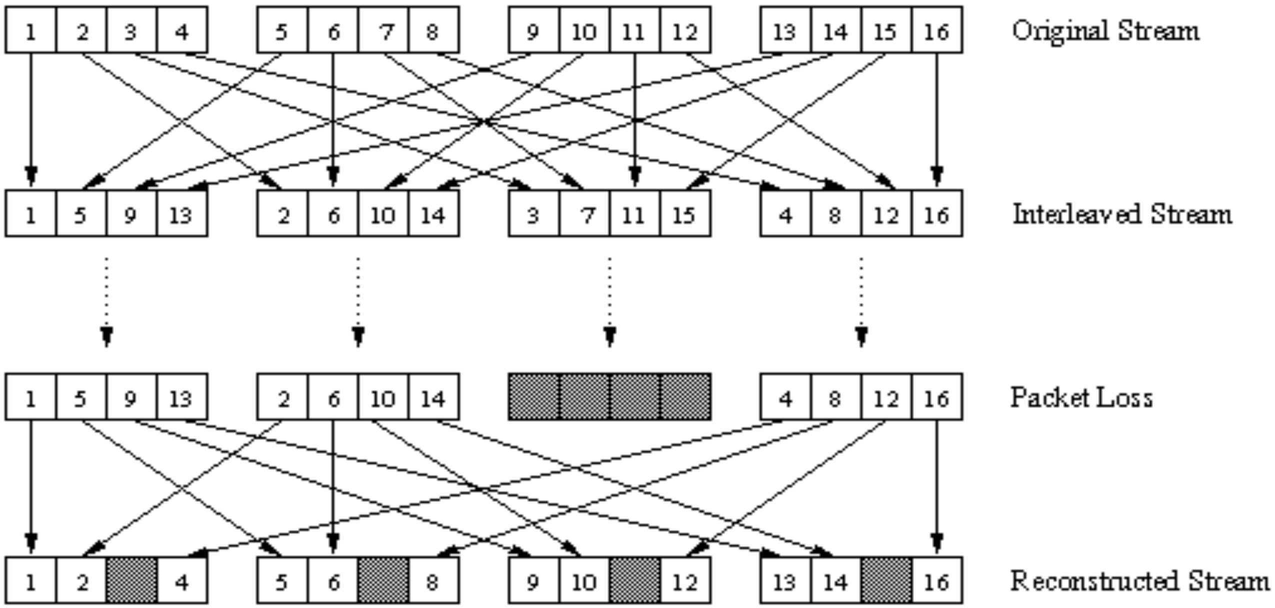
2nd FEC scheme

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



- whenever there is non-consecutive loss, receiver can conceal the loss.
- can also append (n-1)st and (n-2)nd low-bit rate chunk

Recovery from packet loss (3)



Interleaving

- chunks divided into smaller units
- for example, four 5 msec units per chunk
- packet contains small units from different chunks
- if packet lost, still have most of every chunk
- no redundancy overhead, but increases playout delay

Summary: Internet Multimedia: bag of tricks

- use UDP to avoid TCP congestion control (delays) for time-sensitive traffic
- client-side adaptive playout delay: to compensate for delay
- server side matches stream bandwidth to available client-to-server path bandwidth
  - chose among pre-encoded stream rates
  - dynamic server encoding rate
- error recovery (on top of UDP)
  - FEC, interleaving, error concealment
  - retransmissions, time permitting
- CDN: bring content closer to clients

Outline

- Multimedia networking applications
- A digression to Audio and Video compression standards
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
  - RTP, RTCP, MPEG-DASH, and more…

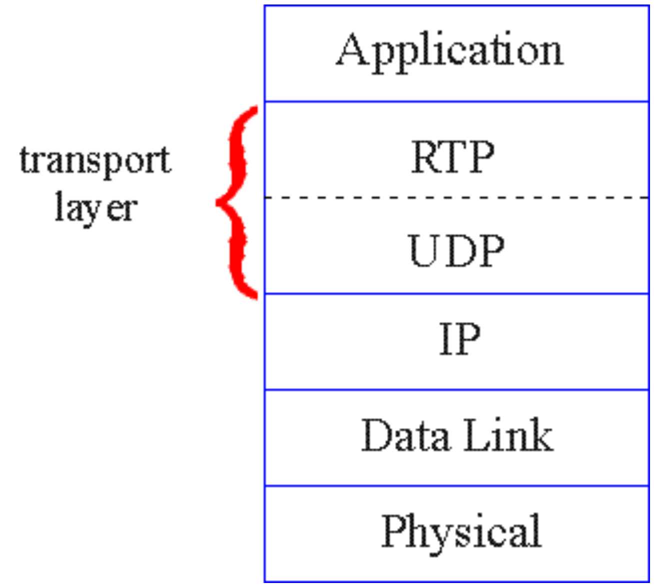
Real-Time Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two Internet phone applications run RTP, then they may be able to work together

RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

- port numbers, IP addresses
- payload type identification
- packet sequence numbering
- time-stamping



RTP Example

- consider sending 64 kbps PCM-encoded voice over RTP.
- application collects encoded data in chunks, e.g., every 20 msec = 160 bytes in a chunk.
- audio chunk + RTP header form RTP packet, which is encapsulated in UDP segment
- RTP header indicates type of audio encoding in each packet
  - sender can change encoding during conference.
- RTP header also contains sequence numbers, timestamps.

RTP and QoS

- RTP does **not** provide any mechanism to ensure timely data delivery or other QoS guarantees.
- RTP encapsulation is only seen at end systems (not) by intermediate routers.
  - routers providing best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter.

RTP Header

RTP Header

Payload Type (7 bits): Indicates type of encoding currently being used. If sender changes encoding in middle of conference, sender informs receiver via payload type field.

- Payload type 0: PCM mu-law, 64 kbps
- Payload type 3, GSM, 13 kbps
- Payload type 7, LPC, 2.4 kbps
- Payload type 26, Motion JPEG
- Payload type 31. H.261
- Payload type 33, MPEG2 video

Sequence Number (16 bits): Increments by one for each RTP packet sent, and may be used to detect packet loss and to restore packet sequence.

RTP Header (2)

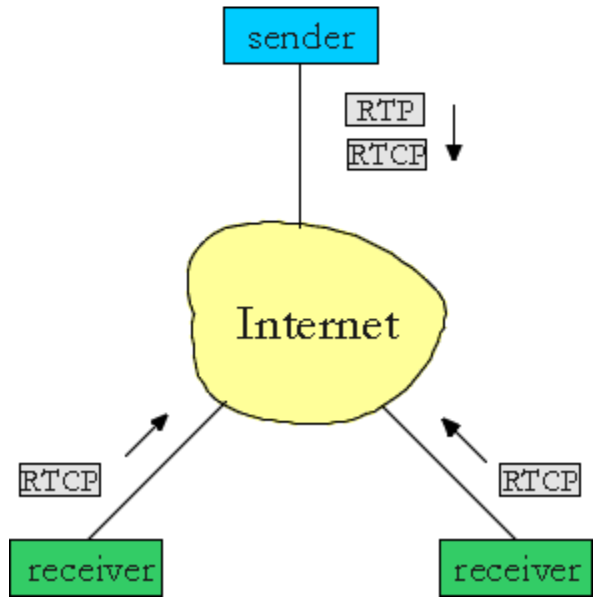
- *Timestamp field (32 bytes long):* sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock typically increments by one for each sampling period (for example, each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, then timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.

- *SSRC field (32 bits long):* identifies source of the RTP stream. Each stream in RTP session should have distinct SSRC.

Real-Time Control Protocol (RTCP)

- works in conjunction with RTP.
- each participant in RTP session periodically transmits RTCP control packets to all other participants.
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter, etc.
- feedback can be used to control performance
  - sender may modify its transmissions based on feedback

RTCP - Continued



- each RTP session: typically a single multicast address; all RTP /RTCP packets belonging to session use multicast address.
- RTP, RTCP packets distinguished from each other via distinct port numbers.
- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases

RTCP Packets

Receiver report packets:

- fraction of packets lost, last sequence number, average interarrival jitter

Sender report packets:

- SSRC of RTP stream, current time, number of packets sent, number of bytes sent

Source description packets:

- e-mail address of sender, sender's name, SSRC of associated RTP stream
- provide mapping between the SSRC and the user/host name

Synchronization of Streams

- RTCP can synchronize different media streams within a RTP session
- consider videoconferencing app for which each sender generates one RTP stream for video, one for audio.
- timestamps in RTP packets tied to the video, audio sampling clocks
  - *not* tied to wall-clock time
- each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created.
- receivers uses association to synchronize playout of audio, video

RTCP Bandwidth Scaling

- RTCP attempts to limit its traffic to 5% of session bandwidth.

Example

- Suppose one sender, sending video at 2 Mbps. Then RTCP attempts to limit its traffic to 100 Kbps.
- RTCP gives 75% of rate to receivers; remaining 25% to sender
- 75 kbps is equally shared among receivers:
  - with R receivers, each receiver gets to send RTCP traffic at 75/R kbps.
- sender gets to send RTCP traffic at 25 kbps.
- participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate

NAT/Firewall Compatibility of RTP/RTCP

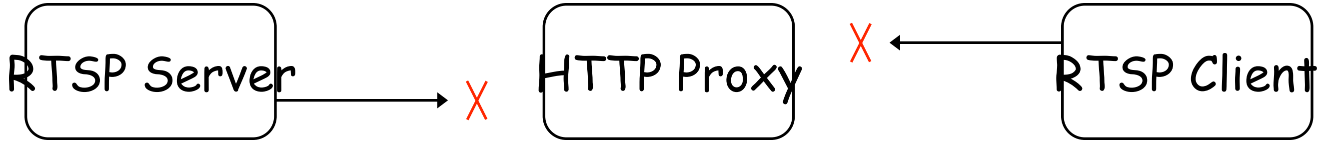
- RTP/RTCP introduces compatibility issues with firewalls and NATs
  - Firewall may block traffic delivered using non-well-known ports (i.e., port number < 1024)
  - Firewall may impose rate limitation on non-TCP traffic
- RTSP/RTP-aware NAT and Firewall
  - NAT can decode the RTSP messages to automatically add port mapping for the negotiated RTP streams.
  - Supported by Cisco's IOS Software via the NBAR feature (Network Based Application Recognition)
  - http://www.cisco.com/en/US/docs/ios/12_3t/12_3t7/feature/guide/gtnrtsp.html

RTP/RTCP over TCP

- Defined in J. Lazzaro, *Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport*, RFC4571, July 2006.
- Define a framing method for multiplexing and transporting RTP and RTCP packets over a connection-oriented transport such as TCP
- Defines a method for server and client to negotiate for RTP/AVP over TCP
- Eliminates the compatibility issues with transporting RTP/RTCP packets over UDP
- Limitation
  - Not all RTSP servers and clients support RTP/RTCP over TCP so fallback to UDP is mandatory.

RTSP-in-HTTP Tunneling

- RTSP+RTP/UDP
  - Requires explicit support from NAT and firewalls
- RTSP+RTP/TCP
  - Compatible with NAT and most firewalls
  - Will be blocked by HTTP-only firewalls
  - Does not work over proxied-clients

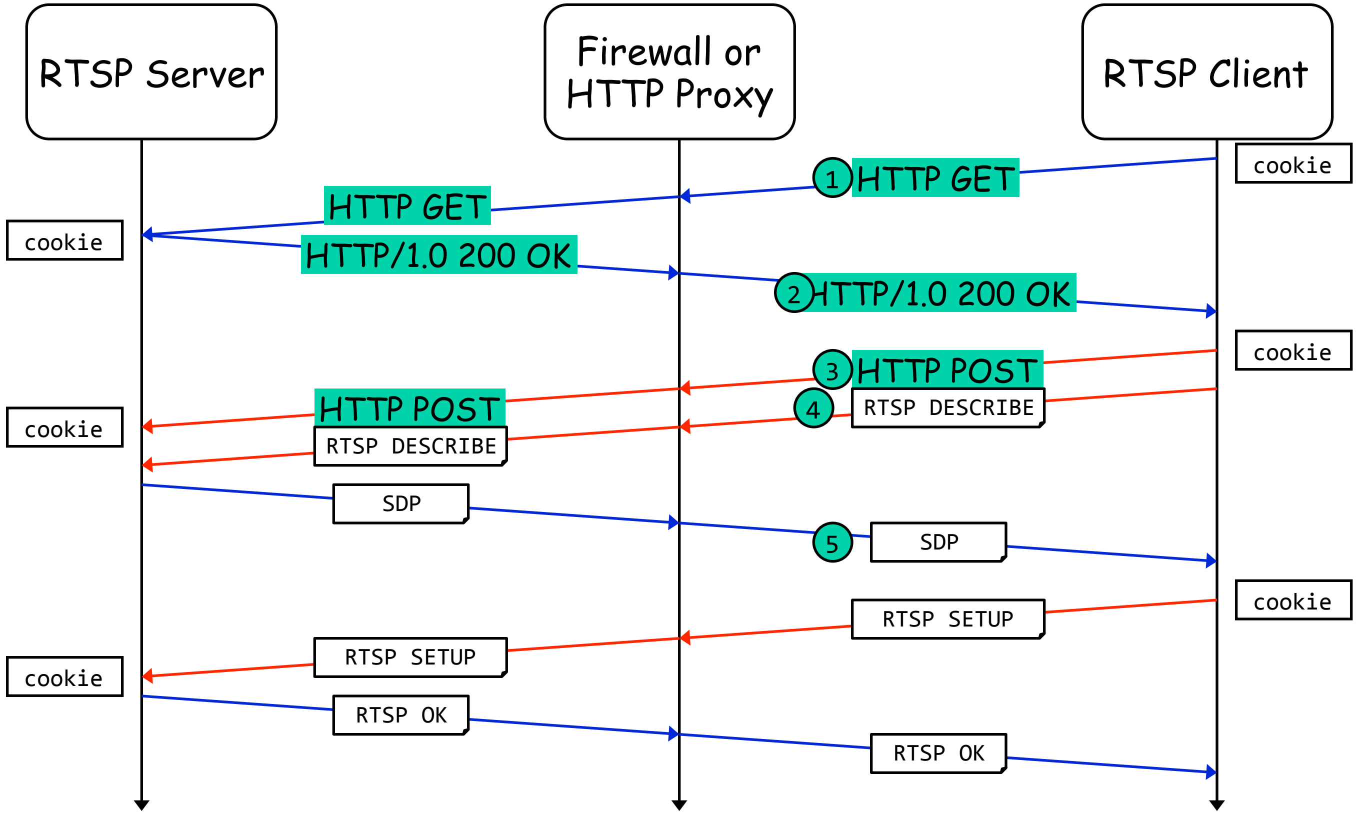


RTSP-in-HTTP Tunneling (cont'd)

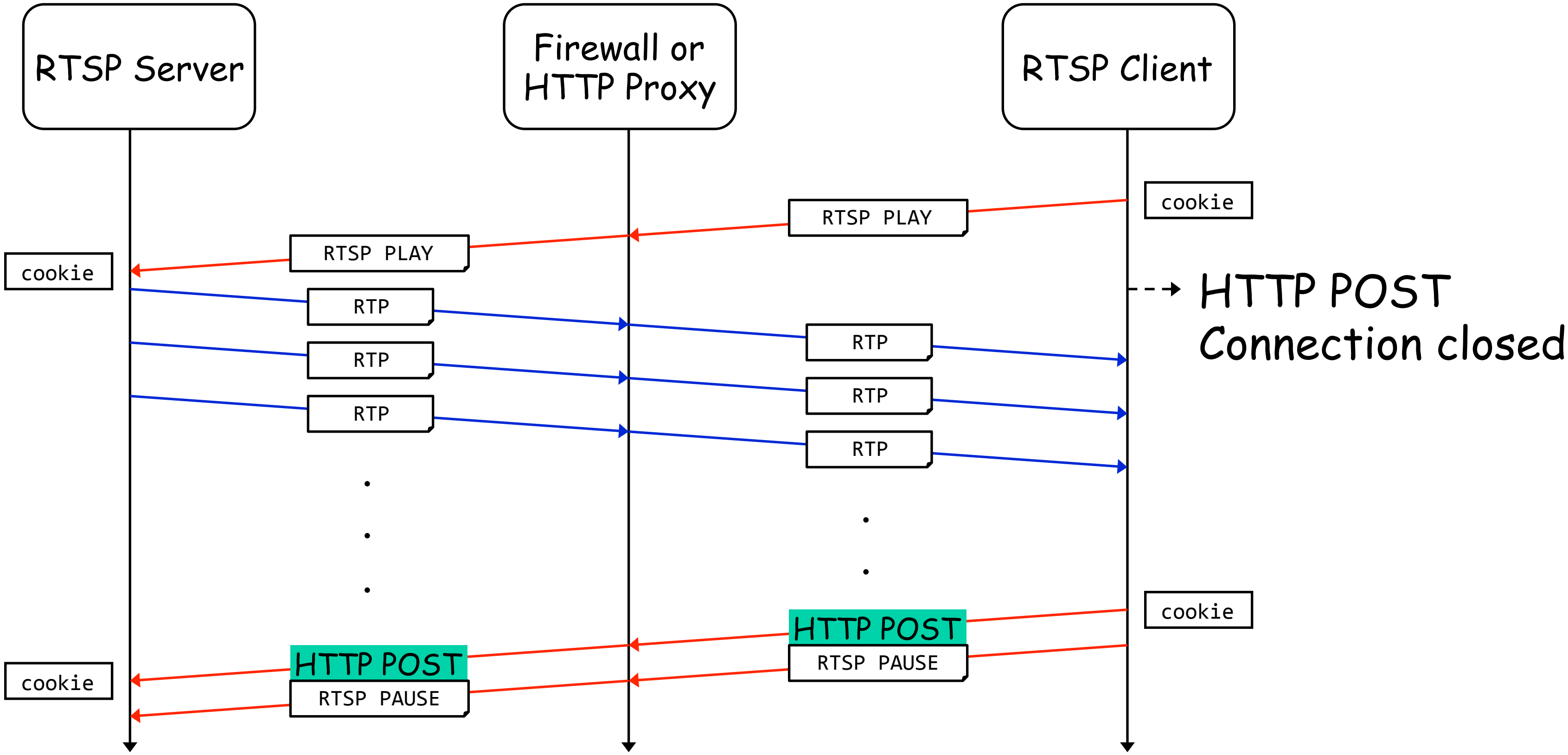
- Tunneling RTSP/RTP over HTTP
  - Proposed by Apple in 2001
    - Gentric and Jones, Tunneling RTSP in HTTP, IETF Draft, Jul 2001. (URL: http://tools.ietf.org/html/draft-gentric-avt-rtsp-http-00)
  - Implementations
    - Apple's Quicktime Player
    - Live555 RTSP/RTP library/server
  - Principles
    - Transfer RTSP protocol messages as HTTP payload
    - Two HTTP connections are used together
      - HTTP GET for returning data from server to client
      - HTTP POST for sending data from client to server
    - RTSP messages are base64 encoded to prevent firewall from identifying the RTSP flow

RTSP-in-HTTP Tunneling (cont'd)

- Tunneling RTSP/RTP over HTTP

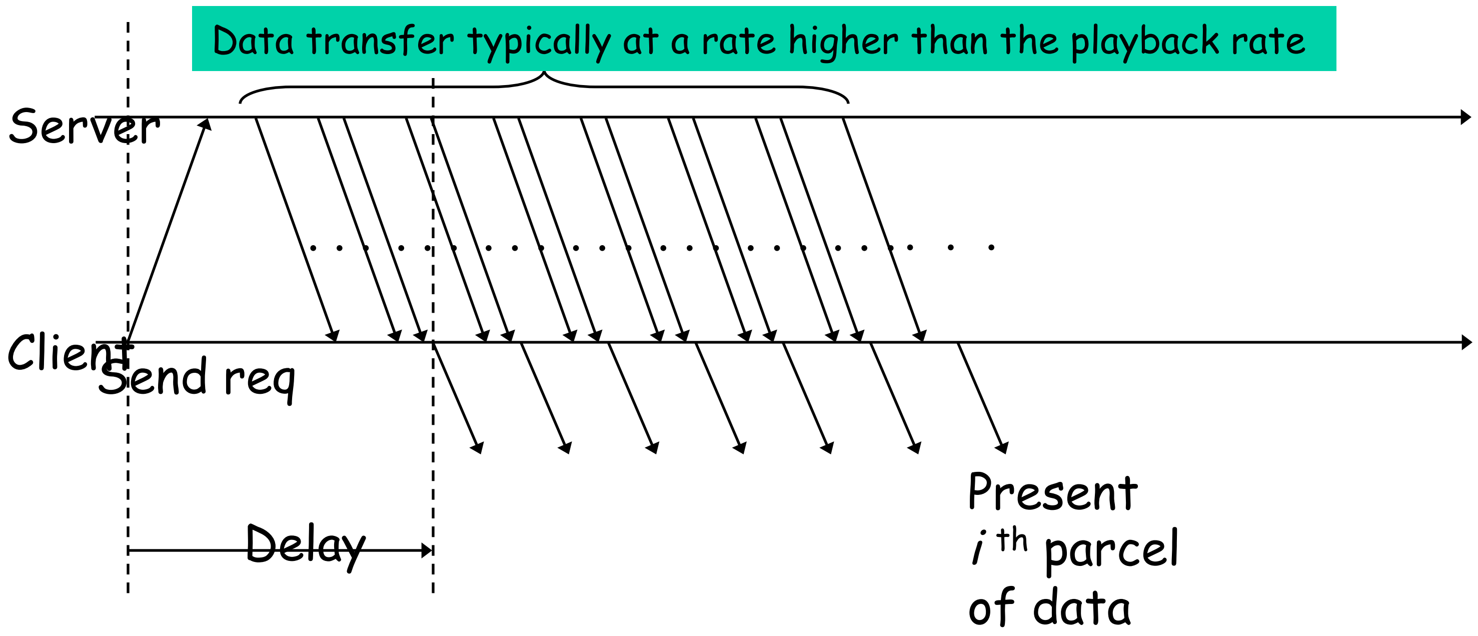


RTSP-in-HTTP Tunneling (cont'd)



HTTP Progressive Download

- HTTP over TCP
  - Streaming according to resource availability (as opposed to video bit-rate)

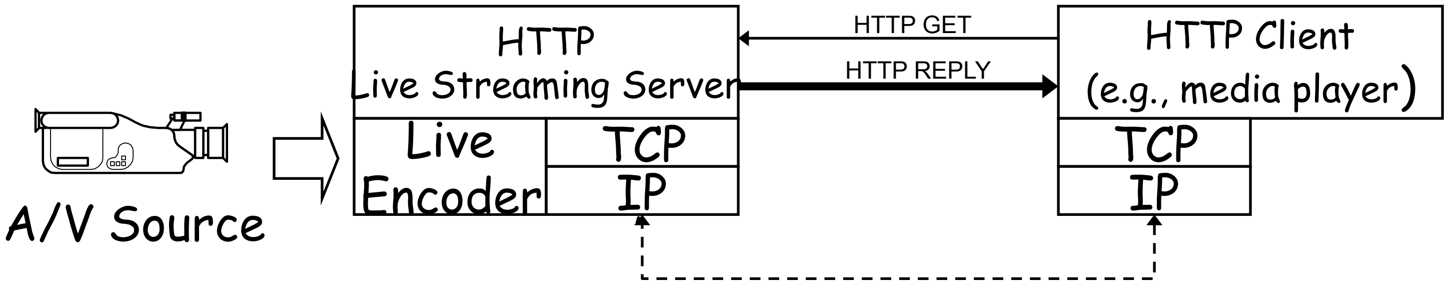


HTTP Progressive Download (cont’d)

- Shortcomings
  - Bandwidth Utilization
    - Not all videos are played from start to finish
    - Buffered video data that are not played, are bandwidth wasted
    - Potential charging and data metering issues in mobile networks
  - Playback Monitoring
    - HTTP server does not know the playback point of the client
    - There is no quality feedback (e.g., packet loss in RTCP) of the channel, except for TCP throughput

HTTP Progressive Download (cont'd)

- Live Video Streaming
  - Live video source which is encoded in real-time cannot be supported by file-based HTTP servers
  - Special "live" HTTP server is needed to support streaming live video over HTTP
    - A module to interface to the live video source
    - Adopt a container format which can support live video (e.g., MPEG2-TS)
    - Generates a virtual file in real-time for streaming



Apple's HTTP Live Streaming Protocol

- Proposed by Apple for use in iOS devices
  - Informational draft defined in R. Pantos and W. May, *HTTP Live Streaming*, draft-pantos-http-live-streaming-20, Sept 2016.
- Design Goals
  - Support streaming of live video (or dynamically-generated video)
  - Support content encryption
  - Support video of unbounded duration
  - Support multiple versions of the video (e.g., in different bit-rates)
- Current Implementation
  - The Quicktime player in Apple's iOS devices (iPhone, iPad, iPod Touch, Apple TV)
  - Not yet supported in desktop version of Quicktime Player in Windows

Other Proprietary Streaming Protocols

- Microsoft Media Services (MMS)
  - Employs TCP for exchange of control messages
  - Delivers media data over either UDP or TCP
- RealNetworks Data Transport (RDT)
  - Delivers media data over UDP
- Adobe Real-Time Messaging Protocol (RTMP)
  - Can be transported over either UDP or TCP
  - Supports Digital Rights Management (RTMP-E)
  - Supported by the Flash player only

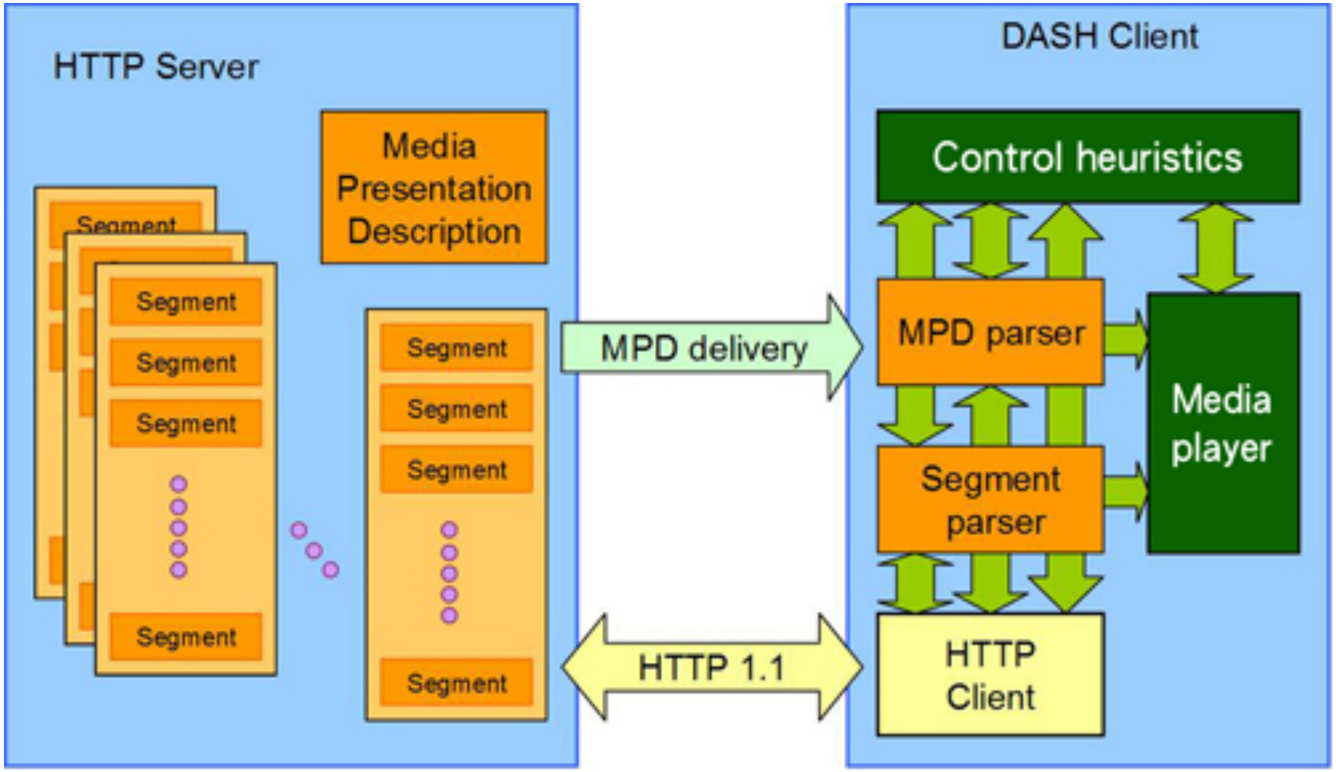
MPEG Dynamic Adaptive Streaming over HTTP (MPEG DASH)

- Background
  - Increasing popularity of HTTP-based streaming in the Internet
  - Emergence of multiple HTTP streaming protocols by different vendors
    - File-based HTTP progressive download
    - Apple's Live HTTP Streaming (m3u8/mpeg-ts)
    - Adobe's HTTP Dynamic Streaming
    - Microsoft's Smooth Streaming
- The Motion Picture Expert Group
  - Develops a new ISO standard for HTTP-based streaming of both pre-recorded and live video
  - Enhance inter-operability of streaming servers, players, and content distribution networks

MPEG DASH (Cont'd)

- MPEG Dynamic Adaptive Streaming over HTTP
  - MPEG issued a Call for Proposal for an HTTP streaming standard in April 2009
  - Fifteen full proposals were received by July 2009, when MPEG started the evaluation of the submitted technologies.
  - In the two years that followed, MPEG developed the specification with participation from many experts and with collaboration from other standard groups:
    - DASH is based on Adaptive HTTP streaming (AHS) in 3GPP Release 9 and on HTTP Adaptive Streaming (HAS) in Open IPTV Forum Release 2.
    - As part of their collaboration with MPEG, 3GPP Release 10 has adopted DASH (with specific codecs and operating modes) for use over wireless networks.
  - MPEG-DASH standard was published as ISO/IEC 23009-1:2012 in April 2012.

Architecture of DASH

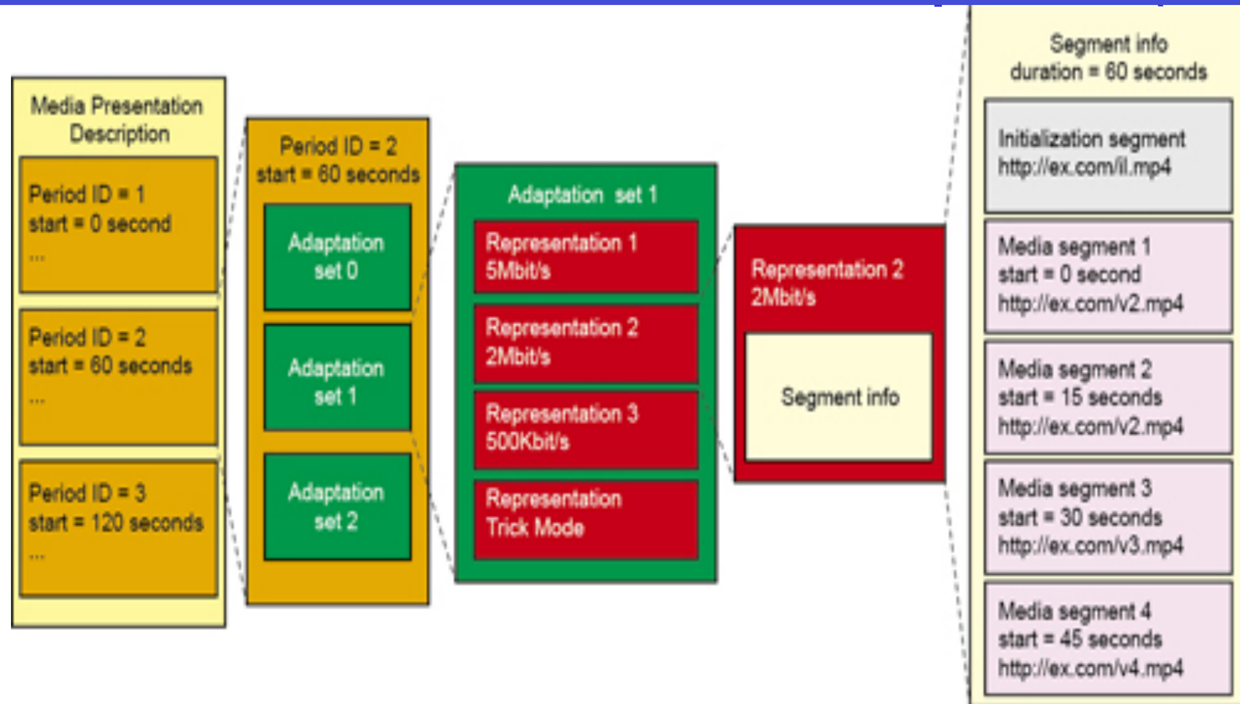


- DASH is an adaptive bitrate streaming technology where a multimedia file is partitioned into one or more segments and delivered to a client using HTTP.
- The multimedia contents stored on server in two parts:
  - A Media Presentation Description (MPD) which describes a manifest of the available content, its various alternatives, their URL addresses, and other characteristics.
  - Segments can contain the media data in chunks.

DASH Playout Mechanism

- In order to play the actual content, the DASH client needs to get the MPD from server first.
- The client then extract the content information from the MPD using MPD parser.
- Based on the extracted info, DASH client uses HTTP GET to obtain the corresponding multimedia segments and conduct a buffered play-out
  - While the DASH client continues to fetch the next segments for playing, it also monitors the fluctuation of available network bandwidth.
  - If needed, the client may choose to grab segments with lower resolution (and thus transmission rate) to maintain play-out quality.

Media Presentation Description (MPD)



- The DASH manifest file, called the Media Presentation Description (MPD), is an XML file that identifies the various content components and the location of all alternative streams.
- The MPD file defines the video sequence with one or more consecutive periods that break up the video from start to finish.
  - Each Period represents a period of program with attributes of start time & program length.
- Each period contains multiple adaptation sets that contain the content that comprises different the audio/video experience.
- This content can be mixed, in which case there might be one adaptation set, or represented in elementary streams, enabling features like multiple languages support for audio.

Early Implementations of DASH

- The first DASH player implementations on desktop computer are:
  - DASH VLC plugin of the Institute of Information Technology (ITEC) at Alpen-Adria University Klagenfurt and
  - The multimedia framework of the GPAC group at Telecom ParisTech.

- The first DASH server and Android (2.2 to 4.x) SDK player implementation was demonstrated by RealNetworks at the IBC 2012 with:
  - Helix Universal Server and
  - Helix SDK for Android demonstrating MPEG2-TS (Smart TV) and
  - ISO BMFF (MP4 Smartphone / Tablet) delivery and playback formats commercially available from November 2012.

Yet Another Development: HTML5 Video

- HTML5 specification introduced the <video> element for the purpose of playing videos/movies, partially replacing the <object> element.

Example

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

An example: http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_all

- Objective is to show video on the Web without the need of browser plugins
- Effort hampered by lack of agreement (between Google, Apple, Microsoft and MPEG) on the which video format should be supported as the "standard" one:
  - Non-free format: H.264/MPEG-4 AVC supported by Apple and Microsoft
  - Free Formats:

    WebM: (VP8 or VP9 video and Vorbis audio) by Google

    Theora video, Vorbis audio and Ogg container formats by Ogg Theora

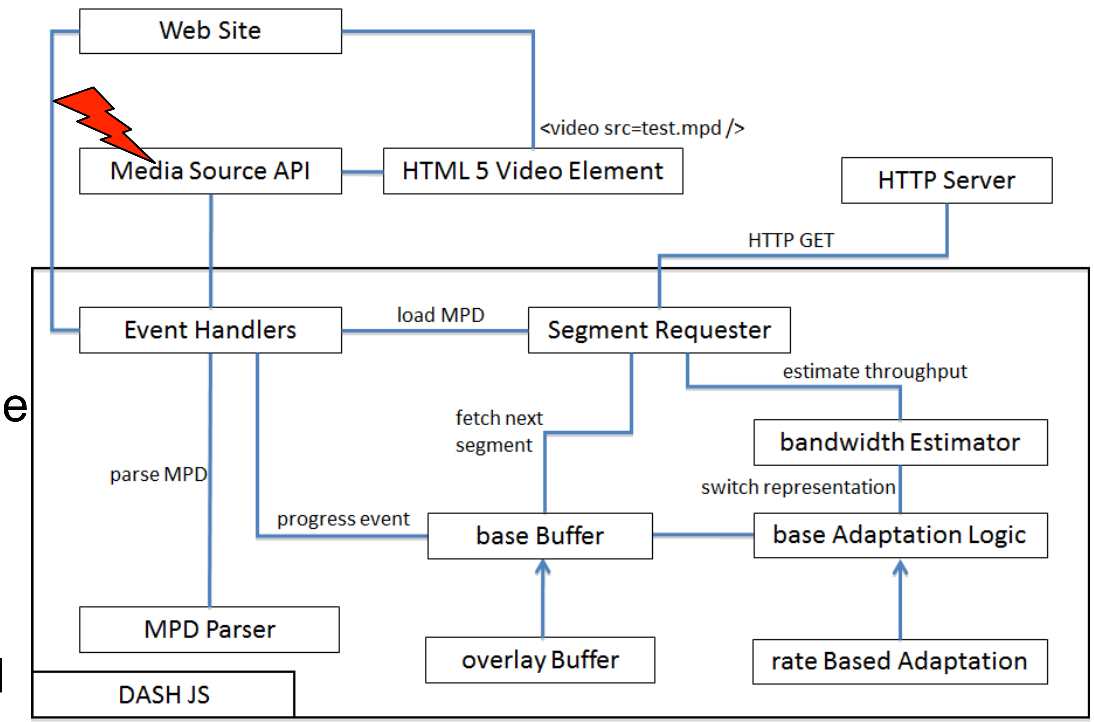
Yet Another Development: HTML5 Video (cont'd)

Browser	Operating System	Theora	H.264 (MP4)	HEVC (MP4)	VP8 (WebM)	VP9 (WebM)
Android browser	Android	Since 2.3[44]	Since 3.0[44]	No[45]	Since 2.3[44]	Since 4.4[46]
Chromium	Unix-like and Windows	Since r18297[47]	Via a plugin[48][49]	No[50]	Since r47759[51]	Since r172738[52]
Google Chrome	Unix-like, Android, OS X, iOS, and Windows	Since 3.0[53][54]	Since 3.0[54][a]	No[45]	Since 6.0[56][57]	Since 29.0[b]
Internet Explorer	Windows	Via OpenCodecs	Since 9.0[60]	No[45]	Via OpenCodecs	No
	Windows Phone	No	Since 9.0[61]		No	
	Windows RT		Since 10.0[61]			
Microsoft Edge	Windows 10	No[62]	Since 12.0[63]	Needs hardware decoder[c]	No	Yes[66]
	Windows 10 Mobile	No	Since 13.0[67]	Needs hardware decoder[c]	No	Yes[66]
Konqueror	Unix-like and Windows	Needs OS-level codecs[d]				
Mozilla Firefox	Windows 7+	Since 3.5[69]	Since 21.0[e]	No[45]	Since 4.0[72][73]	Since 28.0[74][75]
	Windows Vista		Since 22.0[76]			
	Windows XP and N editions		Since 46.0[77]			
	Linux		26.0 (via GStreamer)[f] 43.0 (via FFmpeg)[80]			
	Android		Since 17.0[81]			
	OS X		Since 34.0[82]			
	Firefox OS		Since 1.1[83]			
Opera Mobile	Android, iOS, Symbian, and Windows Mobile	Since 13.0	Since 11.50	No[84]	Since 15.0	Since 16.0
Opera	OS X, Windows, Linux	Since 10.50[85]	Since 24.0[86]		Since 10.60[87][88]	Yes
Safari	iOS	No	Since 3.1[89][90]	No[45]	No	No
	OS X	Via Xiph QuickTime Components			Via a plugin[91]	No
Web	Linux and BSD	Needs OS-level codecs[g]				

Integrating HTML5 video and DASH ?

DASH-JS: an Open-source DASH client in JavaScript [1]

- It integrates HTML5 WebM video with MPEG-DASH by packing each GOP in one WebM cluster and then generates the MPD required by DASH
- Implement DASH entirely in JavaScript, using Media Source API [2,3] (extension to ISO base Media File Format) provided by the Google Chrome browser
  - Media Source API allows one to push different streams of video data directly to the Browser's video decoder
- Adaptation logics can be changed readily.



Source of the figure: [4]

[1] Open source: http://dash.itec.aau.at
[2] http://gpac.wp.mines-telecom.fr/2012/08/23/mpeg-dash-support-in-google-chrome/
[3] https://dvcs.w3.org/hg/html-media/raw-file/tip/media-source/media-source.html
[4] Benjamin Rainer, Stefan Lederer, Christopher Müller and Christian Timmerer, A Seamless Web Integration of Adaptive HTTP Streaming, in Procs. of the 20th European Signal Processing Conference 2012, Bucharest, Romania, August 27-31, 2012
http://www.slideshare.net/christian.timmerer/a-seamless-web-integration-of-adaptive-http-streaming

References for Streaming Protocols

- H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, April 1998.
- H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications, RFC 3550*, July 2003.
- J. Lazzaro, *Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport*, RFC4571, July 2006.
- Gentric and Jones, Tunneling RTSP in HTTP, IETF Draft, Jul 2001. URL: http://tools.ietf.org/html/draft-gentric-avt-rtsp-http-00
- Tunnelling RTSP and RTP through HTTP, Dispatch 28, Apple, https://developer.apple.com/quicktime/icefloe/dispatch028.html
- Apple HTTP Live Streaming, URL: https://developer.apple.com/resources/http-streaming/
- Andrew Fecheyr-Lippens, *A Review of HTTP Live Streaming*, URL: http://files.andrewsblog.org/http_live_streaming.pdf
- Alex Zambelli, IIS Smooth Streaming Technical Overview, Microsoft Corporation, Mar 2009. URL: http://jmvm.vse.cz/wp-content/uploads/2011/05/t_IIS_Smooth_Streaming_Technical_Overview.pdf
- Ali C. Begen, Watching Video over the Web: Adaptive Streaming over HTTP, Cisco, 2011.
- MPEG-DASH Working Documents available at http://mpeg.chiariglione.org/working_documents.php#MPEG-DASH
- MPEG-DASH Promoters' Group, homepage at http://dashpg.com
- I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet", *IEEE Multimedia*, Oct-Nov 2011.
- T. Stockhammer, I. Sodagar, "MPEG DASH: The Enabler Standard for Video Deliver Over The Open Internet," *IBC Conference 2011*, Sept 2011.
- I. Sodagar and H. Pyle, "Reinventing multimedia delivery with MPEG-DASH", *SPIE Applications of Digital Image Processing XXXIV*, Sept 2011.
- http://en.wikipedia.org/wiki/HTML5_video
- http://www.longtailvideo.com/html5/
- http://www.scoop.it/t/html5-and-adaptive-streaming-video

IPTV = Internet Protocol + Television

A sample IPTV service package

Source: Cisco

Top 10 Telco IPTV deployments worldwide

Rank	Carrier	Country	IPTV Subs	Broadband Subs	
1	Iliad (Free)	France	2,170,000	2.77 million	78.4%
2	France Telecom	France	975,000	6.9 million	14.1%
3	PCCW	Hong Kong	818,000	1.18 million	69.3%
4	Neuf Cegetel	France	600,000	3.12 million	19.2%
5	Telefonica	Spain	469,067	4.34 million	10.8%
6	Chunghwa Telecom	Taiwan	358,000	4.07 million	8.8%
7	China Telecom	China	310,000	35.1 million	0.9%
8	Belgacom	Belgium	249,434	1.20 million	20.8%
9	TeliaSonera	Sweden	216,000	1.03 million	21.5%
10	Fastweb	Italy	170,000	1.25 million	13.6%

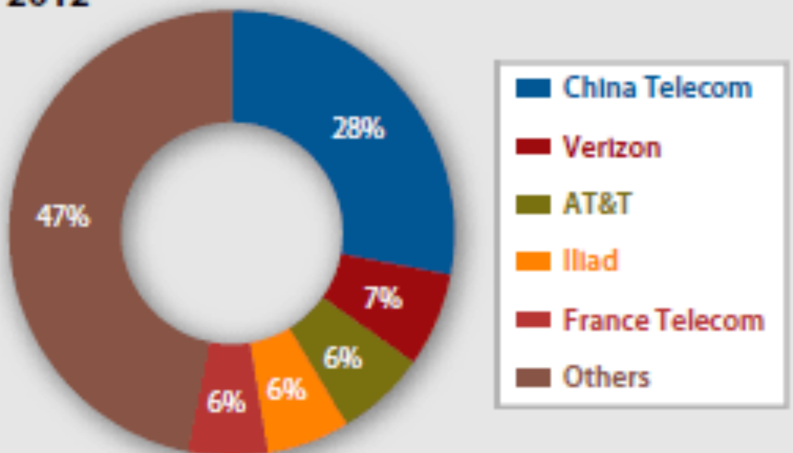
From Light Reading report "Top Ten IPTV Carriers" issued Jan 14, 2008

Verizon FiOS is not included since its broadcast channels are not delivered via IPTV

Source: Cisco

Top IPTV operators by Est. Global Market Share

Top 5 IPTV Operators, by Global Market Share, 2012



Top 5 Countries by IPTV Subscribers

2012 Rank		2012	2013	2014	2015	2016	2017	CAGR '12-'17
		(000)						(%)
1	China	15,172	17,478	18,322	19,661	20,965	22,427	8.1
2	France	13,372	14,179	14,757	15,212	15,565	15,846	3.5
3	U.S.	9,917	11,307	12,639	13,859	15,059	16,226	10.3
4	South Korea	4,380	4,900	5,317	5,703	6,001	6,266	7.4
5	Japan	3,057	3,358	3,694	4,064	4,446	4,864	9.7
	Total Top 5	**30,726**	**33,744**	**36,407**	**38,838**	**41,071**	**43,203**	**7.1**
	Global Total	**65,007**	**73,375**	**79,717**	**86,308**	**93,255**	**100,544**	**9.1**
Share of Global IPTV Market (%)								
1	China	23.3	23.8	23.0	22.8	22.5	22.3	(0.9)
2	France	20.6	19.3	18.5	17.6	16.7	15.8	(5.2)
3	U.S.	15.3	15.4	15.9	16.1	16.1	16.1	1.1
4	South Korea	6.7	6.7	6.7	6.6	6.4	6.2	(1.5)
5	Japan	4.7	4.6	4.6	4.7	4.8	4.8	0.6
	Total Top 5	**47.3**	**46.0**	**45.7**	**45.0**	**44.0**	**43.0**	**(1.9)**

Source: http://www.broadbandtvnews.com/2013/04/25/over-100m-iptv-homes-by-2017/

http://www.broadbandtvnews.com/2013/06/07/top-20-iptv-providers-served-57-7m-subscribers/

IPTV Video Service Requirements

Service	B/W Up (kbps)	B/W Down (kbps)	Jitter/Latency	PLR
Data	100	5000	High/High	10^{-3}
Voice	100	100	Med/Low	10^{-4}
BTV MPEG4	5	2200 SD 9000 HD	Low/Med	10^{-6}
BTV MPEG2	5	4000 SD 15000 HD	Low/Med	10^{-7}
VoD MPEG2	5	4000 SD 15000 HD	Low/Med	10^{-7}
Gaming	20	100	Low/Low	10^{-3}

Source: Cisco

Digital Video Bandwidth Requirements

Uncompressed Digital Video	
SDTV (480i CCIR 601 over SD-SDI SMPTE 259M)	165.9 – 270 Mbps
EDTV (480p or 576p via SMPTE 344M)	540 Mbps
HDTV (1080i or 720p over HD-SDI SMPTE 292M)	1.485 Gbps
HDTV (1080p over Dual link HD-SDI SMPTE 372M)	2.970 Gbps
MPEG-2 Compressed Video	
SDTV Broadcast (3.75 Mbps for cable VOD)	3 – 6 Mbps
HDTV Broadcast (19.3 Mbps for ATSC DTV)	12 – 20 Mbps
SDTV Production (Contribution – 4:2:2 I-frame only)	18 – 50 Mbps
HDTV Production (Contribution – 4:4:4 I-frame 10-bit)	140 – 500 Mbps
MPEG-4 AVC / H.264 Compressed Video	
SDTV Broadcast (about 50% less than MPEG-2)	1.5 – 3 Mbps
HDTV Broadcast (1080i about 4x SDTV)	6 – 9 Mbps

Enable a 200:1 compression ratio for HDTV

Source: Cisco

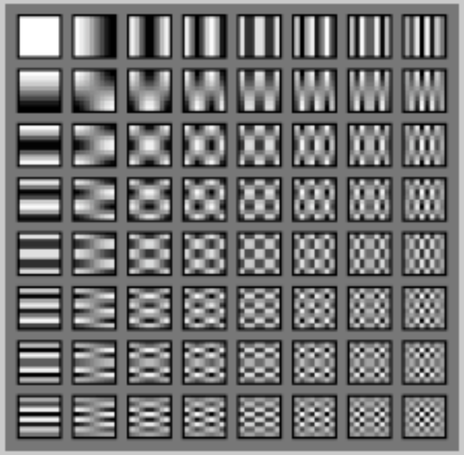
Building Blocks of an IPTV System

Video Acquisition

Satellite Reception
Off-Air Reception
Satellite, Off-Air, and Fiber Receivers
Signal Conversion

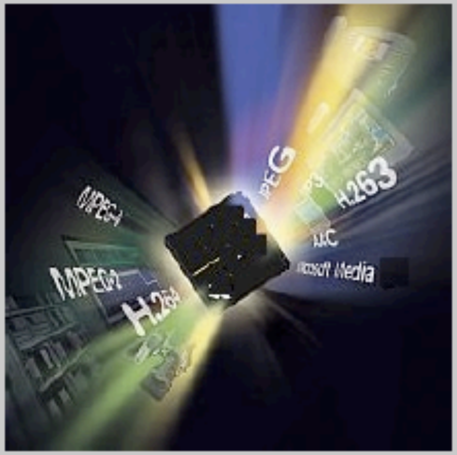
Video Encoding

MPEG-2
MPEG-4 AVC
Standard Definition
High Definition
Audio Encoding



Video Processing

Transcoding
Transrating
Splicing
Multiplexing
Ad-Insertion



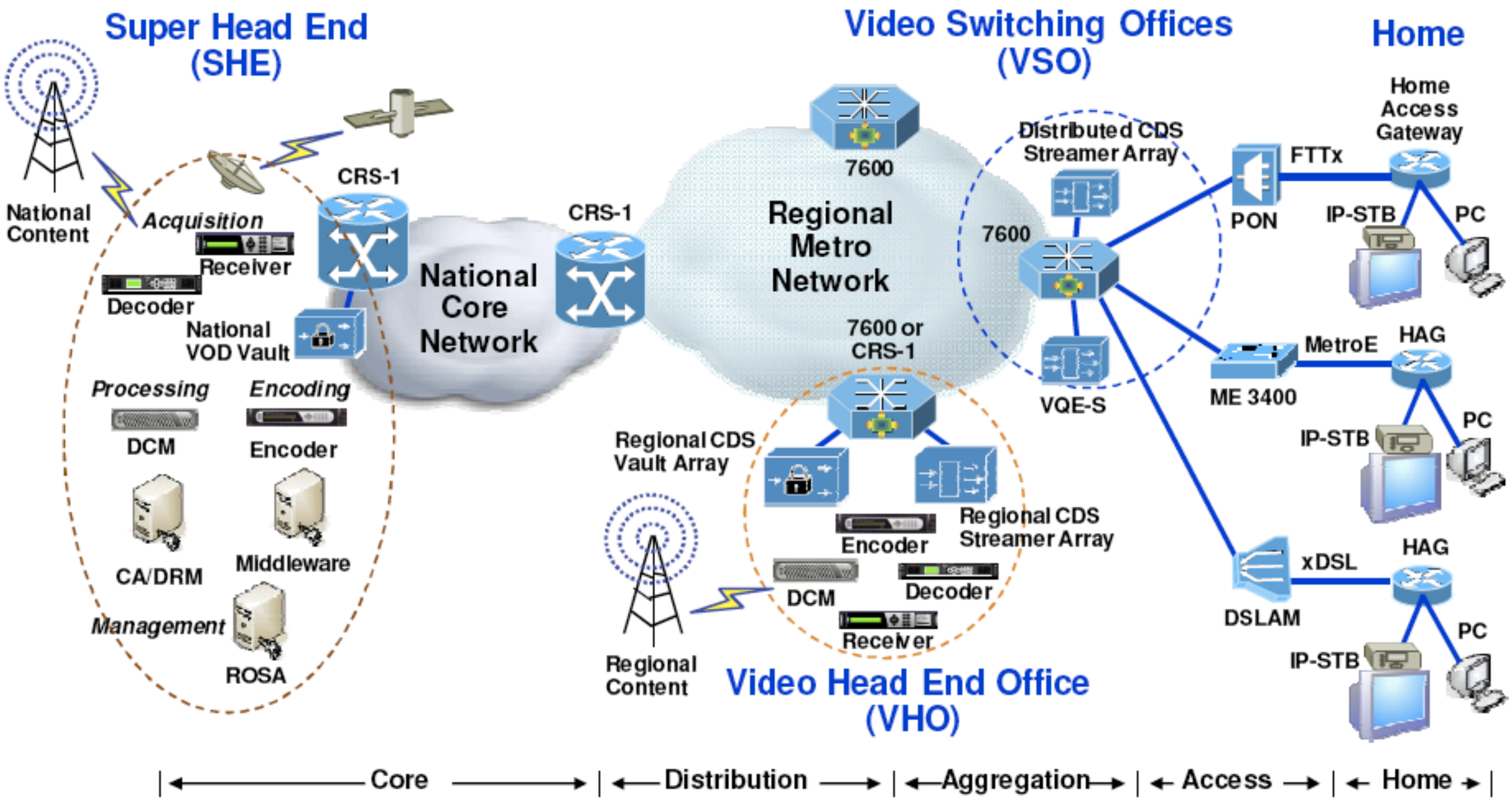
Video Management

CAS/DRM
Remote Operations
Metadata, Billing
VOD Servers
Video Applications

Acquire, Process and Transmit Video

Source: Cisco

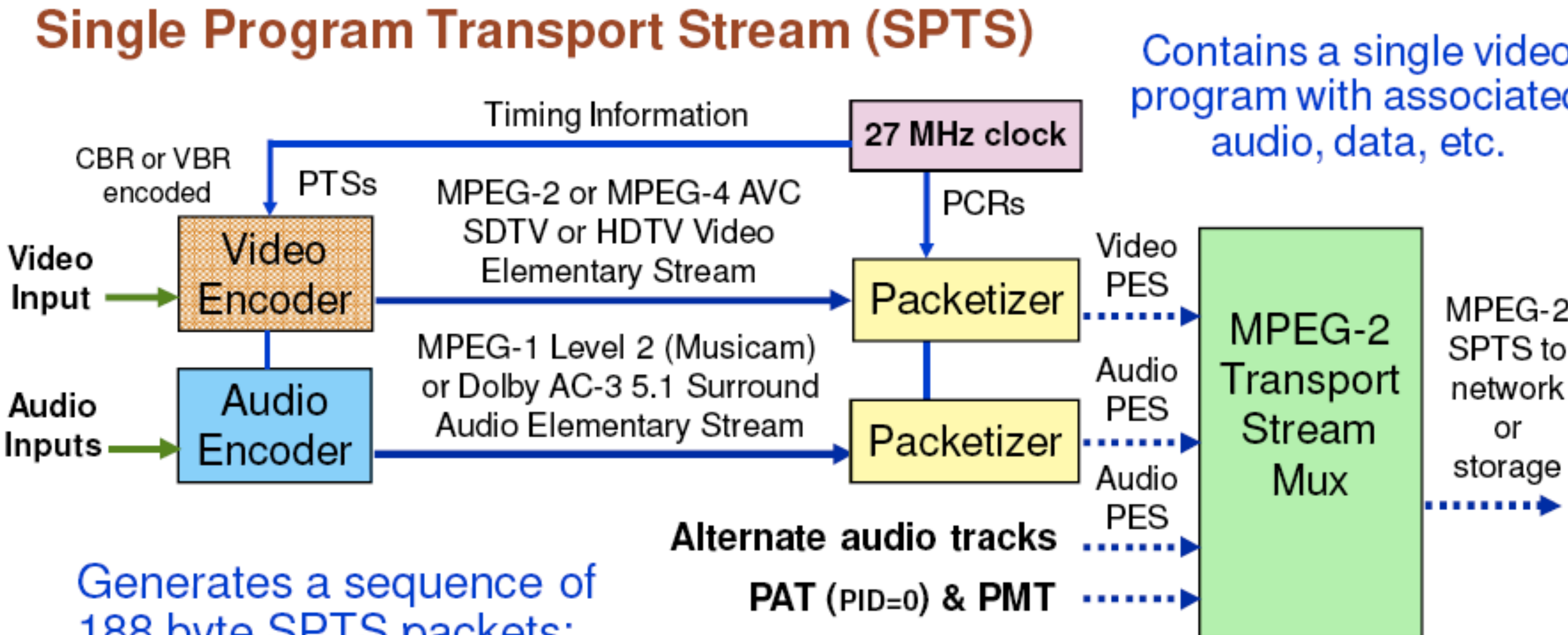
Telco IPTV Network Architecture



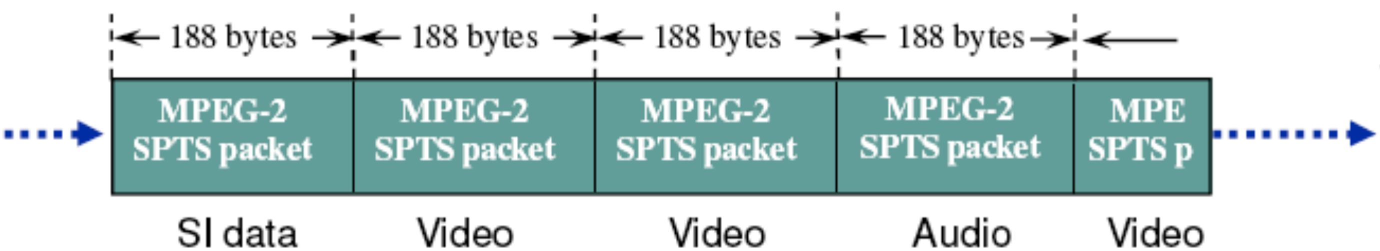
An End-to-end IP network

Packetization into a MPEG-2 SPTS

Contains a single video program with associated audio, data, etc.



Generates a sequence of 188 byte SPTS packets:

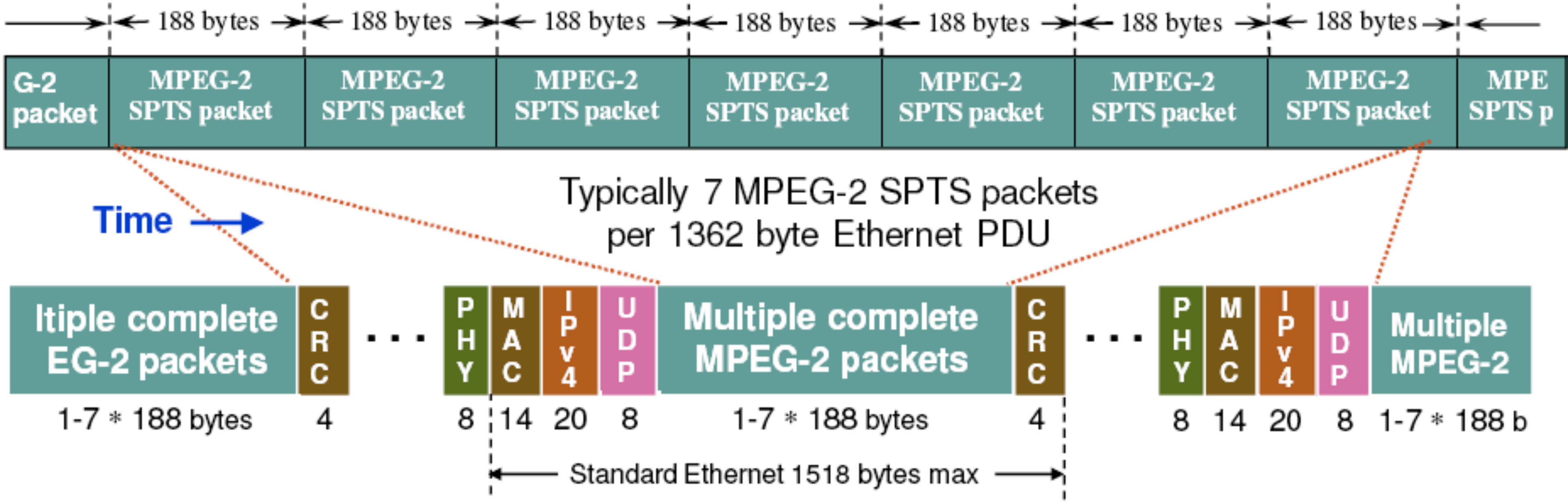


Transport Stream defined by ISO/IEC 13818-1 (ITU-T H.222.0)

Each packet identified by a 13-bit PID value

Source: Cisco

MPEG-2 SPTS over UDP/IP Video Delivery

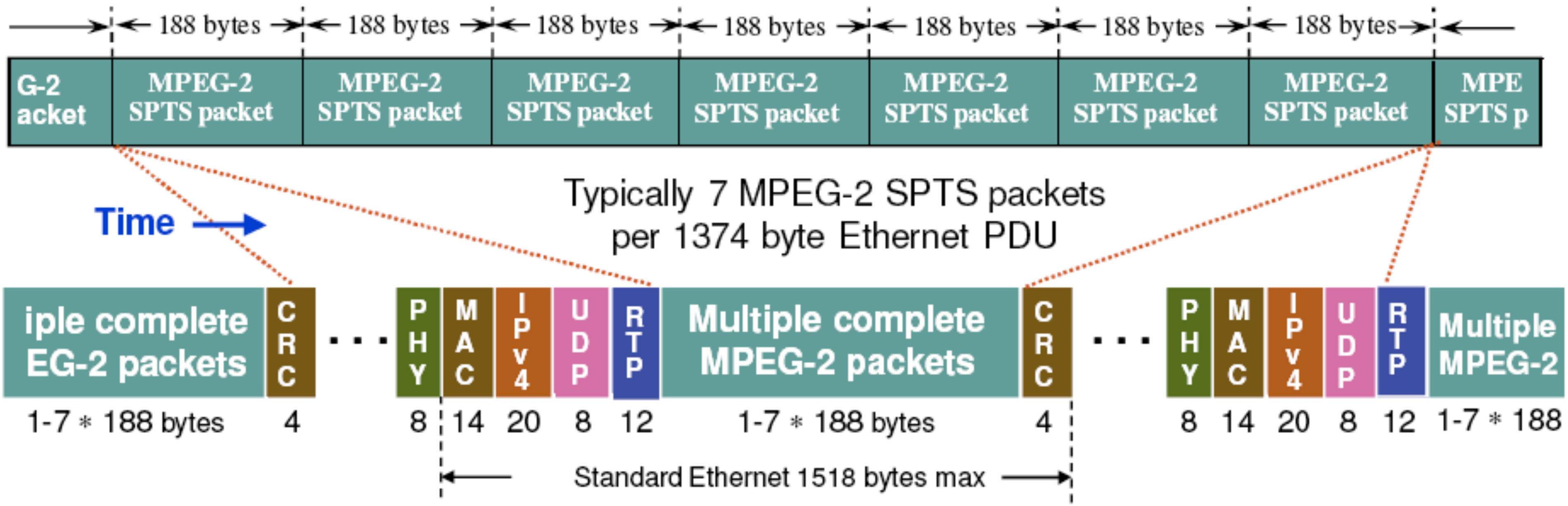


- One to seven MPEG-2 Single Program Transport Stream (SPTS) packets per Ethernet frame delivered directly over UDP/IP/Ethernet
  - For each 3.75 Mbps **MPEG-2** SD stream, one Ethernet frame every ~ 2.8 msec
  - For each 15.0 Mbps **MPEG-2** HD stream, one Ethernet frame every ~ 0.7 msec
- Up to 250 streams at 3.75 Mbps/stream per Gigabit Ethernet output
- UDP/IP/GigE delivery overhead is approximately 1 - (7*188/1370) = 4%

Common format today for Cable/Telco VoD

Source: Cisco

MPEG-2 SPTS over RTP/UDP/IP Video Delivery



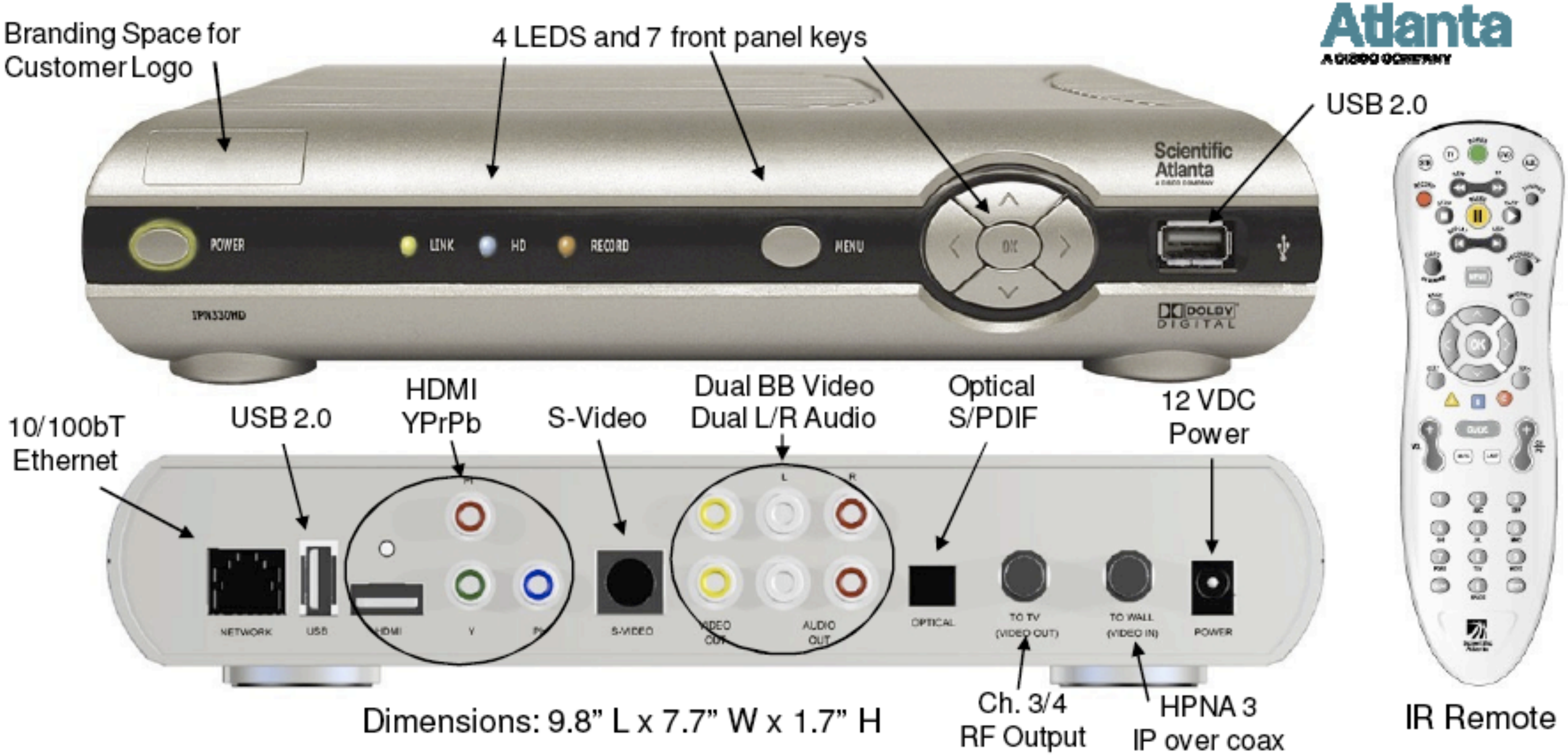
- Adds RTP-layer time stamp, sequence number, and other capabilities defined by IETF RFC 3550 (RTP) and RFC 2250 (MPEG over RTP)
- Still integral number of MPEG-2 TS packets per RTP message
  - For each 2 Mbps **MPEG-4 AVC** SD stream, Ethernet frame every 5.264 msec
  - For each 8 Mbps **MPEG-4 AVC** HD stream, Ethernet frame every 1.316 msec
- RTP/UDP/IP/GigE overhead is approximately 1 - (7*188/1382) = 5%

Preferred stack for all Real-Time IP streams

Source: Cisco

IP Set Top Boxes (IP-STB)

Example: SA's IPN330HD model

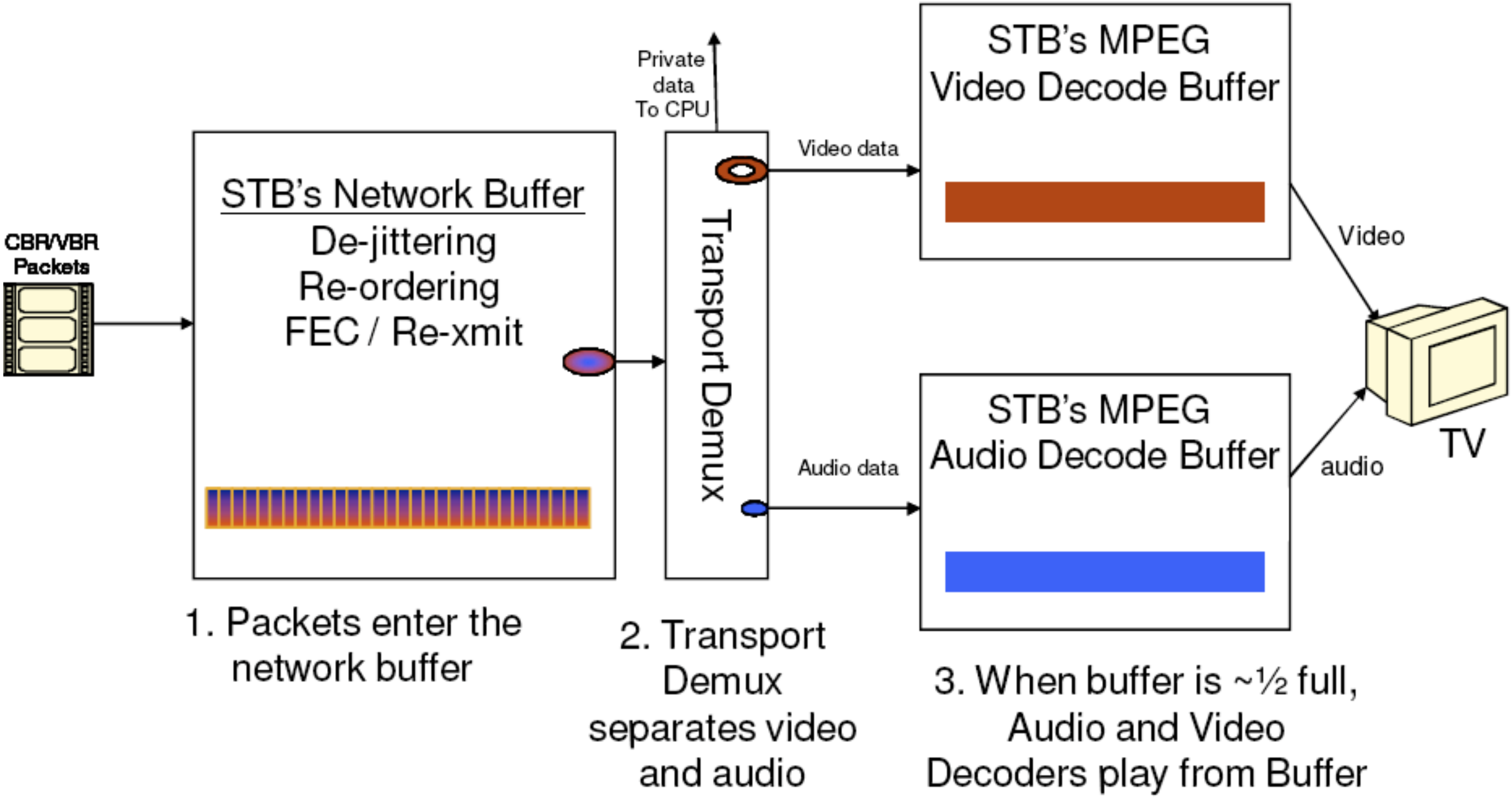


Standard & High Def MPEG-2 and H.264/MPEG-4 AVC codecs, SOC design, Multi-room DVR client, WinCE or Linux OS, Middleware & CA/DRM options

Interfaces between subscriber and IPTV service

Source: Cisco

Simplified Set Top Box (STB) Data Flow



Source: Cisco

< 10^{-6} PLR Requirement for Video

- Most critical: Packet Loss Ratio (PLR)
  - Video is compressed; Each packet carries multiple frames
    - Any loss likely causes visible artifact for a varying amount of time
  - Based on rule of thumb: no more than one artifact per 2 hour movie
    - For MPEG-2 Standard Definition content @ 3.75 Mbps this translates to a PLR of (7 x 188 x 8) / (3,750,000 x 3600 x 2) = < 0.390 x $\mathbf{10^{-6}}$
    - MPEG-4 AVC or SMPTE VC-1 High Definition @ 6 Mbps requires a PLR of (7 x 188 x 8) / (6,000,000 x 3600 x 2) = < 0.244 x $\mathbf{10^{-6}}$

→ Thus packet losses MUST be avoided!

- Causes for Packet Loss:

  1) Set Top Box Jitter or CODEC Buffer Overflow
  2) IP Router or Switch Buffer Overflow
  3) Bit Errors on Physical Links

Solve via accurate stream pacing

Solve with CAC + DiffServ

Solve excessive bit errors on non-fiber (wireless or copper) links using supplemental higher-level FEC or re-transmissions

- A deeper link-layer FEC over burdens VoIP & data applications

Source: Cisco

Retransmission on Access Networks:

Another way to solve packet loss problem

- Error rates on DSL access links are high enough to worry about
  - Can't easily achieve less than one video "glitch" per 2 hour movie
- Bulk L1/L2 FEC (Forward Error Correction) not an attractive option
  - Eats bandwidth (usable ADSL2+ capacity can drop from 28 to 18 mbps)
  - Introduces significant delays for other traffic (e.g. VoIP)
- Application layer FEC also has constant bandwidth overhead and is hard to tune for both BER and burst or congestive losses
  - DSL errors tend to group into 8 ms outages
  - Due to link layer Reed Solomon FEC failures
- Re-transmission on access is attractive since RTTs are short
  - Optimal use of bandwidth, only use bandwidth when correcting errors

Good news!

There is an excellent standard scheme for doing this with RTP

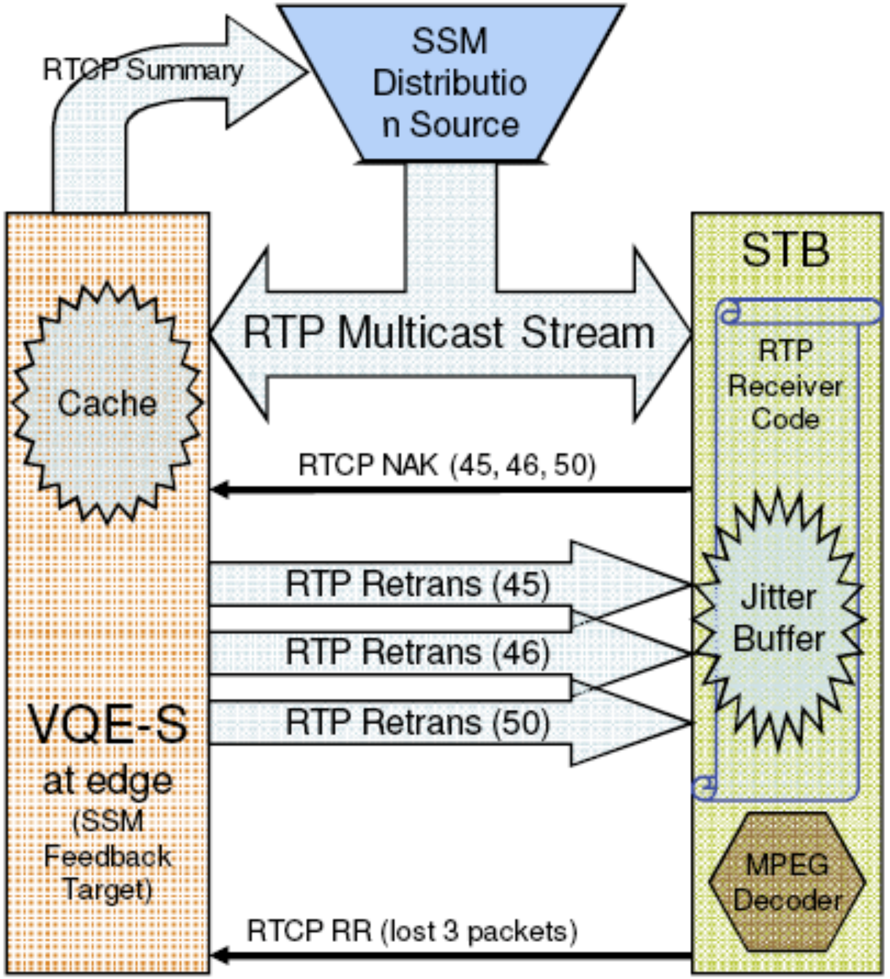
- You just have to "read between the lines"

Source: Cisco

Retransmission Protocol Mechanism:

- STB gets "nervous" after noticing (optional) FEC is over-run
  - STB sends RTCP NAK to feedback target with bitmap of missing packets
- Feedback target pulls missing packets out of the cache and retransmits them
  - Retransmission is on a separate unicast RTP repair session (or multicast session if sufficient collated error reports)
- Reception stats of each STB are sent periodically in RTCP Receiver Reports (RR) to feedback address
  - To monitor end-to-end QoE
- Feedback targets send summary reports to distribution source
  - Provides both fine-grained and aggregated reception quality data

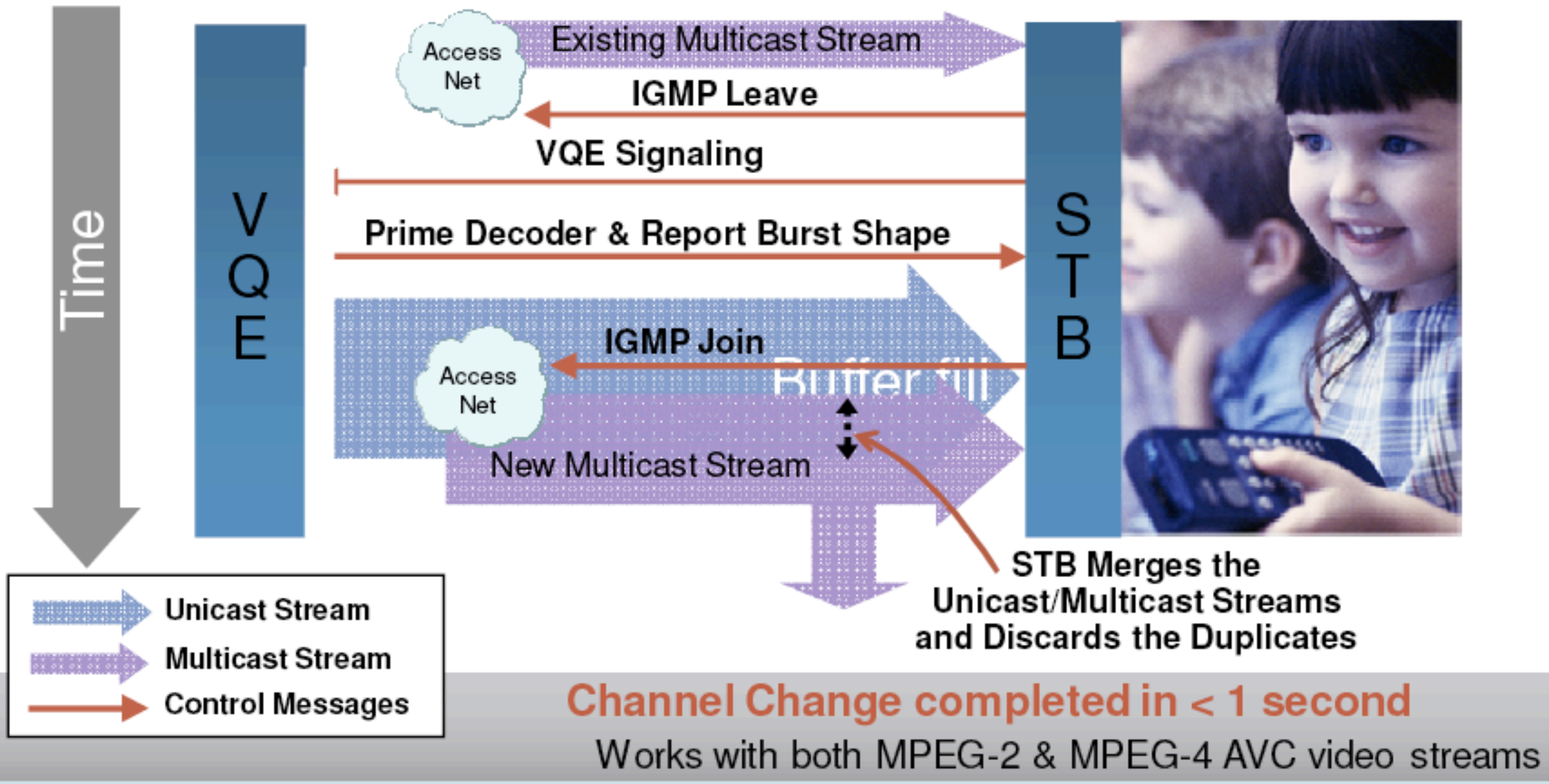
Details of operation



- Distance to Re-Tx server (VQE-S) determines jitter buffer requirements

Source: Cisco

Supporting Rapid Channel Change:



Source: Cisco

- Use RTCP just like the triggering of Re-transmission during error-repair

Summary

Principles

- classify multimedia applications
- identify network services applications need
- making the best of best effort service

Protocols and Architectures

- specific protocols for best-effort: RTSP, RTP, RTCP, MPEG-DASH, and more…
- An Example: IPTV Architecture