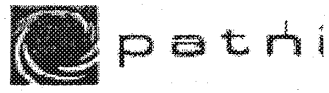
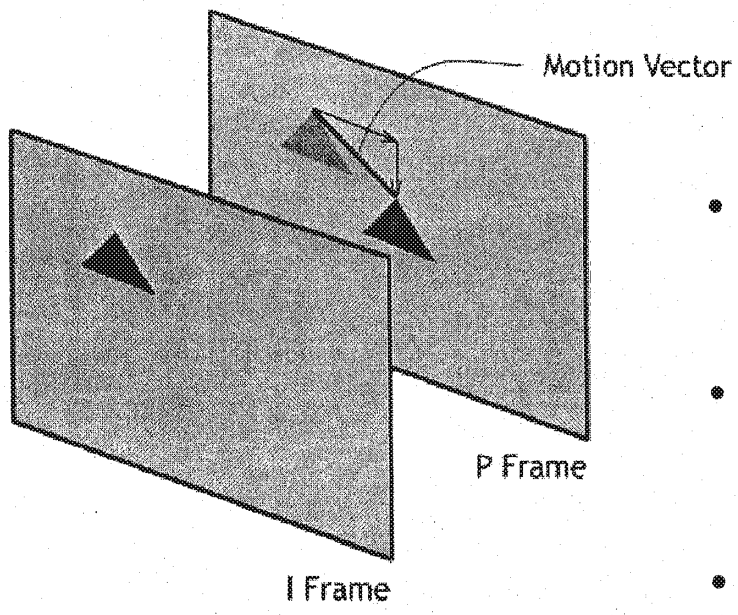
Texture Encoding Algorithms

- Discrete Cosine Transform (Forward/Inverse) based compression
- Two methods for Quantization : H.263 and MPEG.

  Simple Profile uses H.263 quantization only. Core Profile allows both.
- AC/DC Prediction for intra and inter blocks
- Three modes of scanning (Zigzag, Horizontal and Vertical) depending on the prediction direction
- Run-Length coding to transmit the VLCs in the bitstream



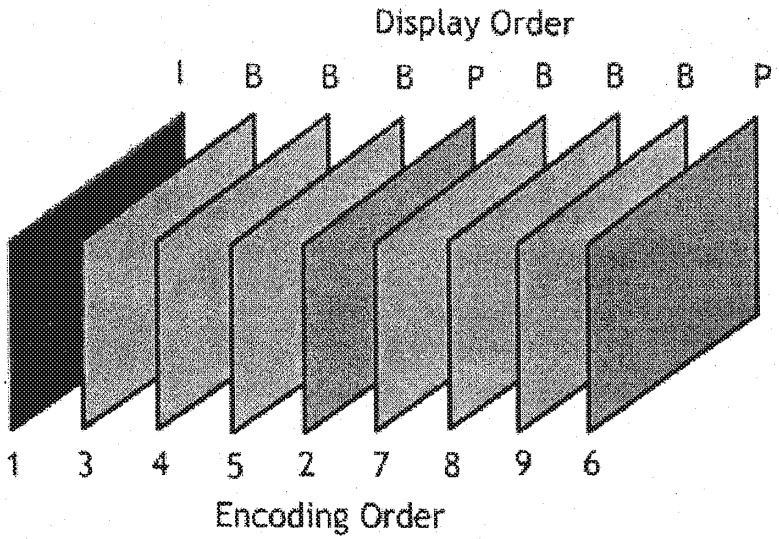
Motion Compensation



- Successive Video frames normally have only few regions that move. Even in presence of motion, the movement is within bounds.
- There exists temporal redundancy between two successive frames in video scene.
- Motion compensation exploits this redundancy for higher compression ratios.
- Both P and B frame based motion compensation is used in MPEG-4.

patni

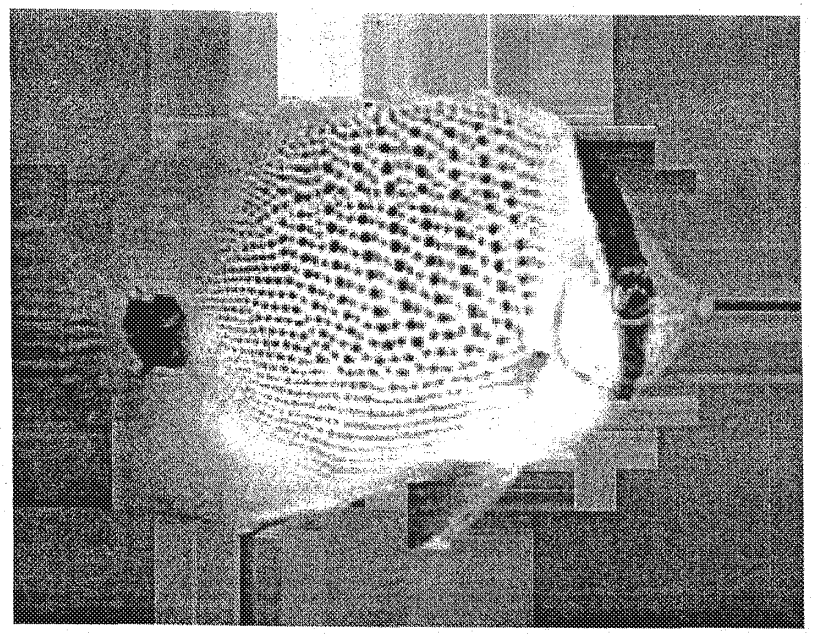
Encoding Sequence



- Encoding of B-frames requires the coding order to be different from display order.
- B-frames use the next and previous I or P frames for motion prediction.
- Encoder requires to buffer the raw video frames to support B-frame coding.

patni

Motion Compensation Algorithms

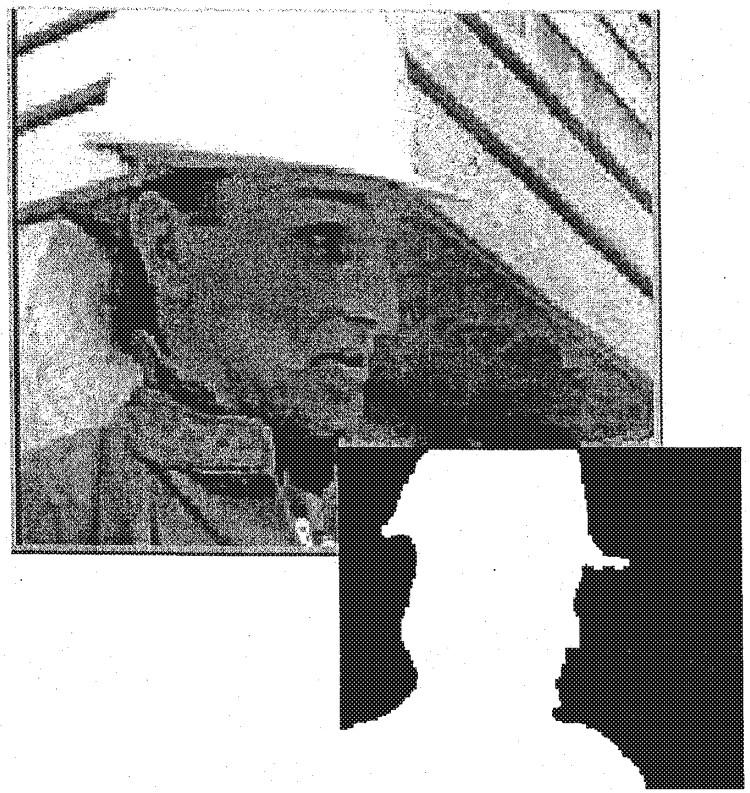


Motion Compensation Uses:

- Repetitive padding algorithm.
- Extended padding algorithm.
- Three step/Four step search for majority voting algorithm.
- Motion vector prediction.
- Intra/Inter encoding mode decision.
- Unrestricted motion vectors
- 4MV technique for better compression.

patni

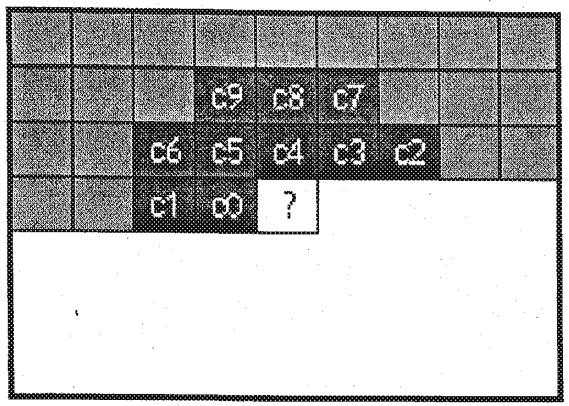
Shape Coding



- In case of constrained bandwidths, the viewer may choose to lose context information rather than have degraded quality for region of interest.
- Shape coding enables encoding of only region of interest by efficient means.

patni

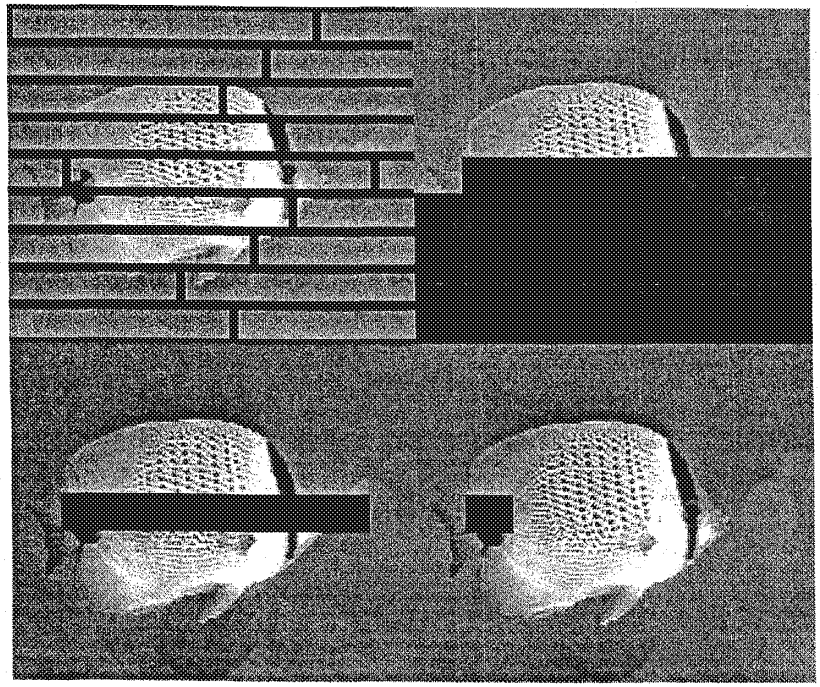
Context based arithmetic encoding



- Alpha plane information in raw form consumes lot of bandwidth.
- Context Based Arithmetic (CAE) algorithm is used for efficient coding of shape information.
- CAE uses probability tables to predict the alpha value of next pel from previous pels in context.
- Context and probability tables used for Intra and Inter frames are different.

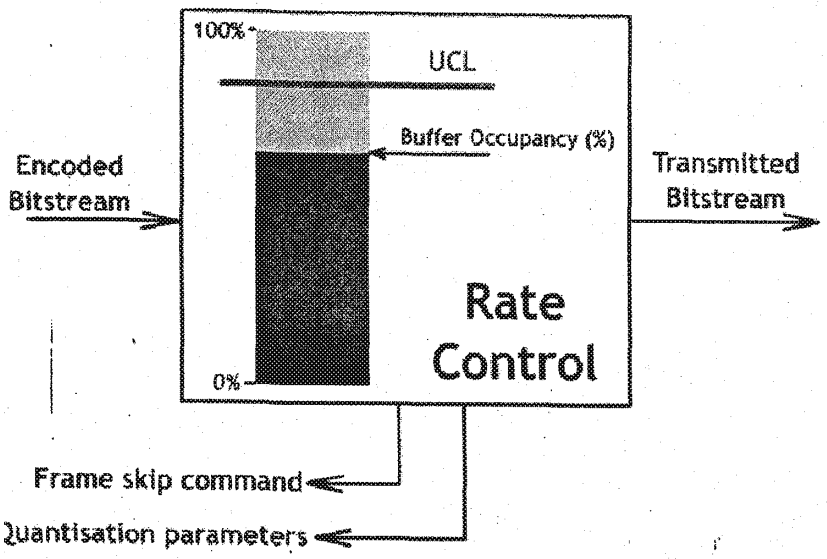
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1940
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Error Resilience Tool



- Error resilience enables to reduce propagation of error across frames or macroblocks within a frame.
- Techniques like Data partitioning and Reversible Variable length coding (RVLC) are employed for this purpose.

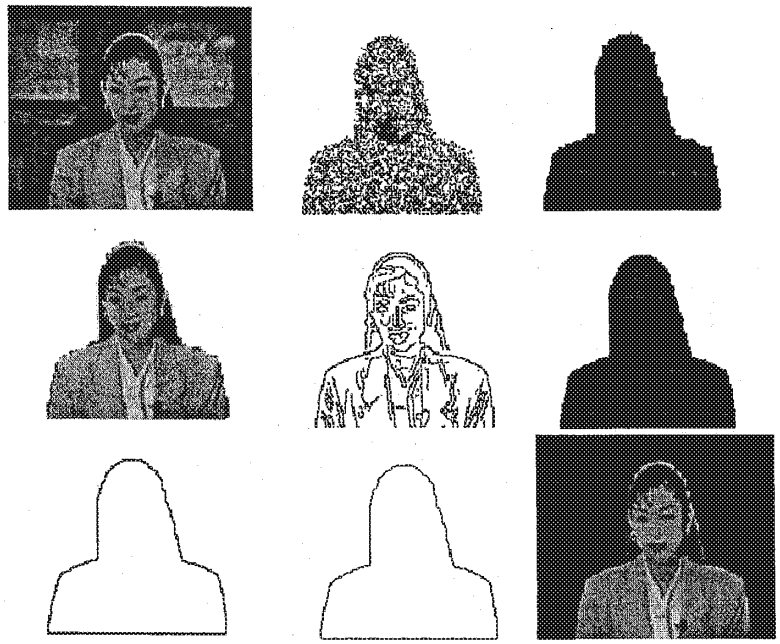
Rate Control



- Different channels have different bandwidth limitations.
- Rate control deals with maintaining the output bit-rate produced by the encoder in presence of rate variations.
- A control buffer is used for controlling the output rate.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1942
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Segmentation



- MPEG-4 does not specify how images should be segmented for shape coding.
- Image Segmentation is still an active area of research.
- Patni has developed a motion based-segmentation scheme for noisy as well as noiseless video.
- The algorithm is suited for Video Conferencing applications.

patni

Features of MPEG-4 Encoder

features

The following features are supported by MPEG-4 Video Encoder:

- Full compliance to ISO/IEC 14496-2 standard.
- Supports Simple Profile @ Level 3 & Core Profile @ Level 2.
- Support for QCIF/CIF encoding.
- Support for Error Resilience tool.
- Rate Control for bandwidth limited channels.
- Provides a Segmentation module that can be used for Video Conferencing/CCTV based applications.

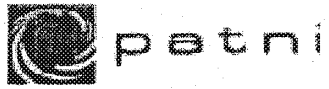
patni

Future Work

future plans

The following are future plans for MPEG-4 Video Encoder:

- Implementation of MPEG-4 Encoder for TI DSP processor.
- Implementation of Core Scalable profile.
- Using Joint Source Channel Coding with MPEG-4 Core Scalable Profile.
- Optimizing segmentation module for diverse applications.



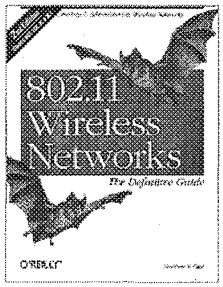
O'REILLY

Home | Community | Store | Safari Books Online | Conferences | School of Technology | About

Complete List | Bestsellers | New Releases | Upcoming Titles | By Publisher | By Series | Out of Print | Order Info

Search Search Tips

Browse Book



See larger cover

- Table of Contents
- Index
- Colophon
- Register Your Book
- View/Submit Errata
- View/Submit Review

Got a Question?

Do you have a question about **O'Reilly's products and services**? Share an idea! Report a problem...

Continue

Service and support by Satisfaction

802.11 Wireless Networks: The Definitive Guide, Second Edition

By Matthew Gast
April 2005
Pages: 654
ISBN 10: 0-596-10052-3 | ISBN 13: 9780596100520
(Average of 1 Customer Reviews)

Description

If you want to deploy your own wireless network--at home or at the office you must first understand the capabilities and risks associated with the 802.11 protocols. And *802.11 Wireless Networks: The Definitive Guide*, 2nd Edition is the perfect place to start. This updated edition covers everything you need to know about integrating wireless technology into your current infrastructure.

Full Description

BUY THIS BOOK

Print Book $44.95 Add to Cart

Print+PDF $58.44 Add to Cart

PDF $35.99 Add to Cart

Safari Books Online Read it Now

What is this?

Print Book £31.95 Add to UK Cart

What is this?

Looking to **Reprint** or **License** this content?

Tell a friend

Customers interested in this book were also interested in:

802.11 Security
Potter, Fleck, 1st Edition $34.95

Jeff Duntemanns Wi-Fi Guide
Duntemann, 2nd Edition $34.99

Wireless Hacks
Flickenger, Weeks, 2nd Edition $24.95

The Book of Wireless
Ross, 1st Edition $29.95

Jeff Duntemanns Drive-By Wi-Fi Guide
Duntemann, 1st Edition $29.99

Linux Unwired
Weeks, Dumbill, Jepson, 1st Edition $24.95

Featured customer reviews

Write a review

Great Book, *March 31 2008*
Rating:
Submitted by Amelotti [Respond | View]

great book, covers from the basic to the complex carachteristics of wireless networks. From the instalation (including site survey) to wireless network teory.

Read all reviews

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1946
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Media reviews

"Gast's comprehensive coverage of the 802.11 standard is definitly worthy of the title 'definitive guide'. The reading is fairly fast paced and written in such a way that readers can skip to certain chapters and gain insight on the high level of the protocol with previous knowledge of the lower chapters...If you are curious about 802.11, or want a deeper understanding of the standard I can think of no other better place to start than with this book."
-- Duff, member of Network Information Management and Security Group, Amazon.com

"This book is loaded with networking insights, explanations, security considerations, operations, radio transmission considerations, addressing and definitions right down to network timing synchronization--all logically and clearly presented. For those interested in networking and how it really works in depth, this is the book."
--Les J. Kizer, Greater Tampa Bay PC User Group, July 2005

"While Mathew S. Gast currently works at Trapeze Networks, he also spent years digging ever deeper into the esoterica of Wireless networking across all platforms and has become a de facto expert on the topic. The 2nd Edition of the book was written three years after the first one came out and covers areas previously unknown or even perhaps unimagined in the area of wireless networking... If you want to drink deeply of the Wonderful World of Wireless Kool-aid, this book is not just a great book to start, but to end with. After all, it really is a 'Definitive Guide,' working with real-world scenarios."
--Robert Pritchett, MacCompanion, July 2005

"If you're in charge of laying out a wireless network (or you just have to have a book in hand), this has a lot of great information and I'd recommend it."
--Eric Weuhler, Amazon.com review, June 2005

Read all reviews

O'REILLY®

© 2008, O'Reilly Media, Inc.
(707) 827-7000 / (800) 998-9938
All trademarks and registered trademarks appearing on oreilly.com are the property of their respective owners.

About O'Reilly
Privacy Policy
Contacts
Customer Service
Authors
Press Room
Jobs
User Groups
Academic Solutions
Newsletters
Writing for O'Reilly
RSS Feeds

Other O'Reilly Sites
O'Reilly Radar
Ignite
Tools of Change for Publishing
Digital Media
Inside iPhone
O'Reilly FYI
makezine.com
craftzine.com
hackszine.com
perl.com
xml.com

Sponsored Sites
Inside Aperture
Inside Lightroom
Inside Port 25
InsideRIA
java.net

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1947
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Tunneling QuickTime RTSP and RTP over HTTP

QuickTime 4.1 adds HTTP to its' Streaming Transport capabilities. The addition of HTTP streaming allows QuickTime to utilize HTTP (RFC 1945 Hypertext Transfer Protocol 1.0 and RFC 2068 1945 Hypertext Transfer Protocol 1.1) proxies so viewers behind a firewall can access QuickTime presentations.

The HTTP transport is built from two separate HTTP GET and POST requests initiated by the client. The server then binds the connections to form a virtual full-duplex connection.
This paper documents the protocol used forming this type of connection, and notes implementation details associated with its use.

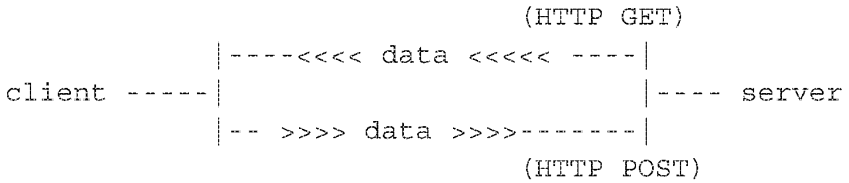
Protocol Requirements

- It must support use of unmodified RTSP/RTP.
- It must be acceptable to HTTP proxies.
- It must identify the requests and replies as non-cacheable by a proxy.
- All requests must originate from the client.
- It must uniquely identify the request pairs so that they may be successfully bound by the server.
- All requests must be made in such a way that they connect to the same RTSP server. This can be a challenge when the server is behind a DNS round robin for example.
- Any request must be identifiable as one that will eventually tunnel an RTSP conversation and RTP data.

Relationship to RTSP/RTP

Using standard RTSP/RTP it is possible to stream a presentation to a user via a single TCP connection. (See RFC 2036 "Real Time Streaming Protocol (RTSP)", section 10.12) Unfortunately, that is not sufficient to reach a significant population of Internet users. These users are typically on private IP networks where the client machines have indirect access to the public Internet via email and HTTP Proxies.
The QuickTime HTTP transport exploits the capability of HTTP GET and POST methods to carry an indefinite amount of data in their reply, and message body respectively. In the most simple case, the client makes a HTTP GET request to the streaming server to open the server to client channel. Then the client makes a POST request to the server to open the client to server channel.

© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.



The virtual connection makes it possible to use unmodified RTSP over the connection.

HTTP Requirements

Version - All requests are made using HTTP version 1.0.

Binding the Channels - Each client HTTP request must include a "x-sessioncookie" header with a Globally Unique Identifier (GUID) as it's value. This makes it possible for the server to unambiguously bind the 2 channels. This protocol uses the value as a simple opaque token passed to the Clib 'strcmp' function.

Caching - Server replies include "Cache-Control: no-store", and client requests include the "Pragma: no-cache" as directives in the headers. These are not required by the protocol. It is recommended that implementations

x-server-ip-address Header - The server may optionally include a "x-server-ip-address" followed by an IP address in dotted quad format in it's reply to the client's initial GET request. When this header is present, the client must direct its' POST request to the stated IP address, regardless of the IP address returned via DNS lookup.

This insures that both requests will connect to the same server among potentially several servers in a server farm. Many times these server clusters will be allocated

© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.

connections via a round robin DNS or other load balancing algorithms.

Meaning of HTTP errors in replies - HTTP errors reflect the unwillingness or inability of the server to form the virtual full duplex connection. They do not imply RTSP errors. In general, the server will always reply with "200 OK", and simply close the connection if problems occur. The POST request is never replied to by the server

Request Identification

QuickTime Streaming uses the 'application/x-rtsp-tunnelled' MIME type in both the Content-Type and Accept headers. This reflects the data type that is expected and delivered by the client and server.

RTSP Request Encoding

The RTSP requests made by the client on the POST channel must be encoded using the base64 method. (See RFC 2045 "Internet Message Bodies", section 6.8 Base64 Content-Transfer-Encoding; and RFC 1421 "Privacy Enhancement for Electronic Mail," section 4.3.2.4, Printable Encoding)
The base64 encoding prevents HTTP proxies from mistaking the embodied RTSP requests as malformed HTTP requests. Our testing revealed that some HTTP proxies did in fact fail to pass the raw RTSP data properly.

Encoding a DESCRIBE request.

Original request:

```
DESCRIBE rtsp://tuckru.apple.com/sw.mov RTSP/1.0
CSeq: 1
Accept: application/sdp
Bandwidth: 1500000
Accept-Language: en-US
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac 8.6)
```


© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.

Encoded format:

```
REVTQ1JJQkUgcnRzcDovL3R1Y2tydS5hcHBsZS5jb20vc3cubW92IFJUU1AvMS4w
DQpDU2VxOiAxDQpBY2NlcHQ6IGFwcGxpY2F0aW9uL3NkcA0KQmFuZHdpZHRoOiAx
NTAwMDAwDQpBY2NlcHQtTGFuZ3VhZ2U6IGVuLVVTDQpVc2VyLUFnZW50OiBRVFMg
KHF0dmVyPTQuMTtjcHU9UFBDO29zPU1hYyA4LjYpDQoNCg==
```

Sample HTTP Requests

Sample GET transaction.

Client ‡ Server

```
GET /sw.mov HTTP/1.0
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac8.6)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Accept: application/x-rtsp-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

Server ‡ Client

```
HTTP/1.0 200 OK
Server: QTSS/2.0 [v101] MacOSX
Connection: close
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/x-rtsp-tunnelled
```

Sample POST request

Client ‡ Server

```
POST /sw.mov HTTP/1.0
User-Agent: QTS (qtver=4.1;cpu=PPC;os=Mac8.6)
x-sessioncookie: tD9hKgAAfB8ABCftAAAAAw
Content-Type: application/x-rtsp-tunnelled
```


© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.

```
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

Server ‡ Client

There is no reponse from the server! The client will continue to send RTSP as the message body of this POST request. The chosen Content-Length of 32767 is an arbitrarily large value. All POST requests are required to have a content-length header by HTTP. In practice the actual value seems to be ignored by proxy servers, so it is possible to send more than this amount of data in the form of RTSP requests. The QuickTime Server ignores the content-length header.

Connection Maintenance

The client may at its' option close the POST connection at any given time. Doing so frees socket and memory resources at the server that might otherwise be unused for a long time. In QuickTime Streaming the best time for this usually occurs after the PLAY request.

Implementation Best Practices

This section covers details that are not required by the protocol, but communicate important information to proxies so that they handle the streams better.

Date Header – In QuickTime streaming, the GET reply includes a "Date" headers with a value at an arbitrary point in the past. This keeps proxies from caching the transaction.

Expires Header – Same as the Date header.

Pragma "No cache" - Tells many HTTP 1.0 proxies not to cache.

Cache-Control "no-cache" – Tells HTTP 1.1 proxies not to cache.

© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.

Unsupported HTTP

Support for HTTP features not documented here is not required for RTSP tunneling. HTTP tunnel should mimic a normal TCP connection as closely as possible without adding unnecessary features.

© 1999 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, Macintosh, PowerBook, Power Macintosh, and QuickTime are trademarks of Apple Computer, Inc., registered in the United States and other countries. iBook, iMac, and Power Mac are trademarks of Apple Computer, Inc. All other product names are trademarks or registered trademarks of their respective holders.

MY RESULTS
MY SUMMARY
ABOUT | FAQ | CONTACT |
Turbo-Charge
MHz
Free scan
Click Here for a Free Scan
Select a pyramid to begin the test.
= Test Using Recommended Server
Internet T1 $349
Free install & Free Router
Full 1.5Mbps T1. No Rebate.
www.isomedia.com
Bandwidth Monitoring Tool
Free bandwidth monitoring and reporting tool using NetFlow.
www.netflowanalyzer.com
Select A Region
What is Speedtest.net?
Speedtest.net is a free broadband speed test with servers located all over the world. Compare your results with others and easily share them.
Learn More
Link to Speedtest.net
Global Stats
How does your ISP perform compared to others?
Analyze broadband speeds around the world.
Moving? See which ISPs rank best at your destination!
About | FAQ | Contact | Host | Link | Speedtest.net Mini
© 2008 Ookla. All Rights Reserved.
License A Speed Test

When Is 54 Not Equal to 54? A Look at 802.11a, b, and g Throughput

Listen
Print
Discuss
Subscribe to Wireless
Subscribe to Newsletters

by Matthew Gast, author of 802.11 Wireless Networks: The Definitive Guide
08/08/2003 *(updated: 8/14/2003)*

Now that the 802.11g standard has been finalized, comparisons with the other standards in the 802.11 family are inevitable. One conclusion that is frequently drawn is that 802.11g offers similar speeds to 802.11a. After all, both products are advertised as having a data rate of 54 Mbps.

This article develops a simple model for the maximum TCP throughput of 802.11 networks so that a comparison can move beyond a simple comparison of nominal bit rates. According to the model, 802.11g is significantly faster than 802.11b. In a network consisting only of 802.11g clients, it is even slightly faster than 802.11a. However, "protection" mechanisms added to 802.11g to ensure backwards compatibility with legacy 802.11b clients can cut the throughput by 50 percent or more.

Technology	Transactions per second	Mbps of TCP payload throughput	Transactional speed relative to 802.11b
11b, 11 Mbps	479	5.6	1.0
11a, 54 Mbps	2,336	27.3	4.9
11g, 54 Mbps/no protection	2,336	27.3	4.9
11g, 54 Mbps/CTS-to-self protection	1,113	13.0	2.3
11g, 54 Mbps/RTS/CTS protection	750	8.8	1.6

The Transactional Model

The basic transactional model assumed by this article is a maximum-length frame containing a single TCP segment followed by a TCP acknowledgement. To cope with the inherent unreliability of radio waves, the 802.11 MAC requires positive acknowledgement of every transmission. Each TCP packet must therefore be wrapped up in a frame exchange. The complete transaction consists of the two 802.11 data frames carrying TCP embedded in their respective exchanges:

The TCP data segment:

- Distributed Interframe Space (DIFS): this interframe space indicates that an exchange has completed, and it is safe to access the medium again.
- The data frame containing the TCP segment.
- A Short Interframe Space (SIFS), which is a small gap between the data frame and its acknowledgement.
- The 802.11 ACK frame.

The TCP ACK:

- DIFS
- The data frame containing the TCP ACK
- SIFS
- 802.11 ACK frame

For this model, I will assume that the host operating system limits the outgoing frame size to 1,500 bytes. 802.11 permits much larger frame sizes, but this flexibility has not traditionally been used by client products. Most access points connect to existing networks with Ethernet, and therefore limit the payload size to the maximum Ethernet payload size. (In fact, this simple precaution is required to obtain Wi-Fi certification.)

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1955
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Encapsulation

In addition to the payload data, there are 36 additional bytes of data added in the encapsulation process. The 802.11 MAC header adds 28 bytes of data for various control and management functions, error detection, and addressing. A further eight bytes are added by the SNAP encapsulation header to identify the network layer protocol. (For further encapsulation details, see Figures 3-9 and 3-13 of *802.11 Wireless Networks*.) The total size of the MAC payload for the TCP data segment is 1,536 bytes (12,288 bits). For the TCP ACK, the 40 bytes of TCP/IP header information also gets a further 36 bytes of SNAP header for a total of 76 bytes (608 bits).

Throughput

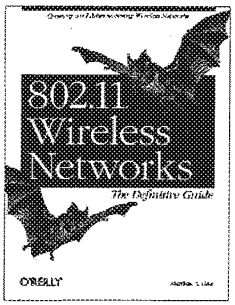
This article measures the throughput of the network at the TCP payload layer. Each transaction delivers one data frame. There are 1,500 bytes of Ethernet payload, but 40 bytes must be subtracted for headers. By adding up the total time required for each component of the transaction, a transaction rate can be derived. For the throughput, simply multiply by 1,460 bytes.

Weaknesses of the Model

The transactional model is quite simple, and neglects several important effects. First of all, it assumes a steady stream of well-ordered frames with no contention for the medium. 802.11 implements collision avoidance and exponential backoff, so in reality, the time between frame exchanges will be longer than one DIFS. Exponential backoff in the presence of contention will further decrease throughput. My initial back-of-the envelope estimate is that contention for the medium would reduce the maximum throughput figures above by 25% to 50%, depending on the exact assumptions made.

Furthermore, there is always the possibility of simultaneous transmission. 802.11 will order layer two frames and prevent collisions, but higher-layer packets may be transmitted simultaneously. 802.11 treats TCP segments and acknowledgements as unrelated higher-layer frames, so a certain number of collisions are expected. (One engineer has told me that up to 10% retransmissions should be expected for TCP/IP on 802.11 due to this effect.)

Related Reading



802.11 Wireless Networks: The Definitive Guide
Creating and Administering Wireless Networks
By Matthew Gast

Table of Contents
Index
Sample Chapter

Read Online--Safari
Search this book on Safari:

Go
Only This Book
Code Fragments only

Furthermore, the model ignores the sophistication in the TCP acknowledgement model. To avoid constraining throughput, TCP uses "sliding windows" and allows multiple outstanding frames to be transmitted before acknowledgement. In practice, TCP acknowledgements can apply to multiple segments, so this model overstates the impact of higher-layer protocol acknowledgements.

The Baseline: 802.11b at 11 Mbps

The baseline speed comes from 802.11b. It is not as fast as the newer specifications, but it's what most of the 802.11 cards out in the world do. I'll do the calculation for 802.11b first, and then compare everything else to the result for 802.11b.

First off, the basic timing numbers for 802.11b:

- SIFS = 10 µs
- Slot time = 20 µs
- DIFS = 2 x Slot time + SIFS = 50 µs

802.11b requires that a preamble be prepended to every frame before it is transmitted to the air. That preamble may be either the traditional "long" preamble, which requires 192 µs for transmission, or it may be an optional "short" preamble that requires only 96 µs. (For a diagram of the physical layer headers in 802.11b, see Figure 10-26 in *802.11 Wireless Networks*.) Support of the long preamble is mandatory, and is the default setting on most devices. In the interest of reducing the number of calculations, this article will perform calculations using only the long preamble.

802.11b running at the max speed divides data up into 8-bit symbols. There are 1,536 8-bit blocks in the TCP data segment, and 76 8-bit blocks in the TCP acknowledgement. The ACK does not have SNAP headers, and is only 14 bytes long.

Encoding the MAC frames is easy. 802.11b divides up the MAC frame into a series of 8-bit "symbols," and then transmits 1.375 million symbols per second. So add up the individual components of the transaction to get the total duration:

	TCP data	TCP ACK
DIFS	50 µs	50 µs
802.11 Data	192 µs + 1536/1.375 Msps = 192 µs + 1,118 µs = 1,310 µs	192 µs + 76/1.375 Msps = 192 µs + 56 µs = 248 µs
SIFS	10 µs	10 µs
	192 µs + 14/1.375 Msps	

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1956
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

802.11 ACK	= 192 µs + 11 µs = 203 µs	= 203 µs
Frame exchange total	1,573 µs	511 µs
Transaction Total	2,084 µs	

Each transaction requires 2,084 µs. At that duration, 479 exchanges can complete per second. With a TCP payload of 1,460 bytes per exchange, the throughput is 5.7 Mbps.

802.11a

802.11a is faster than 802.11b for two reasons: timing relationships between frames in the exchanges are tighter, and the encoding used by 802.11a does not require such long preambles for synchronization.

Here are the basic timing numbers for 802.11a:

- SIFS = 16 µs
- Slot time = 9 µs
- DIFS = 2 x Slot time + SIFS = 34 µs

Like 802.11b, 802.11a divides data up into a series of symbols for transmission. However, the encoding used by 802.11a uses much larger symbols. At 54 Mbps, each symbol encodes 216 bits. (For a full listing of encoding block sizes, see Table 11-3 in *802.11 Wireless Networks*.) The OFDM encoding used by 802.11a adds six bits for encoding purposes to the end of the frame, so our maximum-length frame of 1,536 bytes becomes a series of 12,288 data bits plus the six tail bits. The total bit string of 12,294 bits can be encoded in 57 symbols. The 608 bit TCP ACK plus tail bits requires four symbols, and the 802.11 ACK also requires just one symbol.

Each frame is prepared for transmission in the air with a 20 µs header to synchronize the receiver. Following the 20 µs header is a series of symbols, each requiring 4 µs for transmission.

	TCP data	TCP ACK
DIFS	34 µs	34 µs
802.11 Data	20 µs + 57 * 4 µs/symbol = 20 µs + 228 µs = 248 µs	20 µs + 3 * 4 µs/symbol = 20 + 12 µs = 32 µs
SIFS	16 µs	16 µs
802.11 ACK	20 µs + 1 * 4 µs/symbol = 20 µs + 4 µs = 24 µs	= 24 µs
Frame exchange total	322 µs	106 µs
Transaction Total	428 µs	

At 428 µs per transaction, it is possible to complete 2,336 exchanges per second. That corresponds to a throughput of 27.3 Mbps.

Pages: **1**, 2

Next Page

Search Wireless

Go

Tagged Articles

Post to del.icio.us

This article has been tagged: *wireless*

Articles that share the tag *wireless*:

Wireless Mesh Networking (72 tags)

Protect Yourself from WiFi Snoops (23 tags)

Secure Your Wireless with IPsec (20 tags)

Mapping the 802.11 Protocol (8 tags)

Top Ten 802.11 Myths of 2005 (8 tags)

View All

Sponsored Resources

- Inside Lightroom

Related to this Article

Take Control of Your Wi-Fi Security
October 2007
$10.00 USD

web2.0 EXPO Web 2.0 Expo
April 22-25, 2008
San Francisco, CA

advertisement

CodeSmith Code Generator
Template-driven. Easy to use. Your code. Your way. Faster!
CodeSmithTools.com/FreeDownload
Ads by Google

O'REILLY®
© 2008, O'Reilly Media, Inc.
(707) 827-7000 / (800) 998-9938
All trademarks and registered trademarks appearing on oreilly.com are the property of their respective owners.

About O'Reilly
Privacy Policy
Contacts
Customer Service
Authors
Press Room
Jobs
User Groups
Academic Solutions
Newsletters
Writing for

Other O'Reilly Sites
O'Reilly Radar
Ignite
Tools of Change for Publishing
Digital Media
Inside iPhone
O'Reilly FYI
makezine.com
craftzine.com
hackszine.com
perl.com

Sponsored Sites
Inside Aperture
Inside Lightroom
Inside Port 25
InsideRIA
java.net

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1957
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

O'Reilly xml.com
RSS Feeds

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1958
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHITE PAPER

IEEE 802.11g

The New Mainstream Wireless LAN Standard

Today's popular IEEE 802.11b (802.11b) for wireless local area networks (WLANs) has changed the face of networking, providing tremendous flexibility for mobile workers and hassle-free Internet sharing for home users. The next milestone in the evolution of WLANs is the introduction of IEEE 802.11g (802.11g). This new IEEE standard will dramatically improve the performance of WLANs, while providing compatibility with the existing installed base of 802.11b networks. The purpose of this white paper is to:

- Provide an explanation of existing WLAN standards
- Describe 802.11g specification requirements
- Set expectations for 802.11g performance
- Discuss 54g™, the high performance implementation of the 802.11g standard

WLAN Market

The WLAN market has grown rapidly as wireless technology has evolved to meet fundamental needs of businesses and technology consumers alike:

- Corporate IT departments deploy WLANs to support roving employees, reduce the cost of cabling and recabling the physical plant, and provide a rapid response to changes in demand.
- The mobile worker uses public access WLANs in cafes, airports, trains, lobbies, and remote corporate sites to connect to the corporate network or Internet.
- Small offices and enterprises use WLANs to share Internet connections, printers and peripherals, and create backup connectivity solutions while remaining highly scalable and flexible, without the costs of cabling.
- The home customer uses a WLAN to share a broadband Internet connection among multiple family PCs without drilling holes and installing cable throughout the home.

These customers have adopted WLAN solutions to achieve flexibility, connectivity, mobility, and low-cost deployments not available through conventional wired solutions. The numbers speak for themselves—over 35 million WLAN nodes have already shipped and the number of nodes is expected to grow to 100 million in 2005.

TABLE OF CONTENTS

WLAN Standards ..... Page 2
802.11g Performance and Characteristics ..... Page 3
Network Environment Considerations ..... Page 5
Which Standard Should I Deploy? ..... Page 9
What is 54g™? ..... Page 11
Conclusion ..... Page 12

802.11g-WP104-R
www.54g.org

16215 Alton Parkway • P.O. Box 57013 • Irvine, CA 92619-7013 • Phone: 949-450-8700 • Fax: 949-450-8710

07/02/03

WLAN STANDARDS

The foundation of mainstream WLAN products began with the original 802.11 standard developed in 1997 by the Institute of Electrical and Electronics Engineers (IEEE). That base standard continues to be enhanced through document additions that are designated by a letter following the 802.11 name, such as 802.11b, 802.11a, or 802.11g. The letter suffix represents the task group that defines the extension to the standard. These enhancements bring increases in data rate and functionality leading to rapid progression of the WLAN market. The following table briefly summarizes the enhancements related to data rate (the WLAN physical layer). More information is available at http://grouper.ieee.org/groups/802/11.

Table 1: IEEE 802.11 Specifications

	802.11b	*802.11a*	*802.11g*
Standard approved	July 1999	July 1999	June 2003
Maximum data rate	11 Mbps	54 Mbps	54 Mbps
Modulation	CCK	OFDM	OFDM and CCK
Data rates	1, 2, 5.5, 11 Mbps	6, 9, 12, 18, 24, 36, 48, 54 Mbps	CCK: 1, 2, 5.5, 11 OFDM: 6, 9, 12, 18, 24, 36, 48, 54 Mbps
Frequencies	2.4–2.497 GHz	5.15–5.35 GHz 5.425–5.675 GHz 5.725–5.875 GHz	2.4–2.497 GHz

IEEE 802.11b

Ratified by the IEEE in July 1999, 802.11b extends the original IEEE 802.11 direct sequence spread spectrum (DSSS) standard to operate up to 11 Mbps in the 2.4-GHz unlicensed spectrum using complementary code keying (CCK) modulation. The four data rates of 1, 2, 5.5, and 11 Mbps are specified on up to three non-overlapping channels, and the lowest two rates are also allowed on up to 13 overlapping channels.

IEEE 802.11a

Ratified by the IEEE at the same time as 802.11b, the IEEE 802.11a (802.11a) standard operates in the 5-GHz spectrum. The 802.11a standard was designed for higher bandwidth applications than 802.11b, and includes data rates of 6, 9, 12, 18, 24, 36, 48, 54 Mbps using orthogonal frequency division multiplexing (OFDM) modulation on up to 12 discrete channels.

IEEE 802.11g

In July 1999, the 802.11g subcommittee was tasked to extend the 2.4-GHz unlicensed spectrum to data rates faster than 20 Mbps. The resulting 802.11g standard was ratified in June 2003. The 802.11g standard provides optional data rates of up to 54 Mbps, and requires backward compatibility with 802.11b devices to protect the substantial investments in today's WLAN installations.

The 802.11g standard includes mandatory and optional components. It specifies OFDM (the same technology used in 802.11a) and CCK as the mandatory modulation schemes with 24 Mbps as the maximum mandatory data rate, but it also provides for optional higher data rates of 36, 48, and 54 Mbps.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1960
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Market Acceptance of WLAN Standards

802.11b

The industry standard for WLANs has been 802.11b. Products based on 802.11b gained mainstream acceptance as the first wireless networking products with acceptable speeds, affordable prices, and universal compatibility as certified by the Wi-Fi Alliance. More than 95% of today's WLAN infrastructure includes 802.11b products.

802.11a

The first 802.11a based products became available late 2001. These products provide data rates of up to 54 Mbps in the 5-GHz frequency band. Even though the technology delivers greater data rates, it has had limited market acceptance. This has been due primarily to its lack of backward compatibility with 802.11b products, shorter connectivity range, and higher deployment costs.

802.11g

The next mainstream wireless LAN standard is 802.11g. This technology satisfies the bandwidth needs of the market globally and economically, while remaining compatible with the installed base of mainstream products.

802.11g Approval and Certification

The IEEE 802.11g standard was approved in June 2003. The IEEE standards body, however, does not validate interoperability or compliance with the standard; an industry association (Wi-Fi Alliance) performs these tasks. The Wi-Fi Alliance is already building a testing program to certify 802.11g interoperability based on the 802.11g standard. The Wi-Fi® certification for 802.11g products is expected in the summer of 2003. The 802.11g certification program follows the Wi-Fi Alliance's successful 802.11b and 802.11a certification programs.

Products conforming to the draft 802.11g standard have been available since late 2002. Products shipped prior to June may be upgraded via software download to ensure standards compliance.

802.11g Performance and Characteristics

Data rate, range, throughput, and compatibility vary among the three WLAN standards. These variations are caused by differences in frequency, modulation schemes, and number of data rates.

802.11g Data Rates

The 802.11 standard technologies all support multiple data rates to allow clients to communicate at the best possible speed. Data rate selection is a tradeoff between obtaining the highest possible data rate while trying to minimize the number of communication errors. Whenever there is an error in the data, the systems must spend time to retransmit the data until it is error free. Each 802.11 client performs a procedure to select the best data rate. The 802.11g clients can select from the widest possible range of both OFDM data rates of 54, 48, 36, 24, 18, 12, 9, and 6 Mbps, and the CCK rates of 11, 5.5, 2, and 1 Mbps.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1961
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

802.11g Range and Data Rate

As distance from the access point increases, 802.11 based products provide reduced data rates to maintain connectivity. The 802.11g standard has the same propagation characteristic as 802.11b, because it transmits in the identical 2.4-GHz frequency band. Because 802.11b and 802.11g products share the same propagation characteristics, implementations provide roughly the same maximum range at the same data rate. Because 5-GHz radio signals do not propagate as well as 2.4-GHz radio signals, the 802.11a product range is limited compared to the 802.11b or 802.11g product range.

The following figure illustrates the expected data rate of each technology at different ranges.

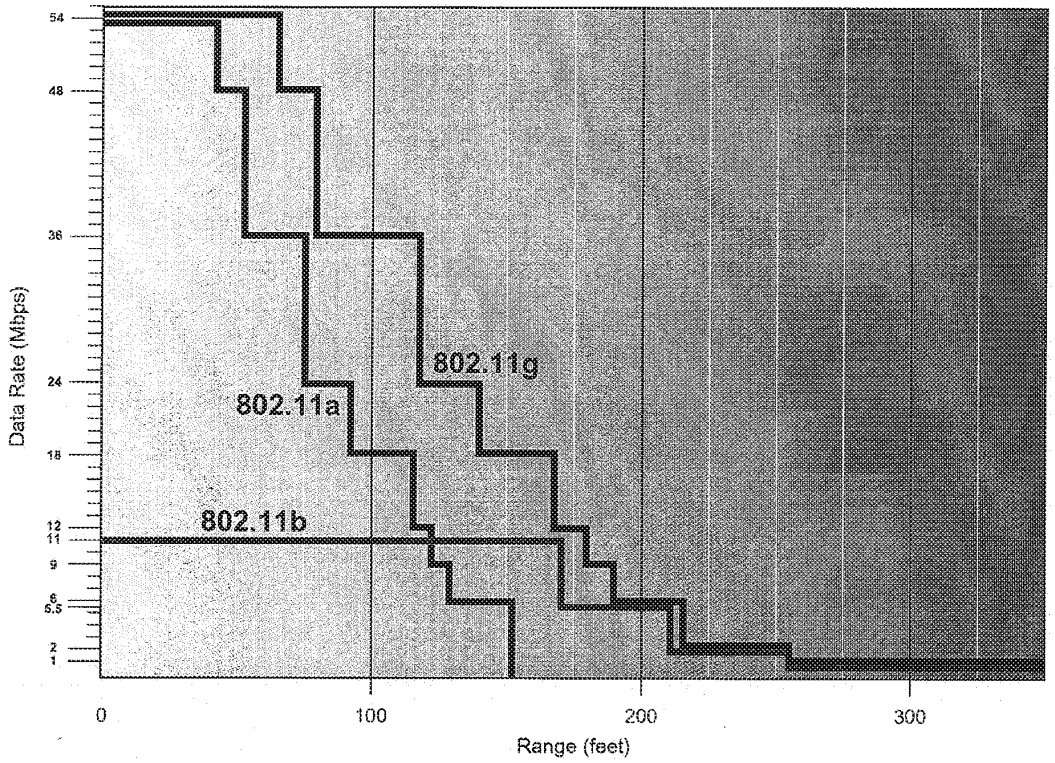


Figure 2: Expected 802.11a, 802.11b, and 802.11g Data Rates at Varying Distance from Access Point

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1962
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Typically, 2.4-GHz 802.11g networks have the same coverage as 2.4-GHz 802.11b networks. The 802.11b standard uses CCK modulation, whereas 802.11g uses both CCK modulation for backward compatibility and OFDM modulation to achieve better throughput at a given distance. The 802.11a standard also uses OFDM modulation, but there is more signal loss as it travels through objects because it uses a higher frequency.

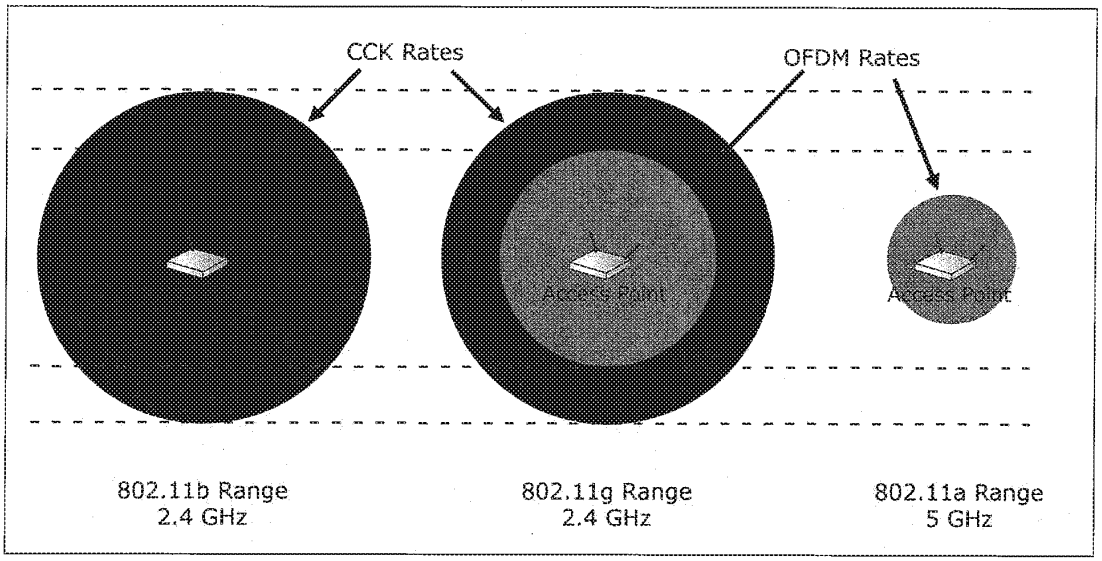


Figure 3: Relative Range of 802.11b, 802.11g, and 802.11a Devices

Network Environment Considerations

One of the major benefits of the 802.11g standard is the mandate for 802.11g and 802.11b devices to communicate with each other. At all 802.11b rates, 802.11b devices communicate with 802.11g products as if they were 802.11b products. However, 802.11g products behave differently when using OFDM data rates if there are 802.11b devices in the network environment. Understanding this behavior is important in setting performance expectations (given the large installed base of 802.11b clients) and the planning of network capacity until 802.11b clients are replaced with 802.11g clients (the presence of an 802.11a 5-GHz network has no effect on these scenarios). The following scenarios show how the type of network environment is determined both by the network infrastructure and client devices.

802.11b-Only Legacy Environment

When the AP and all clients are 802.11b, communication occurs at 802.11b data rates. Overhead communication between the products effectively limits the maximum TCP throughput to 5.8 Mbps. When the products are communicating at greater distances and lower data rates, the throughput is correspondingly reduced.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1963
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

802.11g THROUGHPUT

Throughput is not the same as data rate for networking systems, because of overhead, environment, and network composition. The throughput of 802.11g products can depend on whether there are 802.11b products nearby. Performance is best in environments where an 802.11g access point (AP) is only communicating with 802.11g clients in a homogeneous WLAN. In these environments, the data rate within 75 feet is 54 Mbps and the throughput is 22–24 Mbps when using Transmission Control Protocol (TCP).

In the interest of maximizing performance in the presence of 802.11b products, the 802.11g APs coordinate the use of the transmission medium with *protection* mechanisms (see sidebar). Because the protection mechanisms require overhead communication, compatibility is provided at the expense of throughput. The CTS-to-self protection mechanism lowers the maximum TCP throughput to approximately 15 Mbps, as shown in the following table.

Table 2: Expected Maximum Throughput for IEEE 802.11 Environments

Distance (Feet)	*802.11b (Mbps)*	*802.11a (Mbps)*	*802.11g-only (Mbps)*	*802.11g Mixed Environment with CTS-to-self (Mbps)*	*802.11g Mixed Environment with RTS/CTS (Mbps)*
10	5.8	24.7	24.7	14.7	11.8
50	5.8	19.8	24.7	14.7	11.8
100	5.8	12.4	19.8	12.7	10.6
150	5.8	4.9	12.4	9.1	8.0
200	3.7	0	4.9	4.2	4.1
250	1.6	0	1.6	1.6	1.6
300	0.9	0	0.9	0.9	0.9

802.11g COMPATIBILITY

Because 802.11g uses the same radio signaling (CCK) as 802.11b at the lower four 802.11g data rates, it is fully backward compatible with 802.11b. This enables networks to continue supporting 802.11b enabled devices when migrating to the higher performance standard.

Protection Mechanism: Air Traffic Control

The 802.11g standard provides protection mechanisms for managing communication in a mixed 802.11b/g environment. The 802.11b radios do not hear when the airspace is busy with 802.11g OFDM signals. Protection mechanisms prevent 802.11b clients from transmitting after improperly assessing that the airspace is empty while 802.11g OFDM signals are being transmitted. The 802.11g products still communicate at the same 802.11g OFDM data rates when protection is in use, but a short 802.11b rate message signals to 802.11b products to not transmit for a specified duration because an 802.11g OFDM message is immediately following. The 802.11b protection messages cause signaling overhead and result in reduced throughput to the user.

The AP directs clients to use protection through a signaling mechanism specified within the 802.11g standard. Enterprise-class 802.11g APs may allow users to tune the protection mechanism algorithm to optimize network system performance (some 802.11g APs may allow the administrator to override the use of the protection mechanism for a performance improvement in light traffic networks).

The 802.11g standard allows 802.11g clients to use one of several protection mechanisms in a mixed 802.11b/g environment. The Wi-Fi Alliance will test for one of two signalling methods: RTS/CTS and CTS-to-self. Request to send (RTS) is analogous to a pilot's take off request to an air traffic control tower—the pilot waits to use the airspace until verifying with the control tower that the airspace is clear. The clear to send (CTS) message is like the clearance from the tower.

The CTS-to-self protection mechanism method sends a CTS message using an 802.11b rate to clear the air, and then immediately follows with data using an 802.11g date rate. The CTS-to-self protection mechanism provides a maximum TCP throughput of 14.7 Mbps.

With any of the protection mechanisms, 802.11g throughput is still greater than 802.11b throughput at the same distance.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1964
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

802.11g-Only

When the AP and all clients are 802.11g, communication occurs at the highest possible TCP throughput. The 802.11g AP detects that all of the clients are 802.11g and instructs the network not to use any protection method. Without a protection mechanism engaged, throughput of 24 Mbps or greater is possible.

802.11g AP, Mixed Clients

When the AP is 802.11g and there is a mixture of 802.11g clients and 802.11b clients, the AP senses both technologies on the network. The 802.11g AP instructs 802.11g clients to use a protection mechanism. Effectively, 802.11g clients function at reduced 802.11g TCP throughput (up to 15 Mbps), which is faster than the 802.11b client that communicates at a maximum throughput of up to 5.8 Mbps.

802.11b AP, 802.11g Client

When the AP is 802.11b and the client is 802.11g, the 802.11g client is able to successfully associate and communicate with the 802.11b AP. Communication between the AP and the 802.11g client uses CCK modulation and achieves typical 802.11b speeds. An 802.11g client can always function as an 802.11b client.

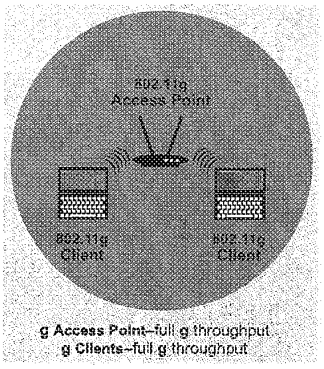


Figure 4: 802.11g-Only Environment

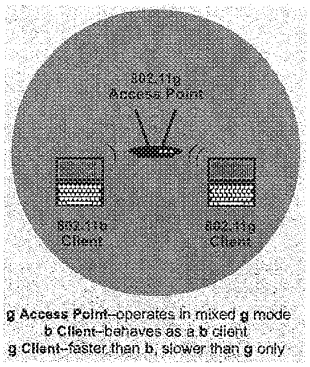


Figure 5: 802.11g AP, Mixed Client Environment

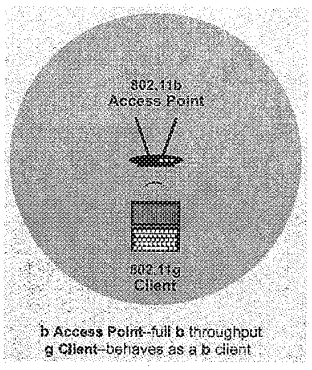


Figure 6: 802.11b AP, 802.11g Client Environment

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1965
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MULTIPLE 802.11g APS, MIXED CLIENTS

When there are multiple 802.11g APs and a single 802.11b client on the same channel, all overlapping 802.11g APs signal the use of the protection mechanism. Effectively, 802.11g clients function at a reduced 802.11g TCP throughput (up to 15 Mbps), which is faster than the 802.11b client, which communicates at a typical 802.11b TCP throughput (up to 5.8 Mbps).

The APs can also be configured to use different channels for their 802.11g clients so that the 802.11g-only networks do not need to use a protection mechanism. This allows 802.11g clients to have full TCP throughput as if they were in an 802.11g-only network.

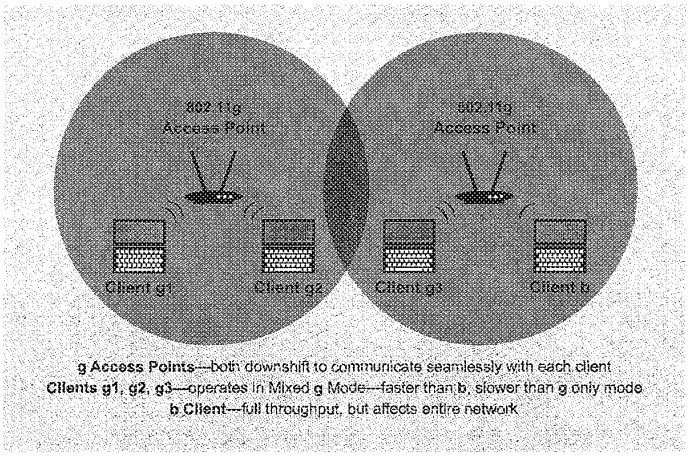


Figure 7: Multiple 802.11g APs, Mixed Clients

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1966
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Which Standard Should I Deploy?

With any network technology, there are tradeoffs between performance and cost. With wireless technologies, other factors such as range and capacity should also be considered. Table 3 summarizes typical user requirements and the characteristics of WLAN technology. In most cases, 802.11g provides the correct mix of characteristics for the different categories of user requirements. Upgrading to 802.11g is the easiest and least expensive choice, because it can be done gradually, without sacrificing the current WLAN infrastructure. The 802.11g APs automatically support existing 802.11b clients while providing increased speed to 802.11g clients.

Planning Network Migration

As network managers plan upgrades from existing wireless 802.11b networks, the two basic technologies to consider are 802.11g and 802.11a/g. An upgrade to 802.11g is a simple proposition. Users equipped with 802.11g-enabled notebooks will gain immediate access to the 802.11b network. These users will experience a throughput boost as APs are upgraded to 802.11g. As budget allows, dual-band 802.11a/g products can be deployed to meet the demand for more network capacity while remaining backward compatible.

The dual-band APs may need to be more closely spaced to ensure coverage for 802.11a clients. Transmit power control can be enabled in the 802.11g network to optimize performance of closely spaced access points. Because 802.11g is forecast to rapidly replace 802.11b, a dual-band a/b AP solution becomes immediately obsolete.

Table 3: WLAN User Requirements and Technology Characteristics in the U.S.

Typical WLAN User Requirements					
Type of WLAN	***Peak Speed***	***Capacity***	***Range***	***802.11b Compatible***	***Budget***
Enterprise	High	Variable	Variable	Yes	High
Public Access	Low	Medium	High	Yes	Medium
Small Business	Medium	Medium	High	Yes	Low
Home	Medium	Low	High	Yes	Low

WLAN Technology Characteristics					
WLAN Technology	***Peak Speed***	***Capacity***	***Range***	***802.11b Compatible***	***Cost***
802.11b	Medium	Low	High	Yes	Low
802.11a	High	High	Low	No	Medium
802.11g	High	Medium	High	Yes	Low
802.11a/g	High	High	High	Yes	High

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1967
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Enterprise

Enterprise WLANs are typically an overlay to a wired infrastructure serving the needs of mobile workers for e-mail, web browsing, and intranet access when they are away from their desks. Coverage is typically more important than connection rate. Enterprises can benefit from the higher bandwidth and backward compatibility of 802.11g networks today, and can outfit densely populated environments or plan for wired network replacement with dual-band 802.11a/g networks. It is expected that enterprise customers will future-proof their networks by providing 802.11g connections on the client side, and upgrade infrastructure as budget permits.

Public Access—Hotspots (Wi-Fi ZONE™)

Compatibility and range matter most to Hotspot operators and public access WLAN customers. Public WLAN Hotspots must be certain that their APs can communicate using mainstream standards that offer connectivity to the widest possible range of subscribers. Today, nearly all Hotspot traffic is 802.11b. Public access operators remain in the 2.4-GHz band because it offers twice the range and four times the coverage area of the 5-GHz alternative, and public access customers continue to use the 2.4-GHz spectrum to maximize connectivity. As Hotspots upgrade, they will add 802.11g to their networks to add range and preserve interoperability.

Small Business

A small business WLAN needs to cover small to medium-sized areas and support a variety of applications, such as e-mail, web browsing, and large file transfer. Because of good performance characteristics, costs, and range, 802.11g again makes the most sense: The 802.11g standard offers compelling value by minimizing the number of APs, lowering deployment costs, while future proofing the installation with a mainstream standard. Many small businesses may simply require a single 802.11g AP and no wired infrastructure at all.

Home

Many homeowners today have multiple computers and want to share a common Internet connection. For these uses, low-cost 802.11g or 802.11b solutions suffice. As the number of home network devices expand and bandwidth-hungry applications grow (such as gaming, home-wide audio and video streaming, and home security), an 802.11g WLAN provides up to five times the performance of an 802.11b WLAN. The 802.11g solutions provide the bandwidth for added devices and new applications, with a range that covers most homes with one or two APs.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1968
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS 54g™?

54g™ is the maximum performance implementation of the IEEE 802.11g standard and is fully compliant. 54g™ technology provides data rates of up to 54-Mbps with the highest throughput allowed by the specification and provides the industry's best range and latest security. The 54g™ logo can be found only on products that achieve this high level of performance (see Figure 8).

54g™—MAXIMUM PERFORMANCE 802.11g

The 54g™ implementation is designed to exceed the requirements of the IEEE 802.11g standard, as shown in Table 4. The 54 Mbps data rate and short slot turnaround time are optional in the standard but are included in 54g™. These features more than double the expected throughput of a product that just meets the standard. The receiver sensitivity of 54g™ products far exceeds the specification for maximum range, and transmit power control is included in 54g™ to enable optimized multiple AP deployments. All 54g™ products are WPA Certified and embed hardware-accelerated AES security, which future-proofs users in advance of the IEEE 802.11i (802.11i) standard due to be adopted next year.

Table 4: 54g™ Pushes the Performance Limits of 802.11g

	54g™ Specification	*802.11g Standard*
Highest mandatory data rate	54 Mbps	24 Mbps
Shortest mandatory slot time	9 µs	20 µs
Maximum receiver sensitivity	–92 dBm	–80 dBm
Transmit power control	Included	Not specified
Next generation AES security	Included	Not specified

LOOK FOR THE 54g™ LOGO

The 54g™ logo indicates that a product is a maximum-performance implementation of the 802.11g standard, is fully backward compatible with 802.11b products, and is compatible at maximum performance with any other product bearing the 54g™ logo.

It provides consumers with an easy way to find high-performance WLANs based on the 802.11g standard, and protects them from investing in 802.11g solutions where the highest data rate is less than 54 Mbps. Major wireless networking and PC vendors are shipping 54g™ products, ensuring a broad selection of compatible, high-performance products.

Figure 8: 54g™ Logo

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1969
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

CONCLUSION

The IEEE 802.11g standard will drive the next growth wave in wireless networking. Products built to the IEEE 802.11g standard are compelling because they:

- Provide a five-fold increase in WLAN speed over current networks
- Remain fully backward compatible with the popular IEEE 802.11b based products
- Offer better range and coverage than current networks

This new WLAN technology provides flexibility, connectivity, mobility, and affordability that is not available through conventional wired solutions.

54g™ is 802.11g implemented at the highest levels of the standard, and available today. 54g™ is fully backward compatible with 802.11b. 54g™ products used together provide the highest levels of speed, reach, and security for maximum performance.

Broadcom Corporation

16215 Alton Parkway
P.O. Box 57013
Irvine, CA 92619-7013
Phone: 949-450-8700
Fax: 949-450-8710
© 2003. All rights reserved
Printed in the U.S.A.

Broadcom®, the pulse logo, and 54g™ are trademarks of Broadcom Corporation and/or its subsidiaries in the United States and certain other countries.
Wi-Fi® and Wi-Fi ZONE™ are trademarks of the Wi-Fi Alliance. All other trademarks are the property of their respective owners.
Broadcom Corporation reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design.
Information furnished by Broadcom Corporation is believed to be accurate and reliable. However, Broadcom Corporation
does not assume any liability arising out of the application or use of this information, nor the application or use of any product or
circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1970
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

TI Developer Conference
August 6-8, 2002 • Houston, TX • Westin Galleria Hotel
Connecting Real People with Real Solutions

ingenient technologies

Video Transrating and Transcoding

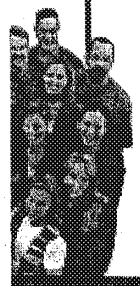
Overview of Video Transrating and Transcoding Technologies

Darek Blasiak, Ph.D.
Engineering Director of Video Technologies
Ingenient Technologies
www.ingenient.com

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1971
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Agenda

- Transrating and Transcoding
  - Definitions & Applications
  - Recoding Generation Loss
  - Types of Transcoding
  - Design Issues
- Statistical Multiplexing
- Implementation on Texas Instruments Programmable DSP's
- Conclusions/Questions



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1972
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Learning Objectives

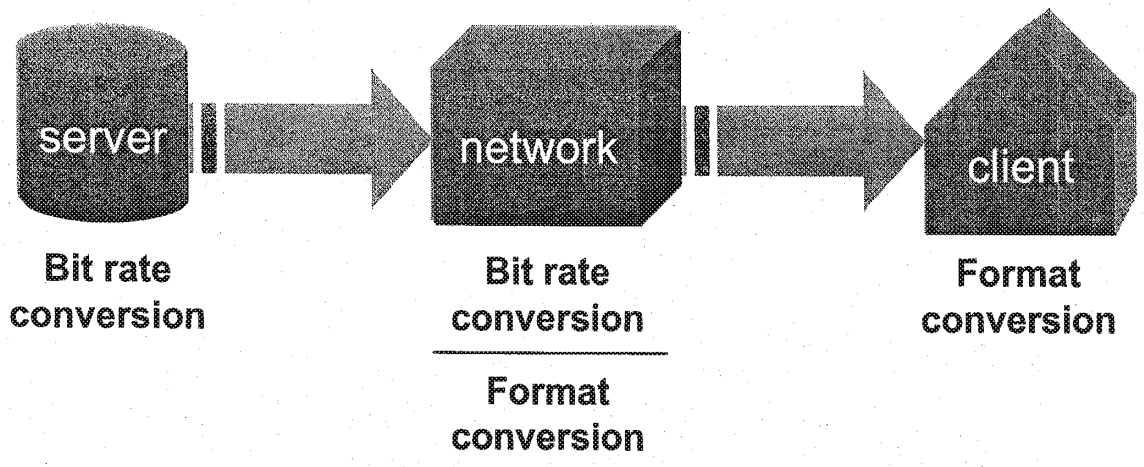
- **Video Transrating and Transcoding**
  - What is it?
  - What is the technology used for?
  - What are the design issues?
  - What are the solution strategies?
  - What is Statistical Multiplexing?

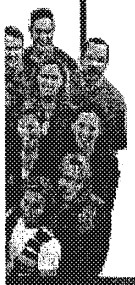
- **How do programmable DSP's play a role?**
  - What benefits do programmable DSP's offer?
  - How are TI DSP's well suited for transcoding?

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1973
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Compressed Video Manipulation

- **Many applications work with compressed video**
  - Video servers/streaming
  - Broadcast distributors
  - Client-side playback/capture
- **Real-time processing in compressed domain**

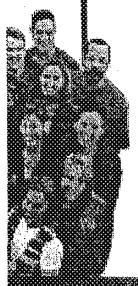




Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1974
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Definitions

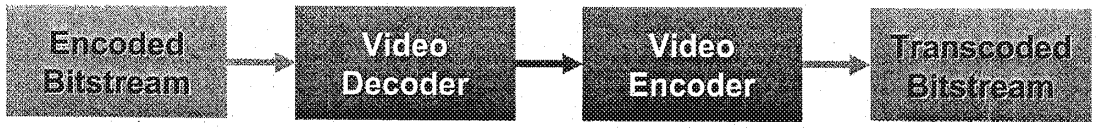
- **Transrating**
  - Reducing bit rate within single compression standard
- **Transcoding**
  - Changing between different compression formats
  - Within a single compression standard, supports:
    - Transrating functions
    - Frame rate or resolution reduction
    - Splicing and logo insertion
- **Recoding**
  - Special (simplified) type of transcoding
  - Full decode followed by full encode

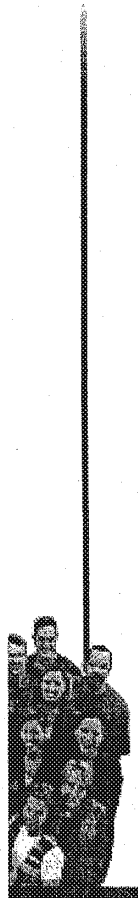


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1975
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Recoding

- **Simple to implement**
  - Only need functional encoder and decoder
- **Produces worst quality of all transcoding types**
  - Suffers from generation loss
- **Most computational form of transcoding**
  - No complexity shortcuts are taken
- **Does produce acceptable results in some cases**
  - When converting from high bit rate compressed source

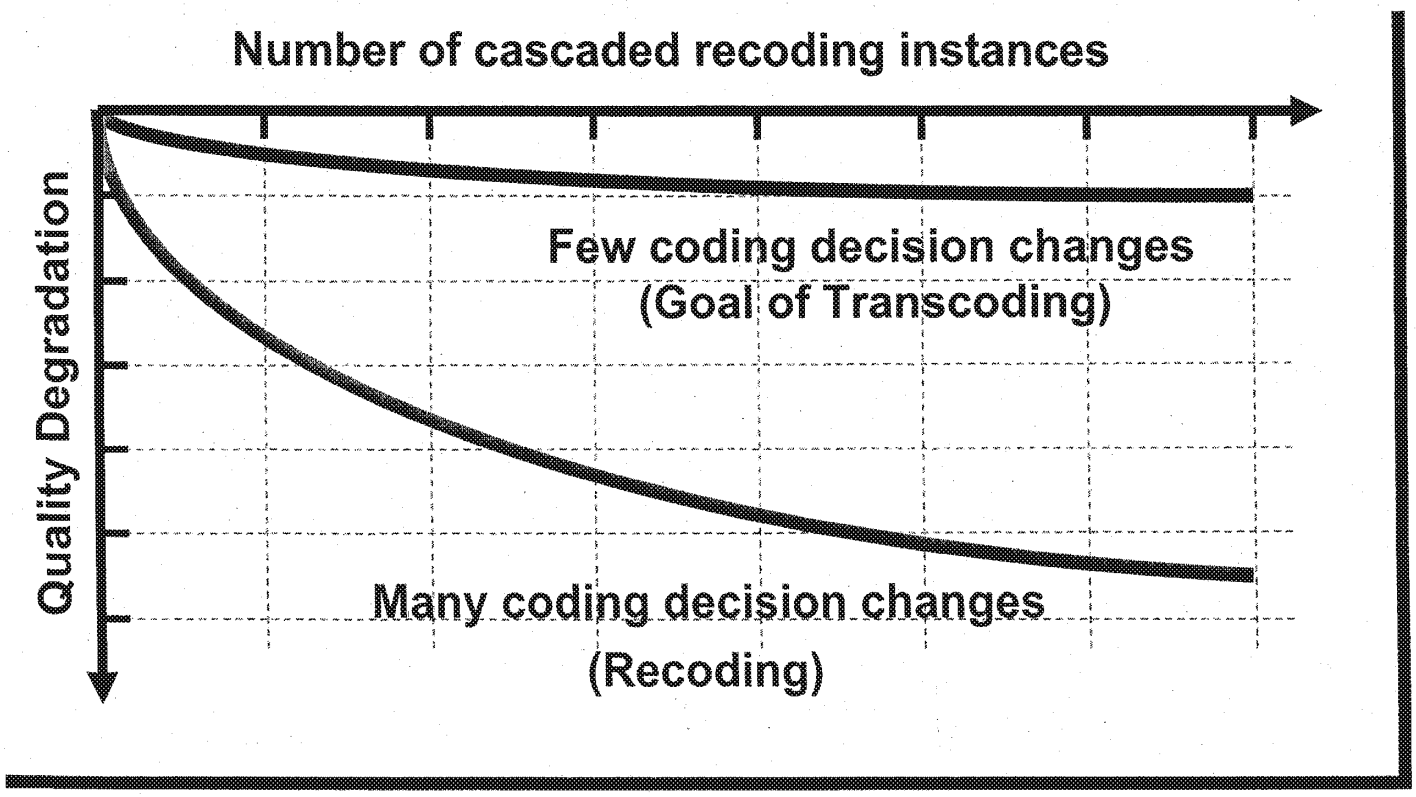




Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1976
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Recoding Generation Loss

- Quality degrades with each recoding generation

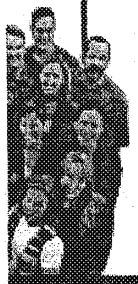


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1977
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Why Not Change Coding Decisions?

- An encoder makes many coding decisions
  - Chooses coding modes (INTRA, INTER, NOT CODED)
  - Selects type of motion compensation
  - Selects type of INTRA prediction
  - Chooses value of motion vectors
  - Chooses quantizer step size

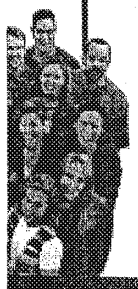
- A transcoder cannot access original video
  - Coding decisions must be made on encoded video
  - Changing coding decisions introduces error
  - Error builds up fast with certain changes
    - When changing coding modes
    - When changing motion vectors



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1978
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

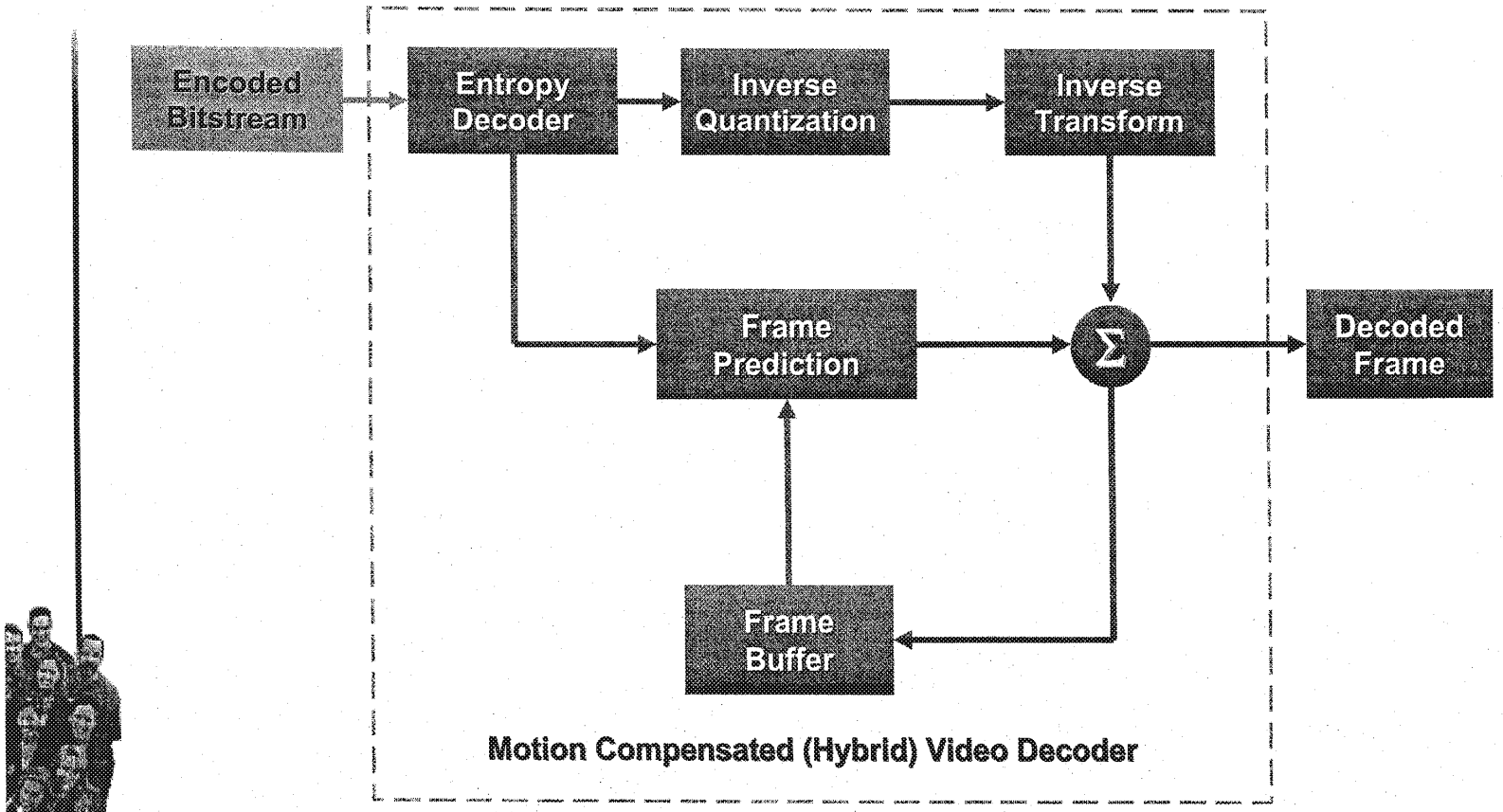
Types of Video Transcoding

- Type 1
  - Handles small bit rate reduction (at good quality)
- Type 2
  - Handles medium bit rate reduction (at good quality)
  - Better quality than Type 1, but higher complexity
- Type 3
  - Handles large bit rate reduction (at good quality)
  - Supports resolution reduction
  - Supports frame rate reduction
  - Supports splicing and logo insertion



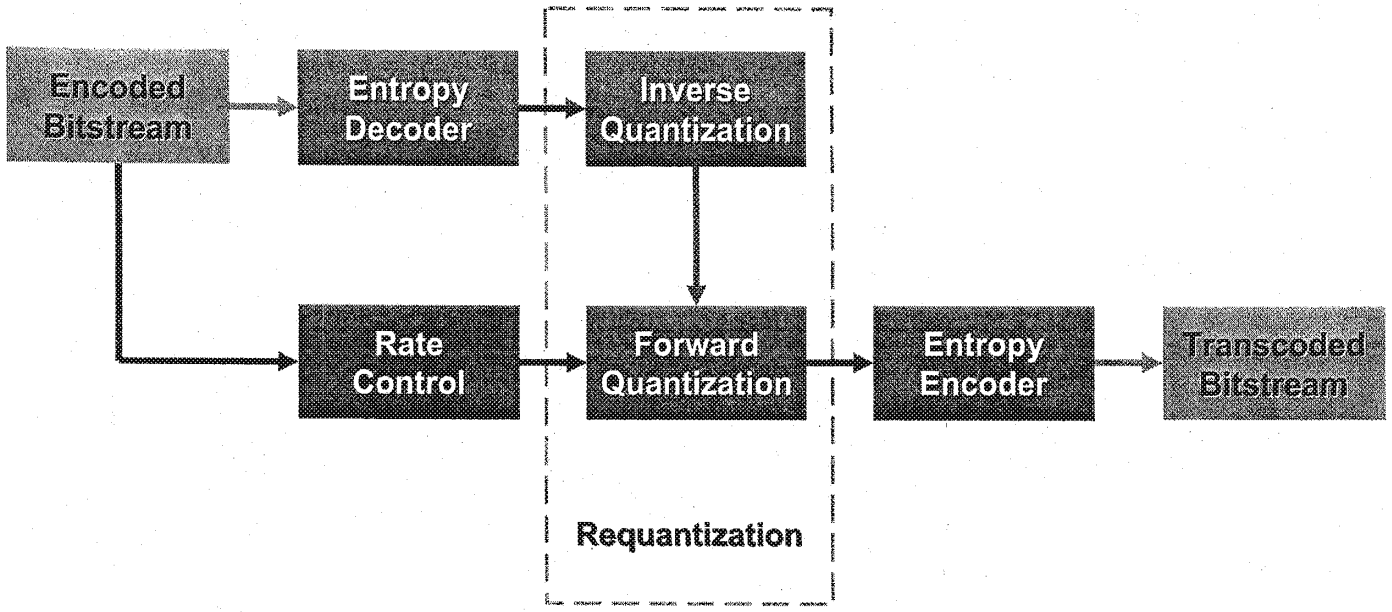
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1979
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

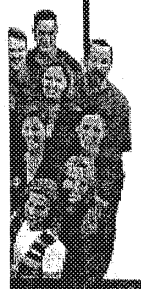
Motion Compensated Video Coding



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1980
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

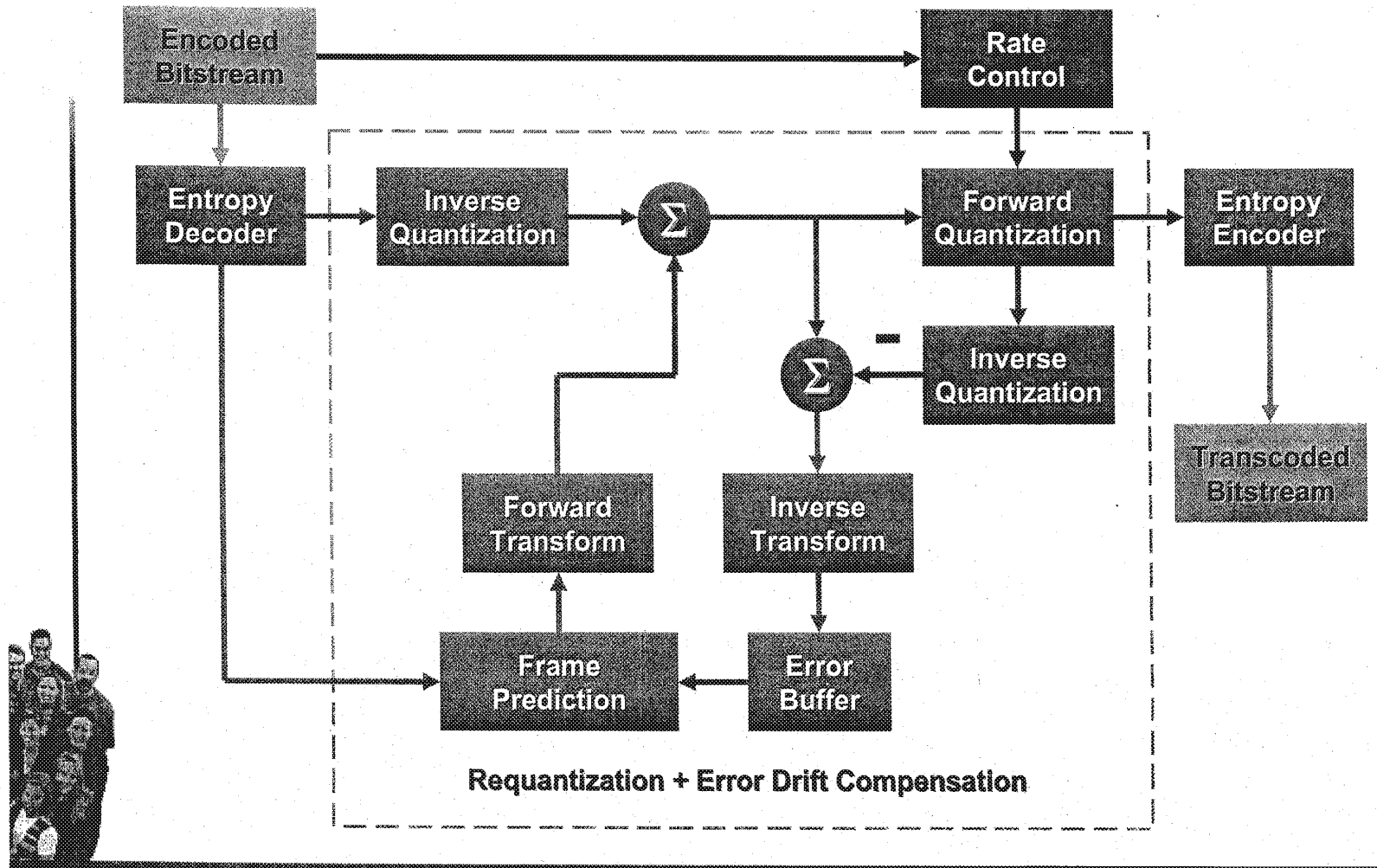
Type 1 Transcoding (Transrating)





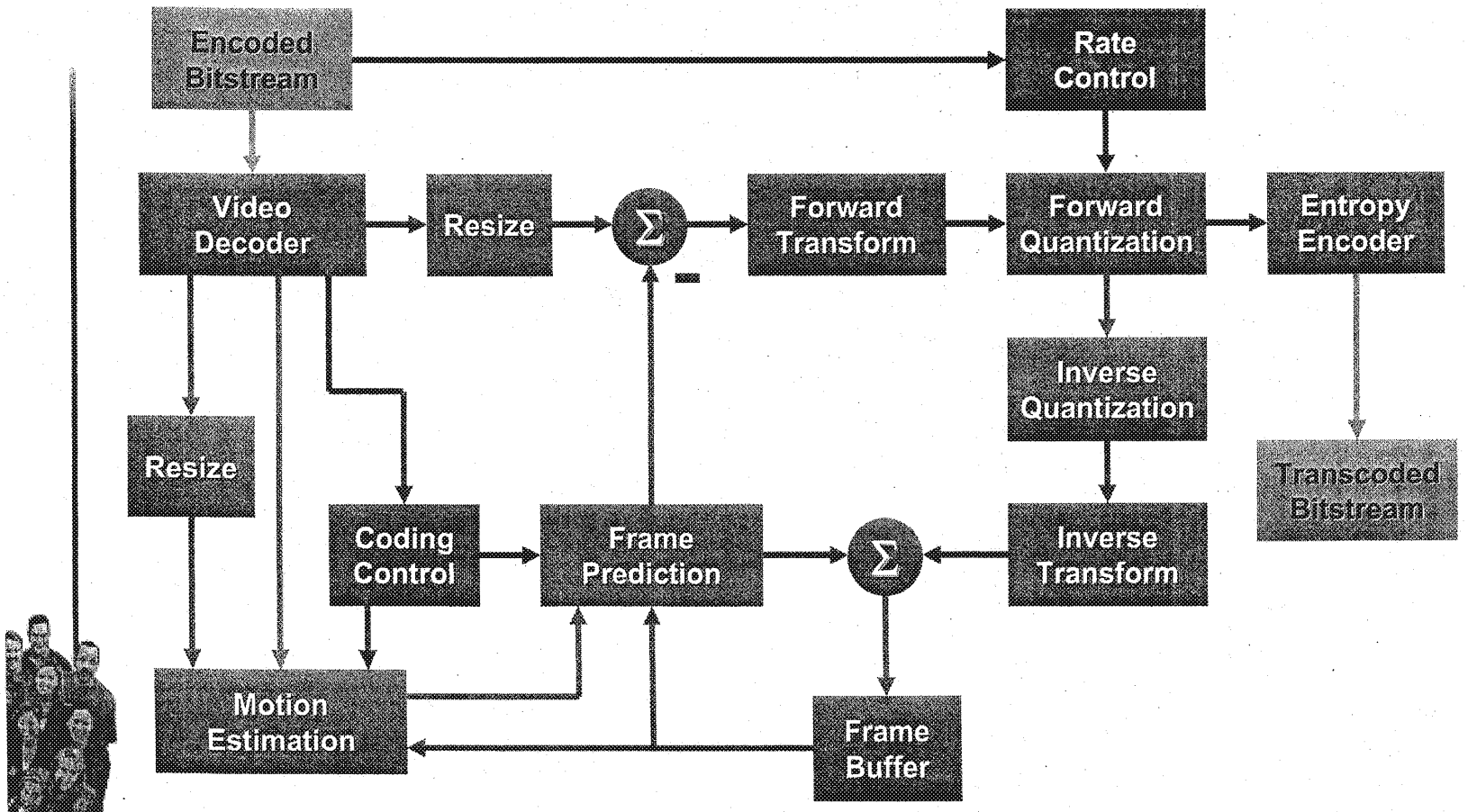
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1981
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Type 2 Transcoding (Transrating)



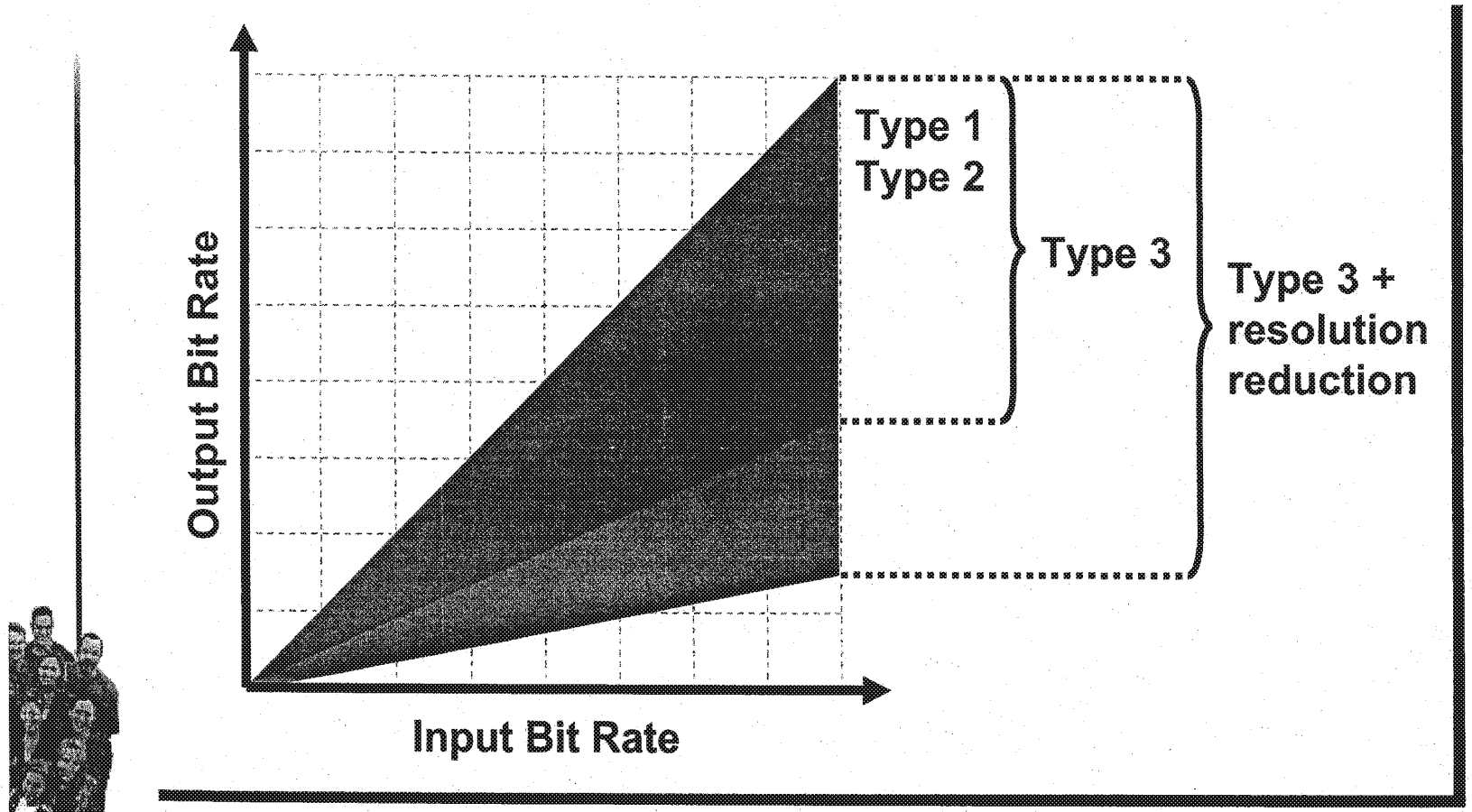
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1982
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Type 3 Transcoding



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1983
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

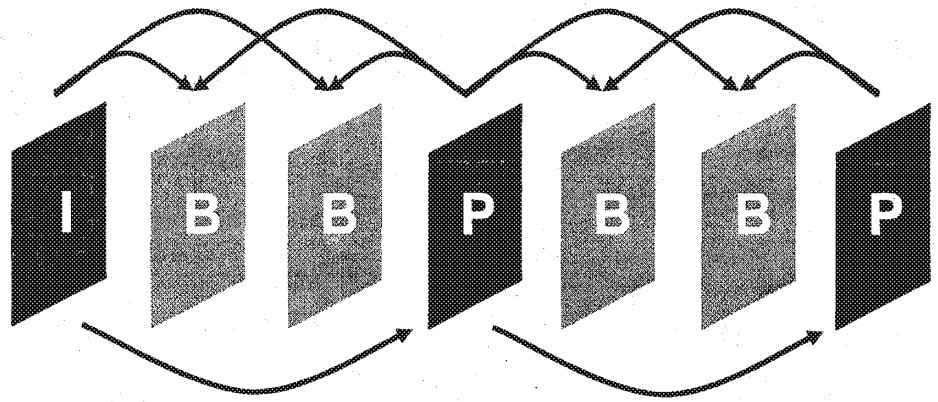
Transcoding Operational Range

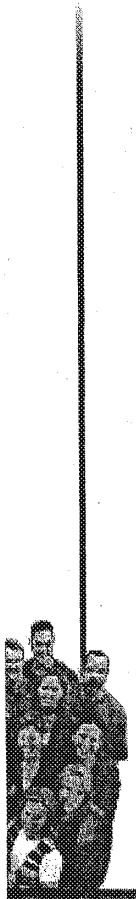


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1984
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Transrating Issues / Strategies

- Some bit rate changes can be difficult for Type 1 & 2
  - Converting between constant & variable bit rate: VBR to CBR
  - Bitstreams with many scene changes or I-pictures
- Transrate more aggressively on B-pictures
  - B-pictures do not contribute to feedback loop
  - B-pictures are allocated less bits than I or P-pictures
  - Can get by with Type 1 if only B-pictures are transrated
  - What if bitstream contains only I and P-pictures?

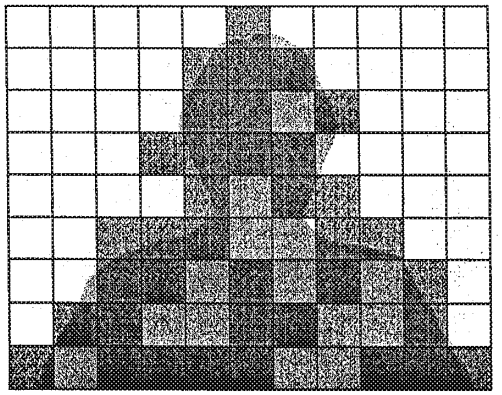




Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1985
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Transcoding Issues / Strategies: Within Standard

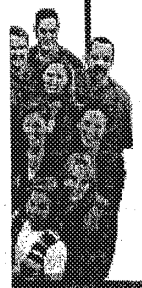
- **Must decide when to make coding changes**
  - Keep original coding decisions whenever possible
  - Can change coding decisions per macroblock or frame
  - Resizing makes reusing coding decisions difficult
  - May have to insert/remove I-pictures
- **Minimize complexity**
  - Reuse original motion vectors to aid motion estimation



Rate change (Type 1 or 2)

Mode change (Type 3)

No mode/rate change

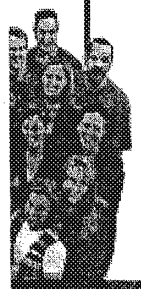


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1986
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Transcoding Issues / Strategies: Across Standards

- Best to match video coding tools
  - Maintain B-pictures and interlace coding tools
  - Maintain frame rate and resolution (if possible)
- Converting across different coding tools is hard
  - Recoding may be a good (only) solution in some cases
- Achieve bit rate reduction via better coding tools
  - Must choose when to use new coding tool

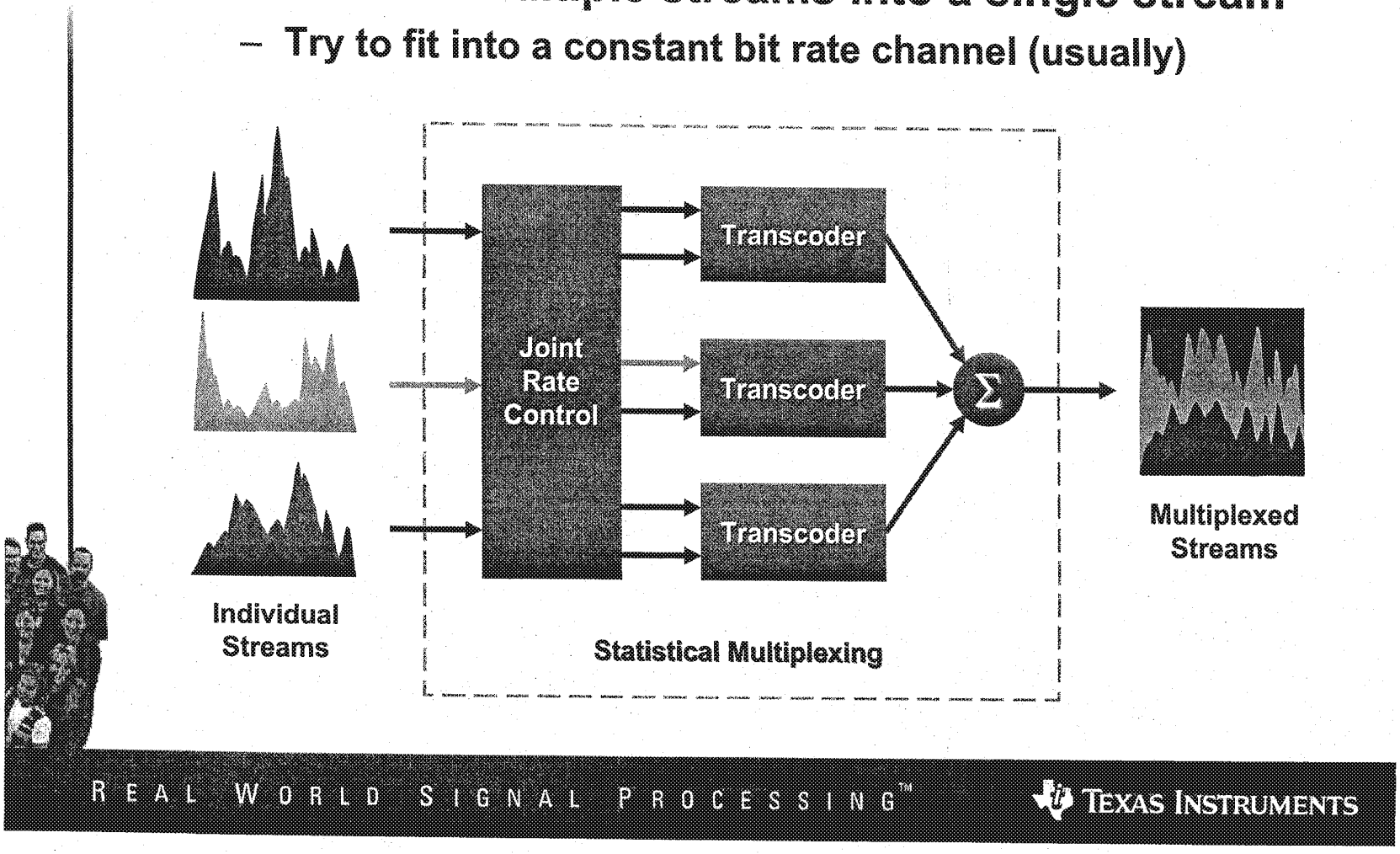
Input Standard	Output Standard	Transcoding Type
MPEG-2 MP@ML	MPEG-4 Advanced Simple Profile	Type 3 (preferred) Recoding
MPEG-2 MP@ML	MPEG-4 Simple Profile	Recoding



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1987
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Statistical Multiplexing

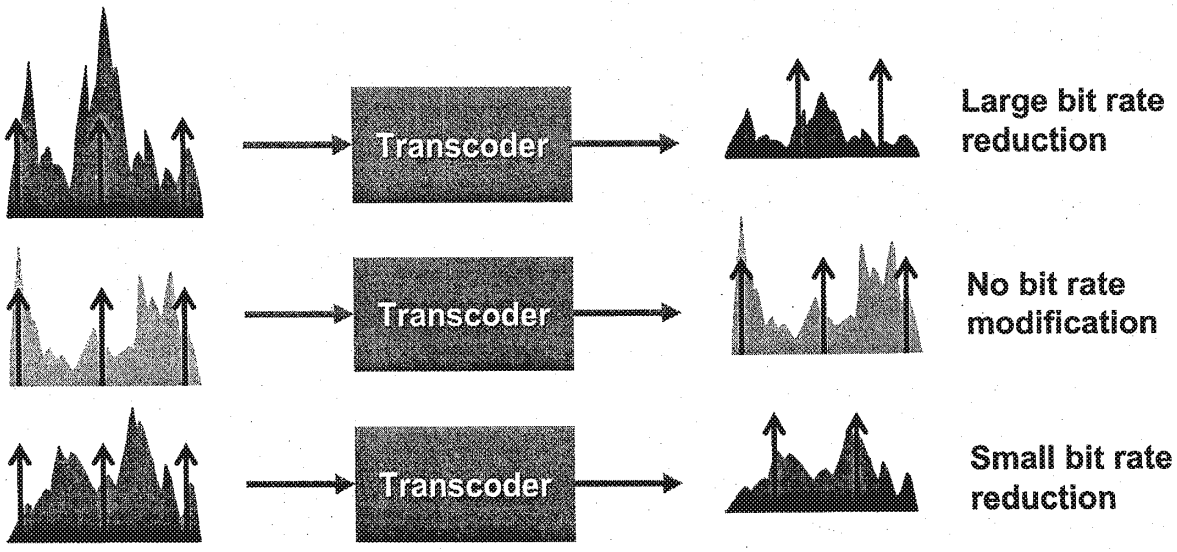
- Transcode multiple streams into a single stream
  - Try to fit into a constant bit rate channel (usually)



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1988
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Statistical Multiplexing Issues

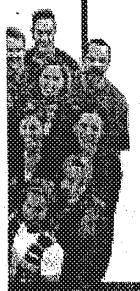
- **Hard to develop good joint rate control**
  - Need to estimate statistics across multiple streams
  - Different streams are (often) allotted different bit rates
  - Stagger Group-Of-Picture (GOP) so I-pictures do not coincide across streams



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1989
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Benefits of Texas Instruments' Programmable DSP's?

- TI's Programmable DSP's are highly optimized for video
  - Competitive performance with ASIC solutions
- Remotely configurable via network
  - Update software with new compression standards
  - Upgrade features (logo insertion, resolution support)
  - Reconfigure transcoding types (e.g. Type 1 to Type 3)
- Shorter development cycle than ASIC solutions
  - Faster support of emerging compression technologies
  - Easily reuse software across compression standards
- Longer product life cycles than ASIC solutions
  - Quicker support of new features & hardware platforms

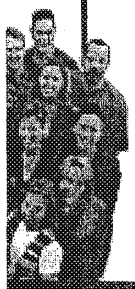


Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1990
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Advantages of TI 'C6000 DSP's

- Advanced VLIW architecture
  - Eight parallel functional units with SIMD capability
  - Conditional execution of all instructions
  - Bandwidth management peripherals: Cache and DMA

- Contains many instructions beneficial for video
  - Sum-of-Absolute-Difference (SAD)
  - Bilinear average
  - Addition/subtraction with saturation
  - Byte-aligned multibyte loads and stores
  - Dot product
  - Bit packing and extracting



Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1991
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Conclusions

- **Use of transcoding types depend on application**
  - Type 1: capable of high-speed bit rate reduction
  - Type 2: capable of moderate-speed, better-quality bit rate reduction (than Type 1)
  - Type 3: capable of high-quality bit rate, resolution and frame rate reduction, including format conversion
  - Recoding: capable of all transcoding, but acceptable only with high-quality input source

- **Programmable DSP's are ideal for transcoding**
  - Remote configurability is very desirable
  - TI 'C6000 family of DSP's are optimized for video

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1992
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Simplifying Video Editing Using Metadata

Digital video is becoming increasingly ubiquitous. However, editing video remains difficult for several reasons: it is a time-based medium, it has dual tracks of audio and video, and current tools force users to work at the smallest level of detail. Based on interviews with professional video editors, we developed a video editor, called Silver, that uses metadata to make digital video editing more accessible to novices. To help users visualize video, Silver provides multiple views with different semantic content and at different levels of abstraction, including storyboard, editable transcript, and timeline views. Silver offers smart editing operations that help users resolve the inconsistencies that arise because of the different boundaries in audio and video. We conducted a preliminary user study to investigate the effectiveness of the Silver smart editing. Participants successfully edited video after only a short tutorial, both with and without smart editing assistance. Our research suggests several ways in which video editing tools could use metadata to assist users in the reuse and composition of video.

KEYWORDS
Digital video editing, multimedia authoring, metadata, Silver, Informedia.

INTRODUCTION

Digital video is becoming increasingly ubiquitous. Digital video equipment is more accessible than ever, and there is an increasing amount of video material available on the World Wide Web and in digital libraries. As technology and content become available, users will shift from passive spectators to active creators of video. Forrester Research predicts that by 2005, 92% of online consumers will create personal multimedia content at least once a month [17]. However, although many exciting research projects are investigating how to search, visualize, and summarize digital video, there is little work on new ways to support the use of the video beyond just playing it.

This is unfortunate, because video editing has several unique challenges not found with other media. One is that digital video is a time-based medium. This property makes it difficult for users to browse and skim video. Users often must linearly search their source video to find the clip they desire.

Another challenge for editing video is that it is a dual medium. Most "video" actually consists not just of a video track but also an audio track. These tracks must be kept

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DIS2002, London. © Copyright 2002.
ACM-1-58113-515-7/02/0006...$5.00

Juan Casares
A Chris Long
Brad A Myers
Rishi Bhatnagar
Scott M Stevens
Laura Dabbish
Dan Yocum
Albert Corbett

Human Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA

juan.casares@cs.cmu.ed

http://www.cs.cmu.edu/~silver

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1993
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

synchronized, but the user must also be able to overlay them when desired, for example during transitions from one shot to another. Further, when a shot is cut from a video for use elsewhere, the user must be able to disentangle overlaid audio and video.

A third problem is that the syntactic units that users want to edit are shots of video and words or sentences of audio, but current tools require users to examine video at the individual frame level and audio using a waveform. To perform most editing operations, such as cutting a shot, the user must manually pinpoint specific frames, which may involve zooming and numerous repetitions of fast-forward and rewind operations. Finding a specific word or sentence using a waveform is similarly tedious.

These problems make editing video a difficult, tedious, and error-prone activity. Commercially available tools, such as Adobe Premiere [2] and Apple iMovie [1], allow creation of high-quality video, but they do not adequately address the issues raised above, which makes them harder to use, especially for novices.

To better understand video editing, we visited a video processing studio and interviewed professional video editors. We also examined commercial and research video editing systems.

We created a system, called Silver, to explore techniques to make video editing easier. The key innovations in the Silver editor include: providing an editable transcript view; coordinating the selection across all views including partial selections and different selections in audio and video; and smart editing through smart snap, smart selection and smart cut and paste. Silver can work with video from any source, although some of its features are especially useful with previously edited video. Presently it uses video and metadata from the Informedia Digital Video Library [22].

This paper is structured as follows. The next section discusses related work on video systems. Next, we discuss a specific challenge for video editing: L-cuts. Then we describe the Silver interface, followed by the evolution of the Silver interface. Next, we discuss issues in the interface and implementation of Silver. Then we describe a pilot user study and its results. Finally, we discuss future work and conclude.

RELATED WORK

In this section, we describe different types of related work. We start with systems that use metadata for video editing. Next, we discuss systems that automate the process of video editing to some degree. Then we discuss work on video visualizations that address the issues of scale and time. Lastly, we describe Informedia, which is our source of video and its corresponding metadata.

METADATA AND VIDEO EDITING

The notion of using metadata for editing video is not new. For example, Mackay and Davenport examined the role of digital video in several interactive multimedia applications and concluded that video is an information stream that can be tagged, edited, analyzed, and annotated [13]. Later, Davenport et al. proposed using metadata for home movie editing assistance [7]. However, they assumed this data would be obtained through manual logging or with a "data camera" during filming.

Currently, there is a large body of work on the extraction and visualization of information from digital video (e.g., [8, 19]) that make a data camera unnecessary.

One example of a system that uses metadata is IMPACT [21], which uses automatic cut detection and camera motion classification to create a high level description of the structure of the video. The user can organize the shots in a tree structure and then edit the composition by moving the branches [20]. IMPACT supports this process by recognizing objects across multiple shots.

AUTOMATION

Several systems edit video with varying degrees of automation. Fully automated systems may be used for news on demand [3] or quick skimming [5], but do not really support authoring.

The Video Retrieval and Sequencing System (VRSS) [6] semiautomatically detects and annotates shots for later retrieval. Then, a cinematic rule-based editing tool sequences the retrieved shots for presentation within a specified time constraint. Examples of cinematic rules include the parallel rule, which alternates two different sets of shots, and the rhythm rule, which selects longer shots for a slow rhythm and shorter shots for a fast one.

The Hitchcock system [9] automatically determines the "suitability" of the different segments in raw home video, based on camera motion, brightness, and duration. Similar clips are grouped into "piles." To create a custom video, the user drags segments into a storyboard and specifies a total desired duration and Hitchcock automatically selects the start and end points of each clip based on shot quality and total duration. Clips in the storyboard are represented with frames that can be arranged in different layouts, such as a "comic book" style layout [4].

The MovieShaker system automates video editing almost completely [18]. The user specifies multiple video clips and a "mood" (e.g., romantic, wild), and the program combines and edits the clips into a single video.

While automation makes editing faster, it usually involves taking away power from the user, which is not always desirable. In fact, user studies led to changes in Hitchcock to give more information and control to the user [10].

VISUALIZING TIME AND SCALE

Video editing is difficult in part because it is hard to visualize time. Due to the semantic and syntactic nature of the video, where selecting a five-second piece involves fiddling with frames in $1/30^{th}$ of a second increments, it becomes important to present the information at different scales. No one scale is sufficient to efficiently perform all the operations needed.

The Hierarchical Video Magnifier [14] allows users to work with a video source at fine levels of detail while maintaining an awareness of the context. It provides a timeline to represent the total duration of the video source; a second timeline shows a detail of the first one and illustrates it with a frame sampling. The user can select which portion of the top timeline that is expanded on the next one. There is also a mechanism that can be used to add a new timeline with a higher level of detail. Successive timelines create an explicit spatial hierarchical structure of the video source.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1994
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The Swim Hierarchical Browser [23] significantly improves on this idea by using metadata. Swim displays automatically detected shots in the higher level layers instead of frame samples.

Hierarchical views in the Hierarchical Video Magnifier and Swim are used for navigation only. Silver extends hierarchical views to allow editing as well.

INFORMEDIA

Silver currently obtains video from the Informedia Digital Video Library [22], which has over 2,000 hours of material and is adding new material daily. Our primary motivation for using Informedia is that it generates several types of metadata, which Silver uses to enhance its editing capabilities.

A textual transcript of the audio track, generated from closed-captioning information and speech recognition is one type of metadata Informedia provides [8, 11]. The transcript is time-aligned with the video using the Sphinx speech recognition system [16]. Silver is the only video editor we know of that uses an audio transcript.

Informedia uses image analysis to detect shot boundaries and extract representative thumbnail images from each shot [5] and automatically creates titles and summaries for video segments.

Informedia also provides search infrastructure to allow users to find video clips based on keywords and to browse the results.

L-CUTS

When working with video that has been professionally edited, it often happens that the start and end times of a segment are different for the audio and the video. This is called an "L-cut". For example, in newscasts, the camera often cuts from an anchorperson to a field reporter while the anchorperson is still talking. L-cuts give the video a continuous, seamless feel, but make extracting pieces much harder because the video and audio portions must be adjusted separately. This requires much fine-tuning to remove extraneous video or audio portions.

We conducted an informal study to investigate the prevalence of these cuts, and found it to be significant, at least in recorded news shows. In looking at 238 transitions in 72 minutes of video clips recorded from CNN by Informedia, 61 (26%) were L-cuts. Of the 61 L-cuts, 44 (72%) were cases where the audio of the speaker came in before the video but ended at the same time. Most of these were interviews where the voice of the person being interviewed would come in first and then the video of the person after a few seconds (see Figure 1, Shots A and E). Most of the other ways to overlap the video and audio were also represented. Sometimes, the audio and the video of the speaker came in at the same time, but the audio continued past the video (Figure 1, Shot B) or the video ended after the audio (Shot D). When an interviewee was being introduced while appearing on screen, the video might come in before the audio, but both end at the same time (Shot C). To copy or delete any of these would require many steps in other editors.

SILVER

To address the problems of video editing mentioned above, we developed Silver, a digital video editor. Silver simplifies video editing primarily by incorporating metadata into the interface and by providing multiple, synchronized views of the video composition. Silver also includes support for managing L-cuts.

We want to make video editing as easy as text editing. To that end, we have examined text editors and have adopted some of their techniques. Text editors have long had multiple representations in order to support different styles of editing and different tasks. For example, Microsoft Word provides "outline," "normal," "print layout," and "print preview" views. The multiple views in Silver allow the user to browse the video by displaying it across space instead of time, to view and manipulate the audio and video separately, and to examine the video at multiple levels of detail. An overview of the Silver interface is shown in Figure 2.

We envision several uses for Silver, such as a high school student getting the latest news footage and putting together a short presentation on current events. Another scenario is processing semi-automatically home video to extract the metadata and then using Silver for editing. Silver could even guide video pre-production, starting with a script, creating a storyboard, and then replacing pieces with the actual video as it is shot.

The remainder of this section describes these views and how each supports video editing. The first subsection describes the two views for managing the video directly, the storyboard and the timeline. The next discusses a view for managing the audio track using a transcript. Following is a brief description of other views in the Silver interface. The final subsection describes the smart editing features in Silver.

STORYBOARD VIEW

The storyboard view allows the user to arrange and rearrange the video composition at a large scale. It displays each *clip*[1] in the composition as a separate icon containing a title and a key frame. Initially, we dynamically analyzed each clip to identify a representative key frame. However, early users were confused when the displayed key frame changed as they edited the clip and preferred using a static frame close to the start of the clip. The clips are sorted chronologically, so moving clips within the storyboard changes the order in which they appear in the final video.

In addition to the composition, the storyboard also contains a "scratchpad," or staging area. Clips in this area are not part of the composition, but are easily available, which is convenient for pieces under consideration or being edited. A vertical line separates clips of video in the scratchpad from clips currently in use. Dragging a clip across this line adds or removes it from the composition.

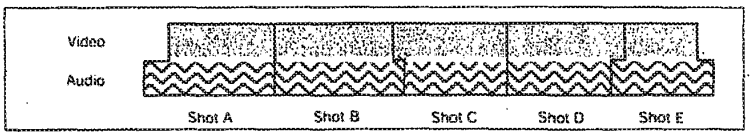


Figure 1: Different kinds of L-cuts

1 A clip is a continuous segment of video from a single source file.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1995
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

HIERARCHICAL TIMELINE VIEW

Silver, like most video editors, provides a timeline view as the primary view for detailed editing. Silver allows users to easily view and manipulate video at different scales by providing a three-level view: clip, shot, and frame levels (see Figure 3). The timeline allows the user to work at a high level of detail (i.e., in the frame level) without losing the context within the composition (i.e., in the shot and clip levels).

The top line is the clip level which represents the entire video. At the top are two colored bars, one for audio and one for video, that show gaps in the tracks and clip boundaries. The central section of the clip level shows a more detailed view of each clip than is shown in the storyboard. In addition to a representative frame, it shows a timeline and if space allows it shows the end frame, title, and duration.

Users control how much is displayed in the other two levels of the timeline using a sophisticated scrollbar at the bottom of the clip level. In addition to the usual dragging behavior, the scrollbar may also be resized to zoom the shot level in or out. It also contains a miniature version of the shot level scrollbar, which shows the position of the frame level within the shot level and can be dragged to scroll the frame level.

The shot level allows users to interact with video at a higher level than individual frames by displaying video as individual *shots*[2]. Algorithms that detect significance between-frame image changes identify the shot boundaries [12]. Each shot is displayed using the representative frame for the shot as chosen by Informedia. Shots smaller than a certain threshold are displayed with proportionally smaller thumbnails (for example, the third shot in Figure 3). As in the clip level, there is an augmented scrollbar at the bottom of the

2 A shot is a continuous segment of video from a single camera.

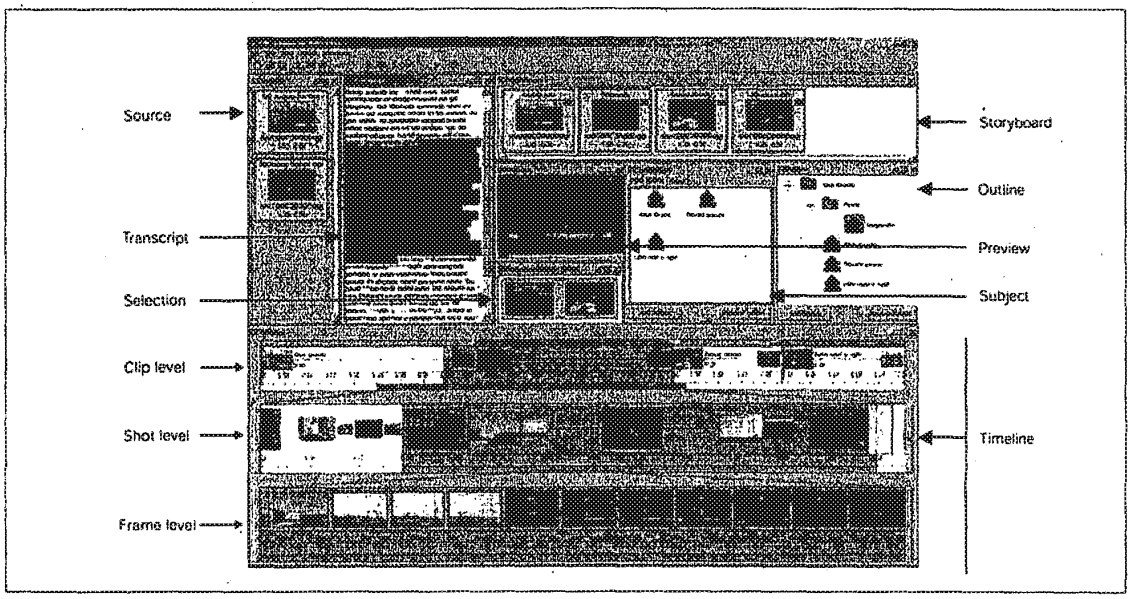


Figure 2: Overview of the Silver interface

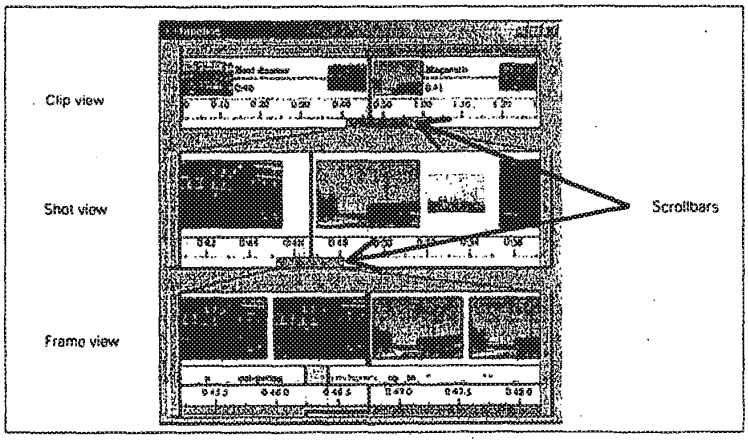


Figure 3: Timeline window. The scrollbar in the clip view can be moved and resized to scroll and zoom the shot view. The shot view scrollbar can be moved and resized to scroll and zoom the frame view.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1996
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

shot level used to scroll and zoom the frame level.

The frame level can display the individual frames of the video, so the user can quickly find particular cut points. The frame level addresses the dual medium issue by displaying the transcript, including the start and end times of every word. In other editors, there is little information about the audio, so users must repeatedly play the audio to find a desired part. The time-aligned transcript Silver provides allows the user to locate the desired part of the audio and video more quickly and easily than is possible with conventional editors.

At all levels, boundaries between clips are shown as blue lines. The user can resize a clip by dragging the clip boundary.

A key feature of the Silver timeline is that it shows different representations together, allowing the user to see the clip boundaries, the time, samples of frames, the cuts, the transcript, annotations, etc. In the future, other representations could be added, such as audio waveform or labels for recognized faces.

TRANSCRIPT VIEW

Unlike any other editor of which we are aware, Silver shows a time-aligned textual transcript alongside the video, under it in the frame level of the timeline view (see Figure 3). This view is very useful for fine-grained audio and video synchronization, but little text can be shown there simultaneously because it is mapped to time in a single line.

So that the user can more easily view and edit the transcript, Silver offers a text-editor-like window dedicated to the transcript (see Figure 4). As in the timeline view, the text in the transcript window is tied to the video, so the user can edit the *video* by cutting and pasting the text in the *transcript*.

One challenge for displaying transcripts is the need to display information in addition to the spoken words. Our solution is to insert special characters into the text. For example, the boundary between clips is shown as a blue double bar ("||"), and sections without audio are represented with dashes ("-"). This is shown near the top of Figure 4.

Because speech recognition can contain mistakes, Silver inserts a green star ("*") where there appears to be a gap, misalignment, or noise. Silver allows the user to manually correct mistakes in timings or recognition. Users can type the correct text and indicate what video is associated with it

OTHER VIEWS

Silver provides the user with many other views. The Outline and Subject views permit the classification of clips in a hierarchical or flat taxonomy to help users organize their composition. The Source view provides access to all the source material in one convenient place. The Selection view shows the first and last frames of the selected video. The Preview window is used for video playback.

Silver is built on top of the Informedia client interface, so the search, browse, and summarization views provided by Informedia are available to the user. For example, the user can use the Informedia Search window to enter a keyword and then drag one of the clips displayed in the Search Results to one of the Silver windows.

These views are described in more detail in another paper [15].

SMART EDITING

Silver tries to reduce the tedium of editing video by using metadata to provide more intuitive interaction. One large source of this tedium is L-Cuts, which are a common feature of edited video. In this subsection, we describe smart snap, selection, and cut and paste.

Smart Selection

One reason that video editing is much more tedious than text editing is that with video the user must select and operate on a frame-by-frame basis. Simply moving a section of video may require many minutes while the beginning and end points are laboriously located.

For selections in an audio-based view, Silver may snap the edge of video selection to a shot break. To avoid flashing a few frames of another shot, we use a threshold

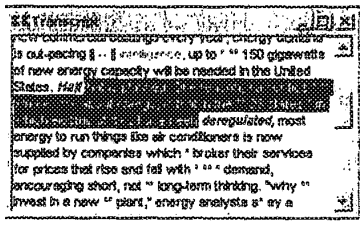


Figure 4: Transcript window. In the top line, we can see blue clip breaks ("||"), enclosing a gap in audio ("- -"). The word *intelligence* was cut in the middle. The green stars represent noise or recognition errors. At the edges of the selection, *half* and *deregulated* are partially selected.

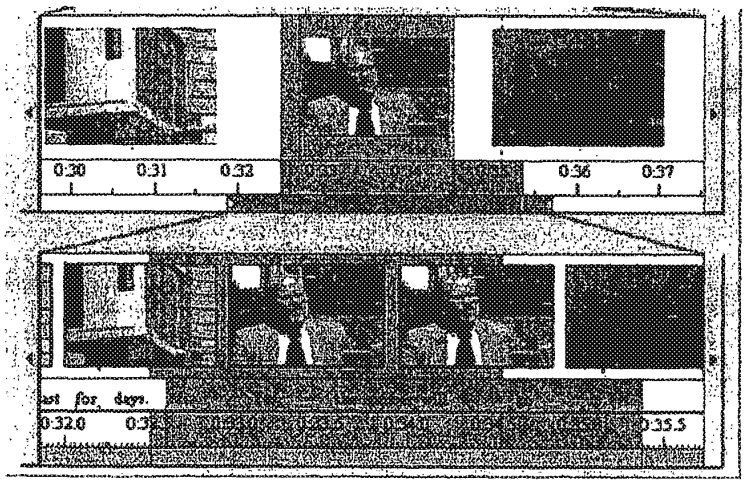


Figure 5: Smart selection. Audio and video can be selected separately, and video selection snaps to shot boundaries.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1997
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

of one second when the shot break is outside of the audio selection. For example, the right edge of the video selection in Figure 5 is snapped to a shot boundary. As shots become longer than a second, they no longer are objectionable. We use a threshold of 0.3 seconds when the shot break is inside the selected region. It is preferable not to jump to the middle of a shot.

For video-based views, if the selection cuts a word in half then Silver extends or reduces the audio selection to the nearest word edge. This is shown in Figure 5 where both edges of the audio selection are snapped to a word. This might take the audio selection across a shot break.

Selecting in a mixed media view causes the selection to snap in both audio and video. However, the user is always in control: the smart selection can be disabled by pressing a modifier key.

Silver eases selection by providing a mechanism to select a meaningful segment based on the view. For example, in the shot level of the timeline, a double click selects the corresponding shot. In some cases, a larger segment is selected with a triple click, such as in the transcript view (where a double click selects a word and a triple click selects a sentence).

When the user selects a region, Silver may adjust the audio or video boundaries based on editing rules of thumb and the available metadata. This is especially useful when the source contains L-cuts.

Smart Snap

When moving the cursor around in the timeline view, the cursor snaps differently to features of the composition depending on which row the mouse is moving over. In video-based rows, the cursor snaps to shot breaks if close by, or to the nearest frame otherwise. In the transcript row, the cursor snaps if close to a word boundary. In the case of silence, the cursor does not snap. In addition, the cursor always snaps to clip boundaries and the edges of the current selection.

Smart Cut and Paste

When a segment with non-aligned audio and video edges is deleted or pasted, the resulting ragged edges would typically require the user to again go down to frame level and adjust the edges or create a special effect to avoid gaps in the composition. For example, if a segment with shorter video than audio (such as Shots A, B, or E in Figure 1) is inserted, the user would need to close the gap in video by either extending the video or shrinking the audio on either end of the gap.

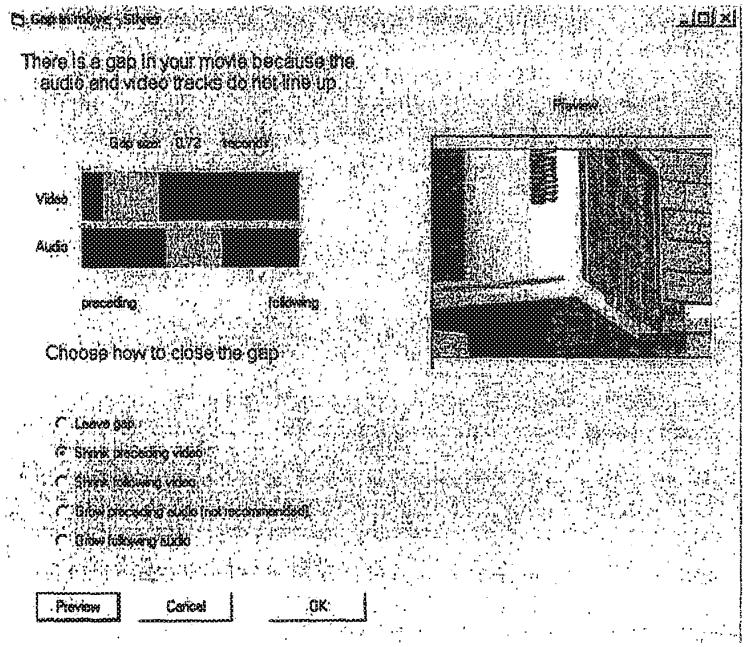


Figure 6: Close gap dialog. This dialog would appear if the user deleted the selection in Figure 5.

Silver makes this task much easier by semi-automatically determining how to deal with the uneven ends. It presents a dialog box to the user with different options to close the gap. This dialog displays the current gap and how the different options would close it. For example, after deleting the segment that is shown selected in Figure 5, a dialog box like the one in Figure 6 would appear. This dialog is needed because more audio than video is selected in Figure 5, so deleting the selection would leave a gap in the audio. Deleting Shot E in Figure 1 would have the same effect.

If there is a gap in audio (e.g., if Shots C or D from Figure 1 are pasted), then Silver recommends leaving it because silence is generally not disruptive. However, it is not acceptable to leave a gap in video, which would show up as black or blank video. Silver recommends extending the edges of involved video as long as it does not take them across a shot boundary.

In cases where the editing operation leaves two gaps, such as when a clip with jagged edges on both ends is inserted, Silver presents a double version of the dialog box where the user can look at both ends simultaneously.

THE EVOLUTION OF SILVER

This section describes how the Silver interface changed over time as a result of informal usage we observed.

TIMELINE

The most complex part of the Silver interface is the timeline. Much of the time we spent designing Silver focused on this view.

Our timeline started with a simple scrollbar, a video track, and an audio track. It showed the clips in the composition, representing them by stretching a representative key frame. However, we wanted to represent more detailed information in the timeline. Our simple timeline could not show different levels of granularity simultaneously. Our approach was to show context and detail in a multiple-level hierarchical timeline.

Other video systems also use multi-level timelines. Some editors (like Premiere) have a two-level timeline. A detailed timeline represents a short amount of time and is used for most editing tasks. It can display different tracks, including multiple video

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1998
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

and audio channels, effects, etc. The second timeline is a very abstract representation of the piece. This timeline is only used to determine what is being shown in the detailed timeline.

TIMELINE SCROLLBARS

There are many interaction challenges when designing a multi-level view like our timeline. In particular, the correlations between the scrollbars required several iterations and fine-tuning. Many of our initial assumptions needed to be changed in response to user testing.

Originally, the scrollbar thumbs were designed to represent a constant proportion and position in relation to the whole scrollbar. So, for example, the top scrollbar thumb could cover the second half of the scrollbar, meaning that the next level represents the second half of the composition. When the window is resized or the composition is edited, the scrollbar would still cover the second half and the next level would still show the second half of the composition, although with a different zoom resolution. This behavior was extremely annoying to users, especially after they had carefully set the zoom level to show, say, shots at a particular resolution.

In the same sense, the middle level's scrollbar thumb would represent a fixed proportion and position in relation to the top scrollbar. To change the resolution of the middle level the user needed to resize the top scrollbar by dragging on one of its edges. However, the middle scrollbar remained the same, meaning that the bottom level's resolution and position changed as well. So, changing the top scrollbar would often change what was displayed in the bottom level and users would need to readjust the middle scrollbar.

Since the middle scrollbar did not change in response to changes in the top scrollbar, displaying the last frame of the composition in the bottom level involved dragging the top scrollbar to the right edge, and then dragging the middle scrollbar to its right edge.

We changed the interaction model so each scrollbar kept its zoom-level constant. We assumed users did not want the bottom level to change when the top scrollbar was resized. So, growing the top scrollbar would also shrink the middle scrollbar so that the bottom level would maintain its resolution. Furthermore, the resizing needed to be centered on the thumb, so that dragging one edge would similarly change the other edge. Otherwise, the bottom level would move even if it maintained its resolution. Pressing the *alt* key while resizing disables the centering feature, allowing the user to change just one side of the scrollbar thumb. This is useful when the user is interested in something that is not in the center.

Users had problems recognizing the relationship between the different levels. That is, they did not understand that each level was a blowup of the previous one. We added guiding lines from the scrollbars to the edges of the next level to help indicate this. Also, we color-coded each thumb, and showed a miniature version of the middle thumb in the top-level thumb.

TIMELINE VIEWS

Each part of the timeline represents video at different levels of detail.

Frame View

Initially, we showed a frame sample row in every level. However, in the less-detailed levels (i.e., shot and clip), samples could seem almost random since a single frame sample represented a large amount of time. Users were confused because they expected the samples to be representative of the time they covered on the timeline. The confusion ended when we removed frame samples from the other levels.

In our original presentation of the frame view, users would assume that the video samples in the frame view represented individual frames. However, the samples only showed individual frames if the level was at the highest level of detail. To make the difference between showing a sample and all individual frames, we added a double-line border to frames when shown individually.

Clip view

We represented clips by stretching the picture of a representative frame to fill its corresponding space in the timeline. For long clips, this distortion made the frame unrecognizable. Now we represent the clip with the first frame. If there is enough space, we also show the last frame. When several clips were shown side by side, it became hard to distinguish start frames from end frames. So we now display the last frame at a reduced size. If there is enough space between the start and end frames for a clip, we also show textual information about the clip such as its name and duration.

Since the composition length, window size, and zoom level may change, an important consideration for our views was that they should always show something meaningful. Clips and shots are displayed in more detail if more space is available, but there is always some indication of the presence of each shot and clip. For example, if a clip has very little space, it only shows that a clip exists (with a thick blue line). With more space, it shows the first frame. With more space, the first and last frames. Finally, it includes title and duration if there is a lot of space. The timeline also shows varying amounts of information depending on available space. As space increases, words go from small black dots to the word with starting and end times. If some words do not fit, we try to rearrange them in two levels. With frames we sample to show as many frames as we can, until we can show each individual frame.

INTERFACE AND IMPLEMENTATION ISSUES

This section discusses issues we encountered in our design of the Silver interface and in its implementation.

Many views, one model

All of Silver's views actually represent the same composition. Any change to the composition is reflected immediately in every view. For example, when the user drops a new clip into one of Silver's views, all others are instantly updated.

In addition, selection in one view is mirrored across the others. This feature brings up a number of interesting user interface design challenges.

First, the selection in video may begin and end at different times from the selection in audio. As is described in the smart editing section, this feature is critical to achieving high quality when composing previously edited content. Some views only show the audio or the video track; these views only show their corresponding selection.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 1999
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

We use color to show selections of different media types: orange for video, green for audio, and yellow for segments selected in both (see Figure 5). Usually, users want to select audio and video together, but sometimes they need to separate the two. Single-track selection is done with the right mouse button instead of the left, which selects both. For example, dragging the right button across the frame view of the timeline would select a piece of video without selecting audio.

The second problem that arises from synchronized selection is defining what the "equivalent" portion is. The different views show different levels of granularity, so it may not be possible to represent the selection accurately in some views. For example, if a few frames are selected in the timeline view, as shown in yellow in Figure 5, it is not obvious what should be shown in the storyboard view, since it shows only complete clips. Silver uses a dark shade to highlight a fully-selected item, and a lighter shade to highlight a partially-selected one.

Another example of this problem arises between video and audio views. Individual words in the transcript span multiple frames. When a word is selected in the transcript, it is straightforward to select all the corresponding frames. However, selecting only one of those frames in the video will correspond to only part of that word. This partial selection is shown in the transcript by making the word italic, for example the words "half" and "deregulated" at the edges of the selected text[3] in Figure 4. In this case, if the selected video is moved, the word will be cut into two pieces. Silver represents this by repeating the word in both places, but showing them in a faded color. For example, the word "intelligence" in Figure 4 was split.

Information Overload

Video is time-based, which makes it difficult to display well statically, and the more comprehensive the static display, the more screen real estate it requires. Also, the dual nature of video demands additional screen space. Organizing the large amount of information users need while editing video has been a significant challenge in the design of the Silver interface.

The problem of information overload is exacerbated by the inadequacy of auxiliary media, such as paper, to supplement the interface. In editing a large text document or program, people often print sections so that they can refer to them while editing without using up screen space. However, this strategy is not effective for editing video because of its dynamic nature.

The primary mechanism we have used to address the information overload problem is the scrollable, zoomable timeline view. With this timeline, the user need not view the entire video at the highest level of detail, but may see only the currently relevant part in detail, and still see the context around it in the other, less-zoomed-in views. This approach may be useful in other information-intensive editing environments.

USER STUDY

We conducted a pilot study to evaluate the smart editing features of Silver. Participants were asked to use Silver to perform a series of editing operations on one video with smart editing enabled, and another series on a different video, this time with smart editing disabled. Our participants were seven paid subjects, graduate students from the School of Computer Science.

Design

This study was a within-subjects design. All participants did both experimental tasks. Each performed one task with smart editing enabled and one with smart editing disabled. The order of the tasks and the order of smart editing were counterbalanced across participants.

PROCEDURE

The experiment had three parts: tutorial, experimental tasks, and post-experiment questionnaire.

In the first part, participants read the Silver tutorial and performed training exercises in it. They learned how to scroll and zoom the timeline, select video, and play video. They also learned how to delete segments of video and clean up the cuts. They performed a practice task in which they manually cleaned up a cut and a practice task in which they used the smart cut and paste.

In the second part of the study, each participant performed the two experimental tasks. For each task, participants were given one of two video compositions and a corresponding list of edits to be performed. One video composition was about the Mars polar lander; the other was about forest fires in Russia. The source videos for each task were obtained from CNN footage. The videos and shots to cut for each of the two tasks were selected to be equally difficult and to be entertaining.

Participants were asked to cut specific shots from the composition. After each cut, they were instructed to play the transition and, if they judged it to be "rough", to edit it to make it "smooth"— that is, with no gaps or abrupt transitions.

At the end, participants filled out a paper questionnaire about the study and demographic information. Users were asked about the system in general, the smart editing features, the tutorial, and each of the two tasks.

Results and Analysis

On average, both experimental tasks together took participants 33 minutes. There were no statistically significant differences in time to complete either task due to task, order, or smart editing.

We did find significant correlations between two questions on the questionnaire and the time to complete the task with smart editing disabled. Participants who rated "exploration of features by trial and error" as more "encouraging" than "discouraging" (on a scale of 1–9) took longer to complete the task (correlation 0.82, $p < 0.046$). This may be due to participants who felt encouraged to explore the interface spending time with parts of the interface that were not directly related to the task.

Qualitatively, nearly all participants made a positive comment about Silver, such as describing it as "cool software, very intuitive to use". Two users believed they experienced a learning effect; one said that one task was "easier because it was the second task". Another user mentioned, "after I could familiarize myself with it, it became easier to use and not as intimidating".

3 We would prefer the selection highlight to be green over fully selected words and light green over partially selected words, to be consistent with other views. Unfortunately, the Visual Basic text component does not support this.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2000
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

However, the times recorded show no significant learning effect.

We were very encouraged by the fact that all of the study participants were able to succeed in both of their editing tasks, especially given the brevity of the tutorial.

The purpose of the timelines and what they displayed were not immediately obvious to any of the participants. Participants recognized that one timeline represented the video, but did not know what the other two were. For all but 2 participants, the tutorial cleared up this confusion.

The existence of L-cuts in video was also not apparent to participants until they encountered a task in the tutorial where they deleted a shot that had an L-cut. The tutorial did not specifically talk about L-cuts, and it seemed that participants were not as watchful for them during the experimental task as they should have been. In retrospect, this is not surprising considering they were novices in video editing.

We believe that the timelines would have been easier to understand and users would have been more attentive to L-cuts if the tutorial had explained what clips, shots, and frames were and what L-cuts were conceptually before describing how the interface worked.

Some users had difficulty with the scrollbars at times because the thumbs were very small. It was difficult to grab the thumbs and move them and even more difficult to resize them. Other than that problem, we were pleased to see that users understood how to scroll and zoom the multiple levels.

We added a double border to individual frames to make it more apparent when individual frames were showing in the frame level. However, some users did not seem to be sure when individual frames were being shown and when they were not. The frame view may need an even more overt visual cue to make this distinction obvious.

Users felt disoriented after deleting a shot. When a shot is deleted, timelines in other systems typically shrink, whereas in Silver it maintains its size by readjusting its zoom level. This sometimes led to participants having difficulty locating its former position. A brief animation might make this explicit to users.

We expected to find a greater effect of using the smart editing options. The few video editing experts that have been involved with this project were enthusiastic about incorporating these features. Apparently, our naive users could not distinguish between the effects of the different options offered by the smart cut and paste dialog, possibly because in many of the L-cuts they encountered in the experiment, the gap in the audio or video track that was created was very small, which caused the different fixes to be only very subtly different from one another. Users did say that the schematic view of the video was intelligible and helpful. This suggests that we should automatically carry out the recommended action when the effect is small. Also, we should reevaluate the dialog box using video where the differences among fixes is clearly perceptible.

We suspect that participant motivation was an important variable in performance. Some users seemed to try to finish as soon as possible, while others experimented with the interface. Time to complete a task might not be the best measurement of the effectiveness of a creative authoring tool, since it penalizes exploring different options and paying attention to detail. This might explain the significant correlation described in the previous section. In the future, we hope to study the quality of the produced video which might provide richer information.

FUTURE WORK

There are many possible directions for future work with Silver. For example, corrections to the transcript by users could guide the speech recognizer in the text alignment, so users would not need to align the text manually. The transcript view could also support the authoring of new productions from scripts. The user could type or import a new script, and then later the system would automatically match the script to the video as it is shot.

We are exploring when these adjustments can be fully automated in a reliable way. We are also looking at other metadata that might be helpful in giving a recommendation. For example, we are considering mixing the audio of both edges when one audio track is music or other background sounds. We also want to increase the palette of options for handling gaps in the composition. For example, a third piece of video footage can be placed to cover the gap area, or a "special effect" like a dissolve can be used to smooth the transition over the gap.

Silver could also incorporate other metadata in the future, such as face detection and recognition, and voice matching and recognition. With these data, Silver could allow users to easily perform editing operations like "Cut all video where person X is talking."

Another type of analysis that would assist editing is object recognition and tracking. Silver could use this information to allow users to make moving objects hyperlinks, or cut and paste all video related to specific objects.

Tracking of people, faces, or objects would also allow interesting synthetic graphics to be overlaid, for example to explain motion in a video for physics instruction.

CONCLUSIONS

Silver is a system designed to make high quality video editing easier. It demonstrates several techniques to overcome some of the main difficulties of authoring in this domain. Namely, it uses multiple views that display not only the raw audio and video but also higher-level metadata. A single selection is coordinated across all views, even when the audio and video are unaligned. A textual transcript is linked to the video, providing a view that affords quick and accurate editing in the audio domain. Silver provides a three-level timeline, presenting both context and detail, which can be a useful tool for video editing.

Silver shows that video metadata can be used to create an advanced interface for aiding novice users in the performance of otherwise difficult or tedious editing tasks. It demonstrates three forms of smart editing, with selection, snap, and cut and paste. Currently the metadata used is only the clip titles, the transcript, and shot breaks. However, as more metadata becomes available, the smart editing features can be improved. For example, we could adjust the selection to match speaker changes or to avoid cutting in the middle of a panning or zooming camera operation.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2001
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The Silver project is investigating some exciting ways to make high quality video editing easier, and points the way to a future in which everyone can be as comfortable editing video as they are today editing text.

ACKNOWLEDGMENTS

We would like to thank our participants. The Silver Project is funded in part by the National Science Foundation under Grant No. IIS-9817527, as part of the Digital Library Initiative-2. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

1 Apple Computer, Inc. iMovie. http://www.apple.com/imovie/.

2 Adobe Systems Incorporated. Adobe Premiere. http://www.adobe.com/products/premiere/main.html.

3 Ahanger, G. and Little, T. D. C. "Automatic Composition Techniques for Video Production." IEEE Transactions on Knowledge and Data Engineering, 10(6):967-987, 1998.

4 Boreczky, J., Girgensohn, A., Golovchinsky, G., and Uchihashi, S. "An Interactive Comic Book Presentation for Exploring Video." CHI Letters: Human Factors in Computing Systems (SIGCHI), 2(1):185-192, April 2000.

5 Christel, M., et al., "Techniques for the Creation and Exploration of Digital Video Libraries." Chapter 8 of Multimedia Tools and Applications, B. Furht, ed. Kluwer Academic Publishers. Boston, MA, 1996.

6 Chua, T. and Ruan, L. "A video retrieval and sequencing system." ACM Transactions on Information Systems, 13(4):373-407, 1995.

7 Davenport, G., Smith, T.A., and Pincever, N. "Cinematic Primitives for Multimedia." IEEE Computer Graphics & Applications, 11(4):67-74, 1991.

8 Gauch, S., Li, W., and Gauch, J. "The VISION Digital Video Library." Information Processing & Management, 33(4):413-426, 1997.

9 Girgensohn, A., Boreczky, J., Chiu, P., Doherty, J., Foote, J., Golovchinsky, G., Uchihashi, S., and Wilcox, L. "A semi-automatic approach to home video editing." CHI Letters: Symposium on User Interface Software and Technology (UIST), 2(2):81-89, 2000.

10 Girgensohn, A., Bly, S., Shipman, F., Boreczky, J. and Wilcox, L. "Home Video Editing Made Easy -- Balancing Automation and User Control." In Human-Computer Interaction INTERACT '01. IOS Press, 464-471, 2001.

11 Hauptmann, A. and Smith, M. "Text, Speech, and Vision for Video Segmentation: The Informedia Project." In AAAI Symposium on Computational Models for Integrating Language and Vision, Cambridge, MA, Nov. 10-12, 1995.

12 Hauptmann, A.G. and Smith, M. "Video Segmentation in the Informedia Project." In IJCAI-95: Workshop on Intelligent Multimedia Information Retrieval, Montreal, Quebec, Canada, 1995.

13 Mackay, W.E. and Davenport, G., "Virtual Video Editing in Interactive Multimedia Applications." Communications of the ACM, 32(7):832-843, 1989.

14 Mills, M., Cohen, J., and Wong, Y. "A Magnifier Tool for Video Data." In SIGCHI '92 Conference Proceedings of Human Factors in Computing Systems, 1992, Monterey, CA: ACM, pp. 93-98.

15 Myers, B., Casares, J., Stevens, S., Dabbish, L., Yocum, D. and Corbett, A., "A multi-view intelligent editor for digital video libraries", in Proceedings of the first ACM / IEEE joint conference on digital libraries, 2001, pp. 106-115

16 Placeway, P., et al. "The 1996 Hub-4 Sphinx-3 System," in DARPA Spoken Systems Technology Workshop, 1997.

17 Schwartz, J., Rhinelander, T. and Dorsey, M., "Personal Rich Media Takes Off", The Forrester Report, Forrester Research Inc., October 2000.

18 Sony Electronics, Inc. MovieShaker. http://www.ita.sel.sony.com/jump/movieshaker/ms.html

19 Stevens, S.M., Christel, M.G., and Wactlar, H.D., "Informedia: Improving Access to Digital Video." Interactions: New Visions of Human-Computer Interaction, 1994, 1(4): pp. 67-71.

20 Ueda, H. and Miyatake, T. "Automatic Scene Separation and Tree Structure GUI for Video Editing." In Proceedings of ACM Multimedia '96, 1996, Boston.

21 Ueda, H., Miyatake, T., Shigeo Sumino and Akio Nagasaka. "Automatic Structure Visualization for Video Editing." In Proceedings of INTERCHI'93: Conference on human factors in computing systems, 1993, Amsterdam: ACM, pp. 137-141.

22 Wactlar, H.D., et al., "Lessons learned from building a terabyte digital video library.", IEEE Computer, 1999, 32(2): pp. 66-73

23 Zhang, H., et al. "Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution," in ACM Multimedia 95: Proceedings of the third ACM international conference on Multimedia, 1995, San Francisco, CA: pp. 15-24.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2002
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INTERNET ARCHIVE
Wayback Machine
http://blog.johndeutscher.com/category/smooth-streaming
Go
FEB MAR OC
28
2012 2013 20
5 captures
28 Mar 13 - 25 Oct 13

Category Archive

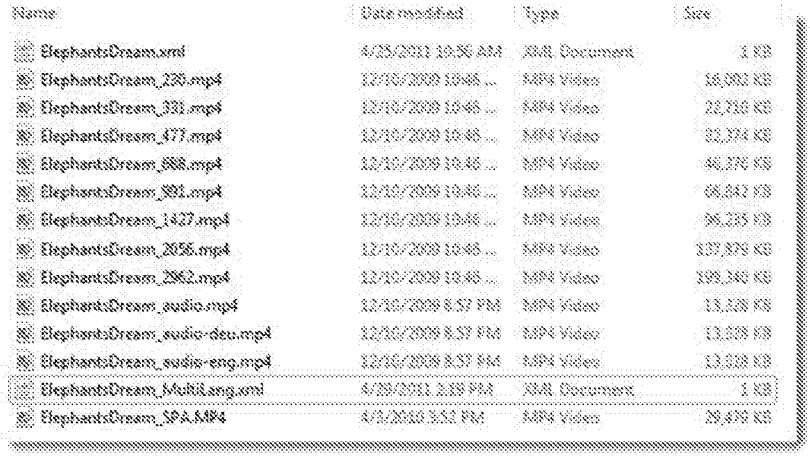
You are currently browsing the category archive for the 'Smooth Streaming' category.

IIS Transform Manager Beta – Using the MP4 to Smooth Task

April 29, 2011 in IIS Media Services, Media, Media Services, Smooth Streaming, Transform Manager | Tags: IIS, IIS Smooth Streaming, IIS Transform Manager, Media, MP4 To Smooth | 1 comment

With the latest release of the IIS Transform Manager Beta, we added a new "task" and Job Template to convert from standard MP4 files (non-fragmented) to Smooth Streaming format files (fragmented MP4). The benefit of this new task is that you can use a lot of existing encoders on the marketplace that allow you to generate regular MP4 files and then easily convert those to Smooth Streaming format for delivery in IIS Media services 4.0.

To use the new MP4 to Smooth Task you need to start out with a set of multi-bitrate encoded MP4 files that meet the requirements for Smooth Streaming encoding:



That means you can use your favorite encoder that can generate H264 and AAC MP4 files (there are a lot of those out there!) as long as you make sure to set them up to create GOP aligned (or Coded Video Sequence aligned for you fellow video nerds) files.

To do that you need to stick to some advanced settings in your H264 encoding tools:

- Make sure that if you switch framerates at lower resolutions, you only use EXACT half framerates of your upper resolutions (for example 29.97 and 14.985)
- Set you KeyFrame Min and Max distinance to 2 Seconds per your framerate
- Disable any forced I-frames at Scene cuts. Usually this is referred to as scene detection or –no-scenecut in x264
- If you are using X264, output your stats file from your first pass and feed it into your second pass.

For example, the following settings worked for me when I used X264 to generate MP4 files using an AVISynth script.

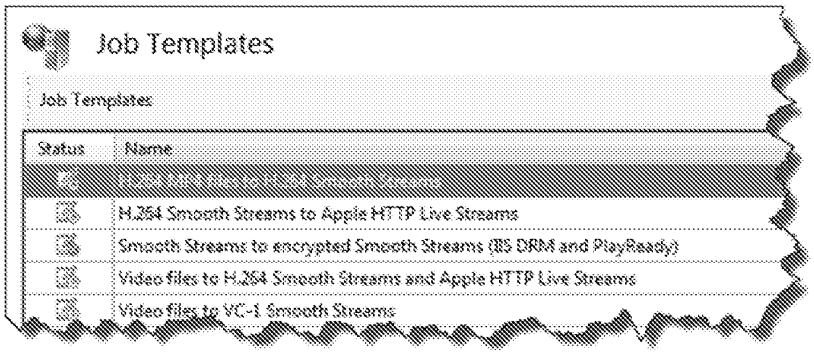
```
x264.exe -pass 1 -bitrate 2962-ssim -output NUL -profile high -preset
```

```
slower -tune film -stats ".stats" -keyint 48 -min-keyint 48 -no-scenecut
ElephantsDream.avs

x264.exe -pass 2 -bitrate 2962-ssim -output "MP4\%~n1_2962.mp4" -profile
high -preset slower -tune film -stats ".stats" -keyint 48 -min-keyint 48
-no-scenecut ElephantsDream.avs
```

After you have your folder of .MP4 files you can then use the new Watch Folder called "MP4 Video Files files to H.264 Smooth Streams" that is available in the IIS Transform Manager Beta.

Just **Enable** and **Start** this new Watch Folder after modifying the Folder Path that you want to use for it.



Next click **Explore Watch Folder** in the Actions menu (or right click and select Explore Watch folder). This will open the Watch folder up in Explorer view.
Before we drag our files into this view, we first need to create a "playlist" XML file in SMIL 2.0 format that will be used by Transform Manager to bind the MP4 files that you drop into the folder into a single Smooth Streaming package.

To do this, open up your favorite XML editor and create a basic SMIL 2.0 file with a <**body**> and <**seq**> element.
UPDATE:Watch out for whitespace characters or tabs in front of the <?xml> processing instruction or it will cause errors. Inside the <**seq**> element you list the <**video**> or <**audio**> tracks that you wish to use from your MP4 files.

You simply point the Video element's "src" attribute to your multiple MP4 source files and the audio element to the specific MP4 file that contains the audio track you want to use in the remux.

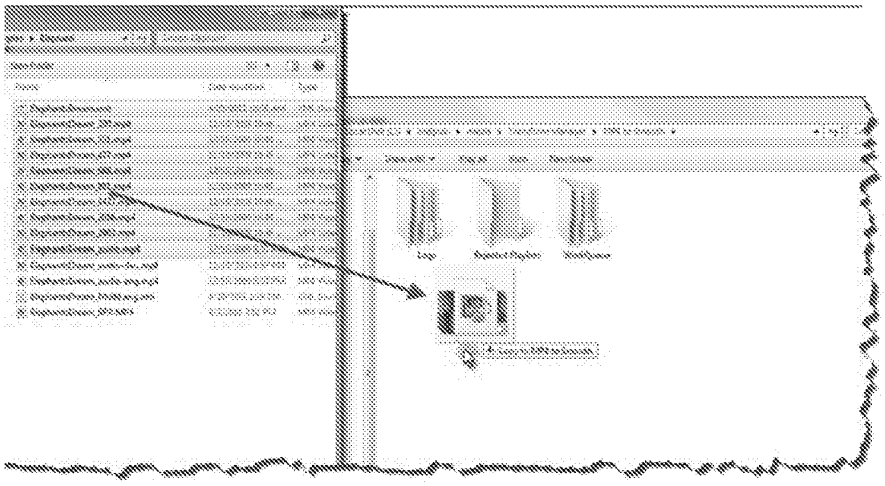
```
<?xml version="1.0" encoding="utf-8"?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<body>
<seq>
<video src="ElephantsDream_1427.mp4"/>
<video src="ElephantsDream_2056.mp4"/>
<video src="ElephantsDream_230.mp4"/>
<video src="ElephantsDream_2962.mp4"/>
<video src="ElephantsDream_331.mp4"/>
<video src="ElephantsDream_477.mp4"/>
<video src="ElephantsDream_688.mp4"/>
<video src="ElephantsDream_991.mp4"/>
<audio src="ElephantsDream_audio.mp4"/>
</seq>
</body>
</smil>
```

If you have multiple language audio files, you can also use the systemLanguage attribute on the <audio> element to set the language tag that will be used in the Smooth Streaming manifest. For example, the playlist below uses the systemLanguage attribute to add a

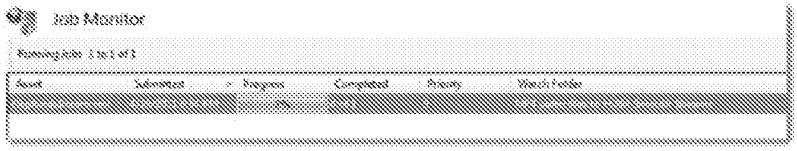
Spanish and English audio track.
This will generate two separate .isma audio tracks and the appropriate manifest entries in the Client and Server manifests.

```
<?xml version="1.0" encoding="utf-8"?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<body>
<seq>
<video src="ElephantsDream_1427.mp4" />
<video src="ElephantsDream_2056.mp4" />
<video src="ElephantsDream_230.mp4" />
<video src="ElephantsDream_2962.mp4" />
<video src="ElephantsDream_331.mp4" />
<video src="ElephantsDream_477.mp4" />
<video src="ElephantsDream_688.mp4" />
<video src="ElephantsDream_991.mp4" />
<audio src="ElephantsDream_SPA.mp4" systemLanguage="spa" />
<audio src="ElephantsDream_audio-eng.mp4" systemLanguage="eng" />
</seq>
</body>
</smil>
```

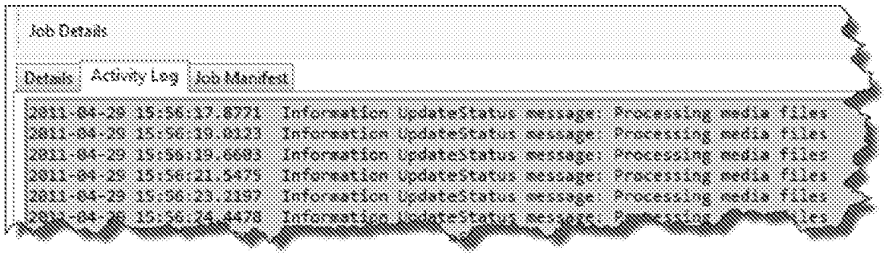
Finally, you drag all of your MP4 files and your new XML playlist into the Watch Folder to kick off the remux job.



After a short period of time the Job Monitor will show the progress of the conversion.

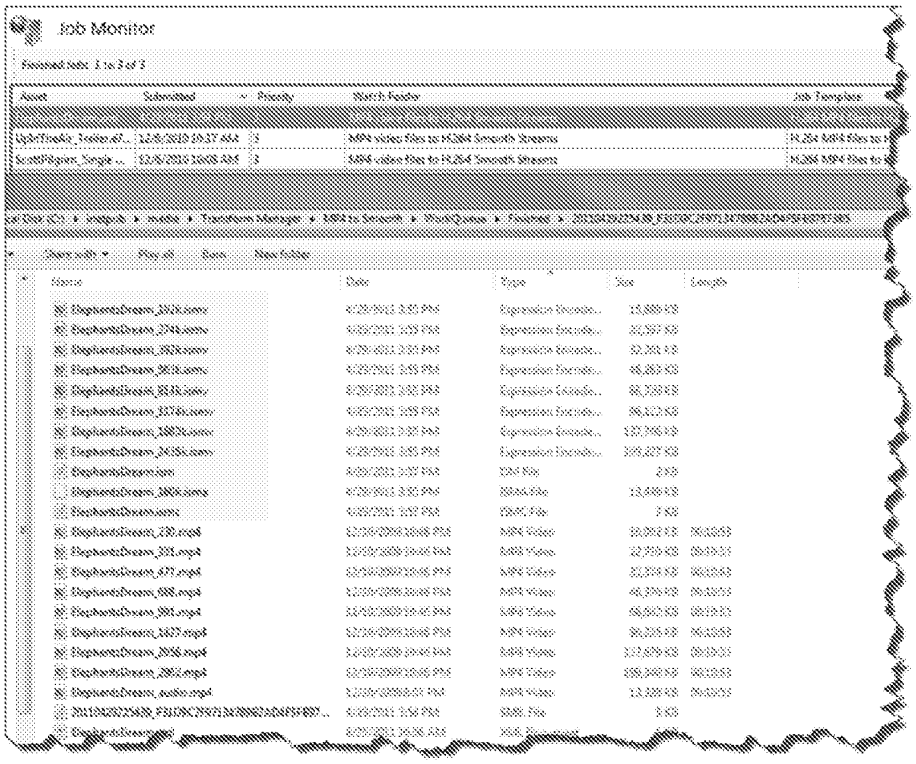


You can select the **Activity Log** tab in the **Job Details** panel to see the detailed progress log.



After the job has finished and you will be able to locate your job in the Finished folder in the Job Monitor view of Transform Manager.

Just double click on the row for your Finished job and it will open up the location of the Finished Folder for your Job ID that will contain your generated Smooth Streaming content. If you want to automate the copying of the output content to a final location on disk, or to another server you can add the RoboCopy Task to the default MP4 To Smooth Job Template and supply the location details on where you want your copy to go.



The new MP4 to Smooth makes it very simple to use a lot of existing encoding software that supports standard MP4 files with H.264 and AAC to generate Smooth Streaming content quickly and easily. This can also be handy when you need to generate MP4 files for HTML 5 progressive download playback in browsers, and also want to quickly and easily add Adaptive HTTP Streaming with Silverlight.

Troubleshoot Live Smooth Streaming Issues

March 15, 2011 in Smooth Streaming | Leave a comment

Sam Zhang is the Dev lead on the IIS Media Services team, and in that role he has the most exposure to troubleshooting and fixing issues with large scale deployments of IIS Media Services. His experience over the past few years of deploying numerous events, including NBC Sunday Night Football and the Olympics, have given him a lot of experience in figuring out what is going right and

wrong on the server.

I'm pleased to post a trackback to a collection of blogs that Sam just published that offer guidance on how to troubleshoot all kinds of issues on IIS Media Services! Erjoy.

Part 1 (Basics)

Part 2 (Publishing Point UI and States)

Part 3 (Event Logs)

Part 4 (Performance Counters)

Part 5 (Client Manifest)

Part 6 (IIS Logs and Others)

Part 7 (Putting It Together)

IIS Media Services 4.0 Released!

November 1, 2010 in IIS Media Services, Media, Smooth Streaming | 6 comments

I'm very excited to announce the release of IIS Media Service 4.0!

Some of the major features that we are anxious to share this week at the Streaming Media West conference include:

Encode once and stream to Silverlight, WP7, iPad, iPhone, Xbox.

Live Smooth Streaming updates:

- **Stability**: Improved stability and memory usage of Live Smooth Streaming under heavy server loads.
- **Low Latency**: Deliver near real-time broadcasts to Silverlight desktop clients. This feature is key for financial news customers, surveillance, e-learning, and on-line gaming (gambling) sites.
- **New Publishing Point Details UI**: Real-time publishing point statistics now available in the IIS Manager UI.
- **Manifest Compression**: Compress live-streaming manifests down to only a few lines to improve delivery of live 24/7 television for over-the-top services.
- **Updated RSCA API**: Real-time statistics available via RSCA API. (Blog post detailing this feature coming soon!)
- **Metadata delivery**: Deliver RDF metadata from encoder to the client creating unique player experiences.
- **Caching and Disk Performance**: Improved disk access and fragment caching.
- **Customer Focused Engineering**: New design changes reflect direct customer feedback and two years of customer experience with live broadcasting.

Apple HTTP Live Streaming: Real-time IIS Server packaging and delivery to iPhone, iPod, and iPad with the same encoded streams you use to reach Silverlight and Windows Phone 7 devices. New advanced capabilities for Apple mobile device deliver including:

- **Live DVR**: Pause, rewind, and seek.
- **Archiving**: Saves live content for DVR use or later on-demand playback.
- **Archive Segmentation**: Breaks a long broadcast up into smaller clips for easier storage and repurposing of on-demand assets.
- **AES Encryption**: Keeps your content safe while streaming by using a wire-level encryption. **Note**: This is not PlayReady for iOS, it's just a wire-line encryption of segments per the Apple Pantos spec.
- **B-Frame support**: Provides better picture quality when delivering to iPad.
- **HE-AAC audio support**: Higher quality audio for delivery to iPhone and iPad.

Smooth Streaming Client 1.5 – (Coming Soon!)

- Low Latency client support.
- Manifest compression client support.
-

Read our official release announcement here.

- Install and try IIS Media Services 4.0 today
- Stay tuned for other IIS Media releases in the weeks and months ahead…
  - IIS Smooth Streaming Client
  - IIS Transform Manager (available in Alpha today and through our TAP program).
  - IIS Smooth Multicast

Also be sure to check out the new article First Look: Microsoft IIS Media Services 4 on StreamingMedia.com.

IIS Transform Manager custom task for X264 encoder

August 16, 2010 in Smooth Streaming, Transform Manager | Leave a comment

I was really excited to see this recent blog post by Geoff Cox of SouthWorks. He has taken our IIS Transform Manager Task API and created an custom Task for using the open source X264 encoder via the CLI (command line interface).
This is very cool, as there are essentially two very popular H.264 encoding libraries available in the marketplace. One of them is a commercial product that is licensed via Main Concept, and the other is the X264 library which was originally developed by Laurent Aimar. If you check the Doom9 forums, there is a lot of discussion on which is faster and better. I won't comment there, and I would note that some of the comparisons up there are old now.

There are also a number of other ones out there built by professional encoder ISV's but the X264 one is very popular with open source encoding applications like HandBrake and FFmpeg. X264 is released under the terms of the GNU General Public License (so consider that a warning if you plan to use it – depending on your usage scenarios and legal requirements).

X264 is compatible with Smooth Streaming since it can generate closed GOP output by default. Only next step is to create multiple GOP aligned output streams at different bitrates and then remux them into fragmented MP4 format. We'll have an answer to the remux from MP4 to Smooth issue soon.

The exciting part about this post from Geoff is that it shows how simple it is to create a custom task for Transform Manager and integrate it into the workflow with IIS Media Services. Currently the task that he has could be extended to support FFMPEG also with some minor tweaking. It could also be used to later generate Smooth Streaming content directly via FFmpeg or X264 once we ship the new **MP4 to Smooth** conversion task that we are creating for the IIS Media Services 4.0 release in October.

Here's the direct link to the blog article by Geoff with the source code for the custom task.

http://blogs.southworks.net/geoff/2010/08/14/x264-encoding-with-iis-transform-manager/

IIS Smooth Streaming Format SDK Beta 2

June 18, 2010 in Silverlight, Smooth Format SDK, Smooth Streaming | Leave a comment

Today we released **Beta 2** of the **Microsoft IIS Smooth Streaming Format SDK 1.0**.

The IIS Smooth Streaming Format SDK provides application developers the capability to mux encoded video and audio elementary streams into Smooth Streaming fragmented-MP4 format that is compliant with the Smooth Streaming Format and Protected Interoperable File Format (PIFF) specifications. The IIS Smooth Streaming Format SDK includes a native C++ static library that can be linked into your applications to support the muxing of fragmented-MP4 into files or sent live via HTTP POST to a server running Internet Information Services (IIS) 7.0 and IIS Media Services 3.0 and IIS Media Services 4.0 Beta 1. The SDK is available for download here.

Documentation for the SDK can be found online here – IIS Smooth Streaming Format SDK MSDN documentation and the release

notes are available here – IIS Smooth Streaming Format SDK Beta 2 Release Notes.

The primary purpose of the IIS Smooth Streaming Format SDK is to enable developers to create applications that can generate PIFF compliant Smooth Streaming formatted fragmented-MP4 files for use in video-on-demand and live streaming scenarios. In addition, the SDK can be used to encrypt content using standard AES encryption as required by the PIFF specification (this SDK only supports the PlayReady specific protection headers).

It is expected that the video and audio encoding functionality is done externally from the SDK. Encoding for VC-1 can be accomplished by using the Microsoft VC-1 Encoder SDK – Professional. If you wish to do H.264 encoding, you will need to acquire a 3rd party H.264 encoding SDK and AAC audio encoder. There are lots of encoding library choices available both free and commercial.

The components of the SDK include:

1. A static-linked packaging library ssfsdk.lib, along with appropriate header files, that delivers f-MP4 wrapping capability to an application for use with the following video and audio codec combinations:
    - Closed GOP encoded VC-1 with Elementary Stream Sequence Headers and WMA Pro, or WMA audio
    - H.264 (AVC1 closed-GOP streams only with IDR frames at the first sample of a GOP – must not be an Annex B stream. PPS and SPS NAL units are not supported in the stream)
    - AAC-LC audio
2. Sample source code for a basic on-demand muxing application that uses DirectShow to source from files.
3. Link to online MSDN documentation.

Updates included in the Beta 2 release of the Smooth Streaming Format SDK are:

- H.264 PlayReady sub-sample encryption support.
- PIFF 1.1 spec compliance.
- Multi-language audio muxing.
- Text stream and TTML track muxing support.

This is in addition to the following supported features:

- Support for Protected Interoperable File Format (PIFF) 1.1 compliant fragmented-MP4 file output.
- Support for ISO Base Media (ISO/IEC 14496-12:2008) spec compliance.
- Support for muxing live and on-demand content.
- Support for appropriate header boxes and formatting required for live streaming using IIS Media Services.
- Support for AES-CTR encryption of VC-1 and H.264 encoded content for use with PlayReady licensing servers and Silverlight 4.0 or higher.
- Support for writing out a compliant Smooth Streaming Transport Protocol server manifest files.
- Support for writing out a compliant Smooth Streaming Transport Protocol client manifest files.

NOTE: Beta 2 includes a **"Go-Live"** license if you would like to use this SDK in production applications.

If you have questions on how to use this SDK in your applications, comments, or feedback on the SDK please send them to me directly or to smoothd@microsoft.com.

Download the Smooth Streaming Format SDK Beta 2 here:

SSFSDK_64.msi

SSFSDK_32.msi

Resources

- Download the Protected Interoperable File Format specification

- Download the IIS Smooth Streaming Transport Protocol Specification
- Download the Smooth Streaming Client 1.0
- API documentation for Smooth Streaming Client
- More Information on Smooth Streaming
- More Information on Live Smooth Streaming
- Supporting documentation on getting started with Smooth Streaming
  - Smooth Streaming for IIS 7.0 – Getting Started
  - Live Smooth Streaming for IIS 7.0 – Getting Started
  - Smooth Streaming Deployment Guide
  - Building your first Smooth Streaming Player using SSME

IIS Media Services 4 Beta 1 – Encode once & deliver to (more-than) three screens across multiple file formats & protocols

June 10, 2010 in Smooth Format SDK, Smooth Streaming, Transform Manager, Uncategorized | Leave a comment

Today begins the next wave of updates to IIS Media Services. We released our first beta of Media Services 4 which includes the new support for live and on-demand streaming to Apple devices including the iPod, iPhone, and iPad using the same encoded elementary streams that are targeted at your Silverlight clients. We have also added a number of improvements to IIS Live Smooth Streaming to Silverlight. In this post, I'll quickly cover all of the new stuff we released today, point you to our new walkthroughs and give you a preview of the upcoming features and more in-depth walkthroughs we are planning.

IIS Media Services 4 Beta 1- Now with iPhone and iPad streaming support!

IIS Smooth Streaming has been the leading platform for broadcasting live events using adaptive HTTP streaming on the internet. It was first used to deliver the 2008 Summer Olympics, and more recently was used for the 2010 Winter Olympics, March Madness, and the 2010 French Open.

The new capabilities of Media Services 4 will extend the reach of these live events to even more devices using the same set of encoded elementary streams. We can now encode a spectrum of bitrates in H.264 and AAC-LC in IIS Smooth Streaming format, push them to the IIS Server and deliver out to Silverlight clients on Mac, PC and Linux, Windows Phone 7, iPhone, iPad, iPod touch, Nokia s60, Broadcom and Intel set-top-boxes. That's an incredibly wide range of screens to hit with a single set of multi-bitrate streams! IIS Media Services will simply unwrap the incoming fragmented-MP4 containers, extract the encoded H.264 and AAC-LC elementary streams, and re-packing them inside MPEG-2 Transport Stream containers for delivery to Apple devices. This means that you only encode the content once, and IIS handles the "packaging" of that content for distribution to multiple endpoints. For you Seinfeld fans out there, you can think of this process as "re-gifting" to your Apple friends, although this is definitely not a "label maker". I'll be posting guidance soon on how to broadcast to all of these screens using a range of encoding settings and bitrates.

Expression Encoder 4 Pro, which launched earlier this week and will be available for purchase at the Microsoft online store, also includes new templates for broadcasting live H.264 streams to IIS Media Services for delivery to multiple device profiles including the iPhone, iPad, and Nokia devices. For an introduction to Live Smooth Streaming with EE4, check out Sam Wookey's blog post and walkthrough.

New! Transform Manager

In addition we are launching a new feature of IIS Media Services called Transform Manager, which provides simple integrated video encoding and batch conversion of video files to IIS Smooth Streaming format and the MPEG-2 Transport Stream adaptive streaming format supported by Apple devices. Transform Manager also integrates directly with Expression Encoder 4 Pro to allow you to use "watch folders" to queue up multiple encoding jobs on your server and deliver them directly to IIS Media Services 4 web folders as IIS Smooth Streaming presentations, complete with Silverlight player templates.

Transform Manager can also convert your IIS Smooth Streaming presentations directly into MPEG-2 Transport Streams and .m3u8 manifests for delivering content to Apple devices. We ship Transform Manager with several example Job Templates and Watch Folder configurations that enable you to drag-and-drop WMV and MP4 files into watch folders (or set up FTP or WebDav on your

IIS server to upload content to your watch folders) and Transform Manager will automate the encoding to Smooth Streaming file format using Expression Encoder 4 Pro, transmux the output from EE4 to MPEG-2 Transport stream and create .m3u8 playlists, and then copy your generated media files to a uniquely named IIS web site for immediate playback.

We have also released a new SDK with Transform Manager so that you can write your own plug-in "tasks" that can be used with Transform Manager to support custom actions, third-party encoders, or your own encoding applications.

This alpha release of the Transform Manager feature currently only supports operating on a single IIS server and uses the local Task Scheduler for job management. In our future releases, we will be expanding the capabilities of the scheduler to support scaling-out Transform Manager.

IIS Smooth Streaming Client 1.0 RTW

Also released today is the IIS Smooth Streaming Client 1.0 RTW. This is the development kit that most of you already refer to as the SSME (Smooth Streaming Media Element). It is the core component that enables Smooth Streaming support for Silverlight clients. The Silverlight Media Framework 2.0 is built on top of this component. The new client development kit enables developers and designers to custom Silverlight players from the ground-up. You can build custom players that support full DVR-style time shifting, live ad insertion, multiple camera angles, closed captioning, and custom metadata tracks.

If you use the new Silverlight Media Framework (smf.codeplex.com) you get all of theses great features built into an open source player that you can extend upon. The new SMF 2.0 player has a more modular architecture, supports Timed Text Markup Language text tracks, support for VAST and MAST, support for the Microsoft Silverlight Analytics Framework and a plug-in API .

IIS Smooth Streaming Format SDK 1.0 Beta 2

And there is more! The IIS Smooth Streaming Format SDK 1.0 Beta 2 will be released to the web next week. This SDK is provided as a C++ static library for building in support to your own encoder applications for muxing IIS Smooth Streaming and PIFF 1.1 compliant streams, and also delivering to IIS Media Services. This update includes bug fixes to the Beta 1 features plus the following additional features:

- Muxing of properly-encoded VC-1 and H.264 video to PIFF 1.1 compliant output for live and On-demand.
- Creation of PIFF 1.1 compliant manifest files
- Support for PlayReady sub-sample encryption to support encrypted H.264
- Multi-language audio muxing support
- Text-streams to allow for creation of Timed Text Markup Language tracks

Coming soon…

Even with all of these great releases and new features, the team refuse to stop for breath (hey, there is no Summer in Seattle this year anyways, so let's just keep working on new features!). Coming later this year you will see the following additional features added to IIS Media Services.

- The Apple HTTP streaming support will expand to include AES encryption, DVR "sliding window" support and segmented archives similar to the standard Smooth Streaming support in Media Services to bring parity across both features. We will be doing the work necessary to make sure that everyone can broadcast to all their devices 24/7 if desired.
- Transform Manager scale-out will be supported through integration with a new scalable scheduler architecture. We will also be expanding our API to support integration with the task editing user interface to allow custom task developers to implement their own task setup and editing user interfaces.
- **More Walkthroughs**! I'll be working on expanding our walkthroughs for both the iPhone/iPad broadcasting, Transform Manager API usage, Transform Manager integration with 3rd party encoders, using TTML, multi-language audio support, and new metadata support in Media Services.

Resources

Get the new IIS Media Services bits now using the Web Platform Installer link here!

New Walkthroughs:
Transform Manager Walkthrough
Live Broadcasting to iPhone and iPad with IIS Media Services Walkthrough

For more details on this release of IIS Media Services, check out the following links:

- IIS Media homepage
- Transform Manager homepage
- Media Services help documentation on TechNet
- Transform Manager help documentation on TechNet
- On-Demand Smooth Streaming
- Live Smooth Streaming
- Advanced Logging

NAB 2010 and IIS Media Services

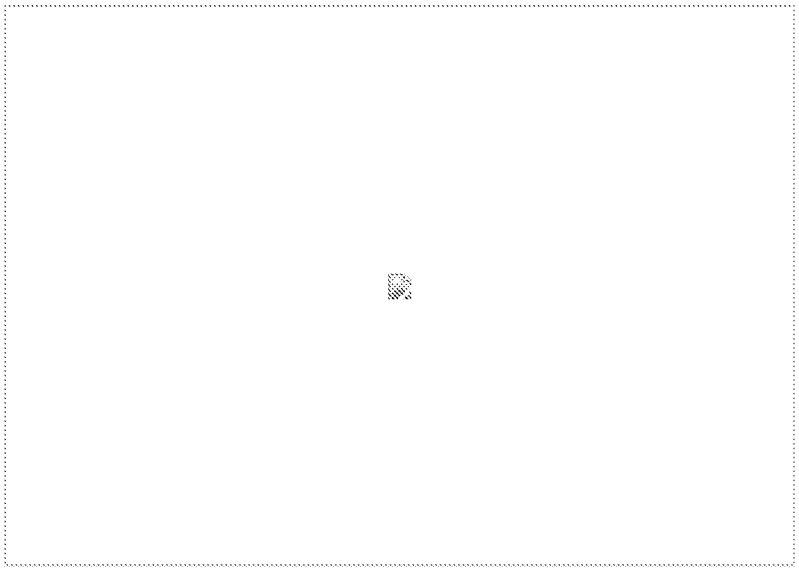
April 17, 2010 in Silverlight, Smooth Format SDK, Smooth Streaming, Uncategorized | 3 comments

It was a great week at the National Association of Broadcasters (NAB) convention in Las Vegas this year. It's always fantastic to run into the extended family of friends, partners, and customers from around the world. We had a lot of great new technology announcements and demonstrations this year and too many partner announcements to cover in a quick blog post, but here is a recap of the highlights.

IIS Smooth Streaming Client and Silverlight Media Framework (SMF) – showcased demos with Multiple audio languages and subtitles using SMF and the demos worked flawless. We demoed how you could build a player with less than 10 lines of code using SMF. In addition the latest version of the SMF player now supports Timed Text Markup language (TTML) for displaying captions and subtitles. We now support "textstream" tracks that work similar to the fragmented MP4 video and audio, but contain caption text in TTML format. We also demonstrated multi-language audio support. You can now seamlessly switch between audio tracks in the player. We also announced several new encoding partners that are adopting the Smooth Streaming Format SDK, and several partners who have integrated TTML caption conversion from 608/708 and Teletext, multi-language audio, and ad-insertion using SCTE-35 signaling via sparse stream events.

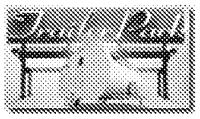
The Smooth Streaming Format SDK Beta 1 is available as of early March this year, and we will be shipping Beta 2 at the end of May which includes additional support for multi-language audio, Text Streams, and H264 PlayReady sub-sample encryption (per the PIFF 1.1 specification).

IIS Smooth Streaming in 3D – this demo was a crowd puller. It may be the colored 3D glasses but people loved watching 3D content in a browser and the same content was also demoed on a TV connected to a set top box running Silverlight. This demo was done working very closely with Level 3. Alex Zambelli, who put this demo together, has more details here.

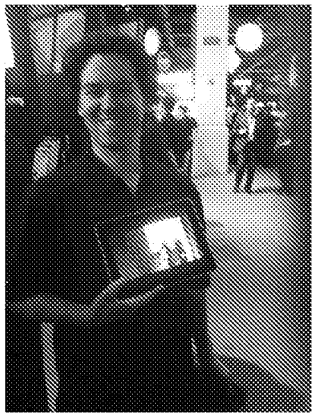


The Silverlight Rough Cut Editor tool – I've talked about this project in the past. This was originally a project that I started on the IMM team for use as a web part in SharePoint. Since the IMM project was closed down, we have kept revising this tool and used it in many Live events including the Winter Olympics. Now the RCE code is available for you to download from the Code Gallery site on MSDN. You can create a new source by combing portions of multiple individual sources (including live Smooth Streaming sources) in a matter of a few minutes. The tool is really easy to use and super useful in creating highlights. It's also easy to customize and extend since it is based on the PRISM framework. You can integrate the tool easily into your own MAM and DAM or custom database systems to create an end-to-end workflow.

Microsoft Silverlight Analytics Framework (MSAF) and visualizations of data using Pivot / Azure (Silverlytics) – SAF already integrates top analytics providers and enables easy analytics integration for Silverlight applications including the ones that use IIS Smooth Streaming. The folks at Location 3 Media used MSAF and Microsoft Pivot to create some super cool visualizations for real-time analytics data.



Silverlight Enhanced Movie Framework – this framework allows for rich Blu-ray like experiences with offline Silverlight apps. The demos included the movie "The Hangover" among others at the booth. This will be the future of movie delivery for me. I don't even own a Blu-ray player because I prefer not to collect plastic landfill discs!



At the booth we demoed live IIS Smooth Streaming to an iPhone and iPad. The streams were generated by Expression Encoder 4.0 that was sending 10 different bitrates to multiple clients. This new feature allows you to enable on-the-fly transmuxing at the IIS Media Services publishing point. All you need to do is send the usual Smooth Streaming fragmented-MP4 streams to the publishing point in H.264/AAC at the right profiles and we will re-wrap to MPEG2 Transport Stream and segment the video on the fly. The same streams can also be consumed from the publishing point in their fragmented MP4 format using Silverlight on multiple clients, including Mac, Windows, Linux, Windows Phone 7, Nokia S60, and the Broadcom and Intel set-top boxes. This feature is going to be included in IIS Media Services 4.0 Beta 1 in June.

IIS Smooth Streaming for Windows Phone 7 – using the exact same streams that we are sending to

the iPad we are able to deliver to Windows Phone 7, which is now running Silverlight and a version of the Smooth Streaming Media Element. The Windows Phone 7 is capable of decoding H264 and AAC at even higher profile settings than Apple's IPad recommendations, and we should see some even higher quality presets coming soon for Expression Encoder 4.0.

IIS Smooth Streaming for Nokia S60 devices

Expression Encoder 4 (EE4) – we demoed encoding and delivering HD content using IIS Smooth Streaming and EE4 running on the same machine. EE4 is going to be a great tool when it ships. It now has support for Live Smooth Streaming! We had a 24 core Intel server streaming 10 bitrates of Live Smooth Streaming (from 6Mb down to 200kbps) along with a second instance of EE4 running with 3 streams targeted at the IPad profile settings. Fantastic!

In addition to all of this great news, we announced that Silverlight adoption has reached 60% globally, on Tuesday just down the street from the NAB convention we launched Visual Studio 2010, and on Thursday we launched Silverlight 4.0.

Here is the full Microsoft press release for more details.

I'm looking forward to seeing everyone at IBC in Amsterdam at the end of the Summer with a lot more cool demos! We have some fantastic stuff in the works.

New Composite Manifest support for building Rough Cut Editing (RCE) scenarios with Smooth Streaming.

January 22, 2010 in Silverlight, Smooth Streaming | 3 comments

Beta 2 of the IIS Smooth Streaming Player Development Kit (SSPDK) contains a very important Silverlight control called the **SmoothStreamingMediaElement.** This control is the core client side component for making Smooth Streaming work in Silverlight.

In the latest release, Vishal Sood and team have added in support for a new manifest format called the Composite Manifest.

This new manifest can be used in scenarios where you want to do one of the following:

- Create a new video clip that is composed of many smaller sections of existing content.
- Create a single or series of highlight clips from a very long clip (which could be the archive of a live event).

I view this new Composite Manifest like a new Edit Decision List (EDL) format since I have a background in the broadcast and post production world. This new manifest allows me to create a cuts only EDL file that can be used by the player to dynamically generate new clips without going through the process of re-assembling, re-encoding, or trans-muxing assets.

This new manifest enables really cool scenarios where you can create Edit Decision Lists (EDLs) on the fly in a web based RCE tool (more coming on this later…) and generate new assets from existing ones without touching the content.

Pay close attention to the fact that even though your Smooth Streaming files may have chunks that are 2 seconds in duration, the new Composite Manifest lets you start and end your clips *within* the boundaries of that chunk. So you are not restricted to making edits at the fragmented-MP4 chunk boundaries.

Read up on the new Composite Manifest format and how to define Clips here:

http://blogs.iis.net/vsood/archive/2010/01/22/composite-manifest-support-for-rough-cut-editing-scenarios-in-ssme.aspx

IIS Smooth Streaming Player Development Kit Beta 2 Released

January 22, 2010 in Silverlight, Smooth Format SDK, Smooth Streaming | Leave a comment

Vishal Sood posted a new blog entry on the latest release of the **Smooth Streaming Player Development Kit Beta 2** last week.

Read all about the new features in the Player SDK, and download the bits here: http://blogs.iis.net/vsood/archive/2010/01/15/iis-smooth-streaming-player-development-kit-beta-2-released.aspx

Key features include:

- Basic Playback controls
    - APIs such as Play, Pause, Stop, etc.
    - Events for Playback and Diagnostics
    - Properties to track position, etc.
- Advanced Playback support
    - DVR support for Live Smooth Streaming
    - Support for keeping track of Live event while in DVR (*__new__ in Beta 2*)
    - Trick Play: Slow Motion
    - Trick Play: Fast Forward / Rewind (*__new__ in Beta 2*)
- H.264 / AAC support (*new in Beta 2*)
- Monetization
    - Ad Playback integration – scheduling capabilities, tracking Ad progress
    - Live Ad Insertion with Live Smooth Streaming
    - Rich Analytics with IIS Advanced Logging
    - Support for specifying markers/Ad ingestion point outside the main manifest (*__new__ in Beta 2*)
- Content Protection – PlayReady integration for VC1 content
- Composite Manifest Support for scenarios like Rough Cut Editing (*__new__ in Beta 2*)
- Selecting Tracks for playback (e.g., restrict the bit-rates available, support multiple camera angles in a single stream, etc.)
- Support for progressive download Ads/content
- SSME is used in Silverlight Media Framework(SMF) and the partnership helps facilitate faster player development

The **Silverlight Media Framework** has also been updated to version 1.1 which includes the latest **SmoothStreamingMediaElement** from Beta 2 of the Smooth Streaming Player SDK. This includes support for progressive download content in addition to Smooth Streaming format support.

Get the latest drop of the **Silverlight Media Framework 1.1** on Codeplex: http://smf.codeplex.com/

Adding Netflix-like thumbnail scrubbing preview to the SMF Player

January 22, 2010 in Media, Silverlight, Smooth Streaming | Leave a comment

Michael Scherotter just posted a great sample of creating Netflix style 'zoetrope' player thumbnails in Silverlight.

His sample uses the latest version of the Silverlight Media Framework that is posted at http://smf.codeplex.com, and Expression Encoder 3.0.

We are looking into adding in support for embedded "Zoetrope" tracks to the Smooth Streaming Format SDK and Silverlight Media Framework in a later release.

Article: http://blogs.msdn.com/synergist/archive/2010/01/22/scrubbing-with-thumbnails-with-the-smf-player.aspx

Demo: http://xmldocs.net/smf`

Source Code: http://code.msdn.microsoft.com/SMFThumbnailScrub

Follow
FOLLOW "JOHN DEUTSCHER"

Get every new post delivered to your Inbox.

Enter your email address

Sign me up

Powered by WordPress.com

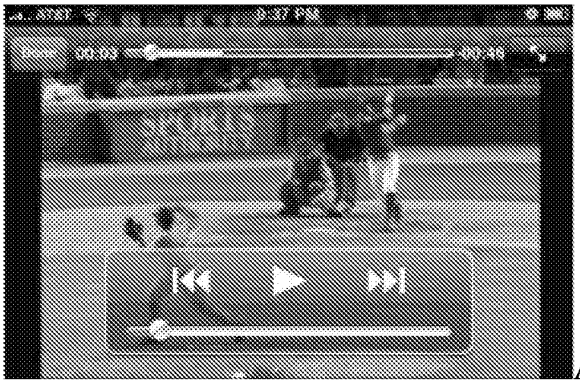
The Lowdown on Apple's HTTP Adaptive Bitrate Streaming

By Liz Gannes Jun. 10, 2009, 12:01am PDT 3 Comments

15

Like Be the first of your friends to like this.

This week Apple embraced adaptive bitrate streaming over HTTP at its developer conference in San Francisco. It will be available with the new iPhone OS 3.0 on June 17 and in September with the new version of QuickTime for Mac OS X Snow Leopard.



After the Apple presentation Monday, we chatted with one of only a couple developers who was given early access to Apple's HTTP streaming. He gave us the lowdown on how Apple will implement the technology.

First, a bit of a refresher. Adaptive bitrate streaming, brought to the fore by Move Networks and now being pushed by Adobe, Microsoft and Apple, is one of the more interesting things going on in video today. I just finished a research report for our paid GigaOM Pro service on the topic and I'd love for you to check it out (only $79 for unlimited access). Here's a bit from my intro describing the concept:

> There's no one-size-fits-all approach to clean up [problems with video quality and delivery], especially because many of them result from complications and hang-ups on the video watcher's computer or Internet connection. That's why many companies are turning toward adaptive bitrate streaming, a technique of detecting a watcher's bandwidth and CPU capabilities in real time and then adjusting the quality of a video stream. That requires encoding a single video at multiple bitrates and switching to the most appropriate one on a moment-by-moment basis. The result: very little buffering, fast start time and a good experience for both high-end and low-end connections.

One contextual note about Apple's move to use HTTP for adaptive streaming is that it paints Adobe into a corner. Adobe does have its own "dynamic streaming" for Flash Media Servers, but that requires buying special servers and switching between streams within a single real-time session. HTTP delivery treats video files like any other file and transfers them in chunks through individual sessions. Since no special servers are required, HTTP streaming is seen as less expensive than traditional streaming, though in some cases it may be less secure or reliable.

Apple invited two outside developers, Inlet Technologies and Envivio, to get early access to its new streaming video plans. We spoke with John Bishop, SVP of strategy and business development at Inlet (pictured), to get the lowdown on Apple's new technology. Raleigh, NC-based Inlet's Spinnaker appliance enables live video delivery, and the company is taking part in most every sports streaming deal out there right now. (Though Bishop wouldn't comment, it's all but expected that Major League Baseball will update its iPhone application with live video when the new OS comes out; the league's subscription live streaming product is powered by Inlet and Swarmcast.)

While Bishop's company isn't attached to any one format, it's clear he thinks HTTP streaming is the future. "It's the first time video works for the web as opposed to the web working for video," he said. Bishop said he thought Adobe would continue to succeed with video on demand but that others may gain ground for live streaming.

Apple's new HTTP streaming will also support live and on-demand H.264 video play directly within a browser. That means companies will no longer have to make and sell branded applications, as MLB and TV.com do today. It also means a lessening of the impact of AT&T meddling with approval of applications that try to stream over the 3G wireless network. Today iPhone apps from video providers like Sling, Poptiq and Joost are Wi-Fi only. Clearly, web pages loaded in browsers have no such gated approval process. Provided that increased video consumption doesn't cripple the network, 3G would make video experiences on the iPhone far more interesting — when you're limited to Wi-Fi only you might as well just open your laptop.

Then, after video publishers have encoded their content in multiple versions for adaptive bitrate streaming, they'll feed the files to Apple's Segmenter, a new piece of Mac desktop software that cuts video into small chunks. "It's like a ginsu knife," Bishop said.

Then, when a video is playing, Apple will check periodically to see which bitrate chunk it should serve to the viewer. Bishop said other providers like Move, Swarmcast and Microsoft are a bit more diligent about this than Apple will be. They check every 2 seconds to see how fast the last 2 seconds were received and pump out the next chunk accordingly. Apple won't be quite so optimized.

Bishop said Inlet has been bombarded with requests to get new live adaptive streaming products out for the upcoming iPhone release — and after that, for the desktop release in September. Which can only mean good things for those of us who like to watch video on our computers and on the go.

Photo of iPhone MLB app from 2008 by Flickr user artifishall.

Do you like this story?

Like Be the first of your friends to like this.

- 15

GigaOM Research

Can Social Media Save the Music Business?
David Card in NewNet
A Tale of Two Vendors: Will Oracle, Red Hat Earnings Converge?
Derrick Harris in Infrastructure
Why the Mobile App Industry Needs to Address Privacy — Now
Colin Gibbs in Mobile

3 Comments

Subscribe to comments
Name
Email

Site

Connect with WordPress.com

Comment

SUBMIT

Notify me of follow-up comments via email.

1. Reply
Scott Wednesday, June 10 2009

This was very useful information. I am "just" an owner of the iPhone and a big video fan. I am not in the video industry at all. But, I found this site though multiple links/hops. Over on the iPhone forums, there is very little discussion going on about the HTTP video streaming capabilities. I am hoping for considerably improved content delivery on the iPhone. If Netflix jumps on board, then life will be very wonderful. :) There is a rumor that Hulu is looking to get on the IPhone. Maybe this HTTP thing will help push them over the edge?

Anyway – thanks again for an informative article.

Scott

2. Reply
Morgan Sunday, August 16 2009

Folks,

You have to see this to believe it. Widevine was showing the HD Adaptive streaming at Cablelabs last week.

It was amazing to see. The clip was 1080i it was encoded in H.264. It seamlessly adapted from 2.5 Mbps to 10Mbps …I think there were 4 or 5 different bit rates. The quality was amazing. It was playing back on Windows, Mac and a Blu-ray player.

What was even more amazing then the HD adaptive technology was that the 1080i Mbps H.264 clip played back on the PC/MAC. They attempted to play back the low bit rate version using a Flash player and silverlight and the CPU maxed out, stuterd and lost frames.

Sonic and Widevine really have something here.

- Morgan

3. Reply
Vittorio Palmisano Tuesday, March 16 2010

Hi all,
we are developing a similar solution in order to enable adaptive http streaming for flash players. Here you can find a simple demo:

http://www.quavlive.com/streamingdemo

24 Trackbacks

1. *iPhone Video, Finally! Plus HTTP Streaming and Over-the-Air Movie Downloads* June 10, 2009Tracked on

   [...] The Lowdown on Apple's HTTP Adaptive Bitrate Streaming on June 10th, 2009 at 12:02 am – Permalink [...]

2. *What to read on the GigaOM network* June 10, 2009Tracked on

   [...] Staff | Wednesday, June 10, 2009 | 10:50 AM PT | 0 comments The lowdown on Apple's HTTP adaptive bitrate streaming (NewTeeVee) Measuring success as a freelancer (WebWorkerDaily) On desktop Linux, the Mac, and market [...]

3. *The Lowdown on Apple's HTTP Adaptive Bitrate Streaming* June 10, 2009Tracked on

   [...] The adaptive bitrate aspect of the streaming is at the forefront of what other players like Adobe and Microsoft are doing in video. Through new Mac software called Segmenter, video publishers will divide their content into tiny chunks of different qualities that are subbed in for each other when a user's bandwidth and CPU constraints improve or degrade. That means improved video start times, no buffering, and streaming at the highest quality possible in any situation. Read the rest of this article on NewTeeVee → [...]

4. *Would you compromise on your TV picture? « Curiously Persistent* June 14, 2009Tracked on

   [...] a technique called adaptive bit rate where the quality of a stream varies according to your broadband speed (though this could result in [...]

5. *Wimbledon on the Web: Smooth HD Comes to US Sports - TDG Opinions* June 22, 2009Tracked on

   [...] on with Smooth and examined the efforts behind the recent French Open, Indy 500 (non-Smooth), and Apple's adaptive bitrate streaming for the new iphone. When it comes to a high quality web viewing experience, adaptive streaming is [...]

6. *Video: See Apple's HTTP Adaptive Video Streaming in Action* June 30, 2009Tracked on

   [...] Technologies and Envivio, in on its plans so they could prepare solutions for content providers. (Here's an interview with an Inlet executive about how exactly Apple's HTTP streaming works.) [...]

7. *Akamai to Make iPhone Video Streaming Smooth* July 2, 2009Tracked on

[...] Stacey Higginbotham | Thursday, July 2, 2009 | 1:29 PM PT | 0 comments Akamai today said it would provide adaptive bit-rate streaming to deliver video content from web sites to the Apple iPhone 3G and devices running the iPhone OS 3.0 operating system. Basically, using adaptive bit-rate streaming means folks can watch streaming video on their iPhones or iPod Touches with fewer stops and starts. Adaptive streaming adjusts the video content to a lower or higher bit rate, depending on how robust the web connection is. Akamai offers a similar service for Microsoft's Silverlight for video on PCs. Adobe Flash and Move Networks also offer adaptive bit-rate streaming, although Adobe uses a proprietary method that requires special servers. [...]

8. *Paul Artz (pragmat) 's status on Saturday, 04-Jul-09 17:23:00 UTC - Identi.ca* July 4, 2009Tracked on

   [...] http://newteevee.com/2009/06/10/the-lowdown-on-apples-http-adaptive-bitrate-streaming/ [...]

9. *Remembering Neal Page - TDG Opinions* July 8, 2009Tracked on

   [...] Through the efforts to stream the Indy 500, Tour de France, Wimbledon, and to bring video to the iPhone, it seems whenever video moves forward Inlet is there and Neal has been their leader. The company [...]

10. *Cuma Ini - Watch Apple's first-ever Live Concert on your iPhone Tonight at 9 pm PT* August 7, 2009Tracked on

    [...] If you are interested in finding out more on how the HTTP Adaptive Bitrate Streaming works then you can checkout this post by Gannes. [...]

11. *Apple Streams First Ever Live Concert to the iPhone Tonight at 9PM PST | The iPhone Guru* August 7, 2009Tracked on

    [...] Live]via[iPhoneHacks]via[NewTeeVee] Share and [...]

12. *Apple transmitió en vivo su primer concierto para iphone. | Tecnota* August 10, 2009Tracked on

    [...] y pareja en diferentes tipos de condiciones de la red. Puedes leer el como este protocolo trabaja aquí y si quieres ver un vídeo de demostración lo encuentras [...]

13. *Home Shopping Network Hits the iPhone* August 17, 2009Tracked on

    [...] appliance. Spinnaker supports Apple's HTTP adaptive streaming, a hot topic we've been following closely. As Liz wrote earlier: HTTP streaming enables publishers to give users a better video experience by [...]

14. *Adaptive Bitrate Streaming | Murat OZORAL* August 17, 2009Tracked on

    [...] playerlar'ın demolarını gördük. Daha sonra Apple'da bu kervana katılacağını duyurdu ve [...]

15. *Apple's Snow Leopard Will Bring Major Video Boosts* August 24, 2009Tracked on

    [...] new QuickTime X includes support for Apple's HTTP live streaming protocol, which we've covered extensively since it was announced and released first for iPhones in June. The technology has [...]

16. *Dispelling the FUD! Silverlight is NOT on the iPhone! | code zen* November 26, 2009Tracked on

[...] have all proprietary adaptive streaming formats. For a better understanding read this article on Apple's adaptive streaming on [...]

17. *Envivio Adds $1M for Video Encoding* December 31, 2009Tracked on

    [...] convertible note filed today with the SEC. The South San Francisco, Calif.-based company, which was one of the first allowed onto Apple's new HTTP streaming platform, had last raised funding in 2008, when it [...]

18. *Some iPhone Video Apps Won't Be Approved Unless They Can Adapt To AT&T's Lousy Network | CanHazBlog.com* March 26, 2010Tracked on

    [...] is increasingly encouraging developers of video apps to include adaptive bitrates which vary based on the user's connection. Currently Apple requires all long-form video [...]

19. *WWDC: Netflix With Adaptive Bitrate Streaming Coming to the iPhone* June 7, 2010Tracked on

    [...] hand. The free application arrives this summer and will support adaptive video playback — a streaming technology that adapts to the connection speed of a device demonstrated last [...]

20. *WWDC: Netflix With Adaptive Bitrate Streaming Coming to the iPhone « Apple News Daily* June 7, 2010Tracked on

    [...] their hand. The free application arrives this summer and will support adaptive video playback — a streaming technology that adapts to the connection speed of a device demonstrated last [...]

21. *mDialog Tackles HTML5 Video Security* June 9, 2010Tracked on

    [...] new feature works by leveraging Apple's HTTP adaptive bit rate streaming protocol to deliver video to the iPad. The Apple implementation of HTTP adaptive streaming breaks video [...]

22. *Some iPhone Video Apps Won't Be Approved Unless They Can Adapt To AT&T's Lousy Network « Gadget Fee* July 1, 2010Tracked on

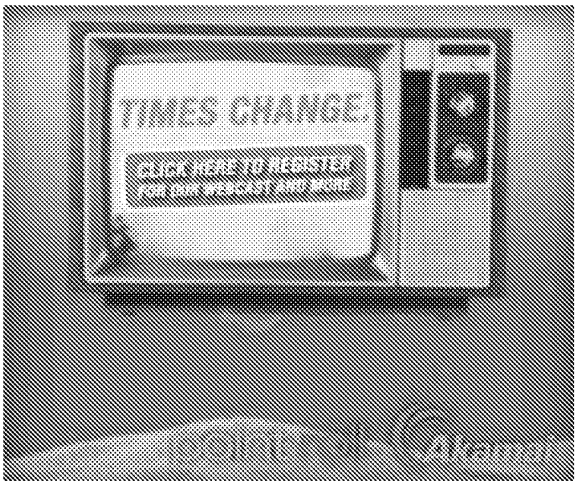
    [...] is increasingly encouraging developers of video apps to include adaptive bitrates which vary based on the user's connection. Currently Apple requires all long-form video [...]

23. *Apple's HTTP Adaptive Protocol - Castfire* September 8, 2010Tracked on

    [...] Brian Walsh on Sep.08, 2010, under Clients, Development, Devices, Features, HTML5, Mobile In June of 2009, Apple introduced a new video format that provides adaptive bitrate over http. If those words [...]

24. *Adobe Flash Pushes Into the Enterprise With P2P, IP Multicast* September 8, 2010Tracked on

    [...] updates to Adobe's Flash Media Server — like FMS 3.5, which introduced HTTP streaming — were focused mainly on making streaming server better suited for use by media companies. [...]



Sign up to get GigaOM news!

Your Email Address SUBSCRIBE

Like 53

- Twitter
- Facebook
- RSS

Most Recent

1. Can Premium VOD Make Up for Losses at the Box Office?
2. 2010 in Review: Elegy for the Video Store
3. Holiday Tips for Your New Boxee Box, Roku, Xbox & PS3
4. Cord Cutters Survival Stories: Online Content Is Exciting
5. The Top 5 Video Gadgets of 2010

GIGAOM TV

Green Overdrive: Mercedes Fuel Cell Car!

VIEW MORE THAN 1,000 VIDEOS ONGIGAOM.TV

RSS Feed for Liz Gannes Email Liz Gannes Liz Gannes

Liz's Posts

- Twitter Relaunches Twitter.com Web App

- Akamai Powering Apple Live Stream (And I Can Prove It)

GigaOM Research

Can Social Media Save the Music Business?

Harnessing the power of social media — and the inherent social nature of music — might just give the still-flaiiling music industry a new boost.

A Tale of Two Vendors: Will Oracle, Red Hat Earnings Converge?

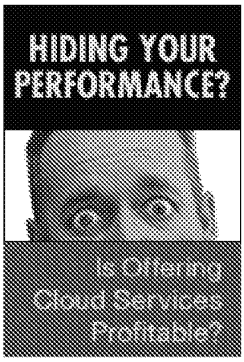
The advent of cloud computing might significantly lessen the overall earnings gap between Oracle and Red Hat.

Why the Mobile App Industry Needs to Address Privacy — Now

Can the mobile app industry implement a privacy policy before federal regulators step in?

GIGAOM PRO

Partner Center



Find Out Now.

The Crunchies 2010

Upcoming Event

STRUCTURE BIG DATA 2011

March 23rd, 2011: Pier Sixty at Pier 60, NYC

Jim Baum
CEO, Netezza

Dave Hitz
CTO, NetApp

Michelle Munson
CEO, Aspera

Scott Yara
Co-founder, Greenplum

Learn how big data precedes big opportunity. Hear the practitioners. Meet the players. Define your big data strategy.

BUY TICKETS

Across *the* Network

- Apple

Rumor Has It: 20 Million Plus iPhones for Q1 2011

- Cleantech

  Snow Day Is Test Day For Electric Cars

- Cloud

  Security Questions to Ask Before Engaging with Cloud Providers

- Collaboration

  Gantt, PERT, or Task Calendar: Which Scheduling Model is Best for Your Project?

- Open Source

  Unity Coming to openSUSE Too?!

- Video

Can Premium VOD Make Up for Losses at the Box Office?

© 2010 GigaOM

An Experimental Study of Throughput for UDP and VoIP Traffic in IEEE 802.11b Networks

Sachin Garg
sgarg@avaya.com
Avaya Labs Research
Basking Ridge, NJ USA

Martin Kappes
mkappes@avaya.com
Avaya Labs Research
Basking Ridge, NJ USA

***Abstract*—In this paper, we present experimental studies on the throughput of IEEE 802.11b wireless networks for UDP and VoIP traffic. Our experiments show that the maximum data throughput of a single station sending out UDP traffic is 6.1 Mbps. The maximum number of VoIP calls in a single cell of an IEEE 802.11b network is six if the ITU G711a-Law codec is used with 10 milliseconds of audio data per RTP packet. The experiments also show that the effective available bandwidth in the wireless network is reduced by ongoing VoIP connections. Specifically, for the above codec settings, each VoIP connection reduces the bandwidth available for data traffic by 900 Kbps.**

I. Introduction

In the last few years, wireless networks based on the IEEE 802.11b standard have gained popularity and have been widely deployed in enterprises mostly to provide wireless data access from Laptops, PDAs, etc. to the wired infrastructure of the enterprise. They have also been deployed in public hot-spots such as airports, hotels, conference facilities etc., mainly for internetwork connectivity.

The maximal data rate 802.11b currently supports is 11 Mbps. Although task groups, in particular 802.11a and 802.11g [10], are working on allowing higher maximum data dates, it is very likely that 802.11b deployments will continue to operate in both enterprises as well as the residential market for the next few years. As opposed to IEEE 802.3 (also known as Ethernet), where the maximal data rate of the network is indeed close to the throughput observed, the maximal achievable throughput for 802.11b networks is far lower than the data rate due to the nature of 802.11's CSMA/CA medium access protocol. Moreover, the bandwidth is shared among all participants in the network whereas most of today's Ethernet deployments are "switched".

As converged networking in the wired world is becoming more and more popular, it is very likely that in the near future wireless networks will also be increasingly used for voice traffic. In this paper, we present our experimental studies on the throughput for VoIP and UDP traffic in a single cell of an 802.11b network.

We chose UDP as transport layer protocol due to its connection-less nature. A UDP sender "flooding" the network gives an accurate estimate of the actual bandwidth that is available in the network. Results obtained with UDP constitute an upper bound for the throughput possible with TCP as transport layer protocol in any of the scenarios we studied. Another reason why we chose UDP is the 802.11 MAC. It was constructed in such a way that higher frame loss and collision rates in the wireless network as compared to a wired network are mitigated by the MAC-layer itself. In particular, the MAC protocol mandates acknowledgments of received frames by the receiver and retransmissions of frames by the sender that are not acknowledged within a time interval. The timer values are such that the higher layer transport protocols, in particular TCP, do not get affected by the loss of a frame once in a while. Hence, studying phenomena related to the connection-oriented nature of TCP does not lead to new insights into network problems that are related to the specific use of an 802.11 network.

This paper is organized as follows. First, in Section II, we outline the laboratory setup for all conducted experiments. Each section from III through VI consists of the description of a single experiment, the results obtained from the experiment and a detailed explanation. The first experiment, in Section III, studies the payload throughput of an 802.11b network as a function of the frame payload size. The second experiment, in Section IV, studies the throughput with multiple senders. The third experiment, in Section V, is aimed to find the maximum number of simultaneous VoIP connections in a single cell. In Section VI, we study the throughput of such a network in the case of converged networking, i.e., simultaneous voice and data traffic. Finally, Section VII discusses the ramifications of the experiments and Section VIII concludes the paper.

II. Experimental Setup

Eight clients, all PCs (some Laptops, some Desktops) running Windows 2000, are associated with a single access point. The access point is connected to an IEEE 802.3 (Ethernet) LAN. On the wired side of the network, we are also using PCs running Windows 2000 which serve as endpoints for VoIP or UDP data connections. All endpoints in the wireless network are in one subnet whereas the nodes in the wired network are on another. Both subnets are connected through a single router and are thus one hop apart from each other.

The IEEE 802.11b access point and all clients are situated in the same room with no physical obstacles between them. Hence, the probability of frame loss due to weak signal strength and/or presence of hidden stations is negligible. The experiments were conducted with access points from different vendors, with no significant differences in the results. The radio cards used in all clients were from the same vendor.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2029
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

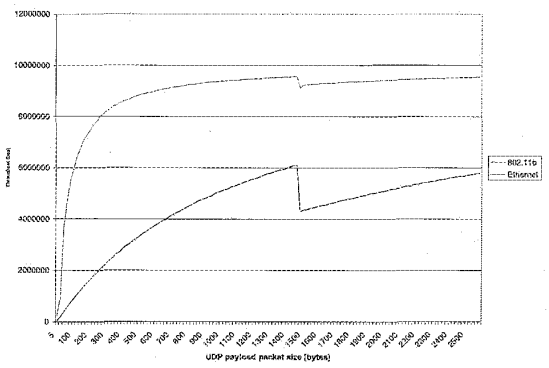


Fig. 1. Throughput as a function of the packet size for Ethernet and 802.11b. The drop in throughput for packet payloads larger than 1472 bytes is due to fragmentation.

III. Throughput for a Single UDP Sender

A. Experiment and Results

We measured the available throughput in terms of sent payload on Layer 4 with IP/UDP as bearer in the scenario of a single client.

We used a program constantly sending out UDP data frames to the wireless medium (and, hence, in some sense "flooding" the wireless network). The stream was sent to an endpoint in the wired network. Our results indicate that the maximal achievable payload data rate in this scenario is approximately 6.1 Mbps which is achieved when the payload of each UDP packet is chosen to be 1472 bytes. In this case, the maximal payload of an Ethernet frame (1500 bytes minus 20+8 bytes for the IP and the UDP header) is used. Payloads larger than 1472 bytes get fragmented into more than one frame which reduces the observed throughput significantly. When the 62 bytes of the IEEE 802.11 frame body and the IP/UDP headers (34+20+8) in each sent frame are accounted for, the overall throughput of the wireless network is approximately 6.36 Mbps.

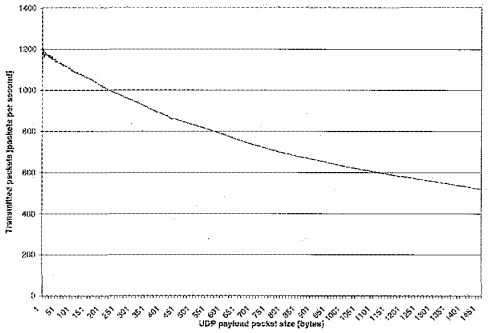


Fig. 2. Transmitted packets per second as a function of the packet size in IEEE 802.11b.

Figure 1 shows our measurements of payload throughput as a function of the payload size in a single client scenario for 10 Mbps Ethernet and 802.11b. Figure 2 shows the number of transmitted frames per second as a function of the packet size.

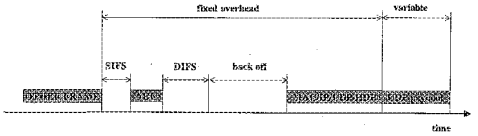


Fig. 3. IEEE 802.11 CSMA/CA medium access scheme.

Based on the number of frames sent per second and the fact that in our setup the payload of each frame is transmitted at 11 Mbps, we can compute the fixed overhead for the transmission of each frame. For our measurements, the overhead evaluates to an average of around 840μs per frame. It should be noted that the transmission of a payload of 1472 bytes at 11 Mbps takes about 1070μs, hence even in this case the fixed overhead time is quite comparable to the actual transmission time for the payload.

B. Explanation

Clearly, the question arises why the fixed overhead per frame is so significant that it takes almost half of the total transmission time even when using the largest possible payload size of 1472 bytes. The overhead stems from the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) medium access scheme according to the Distributed Coordination Function (DCF) of the 802.11 standard [9] which we will briefly describe here. A more detailed description can be found in [8].

As collisions in the wireless medium cannot be detected, the MAC protocol is designed to prevent collisions from occurring. Furthermore, due to the possibility of undetected collisions and the higher number of transmission errors as compared to wired networks (e.g., by interference), unicast frames are acknowledged by the receiving station. The acknowledgment is sent out after the transmission has finished and a certain duration of time called short inter frame spacing (SIFS) has elapsed. If a node wants to transmit a frame and senses the medium idle for a certain duration of time called distributed coordination function inter frame spacing (DIFS), it may start transmitting. As DIFS is longer than SIFS, it is made sure that a well received frame can always be acknowledged for before the next frame is transmitted.

If a node wants to start transmitting while the medium is busy or if it wants to transmit another frame after just finishing a transmission, it also waits for the medium to be idle for the DIFS period. Then, the node does not begin to transmit immediately but enters a contention phase for the medium. Contention is done by choosing an integer random back-off between 0 and a parameter CW (CW stands for contention window size) which is initially set to a value CWmin. The probability distribution among these values is uniform. The random back-off determines the number of time slots the client defers its transmission in addition to the DIFS time.

If the medium is sensed idle in such a "slot", the back-off timer is decreased by one. If the random back-off has decreased to 0, the node starts transmitting. If another node starts transmitting before this happens, the node continues to count down the back-off timer after the medium has been sensed idle for the DIFS period. Thus, if multiple clients want to transmit a

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2030
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

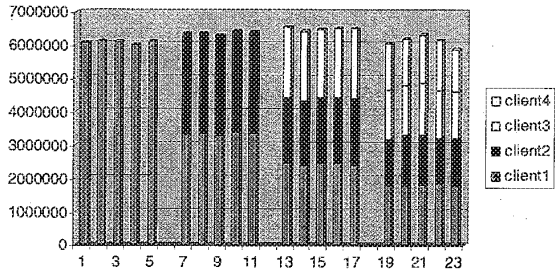


Fig. 4. Throughput for multiple UDP senders sending packets of 1472 bytes. Y-Axis shows throughput in bps.

frame, the one with the lowest random back-off time will win the contention for the medium. However, if more than one node happens to choose the same back-off time, they will start to transmit at the same time and a collision will occur. The clients assume that the frame was lost if an acknowledgment is not received within SIFS. In case of an unsuccessful transmission, the CW value doubles until a CWmax value is reached. The CW parameter is reset to the CWmin after each successful (i.e. acknowledged) transmission.

The IEEE 802.11/802.11b standard defines SIFS to be $10\mu s$. A slot time is $20\mu s$ and the value of DIFS is defined to be the value of SIFS plus two slot times which is $50\mu s$. The size of an acknowledgment frame is 14 bytes which take about $10\mu s$ to transmit at 11 Mbps. However, each transmitted frame also needs some physical layer overhead (PLCP header of $48\mu s$ and a preamble of $144\mu s$) which is about $192\mu s$. Thus, the total time to transmit an acknowledgment is $203\mu s$. The IEEE 802.11b standard [9] defines CWmin to be 31. Therefore, in the scenario of a single client constantly transmitting, the average random back-off time is 15.5 slots which equals $310\mu s$. For the actual data frame we have an overhead of 34 bytes for the 802.11 MAC header, 20 bytes of IP header and 8 bytes of UDP header totaling 62 bytes which take about $45\mu s$ to transmit at 11 Mbps. Together with the $192\mu s$ physical layer overhead this amounts to $237\mu s$. Summing up these values, the fixed overhead per frame as illustrated in Figure 3 can be calculated as $10 + 203 + 50 + 310 + 237 = 810\mu s$.

Indeed, this value matches well with the overhead measured in the experiment. The experimental values are a little bit higher due to the fact that the overhead of periodic beacons sent out by the access point is not included in the calculated value. Such a beacon contains management information about the network and is sent out about every 100 ms. When the effect of the beacons is discounted, the measured value matches with the calculated value.

IV. Throughput for Multiple UDP Senders

A. Experiment and Results

In this experiment, the program constantly sending out UDP data frames is run simultaneously on multiple wireless stations connected to the same Access Point. All UDP streams were sent to endpoints in the wired network.

When running two datagram senders, the aggregated maximal achievable payload data rate increased from approximately 6.1 Mbps in a single sender scenario to approximately 6.4 Mbps for a payload of 1472 bytes each UDP packet in both streams. Moreover, both senders equally share the available bandwidth. Each client is able to send around 3.2 Mbps.

For three senders, the maximal achievable payload data rate further increases to 6.5 Mbps which is also fairly shared among all clients. For four senders, the maximal achievable payload data rate decreases to around 6.1 Mbps.

Assuming all senders transmit with 11Mbps, the fixed transmission overhead per frame can be calculated to be $750\mu s$, $720\mu s$, and $830\mu s$ for two, three and four senders, respectively.

B. Explanation

As mentioned in the explanation of the previous experiment, the average back-off idle time between successive frames on the channel is exactly equal to the average back-off between successive frames of the station.

If more than one station is transmitting, there are two main factors, which determine the maximum channel throughput. First is the average idle time on the channel due to inter-frame back-offs performed by each station. Second is the probability that a transmission on the channel results in a collision. Obviously, a lower average idle time increases the channel throughput while a higher probability of collision reduces the channel throughput.

Let us first study the case of two senders always contending for the medium. In previous work on throughput analysis of 802.11 networks, the average actual back-off has been evaluated both via analytical approximations as well as via simulations [1], [2], [3], [7]. From these, the average back-off window size is determined to be 8.5 slots. Therefore, when calculating the fixed overhead, we obtain a value of $670\mu s$.

However, there is a chance that transmissions of the two senders collide and in turn the time of the transmission is wasted. The collision probability has been shown to be 0.03. These failed transmissions account for "wasted" air-time that has to be taken into account when comparing the experimentally observed value. It takes $1070\mu s$ to transmit a 1472 byte long frame plus a fixed overhead of $670\mu s$. Thus, the total time for such a frame transmission is $1740\mu s$. Crediting this time to all successfully transmitted frames, we obtain a correction factor of $0.03 \cdot 1740 = 52\mu s$.

Hence, we would expect to observe a fixed overhead of $722\mu s$ per frame. This value is very close to the experimentally obtained overhead of $750\mu s$. Again, the deviation between calculated and observed value is due to the transmission of beacons. Furthermore, the time accounted for a collided frame is too optimistic since the MAC scheme requires a longer idle period (Extended Interframe Spacing, EIFS) after a collision.

For three senders, the average back-off window size is determined to be 6.215 slots and the collision probability is 0.056. Thus, the expected fixed overhead per transmission is $719\mu s$. The actual value we measured is $741\mu s$.

In the case of four senders, the average back-off window size is determined to be 5.061 slots and the collision probability is 0.0774. Thus, we would expect a fixed overhead per

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2031
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

transmission of 730μs and hence an aggregated throughput of 6.54 Mbps. However, the measurements revealed a much lower throughput.

The reason was that the peak transmission rate of one or more wireless stations was not 11Mbps, but dropped down occasionally to 5.5Mbps. The stations misinterpret the unusually high collision rate with four active stations as interference. In turn, they decrease the transmission speed to enable better interference robustness. Since interference detection and adaptive transmission rate control is not part of the standard and is implemented in a proprietary, undisclosed manner, it is not possible to calculate the fixed overhead.

V. Maximal Number of VoIP Connections using ITU G711 a-Law Codec

A. Experiment and Results

The same set-up was used to explore the maximal number of VoIP connections possible in a single cell of an IEEE 802.11b network. One end-point of each VoIP call is a wireless client, while the other end-point is on the wired network. Given the short range of an 802.11 access point, "wireless-to-wireless" calls do not seem to be a typical scenario.

For each call, we used the ITU G711 a-Law codec where frames are sent out every 10ms. The codec output is sent over the IP, UDP and RTP protocols. It should be noted that it is possible to use this codec with varying audio data length per packet sent. Each millisecond of audio is encoded into eight bytes of data. In addition to the IP/UDP overhead each packet contains an additional fixed RTP layer overhead of twelve bytes. Hence, when sending audio data packets every 10ms, the payload of such a packet is 92 bytes. When making a VoIP phone call, two such data streams are established, one in each direction.

We tested the number of VoIP connections with acceptable voice quality by successively establishing new calls in addition to ongoing calls. The quality of the connections was monitored through measurements of loss, jitter and round-trip time by a commercially available tool. For the first five calls, the quality of all connections was fine. Loss (0%), round-trip time (around 5ms) and jitter (around 7ms) were all in ranges far below critical although higher than experienced in a single-hop wired Ethernet. When placing the sixth call, except for a sporadic increase in the round-trip time for some of the connections as measured by the commercial tool, the quality of the six simultaneous connections was still fine. Placing the seventh connection lead to unacceptable loss in all "wired to wireless" streams. Except for the round-trip time, all "wireless to wired" streams still were fine.

Thus, our experiments indicate that the maximal number of simultaneous VoIP connections in a single cell of an IEEE 802.11b network when using the G711 a-Law codec with 10ms audio data packets is six.

B. Explanation

The aggregated bandwidth of seven VoIP connections as specified above is approximately 1 Mbps. It can be argued that the bandwidth available when the network is exclusively used for VoIP traffic is equivalent to the bandwidth available in the case of two UDP senders flooding the network [5]. As the bandwidth in this scenario is lower than the bandwidth needed for seven VoIP connections, the unacceptable call quality is caused by overload.

When the seventh connection is started, two additional VoIP-streams are created, pushing the load offered by the nodes to the network beyond its throughput limit. Thus, some of the audio data traffic cannot be sent through the wireless network. Whereas each wireless node only sends a single VoIP stream, namely the one going from it to wired, the access point has to send out all audio data traffic coming from wired in the wireless network. Therefore, it is actually sending 50% of the overall load on the network before the network is saturated. As the CSMA/CD protocol is designed to be fair, in a situation where more load is offered than can be accommodated, the node using the largest number of transmission slots will be curtailed first. Thus, in our scenario the access point is the only node unable to put its load onto the network.

The paper [5] presents a detailed analysis on the number of VoIP connections possible in an 802.11b network for various codecs and audio data payload settings. Here, we would only like to recall the results with respect to the maximal number of connections possible when all stations transmit using a data rate of 11Mbps.

Audio (ms)	G711	G729	G723
10	6	7	
20	12	14	
30	17	21	21
40	21	28	
50	25	34	
60	28	41	42
70	31	47	
80	34	54	
90	36	60	61
100	39	66	

As this table shows, the analytical limit of six connection matches with our experimental findings.

VI. UDP Throughput in the Presence of VoIP Traffic

A. Experiment and Results

Now we consider the case of converged networking, i.e. the simultaneous presence of voice and data traffic in the wireless network. Again, our focus is on throughput. One station is running the UDP datagram sender, whereas other stations connected to the AP run VoIP connections using the same parameters as in the previous experiment. Again, all connections terminate in the wired network. The following table shows the throughput of the UDP sender as a function of the number of ongoing VoIP connections.

VoIP conn.	UDP Throughput
0	6.06 Mbps
1	5.15 Mbps
2	4.26 Mbps
3	3.28 Mbps

As these values show, each VoIP connection diminishes the throughput of the UDP sender by approximately 900 Kbps even though the VoIP connection only consists of two UDP streams

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2032
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

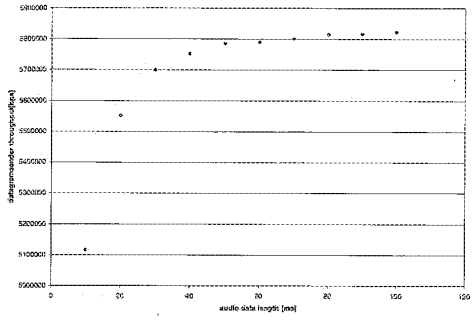


Fig. 5. Throughput of datagramsender when one VoIP connection is active.

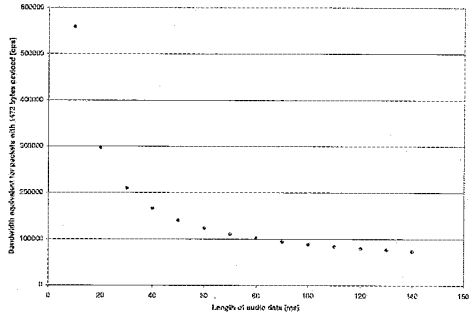


Fig. 6. Equivalent Throughput of a G711 a-Law stream when assuming that same transmission time is used to send UDP traffic with 1472 bytes payload instead.

each of which has a bandwidth of 74 Kbps (64 Kbps voice data plus 10 Kbps RTP overhead). Hence, the aggregated throughput of the network reduces by 760 Kbps for each VoIP connection.

B. Audio Data Size Variation

In a successive experiment, we ran a single VoIP connection using the G711 codec and varying audio data size as well as a datagram sender. We measured the payload throughput of the datagram sender. The results are displayed in Figure 5.

The throughput of the UDP sender increases steeply with an increase of the audio data payload size from 10ms (92 bytes per packet) to 20ms (172 bytes per packet). Then, the returns in terms of bandwidth for using larger audio data payload sizes diminish with increasing audio data payload size.

C. Explanations

When the wireless network is shared between voice and data, the remaining bandwidth for data when running a VoIP-Connection with a small payload size is lower than when running a data connection of the same bandwidth using full size frames.

Denote the maximal throughput possible for a packet payload size of k by R^k_{eff}. For a VoIP stream with payload size k and bandwidth b, the bandwidth equivalent b^{1472}_{eq} is defined by

$$b^{1472}_{eq} = \frac{b}{R^k_{eff}} \cdot R^{1472}_{eff}.$$

Intuitively, bandwidth equivalent is the bandwidth that would become available for data traffic if the VoIP stream was shut down. Figure 6 shows the bandwidth equivalent of a VoIP-Connection for varying audio data packet payloads. The values used for R^k_{eff} and R^{1472}_{eff} are taken from the results of the first experiment. As this figure shows, a single G711 a-Law VoIP-stream sending a packet every 10ms uses the network like a data connection sending UDP packets with 1472 bytes with 560 Kbps. Thus, when running one VoIP-Call with these parameters, the remaining bandwidth for data traffic in the wireless network is only 4.9 Mbps as each of the two VoIP data streams consumes bandwidth equivalent to 560 Kbps of data traffic.

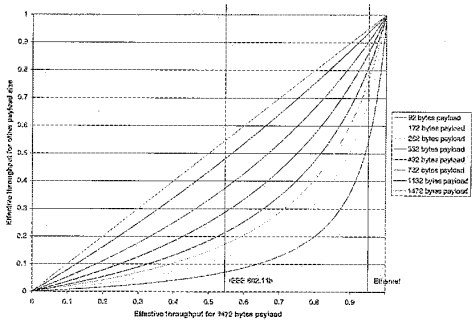


Fig. 7. Effective throughput of a network for 92, 172, 252, 332, 492, 732, 1132 and 1472 bytes (packet sizes used by a G711 a-Law codec with audio data length 10ms, 20ms, 30ms, 40ms, 60ms, 90ms, 140ms, respectively) of payload as a function of the effective throughput for 1472 bytes of payload. The vertical lines mark the values for IEEE 802.11b and Ethernet.

VII. Discussion

A. Use of 802.11b for VoIP

VoIP call quality in IEEE 802.11 networks, as determined by the loss, delay and jitter characteristics of the call, is fine as long as the network throughput limit is not exceeded. In other words, the throughput of the wireless network is the key to determining the number of VoIP connections which may be run simultaneously in a single cell. In contrast with Ethernet, the smaller packet size of VoIP traffic as opposed to data traffic has a dramatic impact on the effective throughput in a wireless network.

Figure 7 shows the effective throughput of a network for varying payload packet sizes as a function of the effective throughput for 1472 bytes of payload under the assumptions that the overhead for all different packet payload sizes is fixed and that the variable transmission time for the packet payload is linear.

Let the effective throughput of a network at payloads of 1472 bytes be denoted by R^{1472}_{eff}. The overhead of the transmission, denoted by T_{oh}, measured in time units needed to transmit one byte, is given by

$$T_{\text{oh}}(R^{1472}_{\text{eff}}) = 1472 \cdot \frac{1 - R^{1472}_{\text{eff}}}{R^{1472}_{\text{eff}}}.$$

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2033
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Consequently, the effective throughput available for packets with a payload size of k bytes, R^k_{eff}, as function of the effective throughput for packets with payload size 1472 is given by

$$R^k_{\text{eff}}(R^{1472}_{\text{eff}}) = \frac{k}{k + T_{\text{oh}}} = \frac{k}{k + 1472 \cdot \frac{1 - R^{1472}_{\text{eff}}}{R^{1472}_{\text{eff}}}}.$$

As shown in the figure, the efficiency of Ethernet which is around 95% for a payload packet size of 1472 bytes still allows for an efficiency of around 55% when the payload packet size is only 92 bytes.

There are two ways of increasing the efficiency of such a network for the VoIP-streams we consider here, namely to either to increase the overall efficiency of the network by reducing the fixed overhead per sent payload or to switch to transmitting larger payload packets.

Let us discuss the first of these alternatives. Whereas standard bodies and committees are in general trying to make their proposals as efficient as possible, fixed overhead per transmitted frame is especially crucial for smaller packet sizes which are common for real-time traffic. However, as the picture shows, the lower the efficiency of the network for payloads of 1472 bytes, the less impact has an increased efficiency on the throughput achievable for lower packet payload sizes. Adding to that, a small packet payload size also decreases the improvement in efficiency for such measures. In fact, if the fixed overhead of IEEE 802.11 could be cut from 840μs to 420 μsec, this would increase the network efficiency for packet payloads of 1472 bytes from 55% to 72% but the efficiency for packet payloads of 92 bytes would only increase from 7% to 14%. Hence, although decreasing the current overhead in IEEE 802.11b in general would significantly help to increase the available throughput for data traffic, it would not be as helpful for VoIP traffic.

So, increasing the audio data length per transmitted packet appears to be a better solution to the problem. Unfortunately, there is a tradeoff between the subjective audio quality of the call and the amount of audio data sent in a single packet. A higher audio data length per packet increases the time gap between the recording of the audio data on one side of the stream and its playback on the other side as the data has to be collected before it is sent. Moreover, loosing a single audio packet with a large amount of audio data is an event that will likely be heard by the receiving party. The right solution therefore might be one that exploits different optimal payload sizes for the wireless and the wired part of the network.

B. Capacity Assessment for Wireless Network

As our experiments show, the maximal aggregated bandwidth available in an 802.11b network is highly dependent on the traffic in the network and can range anywhere from a few hundred Kbps up to 6.1 Mbps even if only a data rate of 11 Mbps is used. Adding to that, stations may also transmit at data rates of 1, 2 or 5.5 Mbps. Hence, the simple questions "can the network handle a flow with 2 Mbps bandwidth" or "can the network accommodate 800 packets per second" cannot be answered when only looking at bandwidth.

Thus, admission control for 802.11b networks cannot be based on bandwidth. However, taking into account the low number of VoIP connections possible, the need for VoIP admission control in wireless networks is apparent. Placing an additional call or an additional data connection that exceeds the capacity of the wireless network will result in unacceptable call quality for all ongoing VoIP calls. Further, if the load offered to the network is higher than its capacity, the DCF medium access scheme of 802.11 curtails the client with the highest load first. In most cases, the access point of the wireless cell puts more traffic on the air than the associated stations. Hence, it gets curtailed first which leads to unacceptable packet loss for all VoIP streams transmitted from the access point to a client resulting in bad call quality for all connections.

Therefore, in [6], an admission control metric was proposed that is based on the definition of bandwidth equivalence as outlined here. This metric allows to accurately assess the capacity of an 802.11b network by calculating the bandwidth equivalence for all flows which allows for an accurate picture of the current and future channel usage.

VIII. Conclusions

In this paper, we studied the behavior of UDP and VoIP over 802.11 networks, from the perspective of number of connections that a single access point can support. An important conclusion is that given the choice of payload sizes in IP phones, 802.11b base-stations prove to be inadequate in handling large number of VoIP calls. In fact, the inherent channel inefficiency of 802.11b, at smaller frames sizes, limits the maximum number of VoIP calls to a very low number, six in the case of G711a-law codec with 10ms of audio data per packet. Furthermore, our experiments revealed that the aggregated bandwidth of the wireless network is diminished by ongoing VoIP connections. Each such G711 VoIP connection reduces the bandwidth available for data traffic by approximately 900 Kbps. In turn, the maximal bandwidth available for other traffic when three of such connections are active is only 3.3 Mbps.

References

[1] G. Anastasi and L. Lenzini. QoS provided by the IEEE 802.11 wireless LAN to advanced data applications: a simulation study. In *Wireless Networks*, Vol. 6, pp 99–108, J.C. Baltzer AG, Science Publishers, 2000.

[2] G. Bianchi. Performance Evaluation of the IEEE 802.11 Distributed Coordination Function. In *IEEE Journal on Selected Areas in Communication*, Vol. 18, No. 3, March 2000, pp. 535–547.

[3] H. S. Chayya and S. Gupta. Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol. In *Wireless Networks* Vol 3, 1997, pp. 217-234.

[4] S. Garg, M. Kappes and M. Mani, *Wireless Access Server for Quality of Service and Location Based Access Control in 802.11 Networks*, Proceedings of the Seventh IEEE Symposium on Computers and Communications (ISCC), Taormina, Italy, 2002.

[5] S. Garg and M. Kappes. *Can I add a VoIP call?*, submitted.

[6] S. Garg and M. Kappes. *A New Admission Control Metric for VoIP Traffic in 802.11 Networks*, submitted.

[7] A. Heindl and R. German. Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri nets. *Performance Evaluation*, 44 (2001), 139-164.

[8] J. Schiller. *Mobile Communications*, Addison Wesley, New York, 2000.

[9] IEEE 802.11, 11a, 11b standard for wireless Local Area Networks. *http://standards.ieee.org/getieee802/802.11.html*

[10] *http://grouper.ieee.org/groups/802/11/Reports/tgg_update.htm*

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2034
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

IIS Smooth Streaming

Enhancing Silverlight Video Experiences with Contextual Data

Jit Ghosh

Download the Code Sample

There are two primary requirements for enabling a glitch-free viewing experience in Web-based, high-definition digital video delivery. First, the video provider needs to support high video delivery bit rates over the network. Second, the client computer needs to support continuous availability of processing capacity to decode the video at its fullest resolution.

The reality, however, is that network bandwidth for connected home computers can fluctuate significantly over time, and in certain parts of the world high bandwidth comes at a very high premium or is unavailable to many consumers. Along with that, the processing capacity of the client computer can vary, depending on CPU load at any given moment. As a result, consumers are susceptible to degradation in the quality of their viewing experience when a video stutters or freezes while the player is waiting to buffer enough data to show the next set of video frames, or waiting for the CPU cycles to decode those frames.

Adaptive streaming is a video delivery approach that addresses the problem of smooth content delivery and decoding. With adaptive streaming, video content is encoded at a range of bit rates and made available through a specialized streaming server. An adaptive streaming player constantly monitors various resource utilization metrics on the client computer and uses that information to compute the appropriate bit rate that the client can most efficiently decode and display at given the current resource constraints.

The player requests chunks of video encoded at that currently appropriate bit rate, and the streaming server responds with content from the video sources encoded at that bit rate. As a result, when resource conditions degrade, the player can continue displaying the video without any significant disruptions, with only a slight degradation in overall resolution, until an improvement or further degradation in conditions causes a different bit rate to be requested.

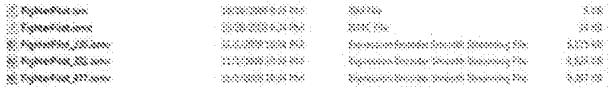
This kind of a continuous collaboration between the player and the server requires a special implementation of processing logic on both the streaming server and the client runtime implementing the player. Internet Information Server (IIS) Smooth Streaming is the server-side implementation of adaptive streaming over HTTP from Microsoft. The client-side implementation is provided as an extension to Microsoft Silverlight.

The IIS Smooth Streaming Player Development Kit is a Silverlight library that lets applications consume content being streamed over IIS Smooth Streaming. It also provides a rich API that offers programmatic access to various aspects of the Smooth Streaming logic.

In this article I will walk you through the basics of Smooth Streaming, and explain how you can use the IIS Smooth Streaming Player Development Kit to build rich user experiences around video. Specifically, I will look at using the Player Development Kit to consume a stream, with a close examination of the client-side data model for streams and tracks. I will show you how to consume additional data streams, such as closed captions and animations, and merge external data streams with an existing presentation. You'll see how to schedule external clips such as advertisements, within a presentation, handle variable playback rates and build composite manifests that lend to robust editing scenarios.

How Smooth Streaming Works

You can encode video for Smooth Streaming by using one of the supplied profiles in Expression Encoder 3.0. For one source video file, several files are created in the destination folder. **Figure 1** shows the files created for a source video named FighterPilot.wmv.



MSDN Magazine Blog

Five Questions: Build a Ticketing Syste
Exchange and Team Foundation Serve
In the current (August 2011) issue of M
Magazine, Mohammad Jalloul wrote a
titled "Build a Ticketing System Using
and Team Foundat... More...

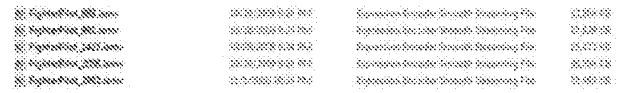
Silverlight Launcher Building Windows 8
Don't look now, but Microsoft is peeli
layers on Windows 8. In June, develop
back-to-back demos and discussion a
at the C... More...

Black Flies in the Code
I'm just returned from a week-long fa
the coast of Maine, where we stayed i
cottage near Boothbay Harbor. As wit
vacation,... More...

BUILD Conference Sells Out
I'm registered to attend the highly-ant
Microsoft BUILD conference, schedule
September 13-16 in Anaheim, Californ
seem to be one... More...

More MSDN Magazine Blog entries >

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2035
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)



Current Issue

Each of the files with an .ismv extension contains the video encoded at a specific bit rate. For example, the FighterPilot_331.ismv contains the video encoded at a bit rate of 331 kbps, while FighterPilot_2056.ismv contains the video encoded at 2 mbps.

For each bit rate, the video content is broken into two-second fragments, and the .ismv files store these fragments in a file format called Protected Interoperable File Format (PIFF). Note that you can have additional audio tracks (or just audio in case the presentation is audio only) encoded in similar files that have an .isma extension.

Getting the Smooth Streaming Environment

To try out the examples discussed in this article, you will need to prepare a Smooth Streaming environment on your development machines.

The server-side ingredient is straightforward: you'll need to download and install IIS Media Services 3.0 for IIS7 from iis.net/media using the Microsoft Web Platform Installer.

You will need a copy of Microsoft Expression Encoder 3.0 to prepare video for Smooth Streaming. While there is a free evaluation version of Expression Encoder 3.0, that version does not include support for Smooth Streaming. You will need a licensed installation of Expression Encoder to create your own video.

For additional details on preparing your environment, please visit learn.iis.net/page.aspx/558/smooth-streaming-for-iis-70---getting-started.

The FighterPilot.ism file is a server manifest, which is structured in Synchronized Multimedia Integration Language (SMIL) format and contains a mapping of quality levels and bit rates to the .ismv and .isma files. This mapping in the server manifest is used by the server to access the right disk files to create the next fragment of content encoded at the right bit rate, before responding to a client side request. **Figure 2** shows an excerpt of a server manifest file.

Figure 2 **Sample Server Manifest**

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <meta name="clientManifestRelativePath"
      content="FighterPilot.ismc" />
  </head>
  <body>
    <switch>
      <video src="FighterPilot_2962.ismv"
        systemBitrate="2962000">
        <param name="trackID"
          value="2" valuetype="data" />
      </video>
      <video src="FighterPilot_2056.ismv"
        systemBitrate="2056000">
        <param name="trackID"
          value="2" valuetype="data" />
      </video>
      ...
      <audio src="FighterPilot_262.isma"
        systemBitrate="64000">
        <param name="trackID"
          value="1" valuetype="data" />
      </audio>
```

msdr

Browse All MSD

Subscribe to MSDN Flash

Receive the MSDN Flash e-mail newsl
other week, with news and informatio
personalized to your interests and ar

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2036
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
    </switch>
  </body>
</smil>
```

The server manifest also contains a mapping to a client manifest file (identified by the extension .ismc), which in my example is FighterPilot.ismc. The client manifest contains all the information that the Silverlight client will need to access the various media and data streams, as well as metadata about those streams, such as quality levels, available bit rates, timing information, codec initialization data and so on. The client-side logic will use this metadata to sample and decode the fragments and request bit rate switches based on prevailing local conditions.

At run time, the presentation begins with the client requesting the client manifest from the server. Once the client receives the manifest, it checks to see what bit rates are available and requests fragments of content starting at the lowest available bit rate. In response, the server prepares and sends the fragments by reading the data from the disk file encoded at that bit rate (using the mapping in the server manifest). The content is then displayed on the client.

The client gradually requests higher bit rates as allowed by the resource-monitoring logic, and eventually

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2037
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The PCI Data Standard: It's Everywhere You Want Your Credit Data to Be. Register now to download this whitepaper.

Ecommerce:

Award Winning Ecommerce Solutions: Volusion
Ecommerce Solution: NetSuite
Ecommerce Solutions

Online Security:

Online Security: Siemens Scalance Wireless
Verizon Business DSL: Internet Security
Internet Security

intra-not.com

FREE Online Event:
Presented by:
Jupiterwebcasts
May 23
2007
2:00 pm EDT
11:00 am PDT

Save, Simplify and Scale with Storage Virtualization

Gain INSIGHT from industry experts without leaving your desk

Webcast produced for:
StoreVault
A NetApp Division
REGISTER NOW

Essential Wi-Fi Research

Now Available!
802.11 Wireless LAN Security: Usage Expectations, & Strategies for the Future
More Research

Case Study: Learn How Virginia Tech Reduced Job-Processing by 92% with a Computer Cluster. Virginia Tech needed more computing power to run complex data & statistical analysis applications. Learn how they did it.

News

The Wireless Digital Picture Frame Arrives

By Eric Griffith

October 16, 2003

In the days of the Internet boom when companies believed any product could be a success -- no matter how useful or useless it might be (3Com Audrey? Microsoft Bob?) -- there came a product category that still seems like it could catch on at any moment: the digital picture frame.

Today, San Jose, Calif.-based start-up Wallflower Systems announced a digital picture frame for the wireless world. The eponymously named Wallflower is an 8-inch by 10 inch photo frame housing a 12.1-inch LCD screen with 16-bit color at 1024x768 pixel resolution -- and built in 802.11b.

Wallflower's marketing

London Stock Exchange Case Study:
Increasing Performance with .NET Framework

The London Stock Exchange needed a scalable, reliable, high-performance stock exchange ticker plant to replace its earlier system. Learn how with .NET Framework.

Click here to read!

Linux Reference Center
Sponsored By: Microsoft

intel Resource Center

Xeon Processors Packed with Latest Server Technologies
By adding more servers, the datacenter can quickly grow out of control, and bring power and cooling costs with it. Using new server technology available from Intel, however, IT managers can keep costs in check while running a more efficient operation. Click here.

Applying RFID Technology in High Volume Manufacturing
To explore the impact of radio frequency identification (RFID) technology in manufacturing, Intel□s Technology and Manufacturing Group deployed an RFID pilot within Intel□s largest semiconductor assembly and test facility. The pilot tracked 80,000 microprocessors from the end of the manufacturing line through Intel□s warehouse, into a major customer□s warehouse, and onto the customer□s factory floor. Click here.

Building a Real-World Model to Assess Virtualization Platforms
To drive business decisions in the data center, Intel IT created a performance-driven methodology to compare the operating costs of virtualization platforms. By measuring workload performance, platform performance, and power consumption they were able to approximate the total cost of ownership (TCO) of each platform. Click here.

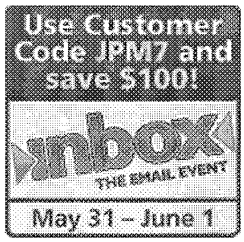
Building an Enterprise Data Warehouse and Business Intelligence Solution
To achieve end-to-end visibility into critical business functions across the company, Intel IT deployed an integrated warehouse solution. The solution employs a consolidated enterprise data warehouse along with business intelligence applications. Today□s EDW occupies over 20TB of usable data space, processes 90 billion rows of data per month, and is accessed by over 11,000 users worldwide. Click here.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2038
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Subscribe Now!
Wi-Fi Planet.com's Daily Newsletter
☑ html * ☐ text
your-email-addres
go
More Free Newsletters

Wi-Fi Glossary
Find a Wi-Fi Term
find

Wi-Fi® is a registered certification mark of the Wi-Fi Alliance



internet.commerce
Be a Commerce Partner
Giveaways
Cheap Airline Tickets
Memory Upgrades
Memory
Web Site Design
Franchise Directory
Condos For Sale
Promotional Items
2007 New Cars
Register Domain
10 Free Global Calls
Buy Text Link Ads
Help Desks
Merchant Accounts

internet.com
Developer
International
Internet Lists
Internet News
Internet Resources
IT
Linux/Open Source
Personal Technology
Small Business
Windows Technology
xSP Resources
Search internet.com

manager, Jason Enamait, says that unlike other digital picture frame from companies like Ceiva and digi-Frame, the Wallflower is itself the image server. It has an internal 1.2GB hard drive that will appear on the home network like any other network drive. Users on the network drop JPEG files on this drive and they're ready for display on the LCD. JPEGs are the only images currently supported.

Migrating Order Management to Itanium 2-Based Servers
Order management is the largest SAP R/3* system at Intel, and is a global mission-critical application. In one large migration, Intel upgraded the system□s hardware infrastructure, operating system, database, and backup software, and installed SAP support packs. Through extensive coordination and a phased approach, we accomplished the move to 64-bit Intel Itanium 2-based servers, with impressive performance gains. Click here.
Visit the Intel Resource Center

>> Wi-Fi Planet Marketplace

802.11n Webinar

Learn about myths of Wi-Fi interference cognio click >>
Easy to use
WiFi Spectrum Analyzer

Handheld
Wi-Fi / WiMAX / RFID
Interference Analysis

"The Wallflower cycles through pictures, but you can set them up in any order, or put them in random mode," says Enamait. Images can also be put into user-created directories. Entire directories can be turned on or off for display -- turn on those vacation pictures even you can't stand when company comes over.

The unit can talk with your wireless router to go out on the Internet and pull images from personal Web sites you designate, as well.

The Wallflower control interface is accessed via a Web browser. Type in the IP address for the product to bring up the interface like you would with a router. From there, you can manage the Wi-Fi settings. The unit comes with DHCP (define) on by default so it will try to connect to your wireless LAN automatically.

The next generation of Wallflower is already under construction by the company. Current plans are to build in speakers and audio-out jacks (including digital audio) and support for file types from MP3 to MPEG to Flash. The screen will be two inches bigger diagonally. The 2.0 product will also have better Internet connectivity, including "the kind of peer-to-peer sharing the RIAA hates," says Wallflower's president Mitch Kahn.

The current Wallflower's come in three styles (Cherry Blossom or White Plumeria for $599 each; Black Cherry for $649) but every style is limited, according to their Web site, since the frame part is hand-made. The prices are pretty steep compared to products like Ceiva's Digital Photo Receiver, which sells at retail for $149.95. The difference is, says Kahn, that Ceiva's product is subsidized by Ceiva service -- Ceiva owner's pay $7.95 a month to have the company send their own images to the frame for them. Wallflower instead hopes to build on the technology of Wi-Fi and broadband access, hoping that the wireless easy and a high quality display will make up for the high price.

"I'm tired of paying 10 bucks a month for [various] services," says Kahn. "Plus, if you line up products from digi-Frame, Ceiva, and Pacific Digital next to ours, our image quality is immensely better. It's like a painting next to a child's drawing."

RELATED ARTICLES

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2039
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Advertise
Corporate Info
Newsletters
Tech Jobs
E-mail Offers

Email this article to a colleague
Go to a printable version of this story

NETWORKING SOLUTIONS

- **DDoS resistant DNS** Self-protects against 1 Gbps attacks. Free whitepaper.
- Free Forrester Wave™ Report: WLAN Infrastructure Q2 2007. Compliments of Trapeze Networks.
- Webcast: Two Roads to Mainstream Virtualization. Learn more about virtual private servers & virtual machines.
- Webcast: Chilling Changes in the Server Room. Learn how to stave off high cooling bills & power costs.
- Whitepaper: The Decision to Deploy IP Telephony on Demand

London Stock Exchange Case Study:
Increasing Performance with .NET Framework

The London Stock Exchange needed a scalable, reliable, high-performance stock exchange ticker plant to replace its earlier system. Learn how with .NET Framework.

Click here to read

Linux Reference Center

Sponsored By: Microsoft

JupiterWeb networks:

Internet.com | EARTHWEB | graphics.com

Search JupiterWeb: Find

Jupitermedia Corporation has two divisions: Jupiterimages and JupiterWeb

Jupitermedia Corporate Info

Copyright 2007 Jupitermedia Corporation All Rights Reserved.
Legal Notices, Licensing, Reprints, & Permissions, Privacy Policy.

Web Hosting | Newsletters | Tech Jobs | Shopping | E-mail Offers

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2040
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

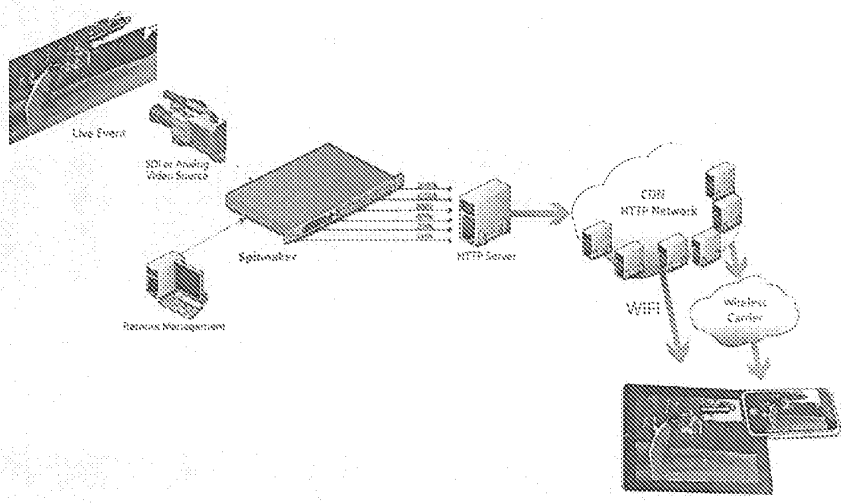
INLET TECHNOLOGIES

SOLUTION BRIEF

Adaptive Delivery to iDevices

Inlet Technologies' Spinnaker™ paves the way

Delivering live video to the most popular connected devices on the market isn't that complicated -- when you select the class leading solution. Spinnaker™—Inlet Technologies' award-winning family of streaming appliances for live audio and video delivery— eliminates the need for additional hardware to reach the Apple® iPhone® and iPad™ with video.



Apple's iOS 4.0 release extends a now proven approach to delivering video to handsets, and Inlet Technologies is at the forefront of this wave. The iPhone and iPad continue to deliver adaptive bitrate live video, with or without an application on the device.

Inlet Technologies collaborates directly with Apple on this architecture and is the de facto solution for broadcasters, media & entertainment enterprises and sports leagues to deliver content to iOS powered devices. Spinnaker has been the engine for nearly every Apple HTTP Live Streaming event since the launch of this technology, and we continue to innovate to enable our customers with the absolute latest technology in the space and the best features. Spinnaker fully supports adaptive bitrate (ABR) delivery technology, allowing viewers to move seamlessly from one resolution and data rate to another without buffering, pausing, or missing a moment of their video. This approach provides the best viewing experience possible, while offering exciting features we've come to expect in other devices, like the ability to pause/rewind/fast forward through content, as we do on our DVRs. With its 5.0 software release, Inlet's Spinnaker supports this effort out of the box and is paving the way to monetization on the iPhone and iPad via our live ad insertion capabilities.

HOW DOES IT WORK?

Spinnaker takes the source video signal and produces H.264/AAC encoded content formatted for adaptive delivery to the iPhone and iPad.

Under the hood, Spinnaker creates MPEG-2 transport streams (.ts), then takes those streams and divides them into small files of equal duration. The Spinnaker also creates an index file (.m3u8) that contains references to these smaller files. This index is continually updated as long as the stream is live. The index and media files can then be deployed using common web server (HTTP) infrastructure - which is the protocol used to serve content to the iPhone and iPad.

HTTP AND ADAPTIVE BITRATE (ABR)

Why HTTP? HTTP is known as a "stateless" protocol, meaning if an HTTP client requests some data, the server responds by sending the data, but won't remember the client or its state. Each HTTP request is handled as a completely standalone one-time session.

By using adaptive delivery methodology, we eliminate the "forgetfulness" factor, because the files are delivered in bite sized chunks, and the player is periodically monitoring bandwidth and switching between streams as needed.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2041
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

HTTP is the ideal mechanism for ingesting segmented TS content to web servers and content delivery networks (CDNs), because:

- HTTP is simple and supported by nearly all platforms
- The mechanism is easy to implement and maintain
- HTTP POST is a "push" from the encoder, and works from behind firewalls/NAT
- There is very little protocol overhead, in terms of bytes or cycles
- In many cases a persistent connection will be maintained

To anticipate changes in bandwidth fluctuation on the networks and on iOS 4, you should plan to offer multiple discrete bitrates. This approach will allow for the iPhone and iPad's native player to periodically check its connection and enable it to pull these discrete data rates and inherent files, switching up and down between them as bandwidth improves or degrades. This is the heart of ABR delivery methodology.

SETTING IT UP

So----referencing the diagram on the previous page----to enable this in your enterprise, you will need:

- Spinnaker (S-5000 for SD, S-7100/S-8100 for HD content, S-6000 for IPTV)
- Web server or CDN to serve the HTTP content

Spinnaker is pre-configured with the settings needed to deliver the streams required -- for iPhone or iPad. All you'll need to do is point the Spinnaker to your CDN or web server, so that it can publish the files there. You can use many different protocols, each with their own benefit and risk.

Most CDNs and web servers will support several ways of ingesting content:

- HTTP (POST / DELETE) - Of all transfer mechanisms, HTTP is the preferred protocol for live ingest
- File Transfer Protocol (FTP) - FTP has more overhead, complexity, and is more likely to cause issues with firewalls
- RSYNC - RSYNC has more overhead, particularly as the number of files increases
- Secure Copy (SCP) and SSH File Transfer Protocol (SFTP) - SCP and SFTP are much more complex to implement and also have more overhead than HTTP, but may be desirable if your content requires higher levels of security during transit. Keep in mind as well that our built-in segmenter can encrypt the segments using AES-128 encryption.

PERFORMANCE GRADE RELIABILITY AND FEATURES

Spinnaker is a tailor made choice for iOS adaptive delivery. Spinnaker's award-winning web-based interface gives you the flexibility of controlling the encoder from any networked computer. You can use one of the pre-loaded presets optimized by the experts at Inlet, or take advantage of Spinnaker's almost unlimited----and unmatched----ability to customize encoding settings. Spinnaker will also automatically detect your video source, and even let you preview your video source before starting the encoder. Additionally, Spinnaker comes with built-in authentication to deliver content to the major CDNs. All Spinnakers are field upgradeable to the latest software release, meaning you won't have to throw away your equipment when Inlet develops new software features.

STATE OF THE ART + AND BEYOND

Embrace the mobile streaming revolution and let Spinnaker make the transition fast and easy. In addition, take advantage of our industry leading reach and partnerships. For iOS live delivery, Inlet has partnered with Akamai to help our customers quickly enable this solution.

Want to learn more? Give us a call at +1-919-856-1080, send an email to sales@inletkd.com, or visit us at www.InletHD.com.

INLET TECHNOLOGIES

3535 Cornwallis Rd. | Research Triangle Park, NC 27709 | 919.856.1080 | www.inlethd.com

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2042
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

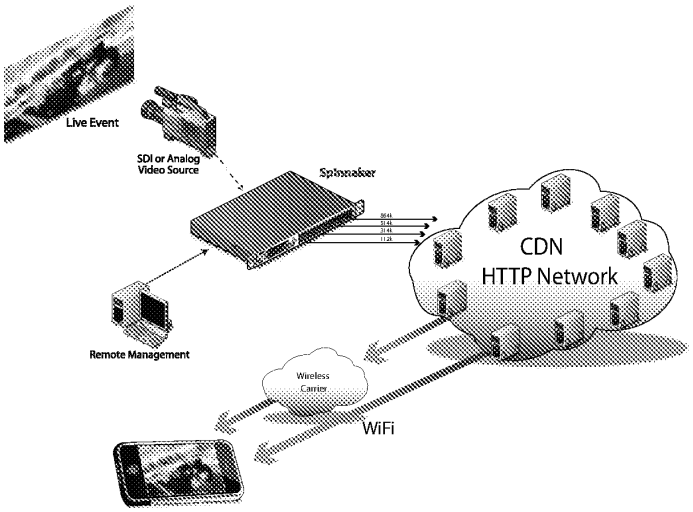
Adaptive delivery to iPhone 3.0

Spinnaker paves the way - and eases the path

SOLUTION BRIEF

INLET TECHNOLOGIES

Delivering live video to the most popular mobile device on the market just got easier. Spinnaker™—Inlet Technologies' award-winning family of streaming appliances for live audio and video delivery—eliminates the need for additional hardware to reach the Apple® iPhone™ with video.



Apple's iPhone 3.0 OS release enables a new approach to delivering video to handsets, and Inlet Technologies is at the forefront of this wave. The iPhone now supports adaptive delivery for live video, with or without an application on the iPhone.

Inlet Technologies is working directly with Apple on this architecture and is the de facto solution for broadcasters, media & entertainment enterprises and sports leagues to deliver content to the iPhone. Inlet supports adaptive bitrate (ABR) delivery technology, allowing viewers to move seamlessly from one resolution and data rate to another without buffering, pausing, or missing a moment of their video. This approach provides the best viewing experience possible, while offering exciting features we've come to expect in other devices, like the ability to pause/rewind/fast forward through content, as we do on our DVRs. With its 4.1 software release, Inlet's Spinnaker supports this effort out of the box and has removed the need for separate Mac hardware and the Stream Segmenter many other solutions require.

How does it work?

Spinnaker takes the source video signal and produces H.264/AAC encoded content formatted for adaptive delivery to the iPhone. Under the hood, Spinnaker creates MPEG-2 transport streams (.ts), then takes those streams and divides them into small files of equal duration. The Spinnaker also creates an index file (.m3u8) that contains references to these smaller files. This index is continually updated as long as the stream is live. This index and media files can then be deployed using common web server (HTTP) infrastructure - which is the protocol used to serve content to the iPhone.

HTTP and Adaptive Bitrate (ABR)

Why HTTP? HTTP is known as a "stateless" protocol, meaning if an HTTP client requests some data, the server responds by sending the data, but won't remember the client or its state. Each HTTP request is handled as a completely standalone one-time session. By using adaptive delivery methodology, we eliminate the "forgetfulness" factor, because the files are delivered in bite sized chunks, and the player is periodically monitoring bandwidth and switching between streams as needed.

HTTP is the ideal mechanism for ingesting segmented TS content to web servers and content delivery networks (CDNs).

- HTTP is simple and supported by nearly all platforms
- The mechanism is easy to implement and maintain
- HTTP POST is a "push" from the encoder, and works from behind firewalls/NAT
- There is very little protocol overhead, in terms of bytes or cycles
- In many cases a persistent connection will be maintained

To anticipate changes in bandwidth on the iPhone, you should plan to offer at least two discrete bitrates, and, depending on the size of your anticipated audience, as many as four total. This design will allow for the iPhone's player to periodically check its connection and ability to pull these files, switching up and down between them as bandwidth degrades or improves. This is the heart of ABR delivery methodology.

Setting it up

So—referencing the diagram on the previous page—to enable this in your enterprise, you will need:

- Spinnaker (S-5000 for SD, S-7000 for HD content)
- Web server or CDN to serve the HTTP content

Spinnaker is pre-configured with templatized settings needed to deliver the streams required. All you'll need to do is point the Spinnaker to your CDN or web server, so that it can publish the files there. You can use many protocols, each with their own benefit and risk.

Most CDNs and web servers will support several ways of ingesting content:

- HTTP (POST / DELETE)
- File Transfer Protocol (FTP)
- rsync
- Secure Copy (SCP)
- SSH File Transfer Protocol (SFTP)

Of all transfer mechanisms, HTTP is the preferred protocol for live ingest. FTP has more overhead, complexity, and is more likely to cause issues with firewalls. Rsync has more overhead, particularly as the number of files increases. SCP and SFTP are much more complex to implement and also have more overhead than HTTP, but may be desirable if your content requires higher levels of security during transit. Keep in mind as well that the Apple Segmenter can encrypt the segments using AES-128 encryption.

Performance grade reliability and features

Spinnaker is a tailor made choice for iPhone adaptive delivery. Spinnaker's award-winning web-based interface gives you the flexibility of controlling the encoder from any networked computer. You can use one of the pre-loaded presets optimized by the experts at Inlet, or take advantage of Spinnaker's almost unlimited—and unmatched—ability to customize encoding settings. Spinnaker will also automatically detect your video source, and even let you preview your video source before starting the encoder. Additionally, Spinnaker comes with built-in authentication to deliver content to the major CDNs. All Spinnakers are field upgradeable to the latest software release, meaning you won't have to throw away your equipment when Inlet develops new software features.

State of the art - and beyond

Embrace the mobile streaming revolution and let Spinnaker make the transition fast and easy. In addition, take advantage of our industry leading reach and partnerships. For iPhone live delivery, Inlet has partnered with Akamai to help our customers quickly enable this solution. For more details on our partnership with Akamai and to see this groundbreaking technology in action, head to iphone.akamai.com.

Want to learn more? Give us a call at 1-866-96-INLET or visit us at www.InletHD.com.

INLET® TECHNOLOGIES

Inlet Technologies
1121 Situs Court, Suite 330
Raleigh, NC 27606

1.866.966.INLET
www.InletHD.com

Specifications and product availability subject to change without notice.
© 2009 Inlet Technologies. All rights reserved. v1.1 0809

HTTP versus RTMP

Which way to go - and why?

SOLUTION BRIEF

INTELLIGENT CONTENT DELIVERY

Like picking the color of your next car, even rich media delivery offers choice. Streaming has seen a lot of changes over the past several years, and continues of late with varied solutions utilizing different transport protocols. Each transport offers its own benefits and cost structure. How and why you integrate each will depend on your specific application and how each protocol applies to you. In this brief, we'll explore HTTP and RTMP and attempt to identify the key data points, considerations and examples to help you make decisions regarding your enterprise.

What's it all about?

As is the case in the media delivery and streaming business, things are continuously evolving. New technologies from Apple, Adobe and Microsoft have introduced new methods of delivery that improve user experience and startup time, and add features and functionality. To cover all of these offerings, we use the term adaptive bitrate (ABR) to denote that the technology provides for varying experiences, depending on the conditions on the user's side. These primary changes in the present digital media delivery landscape borrow from the old adage: what is old is new again. In the early days of web video delivery, we all had to rely on progressive delivery (also referred to as PDL - progressive download) of video. What that meant is that the bits of video were delivered to your player, one packet at a time "in the blind," with no communication between the server and player. Once you had a resonable percentage of the file downloaded to disk, the player would begin playing the file. But too often, the player caught up with the point at which the file was being delivered and playback halted. So streaming was created: a mode through which the video would be passed to the player, with increased communication and monitoring in place, happening in real time between the player and server. If bandwidth degraded on the player side, it would signal the server and "buffer" until it could obtain a suitable amount of packets of video to resume playback. Today, more video technologies are leveraging HTTP again and reinventing your father's progessive download mechanism. By building in monitoring and quality of service (QoS) measurements, HTTP can provide vibrant video experiences, while helping with scale and access issues we'll touch on below. The user with plentiful broadband is rewarded with the absolute best video offering available, while the user experiencing fluctuations in bandwidth or low grade connectivity will still receive the same video, albeit at a lower quality experience.

All these acronyms

HTTP (hypertext tranfser protocol) refers to the protocol used to deliver web pages and images across the Internet worldwide. HTTP is an adopted, open standard—the most ubuitous mode of delivery online. HTTP is a "stateless" protocol; think of it as an airline ticket to anywhere. HTTP can be delivered by a variety of web servers, both commercial and open source.

Real Time Streaming Protocol (RTSP) is a network control protocol used in entertainment and communications systems to control streaming media servers. RTSP is used to establish and control media sessions between two points, usually server and player client. Clients of media servers issue VCR-like commands, such as PLAY and PAUSE, to facilitate real-time control of playback of media files from the server. Microsoft's Smooth Streaming is a hybrid delivery method that acts like traditional RTSP streaming, but is based on HTTP progressive download. Apple's HTTP adaptive streaming is another example of using HTTP to deliver video with new tecniques.

RTMP (real time messaging protocol) refers to the proprietary protocol developed by Adobe Systems for streaming audio, video and data over the Internet, between a Flash player and a Flash Media Server. Like RTSP, this is an example of a traditional streaming protocol, though only one of many versions of streaming protocols for the web. RTMP is defined as a stateful protocol, which means that from the first time a client connects until the time it disconnects, the streaming server keeps track of the client's actions. The client communicates its actions, or "states" to the server by issuing commands such as PLAY or PAUSE. After a session between the client and the server has been established, the server begins sending the media as a steady stream of small information packets. This behavior continues and repeats until the session is closed by the server or player client. In June of 2009, Adobe Systems published the RTMP specification. What this means is that content distribution networks (CDNs) and enterprises that use Flash technology can now author their own implementation - which may result in Flash delivery, even the new Dynamic Streaming, enabled natively in HTTP.

Pros and Cons

Which technology and delivery mode will work best for you? The answer depends on many data points. Let's explore a few and try to narrow down the keys. How large is your anticipated daily traffic or single event? Do you need to protect your content from being downloaded locally and republished by the user? (Think of a YouTube scenario where your favorite television show is available there, but probably shouldn't be.) What is the length of your content: many files with short duration, or fewer files with very long length?

HTTP is less likely to be disallowed by routers, NAT, or firewall settings. No ports need to be opened that are commonly closed by default. Content is therefore more likely to get through to the client in more locations and without special settings. HTTP is also supported by more CDNs, which can affect cost in large distribution models. In general, more available hardware and software works unmodified and as intended with HTTP than with RTSP/RTMP. Expertise in customizing HTTP content delivery using tools such as PHP is also more widespread. Additionally, for large-scale events, HTTP natively and easily supports mirroring and edge caching. This provides for massive scale expansion when needed for the largest events. RTSP/RTMP can also be cached, but HTTP does so natively and without need for proprietary or custom configurations.

Access is another consideration. Can everyone view your content once you make it available? From a site visitor's point of view, one advantage of using HTTP is access. Many corporate networks use firewalls to block specific content. Popular methods of blocking are protocol and port restrictions. Some firewall rules allow only HTTP content served over port 80. However, this luxury isn't always shared with RTMP-delivered Flash Video. The default port for RTMP connections is 1935, which may not be allowed on tight firewalls. If the Flash Player's first attempt to play video over port 1935 fails, it will try to reconnect using a few different methods. Bottom line: if you don't want to deal with firewalls and proxies, you should consider HTTP. One benefit with RTMP to point out here is its ability to provide multicast support. If you run an enterprise and want to take one stream inside your corporate network and deliver it to many without initiating a new connection for each user, RTMP is the way to go. HTTP does not provide this functionality, nor do CDNs.

Security is a big deal with a lot of content, and if you're a distributor or aggregator, you know the term digital rights management (DRM) all too well. Can or will you allow your content to be saved locally on the user's machine? RTMP has broad DRM support and rights holders are very familar with the ecosystem. For example - in the RTMP camp is RTMPE - the 'E' standing for encryption. This is a new method for real-time cryptographic protection of content offered by Adobe. Some HTTP approaches like Smooth Streaming are rapidly moving toward deployment of solutions for DRM as well. However, securing your content may be a concern. Some ABR technologies will provide for secure transmission via HTTPS. This is the secure way of transmitting HTTP video but isn't a commonly accepted form of digital rights management (DRM) and may provide a contractual and legal hurdle with rights holders who stipulate that DRM be used to protect their content. As a mode of transport, HTTP can be captured and saved locally. Chunked delivery can make this local capture more challenging, but the ability still exists. Conversely, RTMP is much less susceptible to this, but there have been rare instances where streams have been circumvented and saved locally using third party applications.

HTTP vs. RTMP - Quick Reference

	HTTP	*RTMP*
Server Components	Open	Specialized
Security/IP Protection	Basic	Enhanced
Caching/Scale	Native	Specialized
Player Interactivity	Basic	Extensive
Multicast Support	N/A	Supported
Plug-in Penetration	Nascent	Extensive
Firewall (port/protocol)	No restrictions	Some restrictions
Variable Bitrate Support	No impact	Susceptible to data spikes

The availablility of plugins may be a concern to you as you consider these technologies. Flash is the most broadly deployed plugin of the bunch, but there are caveats. Dynamic Streaming requires Flash Player 10, which is growing in adoption (and pegged at nearly 80% as of July 2009), but has not reached full market penetration yet. Microsoft's Silverlight has broad support on new PC builds, and is being used for some large scale sports events (March Madness, Wimbledon, 2010 Winter Olympics) but hasn't yet achieved the reach that Flash currently enjoys. Additionally, technlogies from third party providers like Swarmcast and Conviva that utilize HTTP are in use at organizations like MLB and Indy Racing. Bottom line: if your video is compelling and users are required to install one plugin or another, most will.

Tracking your content and how many people view it is a "must have" for most, but how granular must your details be? Because of the nature of RTMP we covered earlier—where the player and server are in constant communication—reporting within RTMP is much more detailed and granular when compared to HTTP. The logs derived from RTMP servers provide very specific data about what preceisely a user did within a video. How long did they watch? At what point in the video did they disconnect? Did they grab the "scroll bar" and advance through the video to a specific point? All these questions can be answered with an RTMP-based solution today. Expect HTTP driven technologies to improve on this deficiency in the future, but reporting is more robust with RTMP now.

Lastly, and perhaps most importantly, what will it cost you to provide this content to your users? What will a CDN charge you to provide your content in one mode versus the other? In years past, many CDNs charged a premium for RTMP streaming transit, due to the specialzed skills needed to setup and manage those servers and the complexity in ensuring the content could get to the user. Today, the "Flash Tax" is gone and the total cost to deliver content via RTMP has decreased. However, HTTP still represents the lion's share of web traffic. Consider that pushing video via those same protocols and methods enables CDNs and those who own their infrastructure to realize economies of scale and the cost savings that accompany them. As a result, your total cost of ownership (TCO) may be less with HTTP versus RTMP.

Summary

So picking a delivery method may not be as easy as picking the color of that new car. All three major ABR technologies have their benefits and risk. Adobe's Flash is the ubuitous choice for video on demand (VoD) and has tremendous penetration on the desktop. Microsoft's Smooth Streaming has had some early live event sucesses and will help provide video for the 2010 Winter Olympics. Apple's HTTP adaptive streaming is available on 25 million handsets, but has limited desktop reach and longer latency and startup because of the data network it operates on.

We hope we've helped answer some of those questions and assisted you in deciding what technology suits you best. The good news is that no matter whether you choose HTTP, RTMP, or both, our family of products supports both technologies now: out of the box, with ease of setup and use and a proven track record with marquee customers. We're ready to help you assemble your next digital media solution at Inlet Technologies. Give us a call at the number below to get rolling.

Inlet Technologies
1121 Situs Court, Suite 330
Raleigh, NC 27606

phone: 1.866.966.INLET
www.inlethd.com

Specifications and product availability subject to change without notice.
© 2009 Inlet Technologies. All rights reserved.

The World's First Live Smooth Streaming Event: The French Open

CASE STUDY

INLET TECHNOLOGIES

Situation

The French Open, organized by the French Tennis Federation, is one of the premier events in the professional tennis calendar, showcasing the best tennis talent in the world playing in the world renowned Roland Garros stadium. In 2008, the Open was broadcast in 198 countries to a global audience of millions.

In 2009, France Télévisions wished to offer tennis fans in France an online high definition (HD) experience to complement the extensive coverage on its three broadcast channels (France 2, France 3 and France 4).

France Télévisions chose to partner with Microsoft, Inlet Technologies and Level 3 Communications to deliver the event online in HD, using Smooth Streaming technology. This was the first live HD Smooth Streaming public event in the world.

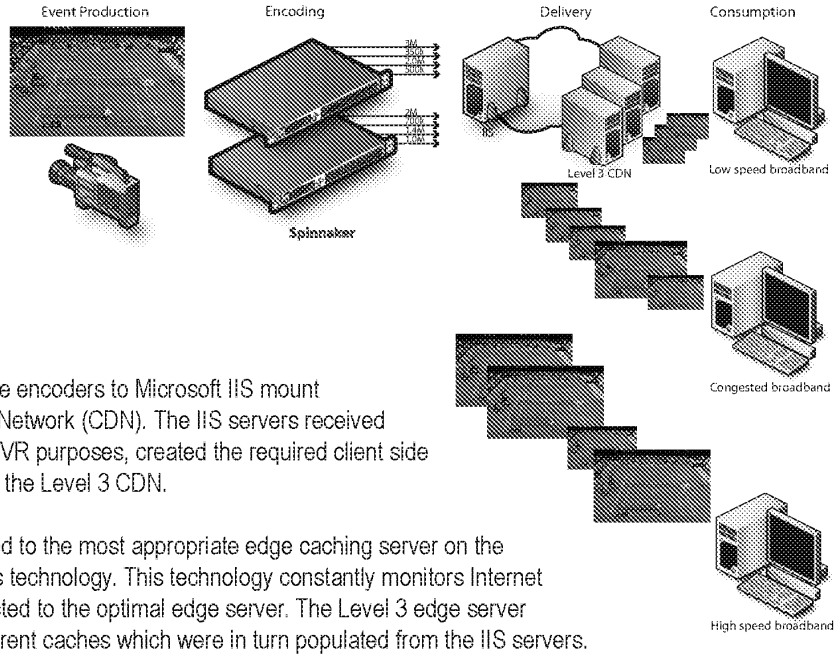
Smooth Streaming is an advanced media delivery method that switches seamlessly between multiple streams depending on the end user's available bandwidth and PC performance at any given moment. The best possible quality video stream is automatically delivered without stuttering, dropped frames or buffering. The process results in a much improved user experience, offering programmers the ability to offer true HD (720p+) streams to customers who have sufficient bandwidth, without alienating those who have lower speed connections.

The advantages of this type of technology are significant. Since viewers have a more positive experience watching live video online, they are more likely to watch for longer periods and more frequently. This in turn provides content owners greater monetization opportunities.

Solution

Microsoft, Inlet and Level 3 rapidly brought together an end-to-end solution that combined the latest encoding, delivery and player technology.

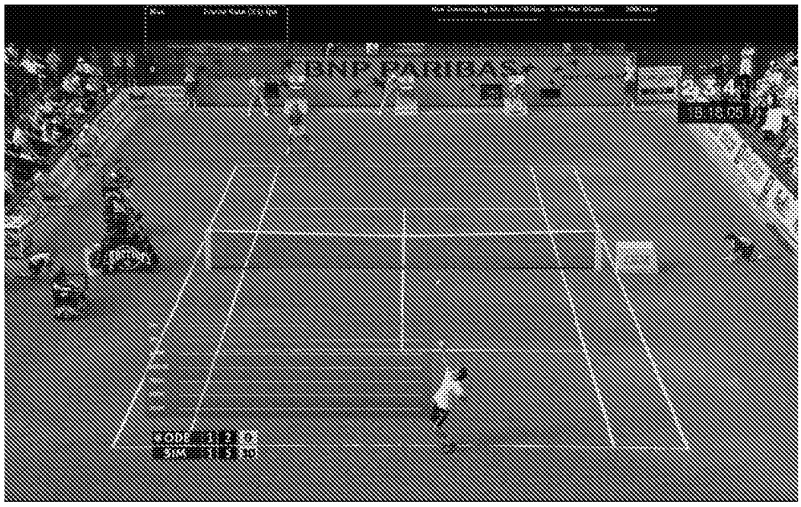
The tennis action was first captured and edited on site at the French Tennis Federation in a 1080i50 HD format. The production broadcast signal was then delivered to on-site Inlet Spinnaker encoders through standard HD-SDI connections. The HD broadcast feed was synchronized across two Spinnakers to deliver 7 time-aligned streams, ranging from 350 kilobits per second to 3 megabits per second.



The video feeds were then pushed from the encoders to Microsoft IIS mount points within the Level 3 Content Delivery Network (CDN). The IIS servers received the multiple video feeds, stored them for DVR purposes, created the required client side description files, and acted as an origin for the Level 3 CDN.

End user requests for content were directed to the most appropriate edge caching server on the Level 3 CDN via a DNS-based rendezvous technology. This technology constantly monitors Internet conditions to ensure end users are connected to the optimal edge server. The Level 3 edge server pulled the requested feed from regional parent caches which were in turn populated from the IIS servers. The edge server then delivered the requested feed to the consumer over HTTP, taking advantage of the massive scale of the Level 3 CDN. The end to end delivery was monitored from the Level 3 Broadcast Operations Centre.

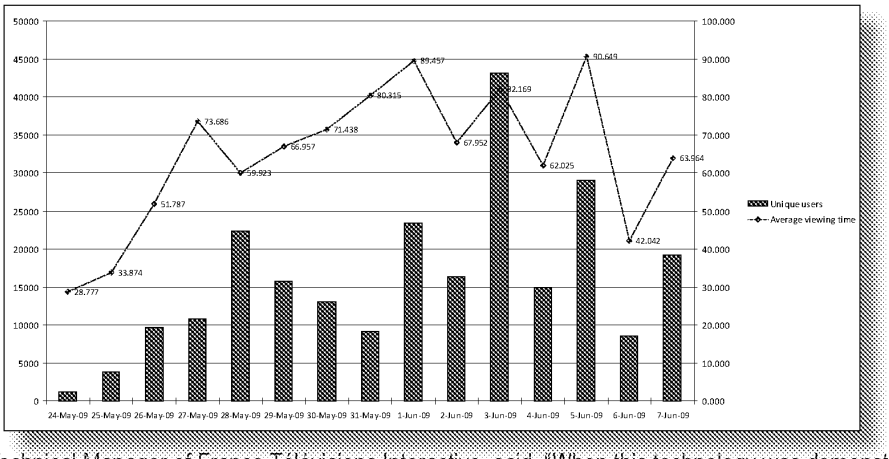
Finally, the Microsoft Silverlight player dynamically detected and seamlessly switched between feeds from the Level 3 CDN based on local client conditions. Consumers with high bandwidth connections were able to experience true 720p HD and others with lower bandwidth speeds or lower powered machines received the appropriate stream for their connectivity.



Screen shot of the live stream showing player switching between feeds

Results

Tennis fans in France were able to experience true HD quality over the internet. Of the approximately 140,000 unique viewers, more than half watched the action in high definition. Viewers who watched the live stream watched it for an average of 64 minutes (this is significant, as the average time people spend watching a single video online is well under 10 minutes.) Eighty-five percent of users watched live.



Philippe Daguerre, Technical Manager of France Télévisions Interactive, said, "When this technology was demonstrated for me, I was amazed by the overall quality of experience. I knew this was what we needed to deliver to our viewers. And now that we have had such tremendous success with the French Open, we have been implementing Smooth Streaming for other major sporting events in France."

Conclusion

This event—as the world's first live public HD Smooth Streaming deployment—shows the tremendous potential of adaptive bitrate technology. The impressive statistics, especially time spent watching, show the impact of the dramatically improved quality of user experience. Given the success of this event, France Télévisions and other major European and American broadcasters are implementing Smooth Streaming for live sporting events such as the Confederations Cup, the World Athletics and Swimming Championships, Wimbledon, and the Tour de France.

INLET® TECHNOLOGIES

www.inlethd.com

Level 3® COMMUNICATIONS

www.level3.com

Specifications and product availability subject to change without notice.
© 2009 Inlet Technologies. All rights reserved.

CNET News

December 17, 1997 5:40 PM PST

IBM closes Cryptolopes unit

By Tim Clark [mailto:letters@news.com?subject=FEEDBACK: IBM closes Cryptolopes unit]
Staff Writer, CNET News

Related Stories

IBM reworks e-commerce lineup

[http://www.cnet.com/IBM-reworks-e-commerce-lineup/2100-1017_3-203017.html] September 8, 1997

Big Blue to open the till

[http://www.cnet.com/Big-Blue-to-open-the-till/2100-1017_3-201808.html] July 25, 1997

Softbank drives hard on commerce

[http://www.cnet.com/Softbank-drives-hard-on-commerce/2100-1017_3-200818.html] June 23, 1997

IBM makes cryptolope deal

[http://www.cnet.com/IBM-makes-cryptolope-deal/2100-1017_3-254146.html] December 11, 1996

Hackers can't interlope a Cryptolope

[http://www.cnet.com/Hackers-cant-interlope-a-Cryptolope/2100-1001_3-210158.html] April 22, 1996

IBM [http://www.ibm.com/] (IBM) [http://www.dbc.com/cgi-bin/htx.exe/squote?source=blq/cnet&ticker=IBM] is closing the operation developing its highly touted Cryptolopes software, a secure container technology that is used for sending content over the Internet and tracking intellectual property rights.

Elements of the technology may be moved into the Lotus Notes groupware product, into IBM's net.Commerce merchant server, or both.

"We will not bring Cryptolopes Live to market as planned," IBM spokeswoman Trink Guarino said, referring to a Java-based version of the technology that has been in beta testing for several months. "The beta told us that our customers would prefer to see

that technology incorporated in other applications."

The unit's demise marks a retreat from a high-profile campaign on the technology launched by IBM last year. The Cryptolopes Live beta announcement in September was featured among several initiatives at a major e-commerce trade show, and the company had showcased the effort at earlier events, too.

Decisions on the product's future are expected in the first half of 1998, Guarino added, and employees in the unit are likely to be offered transfers. IBM was unable to say how many employees worked on the product, but it was believed to be no more than 50.

The Cryptolope Live product built a secure wrapper around digital content or software applications so they could be sent safely over the Internet. It also included rules on how the information inside could be used--pricing per usage, free under certain conditions, and so on.

The setup also included software tools and a "license clearinghouse" server to handle "rights management," as the product category is called. IBM and competitors in the area say the end of Cryptolopes does not mean that the category of protecting intellectual property rights through software is dead.

"Customers don't want to worry themselves about what makes the best sense [in how to use Cryptolopes]," Guarino noted. "They want the whole process in an application. It's still a promising field."

"I believe, as does IBM, that this has nothing to do with the viability of the market or the need for this," said Paul Bandrowski, president of Softbank Net Solutions [http://www.sbnetsolutions.com/], which is building a service to handle rights management not only for application software but also for music, medical records, documents, and other vertical industries.

"We have not seen Cryptolopes as a competitor to us in the electronic software distribution space for quite a while," said Kate Joseph, vice president of product marketing for Portland Software [http://www.portsoft.com/], which offers a system to distribute software over the Internet.

IBM has taken several tacks with the Cryptolopes. The technology originated as part of a development effort called Databolts.

Then Big Blue offered a service called InfoSage to handle content distribution for publishers, but about a year ago it shuttered that operation, saying it competed with IBM's publishing customers. InfoSage had been part of IBM's Infomarket operation, a service that is likewise closing down.

In January, IBM began rewriting Cryptolopes in Java, a task that was finished by September, at which point it went into beta.

Bandrowski thinks the Cryptolopes effort was not well suited to a company as large as IBM. "People want a solution," he said. "They don't want to buy technology to build the infrastructure to do it. They want the system to have flexibility for their needs."

But Bandrowski said he's sorry in some ways that IBM is dropping out. He noted, "They are a good milestone of where things are in the market."

@CBS Interactive. All rights reserved.

INTERNATIONAL STANDARD

ISO/IEC 14496-2

Second edition
2001-12-01

Information technology — Coding of audio-visual objects —

Part 2: Visual

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

Adopted by NCITS (InterNational Committee for Information Technology Standards) as an American National Standard.

Date of ANSI Approval: 8/28/02

Published by American National Standards Institute.
25 West 43rd Street, New York, New York 10036

Copyright 2002 by Information Technology Industry Council (ITI).
All rights reserved.

These materials are subject to copyright claims of International Standardization Organization (ISO), International Electrotechnical Commission (IEC), American National Standards Institute (ANSI), and Information Technology Industry Council (ITI). Not for resale. No part of this publication may be reproduced in any form, including an electronic retrieval system, without the prior written permission of ITI. All requests pertaining to this standard should be submitted to ITI, 1250 Eye Street NW, Washington, DC 20005.

Printed in the United States of America

Reference number
ISO/IEC 14496-2:2001(E)

©ISO/IEC 2001

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ADAMSON
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2001

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/ORDER #1000000
ANSI Store order #3074457, Downloaded: 7/18/2007 1:06:21 PM EST
Single user license only. Copying and networking prohibited.

Contents

1 Scope 1

2 Normative references 1

3 Terms and definitions 2

4 Abbreviations and symbols 12
4.1 Arithmetic operators 12
4.2 Logical operators 13
4.3 Relational operators 13
4.4 Bitwise operators 13
4.5 Conditional operators 13
4.6 Assignment 13
4.7 Mnemonics 14
4.8 Constants 14

5 Conventions 14
5.1 Method of describing bitstream syntax 14
5.2 Definition of functions 15
5.2.1 Definition of next_bits() function 15
5.2.2 Definition of bytealigned() function 15
5.2.3 Definition of nextbits_bytealigned() function 15
5.2.4 Definition of next_start_code() function 16
5.2.5 Definition of next_resync_marker() function 16
5.2.6 Definition of transparent_mb() function 16
5.2.7 Definition of transparent_block() function 16
5.2.8 Definition of byte_align_for_upstream() function 16
5.3 Reserved, forbidden and marker_bit 16
5.4 Arithmetic precision 17

6 Visual bitstream syntax and semantics 17
6.1 Structure of coded visual data 17
6.1.1 Visual object sequence 18
6.1.2 Visual object 18
6.1.3 Video object 18
6.1.4 Mesh object 24
6.1.5 FBA object 25
6.1.6 3D Mesh Object 29
6.2 Visual bitstream syntax 30
6.2.1 Start codes 30
6.2.2 Visual Object Sequence and Visual Object 34
6.2.3 Video Object Layer 35
6.2.4 Group of Video Object Plane 40
6.2.5 Video Object Plane and Video Plane with Short Header 40
6.2.6 Macroblock 53
6.2.7 Block 58
6.2.8 Still Texture Object 59
6.2.9 Mesh Object 73
6.2.10 FBA Object 75
6.2.11 3D Mesh Object 85
6.2.12 Upstream message 103
6.3 Visual bitstream semantics 104
6.3.1 Semantic rules for higher syntactic structures 104
6.3.2 Visual Object Sequence and Visual Object 104
6.3.3 Video Object Layer 109

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #1537445, Downloaded: 7/19/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2055
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.3.4 Group of Video Object Plane 120
6.3.5 Video Object Plane and Video Plane with Short Header 120
6.3.6 Macroblock related 131
6.3.7 Block related 134
6.3.8 Still texture object 135
6.3.9 Mesh object 142
6.3.10 FBA object 144
6.3.11 3D Mesh Object 151
6.3.12 Upstream message 162

7 The visual decoding process 164
7.1 Video decoding process 165
7.2 Higher syntactic structures 166
7.3 VOP reconstruction 166
7.4 Texture decoding 166
7.4.1 Variable length decoding 167
7.4.2 Inverse scan 168
7.4.3 DC and AC prediction for intra macroblocks 169
7.4.4 Inverse quantisation 172
7.4.5 Inverse DCT 175
7.4.6 Upsampling of the inverse DCT output for Reduced Resolution VOP 176
7.5 Shape decoding 177
7.5.1 Higher syntactic structures 177
7.5.2 Macroblock decoding 178
7.5.3 Arithmetic decoding 187
7.5.4 Spatial scalable binary shape decoding 189
7.5.5 Grayscale Shape Decoding 198
7.5.6 Multiple Auxiliary Component Decoding 201
7.6 Motion compensation decoding 201
7.6.1 Padding process 201
7.6.2 Sample interpolation for non-integer motion vectors 205
7.6.3 General motion vector decoding process 207
7.6.4 Unrestricted motion compensation 209
7.6.5 Vector decoding processing and motion-compensation in progressive P- and S(GMC)-VOP 210
7.6.6 Overlapped motion compensation 212
7.6.7 Temporal prediction structure 213
7.6.8 Vector decoding process of non-scalable progressive B-VOPs 214
7.6.9 Motion compensation in non-scalable progressive B-VOPs 214
7.6.10 Motion Compensation Decoding of Reduced Resolution VOP 219
7.7 Interlaced video decoding 224
7.7.1 Field DCT and DC and AC Prediction 224
7.7.2 Motion compensation 225
7.8 Sprite decoding 234
7.8.1 Higher syntactic structures 234
7.8.2 Sprite Reconstruction 235
7.8.3 Low-latency sprite reconstruction 235
7.8.4 Sprite reference point decoding 236
7.8.5 Warping 237
7.8.6 Sample reconstruction 239
7.8.7 GMC decoding 240
7.9 Generalized scalable decoding 241
7.9.1 Temporal scalability 241
7.9.2 Spatial scalability 246
7.10 Still texture object decoding 251
7.10.1 Decoding of the DC subband 251
7.10.2 ZeroTree Decoding of the Higher Bands 252
7.10.3 Inverse Quantisation 257
7.10.4 Still Texture Error Resilience 265
7.10.5 Wavelet Tiling 268
7.10.6 Scalable binary shape object decoding 270

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOLA V. ALAMURU
ANSI Store order #0378487 Downloaded: 7/19/2007 3:09:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2056
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

7.11 Mesh object decoding 276
7.11.1 Mesh geometry decoding 276
7.11.2 Decoding of mesh motion vectors 279
7.12 FBA object decoding 281
7.12.1 Frame based face object decoding 281
7.12.2 DCT based face object decoding 282
7.12.3 Decoding of the viseme parameter fap 1 284
7.12.4 Decoding of the viseme parameter fap 2 284
7.12.5 Fapmasking 285
7.12.6 Frame Based Body Decoding 285
7.12.7 DCT based body object decoding 286
7.13 3D Mesh Object Decoding 287
7.13.1 Start codes and bit stuffing 288
7.13.2 The Topological Surgery decoding process 288
7.13.3 The Forest Split decoding process 291
7.13.4 Header decoder 292
7.13.5 partition type 293
7.13.6 Vertex Graph Decoder 294
7.13.7 Triangle Tree Decoder 298
7.13.8 Triangle Data Decoder 299
7.13.9 Forest Split decoder 303
7.13.10 Arithmetic decoder 309
7.14 NEWPRED mode decoding 314
7.14.1 Decoder Definition 314
7.14.2 Upstream message 314
7.15 Output of the decoding process 314
7.15.1 Video data 315
7.15.2 2D Mesh data 315
7.15.3 Face animation parameter data 315

8 Visual-Systems Composition Issues 315
8.1 Temporal Scalability Composition 315
8.2 Sprite Composition 316
8.3 Mesh Object Composition 317
8.4 Spatial Scalability composition 318

9 Profiles and Levels 318
9.1 Visual Object Types 318
9.2 Visual Profiles 321
9.3 Visual Profiles@Levels 322
9.3.1 Natural Visual 322
9.3.2 Synthetic Visual 322
9.3.3 Synthetic/Natural Hybrid Visual 324

Annex A (normative) Coding transforms 326
A.1 Discrete cosine transform for video texture 326
A.2 Discrete wavelet transform for still texture 327
A.2.1 Adding the mean 327
A.2.2 Wavelet filter 327
A.2.3 Symmetric extension 328
A.2.4 Decomposition level 329
A.2.5 Shape adaptive wavelet filtering and symmetric extension 329
A.3 Shape-Adaptive DCT (SA-DCT) 330
A.3.1 Definition of Forward SA-DCT 330
A.3.2 Definition of Inverse SA-DCT 332
A.4 SA-DCT with DC Separation and ΔDC Correction (ΔDC-SA-DCT) 333
A.4.1 Definition of Forward ΔDC-SA-DCT 334
A.4.2 Definition of Inverse ΔDC-SA-DCT 334

Annex B (normative) Variable length codes and arithmetic decoding 336
B.1 Variable length codes 336
B.1.1 Macroblock type 336

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAWADE
ANSI Store order #1537445, Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2057
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

B.1.2 Macroblock pattern 338
B.1.3 Motion vector 340
B.1.4 DCT coefficients 342
B.1.5 Shape Coding 352
B.1.6 Sprite Coding 357
B.1.7 DCT based facial object decoding 358
B.1.8 Shape decoding for still texture object 367
B.2 Arithmetic Decoding 368
B.2.1 Arithmetic decoding for still texture object 368
B.2.2 Arithmetic decoding for shape decoding 371
B.2.3 FBA Object Decoding 374

Annex C (normative) Face and body object decoding tables and definitions 376

Annex D (normative) Video buffering verifier 409
D.1 Introduction 409
D.2 Video Rate Buffer Model Definition 409
D.3 Comparison between ISO/IEC 14496-2 VBV and the ISO/IEC 13818-2 VBV (informative) 412
D.4 Video Complexity Model Definition 413
D.5 Video Reference Memory Model Definition 415
D.6 Interaction between VBV, VCV and VMV (informative) 416
D.7 Video Presentation Model Definition (informative) 416

Annex E (informative) Features supported by the algorithm 418
E.1 Error resilience 418
E.1.1 Resynchronization 418
E.1.2 Data Partitioning 419
E.1.3 Reversible VLC 419
E.1.4 Decoder Operation 420
E.1.5 Adaptive Intra Refresh (AIR) Method 423
E.1.6 NEWPRED 425
E.2 Complexity Estimation 427
E.3 Resynchronization in Case of Unknown Video Header Format 427

Annex F (informative) Preprocessing and postprocessing 428
F.1 VOP Generation Tools: Automatic and Semi-automatic Segmentations 428
F.1.1 Automatic Segmentation 428
F.1.2 Semi-automatic Segmentation 438
F.1.3 References 446
F.2 Bounding Rectangle of VOP Formation 447
F.3 Postprocessing for Coding Noise Reduction 448
F.3.1 Deblocking filter 448
F.3.2 Deringing filter 450
F.3.3 Further issues 452
F.4 Chrominance Decimation and Interpolation Filtering for Interlaced Object Coding 452

Annex G (normative) Profile and level indication and restrictions 454

Annex H (informative) Patent statements 457
H.1 Patent statements for ISO/IEC 14496 Version 1 457
H.2 Patent statements for the extensions provided in ISO/IEC 14496 Version 2 458

Annex I (informative) Encoder Complexity Reduction Based on Intelligent Pre-Quantisation 460
I.1 Introduction 460
I.2 Feature Selection and Pre-quantisation 460
I.3 Model Verification and Threshold Setting 462
I.3.1 H.263 Quantiser 462
I.3.2 MPEG-4 Quantiser 462

Annex J (normative) View dependent object scalability 464
J.1 Introduction 464
J.2 Decoding Process of a View-Dependent Object 464
J.2.1 General Decoding Scheme 464
J.2.2 Computation of the View-Dependent Scalability parameters 466

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOLE V. ALAMURU
ANSI Store order #537445, Downloaded: 7/18/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2058
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

J.2.3 VD mask computation 468
J.2.4 Differential mask computation 469
J.2.5 DCT coefficients decoding 469
J.2.6 Texture update 469
J.2.7 IDCT 470

Annex K (normative) Decoder Configuration Information 471
K.1 Introduction 471
K.2 Description of the set up of a visual decoder (informative) 471
K.2.1 Processing of decoder configuration information 472
K.3 Specification of decoder configuration information 473
K.3.1 VideoObject 473
K.3.2 StillTextureObject 473
K.3.3 MeshObject 474
K.3.4 FaceObject 474
K.3.5 3DMeshObject 474

Annex L (informative) Rate control 475
L.1 Frame Rate Control 475
L.1.1 Introduction 475
L.1.2 Description 475
L.1.3 Summary 479
L.2 Multiple Video Object Rate Control 479
L.2.1 Initialization 479
L.2.2 Quantisation Level Calculation for I-frame and first P-frame 479
L.2.3 Update Rate-Distortion Model 482
L.2.4 Post-Frameskip Control 482
L.3 Macroblock Rate Control 484
L.3.1 Rate-Distortion Model 484
L.3.2 Target Number of Bits for Each Macroblock 485
L.3.3 Macroblock Rate Control 485

Annex M (informative) Binary shape coding 487
M.1 Introduction 487
M.2 Context-Based Arithmetic Shape Coding 487
M.2.1 Intra Mode 488
M.2.2 Inter Mode 488
M.3 Texture Coding of Boundary Blocks 489
M.4 Encoder Architecture 489
M.5 Encoding Guidelines 490
M.5.1 Lossy Shape Coding 491
M.5.2 Coding Mode Selection 491
M.6 Conclusions 491
M.7 References 492

Annex N (normative) Visual profiles@levels 493

Annex O (informative) 3D Mesh Coding 497
O.1 Introduction 497
O.2 Topological Surgery Representation 497
O.2.1 Simple Polygon Representation 498
O.2.2 Vertex Graph representation 499
O.3 Encoding guidelines for 3D Mesh Coding 500
O.3.1 Topological Surgery Encoding 500
O.3.2 Support for non-manifolds and Non-orientable manifolds 501
O.3.3 Support for Error Resilience 503
O.4 Encoder considerations for efficient compression of Vertex Properties 507
O.5 Progressive Forest Split Representation 508
O.5.1 Encoding the Forest 508
O.5.2 Support for meshes with polygonal faces 509
O.5.3 Method for generating of a PFS Representation of a Triangular 3D Mesh 509
O.5.4 Topological Tests 510

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAWADE
ANSI Store order #1537445, Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2059
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

O.5.5 Geometric Tests 512
O.6 Compexity estimation for Computational Graceful Degradation 512
O.7 QoS for SNHC through upstream 514
O.8 References 516
Bibliography 517

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PATRON, Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2060
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14496-2:1999), which has been technically revised. It incorporates Cor.1:2000, Cor.2:2001 and Amd.1:2000.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized software for MPEG-4 visual tools*

— *Part 8: Carriage of MPEG-4 contents over IP networks*

Annexes A to D, G, J, K and N form a normative part of this part of ISO/IEC 14496. Annexes E, F, H, I, L, M and O are for information only.

© ISO/IEC 2009 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC Y. ADAMANI
ANSI Store order: WS274457 Downloaded: 7/16/2009 2:06:21 PM EST
Single user license only. Copying and networking prohibited.

Introduction

Purpose

This part of ISO/IEC 14496 was developed in response to the growing need for a coding method that can facilitate access to visual objects in natural and synthetic moving pictures and associated natural or synthetic sound for various applications such as digital storage media, internet, various forms of wired or wireless communication etc. The use of ISO/IEC 14496 means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcast channels.

Application

The applications of ISO/IEC 14496 cover, but are not limited to, such areas as listed below:

IMM Internet Multimedia

IVG Interactive Video Games

IPC Interpersonal Communications (videoconferencing, videophone, etc.)

ISM Interactive Storage Media (optical disks, etc.)

MMM Multimedia Mailing

NDB Networked Database Services (via ATM, etc.)

RES Remote Emergency Systems

RVS Remote Video Surveillance

WMM Wireless Multimedia

Multimedia

Profiles and levels

ISO/IEC 14496 is intended to be generic in the sense that it serves a wide range of applications, bitrates, resolutions, qualities and services. Furthermore, it allows a number of modes of coding of both natural and synthetic video in a manner facilitating access to individual objects in images or video, referred to as content based access. Applications should cover, among other things, digital storage media, content based image and video databases, internet video, interpersonal video communications, wireless video etc. In the course of creating ISO/IEC 14496, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax. Hence ISO/IEC 14496 will facilitate the bitstream interchange among different applications.

This part of ISO/IEC 14496 includes one or more complete decoding algorithms as well as a set of decoding tools. Moreover, the various tools of this part of ISO/IEC 14496 as well as that derived from ISO/IEC 13818-2:1996 can be combined to form other decoding algorithms. Considering the practicality of implementing the full syntax of this part of ISO/IEC 14496, however, a limited number of subsets of the syntax are also stipulated by means of "profile" and "level".

A "profile" is a defined subset of the entire bitstream syntax that is defined by this part of ISO/IEC 14496. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by parameters in the bitstream.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... Downloaded: 7/16/2007 1:09:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2062
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

In order to deal with this problem "levels" are defined within each profile. A level is a defined set of constraints imposed on parameters in the bitstream. These constraints may be simple limits on numbers. Alternatively they may take the form of constraints on arithmetic combinations of the parameters.

Object based coding syntax

Video object

A *video object* in a scene is an entity that a user is allowed to access (seek, browse) and manipulate (cut and paste). The instances of video objects at a given time are called *video object planes* (VOPs). The encoding process generates a coded representation of a VOP as well as composition information necessary for display. Further, at the decoder, a user may interact with and modify the composition process as needed.

The full syntax allows coding of rectangular as well as arbitrarily shaped video objects in a scene. Furthermore, the syntax supports both nonscalable coding and scalable coding. Thus it becomes possible to handle normal scalabilities as well as object based scalabilities. The scalability syntax enables the reconstruction of useful video from pieces of a total bitstream. This is achieved by structuring the total bitstream in two or more layers, starting from a standalone base layer and adding a number of enhancement layers. The base layer can be coded using a non-scalable syntax, or in the case of picture based coding, even using a syntax of a different video coding standard.

To ensure the ability to access individual objects, it is necessary to achieve a coded representation of its shape. A natural video object consists of a sequence of 2D representations (at different points in time) referred to here as VOPs. For efficient coding of VOPs, both temporal redundancies as well as spatial redundancies are exploited. Thus a coded representation of a VOP includes representation of its shape, its motion and its texture.

FBA object

The FBA object is a collection of nodes in a scene graph which are animated by the FBA (Face and Body Animation) object bitstream. The FBA object is controled by two separate bitstreams. The first bitstream, called BIFS, contains instances of Body Definition Parameters (BDPs) in addition to Facial Definition Parameters (FDPs), and the second bitstream, FBA bitstream, contains Body Animation Parameters (BAPs) together with Facial Animation Parameters (FAPs).

A 3D (or 2D) *face object* is a representation of the human face that is structured for portraying the visual manifestations of speech and facial expressions adequate to achieve visual speech intelligibility and the recognition of the mood of the speaker. A face object is animated by a stream of *face animation parameters (FAP)* encoded for low-bandwidth transmission in broadcast (one-to-many) or dedicated interactive (point-to-point) communications. The FAPs manipulate key feature control points in a mesh model of the face to produce animated visemes for the mouth (lips, tongue, teeth), as well as animation of the head and facial features like the eyes. FAPs are quantized with careful consideration for the limited movements of facial features, and then prediction errors are calculated and coded arithmetically. The remote manipulation of a face model in a terminal with FAPs can accomplish lifelike visual scenes of the speaker in real-time without sending pictorial or video details of face imagery every frame.

A simple streaming connection can be made to a decoding terminal that animates a default face model. A more complex session can initialize a custom face in a more capable terminal by downloading *face definition parameters (FDP)* from the encoder. Thus specific background images, facial textures, and head geometry can be portrayed. The composition of specific backgrounds, face 2D/3D meshes, texture attribution of the mesh, etc. is described in ISO/IEC 14496-1:1999. The FAP stream for a given user can be generated at the user's terminal from video/audio, or from text-to-speech. FAPs can be encoded at bitrates up to 2-3kbit/s at necessary speech rates. Optional temporal DCT coding provides further compression efficiency in exchange for delay. Using the facilities of ISO/IEC 14496-1:1999, a composition of the animated face model and synchronized, coded speech audio (low-bitrate speech coder or text-to-speech) can provide an integrated low-bandwidth audio/visual speaker for broadcast applications or interactive conversation.

Limited scalability is supported. Face animation achieves its efficiency by employing very concise motion animation controls in the channel, while relying on a suitably equipped terminal for rendering of moving 2D/3D faces with non-normative models held in local memory. Models stored and updated for rendering in the terminal can be simple or complex. To support speech intelligibility, the normative specification of FAPs intends for their selective or complete use as signaled by the encoder. A masking scheme provides for selective transmission of FAPs according to what

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #1537445, Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2063
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

parts of the face are naturally active from moment to moment. A further control in the FAP stream allows face animation to be suspended while leaving face features in the terminal in a defined quiescent state for higher overall efficiency during multi-point connections.

A body model is a representation of a virtual human or human-like character that allows portraying body movements acequate to achieve nonverbal communication and general actions. A body model is animated by a stream of *body animation parameters* (BAP) encoded for low-bitrate transmission in broadcast and dedicated interactive communications. The BAPs manipulate independent degrees of freedom in the skeleton model of the body to produce animation of the body parts. The BAPs are quantized considering the joint limitations, and prediction errors are calculated and coded arithmetically. Similar to the face, the remote manipulation of a body model in a terminal with BAPs can accomplish lifelike visual scenes of the body in real-time without sending pictorial and video details of the body every frame.

The BAPs, if correctly interpreted, will produce reasonably similar high level results in terms of body posture and animation on different body models, also without the need to initialize or calibrate the model. The BDP set defines the set of parameters to transform the default body to a customized body optionally with its body surface, body dimensions, and texture.

The *body definition parameters (BDP)* allow the encoder to replace the local model of a more capable terminal. BDP parameters include body geometry, calibration of body parts, degrees of freedom, and optionally deformation information.

The FBA Animation specification is defined in ISO/IEC 14496-1:1999/Amd.1:2000 and this part of ISO/IEC 14496. This clause is intended to facilitate finding various parts of specification. As a rule of thumb, FAP and BAP specification is found in the part 2, and FDP and BDP specification in the part 1. However, this is not a strict rule. For an overview of FAPs/BAPs and their interpretation, read subclauses "6.1.5.2 Facial animation parameter set", "6.1.5.3 Facial animation parameter units", "6.1.5.4 Description of a neutral face" as well as the Table C-1. The viseme parameter is documented in subclause "7.12.3 Decoding of the viseme parameter fap 1" and the Table C-5 in annex C. The expression parameter is documented in subclause "7.12.4 Decoding of the expression parameter fap 2" and the Table C-3. FBA bitstream syntax is found in subclauses "6.2.10 FBA Object", semantics in "6.3.10 FBA Object", and subclause "7.12 FBA object decoding" explains in more detail the FAP/BAP decoding process. FAP/BAP masking and interpolation is explained in subclauses "6.3.11.1 FBA Object Plane", "7.12.1.1 Decoding of FBA", "7.12.5 FBA masking" . The FIT interpolation scheme is documented in subclause "7.2.5.3.2.4 FIT" of ISO/IEC 14496-1:1999. The FDPs and BDPs and their interpretation are documented in subclause "7.2.5.3.2.6 FDP" of ISO/IEC 14496-1:1999. In particular, the FDP feature points are documented in Figure C-1. Details on body models are documented in annex C.

Mesh object

A 2D *mesh object* is a representation of a 2D deformable geometric shape, with which synthetic video objects may be created during a composition process at the decoder, by spatially piece-wise warping of existing video object planes or still texture objects. The instances of mesh objects at a given time are called *mesh object planes* (mops). The geometry of mesh object planes is coded lossless. Temporally and spatially predictive techniques and variable length coding are used to compress 2D mesh geometry. The coded representation of a 2D mesh object includes representation of its geometry and motion.

3D Mesh Object

The 3D Mesh Object is a 3D polygonal model that can be represented as an IndexedFaceSet or Hierarchical 3D Mesh node in BIFS. It is defined by the position of its vertices (geometry), by the association between each face and its sustaining vertices (connectivity), and optionally by colours, normals, and texture coordinates (properties). Properties do not affect the 3D geometry, but influence the way the model is shaded. 3D mesh coding (3DMC) addresses the efficient coding of 3D mesh object. It comprises a basic method and several options. The basic 3DMC method operates on manifold model and features incremental representation of single resolution 3D model. The model may be triangular or polygonal – the latter are triangulated for coding purposes and are fully recovered in the decoder. Options include: (a) support for computational graceful degradation control; (b) support for non-manifold models; (c) support for error resilience; and (d) quality scalability via hierarchical transmission of levels of detail with implicit support for smooth transition between consecutive levels. The compression of application-specific geometry streams (Face Animation Parameters) and generalized animation parameters (BIFS Anim) are currently addressed elsewhere in this part of ISO/IEC 14496.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOLE LY, ALAMANE
ANSI Store order #0378481 Downloaded: 7/18/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2064
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

In 3DMC, the compression of the connectivity of the 3D mesh (e.g. how edges, aces, and vertices relate) is lossless, whereas the compression of the other attributes (such as vertex coordinates, normals, colours, and texture coordinates) may be lossy.

Single Resolution Mode

The incremental representation of a single resolution 3D model is based on the Topological Surgery scheme. For manifold triangular 3D meshes, the Topological Surgery representation decomposes the connectivity of each *connected component* into a *simple polygon* and a *vertex graph*. All the triangular faces of the 3D mesh are connected is the simple polygon forming a *triangle tree*, which is a spanning tree in the dual graph of the 3D mesh. Figure V2 - 1 shows an example of a triangular 3D mesh, its dual graph, and a triangle tree. The vertex graph identifies which pairs of boundary edges of the simple polygon are associated with each other to reconstruct the connectivity of the 3D mesh. The triangle tree does not fully describe the triangulation of the simple polygon. The missing information is recorded as a *marching edge*.

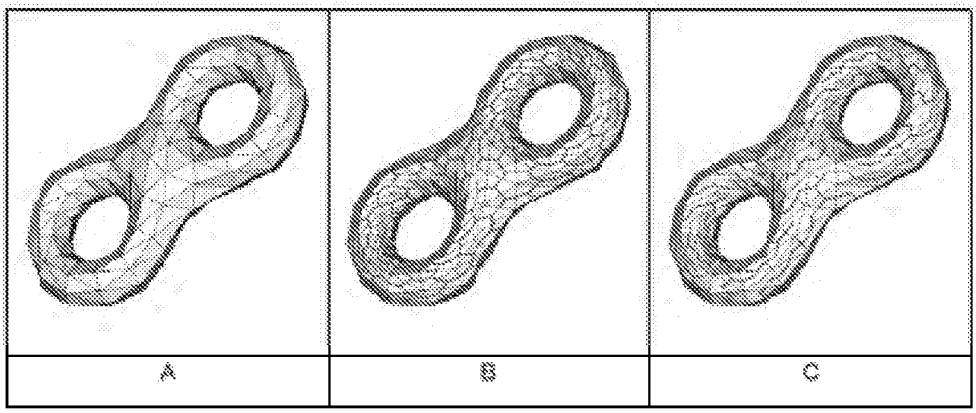


Figure V2 - 1 -- A triangular 3D mesh (A), its dual graph (B), and a triangle tree (C)

For manifold 3D meshes, the connectivity is represented in a similar fashion. The polygonal faces of the 3D mesh are connected in a simple polygon forming a *face tree*. The faces are triangulated, and which edges of the resulting triangular 3D mesh are edges of the original 3D mesh is recorded as a sequence of *polygon_edge* bits. The face tree is also a spanning tree in the dual graph of the 3D mesh, and the vertex graph is always composed of edges of the original 3D mesh.

The vertex coordinates and optional properties of the 3D mesh (normals, colours, and texture coordinates) are quantized, predicted as a function of decoded ancestors with respect to the order of traversal, and the errors are entropy encoded.

Incremental Representation

When a 3D mesh is downloaded over networks with limited bandwidth (e.g. PSTN), it may be desired to begin decoding and rendering the 3D mesh before it has all been received. Moreover, content providers may wish to control such incremental representation to present the most important data first. The basic 3DMC method supports this by interleaving the data such that each triangle may be reconstructed as it is received. Incremental representation is also facilitated by the options of hierarchical transmission for quality scalability and partitioning for error resilience.

Hierarchical Mode

An example of a 3D mesh represented in hierarchical mode is illustrated in Figure V2 - 2. The hierarchical mode allows the decoder to show progressively better approximations of the model as data are received. The hierarchical 3D mesh decomposition can also be organized in the decoder as layered detail, and view-dependent expansion of this detail can be subsequently accomplished during a viewer's interaction with the 3D model. Downloadable

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #US1745, Downloaded: 7/18/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2065
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

scalability of 3D model allows decoders with widely varied rendering performance to use the content, without necessarily making repeated requests to the encoder for specific versions of the content and without the latencies of updating view-dependent model from the encoder. This quality scalability of 3D meshes is complementary to scaleable still texture and supports bitstream scalability and downloading of quality hierarchies of hybrid imagery to the decoder.

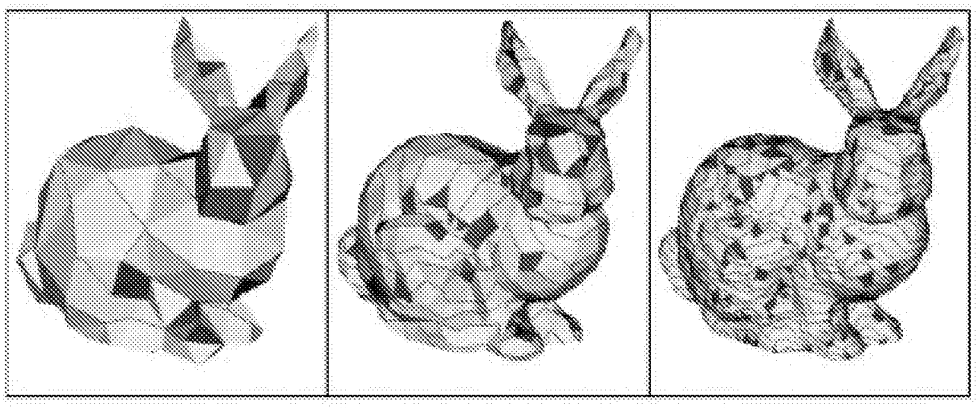


Figure V2 - 2 – A hierarchy of levels of detail

The hierarchical representation of a 3D model is based on the Progressive Forest Split scheme. In the Progressive Forest Split representation, a manifold 3D mesh is represented as a *base 3D mesh* followed by a sequence of *forest split* operations. The base 3D mesh is encoded as a single resolution 3D mesh using the Topological Surgery scheme. Each forest split operation is composed of a *forest* in the graph of the current 3D mesh, and a *sequence of simple polygons*. Figure V2 - 3 illustrates the method. To prevent visual artifacts which may occur while switching from a level of detail to the next one, the data structure supports smooth transition between consecutive levels of detail in the form of linear interpolation of vertex positions.

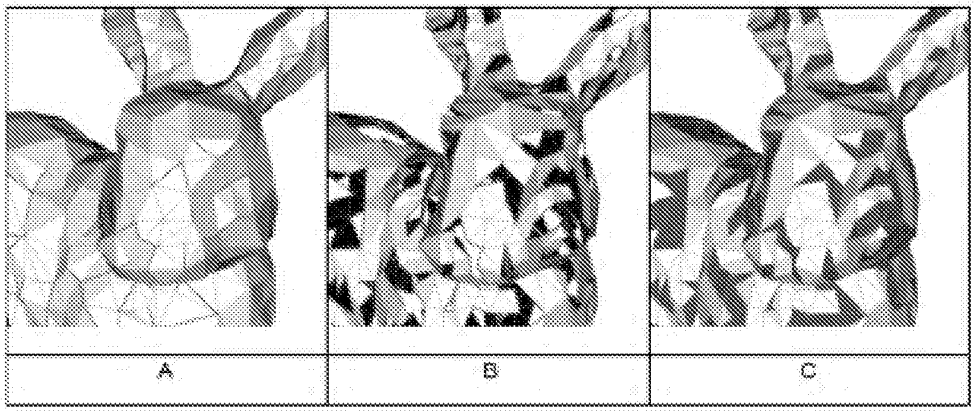


Figure V2 - 3 – The Forest Split operation. A 3D mesh with a forest of edges marked (A). The tree loops resulting from cutting the 3D mesh through the forest edges (B). Each tree loop is filled by a simple polygon composed of polygonal faces (C).

Error Resilience for 3D mesh object

If the 3D mesh is partitioned into independent parts, it may be possible to perform more efficient data transmission in an error-prone environment, e.g., an IP network or datacasting service in a broadcast TV network. It must be possible to resynchronize after a channel error, and continue data transmission and rendering from that point instead of starting over from scratch. Even with the presence of channel errors, the decoder can start decoding and rendering from the next partition that is received intact from the channel.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG X. PATTERSON
ANSI Store order #0376481 Downloaded: 7/16/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2066
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

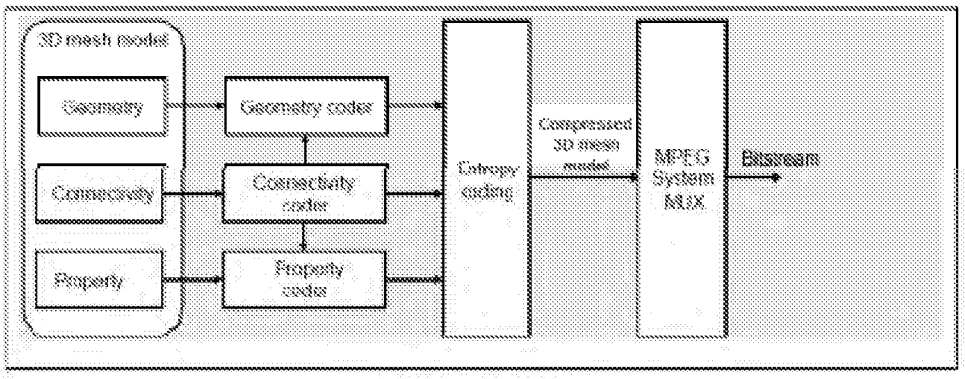
Flexible partitioning methods can be used to organize the data, such that it fits the underlying network packet structure more closely, and overhead is reduced to the minimum. To allow flexible partitioning, several connected components may be merged into one partition, where as a large connected component may be divided into several independent partitions. Merging and dividing of connected components using different partition types can be done at any point in the 3D mesh object.

Stitching for Non-Manifold and Non-Orientable Meshes

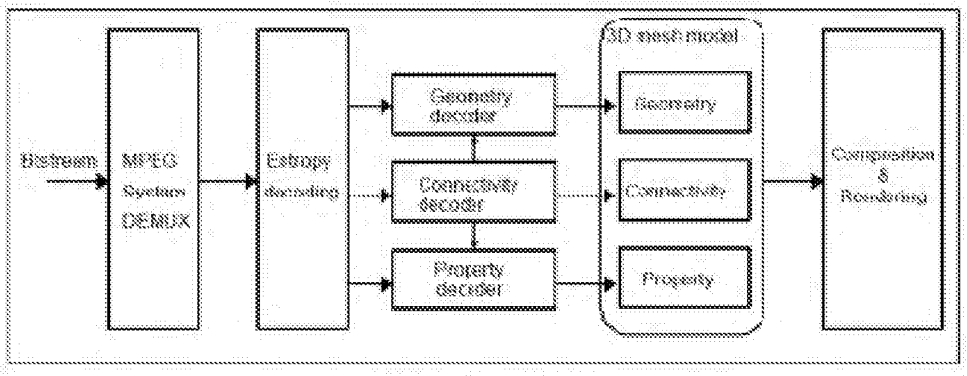
The connectivity of a non-manifold and non-orientable 3D mesh is represented as a manifold 3D mesh and a sequence of *stitches*. Each stitch describes how to identify one or more pairs of vertices along two linear paths of edges, and each one of these paths is contained in one of the vertex graphs that span the connected components of the manifold 3D mesh.

Encoder and Decoder Block Diagrams

High level block diagrams of a general 3D polygonal model encoder and decoder are shown in Figure V2 - 4. They consist of a 3D mesh connectivity (de)coder, geometry (de)coder, property (de)coder, and entropy (de)coding blocks. Connectivity, vertex position, and property information are extracted from 3D mesh model described in VRML or MPEG-4 BIFS format. The connectivity (de)coder is used for an efficient representation of the association between each face and its sustaining vertices. The geometry (de)coder is used for a lossy or lossless compression of vertex coordinates. The property (de)coder is used for a lossy or lossless compression of colour, normal, and texture coordinate data.



(a) 3D mesh encoder



(b) 3D mesh decoder

Figure V2 - 4 – General block diagram of the 3D mesh compres

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUBLAY ADAMSON
ANSI Store order #X0744G Downloaded 7/18/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2067
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Overview of the object based nonscalable syntax

The coded representation defined in the non-scalable syntax achieves a high compression ratio while preserving good image quality. Further, when access to individual objects is desired, the shape of objects also needs to be coded, and depending on the bandwidth available, the shape information can be coded in a lossy or lossless fashion.

The compression algorithm employed for texture data is not lossless as the exact sample values are not preserved during coding. Obtaining good image quality at the bitrates of interest demands very high compression, which is not achievable with intra coding alone. The need for random access, however, is best satisfied with pure intra coding. The choice of the techniques is based on the need to balance a high image quality and compression ratio with the requirement to make random access to the coded bitstream.

A number of techniques are used to achieve high compression. The algorithm first uses block-based motion compensation to reduce the temporal redundancy. Motion compensation is used both for causal prediction of the current VOP from a previous VOP, and for non-causal, interpolative prediction from past and future VOPs. Motion vectors are defined for each 16-sample by 16-line region of a VOP or 8-sample by 8-line region of a VOP as required. The prediction error, is further compressed using the discrete cosine transform (DCT) to remove spatial correlation before it is quantised in an irreversible process that discards the less important information. Finally, the shape information, motion vectors and the quantised DCT information, are encoded using variable length codes.

Temporal processing

Because of the conflicting requirements of random access to and highly efficient compression, three main VOP types are defined. Intra coded VOPs (I-VOPs) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only moderate compression. Predictive coded VOPs (P-VOPs) are coded more efficiently using motion compensated prediction from a past intra or predictive coded VOPs and are generally used as a reference for further prediction. Bidirectionally-predictive coded VOPs (B-VOPs) provide the highest degree of compression but require both past and future reference VOPs for motion compensation. Bidirectionally-predictive coded VOPs are never used as references for prediction (except in the case that the resulting VOP is used as a reference for scalable enhancement layer). The organisation of the three VOP types in a sequence is very flexible. The choice is left to the encoder and will depend on the requirements of the application.

Coding of Shapes

In natural video scenes, VOPs are generated by segmentation of the scene according to some semantic meaning. For such scenes, the shape information is thus binary (binary shape). Shape information is also referred to as alpha plane. The binary alpha plane is coded on a macroblock basis by a coder which uses the context information, motion compensation and arithmetic coding.

For coding of shape of a VOP, a bounding rectangle is first created and is extended to multiples of 16×16 blocks with extended alpha samples set to zero. Shape coding is then initiated on a 16×16 block basis; these blocks are also referred to as binary alpha blocks.

Motion representation - macroblocks

The choice of 16×16 blocks (referred to as macroblocks) for the motion-compensation unit is a result of the trade-off between the coding gain provided by using motion information and the overhead needed to represent it. Each macroblock can further be subdivided to 8×8 blocks for motion estimation and compensation depending on the overhead that can be afforded. In order to encode the highly active scene with higher VOP rate, a Reduced Resolution VOP tool is provided. When this tool is used , the size of the macroblock used for motion compensation decoding is 32×32 pixels and the size of block is 16×16 pixels.

Depending on the type of the macroblock, motion vector information and other side information is encoded with the compressed prediction error in each macroblock. The motion vectors are differenced with respect to a prediction value and coded using variable length codes. The maximum length of the motion vectors allowed is decided at the encoder. It is the responsibility of the encoder to calculate appropriate motion vectors. The specification does not specify how this should be done.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOLA V. ALAVANDI
ANSI Store order #0378481 Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2068
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Spatial redundancy reduction

Both source VOPs and prediction errors VOPs have significant spatial redundancy. The part of ISO/IEC 14496 uses a block-based DCT method with optional visually weighted quantisation, and run-length coding. After motion compensated prediction or interpolation, the resulting prediction error is split into 8×8 blocks. These are transformed into the DCT domain where they can be weighted before being quantised. After quantisation many of the DCT coefficients are zero in value and so two-dimensional run-length and variable length coding is used to encode the remaining DCT coefficients efficiently.

Chrominance formats

This part of ISO/IEC 14496 currently supports the 4:2:0 chrominance format.

Pixel depth

This part of ISO/IEC 14496 supports pixel depths between 4 and 12 bits in luminance and chrominance planes.

Generalized scalability

The scalability tools in this part of ISO/IEC 14496 are designed to support applications beyond that supported by single layer video. The major applications of scalability include internet video, wireless video, multi-quality video services, video database browsing etc. In some of these applications, either normal scalabilities on picture basis such as those in ISO/IEC 13818-2:1996 may be employed or object based scalabilities may be necessary; both categories of scalability are enabled by this part of ISO/IEC 14496.

Although a simple solution to scalable video is the simulcast technique that is based on transmission/storage of multiple independently coded reproductions of video, a more efficient alternative is scalable video coding, in which the bandwidth allocated to a given reproduction of video can be partially re-utilised in coding of the next reproduction of video. In scalable video coding, it is assumed that given a coded bitstream, decoders of various complexities can decode and display appropriate reproductions of coded video. A scalable video encoder is likely to have increased complexity when compared to a single layer encoder. However, this part of ISO/IEC 14496 provides several different forms of scalabilities that address non-overlapping applications with corresponding complexities.

The basic scalability tools offered are temporal scalability and spatial scalability. Moreover, combinations of these basic scalability tools are also supported and are referred to as hybrid scalability. In the case of basic scalability, two layers of video referred to as the lower layer and the enhancement layer are allowed, whereas in hybrid scalability up to four layers are supported.

Object based Temporal scalability

Temporal scalability is a tool intended for use in a range of diverse video applications from video databases, internet video, wireless video and multiview/stereoscopic coding of video. Furthermore, it may also provide a migration path from current lower temporal resolution video systems to higher temporal resolution systems of the future.

Temporal scalability involves partitioning of VOPs into layers, where the lower layer is coded by itself to provide the basic temporal rate and the enhancement layer is coded with temporal prediction with respect to the lower layer. These layers when decoded and temporally multiplexed yield full temporal resolution. The lower temporal resolution systems may only decode the lower layer to provide basic temporal resolution whereas enhanced systems of the future may support both layers. Furthermore, temporal scalability has use in bandwidth constrained networked applications where adaptation to frequent changes in allowed throughput are necessary. An additional advantage of temporal scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over a channel with better error performance, whereas the less critical enhancement layer can be sent over a channel with poor error performance. Object based temporal scalability can also be employed to allow graceful control of picture quality by controlling the temporal rate of each video object under the constraint of a given bit-budget.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #1537445, Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2069
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Object Spatia scalability

Spatial scalability is a tool intended for use in video applications involving mult quality video services, video database browsing, internet video and wireless video, i.e. video systems with the primary common feature that a minimum of two layers of spatial resolution are necessary. Spatial scalability involves generating two spatial resolution video layers from a single video source such that the lower layer is coded by itself to provide the basic spatial resolution and the enhancement layer employs the spatially interpolated lower layer and carries the full spatial resolution of the input video source.

Object based spatial scalability is composed of texture spatial scalability and shape spatial scalability. They can be used indepencently or together by its application. Binary shape spatial scalability is used for the applications that have 'binary shape only' mode as well as the applications of general object based spatial scalability.

An additional advantage of spatial scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over a channel with better error performance, whereas the less critical enhancement layer data can be sent over a channel with poor error performance. Further, it can also allow interoperability between various standards.

Hybrid scalability

There are a number of applications where neither the temporal scalability nor the spatial scalability may offer the necessary flexibility and control. This may necessitate use of temporal and spatial scalability simultaneously and is referred to as the hybrid scalability. Among the applications of hybrid scalability are wireless video, internet video, multiviewpoint/stereoscopic coding etc.

Error Resilience

This part of ISO/IEC 14496 provides error robustness and resilience to allow accessing of image or video information over a wide range of storage and transmission media. The error resilience tools developed for this part of ISO/IEC 14496 can be divided into three major categories. These categories include synchronization, data recovery, and error concealment. It should be noted that these categories are not unique to this part of ISO/IEC 14496, and have been used elsewhere in general research in this area. It is, however, the tools contained in these categories that are of interest, and where this part of ISO/IEC 14496 makes its contribution to the problem of error resilience.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS K. / ARLINGTON
ANSI Store order #: [illegible] Downloaded: 7/16/2007 1:08:21 PM ET
Single user license only. Copying and networking prohibited.
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2070
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Information technology — Coding of audio-visual objects —

Part 2:
Visual

1 Scope

This part of ISO/IEC 14496 specifies the coded representation of picture information in the form of natural or synthetic visual objects like video sequences of rectangular or arbitrarily shaped pictures, moving 2D meshes, animated 3D face and body models and texture for synthetic objects. The coded representation allows for content based access for digital storage media, digital video communication and other applications. ISO/IEC 14496 specifies also the decoding process of the aforementioned coded representation. The representation supports constant bitrate transmission, variable bitrate transmission, robust transmission, content based random access (including normal random access), object based scalable decoding (including normal scalable decoding), object based bitstream editing, as well as special functions such as fast forward playback, fast reverse playback, slow motion, pause and still pictures. Synthetic objects and coding of special 2D/3D meshes, texture, and animation parameters are provided for use with downloadable models to exploit mixed media and the bandwidth improvement associated with remote manipulation of such models. ISO/IEC 14496 is intended to allow some level of interoperability with ISO/IEC 11172-2:1993, ISO/IEC 13818-2:1996 and ITU-T Recommendation H.263.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 14496 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ITU-T Recommendation T.81 (1992)|ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines.*

ISO/IEC 11172-1:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 1: Systems.*

ISO/IEC 11172-2:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 2: Video.*

ISO/IEC 11172-3:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio.*

ITU-T Recommendation H.222.0(2000)|ISO/IEC 13818-1:2000, *Information technology — Generic coding of moving pictures and associated audio information: Systems.*

ITU-T Recommendation H.262(2000)|ISO/IEC 13818-2:2000, *Information technology — Generic coding of moving pictures and associated audio information: Video.*

ISO/IEC 13818-3:1998, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio.*

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUNAY, ADAMARIS
ANSI Store order #: 01744 / Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2071
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ITU-R Recommendation BT.601-5, *Encoding parameters of digital television for studios.*

IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990.

IEC 60461:2001, *Time and control code for video tape recorder.*

ITU-T Recommendation H.261:1993, *Video codec for audiovisual services at p x 64 kbits.*

ITU-T Recommendation H.263:2000, *Enhanced reference picture selection mode.*

3 Terms and definitions

For the purposes of this part of ISO/IEC 14496, the following terms and definitions apply.

3.1 **3D mesh**: A polygonal mesh. It is defined by the positon of its vertices (geometry), by the association between each face and its sustaining vertices (connectivity), and optionaly by colors, normals, and texture coordinates (properties). Properties do not affect the 3D geometry, but influence the way the model is shaded.

3.2 **3D mesh connectivity**: The association between each face of a 3D mesh and its sustaining vertices.

3.3 **3D mesh geometry**: The three dimensional coordinates of the vertices of a 3D mesh.

3.4 **3D mesh properties**: The normals, colors, and texture coordinates of a 3D mesh.

3.5 **AC coefficient**: Any DCT coefficient for which the frequency in one or both dimensions is non-zero.

3.6 **B-VOP; bidirectionally predictive-coded video object plane (VOP)**: A VOP that is coded using motion compensated prediction from past and/or future reference VOPs.

3.7 **backward motion vector**: A motion vector that is used for motion compensation from a reference VOP at a later time in display order.

3.8 **backward prediction**: Prediction from the future reference VOP.

3.9 **base layer**: An independently decodable layer of a scalable hierarchy.

3.10 **binary alpha block**: A block of size 16×16 pels, colocated with macroblock, representing shape information of the binary alpha map; it is also referred to as a bab.

3.11 **binary alpha map**: A 2D binary mask used to represent the shape of a video object such that the pixels that are opaque are considered as part of the object where as pixels that are transparent are not considered to be part of the object.

3.12 **bitstream; stream**: An ordered series of bits that forms the coded representation of the data.

3.13 **bitrate**: The rate at which the coded bitstream is delivered from the storage medium or network to the input of a decoder.

3.14 **block**: An 8-row by 8-column (or 16-row by 16-column in motion compensation decoding of Reduced Resolution VOP) matrix of samples, or 64 DCT coefficients (source, quantised or dequantised).

3.15 **body animation parameters, BAP**: Coded streaming animation parameters that manipulate the body joints, and that govern deformations of the body surface with the use of BAT.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. KAMPS
ANSI Store order #: 01214[illegible] Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2072
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.16 **body definition parameters, BDP**: Downloadable data to customize a baseline body model in the decoder to a particular body, or to download a body model along with the information about how to animate it. The BDPs are normally transmitted once per session, followed by a stream of compressed BAPs. BDPs may include body geometry, body skeleton, and body deformation tables.

3.17 **body deformation table, BodyDefTable**: A downloadable function mapping from incoming BAPs to body surface geometry that provides a combination of BAPs for controlling body surface geometry deformation.

3.18 **boundary (of a 3D mesh)**: The set of boundary edges of a 3D mesh.

3.19 **boundary edge (of a 3D mesh)**: An edge of a 3D mesh which has exactly one incident face.

3.20 **bounding loop**: The boundary of a simple polygon.

3.21 **bounding loop index**: An index of the bounding loop table which contains the list of vertices lying on the bounding loop.

3.22 **branching triangle (of a triangle tree)**: A triangle of a triangle tree with three incident triangle tree edges.

3.23 **branching vertex (of a vertex tree or of a vertex graph)**: A vertex of a vertex tree with three or more incident vertex tree edges.

3.24 **byte aligned**: A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.

3.25 **byte**: Sequence of 8-bits.

3.26 **context based arithmetic encoding**: The method used for coding of binary shape; it is also referred to as cae.

3.27 **channel**: A digital medium or a network that stores or transports a bitstream constructed according to ISO/IEC 14496.

3.28 **chrominance format**: Defines the number of chrominance blocks in a macroblock.

3.29 **chrominance component**: A matrix, block or single sample representing one of the two colour difference signals related to the primary colours in the manner defined in the bitstream. The symbols used for the chrominance signals are Cr and Cb.

3.30 **coded B-VOP**: A B-VOP that is coded.

3.31 **coded VOP**: A coded VOP is a coded I-VOP, a coded P-VOP, a coded B-VOP, or a coded S(GMC)-VOP.

3.32 **coded I-VOP**: An I-VOP that is coded.

3.33 **coded P-VOP**: A P-VOP that is coded.

3.34 **coded S(GMC)-VOP**: A S(GMC)-VOP that is coded.

3.35 **coded video bitstream**: A coded representation of a series of one or more VOPs as defined in this part of ISO/IEC 14496.

3.36 **coded representation**: A data element as represented in its encoded form.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TOUQUA V. ADVANCE ANSI Store order #1121445 Downloaded 7/16/2007 1:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2073
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.37 **coding parameters**: The set of user-definable parameters that characterise a coded video bitstream. Bitstreams are characterised by coding parameters. Decoders are characterised by the bitstreams that they are capable of decoding.

3.38 **component**: A matrix, block or single sample from one of the three matrices (luminance and two chrominance) that make up a picture.

3.39 **composition process**: The (non-normative) process by which reconstructed VOPs are composed into a scene and displayed.

3.40 **compression**: Reduction in the number of bits used to represent an item of data.

3.41 **connected component (of a 3D mesh)**: A 3D mesh can be decomposed into one or more disjoint sets called connected components. Every face of the 3D mesh is assigned to an unique connected component, such that any pair of two faces that share an edge belongs to the same connected component.

3.42 **constant bitrate coded video**: A coded video bitstream with a constant bitrate.

3.43 **constant bitrate**: Operation where the bitrate is constant from start to finish of the coded bitstream.

3.44 **conversion ratio**: The size conversion ratio for the purpose of rate control of shape.

3.45 **corner (of a 3D mesh)**: A (face,vertex) pair, where the vertex is one of the sustaining vertices of the face.

3.46 **data element**: An item of data as represented before encoding and after decoding.

3.47 **DC coefficient**: The DCT coefficient for which the frequency is zero in both dimensions.

3.48 **DCT coefficient**: The amplitude of a specific cosine basis function.

3.49 **decoder input buffer**: The first-in first-out (FIFO) buffer specified in the video buffering verifier.

3.50 **decoder**: An embodiment of a decoding process.

3.51 **decoding order**: The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order.

3.52 **decoding (process)**: The process defined is this part of ISO/IEC 14496 that reads an input coded bitstream and produces decoded VOPs or audio samples, or 3D meshes.

3.53 **dequantisation**: The process of rescaling the quantised DCT coefficients after their representation in the bitstream has been decoded and before they are presented to the inverse DCT.

3.54 **digital storage media; DSM**: A digital storage or transmission device or system.

3.55 **discrete cosine transform; DCT**: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in annex A.

3.56 **display order**: The order in which the decoded pictures are displayed. Normally this is the same order in which they were presented at the input of the encoder.

3.57 **dual graph (of a 3D mesh)**: A graph composed of a set of *dual graph nodes* and a set of *dual graph edges*. Each dual graph node corresponds to a face of the 3D mesh. Each dual graph edge corresponds to an internal edge of the 3D mesh and links two dual graph nodes.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ANDERSON
ANSI Store order #: 01244 / Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2074
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.58 **edge (of a 3D mesh)**: An unordered pair of vertex indices of a 3D mesh, consecutive in one or more faces of the 3D mesh (modulo cyclical permutations of the sequences).

3.59 **editing**: The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this part of ISO/IEC 14496.

3.60 **encoder**: An embodiment of an encoding process.

3.61 **encoding (process)**: A process, not specified in this part of ISO/IEC 14496, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this part of ISO/IEC 14496.

3.62 **enhancement layer**: A relative reference to a layer (above the base layer) in a scalable hierarchy. For all forms of scalability, its decoding process can be described by reference to the lower layer decoding process and the appropriate additional decoding process for the enhancement layer itself.

3.63 **face (of a 3D mesh)**: An alternating sequence of vertices and edges, such that each edge in the sequence connects the vertices immediately preceding and following it in the sequence (with the exception of the last edge that connects the vertex preceding it with the first vertex of the sequence). No vertex may appear more than once in the sequence. In practice, a face is often represented by the list of vertices in the sequence.

3.64 **global motion compensation, GMC**: The use of global spatial transformation to improve the efficiency of the prediction of sample values. The prediction uses global spatial transformation to provide offsets into the past reference VOPs containing previously decoded sample values that are used to form the prediction error.

3.65 **face animation parameter units, FAPU**: Special normalized units (e.g. translational, angular, logical) defined to allow interpretation of FAPs with any facial model in a consistent way to produce reasonable results in expressions and speech pronunciation.

3.66 **face animation parameters, FAP**: Coded streaming animation parameters that manipulate the displacements and angles of face features, and that govern the blending of visemes and face expressions during speech.

3.67 **face animation table, FAT**: A downloadable function mapping from incoming FAPs to feature control points in the face mesh that provides piecewise linear weightings of the FAPs for controlling face movements.

3.68 **face calibration mesh**: Definition of a 3D mesh for calibration of the shape and structure of a baseline face model.

3.69 **face definition parameters, FDP**: Downloadable data to customize a baseline face model in the decoder to a particular face, or to download a face model along with the information about how to animate it. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs. FDPs may include feature points for calibrating a baseline face, face texture and coordinates to map it onto the face, animation tables, etc.

3.70 **face feature control point**: A normative vertex point in a set of such points that define the critical locations within face features for control by FAPs and that allow for calibration of the shape of the baseline face.

3.71 **face forest (of a 3D mesh)**: The set of all the face trees corresponding to the connected components of a 3D mesh.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ADAMANIS
ANSI Store order #X274457 Downloaded 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2075
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.72 **face interpolation transform, FIT**: A downloadable node type defined in ISO/IEC 14496-1:1999 for optional mapping of incoming FAPs to FAPs before their application to feature points, through weighted rational polynomial functions, for complex cross-coupling of standard FAPs to link their effects into custom or proprietary face models.

3.73 **face model mesh**: A 2D or 3D contiguous geometric mesh defined by vertices and planar polygons utilizing the vertex coordinates, suitable for rendering with photometric attributes (e.g. texture, color, normals).

3.74 **face tree (of a 3D mesh)**: A spanning tree constructed in the dual graph of a connected component.

3.75 **feathering**: A tool that tapers the values around edges of binary alpha mask for composition with the background.

3.76 **flag**: A one bit integer variable which may take one of only two values (zero and one).

3.77 **forbidden**: The term "forbidden" when used in the clauses defining the coded bitstream indicates that the value shall never be used. This is usually to avoid emulation of start codes.

3.78 **forced updating**: The process by which macroblocks are intra-coded from time-to-time to ensure that mismatch errors between the inverse DCT processes in encoders and decoders cannot build up excessively.

3.79 **forward compatibility**: A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.

3.80 **forward motion vector**: A motion vector that is used for motion compensation from a reference frame VOP at an earlier time in display order.

3.81 **forward prediction**: Prediction from the past reference VOP.

3.82 **frame**: A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and continuing through successive lines to the bottom of the frame.

3.83 **frame period**: The reciprocal of the frame rate.

3.84 **frame rate**: The rate at which frames are be output from the composition process.

3.85 **future reference VOP**: A future reference VOP is a reference VOP that occurs at a later time than the current VOP in display order.

3.86 **hierarchical 3D mesh**: a BIFS node representing a 3D mesh.

3.87 **hierarchical representation (of a 3D mesh)**: A 3D mesh representation with two or more levels of detail in increasing order of resolution, composed of a first coarse level of detail (base 3D mesh) followed by one or more refinement operations. Each refinement operation determines how to construct a new level of detail as a refinement of the current level of detail.

3.88 **hybrid scalability**: Hybrid scalability is the combination of two (or more) types of scalability.

3.89 **imaginary edge**: An imaginary edge is inserted to triangulate a polygon.

3.90 **incremental representation (of a 3D mesh)**: A representation that allows the terminal to start rendering a subset of the 3D mesh before the whole bitstream has been completely received and/or decoded.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ANDERSON
ANSI Store order #: [illegible] Downloaded: 7/16/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2076
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.91 **IndexedFaceSet**: a BIFS node representing a 3D mesh.

3.92 **interlace**: The property of conventional television frames where alternating lines of the frame represent different instances in time. In an interlaced frame, one of the field is meant to be displayed first. This field is called the first field. The first field can be the top field or the bottom field of the frame.

3.93 **internal edge (of a 3D mesh)**: An edge of a 3D mesh which has exactly two incident faces.

3.94 **I-VOP; intra-coded VOP**: A VOP coded using information only from itself.

3.95 **intra coding**: Coding of a macroblock or VOP that uses information only from that macroblock or VOP.

3.96 **intra shape coding**: Shape coding that does not use any temporal prediction.

3.97 **inter shape coding**: Shape coding that uses temporal prediction.

3.98 **jump edge (of a 3D mesh)**: An edge of the vertex graph which does not belong to any vertex tree.

3.99 **level**: A defined set of constraints on the values which may be taken by the parameters of this part of ISO/IEC 14496 within a particular profile. A profile may contain one or more levels. In a different context, level is the absolute value of a non-zero coefficient (see "run").

3.100 **layer**: In a scalable hierarchy denotes one out of the ordered set of bitstreams and (the result of) its associated decoding process.

3.101 **layered bitstream**: A single bitstream associated to a specific layer (always used in conjunction with layer qualifiers, e. g. "enhancement layer bitstream").

3.102 **leaf triangle (of a 3D mesh)**: A triangle of a triangle tree with exactly one incident triangle tree edge.

3.103 **leaf vertex (of a 3D mesh)**: A vertex of a vertex tree with exactly one incident vertex tree edge.

3.104 **lower layer**: A relative reference to the layer immediately below a given enhancement layer (implicitly including decoding of all layers below this enhancement layer).

3.105 **luminance component**: A matrix, block or single sample representing a monochrome representation of the signal and related to the primary colours in the manner defined in the bitstream. The symbol used for luminance is Y.

3.106 **Mbit**: 1 000 000 bits.

3.107 **macroblock**: The four 8×8 (or 16×16 in motion compensation decoding of Reduced Resolution VOP) blocks of luminance data and the two (for 4:2:0 chrominance format) corresponding 8×8 (or 16×16 in motion compensation decoding of Reduced Resolution VOP) blocks of chrominance data coming from a 16×16 (or 32×32 in motion compensation decoding of Reduced Resolution VOP) section of the luminance component of the picture. Macroblock is sometimes used to refer to the sample data and sometimes to the coded representation of the sample values and other data elements defined in the macroblock header of the syntax defined in this part of ISO/IEC 14496. The usage is clear from the context.

3.108 **manifold (3D mesh)**: A 3D mesh which has neither singular vertices nor singular edges.

3.109 **marching edge (of a 3D mesh)**: An edge shared by two triangles along a triangle run.

3.110 **mesh (2D)**: A 2D triangular mesh refers to a planar graph which tessellates a video object plane into triangular patches. The vertices of the triangular mesh elements are referred to as node points. The straight-line segments between node points are referred to as edges. Two triangles are adjacent if they share a common edge.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ANDREWS
ANSI Store order #: 01244 / Downloaded: 7/18/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2077
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.111 **mesh geometry**: The spatial locations of the node points and the triangular structure of a mesh.

3.112 **mesh motion**: The temporal displacements of the node points of a mesh from one time instance to the next.

3.113 **mesh object plane, MOP**: The instance of a mesh object at a given time.

3.114 **motion compensation**: The use of motion vectors to improve the efficiency of the prediction of sample values. The prediction uses motion vectors to provide offsets into the past and/or future reference VOPs containing previously decoded sample values that are used to form the prediction error.

3.115 **motion estimation**: The process of estimating motion vectors during the encoding process.

3.116 **motion vector**: A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture or field to the coordinates in a reference VOP.

3.117 **motion vector for shape**: A motion vector used for motion compensation of shape.

3.118 **multiple auxiliary components**: Auxiliary components are defined for the VOP on a pixel-by-pixel basis, and contain data related to the video object, such as disparity, depth, and additional texture. Up to three auxiliary components (including the grayscale shape) are possible.

3.119 **non-intra coding**: Coding of a macroblock or a VOP that uses information both from itself and from macroblocks and VOPs occurring at other times.

3.120 **non-manifold (3D mesh)**: A 3D mesh that contains singular vertices or singular edges.

3.121 **non-orientable (3D mesh)**: A manifold 3D mesh which is not orientable.

3.122 **opaque macroblock**: A macroblock with shape mask of all 255's.

3.123 **orientable (3D mesh)**: A manifold 3D mesh for which an orientation can be chosen for each of its faces such that for each internal edge, the two faces incident to the edge induce opposite orientations on the common edge. All non-manifold 3D meshes are non-orientable.

3.124 **orientation (of a face of a 3D mesh)**: One of the two possible cyclical orderings of its sustaining vertices.

3.125 **orientation (of an edge of a 3D mesh)**: One of the two possible orderings of the two vertices of the edge.

3.126 **oriented (manifold 3D mesh)**: A manifold 3D mesh for which an orientation has been chosen for each of its faces such that for each internal edge, the two faces incident to the edge induce opposite orientations on the common edge. Note that a manifold 3D mesh may be orientable but not oriented. Such a mesh can be oriented by choosing a consistent orientation for its faces (which may require inverting the orientation of one or more faces).

3.127 **P-VOP; predictive-coded VOP**: A picture that is coded using motion compensated prediction from the past VOP.

3.128 **parameter**: A variable within the syntax of this part of ISO/IEC 14496 which may take one of a range of values. A variable which can take one of only two values is called a flag.

3.129 **partition**: a bitstream segment of a 3D mesh that can be decoded independently.

3.130 **past reference picture**: A past reference VOP is a reference VOP that occurs at an earlier time than the current VOP in composition order.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ANDERSON
ANSI Store order #X07445, Downloaded: 7/18/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2078
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.131 **picture**: Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. A "coded VOP" was defined earlier. For progressive video, a picture is identical to a frame.

3.132 **prediction**: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.

3.133 **prediction error**: The difference between the actual value of a sample or data element and its predictor.

3.134 **predictor**: A linear combination of previously decoded sample values or data elements.

3.135 **profile**: A subset of the syntax of this part of ISO/IEC 14496, defined in terms of Visual Object Types.

3.136 **progressive**: The property of film frames where all the samples of the frame represent the same instances in time.

3.137 **quantisation matrix**: A set of sixty-four 8-bit values used by the dequantiser.

3.138 **quantised DCT coefficients**: DCT coefficients before dequantisation. A variable length coded representation of quantised DCT coefficients is transmitted as part of the coded video bitstream.

3.139 **quantiser scale**: A scale factor coded in the bitstream and used by the decoding process to scale the dequantisation.

3.140 **random access**: The process of beginning to read and decode the coded bitstream at an arbitrary point.

3.141 **reconstructed VOP**: A reconstructed VOP consists of three matrices of 8-bit numbers representing the luminance and two chrominance signals. It is obtained by decoding a coded VOP.

3.142 **reference layer**: A layer to be referenced for prediction in a scalable hierarchy. The video_object_id of the reference layer should be the same value as the video_object_id of the enhancement layer. The ref_layer_id of the enhancement layer is set to the same value as the video_object_layer_id of the reference layer.

3.143 **reference VOP**: A reference VOP is a reconstructed VOP that was coded in the form of a coded I-VOP, a coded P-VOP, or a coded S(GMC)-VOP. Reference VOPs are used for forward and backward prediction when P-VOPs, B-VOPs, and S(GMC)-VOPs are decoded.

3.144 **regular triangle (of a triangle tree)**: A triangle of a triangle tree with exactly two incident triangle tree edges.

3.145 **regular vertex (of a 3D mesh)**: A vertex of a 3D mesh is *regular* if the set of incident edges and incident faces can be ordered in an alternating sequence of edges and faces, such that each face in the sequence is bounded by the edges immediately preceeding and following it in the sequence.

3.146 **regular vertex (of a vertex tree)**: A vertex of a vertex tree with exactly two incident vertex tree edges.

3.147 **reordering delay**: A delay in the decoding process that is caused by VOP reordering.

3.148 **reserved**: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/EC defined extensions.

3.149 **scalable hierarchy**: coded video data consisting of an ordered set of more than one video bitstream.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG Y. SUH/Version
ANSI Store order #1012445 Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2079
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.150 **scalability**: Scalability is the ability of a decoder to decode an ordered set of bitstreams to produce a reconstructed sequence. Moreover, useful video is output when subsets are decoded. The minimum subset that can thus be decoded is the first bitstream in the set which is called the base layer. Each of the other bitstreams in the set is called an enhancement layer. When addressing a specific enhancement layer, "lower layer" refers to the bitstream that precedes the enhancement layer.

3.151 **side information**: Information in the bitstream necessary for controlling the decoder.

3.152 **run**: The number of zero coefficients preceding a non-zero coefficient, in the scan order. The absolute value of the non-zero coefficient is called "level".

3.153 **S-VOP**: A picture that is coded using information obtained by warping whole or part of a static sprite.

3.154 **S(GMC)-VOP**: A picture that is coded using prediction based on global motion compensation from the past VOP.

3.155 **saturation**: Limiting a value that exceeds a defined range by setting its value to the maximum or minimum of the range as appropriate.

3.156 **simple polygon (of a 3D mesh)**: A connected component of which the dual graph is a tree. A simple polygon has exactly one boundary, and all its vertices lie on the boundary.

3.157 **singular edge (of a 3D mesh)**: An edge of a 3D mesh which has three or more incident faces.

3.158 **singular vertex (of a 3D mesh)**: A vertex of a 3D mesh which is not a regular vertex.

3.159 **source; input**: Term used to describe the video material or some of its attributes before encoding.

3.160 **spanning forest (of a graph)**: A subgraph of a graph composed of one or more trees, each tree spanning a connected component of the graph.

3.161 **spatial prediction**: prediction derived from a decoded frame of the reference layer decoder used in spatial scalability.

3.162 **spatial scalability**: A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer without using motion vectors. The layers can have different VOP sizes or VOP rates.

3.163 **static sprite**: The luminance, chrominance and binary alpha plane for an object which does not vary in time.

3.164 **start codes**: 32-bit codes embedded in that coded bitstream that are unique. They are used for several purposes including identifying some of the structures in the coding syntax.

3.165 **stuffing (bits); stuffing (bytes)**: Code-words that may be inserted into the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.

3.166 **temporal prediction**: prediction derived from reference VOPs other than those defined as spatial prediction.

3.167 **temporal scalability**: A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer using motion vectors. The layers have identical frame size, and but can have different VOP rates.

3.168 **top layer**: the topmost layer (with the highest layer_id) of a scalable hierarchy.

3.169 **transparent macroblock**: A macroblock with shape mask of all zeros.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ANDERSON
ANSI Store order #X07445, Downloaded: 7/18/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2080
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.170 **traversal order**: The traversal order determines which branch to be traversed first at each branching triangle.

3.171 **tree loop (of a 3D mesh)**: An alternating sequence of vertices and edges bounding the hole created by cutting a 3D mesh through a tree of edges within a forest split operation.

3.172 **tree loop face**: a face incident to a tree loop.

3.173 **tree loop corner**: a corner of a tree loop face incident to a tree loop edge.

3.174 **triangle tree (of a 3D mesh)**: A spanning tree in the dual graph of a connected component of a triangular 3D mesh.

3.175 **triangle run**: A path in a triangle tree connecting a first leaf or branching triangle to a last leaf or branching triangle through zero or more intermediate triangles.

3.176 **triangular 3D mesh**: A 3D mesh with triangular faces. Also, a 3D mesh after all the polygonal faces have been triangulated.

3.177 **variable bitrate**: Operation where the bitrate varies with time during the decoding of a coded bitstream.

3.178 **variable length coding; VLC**: A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.

3.179 **vertex graph (of a 3D mesh)**: The graph formed by the vertices and the edges of the 3D mesh which do not correspond to dual graph edges included in the face forest.

3.180 **vertex run**: A path in a vertex tree connecting a first leaf or branching vertex to a last leaf or branching vertex through zero or more intermediate regular vertices.

3.181 **vertex tree**: A rooted tree spanning one connected component of the vertex graph.

3.182 **video buffering verifier; VBV**: Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an encoder or editing process may produce.

3.183 **video complexity verifier; VCV**: Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum processing requirements of the bitstream that an encoder or editing process may produce.

3.184 **video memory verifier; VMV**: Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum reference memory requirements of the bitstream that an encoder or editing process may produce.

3.185 **video object plane; VOP**: The instance of a video object at a given time.

3.186 **video presentation verifier; VPV**: Part of a hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the maximum presentation memory requirements of the bitstream that an encoder or editing process may produce.

3.187 **video session**: The highest syntactic structure of coded video bitstreams. It contains a series of one or more coded video objects.

3.188 **viseme**: the physical (visual) configuration of the mouth, tongue and jaw that is visually correlated with the speech sound corresponding to a phoneme.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS J. ADAMS/ANSI Store order #: 0124457 Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2081
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.189 **virtual triangle**: A virtual triangle is a leaf triangle added, if necessary, at the end of a partition to signal the end of the partition to the decoder.

3.190 **VOP reordering**: The process of reordering the reconstructed VOPs when the decoding order is different from the composition order for display. VOP reordering occurs when B-VOPs are present in a bitstream. There is no VOP reordering when decoding low delay bitstreams.

3.191 **warping**: Processing applied to extract a sprite VOP from a static sprite or a reference VOP. It consists of a global spatial transformation driven by a few motion parameters (0,2,4,6,8), to recover luminance, chrominance and shape information.

3.192 **Y-vertex**: The vertex of a branching triangle opposite to the incident marching edge.

3.193 **zigzag scanning order**: A specific sequential ordering of the DCT coefficients from (approximately) the lowest spatial frequency to the highest.

4 Abbreviations and symbols

The mathematical operators used to describe this part of ISO/IEC 14496 are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin from zero.

4.1 Arithmetic operators

\+ Addition.

\- Subtraction (as a binary operator) or negation (as a unary operator).

++ Increment. i.e. x++ is equivalent to $x = x + 1$

-- Decrement. i.e. x-- is equivalent to $x = x - 1$

$\begin{pmatrix} * \\ \times \end{pmatrix}$ Multiplication.

^ Power.

/ Integer division with truncation of the result toward zero. For example, 7/4 and -7/-4 are truncated to 1 and -7/4 and 7/-4 are truncated to -1.

// Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and -3//2 is rounded to -2.

/// Integer division with sign dependent rounding to the nearest integer. Half-integer values when positive are rounded away from zero, and when negative are rounded towards zero. For example 3///2 is rounded to 2, and -3///2 is rounded to -1.

//// Integer division with truncation towards the negative infinity.

÷ Used to denote division in mathematical equations where no truncation or rounding is intended.

% Modulus operator. Defined only for positive numbers.

Sign( ) $\mathrm{Sign}(x) = \begin{cases} 1 & x >= 0 \\ -1 & x < 0 \end{cases}$

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG YOUNG
ANSI Store order #: 01744 / Downloaded: 2018-07-13 08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2082
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Abs( ) $\quad Abs(x) = \begin{cases} x & x >= 0 \\ -x & x < 0 \end{cases}$

$\sum_{i=a}^{b} f(i)$ The summation of the $f(i)$ with i taking integral values from a up to, but not including b.

4.2 Logical operators

|| Logical OR.

&& Logical AND.

! Logical NOT.

4.3 Relational operators

> Greater than.

>= Greater than or equal to.

≥ Greater than or equal to.

< Less than.

<= Less than or equal to.

≤ Less than or equal to.

== Equal to.

!= Not equal to.

max [, ... ,] the maximum value in the argument list.

min [, ... ,] the minimum value in the argument list.

4.4 Bitwise operators

& AND

| OR

>> Shift right with sign extension.

<< Shift left with zero fill.

4.5 Conditional operators

?: $\quad (condition?\ a : b) = \begin{cases} a & \text{if condition is true,} \\ b & \text{otherwise.} \end{cases}$

4.6 Assignment

= Assignment operator.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC, ADVANCED
ANSI Store order #X274457 Downloaded 7/16/2007 3:09:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2083
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4.7 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in this part of ISO/IEC 14496. Bit strings are generally written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance. For convenience large strings are occasionally written in hexadecimal, in this case conversion to a binary in the conventional manner will yield the value of the bit string. Thus the left most hexadecimal digit is first and in each hexadecimal digit the most significant of the four bits is first.

uimsbf Unsigned integer, most significant bit first.

simsbf Signed integer, in twos complement format, most significant (sign) bit first.

vlclbf Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written. The byte order of multibyte words is most significant byte first.

4.8 Constants

π 3,141 592 653 58...

e 2,718 281 828 45...

5 Conventions

5.1 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in subclause 6.2. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in subclause 6.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:

while (condition) {
  data_element
  ...
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do {
  data_element
  ...
} while (condition)
	The data element always occurs at least once.
	
	The data element is repeated until the condition is not true.

do {
  ...
  continue
  ...
} while (condition)
	Continues execution of the next repetition of the nearest while-do loop.

if (condition) {
  data_element
  ...
} else {
	If the condition is true, then the first group of data elements occurs next in the data stream.
	
	If the condition is not true, then the second group of data

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS CHANDLER
ANSI Store order #: 01/74467 Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2084
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

data_element ... }	elements occurs next in the data stream.

for (i = m; i < n; i++) {
data_element
...
}
	The group of data elements occurs (n-m) times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to m for the first occurrence, incremented by one for the second occurrence, and so forth.

/* comment ... */	Explanatory comment that may be deleted entirely without in any way altering the syntax.

This syntax uses the 'C-code' convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true and a variable or expression evaluating to a zero value is equivalent to a condition that is false. In many cases a literal string is used in a condition. For example:

```
if ( video_object_layer_shape == "rectangular" ) ...
```

In such cases the literal string is that used to describe the value of the bitstream element in subclause 6.3. In this example, we see that "rectangular" is defined in a Table 6-14 to be represented by the two bit binary number '00'.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the brackets { } are omitted when only one data element follows.

data_element [n] data_element [n] is the n+1th element of an array of data.

data_element [m][n] data_element [m][n] is the m+1, n+1th element of a two-dimensional array of data.

data_element [i][m][n] data_element [i][m][n] is the i+1, m+1, n+1th element of a three-dimensional array of data.

While the syntax is expressed in procedural terms, it should not be assumed that subclause 6.2 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include means to look for start codes in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardised.

5.2 Definition of functions

Several utility functions for picture coding algorithm are defined as follows:

5.2.1 Definition of next_bits() function

The function next_bits() permits comparison of a bit string with the next bits to be decoded in the bitstream.

5.2.2 Definition of bytealigned() function

The function bytealigned () returns 1 if the current position is on a byte boundary, that is the next bit in the bitstream is the first bit in a byte. Otherwise it returns 0.

5.2.3 Definition of nextbits_bytealigned() function

The function nextbits_bytealigned() returns a bit string starting from the next byte aligned position. This permits comparison of a bit string with the next byte aligned bits to be decoded in the bitstream. If the current location in the bitstream is already byte aligned and the 8 bits following the current location are '011 1111', the bits subsequent to these 8 bits are returned. The current location in the bitstream is not changed by this function.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ADAMIAK
ANSI Store order #1537445, Downloaded: 2/16/2021 1:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2085
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5.2.4 Definition of next_start_code() function

The next_start_code() function removes any zero bit and a string of 0 to 7 '1' bits used for stuffing and locates the next start code.

next_start_code() {	No. of bits	Mnemonic
zero_bit	1	'0'
while (!byteAligned())		
one_bit	1	'1'
}		

5.2.5 Definition of next_resync_marker() function

The next_resync_marker() function removes any zero bit and a string of 0 to 7 '1' bits used for stuffing and locates the next resync marker; it thus performs similar operation as next_start_code() but for resync_marker.

next_resync_marker() {	No. of bits	Mnemonic
zero_bit	1	'0'
while (!byteAligned())		
one_bit	1	'1'
}		

5.2.6 Definition of transparent_mb() function

The function transparent_mb() returns 1 if the current macroblock consists only of transparent pixels. Otherwise it returns 0.

5.2.7 Definition of transparent_block() function

The function transparent_block(i) returns 1 if the 8×8 with index i consists only of transparent pixels. Otherwise it returns 0. The index value for each block is defined in Figure 6-5.

5.2.8 Definition of byte_align_for_upstream() function

The function byte_align_for_upstream() removes a string of '1' used for stuffing from the upstream message. When the message is already byte aligned before the byte_align_for_upstream() function, additional byte stuffing is no longer allowed.

byte_align_for_upstream() {	No. of bits	Mnemonic
while (!byte_aligned())		
one_bit	1	'1'
}		

5.3 Reserved, forbidden and marker_bit

The terms "reserved" and "forbidden" are used in the description of some values of several fields in the coded bitstream.

The term "reserved" indicates that the value may be used in the future for ISO/IEC defined extensions.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... 2005/02/21 ...
ANSI Store order #X274457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2086
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The term "forbidden" indicates a value that shall never be used (usually in order to avoid emulation of start codes).

The term "marker_bit" indicates a one bit integer in which the value zero is forbidden (and it therefore shall have the value '1'). These marker bits are introduced at several points in the syntax to avoid start code emulation.

The term "zero_bit" indicates a one bit integer with the value zero.

5.4 Arithmetic precision

In order to reduce discrepancies between implementations of this part of ISO/IEC 14496, the following rules for arithmetic operations are specified.

(a) Where arithmetic precision is not specified, such as in the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values.

(b) Where ranges of values are given, the end points are included if a square bracket is present, and excluded if a round bracket is used. For example, [a , b) means from a to b, including a but excluding b.

6 Visual bitstream syntax and semantics

6.1 Structure of coded visual data

Coded visual data can be of several different types, such as video data, still texture data, 2D mesh data or facial animation parameter data.

Synthetic objects and their attribution are structured in a hierarchical manner to support both bitstream scalability and object scalability. ISO/IEC 14496-1:1999 of the specification provides the approach to spatial-temporal scene composition including normative 2D/3D scene graph nodes and their composition supported by Binary Interchange Format Specification. At this level, synthetic and natural object composition relies on ISO/IEC 14496-1:1999 with subsequent (non-normative) rendering performed by the application to generate specific pixel-oriented views of the models.

Coded video data consists of an ordered set of video bitstreams, called layers. If there is only one layer, the coded video data is called non-scalable video bitstream. If there are two layers or more, the coded video data is called a scalable hierarchy.

One of the layers is called base layer, and it can always be decoded independently. Other layers are called enhancement layers, and can only be decoded together with the lower layers (previous layers in the ordered set), starting with the base layer. The multiplexing of these layers is discussed in ISO/IEC 14496-1:1999. The base layer of a scalable set of streams can be coded by other standards. The Enhancement layers shall conform to this part of ISO/IEC 14496. In general the visual bitstream can be thought of as a syntactic hierarchy in which syntactic structures contain one or more subordinate structures.

Visual texture, referred to herein as still texture coding, is designed for maintaining high visual quality in the transmission and rendering of texture under widely varied viewing conditions typical of interaction with 2D/3D synthetic scenes. Still texture coding provides for a multi-layer representation of luminance, color and shape. This supports progressive transmission of the texture for image build-up as it is received by a terminal. Also supported is the downloading of the texture resolution hierarchy for construction of image pyramids used by 3D graphics APIs. Quality and SNR scalability are supported by the structure of still texture coding.

Coded mesh data consists of a single non-scalable bitstream. This bitstream defines the structure and motion of a 2D mesh object. Texture that is to be mapped onto the mesh geometry is coded separately.

Coded FBA parameter data consists of one non-scaleable bitstream. It defines the animation of the face and body model of the decoder. Face and body animation data is structured as standard formats for downloadable models and their animation controls, and a single layer of compressed face and body animation parameters used for remote manipulation of the face and body model. The face is a node in a scene graph that includes face geometry

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #1537445, Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2087
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ready for rendering. The body is also a node in a scene graph that includes body geometry ready for rendering. The face node belongs to the scene graph that is defined by the body node.The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. The body is also controlled by instances of Body Definition Parameter (BDP) sets and/or Body Animation Parameter (BAP) sets.Upon initial or baseline construction, the FBA object contains a generic face with a neutral expression and a generic body with default position. This object can receive FAPs and BAPs from the bitstream and be subsequently rendered to produce animation of the face and the body. If FDPs and BDPs are transmitted, the generic face and body are transformed into a particular face and body of specific shape and appearance. A downloaded face model via FDPs is a scene graph for insertion in the face node, and the downloaded body model via BDPs is a scene graph for insertion of the body node.

6.1.1 Visual object sequence

Visual object sequence is the highest syntactic structure of the coded visual bitstream.

A visual object sequence commences with a visual_object_sequence_start_code which is followed by one or more visual objects coded concurrently. The visual object sequence is terminated by a visual_object_sequence_end_code.

6.1.2 Visual object

A visual object commences with a visual_object_start_code, is followed by profile and level identification, and a visual object id, and is followed by a video object, a still texture object, a mesh object, or an FBA object.

6.1.3 Video object

A video object commences with a video_object_start_code and is followed by one or more video object layers.

6.1.3.1 Progressive and interlaced sequences

This part of ISO/IEC 14496 deals with coding of both progressive and interlaced sequences.

The sequence, at the output of the decoding process, consists of a series of reconstructed VOPs separated in time and are readied for display via the compositor.

6.1.3.2 Frame

A frame consists of three rectangular matrices of integers: a luminance matrix (Y), and two chrominance matrices (Cb and Cr).

6.1.3.3 VOP

A reconstructed VOP is obtained by decoding a coded VOP. A coded VOP may have been derived from either a progressive or interlaced frame.

6.1.3.4 VOP types

There are four types of VOPs that use different coding methods:

1. An Intra-coded (I) VOP is coded using information only from itself.
2. A Predictive-coded (P) VOP is a VOP which is coded using motion compensated prediction from a past reference VOP.
3. A Bidirectionally predictive-coded (B) VOP is a VOP which is coded using motion compensated prediction from a past and/or future reference VOP(s).
4. A sprite (S) VOP is a VOP for a sprite object or a VOP which is coded using predicton based on global motion compensation from a past reference VOP.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. ANDERSON
ANSI Store order #: 01244 / Downloaded: 2018-07-13 08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2088
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.1.3.5 I-VOPs and group of VOPs

I-VOPs are intended to assist random access into the sequence. Applications requiring random access, fast-forward playback, or fast reverse playback may use I-VOPs relatively frequently.

I-VOPs may also be used at scene cuts or other cases where motion compensation is ineffective.

Group of VOP (GOV) header is an optional header that can be used immediately before a coded I-VOP to indicate to the decoder:

1) the modulo part (i.e. the full second units) of the time base for the next VOP to be displayed after having decoded a GOV header

2) if the first consecutive B-VOPs immediately following the coded I-VOP can be reconstructed properly in the case of a random access.

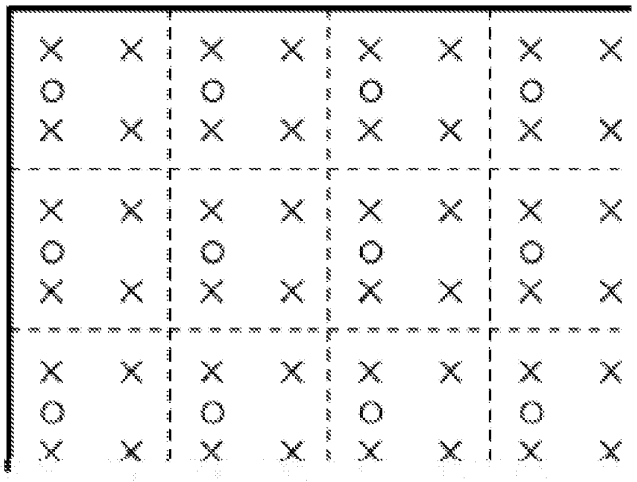
In a non scalable bitstream or the base layer of a scalable bitstream, the first coded VOP following a GOV header shall be a coded I-VOP.

6.1.3.6 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

The luminance and chrominance samples are positioned as shown in Figure 6-1.The two variations in the vertical and temporal positioning of the samples for interlaced VOPs are shown in Figure 6-2 and Figure 6-3.

Figure 6-4 shows the vertical and temporal positioning of the samples in a progressive frame.



X Represent luminance samples

O Represent chrominance samples

Figure 6-1 -- The position of luminance and chrominance samples in 4:2:0 data

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODIA V. ADVANCE/ANSI Store order #: 0 Downloaded: 7/18/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2089
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

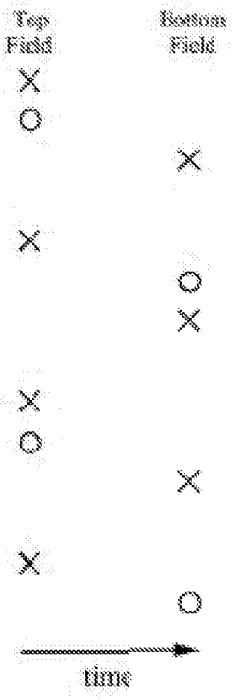


Figure 6-3 – Vertical and temporal positions of samples in an interlaced frame with top_field_first=1

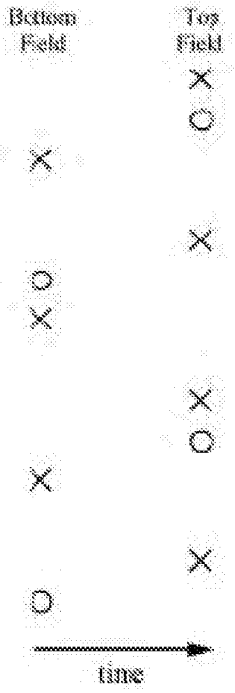


Figure 6-3 -- Vertical and temporal position of samples in an interlaced frame with top_field_first=0

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG YOUNG
ANSI Store order #: 07/44[illegible] Downloaded: 7/18/2007 1:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2090
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

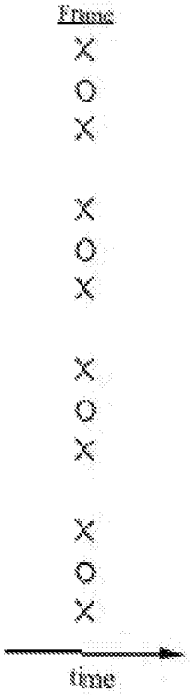


Figure 6-4 -- Vertical and temporal positions of samples in a progressive frame

The binary alpha plane for each VOP is represented by means of a bounding rectangle as described in clause F.2, and it has always the same number of lines and pixels per line as the luminance plane of the VOP bounding rectangle. The positions between the luminance and chrominance pixels of the bounding rectangle are defined in this clause according to the 4:2:0 format. For the progressive case, each 2x2 block of luminance pixels in the bounding rectangle associates to one chrominance pixel. For the interlaced case, each 2x2 block of luminance pixels of the same field in the bounding rectangle associates to one chrominance pixe of that field.

In order to perform the padding process on the two chrominance planes, it is necessary to generate a binary alpha plane which has the same number of lines and pixels per line as the chrominance planes. Therefore, when non-scalable shape coding is used, this binary alpha plane associated with the chrominance planes is created from the binary alpha plane associated with the luminance pane xy the subsampling process defined below:

For each 2x2 block of the binary alpha plane associated with the luminance plane of the bounding rectangle (of the same frame for the progressive and of the same field for the interlaced case), the associated pixel value of the binary alpha plane associated with the chrominance planes is set to 255 if any pixel of said 2x2 block of the binary alpha plane associated with the luminance plane equals 255.

6.1.3.7 VOP reordering

When a video object layer contains coded B-VOPs, the number of consecutive coded B-VOP is variable and unbounded. The first coded VOP shall not be a B-VOP.

A video object layer may contain no coded P-VOPs or no coded S(GMC)-VOPs. A video object layer may also contain no coded I-VOPs in which case some care is required at the start of the video object layer and within the video object layer to effect both random access and error recovery.

The order of the coded VOPs in the bitstream, also called decoding order, is the order in which a decoder reconstructs them. The order of the reconstructed VOPs at the output of the decoding process, also called the

© ISO/IEC 2001 -- All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TOSHIBA V. ADVANCED
AMS Store order #0376481 Downloaded 7/16/2007 1:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2091
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

display order, is not always the same as the decoding order and this subclause defines the rules of VOP reordering that shall happen within the decoding process.

When the video object layer contains no coded B-VOPs, the decoding order is the same as the display order.

When B-VOPs are present in the video object layer re-ordering is performed according to the following rules:

If the current VOP in decoding order is a B-VOP the output VOP is the VOP reconstructed from that B-VOP.

If the current VOP in decoding order is a I-VOP, P-VOP, or S(GMC)-VOPs the output VOP is the VOP reconstructed from the previous I-VOP or P-VOP, or S(GMC)-VOP if one exists. If none exists, at the start of the video object layer, no VOP is output.

The following is an example of VOPs taken from the beginning of a video object layer. In this example there are two coded B-VOPs between successive coded P-VOPs and also two coded B-VOPs between successive coded I- and P-VOPs. VOP '1I' is used to form a prediction for VOP '4P'. VOPs '4P' and '1I' are both used to form predictions for VOPs '2B' and '3B'. Therefore the order of coded VOPs in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder shall display them in the order '1I', '2B', '3B', '4P'.

At the encoder input,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

At the encoder output, in the coded bitstream, and at the decoder input,

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

At the decoder output,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

In terms of VOP reordering, an S(GMC)-VOP can be regarded as a P-VOP.

6.1.3.8 Macroblock

A macroblock contains a section of the luminance component and the spatially corresponding chrominance components. The term macroblock can either refer to source and decoded data or to the corresponding coded data elements. A skipped macroblock is one for which no information is transmitted. Presently there is only one chrominance format for a macroblock, namely, 4:2:0 format. The orders of blocks in a macroblock is illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in Figure 6-5.

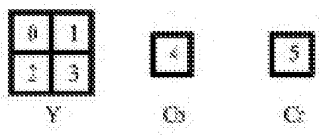


Figure 6-5 -- 4:2:0 Macroblock structure

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG YOUNG
ANSI Store order #1027445: Downloaded 7/16/2007 1:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2092
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The organisation of VOPs into macroblocks is as follows.

For the case of a progressive VOP, the interlaced flag (in the VOP header) is set to "0" and the organisation of lines of luminance VOP into macroblocks is called frame organization and is illustrated in Figure 6-6. In this case, frame DCT coding is employed.

For the case of interlaced VOP, the interlaced flag is set to "1" and the organisation of lines of luminance VOP into macroblocks can be either frame organization or field organization and thus both frame and field DCT coding may be used in the VOP.

- In the case of frame DCT coding, each luminance block shall be composed of lines from two fields alternately. This is illustrated in Figure 6-6.
- In the case of field DCT coding, each luminance block shall be composed of lines from only one of the two fields. This is illustrated in Figure 6-7.

Only frame DCT coding is applied to the chrominance blocks. It should be noted that field based predictions may be applied for these chrominance blocks which will require predictions of 8x4 regions (after half-sample filtering).

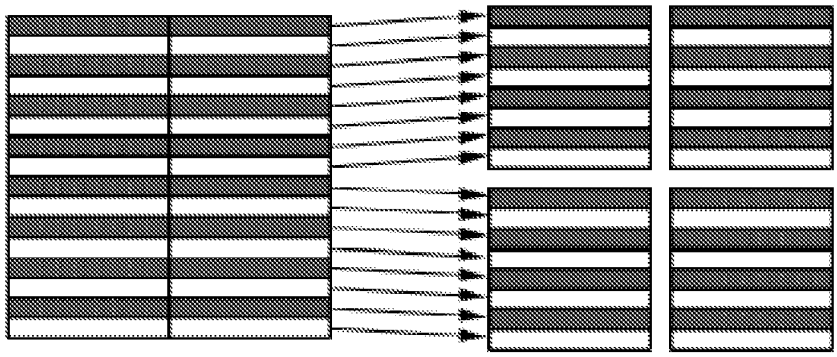


Figure 6-6 – Luminance macroblock structure in frame DCT coding

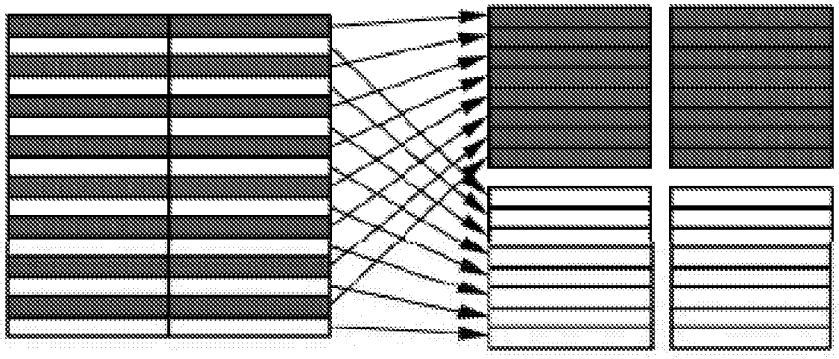


Figure 6-7 – Luminance macroblock structure in field DCT coding

6.1.3.9 Block

The term block can refer either to source and reconstructed data or to the DCT coefficients or to the corresponding coded data elements.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADMINISTRATOR
ANSI Store order #: [illegible] Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensed only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2093
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

When the block refers to source and reconstructed data it refers to an orthogonal section of a luminance or chrominance component with the same number of lines and samples. There are 8 lines and 8 samples/line in the block.

6.1.4 Mesh object

A 2D triangular *mesh* refers to a tessellation of a 2D visual object plane into triangular patches. The vertices of the triangular patches are called *node points*. The straight-line segments between node points are called edges. Two triangles are *adjacent* if they share a common edge.

A *dynamic* 2D mesh consists of a temporal sequence of 2D triangular meshes, where each mesh has the same topology, but node point locations may differ from one mesh to the next. Thus, a dynamic 2D mesh can be specified by the geometry of the initial 2D mesh and motion vectors at the node points for subsequent meshes, where each motion vector points from a node point of the previous mesh in the sequence to the corresponding node point of the current mesh. The dynamic 2D mesh can be used to create 2D animations by mapping texture from e.g. a video object plane onto successive 2D meshes.

A 2D dynamic mesh with *implicit structure* refers to a 2D dynamic mesh of which the initial mesh has either *uniform* or *Delaunay* topology. In both cases, the topology of the initial mesh does not have to be coded (since it is implicitly defined), only the node point locations of the initial mesh have to be coded. Note that in both the uniform and Delaunay case, the mesh is restricted to be *simple*, i.e. it consists of a single connected component without any holes, topologically equivalent to a disk.

A *mesh object* represents the geometry and motion of a 2D triangular mesh. A mesh object consists of one or more *mesh object planes*, each corresponding to a 2D triangular mesh at a certain time instance. An example of a mesh object is shown in the figure below.

A sequence of mesh object planes represents the piece-wise deformations to be applied to a video object plane or still texture object to create a synthetic animated video object. Triangular patches of a video object plane are to be warped according to the motion of corresponding triangular mesh elements. The motion of mesh elements is specified by the temporal displacements of the mesh node points.

The syntax and semantics of the mesh object pertains to the mesh geometry and mesh motion only; the video object to be used in an animation is coded separately. The warping or texture mapping applied to render visual object planes is handled in the context of scene composition. Furthermore, the syntax does not allow explicit encoding of other mesh properties such as colors or texture coordinates.

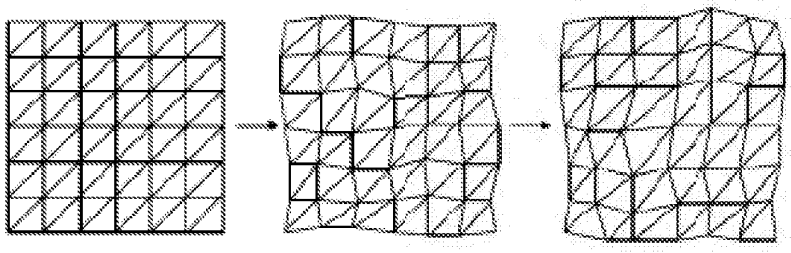


Figure 6-8 -- Mesh object with uniform triangular geometry

6.1.4.1 Mesh object plane

There are two types of mesh object planes that use different coding methods.

An *intra-coded* mesh object plane codes the geometry of a single 2D mesh. An intra-coded mesh is either of uniform or Delaunay type. In the case of a mesh of uniform type, the mesh geometry is coded by a small set of parameters. In the case of a mesh of Delaunay type, the mesh geometry is coded by the locations of the node points and boundary edge segments. The triangular mesh structure is specified implicitly by the coded information.

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ECKSTEIN
ANSI Store order #: 01214B / Downloaded: 2018-07-13 08:21 PM ET
Single user licensed only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2094
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

A *predictive-coded* mesh object plane codes a 2D mesh using temporal prediction from a past reference mesh object plane. The triangular structure of a predictive-coded mesh is identical to the structure of the reference mesh used for prediction; however, the locations of node points may change. The displacements of node points represent the motion of the mesh and are coded by specifying the motion vectors of node points from the reference mesh towards the predictive-coded mesh.

The locations of mesh node points correspond to locations in a video object or still texture object. Mesh node point locations and motion vectors are represented and coded with half pixel accuracy.

6.1.5 FBAobject

Conceptually the FBA object consists of a collection of nodes in a scene graph which are animated by the FBA object bitstream. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. Upon construction, the FBA object contains a generic face with a neutral expression. This face can already be rendered. It is also immediately capable of receiving the FAPs from the bitstream, which will produce animation of the face: expressions, speech etc. If FDPs are received, they are used to transform the generic face into a particular face determined by its shape and (optionally) texture. Optionally, a complete face model can be downloaded via the FDP set as a scene graph for insertion in the face node.

The FDP and FAP sets are designed to allow the definition of a facial shape and texture, as well as animation of faces reproducing expressions, emotions and speech pronunciation. The FAPs, if correctly interpreted, will produce reasonably similar high level results in terms of expression and speech pronunciation on different facial models, without the need to initialize or calibrate the model. The FDPs allow the definition of a precise facial shape and texture in the setup phase. If the FDPs are used in the setup phase, it is also possible to produce more precisely the movements of particular facial features. Using a phoneme/bookmark to FAP conversion it is possible to control facial models accepting FAPs via TTS systems. The translation from phonemes to FAPs is not standardized. It is assumed that every decoder has a default face model with default parameters. Therefore, the setup stage is not necessary to create face animation. The setup stage is used to customize the face at the decoder.

Upon construction, the Body model contains a generic *virtual* human or human-like body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the decoder's generic body into a particular body determined by the parameter contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. Similar to the face, the BAPs can be transmitted also without first downloading BDPs, in which case the decoder animates its local model.

No assumption is made and no limitation is imposed on the range of defined mobilities for humanoid animation. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models.

6.1.5.1 Structure of the face and body object bitstream

A face and body object is formed by a temporal sequence of face and body object planes. This is depicted as follows in Figure 6-9.

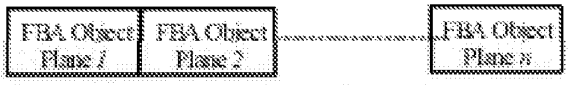


Figure 6-9 -- Structure of the FBA object bitstream

An FBA object represents a node in an ISO/IEC 14496 scene graph. An ISO/IEC 14496 scene is understood as a composition of Audio-Visual objects according to some spatial and temporal relationships. The scene graph is the hierarchical representation of the ISO/IEC 14496 scene structure (see ISO/IEC 14496-1:1999).

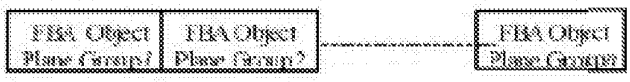
Alternatively, an FBA object can be formed by a temporal sequence of FBA object plane groups (called segments for simplicity), where each FBA object plane group itself is composed of a temporal sequence of 16 FBA object planes, as depicted in the following:

© ISO/IEC 2001 – All rights reserved

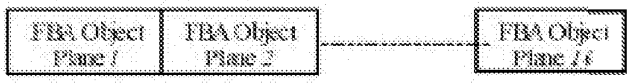
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC / ADAMANT
ANSI Store order #: 01274457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2095
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

FBA object:



FBA object plane group:



When the alternative FBA object bitstream structure is employed, the bitstream is decoded by DCT-based FBA object decoding as described in subclause 7.12.2. Otherwise, the bitstream is decoded by the frame-based FBA object decoding. Refer to Table C-1 for a specification of default minimum and maximum values for each FAP and BAP.

6.1.5.2 Facial animation parameter set

The FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. Exaggerated values permit the definition of actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

The FAP set contains two high level parameters visemes and expressions. A viseme is a visual correlate to a phoneme. The viseme parameter allows viseme rendering (without having to express them in terms of other parameters) and enhances the result of other parameters, insuring the correct rendering of visemes. Only static visemes which are clearly distinguished are included in the standard set. Additional visemes may be added in future extensions of the standard. Similarly, the expression parameter allows definition of high level facial expressions. The facial expression parameter values are defined by textual descriptions. To facilitate facial animation, FAPs that can be used together to represent natural expression are grouped together in FAP groups, and can be indirectly addressed by using an expression parameter. The expression parameter allows for a very efficient means of animating faces. In annex C, a list of the FAPs is given, together with the FAP grouping, and the definitions of the facial expressions.

6.1.5.3 Facial animation parameter units

All the parameters involving translational movement are expressed in terms of the *Facial Animation Parameter Units (FAPU)*. These units are defined in order to allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to fractions of distances between some key facial features and are defined in terms of distances between feature points. The fractional units used are chosen to allow enough precision. annex C contains the list of the FAPs and the list of the FDP feature points. For each FAP the list contains the name, a short description, definition of the measurement units, whether the parameter is unidirectional (can have only positive values) or bi-directional, definition of the direction of movement for positive values, group number (for coding of selected groups), FDP subgroup number (annex C) and quantisation step size. FAPs act on FDP feature points in the indicated subgroups. The measurement units are shown in Table 6-1, where the notation 3.1.y represents the y coordinate of the feature point 3.1; also refer to Figure 6-10.

Table 6-1 – Facial Animation Parameter Units

Description		FAPU Value
IRISD0 = 3.1.y – 3.3.y = 3.2.y – 3.4.y	Iris diameter (by definition it is equal to the distance between upper ad lower eyelid) in neutral face	IRISD = IRISD0 / 1024
ES0 = 3.5.x – 3.6.x	Eye separation	ES = ES0 / 1024
ENS0 = 3.5.y – 9.15.y	Eye - nose separation	ENS = ENS0 / 1024
MNS0 = 9.15.y – 2.2.y	Mouth - nose separation	MNS = MNS0 / 1024
MW0 = 8.3.x – 8.4.x	Mouth width	MW = MW0 / 1024
AU	Angle Unit	10^{-5} rad

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... 2005/12/21 08:21:51 PST
Single user licence only. Copying and networking prohibited.

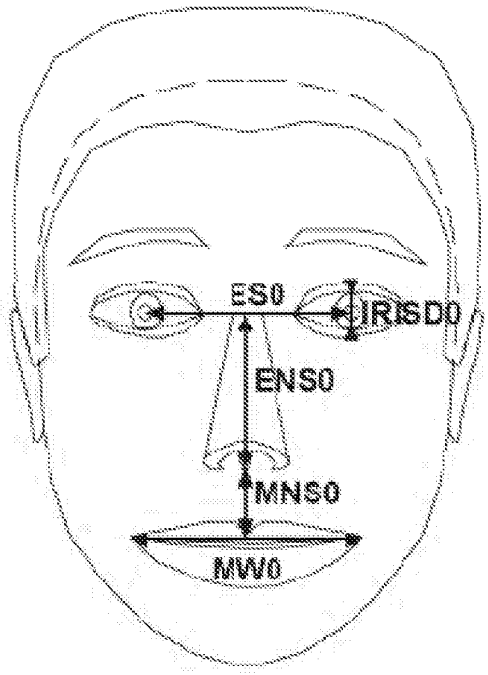


Figure 6-10 -- The Facial Animation Parameter Units

6.1.5.4 Description of a neutral face

At the beginning of a sequence, the face is supposed to be in a neutral position. Zero values of the FAPs correspond to a neutral face. All FAPs are expressed as displacements from the positions defined in the neutral face. The neutral face is defined as follows:

- the coordinate system is right-handed; head axes are parallel to the world axes
- gaze is in direction of Z axis
- all face muscles are relaxed
- eyelids are tangent to the iris
- the pupil is one third of IRISD0
- lips are in contact; the line of the lips is horizontal and at the same height of lip corners
- the mouth is closed and the upper teeth touch the lower ones
- the tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth (feature point 6.1 touching 9.11 in annex C)

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ADAMSON
ANSI Store order #: [illegible] Downloaded: 7/16/2007 1:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2097
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.1.5.5 Facial definition parameter set

The FDPs are used to customize the proprietary face model of the decoder to a particular face or to download a face model along with the information about how to animate it. The definition and description of FDP fields is given in annex C. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs. However, if the decoder does not receive the FDPs, the use of FAPUs ensures that it can still interpret the FAP stream. This insures minimal operation in broadcast or teleconferencing applications. The FDP set is specified in BIFS syntax (see ISO/IEC 14496-1:1999). The FDP node defines the face model to be used at the receiver. Two options are supported:

- calibration information is downloaded so that the proprietary face of the receiver can be configured using facial feature points and optionally a 3D mesh or texture.

- a face model is downloaded with the animation definition of the Facial Animation Parameters. This face model replace the proprietary face model in the receiver.

6.1.5.6 Body animation parameter set

BAP parameters comprise joint angles connecting different body parts. These include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers. The detailed joint list, with the rotation normals, are given in the following subclause. The rotation angles are assumed to be positive in the counterclockwise rotation direction with respect to the rotation normal. The rotation angles are defined as zero in the default posture, as defined below.

Note that the normals of rotation move with the body, and they are fixed with respect to the parent body part. That is to say, the axes of rotation are not aligned with the body or world coordinate system, but move with the body parts.

The hands are capable of performing complicated motions and are included in the body hierarchy. There are totaly 25 degrees of freedom on each hand, assuming that the hand has a standard structure with five fingers.

The unit of rotations (BAPU) is defined as PI/10E-5 radians. The unit of translation BAPs (BAPs HumanoidRoot_tr_vertical, HumanoidRoot_tr_lateral, HumanoidRoot_tr_frontal) is defined in millimeters.

6.1.5.7 Description of the default posture of the body

The default posture is defined by standing posture. This posture is defined as follows: the feet should point to the front direction, the two arms should be placed on the side of the body with the palm of the hands facing inward. This posture also implies that all BAPs have default values as 0.

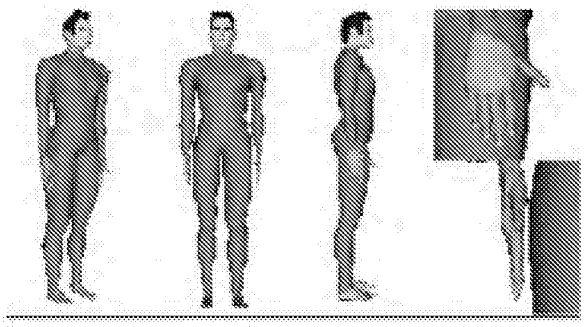


Figure 72 - 5

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS A. YOUNG LIBRARY
ANSI Store order #: 01/74467 Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2098
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.1.5.8 Body Definition Parameter Set

The BDPs are used to customize the proprietary body model of the decoder to a particular body or to download a body model along with the information about how to animate it. The definition and description of BDP fields is given in annex C. The BDPs are normally transmitted once per session, followed by a stream of compressed BAPs. However, if the decoder does not receive the BDPs, model-independence of BAPs ensures that it can still interpret the BAP stream. This insures minimal operation in broadcast or teleconferencing applications. The BDP set is specified in BIFS syntax (see ISO/IEC 14496-1:1999/Amd.1:2000). The BDP node defines the body model to be used at the receiver.

6.1.6 3D Mesh Object

The compressed bitstream for a 3D mesh is composed of a header data block with global information, followed by a sequence of connected component data blocks, each one associated with one connected component of the 3D mesh.

3D Mesh Header	CC Data #1	...	CC Data #nCC

If a 3D mesh is coded in error resilience mode, connected component data blocks are grouped or divided into partitions.

Partition #1	Partition #2	...	Partition #nPT

Additionally, if the 3D mesh is represented in hierarchical mode, the last connected component data block is followed by one or more refinement step data blocks, each one of them representing a Forest Split operation. Forest split can be applied to all the components of a bitstream.

Forest Split #1	Forest Split #2	...	Forest Split #nFS

Each connected component data block is composed of three records, the Vertex Graph record, the Triangle Tree record, and the Triangle Data record.

Vertex Graph	Triangle Tree	Triangle Data

The triangle tree record contains the structure of a triangle spanning tree which links all the triangles of the corresponding connected component forming a simple polygon. The 3D mesh is represented in a triangulated form in the bitstream, which also contains the information necessary to reconstruct the original faces. The vertex graph record contains the information necessary to stitch pairs of boundary edges of the simple polygon to reconstruct the original connectivity, not only within the current connected component, but also to previously decoded connected components. The connectivity information is categorized as *global* information (per connected component) and *local* information (per triangle). The global information is stored in the Vertex Graph and Triangle Tree records. The local information is stored in the Triangle Data record. The triangle data is arranged on a per triangle basis, where the ordering of the triangles is determined by the traversal of the triangle tree.

Data for triangle #1	Data for triangle #2	...	Data for triangle #nT

The data for a given triangle is organized as follows:

marching edge	tid_orientation	polygon_edge	coord	normal	color	texCoord

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ANDREWS
ANSI Store order #OS7445, Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2099
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The *marching_edge*, *fd_orientation* and *polygon_edge* constitute the pes triangle connectivity information. The other fields contain information to reconstruct the vertex coordinates (*coord*) and optionally, normal, color, and texture coordinate (*texCoord*) information.

If the 3D mesh is encoded in hierarchical mode, each Forest Split data block is composed of an optional pre-smoothing data block, an optional post-smoothing data block, a pre-update data block, an optional smoothing constraints data block, a post-update data block, and an optional other-update data bloct.

Pre-smoothing	Post-smoothing	Pre-update	Constraints	Post-update	Other-update

The pre-smoothing data block contains the parameters used to apply a smoothing step as a global predictor for vertex coordinates. The post-smoothing data block contains the parameters used to apply a smoothing step to the vertex coordinates to remove quantisation artifacts. The pe update data block contains the information necessary to update the connectivity, and the property data updates for properties bound per-face and per-corner to the new faces created by the connectivity update. The smoothing constraints data block contains information used to perform the smoothing steps with sharp edge discontinuities and fixed vertices. After the connectivity update is applied, the pre-smoothing operation specified by the parameters stored in the pre-smoothing data block is applied as a global predictor for the vertex coordinates. The post update data block contains tee loop vertex coordinate updates, with respect to the vertex coordinates predicted by the pre-smoothing step, if applied; normal, color, and texture coordinate updates for properties bound per-vertex to tree loop vertices; normal, and color updates for properties bound per-face to tree loop faces; and normal, color, and texture coordinate updates for properties bound per-corner to tree loop corners. The other-update data block contains vertex coordinate updates and property updates for all the vertices, faces, and corners not updated by data included in the post-update data block.

6.2 Visual bitstream syntax

6.2.1 Start codes

Start codes are specific bit patterns that do not otherwise occur in the video stream.

Each start code consists of a start code prefix followed by a start code value. The start code prefix is a string of twenty three bits with the value zero followed by a single bit with the value one. The start code prefix is thus the bit string '0000 0000 0000 0000 0000 0001'.

The start code value is an eight bit integer which identifies the type of start code. Many types of start code have just one start code value. However video_object_start_code and video_object_layer_start_code are represented by many start code values.

All start codes shall be byte aligned. This shall be achieved by first inserting a bit with the value zero and then, if necessary, inserting bits with the value one before the start code prefix such that the first bit of the start code prefix is the first (most significant) bit of a byte. For stuffing of 1 to 8 bits, the codewords are as follows in Table 6-2.

Table 6-2-- Stuffing codewords

Bits to be stuffed	Stuffing Codeword
1	0
2	01
3	011
4	0111
5	01111
6	011111
7	0111111
8	01111111

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #X01445 Downloaded 7/18/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2100
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Table 6-3 defines the start code values for all start codes used in the visual bitstream.

Table 6-3 — Start code values

name	start code value (hexadecimal)
video_object_start_code	00 through 1F
video_object_layer_start_code	20 through 2F
reserved	30 through AF
visual_object_sequence_start_code	B0
visual_object_sequence_end_code	B1
user_data_start_code	B2
group_of_vop_start_code	B3
video_session_error_code	B4
visual_object_start_code	B5
vop_start_code	B6
reserved	B7-B9
fba_object_start_code	BA
fba_object_plane_start_code	BB
mesh_object_start_code	BC
mesh_object_plane_start_code	BD
still_texture_object_start_code	BE
texture_spatial_layer_start_code	BF
texture_snr_layer_start_code	C0
texture_tile_start_code	C1
texture_shape_layer_start_code	C2
stuffing_start_code	C3
reserved	C4-C5
System start codes (see note)	C6 through FF
NOTE System start codes are defined in ISO/IEC 14496-1:1999	

The use of the start codes is defined in the following syntax description with the exception of the video_session_error_code. The video_session_error_code has been allocated for use by a media interface to indicate where uncorrectable errors have been detected.

This syntax for visual bitstreams defines two types of information:

1. Configuration information

a. Global configuration information, referring to the whole group of visual objects that will be simultaneously decoded and composited by a decoder (VisualObjectSequence()).
b. Object configuration information, referring to a single visual object (VO). This is associated with VisualObject().
c. Object ayer configuration information, referring to a singe layer of a single visual object (VOL) VideoObjectLayer().

2. Elementary stream data, containing the data for a single layer of a visual object.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PROQUEST ADVANCED
ANSI Store order #: 0127445 Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2101
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

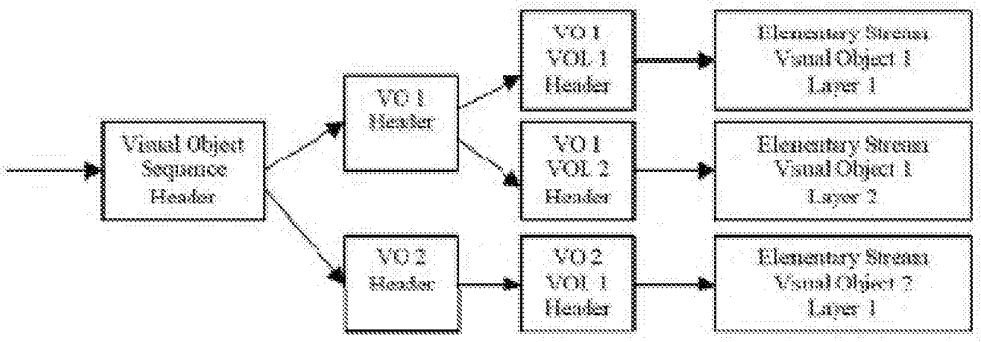


Figure 6-11 -- Example Visual Information – Logical Structure

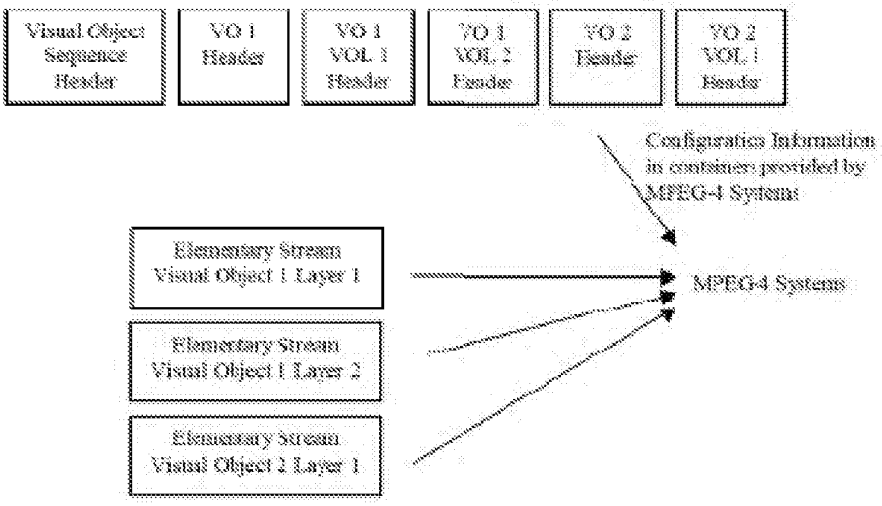


Figure 6-12 -- Example Visual Bitstream – Separate Configuration Information/ Elementary Stream

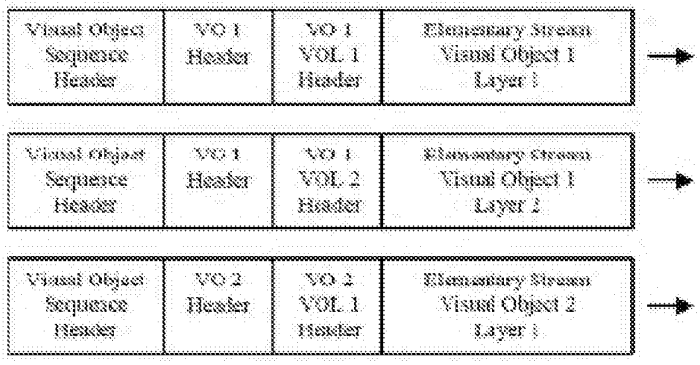


Figure 6-13 -- Example Visual Bitstream – Combined Configuration Information / Elementary Stream

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG V. LEISHMAN
ANSI Store order #1274457, Downloaded: 7/18/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2102
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The following functions are entry points for elementary streams, and entry into these functions defines the breakpoint between configuration information and elementary streams:

1. Group_of_VideoObjectPlane(),
2. VideoObjectPlane(),
3. video_plane_with_short_header(),
4. MeshObject(),
5. fba_object().

For still texture objects, configuration information ends and elementary stream data begins in StillTextureObject() immediately before the first call to wavelet_dc_decode(), as indicated by the comment in subclause 6.2.8.

There is no overlap of syntax between configuration information and elementary streams.

The configuration information contains all data that is not part of an elementary stream, including that defined by VisualObjectSequence(), VisualObject() and VideoObjectLayer().

This part of ISO/IEC 14496 does not provide for the multiplexing of multiple elementary streams into a single bitstream. One visual bitstream contains exactly one elementary stream, which describes one layer of one visual object. A visual decoder must conceptually have a separate entry port for each layer of each object to be decoded.

Visual objects coded in accordance with this Part may be carried within a Systems bitstream as defined by ISO/IEC 14496-1:1999. The coded visual objects may also be free standing or carried within other types of systems. Configuration information may be carried separately from or combined with elementary stream data:

1. *Separate Configuration / Elementary Streams (e.g. inside ISO/IEC 14496-1 Bitstreams)*

When coded visual objects are carried within a Systems bitstream defined by ISO/IEC 14496-1:1999, configuration information and elementary stream data are always carried separately. Configuration information and elementary streams follow the syntax below, subject to the break points between them defined above. The Systems specification ISO/IEC 14496-1:1999 defines containers that are used to carry Visual Object Sequence, Visual Object and Video Object Layer configuration information. For video objects one container is used for each layer for each object. This container carries a Visual Object Sequence header, a Visual Object header and a Video Object Layer header. For other types of visual objects, one container per visual object is used. This container carries a Visual Object Sequence header and a Visual Object header. The Visual Object Sequence Header must be identical for all visual streams input simultaneously to a decoder. The Visual Object Headers for each layer of a multilayer object must be identical.

2. *Combined Configuration / Elementary Streams*

The elementary stream data associated with a single layer may be wrapped in configuration information defined in accordance with the syntax below. A visual bitstream may contain at most one instance of each of VisualObjectSequence(), VisualObject() and VideoObjectLayer(), with the exception of repetition of the Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header as described below. The Visual Object Sequence Header must be identical for all visual streams input simultaneously to a decoder. The Visual Object Headers for each layer of a multilayer object must be identical.

The Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header may be repeated in a single visual bitstream. Repeating these headers enables random access into the visual bitstream and recovery of these headers when the original headers are corrupted by errors. This header repetition is used only when visual_object_type in the Visual Object Header indicates that visual object type is video. (i.e. visual_object_type=="video ID") All of the data elements in the Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header repeated in a visual bitstream shall have the same value as in the original headers, except that first_half_vbv_occupancy and latter_half_vbv_occupancy may be changed to specify the VBV occupancy just before the removal of the first VOP following the repeated Video Object Layer Header.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
ANSI Store order #0378487 Downloaded: 7/19/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2103
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.2 Visual Object Sequence and Visual Object

VisualObjectSequence() {	No. of bits	Mnemonic
do {		
visual_object_sequence_start_code	32	bslbf
profile_and_level_indication	8	uimsbf
while (next_bits()== user_data_start_code){		
user_data()		
}		
VisualObject()		
} while (next_bits() != visual_object_sequence_end_code)		
visual_object_sequence_end_code	32	bslbf
}		

VisualObject() {	No. of bits	Mnemonic
visual_object_start_code	32	bslbf
is_visual_object_identifier	1	uimsbf
if (is_visual_object_identifier) {		
visual_object_verid	4	uimsbf
visual_object_priority	3	uimsbf
}		
visual_object_type	4	uimsbf
if (visual_object_type == "video ID" || visual_object_type == "still texture ID") {		
video_signal_type()		
}		
next_start_code()		
while (next_bits()== user_data_start_code){		
user_data()		
}		
if (visual_object_type == "video ID") {		
video_object_start_code	32	bslbf
VideoObjectLayer()		
}		
else if (visual_object_type == "still texture ID") {		
StillTextureObject()		
}		
else if (visual_object_type == "mesh ID") {		
MeshObject()		
}		
else if visual_object_type == "FBA ID") {		
FBAObject()		
}		
else if visual_object_type == "3D mesh ID") {		
3D_Mesh_Object()		
}		
if (next_xits(!= "0000 0000 0000 0000 0000 0001")		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #1274457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2104
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

next_start_code()		
}		

video_signal_type() {	No. of bits	Mnemonic
video_signal_type	1	bslbf
if (video_signal_type) {		
video_format	3	uimsbf
video_range	1	bslbf
colour_description	1	bslbf
if (colour_description) {		
colour_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coefficients	8	uimsbf
}		
}		
}		

6.2.2.1 User data

user_data() {	No. of bits	Mnemonic
user_data_start_code	32	bslbf
while(next_bits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	uimsbf
}		
}		

6.2.3 Video Object Layer

VideoObjectLayer() {	No. of bits	Mnemonic
if(next_bits() == video_object_layer_start_code) {		
short_video_header = 0		
video_object_layer_start_code	32	bslbf
random_accessible_vol	1	bslbf
video_object_type_indication	8	uimsbf
is_object_layer_identifier	1	uimsbf
if (is_object_layer_identifier) {		
video_object_layer_verid	4	uimsbf
video_object_layer_priority	3	uimsbf
}		
aspect_ratio_info	4	uimsbf
if(aspect_ratio_info =="extended_PAR") {		
par_width	8	uimsbf
par_height	8	uimsbf
}		
vol_control_parameters	1	bslbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC / ADAMMAN
ANSI Store order #: [illegible] Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

if (vol_control_parameters) {		
chroma_format	2	uimsbf
low_delay	1	uimsbf
vbv_parameters	1	blsbf
if (vbv_parameters) {		
first_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
latter_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
first_half_vbv_buffer_size	15	uimsbf
marker_bit	1	bslbf
latter_half_vbv_buffer_size	3	uimsbf
first_half_vbv_occupancy	11	uimsbf
marker_bit	1	blsbf
latter_half_vbv_occupancy	15	uimsbf
marker_bit	1	blsbf
}		
}		
video_object_layer_shape	2	uimsbf
if (video_object_layer_shape == "grayscale" && video_object_layer_verid != '0001')		
video_object_layer_shape_extension	4	uimsbf
marker_bit	1	bslbf
vop_time_increment_resolution	16	uimsbf
marker_bit	1	bslbf
fixed_vop_rate	1	bslbf
if (fixed_vop_rate)		
fixed_vop_time_increment	1-16	uimsbf
if (video_object_layer_shape != "binary only") {		
if (video_object_layer_shape == "rectangular") {		
marker_bit	1	bslbf
video_object_layer_width	13	uimsbf
marker_bit	1	bslbf
video_object_layer_height	13	uimsbf
marker_bit	1	bslbf
}		
interlaced	1	bslbf
obmc_disable	1	bslbf
if (video_object_layer_verid == '0001')		
sprite_enable	1	bslbf
else		
sprite_enable	2	uimsbf
if (sprite_enable== "static" || sprite_enable == "GMC") {		
if (sprite_enable != "GMC") {		
sprite_width	13	uimsbf
marker_bit	1	bslbf
sprite_height	13	uimsbf
marker_bit	1	bslbf
sprite_left_coordinate	13	simsbf
marker_bit	1	bslbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS V. SMITH
ANSI Store order #: 07/44[illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2106
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

sprite_top_coordinate	13	simsbf
marker_bit	1	bslbf
}		
no_of_sprite_warping_points	6	uimsbf
sprite_warping_accuracy	2	uimsbf
sprite_brightness_change	1	bslbf
if (sprite_enable != "GMC")		
low_latency_sprite_enable	1	bslbf
}		
if (video_object_layer_verid != '0001' && video_object_layer_shape != "rectangular")		
sadct_disable	1	bslbf
not_8_bit	1	bslbf
if (not_8_bit) {		
quant_precision	4	uimsbf
bits_per_pixel	4	uimsbf
}		
if (video_object_layer_shape=="grayscale") {		
no_gray_quant_update	1	bslbf
composition_method	1	bslbf
linear_composition	1	bslbf
}		
quant_type	1	bslbf
if (quant_type) {		
load_intra_quant_mat	1	bslbf
if (load_intra_quant_mat)		
intra_quant_mat	8*[2-64]	uimsbf
load_nonintra_quant_mat	1	bslbf
if (load_nonintra_quant_mat)		
nonintra_quant_mat	8*[2-64]	uimsbf
if(video_object_layer_shape=="grayscale") {		
for(i=0; i<aux_comp_count; i++) {		
load_intra_quant_mat_grayscale	1	bslbf
if(load_intra_quant_mat_grayscale)		
intra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
load_nonintra_quant_mat_grayscale	1	bslbf
if(load_nonintra_quant_mat_grayscale)		
nonintra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
}		
}		
}		
if (video_object_layer_verid != '0001')		
quarter_sample	1	bslbf
complexity_estimation_disable	1	bslbf
if (!complexity_estimation_disable)		
define_vop_complexity_estimation_header()		
resync_marker_disable	1	bslbf
data_partitioned	1	bslbf
if(data_partitioned)		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCED
ANSI Store order #X12445 Downloaded 7/16/2007 3:06:21 PM ET
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2107
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

reversible_vlc	1	bslbf
if(video_object_layer_verid != '0001') {		
newpred_enable	1	bslbf
if (newpred_enable) {		
requested_upstream_message_type	2	uimsbf
newpred_segment_type	1	bslbf
}		
reduced_resolution_vop_enable	1	bslbf
}		
scalability	1	bslbf
if (scalability) {		
hierarchy_type	1	bslbf
ref_layer_id	4	uimsbf
ref_layer_sampling_direc	1	bslbf
hor_sampling_factor_n	5	uimsbf
hor_sampling_factor_m	5	uimsbf
vert_sampling_factor_n	5	uimsbf
vert_sampling_factor_m	5	uimsbf
enhancement_type	1	bslbf
if(video_object_layer == "binary" && hierarchy_type=='0') {		
use_ref_shape	1	bslbf
use_ref_texture	1	bslbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
}		
else{		
if(video_object_layer_verid !='0001') {		
scalability	1	bslbf
if(scalability) {		
ref_layer_id	4	uimsbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
resync_marker_disable	1	bslbf
}		
next_start_code()		
while (next_bits()== user_data_start_code){		
user_data()		
}		
if (sprite_enable == "static" && !low_latency_sprite_enable)		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: [illegible] Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

VideoObjectPlane()		
do {		
if (next_bits() == group_of_vop_start_code)		
Group_of_VideoObjectPlane()		
VideoObjectPlane()		
if ((preceding_vop_coding_type == "B" || preceding_vop_coding_type == "S" || video_object_layer_shape != "rectangular") && next_bits() == stuffing_start_code) {		
stuffing_start_code	32	bslbf
while (next_bits() != '0000 0000 0000 0000 0000 0001')		
stuffing_byte	8	bslbf
}		
} while ((next_bits() == group_of_vop_start_code) || (next_bits() == vop_start_code))		
} else {		
short_video_header = 1		
do {		
video_plane_with_short_header()		
} while(next_bits() == short_video_start_marker)		
}		
}		

NOTE — preceding_vop_coding_type has the same value as vop_coding_type in the immediately preceding VideoObjectPlane() in the decoding order.

define_vop_complexity_estimation_header() {	No. of bits	Mnemonic
estimation_method	2	uimsbf
if (estimation_method =='00' || estimation_method == '01') {		
shape_complexity_estimation_disable	1	
if (!shape_complexity_estimation_disable) {		bslbf
opaque	1	bslbf
transparent	1	bslbf
intra_cae	1	bslbf
inter_cae	1	bslbf
no_update	1	bslbf
upsampling	1	bslbf
}		
texture_complexity_estimation_set_1_disable	1	bslbf
if (!texture_complexity_estimation_set_1_disable) {		
intra_blocks	1	bslbf
inter_blocks	1	bslbf
inter4v_blocks	1	bslbf
not_coded_blocks	1	bslbf
}		
marker_bit	1	bslbf
texture_complexity_estimation_set_2_disable	1	bslbf
if (!texture_complexity_estimation_set_2_disable) {		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODIAK T. ADAMIAN
ANSI Store order #1274457 Downloaded 7/18/2007 2:06:21 PM ET
Single user license only. Copying and networking prohibited.

fict_coefs	1	bslbf
fict_lines	1	bslbf
vlc_symbols	1	bslbf
vlc_bits	1	bslbf
}		
motion_compensation_complexity_disable	1	bslbf
if (!motion_compensation_complexity_disable) {		
apm	1	bslbf
apm	1	bslbf
interpolate_mc_q	1	bslbf
forw_back_mc_q	1	bslbf
halfpel2	1	bslbf
halfpel4	1	bslbf
}		
marker_bit	1	bslbf
if(estimation_method == '01') {		
version2_complexity_estimation_disable	1	bslbf
if (!version2_complexity_estimation_disable) {		
sadct	1	bslbf
quarterpel	1	bslbf
}		
}		
}		
}		

6.2.4 Group of Video Object Plane

Group_of_VideoObjectPlane() {	No. of bits	Mnemonic
group_of_vop_start_code	32	bslbf
time_code	18	
closed_gov	1	bslbf
broken_link	1	bslbf
next_start_code()		
while (next_bits()== user_data_start_code){		
user_data()		
}		
}		

6.2.5 Video Object Plane and Video Plane with Short Header

VideoObjectPlane() {	No. of bits	Mnemonic
vop_start_code	32	bslbf
vop_coding_type	2	uimsbf
do {		
modulo_time_base	1	bslbf
} while (modulo_time_base != '0')		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #1274457 Downloaded 7/18/2007 2:06:21 PM ET
Single user licensee only. Copying and networking prohibited.

marker_bit	1	bslbf
vop_time_increment	1-16	uimsbf
marker_bit	1	bslbf
vop_coded	1	bslbf
if (vop_coded == '0') {		
next_start_code()		
return()		
}		
if (newpred_enable) {		
vop_id	4-15	uimsbf
vop_id_for_prediction_indication	1	bslbf
if(vop_id_for_prediction_indication)		
vop_id_for_prediction	4-15	uimsbf
marker_bit	1	bslbf
}		
if ((video_object_layer_shape != "binary only") && (vop_coding_type == "P" || (vop_coding_type == "S" && sprite_enable == "GMC")))		
vop_rounding_type	1	bslbf
if ((reduced_resolution_vop_enable) && (video_object_layer_shape == "rectangular") && ((vop_coding_type == "P") || (vop_coding_type == "I")))		
vop_reduced_resolution	1	bslbf
if (video_object_layer_shape != "rectangular") {		
if(sprite_enable == "static" && vop_coding_type == "I") {		
vop_width	13	uimsbf
marker_bit	1	bslbf
vop_height	13	uimsbf
marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
}		
if ((video_object_layer_shape != " binary only") && scalability && enhancement_type)		
background_composition	1	bslbf
change_conv_ratio_disable	1	bslbf
vop_constant_alpha	1	bslbf
if(vop_constant_alpha)		
vop_constant_alpha_value	8	bslbf
}		
if (video_object_layer_shape != "binary only")		
if(!complexity_estimation_disable)		
read_vop_complexity_estimation_header()		
if (video_object_layer_shape != "binary only") {		
intra_dc_vlc_thr	3	uimsbf
if(interlaced) {		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC, ADAMANS
ANSI Store order #X07445, Downloaded 7/18/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2111
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

top_field_first	1	bslbf
alternate_vertical_scan_flag	1	bslbf
}		
}		
if ((sprite_enable =="static" || sprite_enable=="GMC") &&		
vop_coding_type == "S") {		
if (no_of_sprite_warping_points > 0)		
sprite_trajectory()		
if (sprite_brightness_change)		
brightness_change_factor()		
if(sprite_enable == "static") {		
if (sprite_transmit_mode != "stop"		
&& low_latency_sprite_enable) {		
do {		
sprite_transmit_mode	2	uimsbf
if ((sprite_transmit_mode == "piece") ||		
(sprite_transmit_mode == "update"))		
decode_sprite_piece()		
} while (sprite_transmit_mode != "stop" &&		
sprite_transmit_mode != "pause")		
}		
next_start_code()		
return()		
}		
}		
if (video_object_layer_shape != "binary only") {		
vop_quant	3-9	uimsbf
if(video_object_layer_shape=="grayscale")		
for(i=0; i<aux_comp_count; i++)		
vop_alpha_quant[i]	6	uimsbf
if (vop_coding_type != "I")		
vop_fcode_forward	3	uimsbf
if (vop_coding_type == "B")		
vop_fcode_backward	3	uimsbf
if (!scalability) {		
if (video_object_layer_shape != "rectangular"		
&& vop_coding_type != "I")		
vop_shape_coding_type	1	bslbf
motion_shape_texture()		
while (nextbits_bytealigned() == resync_marker) {		
video_packet_header()		
motion_shape_texture()		
}		
}		
else{		
if (enhancement_type) {		
load_backward_shape	1	bslbf
if (load_backward_shape) {		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #: ... Downloaded: ...
Single user license only. Copying and networking prohibited.

	backward_shape_width	13	uimsbf
	marker_bit	1	bslbf
	backward_shape_height	13	uimsbf
	marker_bit	1	bslbf
	backward_shape_horizontal_mc_spatial_ref	13	simsbf
	marker_bit	1	bslbf
	backward_shape_vertical_mc_spatial_ref	13	simsbf
	backward_shape()		
	load_forward_shape	1	bslbf
	if (load_forward_shape){		
	forward_shape_width	13	uimsbf
	marker_bit	1	bslbf
	forward_shape_height	13	uimsbf
	marker_bit	1	bslbf
	forward_shape_horizontal_mc_spatial_ref	13	simsbf
	marker_bit	1	bslbf
	forward_shape_vertical_mc_spatial_ref	13	simsbf
	forward_shape()		
	}		
	}		
	}		
	ref_select_code	2	uimsbf
	combined_motion_shape_texture()		
}			
}			
else {			
combined_motion_shape_texture()			
while (nextbits_bytealigned() == resync_marker) {			
video_packet_header()			
combined_motion_shape_texture()			
}			
}			
next_start_code()			
}			

6.2.5.1 Complexity Estimation Header

read_vop_complexity_estimation_header() {		No. of bits	Mnemonic
if (estimation_method=="00") {			
if (vop_coding_type=="I") {			
if (opaque)	dcecs_opaque	8	uimsbf
if (transparent)	dcecs_transparent	8	uimsbf
if (intra_cae)	dcecs_intra_cae	8	uimsbf
if (inter_cae)	dcecs_inter_cae	8	uimsbf
if (no_update)	dcecs_no_update	8	uimsbf
if (upsampling)	dcecs_upsampling	8	uimsbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ACQUISITIONS
ANSI Store order #: 1127445 Downloaded: 7/16/2007 3:08:21 PM ET
Single user licensee only. Copying and networking prohibited.

f (intra_blocks)	dcecs_intra_blocks	8	uimsbf
f (not_coded_blocks)	dcecs_not_coded_blocks	8	uimsbf
f (dct_coefs)	dcecs_dct_coefs	8	uimsbf
f (dct_lines)	dcecs_dct_lines	8	uimsbf
f (vlc_symbols)	dcecs_vlc_symbols	8	uimsbf
f (vlc_bits)	dcecs_vlc_bits	4	uimsbf
f (sadct)	dcecs_sadct	8	uimsbf
}			
if (vop_coding_type=="P") {			
f (opaque)	dcecs_opaque	8	uimsbf
f (transparent)	dcecs_transparent	8	uimsbf
f (intra_cae)	dcecs_intra_cae	8	uimsbf
f (inter_cae)	dcecs_inter_cae	8	uimsbf
f (no_update)	dcecs_no_update	8	uimsbf
f (upsampling)	dcecs_upsampling	8	uimsbf
f (intra)	dcecs_intra_blocks	8	uimsbf
f (not_coded)	dcecs_not_coded_blocks	8	uimsbf
f (dct_coefs)	dcecs_dct_coefs	8	uimsbf
f (dct_lines)	dcecs_dct_lines	8	uimsbf
f (vlc_symbols)	dcecs_vlc_symbols	8	uimsbf
f (vlc_bits)	dcecs_vlc_bits	4	uimsbf
f (inter_blocks)	dcecs_inter_blocks	8	uimsbf
f (inter4v_blocks)	dcecs_inter4v_blocks	8	uimsbf
f (apm)	dcecs_apm	8	uimsbf
f (npm)	dcecs_npm	8	uimsbf
f (forw_back_mc_q)	dcecs_forw_back_mc_q	8	uimsbf
f (halfpel2)	dcecs_halfpel2	8	uimsbf
f (halfpel4)	dcecs_halfpel4	8	uimsbf
f (sadct)	dcecs_sadct	8	uimsbf
f (quarterpel)	dcecs_quarterpel	8	uimsbf
}			
if (vop_coding_type=="B") {			
f (opaque)	dcecs_opaque	8	uimsbf
f (transparent)	dcecs_transparent	8	uimsbf
f (intra_cae)	dcecs_intra_cae	8	uimsbf
f (inter_cae)	dcecs_inter_cae	8	uimsbf
f (no_update)	dcecs_no_update	8	uimsbf
f (upsampling)	dcecs_upsampling	8	uimsbf
f (intra_blocks)	dcecs_intra_blocks	8	uimsbf
f (not_coded_blocks)	dcecs_not_coded_blocks	8	uimsbf
f (dct_coefs)	dcecs_dct_coefs	8	uimsbf
f (dct_lines)	dcecs_dct_lines	8	uimsbf
f (vlc_symbols)	dcecs_vlc_symbols	8	uimsbf
f (vlc_bits)	dcecs_vlc_bits	4	uimsbf
f (inter_blocks)	dcecs_inter_blocks	8	uimsbf
f (inter4v_blocks)	dcecs_inter4v_blocks	8	uimsbf
f (apm)	dcecs_apm	8	uimsbf
f (npm)	dcecs_npm	8	uimsbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #: ... Downloaded: ... Single user license only. Copying and networking prohibited.

if (forw_back_mc_q) dcecs_forw_back_mc_q	3	uimsbf
if (halfpel2) dcecs_halfpel2	3	uimsbf
if (halfpel4) dcecs_halfpel4	3	uimsbf
if (interpolate_mc_q) dcecs_interpolate_mc_q	3	uimsbf
if (sadct) dcecs_sadct	3	uimsbf
if (quarterpel) dcecs_quarterpel	3	uimsbf
}		
if (vop_coding_type=='S'&& sprite_enable =='static') {		
if (intra_blocks) dcecs_intra_blocks	3	uimsbf
if (not_coded_blocks) dcecs_not_coded_blocks	3	uimsbf
if (dct_coefs) dcecs_dct_coefs	3	uimsbf
if (dct_lines) dcecs_dct_lines	3	uimsbf
if (vlc_symbols) dcecs_vlc_symbols	3	uimsbf
if (vlc_bits) dcecs_vlc_bits	3	uimsbf
if (inter_blocks) dcecs_inter_blocks	3	uimsbf
if (inter4v_blocks) dcecs_inter4v_blocks	3	uimsbf
if (apm) dcecs_apm	3	uimsbf
if (npm) dcecs_npm	3	uimsbf
if (forw_back_mc_q) dcecs_forw_back_mc_q	3	uimsbf
if (halfpel2) dcecs_halfpel2	3	uimsbf
if (halfpel4) dcecs_halfpel4	3	uimsbf
if (interpolate_mc_q) dcecs_interpolate_mc_q	3	uimsbf
}		
}		
}		

6.2.5.2 Video Plane with Short Header

video_plane_with_short_header() {	No. of bits	Mnemonic
short_video_start_marker	22	bslbf
temporal_reference	8	uimsbf
marker_bit	1	bslbf
zero_bit	1	bslbf
split_screen_indicator	1	bslbf
document_camera_indicator	1	bslbf
full_picture_freeze_release	1	bslbf
source_format	3	bslbf
picture_coding_type	1	bslbf
four_reserved_zero_bits	4	bslbf
vop_quant	5	uimsbf
zero_bit	1	bslbf
do{		
pei	1	bslbf
if (pei == "1")		
psupp	8	bslbf
} while (pei == "1")		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC / ADAMANS
ANSI Store order #: 1127445 Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

gob_number = 0		
for(i=0; i<num_gobs_in_vop; i++)		
gob_layer()		
if(next_bits() == short_video_end_marker)		
short_video_end_marker	22	uimsbf
while(!bytealigned())		
zero_bit	1	bslbf
}		

gob_layer() {	No. of bits	Mnemonic
gob_header_empty = 1		
if(gob_number != 0) {		
if (next_bits() == gob_resync_marker) {		
gob_header_empty = 0		
gob_resync_marker	17	bslbf
gob_number	5	uimsbf
gob_frame_id	2	bslbf
quant_scale	5	uimsbf
}		
}		
for(i=0; i<num_macroblocks_in_gob; i++)		
macroblock()		
if(next_bits() != gob_resync_marker && nextbits_bytealigned() == gob_resync_marker)		
while(!bytealigned())		
zero_bit	1	bslbf
gob_number++		
}		

video_packet_header() {	No. of bits	Mnemonic
next_resync_marker()		
resync_marker	17+23	uimsbf
if (video_object_layer_shape != "rectangular") {		
header_extension_code	1	bslbf
if (header_extension_code && !sprite_enable = "static" && vop_coding_type == "I")) {		
vop_width	13	uimsbf
marker_bit	1	bslbf
vop_height	13	uimsbf
marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
}		
}		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG Y
ANSI Store order #: 1247457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2116
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

macroblock_number	1-14	vlcbf
if (video_object_layer_shape != "binary only")		
quant_scale	5	uimsbf
if (video_object_layer_shape == "rectangular")		
header_extension_code	1	bslbf
if (header_extension_code) {		
do {		
modulo_time_base	1	bslbf
} while (modulo_time_base != '0')		
marker_bit	1	bslbf
vop_time_increment	1-16	bslbf
marker_bit	1	bslbf
vop_coding_type	2	uimsbf
if (video_object_layer_shape != "rectangular"){		
change_conv_ratio_disable	1	bslbf
if (vop_coding_type != "I")		
vop_shape_coding_type	1	bslbf
}		
if (video_object_layer_shape != "binary only"){		
intra_dc_vlc_thr	3	uimsbf
if (sprite_enable == "GMC" && vop_coding_type == "S" && no_of_sprite_warping_points > 0)		
sprite_trajectory()		
if ((reduced_resolution_vop_enable) && (video_object_layer_shape == "rectangular") && ((vop_coding_type == "P") || (vop_coding_type == "I")))		
vop_reduced_resolution	1	bslbf
if (vop_coding_type != "I")		
vop_fcode_forward	3	uimsbf
if (vop_coding_type == "B")		
vop_fcode_backward	3	uimsbf
}		
}		
if (newpred_enable) {		
vop_id	4-15	uimsbf
vop_id_for_prediction_indication	1	bslbf
if (vop_id_for_prediction_indication)		
vop_id_for_prediction	4-15	uimsbf
marker_bit	1	bslbf
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC / ADAMANT
ANSI Store order #1121445 Downloaded 7/18/2007 3:06:21 PM ET
Single user licensee only. Copying and networking prohibited.

6.2.5.3 Motion Shape Texture

motion_shape_texture() {	No. of bits	Mnemonic
if (data_partitioned)		
data_partitioned_motion_shape_texture()		
else		
combined_motion_shape_texture()		
}		

combined_motion_shape_texture() {	No. of bits	Mnemonic
do{		
macroblock()		
} while (nextbits_bytealigned() != resync_marker && nextbits_bytealigned() = '000 0000 0000 0000 0000 0000')		
}		

data_partitioned_motion_shape_texture() {	No. of bits	Mnemonic
if (vop_coding_type == "I") {		
data_partitioned_i_vop()		
} else if (vop_coding_type == "P" || (vop_coding_type == "S" && sprite_enable == "GMC")) {		
data_partitioned_p_vop()		
} else if (vop_coding_type == "B") {		
combined_motion_shape_texture()		
}		
NOTE: Data partitioning is not supported in B-VOPs.		

data_partitioned_i_vop() {	No. of bits	Mnemonic
do{		
if (video_object_layer_shape != "rectangular"){		
bab_type	1-3	vlclbf
if (bab_type >= 4) {		
if (!change_conv_rate_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_code()		
}		
}		
if (!transparent_mb()) {		
if (video_object_layer_shape != "rectangle"){		
do {		
mcbpc	1-9	vlclbf
} while (derived_mb_type == "stuffing")		
} else {		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... (ACCESS)
ANSI Store order #: ... Downloaded: ... Single user license only. Copying and networking prohibited.

mcbpc	1-9	vlclbf
if (derived_mb_type == "stuffing")		
continue		
}		
if (mb_type == 4)		
dquant	2	bslbf
if (use_intra_dc_vlc) {		
for (j = 0; j < 4; j++) {		
if (!transparent_block(j)) {		
dct_dc_size_luminance	2-11	vlclbf
if (dct_dc_size_luminance > 0)		
dct_dc_differential	1-12	vlclbf
if (dct_dc_size_luminance > 8)		
marker_bit	1	bslbf
}		
}		
for (j = 0; j < 2; j++) {		
dct_dc_size_chrominance	2-12	vlclbf
if (dct_dc_size_chrominance > 0)		
dct_dc_differential	1-12	vlclbf
if (dct_dc_size_chrominance > 8)		
marker_bit	1	bslbf
}		
}		
}		
} while (next_bits() != dc_marker)		
dc_marker /* 110 1011 0000 0000 0001 */	19	bslbf
for (i = 0; i < mb_in_video_packet; i++) {		
if (!transparent_mb()) {		
ac_pred_flag	1	bslbf
cbpy	1-6	vlclbf
}		
}		
for (i = 0; i < mb_in_video_packet; i++) {		
if (!transparent_mb()) {		
for (j = 0; j < block_count; j++)		
block()		
}		
}		
}		

NOTE 1: The value of mb_in_video_packet is the number of macroblocks in a video packet. The count of stuffing macroblocks is not included in this value.

NOTE 2: The value of block_count is 6 in the 4:2:0 format.

NOTE 3: The value of alpha_block_count is 4.

© ISO/IEC 2001 – All rights reserved

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/PODIA / ADVANCED ANSI Store order #X274457 Downloaded: 7/18/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2119
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

data_partitioned_p_vop() {	No. of bits	Mnemonic
do{		
if (video_object_layer_shape != "rectangular"){		
bab_type	1-7	vlclbf
if ((bab_type == 1) || (bab_type == 6)) {		
mvds_x	1-13	vlclbf
mvds_y	1-13	vlclbf
}		
if (bab_type >= 4) {		
if (!change_conv_rate_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_code()		
}		
}		
if (!transparent_mb()) {		
if (video_object_layer_shape != "rectangle"){		
do {		
not_coded	1	bslbf
if (!not_coded)		
mcbpc	1-9	vlclbf
} while (!not_coded && derived_mb_type != "stuffing")		
} else {		
not_coded	1	bslbf
if (!not_coded) {		
mcbpc	1-9	vlclbf
if (derived_mb_type == "stuffing")		
continue		
}		
}		
if (!not_coded) {		
if (sprite_enable == "GMC" && vop_coding_type == "S" && derived_mb_type < 2)		
mcsel	1	bslbf
if ((!(sprite_enable == "GMC" && vop_coding_type == "S" && mcsel) && derived_mb_type < 2) || derived_mb_type == 2)		
motion_coding("forward", derived_mb_type)		
}		
}		
} while (next_bits() != motion_marker)		
motion_marker /* 1 1111 0000 0000 0001 */	17	bslbf
for (i = 0; i < mb_in_video_packet; i++) {		
if (!transparent_mb()) {		
if (!not_coded){		
if (derived_mb_type >= 3)		
ac_pred_flag	1	bslbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUNAEVA
ANSI Store order #: 0127445 Downloaded: 7/18/2007 2:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2120
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

skipy	1-6	vlclbf
if (derived_mb_type == 1 || derived_mb_type == 4)		
dquant	2	bslbf
if (derived_mb_type >= 3 && use_intra_dc_vlc) {		
for (j = 0; j < 4; j++) {		
if (!transparent_block(j)) {		
dct_dc_size_luminance	2-11	vlclbf
if (dct_dc_size_luminance > 0)		
dct_dc_differential	1-12	vlclbf
if (dct_dc_size_luminance > 8)		
marker_bit	1	bslbf
}		
}		
for (j = 0; j < 2; j++) {		
dct_dc_size_chrominance	2-12	vlclbf
if (dct_dc_size_chrominance > 0)		
dct_dc_differential	1-12	vlclbf
if (dct_dc_size_chrominance > 8)		
marker_bit	1	bslbf
}		
}		
}		
}		
for (i = 0; i < mb_in_video_packet; i++) {		
if (!transparent_mb()) {		
if (!not_coded) {		
for (j = 0; j < block_count; j++)		
block(j)		
}		
}		
}		
}		

NOTE 1: The value of mb_in_video_packet is the number of macroblocks in a video packet. The count of stuffing macroblocks is not included in this value.

NOTE 2: The value of block_count is 6 in the 4:2:0 format.

NOTE 3: The value of alpha_block_count is 4.

motion_coding(mode, type_of_mb) {	No. of bits	Mnemonic
motion_vector(mode)		
if (type_of_mb == 2) {		
for (i = 0; i < 3; i++)		
motion_vector(mode)		
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FOUGA Y. ADAWAMI
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

6.2.5.4 Sprite coding

decode_sprite_piece() {	No. of bits	Mnemonic
piece_quant	5	bslbf
piece_width	9	bslbf
piece_height	9	bslbf
marker_bit	1	bslbf
piece_xoffset	9	bslbf
piece_yoffset	9	bslbf
sprite_shape_texture()		
}		

sprite_shape_texture() {	No. of bits	Mnemonic
if (sprite_transmit_mode == "piece") {		
for (i=0; i < piece_height; i++) {		
for (j=0; j < piece_width; j++) {		
if (!send_mb()) {		
macroblock()		
}		
}		
}		
}		
if (sprite_transmit_mode == "update") {		
for (i=0; i < piece_height; i++) {		
for (j=0; j < piece_width; j++) {		
macroblock()		
}		
}		
}		
}		

sprite_trajectory() {	No. of bits	Mnemonic
for (i=0; i < no_of_sprite_warping_points; i++) {		
warping_mv_code(du[i])		
warping_mv_code(dv[i])		
}		
}		

warping_mv_code(d) {	No. of bits	Mnemonic
dmv_length	2..12	uimsbf
if (dmv_length != "00")		
dmv_code	1..14	uimsbf
marker_bit	1	bslbf
}		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: 1274457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2122
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

brightness_change_factor() {	No. of bits	Mnemonic
brightness_change_factor_size	1-4	uimsbf
brightness_change_factor_code	5-10	uimsbf
}		

6.2.8 Macroblock

macroblock() {	No. of bits	Mnemonic
if (vop_coding_type != "B") {		
if (video_object_layer_shape != "rectangular" && !(sprite_enable == "static" && low_latency_sprite_enable && sprite_transmit_mode == "update"))		
mb_binary_shape_coding()		
if (video_object_layer_shape != "binary only") {		
if (!transparent_mb()) {		
if (video_object_layer_shape != "rectangular" && !(sprite_enable = "static" && low_latency_sprite_enable && sprite_transmit_mode == "update")) {		
do{		
if (vop_coding_type != "I" && (sprite_enable == "static" && sprite_transmit_mode== "piece"))		
not_coded	1	bslbf
if (!not_coded || vop_coding_type == "I" || (vop_coding_type == "S" && low_latency_sprite_enable && sprite_transmit_mode== "piece"))		
mcbpc	1-9	vlclbf
} while(!(not_coded || derived_mb_type != "stuffing"))		
} else {		
if (vop_coding_type != "I" && !(sprite_enable = "static" && sprite_transmit_mode == "piece"))		
not_coded	1	bslbf
if (!not_coded || vop_coding_type == "I" || (vop_coding_type == "S" && low_latency_sprite_enable && sprite_transmit_mode == "piece"))		
mcbpc	1-9	vlclbf
}		
if (!not_coded || vop_coding_type == "I" || (vop_coding_type == "S" && low_latency_sprite_enable && sprite_transmit_mode == "piece")) {		
if (vop_coding_type == "S" && sprite_enable == "GMC" && (derived_mb_type == 0 || derived_mb_type == 1))		
mcsel	1	bslbf
if (!short_video_header && (derived_mb_type == 3 || derived_mb_type == 4))		
ac_pred_flag	1	bslbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FOUDA Y. ADWAN/ANSI Store order #1127445. Downloaded 7/16/2007 2:06:21 PM ET
Single user license only. Copying and networking prohibited.
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2123
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

if (derived_mb_type != "stuffing")		
cbpy	1-6	vlclbf
else		
return()		
if (derived_mb_type == 1 ||		
derived_mb_type == 4)		
dquant	2	bslbf
if (interlaced)		
interlaced_informations()		
if (!(ref_select_code=="11" && scalability)		
&& sprite_enable != "static") {		
if ((derived_mb_type == 0 || derived_mb_type == 1)		
&& (vop_coding_type == "P"		
|| (vop_coding_type == "S" && !mcsel))) {		
motion_vector("forward")		
if (interlaced && field_prediction)		
motion_vector("forward")		
}		
if (derived_mb_type == 2) {		
for (j=0; j < 4; j++)		
if (!transparent_block(j))		
motion_vector("forward")		
}		
}		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
block(i)		
}		
}		
}		
}		
else {		
if (video_object_layer_shape != "rectangular")		
mb_binary_shape_coding()		
if ((co_located_not_coded != 1 || (scalability && ref_select_code != "11"		
|| enhancement_type == 1)) || (sprite_enable == "GMC"		
&& backward_reference_vop_coding_type == "S"))		
&& video_object_layer_shape != "binary only") {		
if (!transparent_mb()) {		
modb	1-2	vlclbf
if (modb != "1") {		
mb_type	1-4	vlclbf
if (modb == "00")		
cbpb	3-6	vlclbf
if (ref_select_code != "00" || !scalability) {		
if (mb_type != "1" && cbpb!=0)		
dbquant	1-2	vlclbf
if (interlaced)		
interlaced_information()		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG X
ANSI Store order #: 07/44[illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

if (mb_type == '01' || mb_type == '0001') {		
motion_vector("forward")		
if (interlaced && field_prediction)		
motion_vector("forward")		
}		
if (mb_type == '01' || mb_type == 001') {		
motion_vector("backward")		
if (interlaced && field_prediction)		
motion_vector("backward")		
}		
if (mb_type == '1")		
motion_vector("direct")		
}		
if (ref_select_code == '00' && scalability &&		
ckxb !=0) {		
dbquant	1-2	vlclbf
if (mb_type == '01' || mb_type == '1')		
motion_vector("forward")		
}		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
block(i		
}		}
}		
}		
}		
if(video_object_layer_shape=="grayscale"		
&& !transparent_mb()) {		
for(j=1; j<aux_comp_count; j++) {		
if(vop_coding_type=="I" || (vop_coding_type=="P" ||		
(vop_coding_type=="S" && sprite_enable=="GMC"))		
&& !not_coded &&		
(derived_mb_type==3 || derived_mb_type==4)) {		
coda_i	1	bslbf
if(coda_i=="coded") {		
ac_pred_flag_alpha	1	bslbf
cbpa	1-6	vlclbf
for(i=0;i<alpha_block_count;i++)		
if(!transparent_block())		
alpha_block(i)		
}		
} else { /* P, S(GMC) or B macroblock */		
if(vop_coding_type == 'P' || (sprite_enable == "GMC" &&		
(vop_coding_type=="S"		
|| backward_reference_vop_coding_type=="S"))		
|| co_located_not_coded != 1) {		
coda_pb	1-2	vlclbf
if(coda_pb=="coded") {		

© ISO/IEC 2001 – All rights reserved

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/FOUDA Y. ADAWAMI
ANSI Store order #1021445 Downloaded: 7/18/2007 2:06:21 PM EST
Single user license only. Copying and networking prohibited.

cbps	1-6	vlclbf
for(i=0;i<alpha_block_count;i++)		
if(!transparent_block(i)		
alpha_block(i)		
}		
}		
}		
}		

NOTE: The value of block_count is 6 in the 4:2:0 format. The value of alpha_block_count is 4. backward_reference_vop_coding_type means the vop_coding_type of the backward reference VOP as described in subclause 7.6.7.

6.2.6.1 MB Binary Shape Coding

mb_binary_shape_coding() {	No. of bits	Mnemonic
if((scalability && hierarchy_type == '0' && (enhancement_type == '0' || use_ref_shape == '0')) && (scalability && video_object_layer_shape == "binary only")) {		
bab_type	1-7	vlclbf
if (vop_coding_type == 'P' || vop_coding_type == 'B' || (vop_coding_type == 'S' && sprite_enable == "GMC")) {		
if ((bab_type==1) || (bab_type == 5)) {		
mvds_x	1--8	vlclbf
mvds_y	1--8	vlclbf
}		
}		
if (bab_type >=4) {		
if (!change_conv_ratio_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_code()		
}		
} else {		
if (!use_ref_shape || video_object_layer_shape == "binary only") {		
enh_bab_type	1-3	vlclbf
if (enh_bab_type == 3)		
scan_type	1	bslbf
if (enh_bab_type == 1 || enh_bab_type == 3)		
enh_binary_arithmetic_code()		
}		
}		
}		

backward_shape () {	No. of bits	Mnemonic
for(i=0; i<backward_shape_height/16; i++)		
for(j=0; j<backward_shape_width/16; j++) {		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS Y
ANSI Store order #: 0744/5 Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2126
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

bab_type	1-3	vlclbf
if (bab_type >=4) {		
if (!change_conv_ratio_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_code()		
}		
}		
}		

forward_shape () {	No. of bits	Mnemonic
for(i=0; i<forward_shape_height/16; i++)		
for(j=0; j<forward_shape_width/16; j++) {		
bab_type	1-3	vlclbf
if (bab_type >=4) {		
if (!change_conv_ratio_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_code()		
}		
}		
}		

6.2.5.2 Motion vector

motion_vector (mode) {	No. of bits	Mnemonic
if (mode == „direct") {		
horizontal_mv_data	1-13	vlclbf
vertical_mv_data	1-13	vlclbf
}		
else if (mode == „forward") {		
horizontal_mv_data	1-13	vlclbf
if ((vop_fcode_forward != 1) && (horizontal_mv_data != 0))		
horizontal_mv_residual	1-6	uimsbf
vertical_mv_data	1-13	vlclbf
if ((vop_fcode_forward != 1) && (vertical_mv_data != 0))		
vertical_mv_residual	1-6	uimsbf
}		
else if (mode == „backward") {		
horizontal_mv_data	1-13	vlclbf
if ((vop_fcode_backward != 1) && (horizontal_mv_data != 0))		
horizontal_mv_residual	1-6	uimsbf
vertical_mv_data	1-13	vlclbf
if ((vop_fcode_backward != 1) && (vertical_mv_data != 0))		
vertical_mv_residual	1-6	uimsbf
}		
}		

© ISO/IEC 2001 – All rights reserved

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ACQUISITIONS
ANSI Store order #: 01744[illegible] Downloaded: 7/18/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

6.2.6.3 Interlaced Information

interlaced_information( ) {	No. of bits	Mnemonic
if ((derived_mb_type == 3) || (derived_mb_type == 4) || (cbp != 0))		
dct_type	1	bslbf
if (((vop_coding_type == "P") && ((derived_mb_type == 0) || (derived_mb_type == 1))) || ((sprite_enable == "GMC") && (vop_coding_type == "S") && (derived_mb_type < 2) && (!mcsel)) || ((vop_coding_type == "B") && (mb_type != "1"))) {		
field_prediction	1	bslbf
if (field_prediction) {		
if (vop_coding_type == "P" || (vop_coding_type == "B" && mb_type != "001")) {		
forward_top_field_reference	1	bslbf
forward_bottom_field_reference	1	bslbf
}		
if ((vop_coding_type == "B") && (mb_type != "0001")) {		
backward_top_field_reference	1	bslbf
backward_bottom_field_reference	1	bslbf
}		
}		
}		
}		

6.2.7 Block

The detailed syntax for the term "DCT coefficient" is fully described in clause 7.

block(i) {	No. of bits	Mnemonic
last = 0		
if(!data_partitioned && (derived_mb_type == 3 || derived_mb_type == 4)) {		
if(short_video_header == 1)		
intra_dc_coefficient	8	uimsbf
else if (use_intra_dc_vlc == 1) {		
if (i<4) {		
dct_dc_size_luminance	2-11	vlclbf
if(dct_dc_size_luminance != 0)		
dct_dc_differential	1-12	vlclbf
if (dct_dc_size_luminance > 8)		
marker_bit	1	bslbf
} else {		
dct_dc_size_chrominance	2-12	vlclbf
if(dct_dc_size_chrominance !=0)		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS PATTERSON
ANSI Store order #: 1127445 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2128
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

dct_dc_differential	1-12	vlclbf
if (dct_dc_size_chrominance > 8)		
marker_bit	1	bslbf
}		
}		
}		
if (pattern_code[i])		
while (! last)		
DCT coefficient	3-24	vlclbf
}		
NOTE : "last" is defined to be the LAST flag resulting from reading the most recent DCT coeficient.		

6.2.7.1 Alpha Block

The syntax for DCT coeficient decoding is the same as for block(i) in subclause 6.2.8

alpha_block(i) {	No. of bits	Mnemonic
last = 0		
if(data_partitioned && (vop_coding_type == "I" || (vop_coding_type == "P" || (vop_coding_type == "S" && sprite_enable == "GMC")) && !not_coded && (derived_mb_type == 3 || derived_mb_type == 4))){		
dct_dc_size_alpha	2-11	vlclbf
if(dct_dc_size_alpha != 0)		
dct_dc_differential	1-12	vlclbf
if(dct_dc_size_alpha > 8)		
marker_bit	1	bslbf
}		
if (pattern_code[i])		
while (! last)		
DCT coeficient	3-24	vlclbf
}		
NOTE: "last" is defined to be the LAST flag resulting from reading the most recent DCT coeficient.		

6.2.8 Still Texture Object

StillTextureObject() {	No. of bits	Mnemonic
still_texture_object_start_code	32	bslbf
if (visual_object_verid != 0001) {		
tiling_disable	1	bslbf
texture_error_resilience_disable	1	bslbf
texture_object_id	16	uimsbf
marker_bit	1	bslbf
wavelet_filter_type	1	uimsbf
wavelet_download	1	uimsbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCED
ANSI Store order #1374457 Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2129
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

wavelet_decomposition_levels	4	uimsbf
scan_direction	1	bslbf
start_code_enable	1	bslbf
texture_object_layer_shape	2	uimsbf
quantisation_type	2	uimsbf
if (quantisation_type == 2) {		
spatial_scalability_levels	4	uimsbf
if (spatial_scalability_levels != wavelet_decomposition_levels) {		
use_default_spatial_scalability	1	uimsbf
if (use_default_spatial_layer_size == 0)		
for (i=0; i<spatial_scalability_levels - 1; i++)		
wavelet_layer_index	4	uimsbf
}		
}		
if (wavelet_download == "1"){		
uniform_wavelet_filter	1	uimsbf
if (uniform_wavelet_filter == "1")		
download_wavelet_filters()		
else		
for (i=0; i<wavelet_decomposition_levels; i++)		
download_wavelet_filters( )		
}		
wavelet_stuffing	3	uimsbf
if(!texture_error_resilience_disable) {		
target_segment_length	16	uimsbf
marker_bit	1	bslbf
}		
if(texture_object_layer_shape == "00") {		
texture_object_layer_width	16	uimsbf
marker_bit	1	bslbf
texture_object_layer_height	16	uimsbf
marker_bit	1	bslbf
}		
else if (texture_object_layer_shape == "01") {		
horizontal_ref	15	uimsbf
marker_bit	1	bslbf
vertical_ref	15	uimsbf
marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf
/* if tiling_disable == "1" and texture_object_layer_shape == "01" configuration information precedes this point; elementary stream data follows. See annex K */		
if(tiling_disable == "1")		
shape_object_decoding()		
}		
if (tiling_disable == "0"){		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: [illegible] Downloaded: [illegible]
Single user license only. Copying and networking prohibited.
Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2130
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

tile_width	15	uimsbf
marker_bit	1	bslbf
tile_height	15	uimsbf
marker_bit	1	bslbf
number_of_tiles	16	uimsbf
marker_bit	1	bslbf
tiling_jump_table_enable	1	bslbf
if (tiling_jump_table_enable == "1") {		
for (i=0; i<number_of_tiles; i++) {		
tile_size_high	16	uimsbf
market_bit	1	bslbf
tile_size_low	16	uimsbf
market_bit	1	bslbf
}		
}		
next_start_code()		
}		
/* if tiling_disable == "0" or texture_object_layer_shape == "00" configuration information precedes this point; elementary stream data follows. See annex K */		
do {		
if(tiling_disable == "0") {		
texture_tile_start_code	32	bslbf
tile_id	16	uimsbf
if (texture_object_layer_shape == "01") {		
marker_bit	1	bslbf
texture_tile_type	2	uimsbf
marker_bit	1	bslbf
}		
}		
if (!texture_error_resilience_disable) {		
if (texture_object_layer_shape=="01"&& tiling_disable=="0") {		
if (texture_tile_type=="boundary tile")		
shape_object_decoding()		
}		
while (nextbit_bytealigned () == texture_marker) {		
TexturePacketHeader ()		
do {		
while (texture_unit_not_completed) {		
DecodeStuff()		
if (segment_length >= target_segment_length)		
decode_segment_marker()		
}		
} while (nextbit_bytealigned () != texture_marker)		
}		
}		
else {		
if (texture_object_layer_shape=="01"&& tiling_disable=="0") {		

© ISO/IEC 2003 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCE
ANSI Store order #1127445. Downloaded 7/16/2007 3:08:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2131
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

if (texture_tile_type=="boundary tile")		
shape_object_decoding()		
}		
for (color = "y", "u", "v") {		
wavelet_dc_decode()		
}		
if (quantisation_type == 1) {		
TextureLayerSQ ( )		
}		
else if (quantisation_type == 2) {		
if (start_code_enable == 1) {		
do {		
TextureSpatialLayerMQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i =0; i<spatial_scalability_levels; i++)		
TextureSpatialLayerMQNSC()		
}		
}		
else if (quantisation_type == 3) {		
for (color = "y", "u", "v")		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
max_bitplanes	5	uimsbf
if (scan_direction == 0) {		
do {		
TextureSNRLayerSQ ( )		
} while (next_bits() == texture_snr_layer_start_code)		
} else {		
do {		
TextureSpatialLayerSQ ( )		
} while (next_bits() == texture_spatial_layer_start_code)		
}		
}		
/* error_resi_disable */		
if (tiling_disable == "0")		
next_start_code()		
} while (nextbits_bytealigned () == texture_tile_start_code)		
}		
else { /* version 1 */		
texture_object_id	16	uimsbf
marker_bit	1	bslbf
wavelet_filter_type	1	uimsbf
wavelet_download	1	uimsbf
wavelet_decomposition_levels	4	uimsbf

© ISO/IEC 2001 – All rights reserved
Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS on [illegible]
ANSI Store order #: [illegible] Downloaded: 7/18/2007 2:08:21 PM EST
Single user license only. Copying and networking prohibited.

scan_direction	1	bslbf
start_code_enable	1	bslbf
texture_object_layer_shape	2	uimsbf
quantization_type	2	uimsbf
if (quantization_type == 2) {		
spatial_scalability_levels	4	uimsbf
if (spatial_scalability_levels != wavelet_decomposition_levels) {		
use_default_spatial_scalability	1	uimsbf
if (use_default_spatial_layer_size == 1)		
for (i=0; i<spatial_scalability_levels – 1; i++)		
wavelet_layer_index	4	
}		
}		
if (wavelet_download == "1"){		
uniform_wavelet_filter	1	uimsbf
if (uniform_wavelet_filter == "1")		
download_wavelet_filters()		
else		
for (i=0; i<wavelet_decomposition_levels; i++)		
download_wavelet_filters()		
}		
wavelet_stuffing	3	uimsbf
if(texture_object_layer_shape == "00"){		
texture_object_layer_width	15	uimsbf
marker_bit	1	bslbf
texture_object_layer_height	15	uimsbf
marker_bit	1	bslbf
}		
else {		
horizontal_ref	15	imsbf
marker_bit	1	bslbf
vertical_ref	15	imsbf
marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf
shape_object_decoding ( )		
}		
/* configuation information precedes this point; elementary stream data follows. See annex K */		
for (color = "y", "u", "v")		
wavelet_dc_decode()		
if(quantization_type == 1)		
TextureLayerSQ()		
else if (quantization_type == 2) {		
if (start_code_enable == 1) {		
do {		

© ISO/IEC 2001 – All rights reserved

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/PODIATRY ACQUISITIONS
ANSI Store order #: 1274457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

TextureSpatialLayerMQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i =0; i<spatial_scalability_levels; i++)		
TextureSpatialLayerMQNSC()		
{		
}		
else if (quantization_type == 3) {		
for (color = "y", "u", "v")		
do{		
quant_byte		
} while(quant_byte >>7)		
max_bitplanes		
if (scan_direction == 0) {		
do {		
TextureSNRLayerBQ()		
} while (next_bits() == texture_snr_layer_start_code)		
} else {		
do {		
TextureSpatialLayerBQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
}		
}		
}		

NOTE 1 : The value of texture_unit_not_completed is '0' if the decoding of one sub-unit in a texture unit is completed. Otherwise the value is '1'.

NOTE 2 : The value of first_packet_decoded is set to '0' if the first packet has not been decoded. The value of first_packet_decoded is set to '1' after the first packe has been decoded.

TexturePacketHeader() {	No. of bits	Mnemonic
next_texture_marker()		
texture_marker	17	bslbf
do {		
TU_first	8	uimsbf
} while (TU_first >> 7)		
do{		
TU_last	8	uimsbf
} while (TU_last >> 7)		
header_extention_code	1	bslbf
if (header_extention_code) {		
texture_object_id	16	uimsbf
marker_bit	1	bslbf
wavelet_filter_type	1	uimsbf
wavelet_download	1	uimsbf
wavelet_decomposition_levels	4	uimsbf
scan_direction	1	bslbf
star_code_enable	1	bslbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: 0124457 Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2134
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

texture_object_layer_shape	2	uimsbf
quantisation_type	2	uimsbf
if (quantisation_type == 2) {		
spatial_scalability_levels	4	uimsbf
if (spatial_scalability_levels != wavelet_decomposition_levels) {		
use_default_spatial_scalability	1	uimsbf
if (use_default_spatial_layer_size == 0)		
for (i=0; i<spatial_scalability_levels – 1; i++)		
wavelet_layer_index	4	uimsbf
}		
}		
if (wavelet_download == "1") {		
uniform_wavelet_filter	1	uimsbf
if (uniform_wavelet_filter == "1")		
download_wavelet_filters()		
else		
for (i=0; i<wavelet_decomposition_levels; i++)		
download_wavelet_filters( )		
}		
wavelet_stuffing	3	uimsbf
if(texture_object_layer_shape == "00") {		
texture_object_layer_width	15	uimsbf
marker_bit	1	bslbf
texture_object_layer_height	15	uimsbf
marker_bit	1	bslbf
}		
else {		
if (!first_packet_decoded) {		
horizontal_ref	15	uimsbf
marker_bit	1	bslbf
vertical_ref	15	uimsbf
marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf
}		
}		
if(tiling_disable == "0"){		
tile_width	15	uimsbf
marker_bit	1	bslbf
tile_height	15	uimsbf
marker_bit	1	bslbf
}		
target_segment_length	16	uimsbf
marker_bit	1	bslbf
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODJAT, ADVANCE/ANSI Store order #: 0124457 Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

DecodeStu( ) {	No. of bits	Mnemonic
for (color = "y", "u", "v")		
wavelet_dc_decode ()		
if(quantisation_type == 1) {		
TextureLayerSQ()		
}		
else if (quantisation_type == 2) {		
if (start_code_enable == 1) {		
do {		
TextureSpatialLayerNQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i=0; i<spatial_scalability_levels; i++)		
TextureSpatialLayerNQNSC()		
}		
}		
else if (quantisation_type == 3) {		
for (color = "y", "u", "v")		
do{		
quant_byte	8	uimsbf
} while(quant_byte >> 7)		
max_bitplanes	5	uimsbf
if (scan_direction == 0) {		
do {		
TextureSNRLayerSQ()		
} while (next_bits() == texture_snr_layer_start_code)		
} else {		
do {		
TextureSpatialLayerSQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
}		
}		
}		

6.2.8.1 TextureLayerSQ

TextureLayerSQ( ) {	No. of bits	Mnemonic
if (scan_direction == 0) {		
for (y", "u", "v") {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<wavelet_decomposition_loves; i++)		
if (i!=0 || color!= "u","v") {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4==0)		
marker_bit	1	bslbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG A. YOUNG
ANSI Store order #: 01744 Downloaded: 7/18/2007 3:38:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2136
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

}
}
for (i = 0; i<tree_blocks; i++)
for (color = "y", "u", "v")
arith_decode_highbands_bt()
} else {
if (start_code_enable) {
do {
TextureSpatialLayerSQ()
} while (next_bits() == texture_spatial_layer_start_code)
} else {
for (i = 0; i< wavelet_decomposition_levels; i++)
TextureSpatialLayerSQNSC()
}
}
}		

NOTE: The value of tree_block is that wavelet coefficients are organized in a tree structure which is rooted in the low-low band (DC band) of the wavelet decomposition, then extends into the higher frequency bands at the same spatial location. Note the DC band is encoded separatey.

6.2.8.2 TextureSpatialLayerSQ

TextureSpatialLayerSQ() {	No. of bits	Mnemonic
texture_spatial_layer_start_code	32	bslbf
texture_spatial_layer_id	5	uimsbf
TextureSpatialLayerSQNSC()		
}		

6.2.8.3 TextureSpatialLayerSQNSC

TextureSpatialLayerSQNSC() {	No. of bits	Mnemonic
for (color="y","u","v") {		
if((first_wavelet_layer && color=="y") || (second_wavelet_layer && color=="u","v"))		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
if (color =="y")		
max_bitplanes	5	uimbsf
else if (!first_wavelet_layer)		
max_bitplanes	5	uimbsf
}		
for (color="y","u","v")		
if (color="y" || !first_wavelet_layer)		
arith_decode_highbands_bt()		
}		

NOTE:The value of first_wavelet_layer becomes "true" when the variable 'i' of subclause 6.2.8.1 TextureLayerSQ() equals to zero. Otherwise, it is "false". The value of second_wavelet_layer becomes "true" when the variable 'i' of subclause 6.2.8.1 TextureLayerSQ() equals to one. Otherwise, it is "false".

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ANDREWS
ANSI Store order #: [illegible] Downloaded: 7/16/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2137
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.8.4 TextureSpatialLayerMQ

TextureSpatialLayerMQ() {	No. of bits	Mnemonic
texture_spatial_layer_start_code	32	bslbf
texture_spatial_layer_id	5	uimsbf
snr_scalability_levels	5	uimsbf
do {		
TextureSNRLayerMQ( )		
} while (next_bits() == texture_snr_layer_start_code)		
}		

6.2.8.5 TextureSpatialLayerMQNSC

TextureSpatialLayerMQNSC() {	No. of bits	Mnemonic
snr_scalability_levels	5	uimsbf
for (i =0;i<snr_scalability_levels; i++)		
TextureSNRLayerMQNSC ( )		
}		

6.2.8.6 TextureSNRLayerMQ

TextureSNRLayerMQ(){		
texture_snr_layer_start_code	32	bslbf
texture_snr_layer_id	5	uimsbf
TextureSNRLayerMQNSC()		
}		

6.2.8.7 TextureSNRLayerMQNSC

TextureSNRLayerMQNSC(){	No. of bits	Mnemonic
if (spatial_scalability_levels == wavelet_decomposition_levels && spatial_layer_id == 0){		
for (color = "y") {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
else {		
for (color="y", "u", "v") {		
do {		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #: ... Downloaded: ... PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2138
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
if (scan_direction == 0) {		
for (i = 0; i<tree_blocks; i++)		
for (color = "y", "u", "v")		
if (wavelet_decomposition_layer_id != 0 || color != "u", "v")		
arith_decode_highbands_bit()		
} else{		
for (i = 0; i< spatial_layers; i++){		
for (color = "y", "u", "v"){		
if (wavelet_decomposition_layer_id != 0 || color != "u", "v")		
arith_decode_highbands_bit()		
}		
}		
}		
}		
NOTE: The value of spatial_layers is equivalent to the maximum number of the wavelet decomposition layers in that scalability layer.		

6.2.8.8 TextureSpatialLayerBQ

TextureSpatialLayerBQ() {	No. of bits	Mnemonic
texture_spatial_layer_start_code	32	bslbf
texture_spatial_layer_id	5	uimsbf
for (i=0; i<max_bitplanes; i++) {		
texture_snr_layer_start_code	32	bslbf
texture_snr_layer_id	5	uimsbf
TextureBitPlaneBQ()		
next_start_code()		
}		
}		

6.2.8.9 TextureBitPlaneBQ

TextureBitPlaneBQ () {	No. of bits	Mnemonic
for (color = "y", "u", "v")		
if (wavelet_decomposition_layer_id == 0){		
all_nonzero[color]	1	bslbf
if (all_nonzero[color] == 0) {		
all_zero[color]	1	bslbf

© ISO/IEC 2000 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAILY, ADAMANS
ANSI Store order #: 0747457 Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2139
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

if (all_zero[color]==0) {		
lh_zero[color]	1	bslbf
hl_zero[color]	1	bslbf
hh_zero[color]	1	bslbf
}		
}		
}		
if (wavelet_decomposition_layer_id != 0 ||color != "u" "v"){		
if(all_nonzero[color]==1 || all_zero[color]==0){		
if (scan_direction == 0)		
arith_decode_highbands_bilevel_bb()		
else		
arith_decode_highbands_bilevel_tdlj		
}		
}		
}		
}		

6.2.8.10 TextureSNRLayerBQ

TextureSNRLayerBQ() {	No. of bits	Mnemonic
texture_snr_layer_start_code	32	bslbf
texture_snr_layer_id	5	uimsbf
for (i=0;i<wavelet_decomposition_levels; i++){		
texture_spatial_layer_start_code	32	bslbf
texture_spatial_layer_id	5	uimsbf
TextureBiPlaneBQ()		
next_start_code ( )		
}		
}		

6.2.8.11 DownloadWaveletFilters

download_wavelet_filters() {	No. of bits	Mnemonic
lowpass_filter_length	4	uimsbf
highpass_filter_length	4	uimsbf
do{		
if (wavelet_filter_type == 0) {		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS PATENT
ANSI Store order #: 01/74467 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

} while (lowpass_filter_length--)		
do{		
if (wavelet_filter_type == 0){		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		
} while (highpass_filter_length--)		
if (wavelet_filter_type == 0) {		
integer_scale	16	uimsbf
marker_bit	1	bslbf
}		
}		

6.2.3.12 Wavelet dc decode

wavelet_dc_decode() {	No. of bits	Mnemonic
mean	8	uimsbf
do{		
quant_dc_byte	8	uimsbf
} while(quant_dc_byte >>7)		
do{		
band_offset_byte	8	uimsbf
} while (band_offset_byte >>7)		
do{		
band_max_byte	8	uimsbf
} while (band_max_byte >>7)		
arith_decode_stuff()		
}		

6.2.3.13 Wavelet higher bands decode

wavelet_higher_bands_decode() {	No. of bits	Mnemonic
do{		
root_max_alphabet_byte	8	uimsbf
} while (root_max_alphabet_byte >>7)		
marker_bit	1	bslbf
do{		
velz_max_alphabet_byte	8	uimsbf
} while (velz_max_alphabet_byte >>7)		
do{		
velnz_max_alphabet_byte	8	uimsbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ADAMIAK
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2141
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

} while (vlcnz_max_alphabet_byte >>7)		
arith_decode_highbands()		
}		

6.2.8.14 Shape Object Decoding

shape_object_decoding() {	No. of bits	Mnemonic
change_conv_ratio_disable	1	bslbf
sto_constant_alpha	1	bslbf
if (sto_constant_alpha)		
sto_constant_alpha_value	8	bslbf
if (visual_object_verid != 0001) {		
marker_bit	1	bslbf
for(i=0; i<shape_base_layer_height_blocks(); i++) {		
for(j=0; j<shape_base_layer_width_blocks(); j++) {		
bab_type	1-4	vlclbf
if (bab_type == 4) {		
if (!change_conv_ratio_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_decode()		
}		
}		
}		
marker_bit	1	bslbf
if (!start_code_enable) {		
sto_shape_coded_layers	4	uimsbf
marker_bit	1	bslbf
for(k = 0; k < sto_shape_coded_layers; k++){		
for(i=0;i<shape_enhanced_layer_height_blocks(); i++)		
for(j=0; j<shape_enhanced_layer_width_blocks(); j++)		
enh_binary_arithmetic_decode()		
marker_bit	1	bslbf
}		
}		
else{		
next_start_code()		
while (nextbits() == texture_shape_layer_start_code) {		
texture_shape_layer_start_code	32	bslbf
texture_shape_layer_id	6	uimsbf
marker_bit	1	bslbf
for(i=0;i<shape_enhanced_layer_height_blocks(); i++)		
for(j=0; j<shape_enhanced_layer_width_blocks(); j++)		
enh_binary_arithmetic_decode()		
marker_bit	1	bslbf
next_start_code()		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ...
ANSI Store order #: ... Downloaded: ... Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2142
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

texture_spatial_layer_start_code	32	bslbf
texture_spatial_layer_id	5	uimsbf
marker_bit	1	bslbf
}		
} else { /* version 1 */		
for (i=0 ; i<((object_width+15)/16)*(object_height+15)/16) ; i++) {		
bab_type	1-2	vlclbf
if (bab_type==4) {		
if (!change_conv_ratio_disable)		
conv_ratio	1-2	vlclbf
scan_type	1	bslbf
binary_arithmetic_decode()		
}		
}		
}		

6.2.9 Mesh Object

MeshObject() {	No. of bits	Mnemonic
mesh_object_start_code	32	bslbf
do{		
MeshObjectPlane()		
} while (next_bits_bytealigned() == mesh_object_plane_start_code || next_bits_bytealigned() != '0000 0000 0000 0000 0000 0001')		
}		

6.2.9.1 Mesh Object Plane

MeshObjectPlane() {	No. of bits	Mnemonic
MeshObjectPlaneHeader()		
MeshObjectPlaneData()		
}		

MeshObjectPlaneHeader() {	No. of bits	Mnemonic
if (next_bits_bytealigned()=='0000 0000 0000 0000 0000 0001'){		
next_start_code()		
mesh_object_plane_start_code	32	bslbf
}		
is_intra	1	bslbf
mesh_mask	1	bslbf
temporal_header()		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS T. ADAMANIS
ANSI Store order #1274457 Downloaded: 7/18/2007 2:06:21 PM EST
Single user licensed only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2143
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MeshObject*ForwData() {	No. of bits	Mnemonic
if (mesh_mask == 1) {		
if (is_intra == 1)		
mesh_geometry()		
else		
mesh_motion()		
}		
}		

6.2.9.2 Mesh geometry

mesh_geometry() {	No. of bits	Mnemonic
mesh_type_code	2	bslbf
if (mesh_type_code == '01') {		
nr_of_mesh_nodes_hor	10	uimsbf
nr_of_mesh_nodes_vert	10	uimsbf
marker_bit	1	uimsbf
mesh_rect_size_hor	8	uimsbf
mesh_rect_size_vert	8	uimsbf
triangle_split_code	2	bslbf
}		
else if (mesh_type_code == '10') {		
nr_of_mesh_nodes	16	uimsbf
marker_bit	1	uimsbf
nr_of_boundary_nodes	10	uimsbf
marker_bit	1	uimsbf
node0_x	13	simsbf
marker_bit	1	uimsbf
node0_y	13	simsbf
marker_bit	1	uimsbf
for (n=1; n < nr_of_mesh_nodes; n++){		
delta_x_len_vlc	2-2	vlcbf
if (delta_x_len_vlc)		
delta_x	1-14	vlcbf
delta_y_len_vlc	2-2	vlcbf
if (delta_y_len_vlc)		
delta_y	1-14	vlcbf
}		
}		
}		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PROQUEST, Downloaded: 11/6/2007 3:08:21 PM EST
ANSI Store order #: [illegible] Downloaded: 11/6/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

6.2.9.3 Mesh motion

mesh_motion() {	No. of bits	Mnemonic
motion_range_code	3	bslbf
for (n=0; n <nr_of_mesh_nodes; n++) {		
node_motion_vector_flag	1	bslbf
if (node_motion_vector_flag == '0') {		
delta_mv_x_vlc	1-13	vlclbf
if ((motion_range_code != 1) && (delta_mv_x_vlc != 0))		
delta_mv_x_res	1-6	uimsbf
delta_mv_y_vlc	1-13	vlclbf
if ((motion_range_code != 1) && (delta_mv_y_vlc != 0))		
delta_mv_y_res	1-6	uimsbf
}		
}		
}		

6.2.10 FBAObject

fba_object() {	No. of bits	Mnemonic
fba_object_start_code	32	bslbf
do {		
fba_object_planes()		
} while (!(		
(nextbits_bytesligned() == '000 0000 0000 0000 0000 0000') &&		
(nextbits_bytealigned() != fba_object_plane_start_code)))		
}		

6.2.10.1 FBA Object Plane

fba_object_planes() {	No. of bits	Mnemonic
fba_object_plane_header()		
fba_object_plane_data()		
}		

fba_object_plane_header() {	No. of bits	Mnemonic
is_intra	1	bslbf
fba_object_mask	2	bslbf
temporal_header()		
}		

© ISO/IEC 2003 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ANDREWS
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2145
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

fba_object_plane_data() {	No. of bits	Mnemonic
if(fba_object_mask &'01') {		
if(is_intra) {		
fap_quant	5	uimsbf
for (group_number = 1; group_number <= 10; group_number++) {		
marker_bit	1	uimsbf
fap_mask_type	2	bslbf
if(fap_mask_type == '01'|| fap_mask_type == '10')		
fap_group_mask[group_number]	2-16	vlcbf
}		
fba_suggested_gender	1	bslbf
fba_object_coding_type	1	bslbf
if(fba_object_coding_type == 0) {		
is_i_new_max	1	bslbf
is_i_new_min	1	bslbf
is_p_new_max	1	bslbf
is_p_new_min	1	bslbf
decode_new_minmax()		
decode_fap()		
}		
if(fba_object_coding_type == 1)		
decode_i_segment()		
}		
else {		
if(fba_object_coding_type == 0)		
decode_pfap()		
if(fba_object_coding_type == 1)		
decode_p_segment()		
}		
}		
}		
if(fba_object_mask &'10') {		
if(is_intra) {		
bap_pred_quant_index	5	uimsbf
for (group_number = 1; group_number <= BAP_NUM_GROUPS; group_number++) {		
marker_bit	1	uimsbf
bap_mask_type	2	bslbf
if(bap_mask_type == '01')		
bap_group_mask[group_number]	3-32	vlcbf
else if (bap_mask_type == '00') {		
for(i=0; i<BAPS_IN_GROUP[group_number];i++) {		
bap_group_mask[group_mask][] = 0		
}		
}		
else if (bap_mask_type == '11') {		
for(i=0; i<BAPS_IN_GROUP[group_number];i++) {		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG YOUNG
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:08:21 PM EST
Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2146
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

bap_group_mask[group_mask[i]] = 1		
}		
}		
}		
fba_suggested_gender	1	bslbf
fba_object_coding_type	1	bslbf
if (fba_object_coding_type == 0) {		
bap_is_i_new_max	1	bslbf
bap_is_i_new_min	1	bslbf
bap_is_p_new_max	1	bslbf
bap_is_p_new_min	1	bslbf
decode_bap_new_minmax()		
decode_bap_ibap()		
}		
if(fba_object_coding_type == 1)		
decode_bap_i_segment()		
}		
else {		
if (fba_object_coding_type == 0)		
decode_bap_pbap()		
if(fba_object_coding_type == 1)		
decode_bap_p_segment()		
}		
}		

temporal_header() {	No. of bits	Mnemonic
if (is_intra) {		
is_frame_rate	1	bslbf
if(is_frame_rate)		
decode_frame_rate()		
is_time_code	1	bslbf
if (is_time_code)		
time_code	18	bslbf
}		
skip_frames	1	bslbf
if(skip_frames)		
decode_skip_frames()		
}		

6.2.10.2 Decode frame rate and skip frames

decode_frame_rate(){	No. of bits	Mnemonic
frame_rate	3	uimsbf
seconds	8	uimsbf
frequency_offset	1	uimsbf
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PHOTOLAB, ADVANCED
ANSI Store order #: 0127445 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensed only. Copying and networking prohibited.

decode_skip_frames(){	No. of bits	Mnemonic
do{		
number_of_frames_to_skip	4	uimsbf
} while number_of_frames_to_skip = "1111")		
}		

6.2.10.3 Decode new minmax

decode_new_minmax() {	No. of bits	Mnemonic
if (is_i_new_max) {		
for (group_number = 2, j=0; group_number <= 11; group_number++)		
for (i=0; i < NFAF[group_number]; i++, j++) {		
if (!(j & 0x3))		
marker_bit	1	uimsbf
if (fap_group_mask[group_number] & (1<<i))		
i_new_max[j]	5	uimsbf
}		
if (is_i_new_min) {		
for (group_number = 2, j=0; group_number <= 11; group_number++)		
for (i=0; i < NFAF[group_number]; i++, j++) {		
if (!(j & 0x3))		
marker_bit	1	uimsbf
if (fap_group_mask[group_number] & (1<<i))		
i_new_min[j]	5	uimsbf
}		
if (is_p_new_max) {		
for (group_number = 2, j=0; group_number <= 11; group_number++)		
for (i=0; i < NFAF[group_number]; i++, j++) {		
if (!(j & 0x3))		
marker_bit	1	uimsbf
if (fap_group_mask[group_number] & (1<<i))		
p_new_max[j]	5	uimsbf
}		
if (is_p_new_min) {		
for (group_number = 2, j=0; group_number <= 11; group_number++)		
for (i=0; i < NFAF[group_number]; i++, j++) {		
if (!(j & 0x3))		
marker_bit	1	uimsbf
if (fap_group_mask[group_number] & (1<<i))		
p_new_min[j]	5	uimsbf
}		
}		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: [illegible] Downloaded: 7/18/2007 2:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2148
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.10.4 Decode ifap

decode_ifap()	No. of bits	Mnemonic
for (group_number = 1, j=0; group_number <= 10; group_number++) {		
if (group_number == 1) {		
if(fap_group_mask[1] & 0x1)		
decode_viseme()		
if(fap_group_mask[1] & 0x2)		
decode_expression()		
} else {		
for (i= 0; i<NFAP[group_number]; i++, j++) {		
if(fap_group_mask[group_number] & (1 << i)) {		
aa_decode(ifap_Q[j],ifap_cum_freq[ij])		
}		
}		
}		
}		
}		

6.2.10.5 Decode pfap

decode_pfap(){	No. of bits	Mnemonic
for (group_number = 1, j=0; group_number <= 10; group_number++) {		
if (group_number == 1) {		
if(fap_group_mask[1] & 0x1)		
decode_viseme()		
if(fap_group_mask[1] & 0x2)		
decode_expression()		
} else {		
for (i= 0; i<NFAP[group_number]; i++, j++) {		
if(fap_group_mask[group_number] & (1 << i)) {		
aa_decode(pfap_diff[j], pfap_cum_freq[ij])		
}		
}		
}		
}		
}		

6.2.10.6 Decode viseme and expression

decode_viseme() {	No. of bits	Mnemonic
aa_decode(viseme_select1Q, viseme_select1_cum_freq)		vlclbf
aa_decode(viseme_select2Q, viseme_select2_cum_freq)		vlclbf
aa_decode(viseme_blendQ, viseme_blend_cum_freq)		vlclbf
viseme_def	1	bslbf
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC, ADVANCED
ANSI Store order #X37445 Downloaded 7/16/2007 1:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2149
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

decode_expression() {	No. of bits	Mnemonic
aa_decode(expression_select1Q, expression_select1_cum_freq)		vlclbf
aa_decode(expression_intensity1Q, expression_intensity1_cum_freq)		vlclbf
aa_decode(expression_select2Q, expression_select2_cum_freq)		vlclbf
aa_decode(expression_intensity2Q, expression_intensity2_cum_freq)		vlclbf
aa_decode(expression_blendQ, expression_blend_cum_freq)		vlclbf
init_face	1	bslbf
expression_def	1	bslbf
}		

6.2.10.7 Decode i_segment

decode_i_segment(){	No. of bits	Mnemonic
for (group_number= 1, j=0; group_number<= 10; group_number++) {		
if (group_number == 1) {		
if(fap_group_mask[1] & 0x1)		
decode_i_viseme_segment()		
if(fap_group_mask[1] & 0x2)		
decode_i_expression_segment()		
} else {		
for(i=0; i<NFAP[group_number]; i++, j++) {		
if(fap_group_mask[group_number] & (1 << i)) {		
decode_i_dc(dc_Q[j])		
decode_ac(ac_Q[j])		
}		
}		
}		
}		
}		

6.2.10.8 Decode p_segment

decode_p_segment(){	No. of bits	Mnemonic
for (group_number = 1, j=0; group_number <= 10; group_number++) {		
if (group_number == 1){		
if(fap_group_mask[1] & 0x1)		
decode_p_viseme_segment()		
if(fap_group_mask[1] & 0x2)		
decode_p_expression_segment()		
} else {		
for (i=0; i<NFAP[group_number]; i++, j++) {		
if(fap_group_mask[group_number] & (1 << i)) {		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ELLIOTT
ANSI Store order #: [illegible] Downloaded: 7/16/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2150
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

decode_p_dc(dc_Q[j])		
decode_ac(ac_Q[j])		
}		
}		
}		
}		
}		

6.2.10.9 Decode viseme and expression

decode_i_viseme_segment(){	No. of bits	Mnemonic
viseme_segment_select1q[0]	8	uimsbf
viseme_segment_select2q[0]	8	uimsbf
viseme_segment_blendq[0]	8	uimsbf
viseme_segment_def[0]	1	bslbf
for (k=1; k<16; k++) {		
viseme_segment_select1q_diff[k]		vlclbf
viseme_segment_select2q_diff[k]		vlclbf
viseme_segment_blendq_diff[k]		vlclbf
viseme_segment_def[k]	1	bslbf
}		
}		

decode_p_viseme_segment(){	No. of bits	Mnemonic
for (k=0; k<16; k++) {		
vseme_segment_select1q_diff[k]		vlclbf
vseme_segment_select2q_diff[k]		vlclbf
vseme_segment_blendq_diff[k]		vlclbf
vseme_segment_def[k]	1	bslbf
}		
}		

decode_i_expression_segment(){	No. of bits	Mnemonic
expression_segment_select1q[0]	8	uimsbf
expression_segment_select2q[0]	8	uimsbf
expression_segment_intensity1q[0]	8	uimsbf
expression_segment_intensity2q[0]	8	uimsbf
expression_segment_init_face[0]	1	bslbf
expression_segment_def[0]	1	bslbf
for (k=1; k<16; k++) {		
expression_segment_select1q_diff[k]		vlclbf
expression_segment_select2q_diff[k]		vlclbf
expression_segment_intensity1q_diff[k]		vlclbf
expression_segment_intensity2q_diff[k]		vlclbf
expression_segment_init_face[k]	1	bslbf

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FOUDAILY ADMIN/MR
ANSI Store order #: 0174457 Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2151
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

expression_segment_deff[k]	1	bslbf
}		
}		

decode_p _expression_segment(){	No. of bits	Mnemonic
for (k=0;k<16; k++) {		
expression_segment_select1q_diff[k]		vlclbf
expression_segment_select2q_diff[k]		vlclbf
expression_segment_intensity1q_diff[k]		vlclbf
expression_segment_intensity2q_diff[k]		vlclbf
expression_segment_init_face[k]	1	bslbf
expression_segment_deff[k]	1	bslbf
}		
}		

decode_i_dc(dc_q) {	No. of bits	Mnemonic
dc_q	16	simsbf
if(dc_q == -256*128)		
dc_q	31	simsbf
}		

decode_p_dc(dc_q_diff) {	No. of bits	Mnemonic
dc_q_diff		vlclbf
dc_q_diff = dc_q_diff* 256		
if(dc_q_diff == -256)		
dc_q_diff	16	simsbf
if(dc_Q == 0-256*128)		
dc_q_diff	32	simsbf
}		

decode_ac(sc_Q[i]) {	No. of bits	Mnemonic
this = 0		
next = 0		
while(next < 15) {		
count_of_runs		vlclbf
if (count_of_runs == 15)		
next = 16		
else {		
next = this+1+count_of_runs		
for (n=this+1; n<next; n++)		
ac_q[i][n] = 0		
sc_q[i][next]		vlclbf
if(ac_q[i][next] == 255)		
decode_i_dc(ac_q[i][next])		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG Y SUH
ANSI Store order #: 1-2744 / Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

else		
ac_q[i][next] = ac_q[i][next]-256		
this = next		
}		
}		
}		

6.2.10.10 Decode bap min max

decode_bap_new_minmax() {	No. of bits	Mnemonic
if (bap_is_i_new_max) {		
for (group_number=1;group_number<= BAP_NUM_GROUPS; goup_number++)		
for (i=0; i < NBAP_GROUP[group_number]; i++) {		
j=SAFS_IN_GROUP[group_number][i]		
if !(i & 0x3))		
marker_bit	1	uimsbf
if (bap_group_mask[group_number] & (1 <<i))		
bap_i_new_max[j]	5	uimsbf
}		
if (bap_is_i_new_min) {		
for (group_number = 1; group_number <= BAP_NUM_GROUPS; goup_number++)		
for (i=0; i < NBAP_GROUP[group_number]; i++) {		
j=SAFS_IN_GROUP[group_number][i]		
if !(i & 0x3))		
marker_bit	1	uimsbf
if (bap_group_mask[group_number] & (1 <<i))		
bap_i_new_min[j]	5	uimsbf
}		
if (bap_is_p_new_max) {		
for (group_number = 1; group_number <= BAP_NUM_GROUPS; goup_number++)		
for (i=0; i < NBAP_GROUP[group_number]; i++) {		
j=SAFS_IN_GROUP[group_number][i]		
if !(i & 0x3))		
marker_bit	1	uimsbf
if (bap_group_mask[group_number] & (1 <<i))		
bap_p_new_max[j]	5	uimsbf
}		
if (bap_is_p_new_min) {		
for (group_number = 1; group_number <= BAP_NUM_GROUPS; goup_number++)		
for (i=0; i < NBAP_GROUP[group_number]; i++) {		
j=SAFS_IN_GROUP[group_number][i]		
if !(i & 0x3))		
marker_bit	1	uimsbf
if (bap_group_mask[group_number] & (1 <<i))		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ACADEMIC
ANSI Store order #: 0744457 Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

bap_p_new_mir[j]	5	simsbf
}		
}		
}		

6.2.10.11 Decode ibap

decode_ibap(){	No. of bits	Mnemonic
for (group_number = 1; group_number <= BAP_NUM_GROUPS; group_number++) {		
for (i= 0; i<NBAP_GROUP[group_number]; i++) {		
=BAPS_IN_GROUP[group_number][i]		
if(bap_group_mask[group_number] & (1 <<)) {		
aa_decode(ibap_Q[j],ibap_cum_freq[j])		
}		
}		
}		
}		

6.2.10.12 Decode pbap

decode_pbap() {	No. of bits	Mnemonic
for (group_number = 1; group_number <= BAP_NUM_GROUPS; group_number++) {		
for (i= 0; i<NBAP_GROUP[group_number]; i++) {		
j=BAPS_IN_GROUP[group_number][i]		
if(bap_group_mask[group_number] & (1 << i)) {		
aa_decode(pbap_diff[j], pbap_cum_freq[j])		
}		
}		
}		
}		

6.2.10.13 Decode bap i segment

decode_bap_i_segment(){	No. of bits	Mnemonic
for (group_number= 1; group_number = BAP_NUM_GROUPS; group_number++) {		
for(i=0; i<NBAP_GROUP[group_number]; i++) {		
if(bap_group_mask[group_number] & (1 <<)) {		
j=BAPS_IN_GROUP[group_number][i]		
decode_dc(dc_Q[j])		
decode_ac(ac_Q[j])		
}		
}		
}		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG X
ANSI Store order #X07445, Downloaded: 7/18/2007 3:06:21 PM EST
Single user license only. Copying and networking prohibited.

6.2.10.14 Decode bap p segment

decode_bap_p_segment(){	No. of bits	Mnemonic
for (group_number = 1; group_number <= BAP_NUM_GROUPS; group_number++) {		
for (i=0; i<NBAP_GROUP[group_number]; i++) {		
if(bap_group_mask[group_number] & (1 << i)) {		
j = BAPS_IN_GROUP[group_number][i]		
decode_p_dc(dc_Q[j])		
decode_ac(ac_Q[j])		
}		
}		
}		
}		

6.2.11 3D Mesh Object

6.2.11.1 3D_Mesh_Object

3D_Mesh_Object () {	No. of bits	Mnemonic
3D_MO_start_code	16	uimsbf
3D_Mesh_Object_Header()		
do {		
3D_Mesh_Object_Layer()		
} while (nextbits_bytealigned() == 3D_MOL_start_code)		
}		

6.2.11.2 3D_Mesh_Object_Header

3D_Mesh_Object_Header() {	No. of bits	Mnemonic
ccw	1	bslbf
convex	1	bslbf
solid	1	bslbf
creaseAngle	6	uimsbf
coord_header()		
normal_header()		
color_header()		
texCoord_header()		
ce_SNHC	1	bslbf
if (**ce_SNHC** == '1')		
ce_SNHC _header()		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC / ADAMMAN
ANSI Store order #X0744E Downloaded: 7/16/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2155
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.11.3 3D_Mesh_Object_Layer

3D_Mesh_Object_Layer() {	No. of bits	Mnemonic
3D_MOL_start_code	16	uimsbf
mol_id	8	uimsbf
if (ce_SNHC == '1') {		
ce_SNHC _n_vertices	24	uimsbf
ce_SNHC _n_triangles	24	uimsbf
ce_SNHC _n_edges	24	uimsbf
}		
if (mol_id == 0)		
3DMeshObject_Base_Layer()		
else		
3DMeshObject_Refinement_Layer()		
}		

6.2.11.4 3D_Mesh_Object_Base_Layer

3DmeshObject_Base_Layer()	No. of bits	Mnemonic
do {		
3D_MOBL_start_code	16	uimsbf
mobl_id	8	uimsbf
while (!byteAligned())		
one_bit	1	bslbf
qf_start()		
if (3D_MOBL_start_code == "partition_type_0"){		
do {		
connected_component()		
qf_decode(last_component, last_component_context)		vlclbf
} while (last_component == '0')		
}		
elseif (3D_MOBL_start_code == "partition_type_1") {		
vg_number=0		
do {		
vertex_graph()		
vg_number++		
qf_decode(has_stitches, has_stitches_context)		vlclbf
if (has_stitches == '1')		
stitches()		
qf_decode(codap_last_vg, codap_last_vg_context)		vlclbf
} while (codap_last_vg == '0')		
}		
elseif (3D_MOBL_start_code == "partition_type_2") {		
if(vg_number > 1)		
qf_decode(codap_vg_id)		vlclbf
qf_decode(codap_left_bloop_idx)		vlclbf
qf_decode(codap_right_bloop_idx)		vlclbf

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGA
ANSI Store order #: [illegible] Downloaded: 7/18/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2156
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

qf_decode(codap_bdry_pred)		vlclbf
triangle_tree()		
triangle_data()		
}		
} while (nextbits_bytealigned() == 3D_MOBL_start_code)		
}		

6.2.11.5 coord_header

coord_header() {	No. of bits	Mnemonic
coord_binding	2	uimsbf
coord_bbox	1	bslbf
if (coord_bbox == '1') {		
coord_xmin	32	bslbf
coord_ymin	32	bslbf
coord_zmin	32	bslbf
coord_size	32	bslbf
}		
coord_quant	5	uimsbf
coord_pred_type	2	uimsbf
if (coord_pred_type=="tree_prediction" || coord_pred_type=="parallelogram_prediction") {		
coord_nlambda	2	uimsbf
for (i=1; i<coord_nlambda; i++)		
coord_lambda	4-27	simsbf
}		
}		

6.2.11.6 normal_header

normal_header() {	No. of bits	Mnemonic
normal_binding	2	uimsbf
if (normal_binding != "not_bound") {		
normal_bbox	1	bslbf
normal_quant	5	uimsbf
normal_pred_type	2	uimsbf
if (normal_pred_type=="tree_prediction" || normal_pred_type=="parallelogram_prediction") {		
normal_nlambda	2	uimsbf
for (i=1; i<normal_nlambda; i++)		
normal_lambda	3-17	simsbf
}		
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FOUDA Y. ABOUKHADRA
ANSI Store order #1274457 Downloaded 7/16/2007 2:06:21 PM ET
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2157
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.11.7 color_header

color_header() {	No. of bits	Mnemonic
color_binding	2	uimsbf
if (color_binding != "not_bound") {		
color_bbox	1	bslbf
if (color_bbox == "1") {		
color_rmin	32	bslbf
color_gmin	32	bslbf
color_bmin	32	bslbf
color_size	32	bslbf
}		
color_quant	5	uimsbf
color_pred_type	2	uimsbf
if (color_pred_type=="tree_prediction" || color_pred_type=="parallelogram_prediction") {		
color_nlambda	2	uimsbf
for (i=1; i<color_nlambda; i++)		
color_lambda	4..9	simsbf
}		
}		
}		

6.2.11.8 texCoord_header

texCoord_header() {	No. of bits	Mnemonic
texCoord_binding	2	uimsbf
if (texCoord_binding != "not_bound") {		
texCoord_bbox	1	bslbf
if (texCoord_bbox == "1") {		
texCoord_umin	32	bslbf
texCoord_vmin	32	bslbf
texCoord_size	32	bslbf
}		
texCoord_quant	5	uimsbf
texCoord_pred_type	2	uimsbf
if (texCoord_pred_type=="tree_prediction" || texCoord_pred_type=="parallelogram_prediction") {		
texCoord_nlambda	2	uimsbf
for (i=1; i<texCoord_nlambda; i++)		
texCoord_lambda	4..9	simsbf
}		
}		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #1121445, Downloaded: 2018-03-21 08:57
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2158
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.11.9 ce_SNHC_header

ce_SNHC_header() {	No. of bits	Mnemonic
ce_SNHC_n_proj_surface_spheres	8	uimsbf
if (ce_SNHC_n_proj_surface_spheres != 0) {		
ce_SNHC_x_coord_center_point	32	bslbf
ce_SNHC_y_coord_center_point	32	bslbf
ce_SNHC_z_coord_center_point	32	bslbf
ce_SNHC_normalized_screen_distance_factor	9	uimsbf
for (j=0; j< ce_SNHC_n_proj_surface_spheres; j++) {		
ce_SNHC_radius	32	bslbf
ce_SNHC_min_proj_surface	32	bslbf
ce_SNHC_n_proj_points	8	uimsbf
for (j=0; j< ce_SNHC_n_proj_points; j++) {		
ce_SNHC_sphere_point_coord	11	uimsbf
ce_SNHC_proj_surface	32	bslbf
}		
}		
}		
}		

6.2.11.10 connected_component

connected_component() {	No. of bits	Mnemonic
vertex_graph()		
qf_decode(has_stitches, has_stitches_context)		vlclbf
if (has_stitches == '1')		
stitches()		
triange_tree()		
triange_data()		
}		

6.2.11.11 vertex_graph

vertex_graph() {	No. of bits	Mnemonic
qf_decode(vg_simple, vg_simple_context)		vlclbf
depth = 0		
code_last = '1'		
openloops = 0		
do {		
do {		
if (code_last == '1') {		
qf_decode(vg_last, vg_last_context)		vlclbf
if (openloops > 0) {		
qf_decode(vg_forward_run, vg_forward_run_context)		vlclbf
if (vg_forward_run == 0') {		
openloops--		

© ISO/IEC 2003 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCED
ANSI Store order #1274457 Downloaded 7/18/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2159
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

if (openloops > 0)		
qf_decode(**vg_loop_index**, vg_loop_index_context)		vlclbf
break		
}		
}		
}		
qf_decode(**vg_run_length**, vg_run_length_context)		vlclbf
qf_decode(**vg_leaf**, vg_leaf_context)		vlclbf
if (**vg_leaf** == '1' && **vg_simple** == '0') {		
qf_decode(**vg_loop**, vg_loop_context)		vlclbf
if (**vg_loop** == '1')		
openloops++		
}		
} while (0)		
if (**vg_leaf** == '1' && **vg_last** == '1' && code_last == '1')		
depth--		
if (**vg_leaf** == '0' && (**vg_last** == '0' || code_last == '0'))		
depth++		
code_last = **vg_leaf**		
} while (depth >= 0)		
}		

6.2.11.12 stitches

stitches() {	No. of bits	Mnemonic
for each vertex in connected_component {		
qf_decode(**stitch_cmd**, stitch_cmd_context)		vlclbf
if (**stitch_cmd**) {		
qf_decode(**stitch_pop_or_get**, stitch_pop_or_get_context)		vlclbf
if (**stitch_pop_or_get** == '1') {		
qf_decode(**stitch_pop**, stitch_pop_context)		vlclbf
qf_decode(**stitch_stack_index**, stitch_stack_index_context)		vlclbf
qf_decode(**stitch_incr_length**, stitch_incr_length_context)		vlclbf
if (**stitch_incr_length** != 0)		
qf_decode(**stitch_incr_length_sign**, stitch_incr_length_sign_context)		vlclbf
qf_decode(**stitch_push**, stitch_push_context)		vlclbf
if (total length >0)		
qf_decode(**stitch_reverse**, stitch_reverse_context)		vlclbf
}		
else		
qf_decode(**stitch_length**, stitch_length_context)		vlclbf
}		
}		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS PATENT
ANSI Store order #: 1744457 Downloaded: 7/18/2007 2:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2160
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.11.13 triangle_tree

triangle_tree() {	No. of bits	Mnemonic
depth= 0		
ntriangles = 0		
branch_position = -2		
do {		
qf_decode(**tt_run_length**, tt_run_length_context)		vlclbf
ntriangles += **tt_run_length**		
qf_decode(**tt_leaf**, tt_leaf_context)		vlclbf
if (**tt_leaf** == "1") {		
depth--		
}		
else {		
branch_position = ntriangles		
depth++		
}		
} while (depth >= 0)		
if (**3D_MCBL_start_code** == "partition_type_2")		
if (**codap_right_bloop_idx** - **codap_left_bloop_idx** - 1 > ntriangles) {		
if (branch_position == ntriangles – 2) {		
qf_decode(**codap_branch_len**, codap_branch_len_context)		vlclbf
ntriangles -= 2		
}		
else		
ntriangles--		
}		
}		

6.2.11.14 triangle_data

triangle_data() {	No. of bits	Mnemonic
qf_decode(**triangulated**, triangulated_context)		vlclbf
depth=0		
root_triangle()		
for (i=1; i<ntriangles; i++)		
triangle(i)		
}		

6.2.11.15 root_triangle

root_triangle() {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(**marching_pattern**, marching_pattern_context[**marching_pattern**])		vlclbf
else {		
if (**3D_MCBL_start_code** == "partition_type_1")		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCED
ANSI Store order #: [illegible] Downloaded: 7/16/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

if (tt_leaf == '0' && depth==0)		
qf_decode(td_orientation, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
}		
if (3D_MOBL_start_code == "partition_type_2")		
if (triangulated == '0')		
qf_decode(polygon_edge, polygon_edge_context(polygon_edge))		vlclbf
root_coord()		
root_normal()		
root_color()		
root_texCoord()		
}		

root_coord(){	No. of bits	Mnemonic
if (3D_MOBL_start_code == "partition_type_2") {		
if (visited[vertex_index] == 0) {		
root_coord_sample()		
if (visited[vertex_index] == 0) {		
coord_sample()		
coord_sample()		
}		
}		
}		
else {		
root_coord_sample()		
coord_sample()		
coord_sample()		
}		
}		

root_normal() {	No. of bits	Mnemonic
if (normal_binding != "not_bound")		
if (3D_MOBL_start_code == "partition_type_2"){		
if (normal_binding != "bound_per_vertex" || visited[vertex_index] == 0) {		
root_normal_sample()		
if (normal_binding != "bound_per_face" && (normal_binding != "bound_per_vertex" || visited[vertex_index] == 0)) {		
normal_sample()		
normal_sample()		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PROQUEST
ANSI Store order #: 0134457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2162
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

}		
else {		
root_normal_sample()		
if (**normal_binding** != "bound_per_face"){		
normal_sample()		
normal_sample()		
}		
}		
}		

root_color() {	No. of bits	Mnemonic
if (**color_binding** != "not_bound")		
if (**3D_MOBL_start_code** == "partition_type_I") {		
if (**color_binding** != "bound_per_vertex" || visited[vertex_index] == 0) {		
root_color_sample()		
if (**color_binding** != "bound_per_face" && (**color_binding** != "bound_per_vertex" || visited[vertex_index] == 0)) {		
color_sample()		
color_sample()		
}		
}		
else {		
root_color_sample()		
if (**color_binding** != "bound_per_face") {		
color_sample()		
color_sample()		
}		
}		
}		

root_texCoord() {	No. of bits	Mnemonic
if (**texCoord_binding** != "not_bound")		
if (**3D_MOBL_start_code** == "partition_type_I") {		
if (**texCoord_binding** != "bound_per_vertex" || visited[vertex_index] == 0) {		
root_texCoord_sample()		
if (**texCoord_binding**!= "bound_per_vertex" || visited[vertex_index] == 0) {		
texCoord_sample()		
texCoord_sample()		
}		
}		
else {		
root_texCoord_sample()		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FOUJA V. ADAMANIE
ANSI Store order #: 07/44 67 Downloaded: 7/16/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

texCoord_sample()		
texCoord_sample()		
}		
}		

6.2.11.16 triangle

triangle() {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(marching_edge, marching_edge_context[marching_edge])		vlclbf
else {		
if (3D_MOBL_start_code == "partition_type_2")		
if (tt_leaf == '0' && depth==0)		
qf_decode(td_orientation, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
if (triangulated == '0')		
qf_decode(polygon_edge, polygon_edge_context[polygon_edge])		vlclbf
coord()		
normal()		
color()		
texCoord()		
}		

coord() {	No. of bits	Mnemonic
if (3D_MOBL_start_code == "partition_type_2") {		
if (visited[vertex_index] == 0)		
if (no_ancestors)		
root_coord_sample()		
else		
coord_sample()		
}		
else {		
if (visited[vertex_index] == 0)		
coord_sample()		
}		
}		

normal() {	No. of bits	Mnemonic
if (normal_binding == "bound_per_vertex") {		
if (3D_MOBL_start_code == "partition_type_2"){		
if (visited[vertex_index] == 0)		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... (2002/2)
ANSI Store order #: ... Downloaded: 7/18/2007 3:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

if (no_ancestors)		
root_normal_sample()		
else		
normal_sample()		
}		
else {		
if (visited[vertex_index] == 0)		
normal_sample()		
}		
} else if (**normal_binding** == "bound_per_face") {		
if (**triangulated** == '1' || **polygon_edge** == '1')		
normal_sample()		
} else if (**normal_binding** == "bound_per_corner") {		
if (**triangulated** == '1' || **polygon_edge** == '1') {		
normal_sample()		
normal_sample()		
}		
normal_sample()		
}		
}		

color() {	No. of bits	Mnemonic
if (**color_binding** == "bound_per_vertex") {		
if (**3D_MOBL_start_code** == "partition_type_2") {		
if (visited[vertex_index] == 0)		
if (no_ancestors)		
root_color_sample()		
else		
color_sample()		
}		
else {		
if (visited[vertex_index] == 0)		
color_sample()		
}		
} else if (**color_binding** == "bound_per_face") {		
if (**triangulated** == '1' || **polygon_edge** == '1')		
color_sample()		
} else if (**color_binding** == "bound_per_corner") {		
if (**triangulated** == '1' || **polygon_edge** == '1') {		
color_sample()		
color_sample()		
}		
color_sample()		
}		
}		

© ISO/IEC 2001 – All rights reserved

Downloaded by UNIVERSITY OF TEXAS AT ARLINGTON/LIBRARY ACQUISITIONS under ANSI Store order #1031445 Downloaded: 7/16/2007 2:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

texCoord() {	No. of bits	Mnemonic
if (**texCoord_binding** == "bound_per_vertex") {		
if (**3D_MOBL_start_code** == "partition_type_2"){		
if (visited[vertex_index] == 0)		
if (no_ancestors)		
root_texCoord_sample()		
else		
texCoord_sample()		
}		
else{		
if (visited[vertex_index] == 0)		
texCoord_sample()		
}		
} else if **texCoord_binding** == "bound_per_corner" {		
if (**triangulated** == "1" || **polygon_edge** == "1") {		
texCoord_sample()		
texCoord_sample()		
}		
texCoord_sample()		
}		
}		

root_coord_sample() {	No. of bits	Mnemonic
for (i=0; i<3; i++)		
for (=0; i<**coord_quant**; i++)		
gf_decode(**coord_bit**, zero_context)		vlclbf
}		

root_normal_sample() {	No. of bits	Mnemonic
for (i=0; i<1; i++)		
for (=0; i<**normal_quant**; i++)		
gf_decode(**normal_bit**, zero_context)		vlclbf
}		

root_color_sample() {	No. of bits	Mnemonic
for (i=0; i<3; i++)		
for (=0; i<**color_quant**; i++)		
gf_decode(**color_bit**, zero_context)		vlclbf
}		

root_texCoord_sample() {	No. of bits	Mnemonic
for (i=0; i<2; i++)		
for (=0; i<**texCoord_quant**; i++)		
gf_decode(**texCoord_bit**, zero_context)		vlclbf
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG A. YOUNG
ANSI Store order #: 01744/5, Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

coord_sample() {	No. of bits	Mnemonic
for (i=0; i<3; i++) {		
j=0		
do {		
qf_decode(coord_leading_bit, coord_leading_bit_context[3*j+i])		vlclbf
j++		
} while (j<coord_quant && coord_leading_bit == '0')		
if (coord_leading_bit == '1') {		
qf_decode(coord_sign_bit, zero_context)		vlclbf
do {		
qf_decode(coord_trailing_bit, zero_context)		vlclbf
} while (j<coord_quant)		
}		
}		
}		

normal_sample() {	No. of bits	Mnemonic
for (i=0; i<1; i++) {		
j=0		
do {		
qf_decode(normal_leading_bit, normal_eading_bit_context[j])		vlclbf
j++		
} while (j<normal_quant && normal_leading_bit == '0')		
if (normal_leading_bit == '1') {		
qf_decode(normal_sign_bit, zero_context)		vlclbf
do {		
qf_decode(normal_trailing_bit, zero_context)		vlclbf
} while (j<normal_quant)		
}		
}		
}		

color_sample() {	No. of bits	Mnemonic
for (i=0; i<3; i++) {		
j=0		
do {		
qf_decode(color_leading_bit, color_leading_bit_context[3*j+i])		vlclbf
j++		
} while (j<color_quant && color_leading_bit== '0')		
if (color_leading_bit == '1') {		
qf_decode(color_sign_bit, zero_context)		vlclbf
do {		
qf_decode(color_trailing_bit, zero_context)		vlclbf
} while (j<color_quant)		
}		
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUAA Y. AGARWAL
ANSI Store order #1274457 Downloaded 7/18/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2167
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

texCoord_sample() {	No. of bits	Mnemonic
for (i=0; i<2; i++) {		
i=0		
do {		
gf_decode(**texCoord_leading_bit**, texCoord_leading_bit_context[2*j+i])		vlclbf
++		
} while (j<**texCoord_quant** && **texCoord_leading_bit** == '0')		
if (**texCoord_leading_bit** == '1') {		
gf_decode(**texCoord_sign_bit**, zero_context)		vlclbf
do {		
gf_decode(**texCoord_trailing_bit**, zero_context)		vlclbf
} while (j<**texCoord_quant**)		
}		
}		
}		

6.2.11.17 3DMeshObject_Refinement_Layer

3DMeshObject_Refinement_Layer () {	No. of bits	Mnemonic
do {		
3D_MORL_start_code	16	uimsbf
mor'_id	8	uimsbf
connectivity_update	2	uimsbf
pre_smoothing	1	bslbf
if(**pre_smoothing** == '1')		
pre_smoothing_parameters()		
post_smoothing	1	bslbf
if(**post_smoothing** == '1')		
post_smoothing_parameters()		
while (!bytealigned())		
pre_bit	1	bslbf
gf_start()		
if(**connectivity_update** == "fs_update")		
fs_pre_update()		
if(**pre_smoothing** == '1' || **post_smoothing** == '1')		
smoothing_constraints()		
/* apply pre smoothing step */		
if(**connectivity_update** == "fs_update")		
fs_post_update()		
if(**connectivity_update** != "not_updated")		
gf_decode(**other_update**, zero_context)		vlclbf
if(**connectivity_update**=="not_updated" || **other_update** == '1')		
other_property_update()		
/* apply post smoothing step */		
} while (nextbits_bytealigned() == **3D_MORL_start_code**)		
}		

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUG YOUNG
ANSI Store order #: 07/34457 Downloaded: 7/18/2007 3:08:21 PM EST
Single user license only. Copying and networking prohibited.

6.2.11.17.1 pre_smoothing_parameters

pre_smoothing_parameters() {	No. of bits	Mnemonic
pre_smoothing_n	8	uimsbf
pre_smoothing_lambda	32	bslbf
pre_smoothing_mu	32	bslbf
}		

6.2.11.17.2 post_smoothing_parameters

post_smoothing_parameters() {	No. of bits	Mnemonic
post_smoothing_n	8	uimsbf
post_smoothing_lambda	32	bslbf
post_smoothing_mu	32	bslbf
}		

6.2.11.17.3 fs_pre_update

fs_pre_update() {	No. of bits	Mnemonic
for each connected component {		
forest()		
for each tree in forest {		
triangle_tree()		
/* for each tree loop vertex set visited[vertex_index]='1' */		
triangle_data()		
}		
}		
}		

forest() {	No. of bits	Mnemonic
for each edge in connected component		
if (creates no loop in forest)		
qf_decode(pfs_forest_edge, pfs_forest_edge_context)		vlclbf
}		

6.2.11.17.4 smoothing_constraints

smoothing_constraints() {	No. of bits	Mnemonic
qf_decode(smooth_with_sharp_edges, zero_context)		vlclbf
if (smooth_with_sharp_edges == '1')		
sharp_edge_marks()		
qf_decode(smooth_with_fixed_vertices, zero_context)		vlclbf
if (smooth_with_fixed_vertices == '1')		
fixed_vertex_marks()		
}		

© ISO/IEC 2003 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODAAC, ADMINISTRATOR
ANSI Store order #: 07/44457 Downloaded: 7/16/2007 2:06:21 PM EST
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2169
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

sharp_edge_marks () {	No. of bits	Mnemonic
for each edge		
sf_decode(smooth_sharp_edge, smooth_sharp_edge_context)		vlclbf
}		

fixed_vertex_marks () {	No. of bits	Mnemonic
for each vertex		
sf_decode(smooth_fixed_vertex, smooth_fixed_vertex_context)		vlclbf
}		

6.2.11.17.5 fs_post_update

fs_post_update() {	No. of bits	Mnemonic
for each connected component {		
for each tree in forest		
tree_loop_property_update()		
}		

tree_loop_property_update () {	No. of bits	Mnemonic
loop_coord_update()		
loop_normal_update()		
loop_color_update()		
loop_texCoord_update()		
}		

loop_coord_update () {	No. of bits	Mnemonic
for each tree loop vertex		
coord_sample()		
}		

loop_normal_update () {	No. of bits	Mnemonic
if (normal_binding == "bound_per_vertex") {		
for each tree loop vertex		
normal_sample()		
}		
else if (normal_binding == "bound_per_face") {		
for each tree loop face		
normal_sample()		
}		
else if (normal_binding == "bound_per_corner") {		
for each tree loop corner		
normal_sample()		
}		
}		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... 2006/02/21
ANSI Store order #: ... Downloaded: ... (ISO/IEC 14496-2:2001 PR 8)
Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2170
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

loop_color_update () {	No. of bits	Mnemonic
if (**color_binding** == "bound_per_vertex") {		
for each tree loop vertex		
color_sample()		
}		
else if (**color_binding** == "bound_per_face") {		
for each tree loop face		
color_sample()		
}		
else if (**color_binding** == "bound_per_corner") {		
for each tree loop corner		
color_sample()		
}		
}		

loop_texCoord_update () {	No. of bits	Mnemonic
if (**texCoord_binding** == "bound_per_vertex") {		
for each tree loop vertex		
texCoord_sample()		
}		
else if (**texCoord_binding** == "bound_per_corner") {		
for each tree loop corner		
texCoord_sample()		
}		
}		

6.2.11.17.6 other_property_update

other_property_update() {	No. of bits	Mnemonic
other_coord_update()		
other_normal_update()		
other_color_update()		
other_texCoord_update()		
}		

other_coord_update () {	No. of bits	Mnemonic
for each vertex in mesh		
if (vertex is not a tree loop vertex)		
coord_sample()		
}		

other_normal_update () {	No. of bits	Mnemonic
if (**normal_binding** == "bound_per_vertex") {		
for each vertex in mesh		

© ISO/IEC 2003 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. V. ADAMIAK
ANSI Store order #X07445, Downloaded 7/16/2007 2:08:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2171
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

if (vertex is not a tree loop vertex)		
normal_sample()		
}		
else if (normal_binding == "bound_per_face") {		
for each face in mesh		
if (face is not a tree loop face)		
normal_sample()		
}		
else if (normal_binding == "bound_per_corner") {		
for each corner in mesh		
if (corner is not a tree loop corner)		
normal_sample()		
}		
}		

other_color_update () {	No. of bits	Mnemonic
if (color_binding == "bound_per_vertex") {		
for each vertex in mesh		
if (vertex is not a tree loop vertex)		
color_sample()		
}		
else if (color_binding == "bound_per_face") {		
for each face in mesh		
if (face is not a tree loop face)		
color_sample()		
}		
else if (color_binding == "bound_per_corner") {		
for each corner in mesh		
if (corner is not a tree loop corner)		
color_sample()		
}		
}		

other_texCoord_update () {	No. of bits	Mnemonic
if (texCoord_binding == "bound_per_vertex") {		
for each vertex in tree loop		
if (vertex is not a tree loop vertex)		
texCoord_sample()		
}		
else if (texCoord_binding == "bound_per_corner") {		
for each corner in mesh		
if (corner is not a tree loop corner)		
texCoord_sample()		
}		
}		

© ISO/IEC 2001 - All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS
ANSI Store order #: [illegible] Downloaded: [illegible]
Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2172
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.2.12 Upstream message

6.2.12.1 upstream_message

upstream_message() {	No. of bits	Mnemonic
upstream_message_type	3	bslbf
if (upstream_message_type == Video_NEWPRED)		
upstream_Video_NEWPRED()		
if (upstream_message_type == SNHC_QoS)		
upstream_SNHC_QoS()		
byte_align_for_upstream()		
}		

6.2.12.2 upstream_Video_NEWPRED

upstream_Video_NEWPRED() {	No. of bits	Mnemonic
newpred_upstream_message_type	2	bslbf
if (newpred_upstream_message_type == "NP_NACK")		
unrelable_flag	1	bslbf
vop_id	4-15	uimsbf
macroblock_number	1-14	uimsbf
if (newpred_upstream_message_type == "Intra Refresh Command")		
end_macroblock_number	1-14	uimsbf
if (newpred_upstream_message_type == "NP_NACK")		
requested_vop_id_for_prediction	4-15	uimsbf
}		

6.2.12.3 upstream_SNHC_QoS

upstream_SNHC_QoS() {	No. of bits	Mnemonic
screen_width	12	uimsbf
screen_height	12	uimsbf
n_rendering_modes	4	uimsbf
for (rendering_mode = 0 ; rendering_mode < n_rendering_modes ; rendering_mode++) {		
rendering_mode_type	4	uimsbf
n_curves	8	uimsbf
for (i = 0 ; i < n_curves ; i++) {		
triangle_parameter	24	uimsbf
n_points_on_curve	8	uimsbf
for (j = 0 ; j < n_points_on_curve ; j++) {		
screen_coverage_parameter	8	uimsbf
frame_rate_value	12	bslbf
}		
}		
}		
}		

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PRODUCT ADVANCED
ANSI Store order #: [illegible] Downloaded: 7/18/2007 3:06:21 PM EST
Single user licensee only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2173
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

6.3 Visual bitstream semantics

6.3.1 Semantic rules for higher syntactic structures

This subclause details the rules that govern the way in which the higher level syntactic elements may be combined together to produce a legal bitstream. Subsequent subclauses detail the semantic meaning of all fields in the video bitstream.

6.3.2 Visual Object Sequence and Visual Object

visual_object_sequence_start_code: The visual_object_sequence_start_code is the bit string '000001B0' in hexadecimal. It initiates a visual session.

profile_and_level_indication: This is an 8-bit integer used to signal the profile and level identification. The meaning of the bits is given in Table G-1.

visual_object_sequence_end_code: The visual_object_sequence_end_code is the bit string '000001B1' in hexadecimal. It terminates a visual session.

visual_object_start_code: The visual_object_start_code is the bit string '000001B5' in hexadecimal. It initiates a visual object.

is_visual_object_identifier: This is a 1-bit code which when set to '1' indicates that version identification and priority is specified for the visual object. When set to '0', no version identification or priority needs to be specified.

visual_object_verid: This is a 4-bit code which identifies the version number of the visual object. Its meaning is defined in Table 6-4. When this field does not exist, the value of visual_object_verid is '0001'.

Table 6-4 – Meaning of visual_object_verid

visual_object_verid	Meaning
0000	eserved
0001	bject type listed in Table 9-1
0010	bject type listed in Table V2 - 39
0011 - 1111	eserved

visual_object_priority: This is a 3-bit code which specifies the priority of the visual object. It takes values between 1 and 7, with 1 representing the highest priority and 7, the lowest priority. The value of zero is reserved.

visual_object_type: The visual_object_type is a 4-bit code given in Table 6-5 which identifies the type of the visual object.

Table 6-5 – Meaning of visual object type

code	risual object type
0000	eserved
0001	ideo ID
0010	still texture ID
0011	mesh ID
0100	FBA ID
0101	3D mesh ID
01101	eserved
:	
:	
1111	eserved

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOUGLAS ... 2005/02/21
ANSI Store order #: [illegible] Downloaded: [illegible]
Single user licence only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2174
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

video_object_start_code: The video_object_start_code is a string of 32 bits. The first 27 bits are '0000 0000 0000 0000 0000 0001 000' in binary and the last 5-bits represent one of the values in the range of '00000' to '11111' in binary. The video_object_start_code marks a new video object.

video_object_id: This is given by the last 5-bits of the video_object_start_code. The video_object_id uniquely identifies a video object.

video_signal_type: A flag which if set to '1' indicates the presence of video_signal_type information.

video_format: This is a three bit integer indicating the representation of the pictures before being coded in accordance with this part of ISO/IEC 14496. Its meaning is defined in Table 6-6. If the video_signal_type() is not present in the bitstream then the video format may be assumed to be "Unspecified video format".

Table 6-6 -- Meaning of video_format

video_format	Meaning
000	Component
001	PAL
010	NTSC
011	SECAM
100	MAC
101	Unspecified video format
110	Reserved
111	Reserved

video_range: This one-bit flag indicates the black level and range of the luminance and chrominance signals.

colour_description: A flag which if set to '1' indicates the presence of colour_primaries, transfer_characteristics and matrix_coefficients in the bitstream.

colour_primaries: This 8-bit integer describes the chromaticity coordinates of the source primaries, and is defined in Table 6-7.

© ISO/IEC 2001 – All rights reserved

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/TODD A. ADAMSON
ANSI Store order #X37445. Downloaded 7/16/2007 3:06:21 PM ET
Single user license only. Copying and networking prohibited.

Netflix, Inc. and Hulu, LLC - Ex. 1002, Page 2175
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)