Video Streaming Primer

Christopher Benes, CTO, rVibe

This document is a primer on video streaming. It describes the video streaming process, video compression (encoding) and decompression (decoding), high and standard definition (HD and SD) video, progressive and interlaced video, analog and digital video, and common settings to encode video for live and Video on Demand (VoD) streaming using the ITU-T H.264 / MPEG-4 Part 10 Advanced Video Coding (AVC) standard.

What is video streaming?

Video streaming is the act of transmitting compressed (typically) video across a private or public network (like the Internet, a LAN, satellite, or cable television) to be uncompressed and played on a computing device (such as a TV, smart phone or computer).

Why compress video?

Video is compressed in size to save transmission bandwidth and storage space. Compression reduces original video information and removes redundancies. Uncompressed video requires a tremendous amount of bandwidth to transmit across a network or space to store.

The table below illustrates the advantages of compression.

RESOLUTION	FORMAT	UNCOMPRESSED	COMPRESSED	APPLICATION
720x480	480i 29.97fps	124.29 Mbps	2.49 Mbps	DVD movies
1,280x720	720p30	663 Mbps	13.26 Mbps	Broadcasting
1,280x720	720p60	1.66 Gbps	33.2 Mbps	Sports broadcasting
1,920x1,080	1080i60	1.49 Gbps	29.8 Mbps	Broadcasting
1,920x1,080	1080p24	1.19 Gbps	25 Mbps	Blu-Ray movies

TABLE 1. UNCOMPRESSED VS COMPRESSED VIDEO

It compares the streaming bit-rate of uncompressed video with that of compressed video assuming a YUV 4:2:0 color space for SD video and YUV 4:4:4 color space for HD video, both using a 8-bit color depth. An H.264[1] encoder can achieve a wide range of compression ratios depending on a number of fac-

[1] H.264 is a standard for video compression. The ITU-T (International Telecommunications Union) and ISO/IEC MPEG (Motion Picture Experts Group) standards bodies jointly defined and maintain the standard. It is occasionally referred to as MPEG-4 Part 10 or AVC (Advanced Video Coding).

tors, including the video itself and target bit rate. The table below uses a compression ratio of 50:1 which may be too low depending on target bit rate. A compression ratio of 1000:1 may actually be required.

How does VoD streaming differ from live streaming?

Streaming video can either be *live* or *VoD*. With live streaming, source video is captured and encoded in *real-time*. This requires more computer processing, network resources and efficiency (lower latency during capture, encoding and stream delivery) compared to VoD streaming. With VoD streaming, source video is previously encoded and then streamed removing the need to capture, preprocess, convert and encode the video in real-time.

VoD encoding should utilize better (but more time consuming) encoding techniques (such as two-pass Variable Bit Rate) that increase quality whenever possible. These encoding techniques are typically not used for live streaming since the extra encoding time increases overall latency (delay). Since the size and location of VoD files are known, delivery of VoD files across a computer network can be optimized to further reduce delivery latency.

What is the video streaming process and what resources are required to stream?

The video streaming process consists of a number of steps and requires both hardware, software and network resources. Table 2 below summarizes the process and the required equipment.

STEP	DESCRIPTION	REQUIRED HARDWARE AND/OR SOFTWARE
Capture	Obtain source analog or digital video and audio from cameras, video mixers, etc.	Video cameras, video mixer, microphones
Conversion	Convert analog source video to digital; no conversion required for digital source video	Video capture card such as one from Digital Rapids, Aja, ViewCast, Blackmagic, etc.
Preprocess	Remove noise, de-interlace interlaced video, etc. for more efficient encoding	Video capture card from Digital Rapids
Encode	Encode and compress digital video	A hardware or software encoder

STEP	DESCRIPTION	REQUIRED HARDWARE AND/OR SOFTWARE
Transmission	Transmit the compressed video across a computer network for multiple viewers with varying bandwidth conditions	Network bandwidth and possibly a scalable delivery system
Decode	Decode and uncompress digital video so it can be rendered	A hardware or software decoder
Presentation	Render (play) the digital video (on a TV, mobile device or computer)	A video player such as Flash, Windows Media or HTML 5

TABLE 2. VIDEO STREAMING PROCESS AND REQUIREMENTS

What are the network and bandwidth requirements for live streaming?

Certain streaming platforms - such as those from rVibe - don't require the reconfiguration of existing networking infrastructure devices (routers, switches and gateways) or for security controls to be changed or bypassed. If Flash streaming is used, the ports Flash uses (80, 1935, 443) must be open on infrastructure routers and gateways. In most corporate environments this is already the case since many software applications used by businesses include Flash. HTTP streaming (Apple HTTP D Streaming, Adobe Flash Dynamic Streaming, Microsoft Smooth Streaming with Silverlight) does not have this requirement unless firewall packet inspection inhibits smooth traffic flow of multimedia packets. That must be addressed.

Minimally, it is recommended that between 650 Kbps and 1 Mbps **sustained** upload network bandwidth be available at an encoding location to stream a single high quality 512x384 or 768x432 resolution video. More bandwidth is required to encode multiple streams of different quality to allow users with varying bandwidth conditions to watch a stream that best matches their local conditions.

At the receiving (or viewing) location, that same amount of **sustained** download bandwidth (650 Kbps to 1 Mbps) is required by viewers to avoid excessive buffering, choppiness, dropped frames, and freezing of the video stream. Viewers should minimize the number of running applications (such as virus checkers) on their devices so the maximum amount of computing resources (CPU cycles and memory) can be allocated to the timely decoding of the video stream.

Play back devices must be powerful enough to decode the video stream at the broadcast bit rate. Extensive buffering and dropped frames can otherwise result.

When streaming over the Internet to viewers across the country or world, a Content Delivery Network (CDN) - such as that used by rVibe - should be used. A CDN allows scalable delivery to tens of thousands (even hundreds of thousands) of viewers with no intervention required at the encoding location. rVibe's streaming platform can be configured (on request) to not use a CDN. In this case, streaming media servers must be accessed on rVibe's network or installed and accessed within a LAN.

Does video streaming work over a VPN and corporate LAN?

Yes. However, a VPN and LAN have a bandwidth cap that will limit the total number of simultaneous viewers. The VPN or LAN must, per above, allow streaming traffic over typical streaming ports. Normal network applications such as VoIP, email and web surfing will still work with concurrent video streaming but performance may be degraded based on the amount of network traffic generated by these applications, the number of viewers and bandwidth capacity of the VPN or LAN. Congestion can be mitigated by intelligently prioritizing and filtering certain types of traffic (streaming, VoIP) at routers over lower priority, non-time critical traffic (web surfing, email).

In order to more efficiently deliver streaming video within a LAN, multicast networking can be used. The networking equipment (routers, gateways, etc.) within a LAN must be configured to support multicast video distribution to host computers in the multicast group, a non-trivial task. Streaming video servers to distribute multicast video must be placed and configured within the LAN. Finally, the video player must be able to play a received multicast video stream.

What is a CODEC?

A CODEC[2] (COder-DECoder or COmpressor-DECompressor) is hardware or software that compresses (*encodes*) digital data (like digital video or audio) <u>and</u> uncompresses (*decodes*) the data back to its original form. A CODEC collectively refers to two components - an encoder and decoder.

CODECs used for live streaming must be very efficient in order to complete the encoding and decoding process fast enough to keep up with the rate of input video, minimize the size of the compressed output video, minimize latency and play back decoded frames fluidly.

Encoders can be *lossy* or *lossless*. A *lossless* encoder reduces, but does not loose, original audio or video information during the compression process. Uncompressed source video matches the original exactly but less space is used to transmit or store the compressed video. The reduction (and space savings) however is no where near as significant as that provided by *lossy* encoding. A *lossy* encoder destructively reduces original video and audio information during the compression process substantially. Modern *lossy* encoders, like H.264 or HE-AAC V2, can be implemented and configured so it is difficult for most people to observe a difference between source video and uncompressed video. The purpose of *lossy* encoders is

[2] Common video CODECs are H.264, VP 8 and VC-1.

to save considerable amounts of space while maintaining acceptable quality. Compression artifacts (jagged edges, moire patterns, motion blur, loss of detail) result when a *lossy* encoder destructively reduces video information so much so it becomes easily discernible to a viewer after the video is uncompressed. The challenge is to configure a *lossy* encoder to provide an acceptable level of quality while using an acceptable level of bandwidth or space. This balance is not always easily attainable since *lossy* encoders aren't aware when the reduction process starts to do more harm then good.

Encoders and decoders can be implemented in *hardware* (a chip or set of chips within a computing device) or *software*, such as an application running on a computing device, e.g. Adobe's *Flash Media Encoder Live* or the open source *x264*.

To be accurate, H.264 is a decoding standard. Software or hardware encoders used in combination with H.264 compliant decoders must encode source video so that any H.264 compliant decoder can decode it without issue.

What is a video frame?

Digital video consists of a sequence of pictures (frames) of video images in a particular aspect ratio (width/height) representing the source video at a point in time. Each frame consists of a set of pixels[3]. Each pixel contains information representing the color (e.g. red, green, blue or cyan, magenta, yellow and black) intensity of the video at a particular location in the frame. The number of pixels in a frame depends on the frame's resolution. Image 1 is an example of a video frame divided into groups of 8x8 pixels.

rVibe can stream compressed HD video over a LAN if bandwidth is available.

[3] A pixel is also known as a "sample."

IMAGE 1. VIDEO FRAME DIVIDED INTO GROUPS OF 8X8 PIXELS

What is Standard Definition video?

SD video is any video that is not HD. One of the most common SD video modes is:

- **480i** - 720x480 resolution @ 4:3 aspect ratio @ 24 frames per second (fps); there are 480 vertical lines of resolution in each interlaced video frame.

DVDs and non-HD television use 480i.

The highest resolution SD video frame has a height of 576 lines.

What is High Definition video?

HD video is any video that is not SD or any video with a frame height greater than 576 lines. Two of the most common HD video modes (and what is popularly considered HD) are:

- **720p** - 1280x720 resolution @ 16:9 aspect ratio @ 25/30/60 frames per second (fps); there are 720 vertical lines of resolution in each video frame; and
- **1080i/p** - 1920x1080 resolution @ 16:9 aspect ratio @ 25/30/60 frames per second (fps); there are 1080 vertical lines of resolution in each video frame.

It's important to recognize that resolution *alone* determines if a video is HD or SD. Not quality. Not frame rate. Not bit rate.

What is progressive and interlaced video?

SD and HD video is either “progressive” or “interlaced.”

Progressive HD video consists of frames with **each and every** pixel of a video image. It’s a complete picture of all lines in a video image. Someone looking at an individual progressive frame would see an entire video image. Image 1 is an example of a progressive frame. The ‘p’ in “720p” standards for “progressive.”

rVibe can encode and stream interlaced or progressive video.

Interlaced HD video consists of fields with **1/2** the pixels of a video frame. It’s a picture of the video frame containing every other vertical line in the frame - the even lines or the odd lines. Someone looking at an individual interlaced field would see 1/2 a video frame. Two interlaced fields are combined (the even and odd lines) to form one complete video frame with all its pixels. Image 2 below illustrates the concept of interlaced video. The ‘i’ in “1080i” standards for “interlaced.”

If an interlaced video mode mode is followed by a number such as 1080i60, the number stands for the field rate. Dividing by two gives the frame rate. E.g. 1080i60 has a frame rate of 30.

If a progressive video mode is followed by a number such as 720p30, the number stands for the frame rate. E.g. 720p30 has a frame rate of 30, 720p60 and 1080p60 have a frame rate of 60.

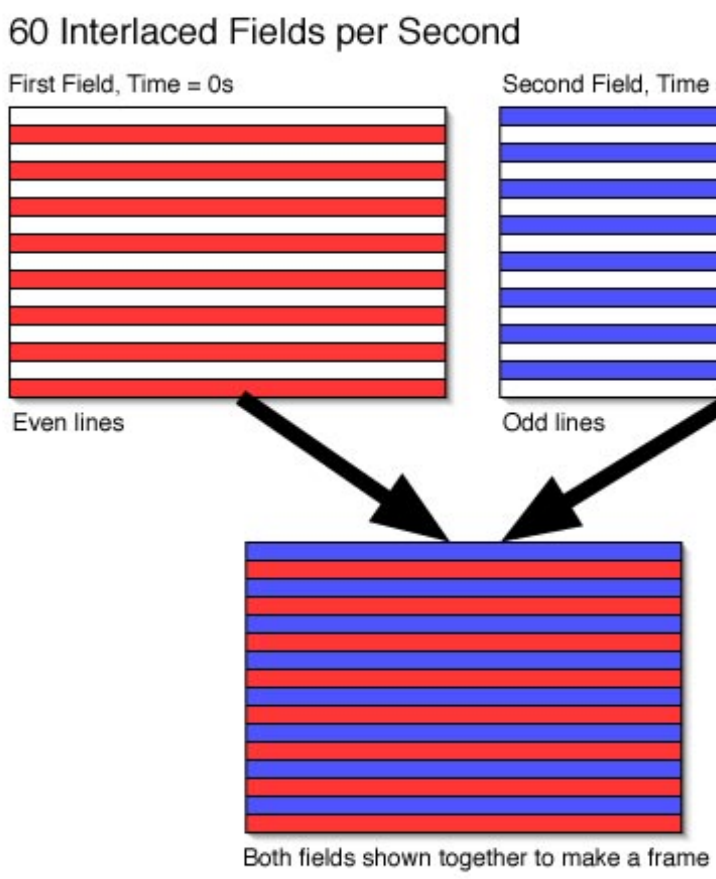

60 Interlaced Fields per Second
First Field, Time = 0s
Second Field, Time = 1/60s
Even lines
Odd lines
Both fields shown together to make a frame

IMAGE 2. INTERLACED VIDEO

When should progressive or interlaced video be used?

Progressive video captures high motion events more effectively since complete frames of video more fluidly represent motion and are compressed more efficiently by an encoder. ESPN HD for instance is in 720p60. Most HD television content is in 1080i. 720p30 and 1080i60 carry close to the same amount of information.

1080p is not used by any broadcast network or cable provider. Blue-Ray video is one source of 1080p content.

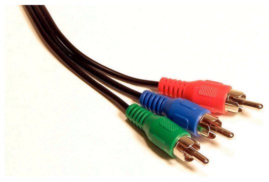
What is the difference between analog and digital video?

Video cameras generate output video in **analog**[4] or **digital**[5] SD or HD format. SD video can be in a digital output format (SDI). Analog (such as component) video can be HD in 720p, 1080i or 1080p. Therefore,

[4] Composite, S-Video and component are analog video formats. Composite is the worst quality, component the best.

[5] Serial Digital Interface (SDI) is an uncompressed digital video format used in professional applications.

either type of video - analog or digital - can be SD or HD. The first picture to the right is an analog video cable, the second a digital SDI cable w/BNC connectors.



Digital video is preferred over analog because:

- Digital video is less susceptible to noise and attenuation which degrades the video, especially over long cabling distances.
- Digital video is much easier to manipulate in an end to end workflow consisting of multiple digital video devices such as cameras, mixers, recorders and streaming computers.
- Synchronizing video to one point in time (common reference) across multiple pieces of equipment is much easier to do digitally than in the analog domain.

The main drawback of SDI digital video is cost. SDI cameras, mixers, cables, recorders and streaming computers are all more expensive (two to four times) than those that process analog video and HD-SDI more so. For instance, a SD analog video mixer with monitoring LCDs costs $3,000 to $5,000. A SD (not HD) digital mixer similar in capability costs between $7,000 upwards of $20,000. In part this is because of the rate this equipment must process digital information. SD SDI video has a rate of 270 Mbps. HD 720p or 1080i SDI video has a rate of 1.485 Gbps. HD 1080p SDI video is 2.9 Gbps. This is a massive amount of information to process (route, interpret and manipulate) and thus complicated (and expensive!) hardware and software are required to do so.

rVibe can encode analog component and S-Video and digital SDI and HD-SDI video.

High-Definition Multimedia Interface (HDMI) and Digital Visual Interface (DVI) are two other uncompressed digital video formats used primarily in consumer applications. HDMI and DVI equipment is much cheaper than SDI. Both SDI and HDMI can carry digital audio channels in addition to digital video but DVI can not. HDMI and DVI can carry HD video.

What affects the quality of streaming video?

Bit rate, color depth, and production factors (such as proper lighting) affect video quality. Resolution (if the video is HD or SD) does not. An SD video stream may look better than an HD video stream.

HD resolution video can be very low quality for a number of reasons, one being insufficient bit rate to transmit enough information to “fill” HD video frames with enough detail at an acceptable rate.

Here are some key factors that affect the quality of streaming video:

- **Quality of original source video.** Source video should be digital (preferably SDI) at the highest frame rate and resolution possible.

- **Input video resolution**. The aspect ratio of a video can positively or negatively influence the efficiency of an encoder. The video frame's height and width should both be wholly divisible by 16 if possible, or wholly divisible by 8 if not.

- **Video aspect ratio.** The input video aspect ratio should match the target output video aspect ratio. The most common aspect ratios are 16:9 and 4:3.

- **Content of the source video including the amount of motion and proper contrast.** Motion requires more information to represent accurately thus an encoder has to use more bandwidth to compress motion. Less motion (changes between or within video frames) requires less information and thus less bandwidth. Therefore, do not include unnecessary motion in source video. The background and foreground subjects in the video should be lit and contrasted properly. See *Streaming Production Best Practices* for more information.

- **Efficiency of the encoder**. Not all encoder implementations are equal!

- **Configuration of the encoder.** The number of I-frames, B-frames, Variable or Constant Bit Rate Encoding, CAVLAC or CABAC, etc.

- **Streaming bit rate.** Streaming video is like water rushing through a pipe. The bigger the pipe, the more water that can flow through it. An encoder *fills* a pipe with compressed video. Therefore, the larger the bit rate the more video information an encoder can compress. This additional information allows the video to be higher quality. The quality difference between 320 Kbps and 650 Kbps video is quite evident. However, a diminishing point of return exists so adding more bandwidth to an already high quality 320x240 video stream may not increase perceptible quality.

What is the quality of rVibe streaming video?

rVibe uses 16:9 aspect ratio based resolutions for video streaming from HD SDI progressive source video whenever possible:

- **Low resolution - 512x384**

- **High resolution - 768x432, 960x540, and 1280x720 (720p)**

An end user must have the ability to support the higher bit rates required with the higher resolution. rVibe uses between 500 Kbps and 1.5 Kbps.

Can rVibe support HD streaming? Yes, if the end user has enough bandwidth to view the encoded video stream at the desired resolution.

rVibe has the equipment and software to encode, stream and decode analog and digital HD video.

What are CBR and VBR?

Most encoders offer Constant Bit Rate (CBR) or Variable Bit Rate (VBR) encoding. VBR compresses video to a variable bit rate after analyzing video content to optimize quality and bandwidth use. CBR compresses video to a constant bit rate regardless of content. CBR helps ensure smooth playback without interruption since the rendering player doesn't have to deal with bit rate fluctuations. CBR by default prevents bandwidth spikes in compressed video while VBR can be configured to compress to no greater than a maximum bit rate which will also prevent bandwidth spikes.

If the source video has little or no movement or change over time, CBR can waste bandwidth encoding redundant or static information that is not needed to maintain quality. This is where VBR excels.

VBR uses bandwidth for more detailed or complex aspects of the video or in areas with high motion content instead of wasting it on static or less detailed imagery. However, VBR takes longer to compress video compared to CBR.

Despite CBR not using bandwidth as efficiently as VBR, CBR should be used for all live streaming applications to minimize encode latency. VBR can (and should) be used when creating VoD since this work typically doesn't have a time constraint.

When using VBR, set an average and maximum bit rate to avoid extreme bandwidth peaks in compressed video. Bandwidth spikes can cause extensive buffering and dropped frames for end viewers.

What is multi-pass encoding?

An encoder needs at least one pass through source video to analyze and compress it. The quality of compressed video (and how efficiently bandwidth is used) can be improved by allowing an encoder to make additional passes through source video to perform a more detailed analysis. This technique, known as *multi-pass encoding*, can be applied to both CBR and VBR encoding.

But (of course) there is a tradeoff with multi-pass encoding: *speed*. Additional passes through the source video increases overall encoding time (two-pass encoding typically doubles it) which may not be problematic for VoD, but can be problematic for live streaming where low latency may be required. Therefore, multi-pass encoding is not usually used to encode live events.

Two-pass encoding typically produces a noticeable increase in quality with most source video at the expense of twice the encoding time. Adding additional passes usually does not cause a discernible increase in quality or more efficient use of bandwidth. Once again, a factor of diminishing encoding return exists.

How powerful of a device is required to play streamed video?

A computing device must decode compressed video fast enough for fluid video playback. High bit rate compressed video requires significant processing power and memory to decode and play. Therefore, if viewers have limited processing power, adjustments must be made to the encoding settings to compensate. This is partially addressed by dynamic bit rate streaming (described later).

The power required varies with the complexity of the compressed video. A high motion 720p60 stream encoded at 1.5 Mbps requires a much more powerful rendering device compared to a 320x240 24 fps encoded stream at 350 Kbps.

Due to the amount of computational power required to decode and play streaming video, battery consumption in mobile devices also increases. Thus, care should be taken when determining target bit rates, frame rates and resolution for mobile devices.

Why is frame rate important and what are macroblocks?

The frame rate must reasonably convey sense of movement in the video source content. If the content contains fast-moving video, use a higher frame rate. As frame rate is increased, the frame size of the video may need to be reduced to retain reasonable visual quality for a specific bit rate. 24 fps is the cinematic frame rate. 29.97 (30) fps is the television and DVD frame rate. ESPN uses a frame rate of 60 to convey motion in sporting events.

Most video encoders including H.264, define each video frame as a series of pixel (e.g. 16x16) blocks. Each block is called a macroblock.

W	H	ASPECT RATIO	MACRO-BLOCK
1920	1080	16:9	8
1280	720	16:9	16
1024	576	16:9	16
960	540	16:9	4
896	504	16:9	8
768	432	16:9	16
640	360	16:9	8
512	384	4:3	16
512	288	16:9	16

W	H	ASPECT RATIO	MACRO-BLOCK
448	336	4:3	16
384	288	4:3	16
320	240	4:3	16

TABLE 3. ASPECT RATIO AND MACROBLOCK

Encoders compress and encode video by operating on differences between macroblocks in each frame (intraframe) and between previous and successive frames (interframe). This is known as spatial and temporal encoding respectfully.

By using frame resolutions that evenly distribute a macroblock (and not subportions), the video encoder can more easily encode the video source (which is more easily decoded during playback). Image 3 is an example of a video frame divided into groups of 16x16 pixel macroblocks.

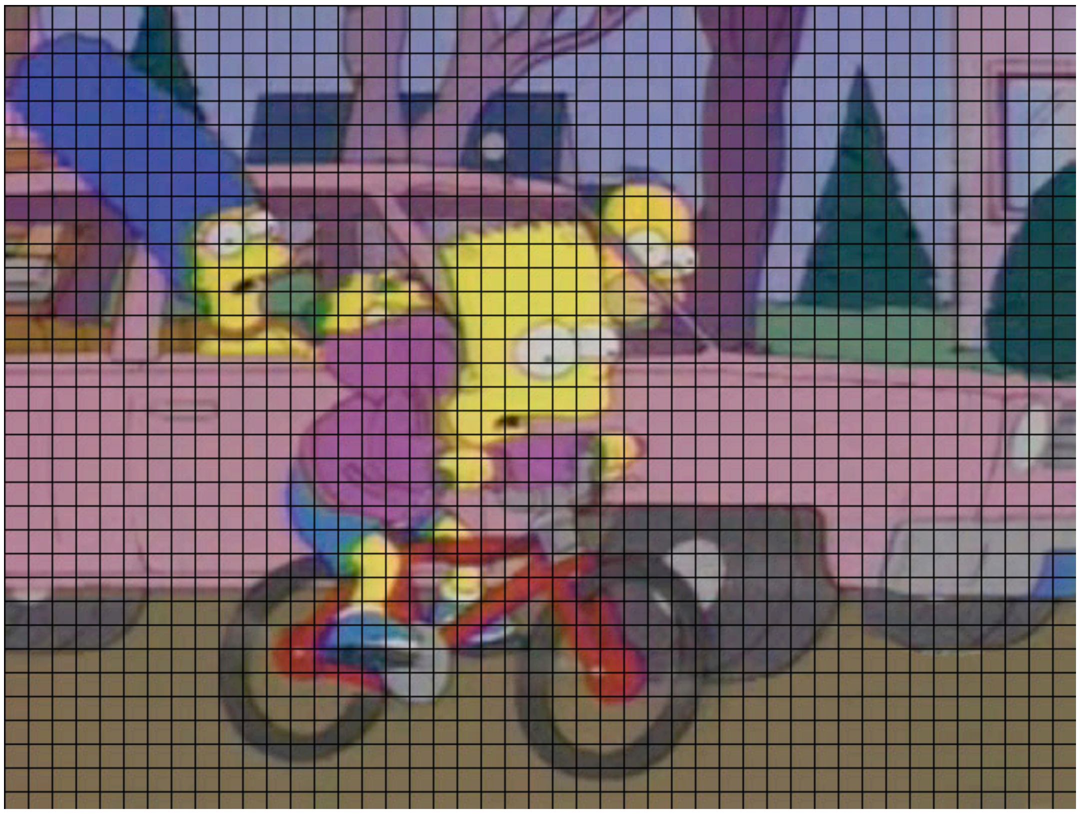


IMAGE 3. VIDEO FRAME DIVIDED INTO GROUPS OF 16X16 PIXELS OR MACROBLOCKS

Video encoders such as H.264 and VP 8 perform best when the frame width and height use multiples of 16. This allows macroblocks to be easily and efficiently created from the source pixels. For the best image quality and playback, always use width and height dimensions that are each wholly divisible by 16 (best), 8 (better), or 4 (good).

The width and height of video destined for computer playback should be kept to a square pixel aspect ratio. Any non-square pixel aspect video should be adjusted for computer playback. For example, source with a 720 by 480 frame size can be resized during the encoding process to 640 by 480, 480 by 360, 320 by 240, 160 by 120, and so on. Similarly, content with a 1440 by 1080 native non-square pixel frame size should be resized to 1920 by 1080, 1024 by 576, 768 by 432, 512 by 288, 256 by 144, and so on.

What is a keyframe?

An encoder generates a sequence of video frames of different types, with each type having a specific purposes. This is illustrated in the image below. For H.264, these frame types are:

- I-frames (intraframes, a.k.a. keyframes) are self-contained video frames that can be independently decoded without reference to other frames. They are the least compressible and contain the most information.
- P-frames (predictive frames) hold only the *differences* between the current frame and a previous frame. They are more compressible than I-frames but require I-frames to decode.
- B-frames (Bi-predictive frames) are a second level of prediction that hold *differences* between the current frame and both a preceding frame and subsequent frame. They are the most compressible and require I-frames to decode.

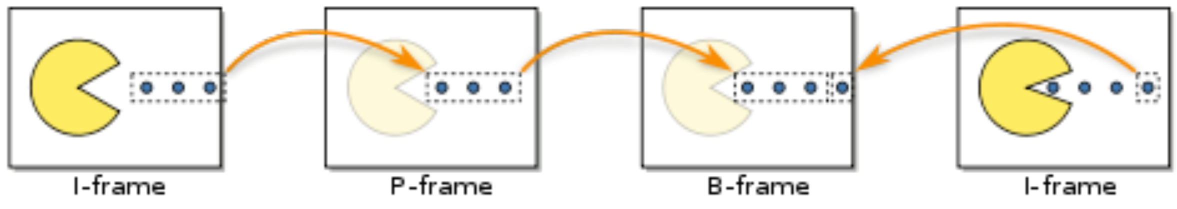


IMAGE 4. VIDEO FRAME SEQUENCING

A keyframe (I-frame) is used as a starter frame, drawing the initial video. Subsequent P-frames and B-frames store only the *changes* in the video frame relative to the previous keyframe.

A group of pictures (GOP) is the number of frames from one I-frame to another. In one GOP, there is a full I-frame and a number of frames predicting the motion associated with that full frame. Higher levels of compression (less bandwidth for a given clip of source video) are achieved through larger GOP sizes and deeper frame prediction (i.e., using an IBP compression sequence instead of an IP sequence). However, deeper frame compressions require more time to encode, resulting in higher latencies. Note that these parameters are typically established during system setup and not changeable during live encoding.

Performance tuning

Low bandwidth/high latency - If a tighter bandwidth budget exists, select a deep compression (IBBP) with a large GOP size (~40-200); latency will suffer to some extent.

High bandwidth/low latency - If 100% accuracy and immediacy is required, select low compression (I-frame only, or IP) and a very small GOP size (~10-40).

Keyframe interval

The frequency of keyframes (I-frames) in the video can be specified in the encoder. This is known as the keyframe interval.

Keyframes are best inserted in multiples (or fractions) of the compressed file's frame rate.

Keyframes are best inserted in multiples (or fractions) of the compressed file's frame rate. For example, if a frame rate of 15 fps is used, use a keyframe interval of 150 frames (one keyframe will be generated every 150 frames). Some encoders can be provisioned to select automatic or natural keyframes, which essentially tell the encoder to make a keyframe whenever enough of the video frame has changed.

The keyframe interval can greatly affect the quality of the overall video clip. If the encoder creates too many keyframes for an allocated video bit rate, the visual quality degrades. If too few keyframes are created, one loses the ability to start the playback of the video or seek or scrub the video very smoothly or accurately.

For higher quality streaming, having too few keyframes is better than having too many.

One important consideration for keyframe intervals used with Flash-based video is the ability to seek to more points of a VoD file. A viewer can only seek to keyframes within the VoD—if the viewer tries to seek to a time in the clip that doesn't contain a keyframe, the Flash player jumps to the closest keyframe. This may be noticed when scrubbing a video clip; the frequency of updates while scrubbing indicates the number of keyframes. However, if enhanced seek is used with video content served by Flash Media Server, the server can generate keyframes on the fly—enabling a viewer to scrub the video more smoothly.

Keyframes and Dynamic Bit Rate Streaming

The advent of Dynamic Bit Rate Streaming in Flash Media Server 3.5 (and Flash player 10), Microsoft's Internet Information Service (IIS) Smooth Streaming (and Silverlight player) and Apple's HTTP Streaming, allows the quality of an end user's viewing experience to be adjusted continuously based on the end user's bandwidth conditions.

An end user's video player constantly evaluates available end user bandwidth and instructs the streaming server to stream at a bit rate that can be supported by the end user's available bandwidth. A player autonomously switches between individually encoded higher and lower bit rate video streams based on the available bandwidth. There can be many available streams from which the player can choose—the number is dependent on the encoding system, media server and bandwidth availability at the source and destination.

In order to facilitate smooth switching between streams, two provisions must be made when encoding.

1. Keyframes must be encoded at a much higher ratio to P-frames compared with previous encoding recommendations (e.g. one every 150 frames). A 25% (one out of every four) ratio to P-frames is recommended. Some even advise a 50% (every other frame is a keyframe) ratio.

2. All streams must be encoded using the ***same*** keyframe ratio.

Implementing these provisions will ensure stream switching is done quickly and smoothly with no noticeable delay and/or staggering.

H.264 B-frames in more detail

As mentioned H.264 content can utilize a special video frame type called a B-frame, or a Bi-predictive frame. B-frames can refer to previous or future frames. How can a frame borrow details from a frame that has not yet played? The encoder can arrange frames out of sequence as they're written to a compressed video file. On playback, the decoder knows to display the sequence in the correct order—a process known as frame reordering. For example, if one had a sequence of frames such as:

0 1 2 3 4 5 6

and the encoder wanted to make frame 3 a B-frame that referred to data in frames 2 and 4, the frames would be stored in the following sequence:

0 1 2 4 3 5 6

Upon playback, the decoder reorders the frames to the original sequence. B-frames, though, are more decode processor-intensive as a result of the frame reordering operation. The B-frame count refers to the number of B-frames per keyframe interval. B-frames are sometimes not even used for live streaming where real-time delivery is important because they increase latency.

Use a B-frame count of three or lower.

What are encoding profiles and levels?

The H.264 standard defines seven profiles to serve different applications, categorized by quality verse computational cost. The most common are Baseline, Main and High.

Baseline requires the lowest amount of processing power but provides the least efficient compression. It is typically used for video conferencing, security cameras and mobile devices such as iPhones.

Main provides a medium level of quality for a slightly higher computational cost.

The High profile provides the best quality because the encoder can use the widest range of encoding techniques provided by the H.264 standard but requires the highest processing power to decode. Some devices don't even support it (e.g. iPhone). It was originally intended for Blu-ray devices but is gaining in popularity as computing devices become ever more powerful.

H.264 has 11 levels or degree of capability to limit performance, bandwidth and memory requirements. Each level defines the bit rate and encoding rate in macroblock per second for resolutions from SD to HD. The higher the resolution, the higher the level required.

What is CAVLC and CABAC in H.264?

The Main and High profiles of the H.264 codec support encoding parameters such as B-frames, reference frames, Context-Adaptive Variable-Length Coding (CAVLC) - the default - and Context-Adaptive Binary Arithmetic Coding (CABAC) entropy encoding. These parameters can greatly improve the visual quality of a given video clip at any bit rate.

CABAC is more processor intensive than CAVLC. However, one typically sees a marked improvement in the visual quality of H.264 content with CABAC.

If the targe audience is using dual-core (or better) processing, use CABAC for higher quality video. If lower latency is required, use CAVLC.

CABAC creates more latency in live video streams as opposed to CAVLC so use CAVLC when real-time delivery is a concern.

Minimizing the latency in live streaming video

Live streaming sometimes requires low latency. This is particularly important for live streaming applications that require interaction between subjects in the encoded video (such as a professor) and those watching the live stream remotely (such as students) and those that require interaction between the remote users themselves (e.g. video conferencing).

Before delving into ways to mitigate latency in live streaming video (specifically encoded with H.264), let's first examine where latency occurs *significantly* in the video streaming process.

STEP	LATENCY
Capture	Insignificant to non-existent
Conversion	Insignificant; low ms or even μs to convert analog video to a digital format
Preprocess	Insignificant; in the low ms or even μs to remove noise and other artifacts from the digital video

STEP	LATENCY
Encode	Marginally significant; typically a few ms to 20 or more dependent on the disposition of video content, desired quality of encoded video (as affected by encoder settings), processing power available to perform the encode and whether or not encoding is done in hardware or software
Transmission	Highly significant; typically anywhere from a few hundred ms to seconds; dependent on fluctuating bandwidth conditions at the encoding and decoding locations and during transmission over the network, the physical distance the stream is required to traverse, quality and configuration of network equipment (routers, gateways) relative to the prioritize of streaming traffic to other types of traffic and finally, security settings in the network and decode location
Decode	Marginally significant; typically a few ms to 20 or more depending on the complexity of the encoded video, processing power available to perform the decode and whether or not decoding is done in hardware or software
Presentation	Marginally significant; typically a few ms to 20 or more depending on the processing power available to render the video

TABLE 4. LATENCY DURING VIDEO STREAMING PROCESS

Therefore, latency is *significantly* comprised of:

total latency = encode time + transmission time + decode time + presentation time

Now let's explores ways to mitigate total latency by minimizing where it occurs in the live video streaming process.

Minimizing H.264 encode latency

Some techniques to minimize latency introduced by H.264 encoding:

- Use CAVLC not CABAC
- Use a low GOP such as 3
- Don't use B-frames or keep the B-frames very low
- Use CBR, not VBR
- Use the Constrained Baseline Profile
- Reduce the frame rate (to 12 or 15); take care not to degrade quality with a low frame rate

- Ensure all CPUs and all cores on each CPU are used by the H.264 encoder, use the most powerful hardware available to run a software encoder and allocate as much processing power to the encoding software as possible by minimizing the use of other software applications
- Use a hardware H.264 encoder (this can be costly)
- Reduce motion in the video source, subjects should wear clothes without brightness extremes and fine patterns; ensure the scene and subjects are well lit; see [7] for more information
- Use multi-bit rate streaming

Minimizing transmission latency

Some techniques to minimize latency introduced by transmission:

- Ensure there is more than enough sustained bandwidth at the encode site
- Use a CDN or a peer-to-peer UDP based stream distribution technology
- Use a point to point dedicated network link between the encode and decode sites; this can be cost prohibitive and impractical if there are many viewers worldwide at unknown viewing locations
- Prioritize streaming traffic over other types of network traffic on a private LAN or VPN
- Ensure streaming ports are open on network infrastructure devices on a private LAN or VPN
- Separate streaming traffic onto a dedicated VLAN in a private LAN
- Do not stream over a wireless network since data transmission collisions can be more frequent, and more adverse, than over a wired network

Minimizing H.264 decode latency

Some techniques to minimize latency introduced by decoding:

- Ensure all CPUs and all cores on each CPU are used by the H.264 decoder, use the most powerful hardware available to run a software decoder and allocate as much processing power to the decoding software as possible by minimizing the use of other software applications

Minimizing presentation latency

Some techniques to minimize latency introduced by presentation:

- Reduce buffering in the player but not at the expense of causing the video frames to be dropped or skipped
- Ensure the player has the capability to decode multiple bit-rate video streams

Keep in mind that often times optimizing a system for low latency will adversely affect the quality of the final rendered video so a balance must be struck.

References

[1] Optimal Frame Dimensions for Flash Video, Reinhardt, R.

[2] Flash Video Bitrate Calculator, Reinhardt, R.

[3] Video with Adobe Flash CS4 Professional: Compression and Encoding Primer, Reinhardt, R.

[4] Video with Adobe Flash CS4 Professional: Delivery and Deployment Primer, Reinhardt, R.

[5] The H.264 Advanced Compression Standard, 2nd. Edition, I. Richardson, Wiley.

[6] The H.264 / AVC Standard: Overview and Introduction to Fidelity Range Extensions, G. Sullivan, P. Topiwala, A. Luthra, SPIE Conference on Applications of Digital Image Processing XXVII.

[7] HTTP Streaming: What You Need to Know, Siglin, T.

[8] White Paper, H.264 Video Compression, Haivision, August 2010.

[9] Streaming Production Best Practices, Benes C., rVibe.

[10] H.264 For The Rest Of Us, K. Amerasinghe, Adobe.

[11] H.264 video compression standard, Axis Communications.