A/V Crash Course: Basics of Video Processing

March 28, 2017

Andy Gallerstein

Overview

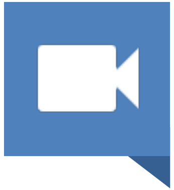
This presentation will introduce select video processing fundamentals

- Example Video Life-Cycle
- Understanding Human Vision
- Common Codecs & Containers
- Frame-Based Video Encoding
- Burst Streaming

Example Video Life-Cycle

Mile-High View of Getting Video to Clients

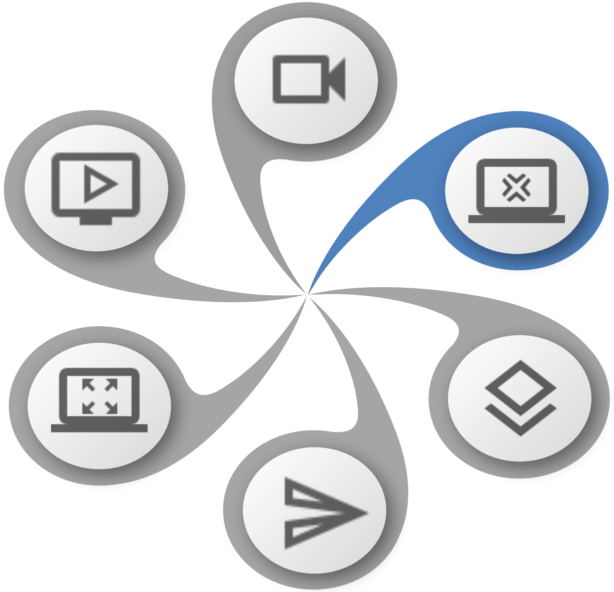
Example Video Life-Cycle: Step 1 of 6



Video Camera Records Footage

- Input:
  - Light enters the Camera Lens
- Light is converted into raw images
  - Photographic Film - chemistry
  - CMOS Sensor - photons to electrons
  - CCD Sensor - photons to electrons (fancier)
  - DVS Sensor - only senses changes
  - Etc.
- Output:
  - Raw video (not encoded)

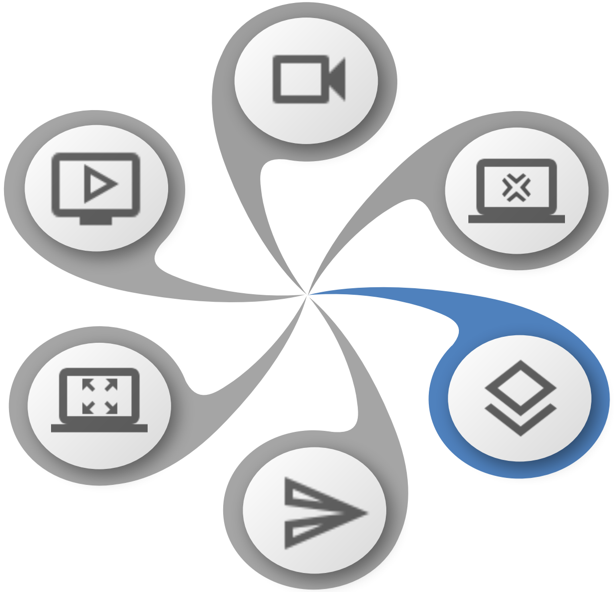
Example Video Life-Cycle: Step 2 of 6



Encoder Compresses Video

- Input:
  - Raw video (not encoded)
  - Encoding Configuration/Parameters
  - Encoding Hardware (and/or software)
- Video is formatted and compressed:
  - Specific Encoding ("codec")
  - Frame Rates (FPS = frames per second)
  - Bit-Rates (CBR vs VBR → const size vs quality)
  - Resolutions
  - Etc...
- Output:
  - Encoded video

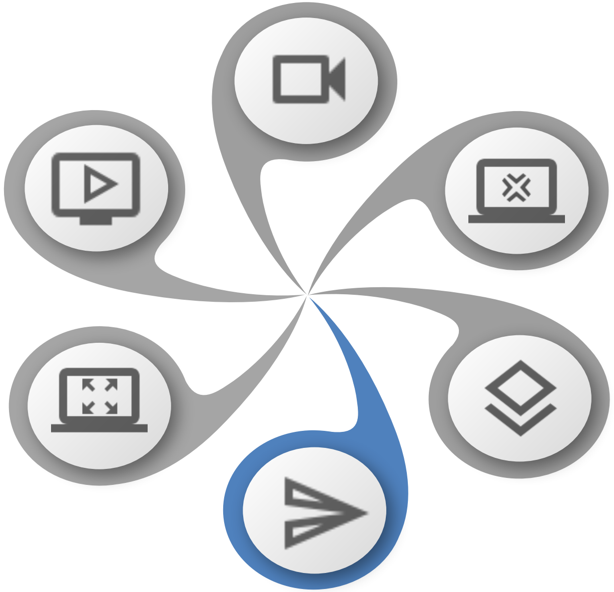
Example Video Life Cycle: Step 3 of 6



Elementary Media Streams “Muxed”

- Input:
  - Encoded Video stream(s)
  - Encoded Audio stream(s)
  - Subtitles, etc.
- All media streams combined & packaged
  - Various streams “muxed” together
  - Packaged into a “container” format
  - Meta-data computed, such as:
    - SDP to define media for RTSP transfer
    - Headers in an MP4 file
- Output:
  - A transportable media file/stream

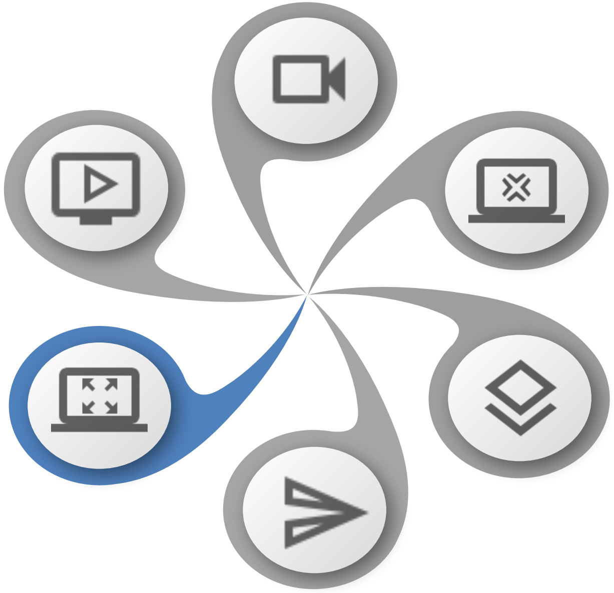
Example Video Life-Cycle: Step 4 of 6



Media Transport

- Input:
  - A transportable media file/stream
  - Connection between Sender & Receiver
- Video transported/transferred:
  - Sender may need to packetize for transfer
  - Receiver must put the pieces back together
  - Networking introduces complexities:
    - TCP vs UDP (speed, ordering, retries)
    - Buffer too little → what if network slows?
    - Buffer too much → big playback latency
- Output:
  - A transportable media file/stream
  - (or at least part of it)

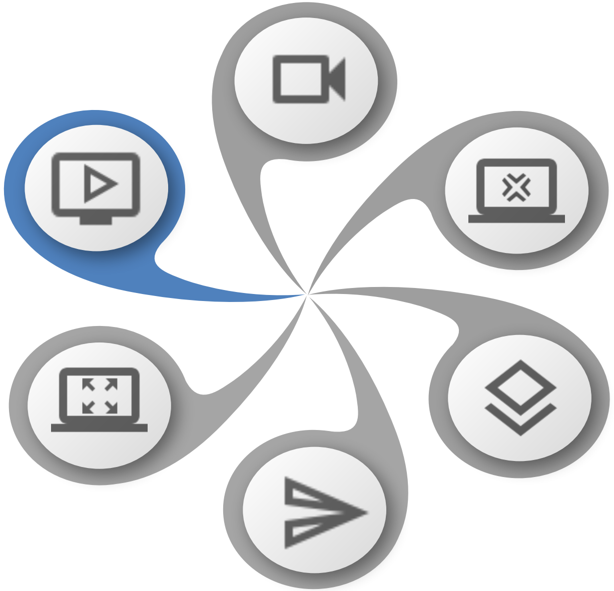
Example Video Life-Cycle: Step 5 of 6



Media Demuxed & Decoded

- Input:
  - Encoded audio/video streams
  - Typically wrapped in some container format
  - Potentially pre-buffered for network transfer
- Media is unwrapped & uncompressed
  - Packetized media buffered & put back together
  - Media streams “demuxed” from container
  - Media streams decoded into raw playable format
- Output:
  - Playable, decoded raw media stream(s)

Example Video Life-Cycle: Step 6 of 6



Media Played

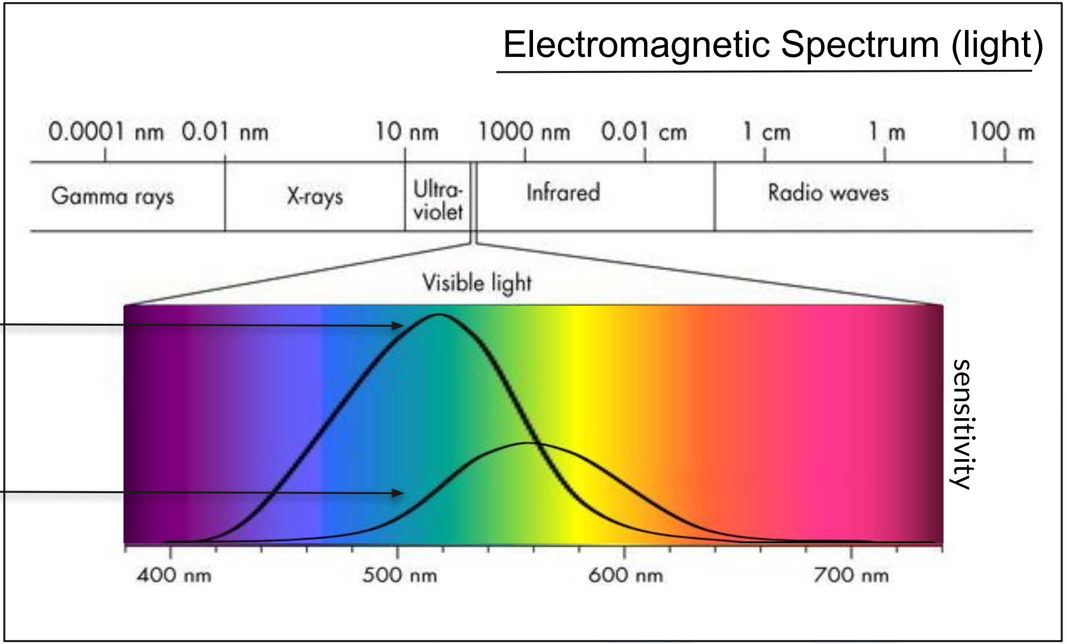
- Input:
  - Raw video/audio streams ("elementary streams")
  - Throttled constant feed of stream data
- Streams rendered to screen (or speakers...)
  - Software meets hardware
  - Synchronization logic might live here?
  - Feeding hardware drivers for constant playback
- Output:
  - Video on the screen
  - Audio out of the speakers

Understanding Human Vision

What we see & how we represent it

Understanding What We See

- Objective vs Perceived Light
  - We only see “visible light”
  - We don’t see all visible light equally
- Sensitivity
  - Varies per person/wavelength/intensity
  - Female eye most acute on avg
- Night Vision
  - Scotopic - dim light
  - Faster but less detail
  - Biology: Rods - B&W (brightness)
- Color Vision
  - Photopic - well lit
  - Yellow-Blue vs Green-Red
  - Biology: Cones - Color (3+ types)



Measuring What We See

- Luminosity (L) - power of a light source (ex: the sun emits 3.85×10^{26} Joules/second)
  - Radiant Flux/Power (Φ_e) - measure of luminosity (Φ_e = Watts = J/s)
  - Spectral Flux/Power ($\Phi_{e,v}$) - measure of luminosity for subset of wavelengths

- Luminance (L_v) - intensity of light on given surface (ex: sun's luminance in NYC in Summer > Winter)
  - **Objective Luminance** - how much light is actually emitted via some radial area
  - **Relative Luminance** - brightness, filtering luminance by what we humans perceive
    - Spectral Sensitivity - relative efficacy of detection per light wavelength (ex: what human eye detects)
    - Luminous Flux (Φ_v) - measure of *perceived* power of light (Φ_v : Lumens (lm) = candela/radial^2)
    - Luminosity Function ($y(\lambda)$) - perceived brightness by human eye (subjective estimate)
      - Photopic Luminosity Fxn - for everyday light levels
      - Scotopic Luminosity Fxn - for low light levels (applies curve to photopic)

Modeling What We See

- Chroma SubSampling
  - Reduce bits used for representing color (chroma) while maintaining luminance (luma more important)
  - Various methods/algorithms (eg: 4:4:4, 4:2:2, 4:2:1, 4:1:1, 4:2:0, 4:1:0, 3:1:1)
  - Ex: 4:2:2 - cut horizontal color resolution in half (barely noticeable), compressing to 2/3rd original bytes
- Gamma Correction (γ)
  - Compress bits needed to represent luminance by leveraging how humans perceive light
  - Luma - weighted sum of gamma-compressed R'G'B' (symbol ' denotes gamma compressed)
    - `Y' = 0.213R' + 0.715G' + 0.072B' [BT. 709]`
    - `Y' = 0.212R' + 0.701G' + 0.087B' [SMPTE 240M]`
    - `Y' = 0.299R' + 0.587G' + 0.114B' [CCIR 601,ITU-R BT.601-7]`
- Color Spaces (common models)
  - RGB[A] - Red, Green, Blue, and sometimes Alpha. (traditionally 8-bit granularity, 0-255 * 3)
  - R'G'B' - RGB with Gamma compression
  - Y'CbCr - Luma (brightness), Chroma blue, Chroma Red
    - Y = krR + kgG + kbB (see weighting coefficients above in Gamma Correction)
    - Conversion to RGB: Cr = R – Y, Cg = G – Y, Cb = B – Y
  - YUV - analog version of Y'CbCr
  - HSL - Hue, Saturation, Lightness (Radial model, H=RGB, S=strength of hue, L=brightness)
  - CMYK - Cyan, Magenta, Yellow, and Black (e.g. for printing on paper)

Common Codecs & Containers

Standards for video compression & storage

Common Video Codecs

Video Codec - a format for [de]compressing the size of a video stream (codec = encoder/decoder)

Codec	Description
H.264	Most popular/supported. License fee for large-scale use.
H.265	Improvements on H.264. Less popular yet. Licensing.
AV1	In development by Alliance for Open Media. License Free.
VP8 (and 9)	By Google, predecessors to AV1. License free.
Theora	Part of VP3 (early version of above).
MJPEG	Bunch of JPEG images

([redacted])

Example Audio Codec (old school telephony)

- Pulse Code Modulation (PCM) - conversion between Analog & Digital signals (most notably audio)
    - Used since the mid-1800's for telecomm
    - G.711 Codec/Standard for Audio first introduced in the 1970's
        - Limits the dynamic range (frequency) to optimize for encoding/compressing human speech
        - μ-Law: Telephony in the United States & Japan (8Kb = 1 second audio)
        - a-Law: Telephony everywhere else (8Kb = 1 second audio)
        - Other versions have been released since the 70's (e.g. G.711.1, G.726, ...)
    - Sampling-Based Streaming
        - Uses "samples" instead of frames, where each sample is played for a fixed number of microseconds
        - Sample Size - the number of bits used per sample (e.g. G.711 uses 8-bit samples)
        - Sample Rate - number of samples per second (e.g. G.711's uses 8KHz sample rate)
        - Note: some audio codecs use frame-based compression

Common Containers

Transport Container - a format for wrapping/transporting 1+ related elementary media streams

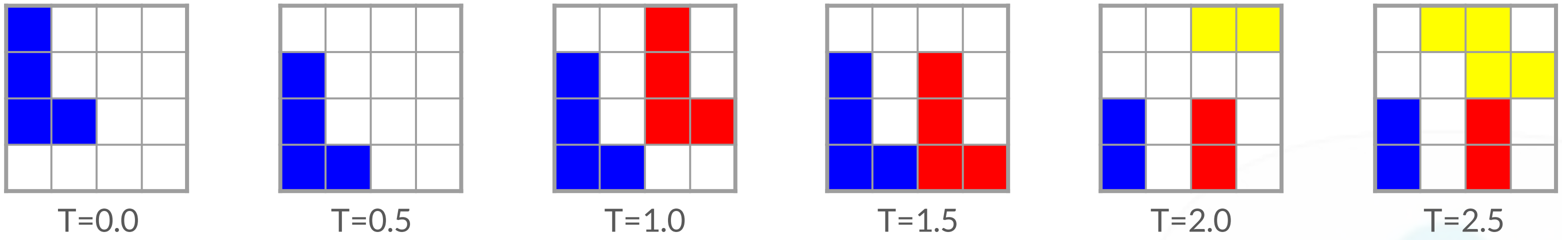
Category	Container	Standard For
File	MP4	MPEG (mpeg-4: H264/AAC/etc)
	Matroska (.mkv)	Open source
	AVI	Microsoft (still around?)
	QuickTime (.mov)	Apple (and their QuickTime framework)
Network	MPEG TS	MPEG (packetizing audio/video/subtitles/program info/etc)
	RTP	IETF (wrapping + packetizing, specs for many codecs)

Frame-Based Video Encoding

Basics of video compression

Frame-Based Video Encoding (pg 1 of 3)

Say we have take 6 consecutive pictures of a Tetris game:



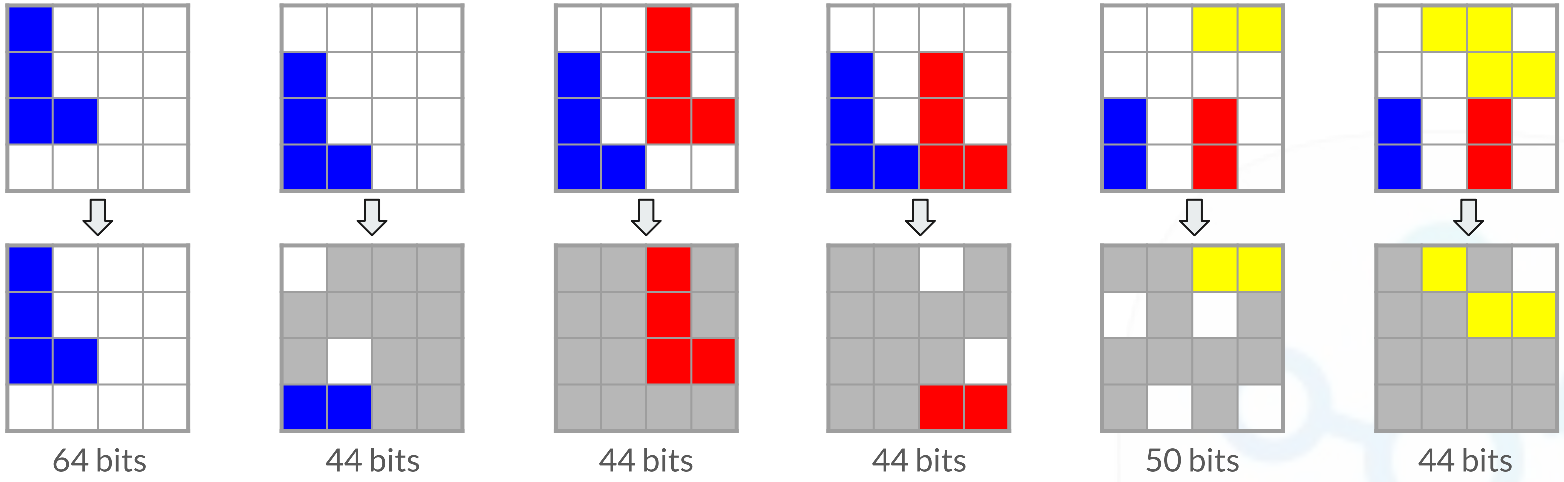
And say we serialize each frame as follows:

- Initial 16 bits store the frames timestamp
- Frame divided into 4 x 4 grid of squares
- 3 bits are reserved for each square
    - 1st bit = Red
    - 2nd bit = Green
    - 3rd bit = Blue

384 Bits Total (6 frames @ 64 bits/frame)

Frame-Based Video Encoding (pg 2 of 3)

Represent frames 2-6 as a delta from previous frame instead of the entire picture



1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	← 16 Bits for timestamp
1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	← 4x4 Grid: what pixels changed
1	1	1	1	1	1	0	0	1	0	0	1					← RGB for each pixel that changed

Total: 290 bits
(25% compression)

Frame-Based Video Encoding (pg 3 of 3)

Useful Vocabulary:

- Lossless Compression - *all the information is still available (like png)*
- Lossful Compression - *some of the information lost to improve compression. Very common. (like jpeg)*
- Resolution - *How many pixels are used to make a picture*
- Bit Rate - *How many bits per second (may be constant, may be variable to achieve ~constant quality)*
- Frame Rate - *How many frames (i.e. pictures) displayed per second*
- Group Of Pictures (GOP) - *Set of consecutive encoded pictures. Can be decoded w/out another set*
    - Intra Coded Pictures - *Contain the entire picture ("I-Frame" or "Key frame" or "IDR")*
    - Predictive Pictures - *Contain changes from previous frame(s) ("P-Frame" or "Slice")*
    - Bi-Directional Pictures - *Contain changes from previous OR future frame(s) ("B-Frame" or "Slice")*
- H.264 Video - *Common video codec utilizing motion-based-compensation & macroblocks*
    - *Specified in Part 10 of the MPEG-4 AVC (conglomerate of algorithms, patents, companies, etc)*
    - *Requires licensing for encoding & decoding. Most mobile phone manufacturers license & provide H.264 HW codec.*
    - *Defines a set of Profiles which dictate what features can be used for the compression (e.g. Baseline Profile = simple)*

Burst Streaming

Performance Technique For Live Streaming

Burst Streaming (pg 1 of 3)

Streaming servers will often employ a technique called a *Burst Streaming* when initiating a player's stream

In order to discuss the purpose and mechanisms of the Stream Burst, we must lay some groundwork:

- We use the following metrics to quantify the effects of this technique:
  - *Time To First Frame* - how long is the delay from clicking [play] to when playback begins
  - *Playback Latency/Lag* - the delay from when an event is recorded to when it is played on screen
- Video streams are generally composed of sets of sequential frames (called a Group of Pictures):
  - A Group of Pictures will start with a Key Frame (a.k.a. IDR) - a stand-alone picture (large amount of data)
  - All Subsequent frames are called slices - they reference an IDR and basically say which pixels changed
- Streams will be transmitted to the player over a network, incurring:
  - Some minimum latency while data travels over the network
  - Some fluctuation in latency/bandwidth which will vary over time
- A player must pre-buffer stream data before initiating playback to mitigate network delays:
  - A small buffer may not be effective at mitigating/covering network issues, resulting in dropped frames/etc
  - Increasing the buffer size will:
    - Decreases risk of network issues affecting playback
    - Increases *playback latency/lag* (delaying playback as buffer fills)
    - Increases Time To First Frame (However this may be mitigated by the Stream Burst)

Burst Streaming - Example (pg 2 of 3)

Below is an example of stream data arriving on the server:

Video																						
RTP Packet #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Derived Frame #	1			2		3	4	5	6		7	8	9	10	11				12		13	...
Derived PTS (start of frame)	0.0			0.1		0.2	0.3	0.4	0.5		0.6	0.7	0.8	0.9	1.0				1.1		1.2	...

Audio												
Derive PTS (first sample)	0.0	.12	.24	.36	.48	.60	.72	.84	.96	1.08	1.20	...
RTP Packet #	1	2	3	4	5	6	7	8	9	10	11	12

Server Time →

Now let's assume a client connects to play the stream

- The player requires a Key Frame to start rendering video, which leaves our server with 3 options:
    a. Simply send stream data as it arrives
        - Video start @ RTP packet #10 (half of frame #6) -- ½ second of green screen (unrenderable video)
        - Audio start @ RTP packet #6 -- playable depending on the player (may be slightly out of sync)
    b. Wait about ½ second for the next Key Frame to arrive before sending stream data
        - Video start @ RTP packet #15 -- ½ second of black screen (waiting for video)
        - Audio start @ RTP packet #9 -- playable (may be slightly out of sync, e.g. 0.4 seconds)
    c. Send a "burst" of cached stream data (since start of recent key frame) then send stream data as it arrives
        - Video start @ RTP packet #1 -- playable, but player must recognize & handle the burst
        - Audio start @ RTP packet #1 -- playable

Burst Streaming (pg 3 of 3)

Consider the trade-offs between the 3 options listed on the previous slide:

Starting Stream	Playback Latency	TTFF	Other Artifacts
1. Send as data arrives	not affected	delayed	Insufficient Buffer
2. Wait for next Key Frame	not affected	delayed	Insufficient Buffer
3. Burst Streaming	may increase	significant decrease	none

We can conclude that Burst Streaming:

- Significantly decreases the Time To First Frame
- Introduces risk of increases Playback Latency
    - Note: this can be completely mitigated by ensuring burst size is <= player's buffer size
- Does not introduce risk of Frame Drops due to insufficient buffer
    - Note: Player implementation may ensure sufficient buffer size, but will further delay TTFF

Backup & Extras

...

H264 Formatting

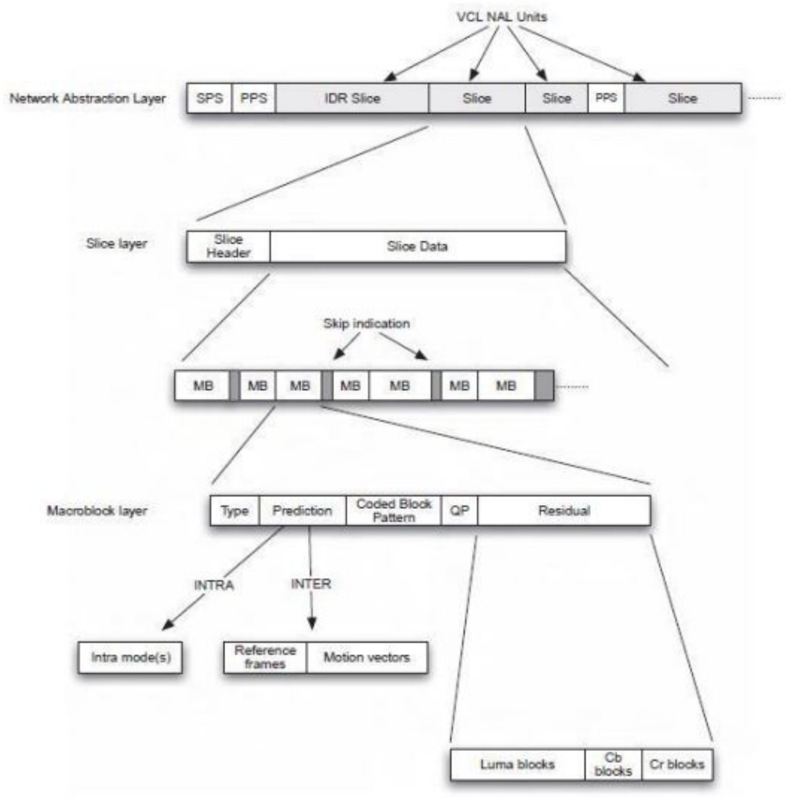
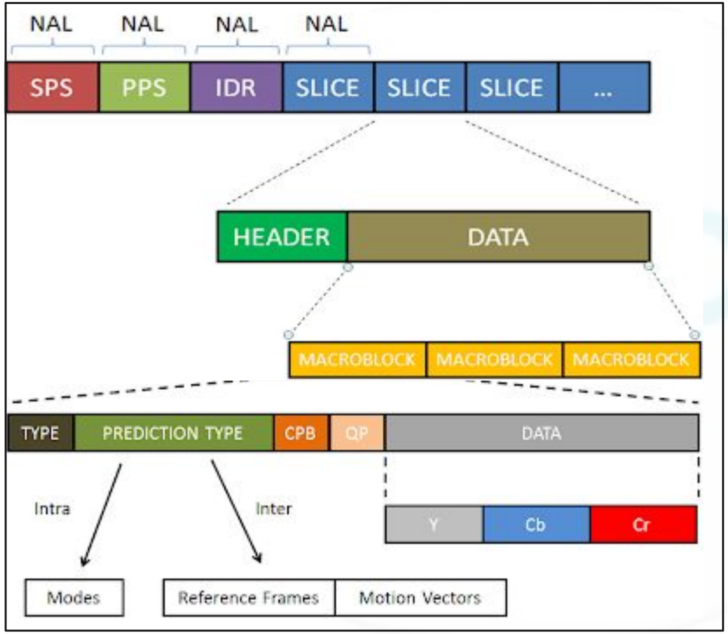


Fig. 2 the H264 syntax [1]

http://gentlelogic.blogspot.com/2011/11/exploring-h264-part-2-h264-bitstream.html



H264 NAL (1/2)

- Coded Video Sequence - Series of sequential access units in the NAL, all point to same PPS
- (NALU) Network Abstraction Layer Unit - a packet of bytes with some organization header data
  - Byte-Stream Format - Stream is ordered with each NALU has 3-Byte *Start Code Prefix*
  - Also used for organization when transporting network packets (e.g. RTP and fragmented NALU)
  - (VCL) Video Coding Layer NALUs - contain the data that represents the values of the samples in the video pictures
  - Non-VCL NALUs - contain any associated additional information such as parameter sets
- Access Unit - A set of NAL units in a specified form the decode into 1 decoded picture
  - *Access Unit Delimiter* - optional prefix to help id start of each access unit
  - *Supplemental Enhancement Information (SEI)* - optional prefix containing data such as picture timing info
  - *Primary Coded Picture* - a set of VCL NAL units that together compose a single picture
  - *End Of Sequence* - may be present to id end of sequence if its last picture in the sequence
  - *End of Stream* - may be present if its the last picture in the entire stream
- (PS) Parameter Set - important header data that can apply to decoding a large number of VCL NAL units
  - (SPS) Sequence Parameter Set - apply to a series of consecutive coded video pictures
  - (PPS) Picture Parameter Set - apply to the decoding of one or more individual pictures
  - Each NALU points to a PPS, the PPS points to the SPS
-

H264 NAL (2/2)

NAL TYPES		
Type	**Definition**	
0	Undefined	
1	Slice layer wout partitioning non IDR	
2	Slice data partition A layer	
3	Slice data partition B layer	
4	Slice data partition C layer	
5	Slice layer without partitioning IDR	
6	Additional information (SEI)	AU prefix w/ picture timing info,etc
7	Sequence parameter set (SPS)	Header data for series of consecutive pics
8	Picture parameter set (PPS)	Header data for decoding 1+ individual pics
9	Access unit delimiter	AU prefix to help id start of each access unit
10	End of sequence	AU postfix if last picture in the sequence
11	End of stream	AU postfix if last picture in the stream
12	Filler data	
13..23	Reserved	
24..31	Undefined	Ex: 28 to break NAL across RTP network pkts

SLICE TYPES	
Type	**Description**
0	P-slice. Consists of P-macroblocks (each macro block is predicted using one ref frame) and / or I-macroblocks.
1	B-slice. Consists of B-macroblocks (each macroblock is predicted using one or two ref frames) and/or I-macroblocks
2	I-slice. Contains only I-macroblocks. Each macroblock is predicted from previously coded blocks of the same slice.
3	SP-slice. Consists of P and / or I-macroblocks and lets you switch between encoded streams.
4	SI-slice. It consists of a special type of SI-macroblocks and lets you switch between encoded streams.
5	P-slice.
6	B-slice.
7	I-slice.
8	SP-slice.
9	SI-slice.

H264 MacroBlock

- typically consists of 16×16 samples, and is further subdivided into transform blocks, and may be further subdivided into prediction blocks
- Transform Blocks
  - input to the linear block transform
  - Originally were fixed size of 8x8 samples, H.264 Profiles may use different sizes (some even dynamic)
- Prediction Blocks
  - In early codecs, there was one motion vector per macroblock
  - In H.264, a macroblock can be split into multiple variable-sized prediction blocks, called partitions

Bitstream representation [edit]

A possible bitstream representation of a macroblock in a video codec which uses motion compensatio

```
+------+------+-------+--------+-----+----+----+--------+
| ADDR | TYPE | QUANT | VECTOR | CBP | b0 | b1 | ... b5 |
+------+------+-------+--------+-----+----+----+--------+
```

- ADDR — address of block in image
- TYPE — identifies type of macroblock (intra frame, inter frame, bi-directional inter frame)
- QUANT — quantization value to vary quantization
- VECTOR - motion vector
- CBP — Coded Block Pattern, this is bit mask indicating for which blocks coefficients are present.
- bN — the blocks (4 Y, 1 Cr, 1 Cb)