

AT&T Video Optimizer

- Tour
- Docs
- Best Practices
- Blog

Streaming Separate Audio and Video

Introduction

Audio normally accompanies video when streaming, but it is often an afterthought.

And there are different ways of matching them together.

Originally in internet media streaming, most audio and video services were multiplexed together, or **muxed**.

There has been a steady transition to **demuxed** streaming, where the audio and video segments (A.K.A. chunks) are sent independently, then matched together by the player.

Demuxed streaming **can save money** by requiring less server storage, and it can improve CDN usage. It also offers **more audio options**.

While there are advantages to demuxing audio, you need to plan and monitor. And there are related best practices you should follow.

In the Background section below we offer some context on demuxed streaming. In the Issue section we identify some potential problems, and their underlying causes. In the Best Practices section we offer design principles and recommendations that can help you mitigate these issues, leading to better Quality of Experience (QoE).

Background

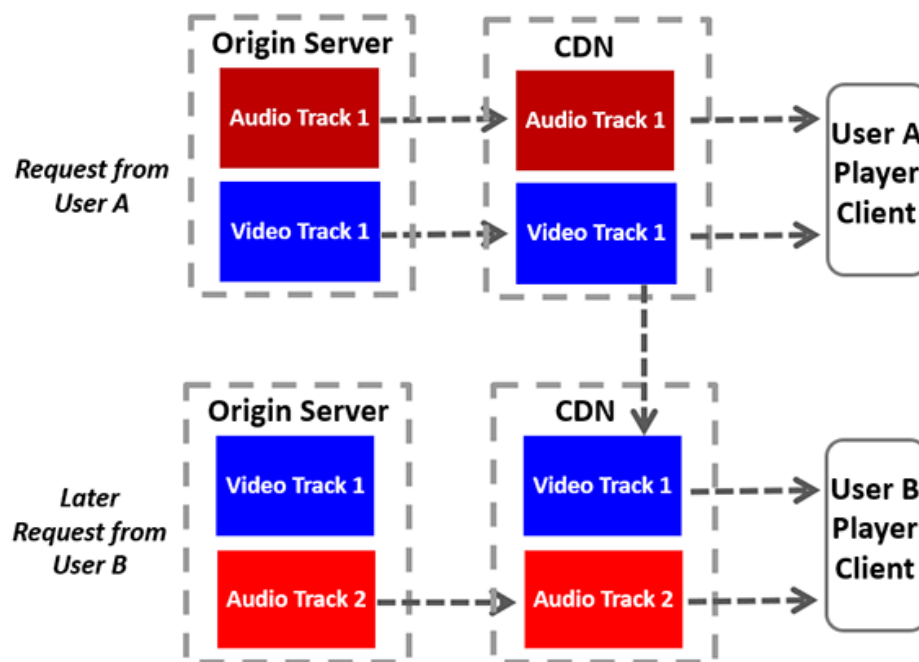
When the audio and video tracks are stored separately in demuxed mode, it offers a number of advantages.

1. One advantage of demuxing would be in situations where a video requires **multiple language tracks**.
2. Another advantage would be any scenario where **multiple audio quality levels** are desired.

3. A third advantage of demuxed audio is that it **requires less storage on the origin server**. You only need to store the appropriate audio once for all the video segments. When the audio is muxed, each video needs to be replicated to include all the audio variants.
4. **Demuxed mode can also increase CDN caching efficiency**. For example, suppose **user A** requests video track #1 and audio track #1. Then **user B** requests video track #1 and audio track #2 at a later time.

In muxed mode, user B will need to download a variant containing both video track #1 and audio track #2 from the origin server.

But in demuxed mode, user B can get video track #1 from the CDN cache (which was downloaded previously to the CDN due to user A's request), then User B only needs to retrieve audio track #2 from the origin server. The image below illustrates the video caching.



Due to the above advantages, more services are moving towards demuxed audio and video tracks.

The Player

The logic in the player needs to dynamically determine which audio and which video segments to select.

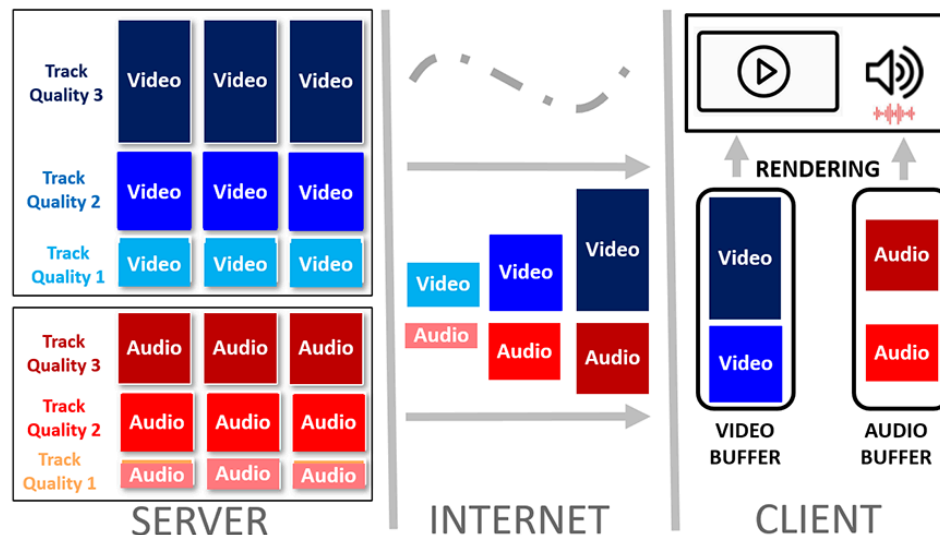
Adaptive bitrate (ABR) streaming allows adaptation to dynamic network conditions by providing multiple tracks/variants that all represent the same content, but which are encoded at different bitrates and quality levels. Each track is divided into multiple segments (chunks), each containing a few seconds worth of content.

During playback, the client player dynamically selects a segment from the multiple available variants based on existing network conditions.

On the server, a video track and its corresponding audio can be combined together as a single multiplexed track (in muxed mode), where each segment in

the track contains the associated video and audio content.

Alternatively, the video and audio content can be kept separately as demultiplexed tracks (in demuxed mode), as shown in the image below.



ABR Protocols

DASH and **HLS** are the two predominant ABR streaming protocols. Both of them now support demuxed audio and video tracks.

The DASH protocol defines an **Adaptation Set** as a set of interchangeable encoded versions of one or several media content components. For demuxed audio and video tracks, it defines one Adaptation Set for video tracks and another set for audio tracks. The bandwidth requirement of each track is specified in the bandwidth attribute.

In HLS, a top-level master playlist uses **EXT-X-STREAM-INF tag** to specify a set of audio and video track combinations. It could be all the desired combinations, or it could be a subset of combinations. The bandwidth requirement is defined in the bandwidth attribute for each combination (instead of each audio/video track).

The Issue

While demuxing can increase your options and reduce CDN costs, limitations in handling demuxed audio can create new problems.

One issue we see are video streams that stall because of a late arriving audio segment.

Because demuxed streams decouple the video and the audio, losing sync between the video and the audio can also become an issue.

Another issue is the common assumption that audio bitrates are always significantly lower than video bitrates. This is interpreted as meaning audio track decisions have little impact on the video track selection. However, increasingly, an audio track can be of a much higher bitrate than some video tracks.

Yet another challenge with demuxing is the increased complexity. This complexity means that audio cannot be an afterthought. Even in relatively simple combinations, demuxing requires attention to all the variants. As an example, the

table below lists 18 audio and video (i.e. V1 + A1) combinations for one sample stream.

Video / Audio Combinations
Video Quality 1 + Audio 1
Video Quality 1 + Audio 2
Video Quality 1 + Audio 3
Video Quality 2 + Audio 1
Video Quality 2 + Audio 2
Video Quality 2 + Audio 3
Video Quality 3 + Audio 1
Video Quality 3 + Audio 2
Video Quality 3 + Audio 3
Video Quality 4 + Audio 1
Video Quality 4 + Audio 2
Video Quality 4 + Audio 3
Video Quality 5 + Audio 1
Video Quality 5 + Audio 2
Video Quality 5 + Audio 3
Video Quality 6 + Audio 1
Video Quality 6 + Audio 2
Video Quality 6 + Audio 3

Beyond the issues above, limitations in both the protocols and the players can cause undesirable behavior, such as selection of potentially undesirable combinations, such as very low quality video combined with very high quality audio.

Protocol Limitations

There are key differences between DASH and HLS specifications in the treatment of demuxed audio and video tracks. These differences have implications for player design and the resulting QoE.

One difference between the two protocols is that DASH does not provide a mechanism to specify the desired subset of combinations. HLS provides an explicit mechanism to do so. This can make DASH more vulnerable to choosing undesirable combinations.

The player typically lacks detailed domain knowledge of the content being streamed. Therefore when streaming DASH content, the player is forced to either consider all possible combinations of video and audio track variants, or apply its own policy to determine the subset of allowed combinations.

Both approaches have drawbacks. The first option could lead to a player selecting an unsuitable combination for certain types of content. The second option could potentially exclude more desirable combinations.

Another key difference between DASH and HLS is that DASH provides bandwidth requirements of individual audio and video tracks, while HLS provides the total bandwidth requirements of audio and video combinations. This difference should be recognized by the player to avoid undesirable behaviors.

Player Logic Limitations

When a player does not perform rate adaptation for audio it can lead to poor performance, such as a large amount of rebuffering. Therefore, it is a good idea to perform rate adaptation for both audio and video.

- Some players select undesirable audio and video combinations, which can lead to poor viewing quality. An example would be combining the lowest quality video and highest quality audio tracks.
The problem is more severe for DASH, which, as mentioned above, does not provide a mechanism to specify a subset of desirable audio and video combinations, making the player more vulnerable to selecting bad combinations.
- Some players make the rate adaptation decisions for audio and video independently. Considering audio and video jointly is better than considering them independently. But considering them jointly in an overly simplified way is also not desirable. Following this practice in certain players can lead to frequent audio/video track changes.
- Some players sync audio and video buffer on a very coarse granularity. This can lead to unbalanced content in the buffer. Since we need both video and audio in the playback, having a lot more content in video/audio is not helpful. Instead, it is better to use a finer granularity when syncing them.
- An HLS manifest can specify a subset of audio and video combinations, but some players do not conform to the manifest file. For example, ExoPlayer may select some undesirable combinations that are not in the manifest file.

Best Practice Recommendation

Below are some Best Practice recommendations for demuxed streaming

Server-Side Recommendations

Audio and video combinations. In the manifest file, we recommend specifying the allowed combinations of video and audio track variants, which is the subset of all possible combinations. A player can only select segments from the allowed combinations.

This practice allows the content owner to specify combinations that are suitable for the specific content, such as a music video versus an action movie, or specific device configurations, such screen size, or sound system. It also simplifies the rate adaptation task for the player, which helps the player make better choices.

HLS already supports this feature, so it is a good idea to leverage it. Do not specify all possible combinations unless they are all desirable.

DASH at present does not offer a way to list only specific allowed combinations in the manifest. A practical short term workaround would be for the client to get

this information from the server out-of-band, e.g., via an auxiliary file download over HTTP2. In the long term, the DASH specification can be expanded to support this feature.

Audio and video bandwidth declaration. In the manifest file, it is good to specify sufficient information about the aggregate bandwidth requirements of audio and video combination, as well as the bandwidth requirements of individual audio/video tracks.

This is particularly important when audio and video are fetched over different network paths that have different network characteristics (e.g., when they are stored at different servers).

Currently, DASH specifies the bandwidth for each audio and video track, and the aggregate bandwidth requirement for audio and video combinations (if provided following the earlier suggestion) can be calculated from the individual tracks.

HLS uses two levels of manifest files: the top-level master playlist that specifies the aggregate bitrate for the combination of the audio and video, and the second level media playlists, each providing detailed information of the individual audio/video segments inside a track.

When the packaging is in Fragmented MP4 format, the media playlists specify the EXT-X-BYTERANGE information, which can be used to obtain the audio/video bitrate (the byte range information is used to specify the start and end byte positions to fetch the content, but can also be used to obtain segment size, and hence bitrate with the segment duration information specified in the media playlists).

When the packaging is in MPEG-2 TS format, no byte range information is provided.

On the other hand, HLS provides an optional EXT-X-BITRATE tag that can be used to obtain per-segment bitrate information. This option needs to be made mandatory. In addition, for both packaging formats, since the information is in per track manifest files, we suggest that the player can download the per-track manifests and read the information at the very beginning, before any adaptation decision is made.

Client-Side Recommendations

Adopt audio adaptation. For dynamic network bandwidth scenarios, it is important to perform audio adaptation.

High quality audio tracks can have higher bitrate than that of low video tracks. Audio adaptation is just as important as video adaptation to avoid adverse impact on QoE.

Joint adaptation of audio and video. We recommend that the selection of the audio, the selection of the video variants (tracks) and the positioning in playback order, be considered jointly.

In addition, only consider the candidates from the audio and video combinations specified in the manifest file, if provided.

The combinations listed in the manifest file suggests the desired audio and video combinations based on the content. Therefore, it is important for the client to

select only from those combinations. For both video and audio adaptation, it is desirable to satisfy conflicting goals in maximizing quality, minimizing stalls and minimizing quality variation.

Since the selection of audio and video is inherently coupled, in order that good combinations of audio and video tracks are chosen, we recommend considering the specified combinations of audio and video (i.e., those specified in the manifest file if provided) while making rate adaptation decisions, instead of considering audio and video individually. The rate adaptation should be done carefully to avoid frequent changes in either audio or video tracks.

Maintain a balance between audio and video prefetching. It is recommended keep the audio and video buffer level (in seconds) balanced.

This is because either empty audio or video buffer leads to stalls. It makes no sense no having many audio segments ready if the next video segment is not available, and vice versa.

The balance can be achieved by synching the duration of prefetched audio and video content at a fine granularity, for example, at the segment level or in terms of a small number of segments.

Video Optimizer

If your stream is demuxed, with separate audio and video segments, Video Optimizer breaks out the segments on the Video Tab.

Below is a screenshot from the Video tab, showing separate video and audio in their own tables. You can control the view through checkboxes, displaying just video, just audio, or both video and audio side by side, as seen below.

The screenshot shows the 'Video Stream' interface with a 'Load' button and a status bar indicating 'Stream: b048b329-ad6a-47e4-a7f4-06e81e5f1550_corrected.mpd, segment count:135'. Below the status bar are checkboxes for 'set StartupDelay', 'AMVOTS', 'Video', and 'Audio'. The 'Video' and 'Audio' checkboxes are checked. The interface displays two tables: 'Video' and 'Audio'.

Segment	DL	End Time	Start Time	PlayTime	Duration	Track	TCP State
1	75.957	0.000	0.000000	9	4.963292		
1	76.973	0.000	109.200000	0.000000	10	4.963292	
2	76.976	0.000	-	4.963292	9	4.421083	
2	77.278	0.000	-	4.963292	9	4.421083	
2	77.485	0.000	114.163292	4.963292	10	4.421083	
3	79.970	0.000	118.584375	9.384375	10	4.462792	
4	80.177	0.000	123.047167	13.847167	10	4.963292	
5	81.301	0.000	128.010458	18.810458	10	4.546208	
6	82.533	0.000	132.556667	23.556667	10	4.587917	
7	83.922	0.000	137.144583	27.944583	10	3.795458	
8	84.798	0.000	140.940042	31.740042	10	2.544208	
9	88.255	0.000	143.484250	34.284250	10	5.005000	
10	91.434	0.000	148.489250	39.289250	10	4.170833	
11	108.955	0.000	152.660083	43.460083	10	4.045708	fin
12	99.109	0.000	-	47.505792	10	4.671333	
12	110.291	0.000	156.705792	47.505792	10	4.671333	fin
13	117.238	0.000	161.377125	52.177125	10	4.337667	fin
14	121.634	0.000	165.714792	56.514792	10	2.794458	fin
15	123.697	0.000	168.509250	59.309250	10	3.044708	fin
16	125.231	0.000	171.553958	62.353958	10	3.086417	fin
17	126.773	0.000	174.640375	65.440375	10	2.585917	fin
18	128.916	0.000	177.226292	68.026292	10	4.129125	fin
19	131.555	0.000	181.355417	72.155417	10	3.962292	fin
20	137.985	0.000	185.317708	76.117708	10	2.711042	fin

Segment	DL	End Time	Start Time	PlayTime	Duration	Track	TCP State
1	76.161	0.000	109.200000	0.000000	1	2.005333	
1	76.875	0.000	109.200000	0.000000	1	2.005333	
2	76.873	0.000	111.205333	2.005333	1	2.005333	
2	77.485	0.000	111.205333	2.005333	1	2.005333	fin
3	78.722	0.000	113.210667	4.010667	1	2.005333	
3	89.594	0.000	113.210667	4.010667	1	2.005333	fin
4	79.662	0.000	115.216000	6.016000	1	1.984000	fin
4	80.485	0.000	115.216000	6.016000	1	1.984000	
5	81.093	0.000	117.200000	8.000000	1	2.005333	
5	83.764	0.000	117.200000	8.000000	1	2.005333	
6	81.301	0.000	119.205333	10.005333	1	2.005333	fin
7	83.450	0.000	121.210667	12.010667	1	2.005333	
8	82.533	0.000	123.216000	14.016000	1	1.984000	fin
9	85.912	0.000	125.200000	16.000000	1	2.005333	
10	85.099	0.000	127.205333	18.005333	1	2.005333	
11	89.490	0.000	129.210667	20.010667	1	2.005333	
12	91.334	0.000	131.216000	22.016000	1	1.984000	
13	92.360	0.000	133.200000	24.000000	1	2.005333	
14	94.197	0.000	135.205333	26.005333	1	2.005333	
15	99.213	0.000	137.210667	28.010667	1	2.005333	
15	107.307	0.000	137.210667	28.010667	1	2.005333	
16	100.145	0.000	139.216000	30.016000	1	1.984000	
16	107.622	0.000	139.216000	30.016000	1	1.984000	
17	108.449	0.000	141.200000	32.000000	1	2.005333	

This allows quicker detection of behavior on demuxed streams that would cause issues, such as late-arriving audio segments that would cause stalling.

Learn More

You can get a lot more information about video streaming and other mobile development practices on our [Mobile Development Best Practices](https://developer.att.com/mobile-development/best-practices) web site.

ADDITIONAL RESOURCES

[Release Notes](#)

[User Guide](#)

[Best Practices](#)

SUPPORT

[Contact Us](#)

[FAQs](#)

[Twitter](#)



[Terms of Use](#) [Privacy Policy](#) [Your Privacy Choices](#)  

©2025 AT&T Intellectual Property. All rights reserved

AT&T, the AT&T logo and all other AT&T marks contained herein are trademark of AT&T Intellectual Property and/or AT&T affiliated companies.