Addressing IP Video Adaptive Stream Latency and Video Player Synchronization

A Technical Paper prepared for SCTE/ISBE by

Jeffrey Tyre
Distinguished System Engineer
ARRIS
2450 Walsh Ave,
Santa Clara, CA 95051
408-940-2095
jeffrey.tyre@arris.com

Wendell Sun
Member of Technical Staff
Viasat
6155 El Camino Real
Carlsbad, CA 92009
760-893-5577
wendell.sun@viasat.com

Table of Contents

List of Figures

List of Tables

Introduction

Since early deployment of IP-based video networks, various technologies have emerged to help cope with the variability associated with delivering video over non-deterministic, best effort IP networking. In managed environments, IPTV operators have traditionally used MPEG-2 TS as the transport mechanism for video over IP networks. Hyper Text Transfer Protocol (HTTP), by virtue of being the content transport protocol for web-based applications, is almost ubiquitously used for video delivery over the Internet.

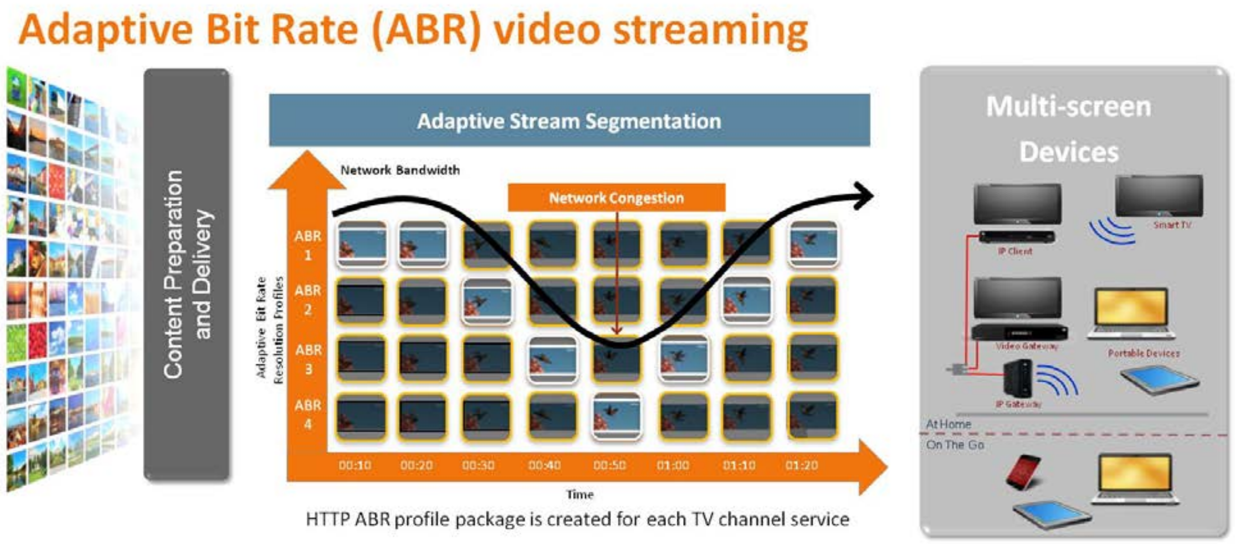
Traditionally HTTP video was delivered by progressive file download. However, a newer technology called adaptive bit rate (ABR) streaming has become widely used. ABR streaming promises to enable videos to be delivered over unmanaged networks with a very high quality of experience, and is thus applicable to both Internet video environments and managed video networks that are seeking to extend the delivery of premium content to devices other than the television set. ABR streaming has emerged as a technology of choice for many types of video delivery. For managed-networks, Pay TV Operators migration to adaptive streaming fits into an overall strategic objective for a unified, video delivery IP network infrastructure supporting every device screen and all subscriber services using web and cloud-based technologies.

Unlike previous HTTP video technologies, such as progressive download, adaptive streaming introduces the ability to dynamically react to changes in network conditions by switching to a video encoded at a different bit rate. This ability to adapt in real time more accurately reflects the dynamic conditions of today’s networks, content, and devices. With users streaming more premium long form content, it is natural to expect that there will be fluctuations in the amount of bandwidth available during a two-hour movie, than say a 3-minute video clip. Adaptive streaming is a recognition of this fact and enables viewers to watch this premium content with a superior quality of experience (QoE).

Adaptive streaming works by leveraging the same content encoded in various bit rates—in a range that reflects the expected quality of the content itself, the network performance, and the screen resolution desired. For example, a video could be encoded in bit rates ranging from 300 kbps (low-quality, online video) up to 6 Mbps or higher (high quality streaming content to the TV). A typical video could be encoded in as many as eight different bit rate profiles, depending on the range of devices and quality desired. Each of these files are then further segmented—or “chunked”—into short segments (typically two to ten seconds long) that are each precisely time-stamped.

As the video is delivered, the HTTP client maintains a communication channel with the adaptive bit rate server. The client (the viewing device) downloads these chunks as individual files, which are buffered by the client, decoded, and played out as a continuous presentation of video and audio to the viewer. During the viewing session, the client player monitors the rate at which the buffer is filling and can thereby infer the performance of the network.

If there is degradation in network performance, the client can request that chunks be delivered from one of the lower bit rate files. This is all seamless to the viewer since each source file is chunked and time-stamped in the same, very precise intervals—so there is no visible interruption or hesitation when switching to a different bit rate. Likewise, if the player detects an improvement in performance, it can request HTTP file segments from one of the higher bit rates.



Adaptive Bit Rate (ABR) video streaming
Content Preparation and Delivery
Adaptive Stream Segmentation
Network Bandwidth
Network Congestion
Adaptive Bit Rate Resolution Profiles
ABR 1
ABR 2
ABR 3
ABR 4
00:10
00:20
00:30
00:40
00:50
01:00
01:10
01:20
Time
HTTP ABR profile package is created for each TV channel service
Multi-screen Devices
Smart TV
IP Client
Video Gateway
Portable Devices
IP Gateway
At Home
On The Go

Figure 1 – Adaptive Bit Rate (ABR) Ecosystem in IP Networks

Since adaptive streaming video content is prepared and conditioned into multiple bit rate sources to allow the client player to dynamically select the appropriate bit rate source and seamlessly switch between different bit rate sources per broadband network status, it can be used by almost any type of multiscreen device, provide transport resiliency to a network's condition and gives a much better user viewing experience.

For the video on demand (VOD) type of service, this is near perfect since VOD service is delivered to each end device individually in non-real-time fashion, e.g. IP unicast, and there is no concern for delivery latency and no demand to coordinate viewing experience among end users. The initial buffering delay most likely is acceptable, especially when there is a pre-roll ad play. The commercial application is very successful for Internet-based VOD services, such as Netflix, Hulu, Amazon, etc. However, the HTTP-based transport has major shortcomings when it is applied for live or linear video delivery.

Content

1. Live / Liner TV Service Requirements

The live or linear TV service presents some unique challenges for using adaptive streaming technologies including some of the following characteristics:

- There is a real-time timeline that is referenced by all viewing users at the same time when the content is captured, processed, delivered, and consumed. For example, a sport event is being broadcasted while the event is happening
- Compared with an audience on site, normally there is a constant viewing delay for broadcast viewers. The viewing delay is counted from event happening to being viewed remotely and usually is caused by video content acquisition, editing, processing and delivering, as well as mandatory regulation delay request. The smaller the viewing delay is, the better the user viewing experience will be. See the Figure 2 for reference
- Content is delivered to a large number of viewers simultaneously
- Constant viewing delay is the same to all viewers

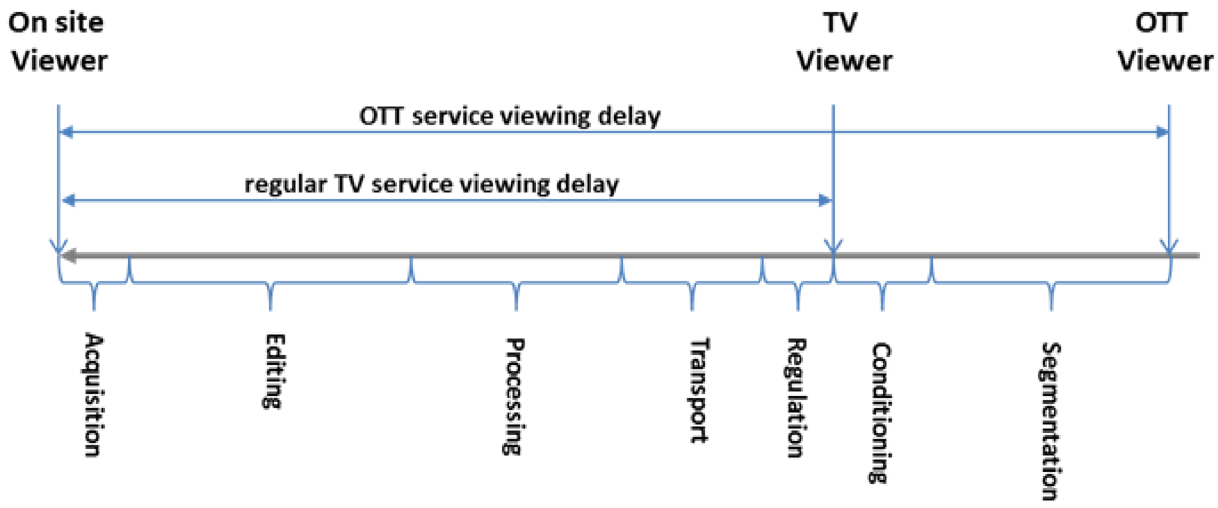


Figure 2 - TV Service Viewing Delays

In the ABR case, HTTP is used for content delivery. In general, HTTP is a transaction based protocol designed for file download. The current adaptive bit rate streaming uses small segments to compromise HTTP file transfer request. Instead of bit by bit streaming, content is transported segment by segment.

Normally an HTTP transfer will not start until the whole segment is ready. This will add, at minimum, one segment length of extra time to delivery delay besides other processing delays. This is shown in figure 2 as segmentation delay. The bigger the segment is, the longer the extra latency is. This may force small segments to be defined if low latency live TV service is desired.

However, for seamless switching purposes, a segment is bounded with an instantaneous decoding refresh (IDR) frame, e.g. closed group of pictures (GOP), which requires more coding bandwidth. The smaller the segment, the worse the video encoding efficiency will be. In addition, each segment delivery corresponds with one HTTP get/reply transaction. The smaller the segment is, the more HTTP protocol overhead will be in network.

Before Internet-based video services were available, IP-based streaming TV services, such as IPTV service, had a long history. IPTV services have been offered by telco operators to compete with cable operators for more than a decade. IPTV services provide subscribers with similar TV viewing experience, such as fast channel change time and low end-to-end transport latency, just like traditional broadcast or linear Pay TV service offers. It uses IP multicast as its primary protocol, which provides true data streaming, minimizes delivery latency, and supports content sharing among multiple clients. However, IPTV services require guaranteed the bit rate to match the bandwidth for smooth service delivery. As a result, an IPTV service is only provided in a managed IP network.

This paper presents how adaptive transport stream (ATS) segment markers, HTTP chunked transfer encoding (CTE), and ABR playlist manifests can be combined into a solution optimized for live video content delivery and become an effective tool for Pay TV Linear Services. It briefly reviews current IP multicast streaming and examines existing or under-developing protocols and standards, such as the HTTP chunked transfer encoding [7] and the CableLabs/SCTE adaptive transport stream [3]. This paper proposes a solution, which combines ABR encoding with content segment markers and HTTP CTE streaming, to minimize the delivery latency for live/linear video service without sacrificing video encoding efficiency. Additionally, a general modification to ABR manifest formats is proposed to address ABR player synchronization challenge.

2. TV Services via Satellite Broadcast and IP Multicast Streaming

Traditional TV services are all in broadcast mode. The content is acquired from source, edited, and processed for transportation over satellite. Satellite is used for large area content distribution. Depending on the business model, there are a couple of ways of receiving content for consumption:

- For Pay TV cable operators, an integrated receiver/decoder (IRD) at the multichaiidnnel video programming distributor (MVPD) headend receives and decodes the content, and then the content is re-distributed to subscribers via the cable plant network.
- For Pay TV satellite operators, a satellite set-top box in the subscriber’s home receives, decodes, and renders the content.
- For over-the-air operation, it is similar to the Pay TV cable operation in the front line of receiving and decoding, but the last leg of content re-distribution is done by over-the-air radio transmission.

All of these are broadcast one-way from the content source to many content consumption terminals. This usually carries a constant viewing delay, as shown in Figure 2, as regular TV service viewing delay.

Another type of TV service, IPTV, uses broadband Internet as its distribution network. The Internet can support different types of content distributions, such as one-to-one unicast, one-to-many multicast, and one-to-any broadcast. However, IP multicast has significant advantages over IP unicast and broadcast for TV-like services.

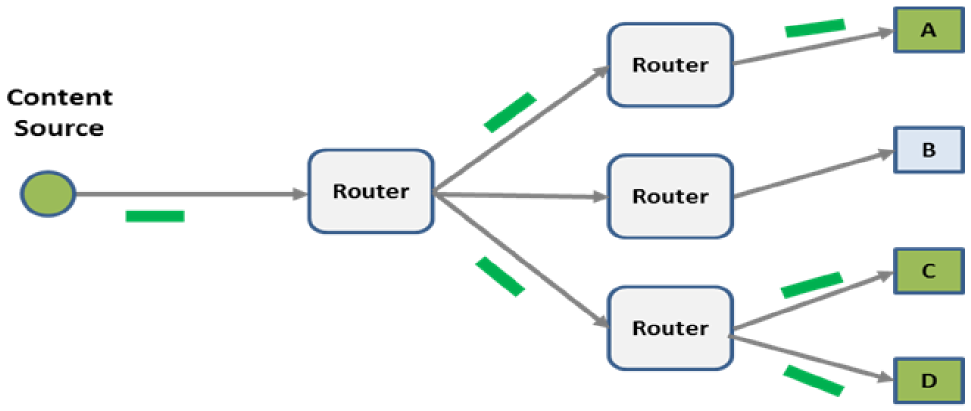


Figure 3 - IP Multicast Illustration

IP Multicast offers a one-to-many kind of distribution model, similar to traditional broadcast TV services that are mentioned above with one difference. In the world of the public Internet, IP multicast supports content delivery on per request or per registration basis. To receive content, the end user's device needs to send Internet Group Management Protocol (IGMP) requests to its router to ask for delivery. If the requested content is not available on the end user's device, the router must relay the request all the way back to the content source. To stop receiving content, the end user device can send an IGMP quit command to its router to stop the sending. The user B in Figure 3 is exempted from receiving content. The routers that support IP multicast service only send one copy of same content to the duplicated requests from the next router. For example, if both users C and D ask for the same content, the upstream router just needs to send one copy of content to their router. In this case, even if there is a single user, such as user A, or there are thousand users under the same router, the overall network traffic will look same. Thus it scales very well and is quite efficient for live or linear type of content distribution. Robinson's paper [1] has a lot more details on IP multicast discussion.

More importantly, IP multicast is a true IP streaming protocol. Although content is delivered by IP packets, the protocol is designed for content flowing from point A to point B as long as point B joins the multicast group. IP packets are relatively, very small and do not introduce much extra transportation delay compared with other forms of bit-stream content delivery.

IP multicast has been used extensively by telco operators to provide IPTV service based on delivering MPEG-2 transport streams (TS) to the telco set-top box in the home. It allows the IPTV service to provide a similar user experience as traditional broadcast TV service does. Even for some cable operators, IP multicast may also be used in their backbone for content distribution due to the popularity of Internet and IP network integration.

To supply a satisfied IPTV service or a reliable backbone for content delivery, the only constraint of using IP multicast is the requirement of guaranteed bandwidth – that the bandwidth should be equal or higher than the bit rate of the content stream. For this reason, IP multicast is usually applied in a managed IP network with well-engineered bandwidth allocation.

Whether in a traditional broadcast TV service or in an IPTV service, MPEG-2 TS is used for carrying video and audio content. MPEG-2 TS is the currently the dominant container format for content distribution of most Pay TV services, and several key TS advances recently introduced are providing enhancements for adaptive content encoding in preparation of adaptive streaming delivery using HTTP as the transport protocol for multi-screen device services.

3. Adaptive Transport Stream and Segment Boundary Points Support

An adaptive transport stream (ATS) is a fully compliant MPEG-2 TS with an embedded segment boundary indicator. The ATS is designed for adaptive streaming, which requires encoding/transcoding of multiple bit rate (MBR) streams, segmentation, and alignment to support seamless bit rate switching. Originally the MBR encoding/transcoding is typically done by encoder or transcoder, while the segmentation and alignment is processed by the adaptive streaming packager. The purpose of ATS is to further exploit the encoding/transcoding process to label the media segment boundary, relieving the effort of parsing and aligning segments in the adaptive streaming packager.

The ATS definition originated in the CableLabs encoding boundary point (EBP) specification [2] and adaptive transport stream specification [3], in which an EBP structure is defined in the private data of the adaptation field in MPEG-2 TS header to indicate the beginning of each segment. The work was promoted to ANSI/SCTE standardization process, in which ANSI/SCTE 223 [4], has been created. Furthermore, it has been contributed to MPEG and an amendment to MPEG-2 TS specification [5] has been generated to define a set of adaptation field (AF) descriptors to serve the same purpose.

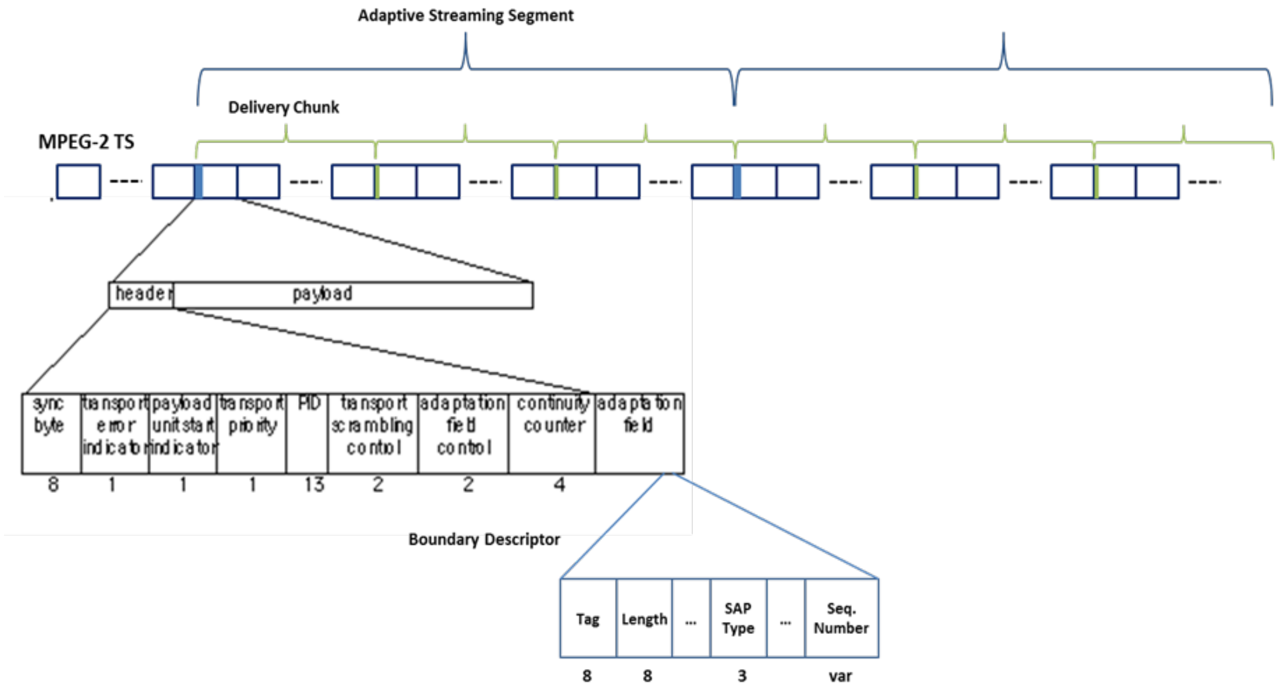


Figure 4 - Structure of Adaptive Transport Stream

As it is shown in Figure 4, the Boundary Descriptor is introduced into the adaptation field of MPEG-2 TS header. The two major parts are the “SAP type” and “sequence number” field. The SAP stands for stream access point and is defined by ISO/IEC 14496-12. A SAP type is a definition of media decoding attribute in that stream point. For example, a SAP type 1 means the media sample at the point can be fully decoded without referring other samples and all samples following it can also be correctly decoded. The sequence number field has length of 2, 4, or 8 bytes depending on application. It provides a unique identifier for a segment within its context.

To support regular MBR adaptive streaming, the signaling of a boundary descriptor in the segment level is good enough. The packager takes ATS as input stream and needs only to parse the bytes in the transport stream packet headers in order to obtain the boundary information. It does not need to parse any bytes in the packet payload. However, to resolve the extra viewing delay caused by segmentation as it is discussed in the introduction section for live/linear TV service, a smaller segment is desired; and coding efficiency should not be impacted. The segment is designed for bit rate adaptive switching; thus, it requires closed GOP at segment boundary. While the segment structure is maintained, a smaller delivery unit can be designed for low latency delivery.

The MPEG DASH specification [6] introduces a concept of delivery unit media segment to support low latency application. As shown in the Figure 4 - Structure of Adaptive Transport Stream, the delivery chunk is a smaller delivery unit within the adaptive streaming segment. It can be made up by any group of meaningful coding samples, such as a GOP or even a frame, while making it small enough not to cause transportation delay. Similar to each segment, the delivery chunk can also be identified by the boundary descriptor embedded in MPEG-2 TS header.

The insertion of boundary descriptors into MPEG-2 TS headers can be easily achieved as part of the content encoding/transcoding process. It does not add much extra processing to the encoder/transcoder, yet it is a big savings for the packager to not need to look up the coding payload. It is even more essential when the segment durations are not fixed interval - whether that's due to flexible GOP structures in the encoded video, or a content-related change, such as a frame-accurate demarcation in a program or advertisement.

4. Addressing ABR Content Delivery Latency with HTTP Chunked Transfer Encoding Streaming

The HTTP chunked transfer encoding (CTE) is defined by HTTP/1.1: Message Syntax and Routing [7]. It is a data transfer mechanism in HTTP 1.1, in which data can be sent in a series of "chunks" in responding a single HTTP data request. It uses the transfer-encoding header, instead of the content-length header. The HTTP sender does not need to wait for the total size of content being available and can start sending data as small chunks with any amount available while still receiving the content. When the chunk is sent, its size is indicated in the transfer-encoding header. At the end of content, it sends the last chunk with its size set to be zero. For example, the following is a HTTP transaction with chunked transfer encoding:

```
GET /cte-example.html HTTP/1.1
Host: doc.micrium.com
User-Agent: Mozilla/4.61 [en] (Windows 7 6.1)
TE: chunked
```

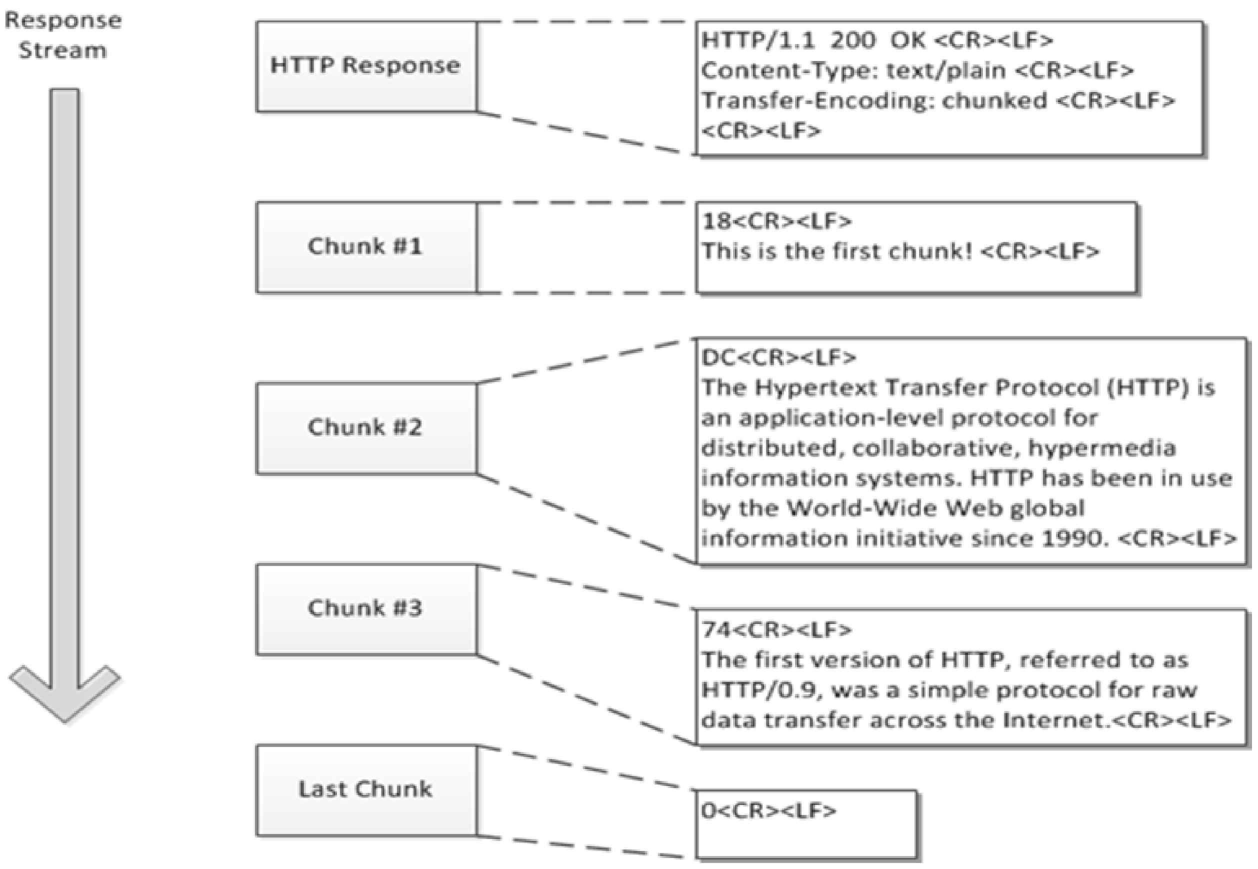

Response Stream
HTTP Response
HTTP/1.1 200 OK <CR><LF>
Content-Type: text/plain <CR><LF>
Transfer-Encoding: chunked <CR><LF>
<CR><LF>
Chunk #1
18<CR><LF>
This is the first chunk! <CR><LF>
Chunk #2
DC<CR><LF>
The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. <CR><LF>
Chunk #3
74<CR><LF>
The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet.<CR><LF>
Last Chunk
0<CR><LF>

Figure 5 - Example of HTTP Chunked Transfer Encoding Response [8]

As we have discussed in the adaptive transport stream section, ATS are MPEG-2 TS streams with embedded boundary descriptors to virtually divide them into delivery chunks, and further into adaptive streaming segments that are published to ABR players as ABR manifest playlists to select ABR content files. The client side remains with HTTP based ABR streaming, which allows content to pass most access networks and reach almost all types of client devices based on the ubiquity of HTTP content delivery.

The ABR-aware HTTP streaming (A2HS) server is a new component that bridges the two sides of ATS segmentation encoding and adaptive stream delivery [11]. It can essentially be an off-the-shelf HTTP web server with additional key adaptive streaming functions such as ATS parsing, ABR packaging with various ABR format outputs, and HTTP CTE support.

In current ABR content delivery implementations, a content segment is not made available in the manifest published to an ABR client until the segment is ready for delivery. This is the primary cause of content segmentation delay. In the proposed A2HS server solution, especially with the modified manifest playlist (see section below), the content segment(s) will be published in a manifest even before the input ATS is pulled, e.g. IP multicast join. When the A2HS server receives content segment requests from an ABR client and the corresponding IP multicast ATS stream is available (otherwise it initiates IGMP to join to the IP multicast group to get it), the server starts parsing the MPEG-2 TS header of the input ATS seeking the boundary descriptor of delivery chunks and sends the delivery chunks via the HTTP CTE method while it continues to receive the IP multicast ATS input.

The ABR client receives the manifest playlist, and it selects a segment from one bit rate stream per its condition, such as network bandwidth, screen size, etc. to begin pulling the segment from the HTTP server. Each pull is a HTTP file transfer transaction. And among a group of ABR clients, each client acts independently. If bit rate switching is required, the ABR client sends the next segment request of the switch-to ATS representation. Since ABR clients can only switch on an adaptive streaming segment boundary, the proposed A2SH server maintains transport of delivery chunks of current segment until the segment boundary indicated by the boundary descriptor is reached, then it starts transport of the first delivery chunk of the switch-to bit rate segment. In this fashion, the content delivery is in the delivery chunk interval. If the delivery chunk is designed small enough, the HTTP CTE based transport provides a similar function of true video transport streaming.

5. Addressing ABR Player Synchronization Using an Adaptive Bit Rate-Tiered Manifest Playlist

5.1. Current Manifest File Formats and Unsynchronized Video Playout

All HTTP based adaptive streaming approaches currently use a manifest playlist file with segmented content for downloading by ABR video players. The ABR client receives the manifest playlist, and it selects a segment from one of bit rate stream per its condition, such as network bandwidth, screen size, etc. to start pull the segment from HTTP server. Each pull is a HTTP file transfer transaction. And among a group of ABR clients, each client acts independently.

For live/linear TV service, all clients are supposed to pull the same media segment simultaneously, thus giving viewers a synchronized viewing experience. In practice, this does not occur when using adaptive streaming due to the nature of when individual ABR clients initiate their HTTP session requests in reference to the ABR manifest playlist of published ABR segments. The problem may get worse when individual client runs with different segment selection algorithm.

Table 1 - Manifest Example Using Apple HTTP Live Streaming for a Particular Bit Rate Stream

At the moment of 12:00:00	At the moment of 12:00:07	At the moment of 12:00:10
#EXTM3U #EXT-X-TARGETDURATION:10 #EXTINF:10, ./segment**3511**.ts #EXTINF:10, ./segment**3512.**ts #EXTINF:10, ./segment**3513**.ts #EXT-X-ENDLIST	#EXTM3U #EXT-X-TARGETDURATION:10 #EXTINF:10, ./segment**3511**.ts #EXTINF:10, ./segment**3512**.ts #EXTINF:10, ./segment**3513**.ts #EXT-X-ENDLIST	#EXTM3U #EXT-X-TARGETDURATION:10 #EXTINF:10, ./segment**3512**.ts #EXTINF:10, ./segment**3513**.ts #EXTINF:10, ./segment**3514**.ts #EXT-X-ENDLIST

If there are four clients, each starts playback individually, the clients may end up with the following result of segment pulling and playback:

- Client A asks and receives the manifest file at time 12:00:00, and it starts playback of segment 3511

- Client B asks and receives the manifest file at time 12:00:07, and it also starts playback of segment 3511, but comparing with Client A, it is 7 seconds behind in real time

- Client C asks and receives the manifest file at time 12:00:07, and it starts playback of segment 3512, at this moment, it is 3 seconds ahead of Client A and 10 seconds ahead of Client B in real time

- Client D asks and receives the manifest file at time 12:00:10, and it starts playback of segment 3514, at this moment, it is 20 seconds ahead of Client A, 27 seconds ahead of Client B, and 17 seconds ahead of Client C in real time

The time difference of playback among these clients is between 3 – 27 seconds.

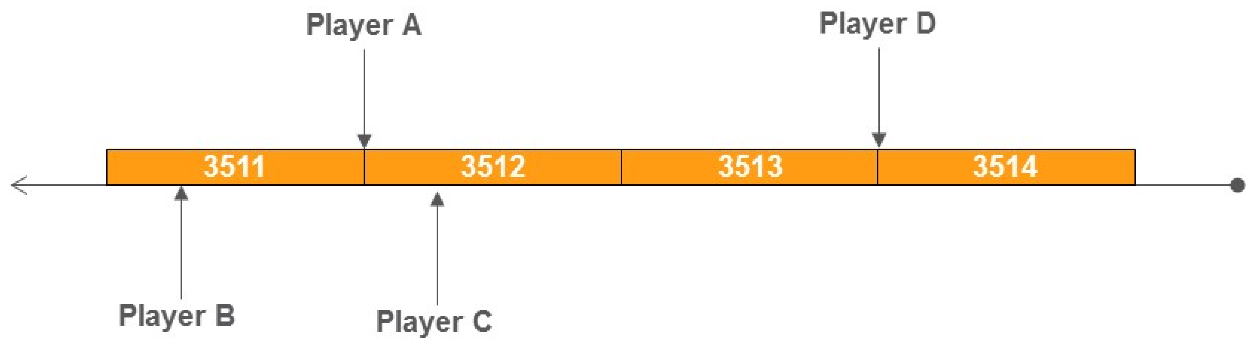


Figure 6 - Unsynchronized Playback Based on Current Manifest File Formats

5.2. A Proposed Adaptive Bit Rate-Tiered Manifest Playlist

For a live/linear media service, we are introducing a manifest playlist file that only requires listing different bit rate tier representations. This differs from current ABR manifest formats by not being dependent on individual available segment URLs but instead using a single "virtual" segment URL for the selection of bit rate stream tier by ABR players. The revised manifest file design can optimize the solution by adding one more entry to represent a "zero bit rate" stream, which can be used by the ABR client to signal stopping the HTTP CTE server's media segment output.

For example:

```
#EXTM3U

#EXT-X-TARGETDURATION:10

./segment-720p-3000kbps.ts

#EXTINF:10,

./segment-480p-800kbps.ts

#EXTINF:10,

./segment-320p-500kbps.ts
```

```
#EXTINF:10,

./segment-0kbps.ts

#EXT-X-ENDLIST
```

The target segment duration (e.g. 10 seconds) in the manifest file indicates time interval of virtual segment boundary that can be used for seamless bit rate switching.

5.3. Combining the Adaptive Bit Rate-Tiered Manifest with HTTP CTE Streaming

The HTTP server, which supports HTTP chunked transfer encoding, receives MPEG ATS streams with media segments and maintains the status of available media segments. At any moment, there is only one media segment called the "current segment", which matches to current media presentation time, for each bit rate stream. This current segment definition moves along with the timeline of media presentation.

Each segment is delivered incrementally in small chunks per chunked transfer encoding, such as one second, half second chunk or even smaller chunk duration to match with video frame, the HTTP server also maintains one current chunk position within the current segment to even closely represent the current moment of media presentation time.

As usual, the playback client initiates a request to receive the manifest file, then it selects one bit rate stream to start for playback, and asks for delivery of media segment via the "virtual" segment URL. Since the HTTP server supports chunked transfer encoding, the client does not need to repeat the media segment request, instead the HTTP server keeps pushing the chunks (and the segments) one after another until either:

- The client elects to switch bit rate streams, then sends a request for the new bit rate stream

or

- The client decides to stop receiving media segment/chunk, and sends a request for zero bit rate stream

When the HTTP server receives the media segment request by the "virtual" segment URL, it simply sends back the current chunk of the current segment in the matched bit rate stream, and keeps doing so for the follow up chunks/segments. If the server receives a request for bit rate switch, it will continue and finish sending chunks in the current bit rate and switch to the chunk of the new bit rate in the next segment boundary. It keeps on until it receives a request of zero bit rate stream, then it stops.

In this way, all playback clients receive the same media chunk if their initial media segment requests fall within media chunk interval. Since the chunk size is designed relatively small, such as one second or half second, it limits the synchronization gap to the minimum degree.

Taking the same example discussed in Table 1 with the current ABR manifest formats presented above:

- Client A asks and receives the manifest file at time 12:00:00, and it starts playback of chunk 3512-0 of segment 3512.
- Client B asks and receives the manifest file at time 12:00:07, and it receives and starts playback of chunk 3512-7 of segment 3512. Even though Client A, started 7 seconds earlier with chunk 3512-0, it now also plays back the same chunk.
- Client C asks and receives the manifest file at time 12:00:07, and it should have same result as Client B does.
- Client D asks and receives the manifest file at time 12:00:10, and it starts playback of chunk 3513-0 of segment 3513. At this moment, Client A, B and C should also start playback of the same chunk.

"*" indicates the current chunk of the current segment being delivered to each ABR player with a chunk size assumed to be a 1 second duration.

At the moment of 12:00:00	At the moment of 12:00:07	At the moment of 12:00:10
#720p-3000bps	#720p-3000bps	#720p-3000bps
./segment**3511**.ts	./segment**3511**.ts	./segment**3512**.ts
./segment**3512-0**.ts *	./segment**3512-0**.ts	./segment**3513-0**.ts *
./segment**3512-1**.ts	.	./segment**3513-1**.ts
.	.	.
.	./segment**3512-7**.ts *	.
./segment**3512-8**.ts	./segment**3512-8**.ts	./segment**3513-8**.ts
./segment**3512-9**.ts	./segment**3512-9**.ts	./segment**3513-9**.ts
./segment**3513**.ts	./segment**3513**.ts	./segment**3514**.ts
#480p-800bps	#480p-800bps	#480p-800bps
./segment**3511**.ts	./segment**3511**.ts	./segment**3512**.ts
./segment**3512-0**.ts *	./segment**3512-0**.ts	./segment**3513-0**.ts *
./segment**3512-1**.ts	.	./segment**3513-1**.ts
.	.	.
.	./segment**3512-7**.ts *	.
./segment**3512-8**.ts	./segment**3512-8**.ts	./segment**3513-8**.ts
./segment**3512-9**.ts	./segment**3512-9**.ts	./segment**3513-9**.ts
./segment**3513**.ts	./segment**3513**.ts	./segment**3514**.ts
#320p-500bps	#320p-500bps	#320p-500bps
./segment**3511**.ts	./segment**3511**.ts	./segment**3512**.ts
./segment**3512-0**.ts *	./segment**3512-0**.ts	./segment**3513-0**.ts *
./segment**3512-1**.ts	.	./segment**3513-1**.ts
.	.	.
.	./segment**3512-7**.ts *	.
./segment**3512-8**.ts	./segment**3512-8**.ts	./segment**3513-8**.ts
./segment**3512-9**.ts	./segment**3512-9**.ts	./segment**3513-9**.ts
./segment**3513**.ts	./segment**3513**.ts	./segment**3514**.ts

Table 2 - Example of CTE Server Output Using Proposed Manifest Format in Delivering Adaptive Bit Rate-tiered Manifests and HTTP CTE Delivered Chunks

The overall solution achieves playback synchronization among all playback clients with low latency delivery. A less than chunk-size time difference of playback should exist among those clients.

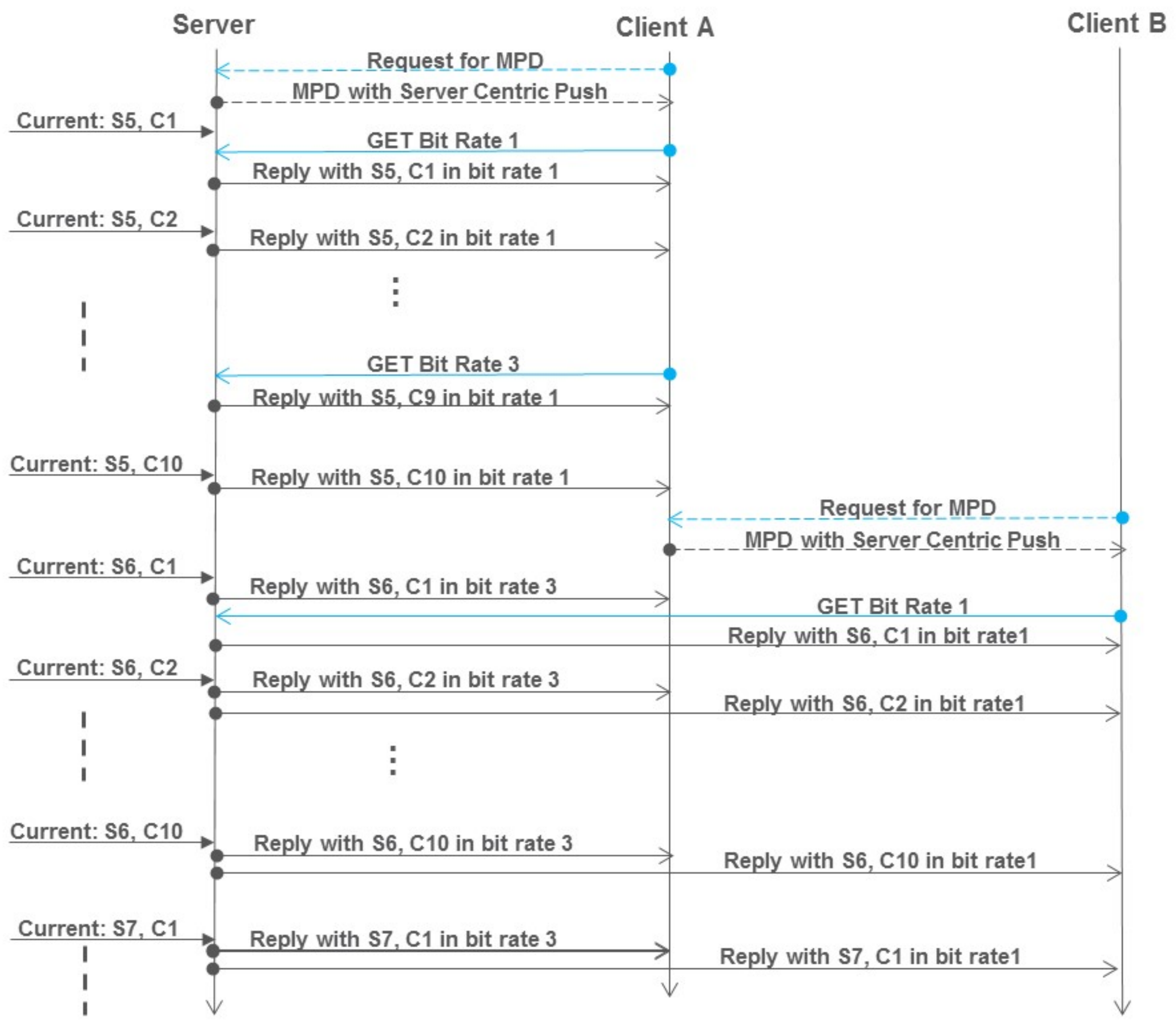

Server
Client A
Client B
Request for MPD
MPD with Server Centric Push
Current: S5, C1
GET Bit Rate 1
Reply with S5, C1 in bit rate 1
Current: S5, C2
Reply with S5, C2 in bit rate 1
GET Bit Rate 3
Reply with S5, C9 in bit rate 1
Current: S5, C10
Reply with S5, C10 in bit rate 1
Request for MPD
MPD with Server Centric Push
Current: S6, C1
Reply with S6, C1 in bit rate 3
GET Bit Rate 1
Reply with S6, C1 in bit rate1
Current: S6, C2
Reply with S6, C2 in bit rate 3
Reply with S6, C2 in bit rate1
Current: S6, C10
Reply with S6, C10 in bit rate 3
Reply with S6, C10 in bit rate1
Current: S7, C1
Reply with S7, C1 in bit rate 3
Reply with S7, C1 in bit rate1

Figure 7 – ABR Client Session Flow with Proposed Adaptive Bit Rate-tiered Manifests

As illustrated in Figure 7, ABR client session synchronization is maintained even when individual clients are selecting content from different adaptive bit rate tier levels. This allows each ABR client to monitor quality of service (QoS) network conditions specific to it, which is one of the cornerstones of adaptive bit rate streaming, and effectively spreads the load of monitoring of IP Video network conditions across the entire population of ABR clients being served.

Conclusion

Compared with existing TV services, especially live and linear TV service, HTTP-based video services add extra viewing delay caused by ABR segmentation. This paper reviewed the advantages of satellite based broadcast TV service and IP multicast based IPTV service, as well as the ANSI/SCTE ATS standard and HTTP CTE protocol.

The paper proposed combining ATS segmentation description for ABR content preparation and HTTP CTE for low latency content delivery to ABR clients with a bit rate-tiered ABR manifest for video / audio playout synchronization between a population of ABR clients. This approach can both minimize the extra viewing delay caused by ABR packaging segmentation and improve live TV services based on HTTP / adaptive streaming technology's quality of experience.

This paper demonstrated that combining multiple advances in the area of HTTP-based, adaptive streaming video delivery can be used to address two of the key challenges currently facing cable operators' migration from legacy to new IP video based, live TV services, potentially paving a path for investment in next generation technologies.

Abbreviations

A2HS	ABR-aware HTTP streaming
ABR	adaptive bit rate
AF	adaptation field
ATS	adaptive transport stream
CTE	chunked transfer encoding
EBP	encoding boundary point
GOP	group of pictures
HTTP	Hyper Text Transfer Protocol
IDR	instantaneous decoding refresh
IGMP	Internet Group Management Protocol
IRD	integrated receiver/decoder
MBR	multiple bit rate
MVPD	multichannel video programming distributor
QoE	quality of experience
QoS	quality of service
SAP	stream access point
SCTE	Society of Cable Telecommunications Engineers
TS	transport stream
TV	television
VOD	video on demand

Bibliography & References

The Return of Multicast: Why it Succeeds in a Live Linear World, D. Robinson, http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-Return-of-Multicast-Why-it-Succeeds-in-a-Live-Linear-World-108621.aspx

CableLabs OC-SP-EBP-I01-130118, Encoder Boundary Point Specification, www.cablelabs.com

CableLabs OC-SP-ATS-I01-140214, Adaptive Transport Stream Specification, www.cablelabs.com

ANSI/SCTE 223 2017, Adaptive Transport Stream, www.scte.org

ISO/IEC 13818-1:2015 PDAM 7 Virtual Segmentation

ISO/IEC 23009-1:2014, Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats

IETF RFC7230, Section 4.1, Chunked Transfer Encoding, http://tools.ietf.org/html/rfc7230#section-4.1

HTTP Chunked Transfer Encoding, https://doc.micrium.com/display/httpref/Chunked+Transfer+Encoding