AN PEER TO PEER TV NETWORK FOR SCALABILITY BASED ON LIVE VIDEO STREAMING SYSTEMS

MADA SURENDRA BABU 1*, S.SHANAWAZ BASHA 2*

1. *II-M.Tech, AVR & SVR Engineering College, KURNOOL.*
2. *ASSISTANT PROFESSOR, AVR & SVR Engineering College.*

Abstract: *The basic tenet of HTTP streaming is to deliver fragments of video and audio that are individually addressable chunks of content over HTTP. Some media players consume incoming video and audio data only in a time ordered multiplexed format. If alternate tracks need to be added post packaging of the media, it has to be repackaged that involves duplication resulting in multiple multiplexed files. Additionally for adaptive streaming, a set of all those files need to be added for each bitrate. Alternatively, it is more efficient to store component tracks separately, fetching only the required tracks and multiplexing audio and video in the client before sending the data to the decoder. To deliver an optimal viewing experience, the client has to take care of the seemingly conflicting constraints viz., handling the network jitter, minimizing the time to switch to an alternate track and minimizing the live latency. For instance, to absorb more network jitter more data should be available in the buffers but this would increase the switching latency. We introduce a formal buffer model for a client that gathers video and audio fragments and multiplexes them on the fly. This model uses separate video and audio buffers, a multiplexed buffer in the application, and decoding buffer associated with the decoder. We model the buffer sizes, their thresholds to request data from the network, and the rate of transfer of data between buffers. We show that these buffers can be designed varying these parameters to optimize for the above constraints. This buffer model can also be leveraged for deciding when to switch in adaptive bitrate streaming. We further validate these by experimental results from our implementation.*

1. INTRODUCTION

HTTP streaming has caught on in the last few years as an efficient simple streaming solution using the existing welloiled infrastructure of HTTP caches and web servers to deliver individually addressable chunks of media over the Hyper Test Transfer Protocol (HTTP) [1-3]. The basis of HTTP streaming is that the video asset is divided into small fragments, which are individually addressable by unique URIs. The client requests these fragments as required using the standard HTTP Request/Response mechanism. The downloaded fragments are played back sequentially for a seamless playback experience.

Realtime streaming and progressive download were predominantly used for delivering video before the advent of HTTP streaming. Realtime streaming works well for streaming, but is expensive since it requires dedicated servers and caches. In progressive download, the player renders the video that is already downloaded and buffered in the client, while the rest of the video gets downloaded simultaneously [4]. One of the pitfalls is that even if the user does not watch the entire video, it gets buffered in the client. Also, switching to a different bitrate midway in a presentation is difficult. HTTP streaming provides a cheaper alternative to realtime streaming by using the abundantly available standard web caches and HTTP servers. In HTTP streaming, only the portions to be watched are downloaded as fragments. At any point of time a switch to a different birate can be made by requesting the corresponding fragment of that bitrate. Thus, it alleviates the problems of progressive download. In addition, it can support streaming of live events.

**** MADA SURENDRA BABU***
II-M.Tech, AVR & SVR Engineering College, KURNOOL

Motivation. There has been a proliferation of video on the web, using HTTP streaming as the mechanism for delivery. Often, multiple tracks need to be present for a single content, such as multiple camera angles for a football game or multiple language tracks for a movie. In multiplexed stream systems, clients consume data in a multiplexed format (e.g., Flash video [5], MPEG2-TS [6]). When an alternate track has to be added to the media, the content has to be repackaged for those clients. This entails duplication of large amounts of data for accommodating the new track to create the multiplexed files. For adaptive streaming, these set of files need to be further duplicated for each bitrate. It is economical instead, to store the alternate tracks separately, and multiplex video and audio in the client, assuming the client has enough computation resources. In such clients, the simplest approach is to multiplex data as and when available and push the data to the decoder. This would give rise to the following problems in the client:

•for large fragments, the processing and multiplexing of large amount of bits will use up computing cycles for multiplexing starving the rendering thread,
•if the buffers run too low the capacity to handle network jitters will decrease,
•if the buffers run high, switching to a different track or bitrate will be delayed as the data existing in the decoding buffer has to be played out fully delaying the manifestation of the switch,

•if more data is required in the decoder to start playback to take care of jitter, it affects the live latency adversely,
•in memory constrained devices there may be a hard requirement on the size of buffers.

Our Solution. We proceed to address these problems by introducing a formal buffer model in the client that uses tunable buffer parameters such as sizes, thresholds, and rate of flow of data. In our model we use separate audio and video buffers to gather respective fragments, and keep the multiplexed data in a multiplexed buffer. Multiplexed data is then moved to a decoding buffer. We introduce thresholds for each buffer to trigger requests for fragments, and a rate to throttle the movement of data to the decoding buffer from the multiplexed buffer. We derive the relationships of these parameters to the client constraints viz., the buffer sizes, the live latency, the network jitter handling capacity, and the time to switch to a different track. We show that these parameters can be tuned for desired values of the above constraints. Contributions. The contributions of this paper are summarized as follows:

1)A buffer model in the client to multiplex video and audio on the fly with tunable parameters,
2)Analysis of the effects of thresholds and rate of movement of data among buffers to latency, jitter and switching streams and a methodology to design the buffers based on the constraints,
3)A prototype implementation of switching of streams. Experimental results show the practical applicability of our technique.

II. RELATED WORK

HTTP streaming has been in vogue utilizing the vast HTTP and web cache infrastructure to serve media [7]. The industry has started moving along those lines recently for adaptive bitrate streaming [8], Adobe HTTP Dynamic Streaming [9], Microsoft Smooth Streaming [10], and Apple HTTP Live Streaming [11]. Stockhammer studied the salient design aspects of DASH [2]. Kuschnig et al. and Akhshabi et al. do a comparative study of the emerging players [1, 12]. To reduce latency of live streaming, Swaminathan et al. propose using HTTP chunked encoding to reduce live latency [13]. MPEG has standardized HTTP Streaming in the DASH specification [14], based on the adaptive HTTP streaming specification of 3rd Generation Partnership Project (3GPP) [15].

Liu et al. explore efficient ways of http streaming using parallel gets using multiple HTTP sessions to fetch media segments simultaneously [16]. Lohmar et al. analyze latency in Live streaming using MPEG DASH [17]. Van Lancker et al. develops and validates a media fragment URI scheme for HTTP streaming [18]. Bica et al. propose a novel caching scheme in multicast networks for higher efficiency in serving a large number of clients [19].

III. BUFFER MODEL

Fig. 1 shows the architecture of a HTTP Streaming client. Initially, the client requests a manifest file which gives, implicitly or explicitly, the URLs of the different fragments

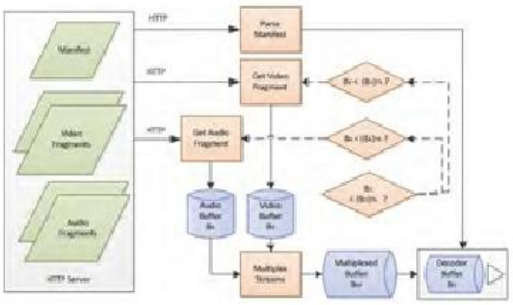

Fig. 1. Architecture of late binding client.

constituting the stream. The data in the HTTP Response is gathered in the audio and video buffers, BA and BV . Multiplexed data is stored in the multiplexed buffer, BM . Data from BM is moved to the decoding buffer, BD throttled at a rate r. Open Source Media Framework (OSMF) [20] is an example of an application layer framework built on top of the Flash Player.

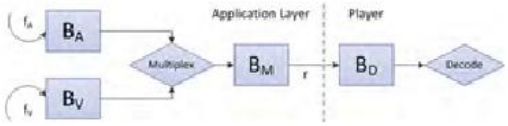

Fig. 2. Buffers in the Client

The relevant buffers are extracted from Fig. 1 and shown in Fig. 2. Multiplexed data is moved from BM in the application layer to BD that is the decoder buffer residing typically in the player at the rate r. A list of definitions related to the buffers follow in Table I.

TABLE I
DEFINITIONS AND TERMINOLOGIES USED IN OUR SCHEME.

Parameters	Definition
B_A	the buffer for storing audio fragments.
B_V	the buffer for storing video fragments.
B_M	the Multiplexed buffer used for storing data after multiplexing data from B_A and B_V .
B_D	the Decoder Buffer.
$(B_X)_{Th}$	the threshold of B_X to request more data from the network.
F_X	the size of a fragment in bits.
r	the rate of movement of data from B_M to B_D in bits/seconds (bps).
f_X	the rate at which B_X is filled up from the network.
b_X	the bitrate.
$(B_X)_\Delta$	the maximum increase in buffer B_X during a fragment arrival time.
$(B_X)_{Max}$	the maximum number of bits the buffer B_X can store when it starts to overflow.
J	The network jitter handling capacity, where network jitter is the variation in network bandwidth or in this context the number of seconds data does not arrive for an active HTTP request for media.
L	The live latency or the difference in time between the event and the playback of the frame that captures that event.
S	The switching latency or the difference in time between when the switch was issued and the playback of the first frame of the switched stream.

IV. CONCLUSION

In this paper, we introduced a formal buffer model for an HTTP streaming client multiplexing audio and video on the fly for an optimal viewing experience. We introduced tunable parameters and demonstrated their effects on the constraints such as buffer sizes, the switching latency, and network jitter absorption capacity. We implemented and validated our findings under various network scenarios.

REFERENCES

[1]Robert Kuschnig, Ingo Kofler, and Hermann Hellwagner. Evaluation of HTTP-based request-response streams for internet video streaming. In IEEE Intl. Conf. on Multimedia Systems, pages 245-256, 2011.
[2]Thomas Stockhammer. Dynamic adaptive streaming over HTTP: stan-dards and design principles. In IEEE International Conference on Multimedia Systems, pages 133-144, 2011.
[3]HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html.
[4]Davy Van Deursen, Wim Van Lancker, and Rick Van de Walle. On media delivery protocols in the web. In IEEE International Conference on Multimedia and Expo (ICME), pages 1028-1033, 2010.
[5] Flash Video Format. http://www.adobe.com/devnet/f4v.html.
[6]MPEG2 Transport Stream. ISO/IEC 13818-1:2000.
[7]Mukaddim Pathan, Rajkumar Buyya, and Athena Vakali. Content delivery networks. Lecture Notes in Electrical Engineering, SpringerLink, 9(I):3-32, 2008.
[8]Luca De Cicco and Saverio Mascolo. An experimental investigation of the akamai adaptive video streaming. In USAB WIMA, pages 521-527, 2010.
[9]Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/httpdynamicstreaming/.
[10]Microsoft Smooth Streaming. http://www.iis.net/download/SmoothStreaming.
[11]Apple HTTP Live Streaming. http://developer.apple.com/resources/http-streaming/.
[12]Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In IEEE International Conference on Multimedia Systems, pages 157-168, 2011.
[13]Viswanathan Swaminathan and Sheng Wei. Low latency live video streaming using HTTP chunked encoding. In 13th International IEEE Workshop on Multimedia Signal Processing (MMSP), pages 1-6, 2011.
[14] MPEG Dynamic Adaptive Streaming over HTTP. ISO/IEC 23009-1:2012.
[15]3rd Generation Partnership Project (3GPP). http://www.3gpp.org.