Rate Adaptation in WebRTC based Peer Assisted Streaming Systems

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Midhul Varma Vuppalapati
(130101047)

under the guidance of

Dr. T Venkatesh

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Rate Adaptation in WebRTC based Peer Assisted Streaming Systems**" is a bonafide work of **Midhul Varma Vuppalapati** (**Roll No. 130101047**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. T Venkatesh**

Assistant/Associate

Professor,

May, 2016 Department of Computer Science & Engineering,

Guwahati. Indian Institute of Technology Guwahati, Assam.

Abstract

HTTP based adaptive streaming technologies like DASH and HLS have become the primary vehicle for delivering video content to large populations of viewers on the Internet. The explosion in volume of Internet video traffic during recent years has lead to an increasing amount of load on streaming servers and CDNs which are responsible for serving content in these systems. The WebRTC stack has opened up the possibility of creating a purely browser based and plugin-free mechanism of utilizing peer to peer upload bandwidth to assist HTTP based streaming systems and hence make them easier to scale.

An important aspect of any adaptive streaming technology is the Bitrate Adaptaion Algorithm (ABR) which plays an important role in determining the Quality of Experience (QoE) perceived by the end user. There has been a significant amount of literature dedicated to studying the design and analysis of these algorithms in the context of HTTP based streaming systems. Addition of WebRTC based peer assistance further complicates the problem of bitrate control. The goal of this project is to identify the key constraints and challenges pertaining to rate adaptation in these kinds of systems and to design better algorithms in order to optimize objective QoE metrics.

Acknowledgements

I would like to extend my gratitude to Hema Kumar Yarnagula, Resarch Scholar, Department of CSE, IIT Guwahati for his constant suggestions, support and motivation during the course of the project. I am also very grateful to my close friend K. Mohan Sai Krishna, BTech 4th year student, Department of CSE, IIT Guwahati, for helping me quickly get up to pace with the HLS streaming technology using infrastructure of the Campus Broadcasting System (CBS) IIT Guwahati as a practical example

Contents

Chapter 1

Introduction

This chapter begins by describing the background concepts and terminology required to bring the reader into context. Following this, the motivation and objective of the project are explained. The chapter concludes by outlining the structure of the remainder of the report.

1.1 Background

1.1.1 HTTP based Streaming

Previously, streaming of multimedia (both video and audio) content over IP networks was done using specialized protocols such as Real-time Transport Protocol (RTP), Real Time Streaming Protocol (RTSP) and Real Time Control Protocol (RTCP) which typically run on top of UDP. Nowadays, using HTTP to stream video content has become the norm and several major video streaming giants like YouTube and Netflix are presently using this method. In a HTTP based streaming system, video content is hosted on HTTP servers and clients are allowed to fetch this content using HTTP requests. The main reasons for the triumph of HTTP over traditional streaming protocols such as RTSP are it's NAT/Firewall friendliness and the ability to scale using Content Delivery Networks

(CDN). Being inherently client-server in nature, HTTP based streaming systems do not suffer from the presence of NAT middle-boxes in the network path. Even in secure firewall protected networks, web browsing is almost always allowed and hence HTTP is not blocked. Another advantage of HTTP based streaming is that it can be consumed by HTML5 players which can stream video content to users directly on their web browsers without the need of any additional plug-ins or software.

1.1.2 Adaptive Bitrate Streaming

Technologies like Dynamic Adaptive Streaming over HTTP (MPEG-DASH), Apple's HTTP Live Streaming (HLS), Adobe's HTTP Dynamic Streaming (HDS) and Microsoft's Smooth Streaming add the concept of Adaptive Bitrate to HTTP based streaming, making it possible to automatically adjust the bitrate of a video stream based on the network conditions. Although the implementation specifics of each of these technologies may be different, the basic idea is the same. The video to be streamed is split into chunks of almost fixed duration and each chunk is encoded at different discrete bitrates. These chunks are served over a normal HTTP server. The client player then fetches these chunks one after another for playback and has the liberty to choose which bitrate version of each chunk it downloads. The algorithm running on the client player which is responsible for selecting the bitrates of each of the chunks is called the Rate Adaptation Algorithm.

Quality of Experience

Simple network QoS (Quality of Service) metrics such as bandwidth, delay and packet loss are not sufficient to evaluate video streaming systems. This is because the quality perceived by the end user depends not only the performance of the network but also on the behavior and logic of the application itself. Quality of Experience (QoE) metrics aim to measure user level satisfaction in video streaming systems. These metrics can be both subjective and objective. Examples of objective QoE metrics are number of playback interruptions,

average video bitrate during playback and number of bitrate switches.

Rate Adaptation Algorithms

The Bitrate Adaptation Algorithm plays a major role in determining the Quality of Experience (QoE) perceived by the viewer. Trying to fetch chunks at a bitrate much higher than the available bandwidth will lead to frequent playback interruptions or re-buffering events. On the contrary trying to fetch chunks at a very low bitrate will lead to overall poor quality. The naive solution of setting bitrate equal to the estimated bandwidth suffices in the case when the bandwidth is steady, but fails when there is significant variation in bandwidth which is very common in realistic scenarios. Rate adaptation algorithms which have been proposed in recent literature make use of indicators such as playback buffer occupancy along with the estimated bandwidth to better optimize QoE.

1.1.3 WebRTC

The WebRTC standard is a collection of protocols and APIs whose aim is to enable better real time communication on the web. WebRTC's DataChannel API enables peer to peer transfer of arbitrary data between web browsers using just javascript code running on them. Setup of such peer to peer sessions and connections is done through an elaborate process.

Session & Connection setup

The set up of the peer to peer sessions & connections requires a means of out of bound communication for exchange of information like IP addresses and port numbers. Out of bound means that how exactly this information is transmitted is not part of the WebRTC protocol specification. Typically a signaling server which is accessible to all the peers is used for this purpose. The Session Description Protocol (SDP) is used to carry information and signals for session initiation. During the setup WebRTC uses sophisticated techniques to try and traverse any NAT/Firewalls in the connection path. If the NAT traversal fails,

it even offers the possibility of using a relay server as fallback using the TURN protocol. Once the setup is successful all of the data flows directly peer to peer.

DataChannel

As mentioned before, WebRTCs DataChannel component provides a javascript API which browsers can use to send and receive arbitrary binary data to and from other browsers in a peer to peer fashion. Under the hood the Stream Control Transmission Protocol (SCTP) which provides tunable reliability is used. WebRTC has it's own SCTP implementation which runs on top of DTLS (Datagram Transport Layer Security) for encryption which ultimately runs on top of UDP. A single persistent SCTP association (connection in SCTP terminology) is created between peers during connection setup and multiple DataChannels can be multiplexed over the same association.

1.2 Motivation & Goal

Video traffic accounted for 70% of all IP traffic in 2015 and is predicted to rise to 82% by 2020 [10]. This exploding volume of Internet video traffic has lead to an increasing amount of load on streaming servers and CDNs which are responsible for serving content in HTTP based streaming systems which have become commonplace nowadays.

WebRTC's DataChannel API opens up the possibility of adding peer assistance to HTTP based streaming systems. In such a system viewers simultaneously watching a stream can make peer to peer DataChannel connections with each other and utilize these to share already fetched video chunks with each other, hence reducing the load on the streaming server. The most attractive feature of these kinds of systems is their ease of deployment. Users of a streaming service will not be required to install any additional plug-ins or software in order to enable peer assistance. The service provider will just have to add some javascript code to the webpage containing the video player and peer assistance will be live. Some start-up firms like StreamRoot and Peer5 have developed and deployed

such technology but make use of proprietary content delivery mechanisms. In fact there is very less literature on the design and analysis of these kinds of systems.

The problem of bitrate adaptation becomes more complex and challenging when peer assistance is added to HTTP based streaming systems. This is because chunks of the video are now being fetched from multiple sources and the number of parameters that need to be controlled increases. The goal of this project is to identify the key constraints and challenges pertaining to rate adaptation in these kinds of systems. With the insights gained during this process, the aim is to design better rate adaptation algorithms to optimize objective QoE metrics while simultaneously reducing load on the streaming server.

1.3 Organization of The Report

A brief survey of related prior literature is given in chapter 2. Following this, in order to give a clear picture of the system model being considered, chapter 3 describes a preliminary implementation which was developed for experimentation. Chapter 4 discusses the problem of rate adaptation in the context of such a system in more detail and highlights various challenges in the design of these algorithms. The report concludes with chapter 5 which briefly describes the future work plan.

Chapter 2

Review of Prior Works

This section provides a brief survey of prior literature in the areas of Peer Assisted streaming and Adaptive Bitrate streaming.

2.1 Peer Assisted Streaming

The idea of a Peer Assisted streaming system is not a new concept, and several such systems have been designed and deployed in the past. The work in [6] lays the theoretical foundations for the design of a peer assisted streaming systems, by establishing performance bounds on the maximum possible server load reduction under different constraints. Although the paper is motivated by peer assisted streaming in IPTV networks, the results derived apply to any peer assisted streaming systems in general. LiveSky [12] is a Hybrid P2P-CDN streaming system, which has a mechanism of switching between P2P and CDN modes based on number of viewers watching the stream. It however supports only single bitrate video and does not consider the case of Adaptive Bitrate Streaming. SmoothCache 2.0 [9] implements peer assistance for HTTP based adaptive bitrate streaming systems by installing a HTTP proxy on every client to act as a cache for fetched video segments. The work in [3] details the design and implementation of a WebRTC based peer assisted streaming system to support MPEG-DASH. The main focus of this work is however on the

issue of overlay construction and no in depth analysis on the issue of rate adaptation is done.

2.2 Adaptive Bitrate Streaming

There is a significant amount of recent literature detailing the design and analysis of rate adaptation algorithms for HTTP based streaming systems. The algorithms proposed can be broadly classified based on the parameters/indicators which they use to control the bitrate of video segments. At the ends of the spectrum are purely throughput based and purely buffer based algorithms. There are also a number of algorithms in the middle, which make use of both of these indicators to different extents. The Adobe Open Source Media Framework player was one of the earliest implementations of MPEG-DASH and made use of a purely throughput based rate adaptation algorithm which estimates current throughput based on historical throughput information (time taken to fetch each of the previous segments). Different algorithms use different functions for this purpose. The work in [1] demonstrates the difficulties in controlling bitrate using throughput estimated by HTTP requests on TCP. The authors of [2] introduce the idea of using playback buffer occupancy as the primary indicator for bitrate control. If the buffer occupancy is low, it can be inferred that the bandwidth is not sufficient to fetch the present bitrate, and hence a switch down is required. Similarly in the case of high buffer occupancy the bitrate can be switched up. Buffer based algorithms like these are typically specified using rate maps which can be represented as graphs of bitrate vs buffer occupancy. Works like [8, 11, 4] make use of both estimated throughput and buffer occupancy to better control video bitrate. The authors of [5] give a comprehensive survey of QoE metrics which can be used to evaluate rate adaptation algorithms.

Chapter 3

Preliminary Implementation

A preliminary version of a WebRTC based peer assisted streaming live streaming system was implemented to create a platform for experimentation. HLS was chosen as the adaptive bitrate technology because of the simplicity of its specification and format in the context of live videos.

3.1 System Architecture

The architecture of the system is shown in figure 3.1. The HTTP Streaming Server publishes a live video stream (Big Buck Bunny movie played in loop to emulate a live stream) as per the HLS specification. Each viewer client has an instance of a HLS enabled video player which fetches chunks from the streaming server and plays them. There is a special P2P controller at every client which takes care of all of the peer assistance functionality. The tracker helps in constructing and maintaining the P2P overlay network and the signaling server assists in WebRTC peer to peer connection setup. The functionality of each of the components is discussed in the following subsections.

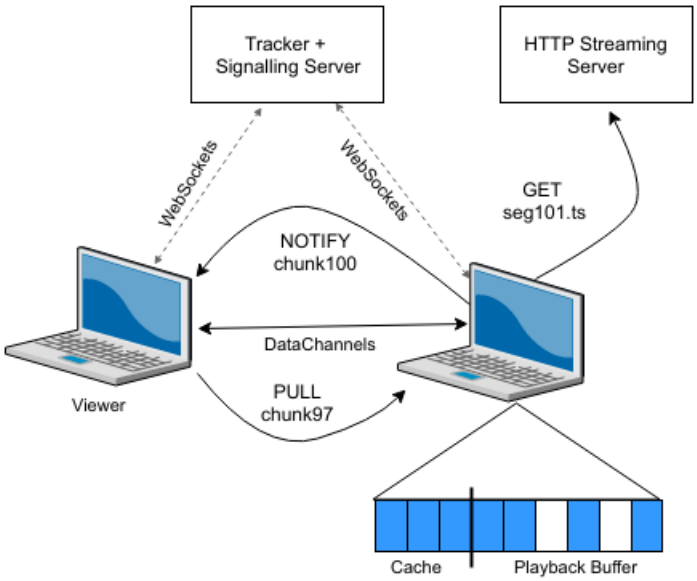

Fig. 3.1 Pictorial illustration of WebRTC-based peer assisted HLS system

3.2 HTTP Live Streaming (HLS)

3.2.1 Server

In HLS based live streaming, the streaming server maintains a sliding window of a fixed number of video segments. New video segments are published periodically at a fixed interval (2-4 secs). Whenever a new segment is published the window maintained by the server slides forward in time by one segment duration. This means the new segment is added to the window and the oldest segment in the window is removed. Information about the state of the sliding window and the URLs of the segments present in the window is stored in a metadata file of M3U8 format. The segments are encoded at different discrete bitrates and each bitrate stream has it's own M3U8 file. There is a single static master playlist file also of M3U8 format which contains the details about the different bitrate streams and URLs of their corresponding M3U8 descriptors.

3.2.2 Player

At the client side Dailymotion's open source HTML5 video player HLS.js was used. The video player first downloads the master playlist file and obtains information about the various bitrate encodings available. It then selects an initial bitrate and downloads the corresponding M3U8 file. After that it starts by fetching the oldest segment in the present window. To keep track of the sliding window and obtain information about newly published segments, it periodically re-fetches the M3U8 file. When the player wants to switch to a different bitrate it downloads the corresponding M3U8 file and follows the same process.

3.3 Peer Assistance

The P2P controller needs to work in unison with the video player. HLS.js by default fetches segments only from the streaming server over HTTP. In the present implementation this is simply replaced with a method which is in the hands of the P2P Controller. The tracker keeps track of the viewers who are presently watching the live stream. Setup of WebRTC peer connections requires an initial signaling phase where network, session and encryption information are exchanged between peers. The signaling server acts as the medium for this initial communication to happen. It is important to note that once the connection is setup video data is transferred directly between peers. Both of these components communicate with clients using bidirectional WebSockets channels and are implemented using node.js and socket.io. Although these are shown as a separate components in the figure, they could very well be hosted on the same physical server on which the HTTP Streaming Server is present.

3.3.1 Bootstrapping & Overlay formation

The peer assistance mechanism is very similar to mesh based peer to peer streaming systems which have proven to be robust and scalable in the past [7]. When a new viewer starts

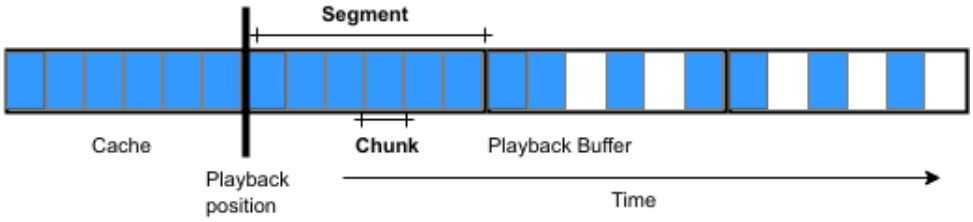

Fig. 3.2 Structure of video buffer of a peer

watching the live stream, his P2P Controller initiates a connection with the tracker and sends a bootstrap request. The tracker acknowledges this and sends back a list of randomly selected peers. The P2P controller then tries to establish a WebRTC peer connection to each of these peers, by making use of the signalling server. The peers hence form a randomly connected unorganized overlay network. Once the peer connection is set up, data channels are established for transmission of video content and meta-data information.

3.3.2 Transfer of Video Data

The unit of sharing in the P2P network is a *chunk* whose size is smaller than the size of an individual *segment*. A *segment* is the unit of video content which is published by the HLS streaming server during every period. These segments may be further divided into smaller units called *chunks* which are the fundamental units of sharing between peers. This means that different parts of the same segment can be fetched from different peers. Whenever a peer receives a chunk, it notifies it's neighbours (the peers to whom it is connected to) that it is now a holder of this chunk using a NOTIFY message over the DataChannel. Hence, every peer has almost up to date information about which segments each of it's neighbours possesses. Now if a peers is in need of a chunk which one of it's neighbours has, it can send a request over the data channel through a PULL message on the DataChannel. Upon receiving a PULL request, peers transfer the chunk which was requested over the DataChannel.

3.3.3 Chunk Scheduling

As shown in figure 3.2, the buffer of video chunks at each peer can be logically divided into two parts. The playback buffer consists of chunks from segments which are yet to be played. Chunks missing in this part of the buffer need to be fetched either from the server or from other peers. Once a segment has been consumed by the player, it's corresponding chunks enter the cache area, where they are cached for a certain additional duration to serve other peers. There is a scheduler running on each of the peers which is responsible for dispatching chunk requests to server/peers. The scheduler is triggered periodically with a fixed period (6 secs). Depending on the number of chunks requested from each of the peers during a period, the scheduler can control the rate that is received from each of the peers.

3.3.4 Preliminary Evaluation

The implemented system was tested on a small scale using a server and upto 30 peers spread across machines connected to a local network in the Department of Computer Science & Engineering (CSE), IIT Guwahati. The HLS streaming server generates a live stream emulated by playing the popular Big Buck Bunny movie in loop. The peers' video players were allowed to consume the stream for approximately 13 minutes. At each peer two statistics were recorded: the number of segments fetched from the server and the number of segments fetched from other peers. Finally the statistics from all the peers are accumulated and the percentage of the total segments which are fetched from peers and from the server are calculated to measure the reduction in server load.

# of Peers	P2P Segments	Server Segments	P2P %	Server %
5	375	115	76.53	23.46
15	1028	274	78.95	21.04
30	2710	429	86.33	13.66

Chapter 4

Rate Adaptation Problem

This chapter discusses the issues and challenges involved in designing rate adaptation algorithms for a system similar to the one described in the previous section and compares them with those in a traditional HTTP based streaming system

4.0.1 Traditional HTTP based streaming

In traditional HTTP based streaming systems, the function of a rate adaptation algorithm is to control a single parameter, the bitrate of the chunks being fetched. These algorithms typically make use of indicators such as estimated throughput and buffer occupancy as inputs to make their control decisions. The ultimate goal of these algorithms is to optimize objective QoE metrics such as average playback bitrate, playback interruptions and number of bitrate switches.

4.0.2 Peer Assisted streaming

In the context of a peer assisted streaming systems, more complexity is added to each of the above mentioned aspects. Bitrate of chunks being fetched is no longer the only parameter that needs to be controlled. It is also important to decide what fraction of the selected bitrate is fetched from the server and each of the peers. For example, if the bandwidth

from a certain peer becomes low, it would be desirable to reduce the fraction of the total bitrate received from it and redistribute to the server or other neighbors. Another example is when the playback buffer is almost on the verge of depleting, it might be safer to fetch a larger fraction of the bitrate from the server rather than the peers which may be unreliable. Hence, the underlying control problem now becomes multidimensional. At the same time the adaptation algorithm now has more indicators which it can use as input: the estimated bandwidth from each of the peers instead of just from the server. Finally, the goal of the adaptation algorithm also needs to be augmented. In addition to optimizing objective QoE metrics the algorithm should also try to minimize load on the server. The key design challenge would then be to strike a balance between these goals. Additionally the algorithm also needs to be able to recover quickly from peer churn (leaving of peers in the network), without causing drastic bitrate fluctuations.

Chapter 5

Conclusion and Future Work

The problem of rate adaptation in WebRTC based peer assisted streaming systems is complex and multifaceted and deserves detailed study and analysis. The aim is to tackle this problem by breaking it down into simpler problems, solving them and incrementally taking into consideration more complexity. During the process, the plan is to experimentally evaluate proposals using the system implemented on real networks, NS3/MiniNet emulations and possibly even PlanetLab deployments.

References

[1] Te-Yuan Huang, Ramesh Johari, and Nick McKeown. Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, FhMN '13, pages 9–14, New York, NY, USA, 2013. ACM.

[2] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198, New York, NY, USA, 2014. ACM.

[3] Marcus Wallstersson Jimmy Zger. Peer Assisted live video streaming in web browsers using WebRTC. 2014.

[4] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1765–1770. IEEE, 2015.

[5] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):401–418, 2016.

[6] Shao Liu, Rui Zhang-Shen, Wenjie Jiang, Jennifer Rexford, and Mung Chiang. Per-

formance bounds for peer-assisted live streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):313–324, June 2008.

[7] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1424–1432, May 2007.

[8] Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. Adaptation algorithm for adaptive streaming over http. In *2012 19th International Packet Video Workshop (PV)*, pages 173–178. IEEE, 2012.

[9] Roberto Roverso, Riccardo Reale, Sameh El-Ansary, and Seif Haridi. Smoothcache 2.0: Cdn-quality adaptive http live streaming on peer-to-peer overlays. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 61–72, New York, NY, USA, 2015. ACM.

[10] Cisco Systems. The Zettabyte Era: Trends and analysis, Cisco Visual Networking Index. Technical report, 2016.

[11] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 109–120. ACM, 2012.

[12] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: Experiences with livesky. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 25–34, New York, NY, USA, 2009. ACM.