Rate Adaptation in WebRTC based Peer Assisted Streaming Systems

Midhul Varma Vuppalapati

Department of Computer Science & Engineering, IIT Guwahati

Abstract

The WebRTC stack has opened up the possibility of creating a purely browser based and plugin-free mechanism of adding peer assistance to HTTP based adaptive video streaming systems which can greatly reduce the load on streaming servers and CDNs. The goal of this project is to tackle the problem of dynamic rate adaptation in these kinds of systems.

Background

- **HTTP based Adaptive Streaming** - Video 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 tries to choose the optimal bitrate for each segment based on network conditions. The algorithm responsible for selecting the bitrates of each of the chunks is called the *Rate Adaptation Algorithm*. It plays a significant role in determining the Quality of Experience (QoE) perceived by the end user. Examples of such technologies are DASH & HLS.
- **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 requires out of band communication but once setup data flows completely peer to peer.

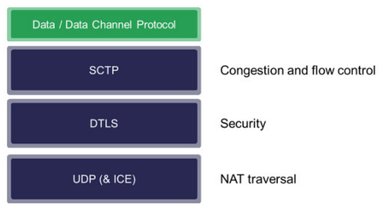

Figure 1: WebRTC's DataChannel

Preliminary Implementation

Preliminary version of a WebRTC based peer assisted live streaming system was implemented to create a platform for experimentation. Components as shown in Figure 2:

1. **HLS Streaming Server** - Serves live stream video segments over HTTP according to HLS spec. Maintains a sliding window of segments which is updated whenever a new segment is available.
2. **Tracker + Signaling Server** - Tracker keeps track of peers viewing live stream and signaling server enables setup of WebRTC P2P connections. Communication with peers is done via WebSockets protocol.
3. **Clients** - Every client has a HTML5 video player (HLS.js) which can fetch segments from the streaming server and play them. Additionally there is a *P2P Controller* which works in unison with the player to enable peer assistance.

P2P Component

Each peer creates WebRTC P2P connections to a set of randomly selected peers with the help of the tracker and signaling server. DataChannels are established over the peer connections for exchange of video and meta-data. Video *segments* are split into smaller *chunks* and these are exchanged in the P2P network as follows:

- Whenever a peer receives a chunk, it notifies it's neighbours that it is now a holder of this chunk using a *NOTIFY* message. Hence, every peer has almost up to date information about which segments each of it's neighbours possesses.
- If a peers is in need of a chunk which one of it's neighbour's has, it can send a *PULL* message.
- Upon receiving a *PULL* message, peers transfer the chunk which was requested over the DataChannel

System Architecture

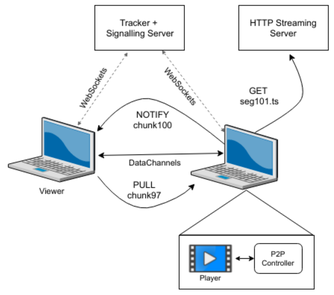

Figure 2: High Level System Architecture

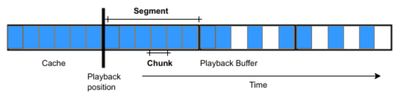

Figure 3: Magnified view of buffer at a client

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 generated 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. Fully connected topology was used. Total number of segments fetched from server and peer to peer were measured. The observations are as follows:

Peers	P2P-Segs	Server-Segs	P2P %
5	375	115	76.53
15	1028	274	78.95
30	2710	429	86.33

Table 1: Preliminary Evaluation

Rate Adaptation

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:

- **Throughput based** - Bandwidth estimated using previous segment fetch times is used to select bitrate.
- **Buffer based** - Level of playback buffer occupancy is used to select bitrate.

Algorithms typically use a combination of both of these indicators to different extents. The ultimate goal of these algorithms is to optimize objective QoE metrics such as average playback bitrate, playback interruptions and number of bitrate switches.

Challenges

Addition of peer assistance to HTTP based streaming systems further complicates the rate adaptation problem:

- In addition to bitrate, it is also important to decide what fraction of the selected bitrate is fetched from the server and each of the peers.
- In addition to optimizing objective QoE metrics the algorithm should also try to minimize load on the server.
- Need to recover quickly from peer churn without causing drastic bitrate fluctuations.

Future Work

The aim is to further study and analyze the problem of rate adaptation in the context of the system which was implemented. For this the system needs to be subject to varying bandwidth conditions. The plan is to perform experiments using the system implemented on real networks, NS3/MiniNet emulations and possibly even PlanetLab deployments.