Fall 2015

MOBILE VIDEO DELIVERY WITH MINIMAL BITRATE CHANGES

Mario Atallah
University of New Hampshire, Durham

MOBILE VIDEO DELIVERY WITH MINIMAL BITRATE CHANGES

BY

MARIO ATALLAH

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science

in

Computer Science

September 2015

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Computer Science by:

Thesis director, Dr. Radim Bartos
Associate Professor of Computer Science

Dr. Philip Hatcher
Professor of Computer Science

Dr. Kevin J. Ma
Affiliate Assistant Professor of Computer Science

On July 13, 2015

Original approval signatures are on file with the University of New Hampshire Graduate School.

Dedication

To my uncle, Michael, who helped me join UNH as an international student, and provided tremendous support along the way. This paper is dedicated to him as a sign of my appreciation and gratitude.

Acknowledgments

A number of people have made this paper possible. In particular, I want to thank Dr. Radim Bartos and Dr. Kevin Ma for their endless support. There were a few challenging parts in this project, however Dr. Bartos and Dr. Ma made it seem not so daunting. On the contrary, working on this thesis was an enjoyable experience.

Table of Contents

List of Figures

ABSTRACT

MOBILE VIDEO DELIVERY WITH MINIMAL BITRATE CHANGES

by

MARIO ATALLAH

University of New Hampshire, September, 2015

Mobile data traffic keeps increasing year after year, as does the need for devices and technologies that support that growth. Video streaming, in particular, has been the major concern for mobile data traffic due to the complexity in handling the bulky nature of the data. HTTP has become the main medium for video streaming over mobile devices due to its existing popularity.

When streaming video to multiple clients on the same network, a bandwidth allocation manager is required to efficiently distribute the available bandwidth among the clients and to ensure a high Quality of Experience (QoE). At the same time, the bandwidth allocation manager should ensure high utilization of the available bandwidth as well as seamlessly adapt to network changes.

In this project we developed a bandwidth allocation management mechanism that reduces the number of bitrate changes while maintaining the high bandwidth utilization and therefore improving the Quality of Experience for the user. To show the results of the bandwidth allocation manager, we developed a simulator that represents a video streamed to many mobile clients from different classes of service. Since the QoE can be negatively affected by the number of bitrate changes, we use Exponential Smoothing and Bitrate Changes Optimization techniques to distribute the bandwidth while ensuring the low number of bitrate changes.

Chapter 1

Introduction and Background

1.1 Nature of the problem

1.1.1 Introduction

Today, everyone is connected to media all over the world through many social portals, especially on mobile devices. Mobile video traffic, in particular, is expected to be a major part of mobile data traffic in the next few years, according to Cisco's forecast that video traffic will claim nearly three-fourths of world's mobile data traffic by 2019 [4]. Supporting this amount of video traffic can be challenging especially in networks with limited but fast changing resources. Media servers have to be able to stream to a large number of devices that request the media simultaneously. Peak hours, newly released content, and content released from popular artists, can all cause a large number of users to request content simultaneously and therefore put strain on the media server. There are three techniques for media delivery — traditional streaming, progressive download, and adaptive streaming [13].

1.1.2 Traditional Streaming

Traditional Streaming is based on a stateful protocol where the media is sent as a series of small packets. The server keeps a record of each client for things such as playback state, streaming position, selected bit rate, etc.. The server keeps track of the state of the client from the time the connection is established until the client is disconnected. Real-Time Streaming Protocol, RTSP, is an example of a traditional streaming protocol. RTSP is a network control protocol that uses Real-Time Transport Protocol, RTP, to transmit the media to the client. RTSP packets can be

transmitted over UDP or TCP. The latter is a better option since firewalls can block UDP packets, but one of its biggest drawbacks is that TCP has an increased latency since its packets are re-sent until received.

1.1.3 Progressive Download

Progressive Download is a video delivery technique very similar to a file download from an HTTP-based server. Using the Progressive Download delivery model, the player client allows the media to be played while the download is still in progress, before the entire media file has been downloaded. Unlike traditional streaming where the client can buffer a fixed and short amount of downloaded media, typically 5 or 10 seconds, Progressive Download will keep the media download running until the entire file has been downloaded [11]. This, in fact, can be a bad technique when a viewer pauses the playback of the stream, and then after the remaining un-played content is downloaded, the viewer decides to abandons the video. Every time this case occurs, bandwidth is being wasted for both the network and the content provider.

1.1.4 Adaptive Streaming

HTTP-based Adaptive Streaming is an advanced concept of media streaming that sends the media in very small progressive downloads [7]. Each media segment is encoded to multiple different formats. The client starts playing the video content with a specific bitrate based on the manifest file, and then adapts according to the available bandwidth. To provide fast start-up and seek times, streaming can start on the lowest bitrate encoding and improve the experience gradually as more bandwidth become available. The protocol's capability of delivering different media rates based on the network conditions also enables seamless and smooth playback experience to the client.

1.2 Architecture

Adaptive Bitrate Streaming consist of a Server, a Distributor (the network), and a Client (the Video Player). Each video will be pre-loaded on the server. The segmenter/packager will divide the video stream into small chunks of streams that can be played individually. The transcoder

will then generate different bitrate encoded streams that are made available to the client. The client will select the appropriate file stream based on the network conditions to deliver a good user experience. This is known as Adaptive Bitrate Streaming.

Over the past several years, the industry has shifted from the traditional streaming over UDP and TCP to the HTTP-based streaming. The following are some of the main reasons for that shift:

- Traditional media protocols often have difficulty getting past firewalls and routers because they are based on UDP packets over unusual port numbers. HTTP on the other hand, uses TCP port 80, that is recognized by any server that supports HTTP-based requests.
- HTTP does not require a specialized streaming server, special proxies, or caches. The media files are served just like any other files on the Web.
- HTTP-based delivery permits leveraging of existing Content Delivery Networks.

1.3 Existing broad solutions

1.3.1 Apple — HTTP Live Streaming (HLS)

HTTP Live Streaming is a HTTP-based video streaming protocol developed by Apple [2]. HLS supports live broadcasts as well as streaming pre-recorded content on demand. HLS supports streams of different bit rates where the client can switch streams to adapt to network changes. The HLS architecture consists of the server, the distributor, and the client. A live stream utilizes a media encoder, stream and file segmenters to produce media segment files, and index files that are served using a standard HTTP server. The media encoder encodes the media that is received from a real-time audio-video device and encapsulates the stream for transport. The encoder outputs the encoded media in an MPEG-2 transport stream. The Media Segmenter will segment the encoded stream into small media files; it is recommended that different encodings of the same segment have the same duration. Also, the Media Segmenter will create the index file, of type .m3u/.m3u8, that is associated with the segmented media files. The index file is updated every time the Media Segmenter completes processing a new media file. The File Segmenter can be used to encapsulate

the encoded media file in an MPEG-2 transport stream. The File Segmenter will then break the file into small file segments of equal duration. The Media Segment Files are .ts media files that are generated by the Media Segmenter. The Media Segment Files are referenced in the index file for processing by the client. The distribution system is typically a web server that delivers the Media Segment Files and the Index Files to the client over HTTP. The client device will query the media segment files and play them as specified in the index files. When encryption is used, the server encrypts the media segments individually. The server will then send the reference to the decryption key in the index file. The client is responsible for retrieving the key from the index file and decrypting the media segments as they are delivered. In a live scenario, the index file will be updated periodically to include the file names and location of the new encoded media segment files.

1.3.2 MPEG — Dynamic Adaptive Streaming over HTTP (DASH)

DASH is similar to HLS as it uses the existing HTTP protocol and works with the existing Internet infrastructure. DASH is being developed by MPEG - Moving Picture Expert Group. Using DASH, the content on the server consists of the Media Presentation Description (MPD) and the media segments [9]. MPD describes the media content on the server. It contains all the information needed by the DASH client to make informed decisions about selecting the appropriate encoded media content for streaming. The information found in the MPD consists of media content availability, media types, resolution, minimum and maximum bandwidth, various encoded alternatives of the media streams and other media characteristics. A DASH client first requests the MPD XML document, parses it, and then selects the set of representations that it will use based on the client capabilities as well as on the information in the MPD document. Each representation is an encoded alternative of the same media components. The MPD defines the characteristics of the media including the bitrate and the resolution. Each representation will contain one or more segments. Each segment is assigned a unique URL, an index, start time, and duration. The DASH specification only specifies the MPD and the media segments formats. DASH does not specify the delivery of the MPD nor the behavior of a DASH client. Another important component of DASH is Common Encryption (CENC). CENC specifies standard encryption and key mapping methods

that can be utilized by one or more digital rights management (DRM) systems to add a protection layer to the content stream.

1.3.3 Adobe — HTTP Dynamic Streaming (HDS)

HTTP Dynamic Streaming is developed by Adobe and is based on MP4 media formats, FLV and F4V, and is part of the Adobe Flash Platform. HDS leverages existing standard HTTP infrastructure. HDS supports quality of service monitoring, adaptive bitrate, and DVR functionality. The architecture of HDS consist of a Preparation Phase, a Distribution phase, an optional Protection phase, and then the Consumption phase [3]. As part of the packaging phase, HDS uses two tools for preparing the media. File Packager is used to prepare the pre-recorded media, and Live Packager is used to prepare the live RTMP streams. The packagers will generate the MP4 fragments, generate an XML-based manifest file, and finally if applicable apply the content protection.

1.3.4 Microsoft — Microsoft Smooth Streaming

Microsoft Smooth Streaming (MSS) is an adaptive streaming technology provided by Microsoft that leverages existing HTTP infrastructure. MSS is an IIS Media Services extension that enables media streaming to clients over HTTP. MSS dynamically monitors the local bandwidth and video rendering performance and optimizes the content playback by switching video quality in real-time [13]. MSS uses Expression Encoder 4 to encode on-demand Smooth Streaming-compatible video. For each bitrate, the encoder creates one MP4 container file for video fragments and one MP4 container file for audio fragments. In addition, the encoder creates an XML-based server manifest file and another XML-based client manifest file. The manifest files enable clients to use heuristics to determine when and how to switch bitrates. The cache-able MP4 fragments are created and stored on the server disk. The multi-second fragments are then delivered to the client upon request. MSS supports both, live streaming and on-demand streaming.

1.4 Specific problem: Bandwidth allocation

1.4.1 Introduction to the Problem

Bandwidth allocation is a critical part of any video delivery system. If the client has not been allocated enough bandwidth, or if the bitrate of the media streaming fluctuates very often, this will result in a bad quality of experience for the client. In a simple scenario, many clients from different classes of service could be streaming content from the same server at the same time through a potentially overloaded streaming server. Bandwidth allocation management algorithms are designed to efficiently and fairly stream content to mobile clients, especially from an overloaded server. To deliver a Class-Weighted Fairness bandwidth allocation, clients in the higher class of service should always receive higher bitrate encoding and better service than clients in the lower classes of service. This can be enforced by the bandwidth allocation manager. A good bandwidth allocation manager is needed to provide the following:

- High bandwidth utilization to minimize the wasted bandwidth and therefore provide optimal experience to all clients in the network.
- Low number of bitrate changes to minimize the noticeable video quality changes or interruptions to the end user.
- Fast adaptation to network changes in the cases where the amount of available bandwidth decreases or increases significantly.
- Enforcing fairness among the different classes of service where clients in higher classes of service receive better service than clients in lower classes of service.

1.4.2 Existing Network-Based Solutions

Different network-based bandwidth allocation approaches have been proposed to enhance the client user experience. All those algorithms can be integrated with the existing rate adaptation protocols to provide optimal Quality of Experience. Ma et al. [5] have developed a congestion-aware rate adaptation scheme using intelligent segment selection to provide better bandwidth allocation

management. They used a breadth-first bandwidth allocation scheme that will iterate through each class of service from high to low and keep increasing the bitrate for each class of service until maximum bandwidth has reached. However, their algorithm assigns equal bitrates to all clients in the same class of service and therefore wastes bandwidth when the remaining bandwidth is not sufficient for all clients in a class of service.

Another algorithm Ma et al. [6] have developed is a client-based CoS enforcement technique as part of an HTTP adaptive bitrate protocol using a segment-based rate adaptation algorithm. The latter includes CoS-weighted abort, random backoff, and dynamic rate adaptation that has shown to reduce network congestion and improve network utilization.

Sungsu el al. [12] have developed a bandwidth allocation and request routing algorithm using cognitive approaches to understand the current state of the network and the video streaming service. The cognitive system can reason about which actions should be taken using a low-level monitoring scheme that can determine the current state of the network (i.e., bandwidth utilization, number of users, etc.) and take appropriate actions based on the observed state.

Staelens et al. [10] conducted a subjective video quality assessment and showed that frequent bitrate fluctuations and video stallings have significant influence on the end user QoE. Their experiments were done in an ecological environment for long duration video using iPads. They also assessed the end-user's quality rating for different video content and found quality fluctuation is more visible in action and sports when compared to music and drama.

1.4.3 Unsolved Problem - Bitrate Changes

Reducing bitrate change techniques have not been incorporated in the previously mentioned bandwidth allocation algorithms. By tracking the number of bitrate changes for each client, an algorithm could use that data to make more informed decisions about each client. The bandwidth allocation algorithm could attempt to reduce the number of bitrate changes leading to a better quality of service for the client. When network conditions fluctuate often (too many leave or join an already congested network), it might lead to consecutive changes in the bitrate selected by the client. This could in turn affect the individual client's quality of experience. The fewer bitrate changes the

client encounters in a period of time, the smoother and the better the video streaming experience is, especially when the difference between the available bitrate encoding is big.

Chapter 2

Proposed Solution

2.1 Solution Design

In this project, we developed a bandwidth allocation algorithm that reduces the number of bitrate changes while maintaining the high utilization to improve the overall Quality of Experience. The algorithm is a network-based algorithm that has control over the bandwidth flow of all clients in the network. This approach enables the bandwidth allocation manager to make better decisions and to distribute the bandwidth more efficiently among the clients. The algorithm is designed with a set of features that can be enabled or disabled for each experiment. Those features are:

- Rule 1 - (CWF): Ensure that, at all times, clients from a lower class of service get less bitrate encoding than clients from higher class of service. When this feature is disabled, then clients from lower class of service get less than or equal bitrate encoding than clients from higher class of service. At no time a client from a lower class of service can get higher bitrate encoding than a client from a higher class of service regardless whether the feature is enabled or disabled.

- Rule 2 - (ESV): When new clients join and there's not enough bandwidth, or when some clients leave and there's extra bandwidth to distribute, change the bitrate of the client with the lowest exponential smoothing value of bitrate changes.

- Rule 3 - (BCO): Reduce the number of bitrate changes per client while compromising the class-weighted fairness rule and/or the overall bandwidth utilization.

2.2 The Algorithm

The algorithm attempts to optimize the user experience while respecting the level of priority of each class of service. To achieve that, the algorithm uses sophisticated decision making to determine the best client to get more or less bandwidth or simply to get a bitrate change. Unlike existing bandwidth allocation algorithms, for example Ma's algorithm [5], this algorithm uses the number of bitrate changes as a factor in the decision making when allocating bandwidth. The algorithm keeps track of the number of bitrate changes and uses Exponential Smoothing Value and Bitrate Changes Optimization techniques to determine the best client to perform a bitrate change on based on the previous client's history of bitrate changes. The number of bitrate changes for each client are based on the duration when the client is active, and is updated any time the bitrate changes for the client.

The algorithm assumes a pre-set number of classes of service. It iterates through all clients in each class of service and increments their bitrate until all the available bandwidth has been assigned, or until all clients have reached the maximum bitrate for its class of service. To maintain the Class-Weighted Fairness rule, that ensures that clients from higher class of service will always get more bandwidth than clients in lower classes of service, the algorithm will iterate through each class of service, starting with the highest priority one. It will assign the first bitrate level to all clients in the first class of service, and then start over. In the second round it will increment the bitrate of the clients in the first class of service to the second level, and assign the first level of bitrate to the clients in the second class of service. The algorithm will proceed in this pattern until any of the following conditions is met:

- There is no more bandwidth to distribute.
- All clients from a certain class of service have reached the maximum amount of available bandwidth — meaning incrementing the bitrate of clients in a lower class of service will break the rule of Class-Weighted Fairness described earlier.

Figure 2-1 shows a simple video streaming setup with 3 classes of service, 9 clients, and 19 units of available bandwidth. Here we assume that the bandwidth units are all identical. We also assume

there are 3 different levels of bitrate. The columns High, Medium, and Low represent the different classes of service, High being the class of service that has the highest priority and Low being the class of service with the lowest priority. Units 0, 1, and 2, are first distributed to the clients of the High class of service. Then units 3, 4, and 5, are distributed to the clients of the High class of service again, ensuring that clients in the Medium class of service won't have equal bandwidth compared to clients from the High class of service. At this point, units 6, 7, 8, and 9, are distributed to the clients in the Medium class of service. This pattern goes on until all available units are distributed, or until all clients from a certain class of service reaches their maximum allowed bitrate. Figure 2-1 is an example of Rule 1 (Class-Weighted Fairness) only, and doesn't take into account Rule 2 or Rule 3.

Available Bandwidth Units	Priority			Bitrate
	High	Medium	Low	
0 - 2				Bitrate 3
				Bitrate 2
	0 1 2			Bitrate 1
3 - 9				Bitrate 3
	3 4 5			Bitrate 2
		6 7 8 9		Bitrate 1
10-18	10 11 12			Bitrate 3
		13 14 15 16		Bitrate 2
			17 18	Bitrate 1

Figure 2-1: Example of Bandwidth Assignment

The algorithm is designed to support enabling and disabling any combination of Rule 1 (CWF), Rule 2 (ESV), and Rule 3 (BCO). Regardless of the enabled features, every time the algorithm runs, it updates the number of bitrate changes for each client. If Rule 2 is enabled, then the exponential smoothing value of the past bitrate changes is calculated after every run. The exponential smoothing

value is used to sort the clients in each class of service, so that the clients with the lowest exponential smoothing value will get a bitrate change before clients with higher exponential smoothing value. In certain conditions, there would be enough bandwidth remaining to cover some, but not all, clients in a certain class of service; that is when tracking the number of bitrate changes and calculating the exponential smoothing value is beneficial. Using ESV for the number of bitrate changes doesn't necessarily minimize the overall number of bitrate changes of the maximum bitrate change since it works on a per class of service basis. In other words, ESV for the number of bitrate changes helps evenly distribute the number of bitrate changes among clients in the same class of serivce. One scenario when we won't see an evenly distributed number of bitrate changes for all clients in the same class of service is when a new client joins the network later in the process and therefore doesn't have a chance to reach the same number of bitrate changes as others in the same class of service. For example, if an existing active client (c1) already had four bitrate changes when a new client (c2) joins the network, and by the time c2 reached two bitrate changes many other clients left the network and therefore no further bitrate changes occurred, c1 will end up with four bitrate changes while c2 had only two, which leads to unevenly distributed number of bitrate changes within the same class of service. If Rule 3 is enabled, we utilize bitrate changes optimization techniques to reduce the number of bitrate changes even more. The bitrate changes optimization technique is run after all the available bandwidth has been assigned in each run. Then, before the bitrate encoding for each client is set and finalized, the algorithm will find all the clients that had a bitrate change in the previous round. For those clients, the algorithm will pick the ones that had an increase in bitrate and revert their bitrate back to the lower level. Even though this might prevent certain clients from receiving slightly better video quality at some rounds, the algorithm aims to reduce the total number of bitrate changes and therefore to improve the overall quality of experience. For those clients that had a decrease in bitrate level in the current round, the algorithm first finds a client that had an increase in bitrate level in the current round and reverts their bitrate level, and uses that extra bandwidth to increase the bitrate for the original client that had a recent decrease in bitrate level. By using the BCO technique, the algorithm, by design, violates the class-weighted fairness rule for some clients to allow other clients in a higher class of service to have equal bitrate

to clients in the lower class of service. In this case, the algorithm will be saving two bitrate changes at the same time. Obviously this technique will lower the overall utilization, however it has been shown by Robinson et al. to improve the quality of experience by reducing the overall number of bitrate changes [8]. The overall utilization can also be configured to not go beyond a pre-set threshold. This means that if saving a bitrate change for a certain client will result in a utilization below that threshold, then the bitrate optimization will stop. This sets a limit on the number of clients not being serviced to the benefit of reducing the total number of bitrate changes.

2.2.1 Algorithm in Pseudo-Code

Algorithm 1 Bandwidth Allocation Algorithm

```
for all classes of service do
    if ESV is enabled then
        sort clients in this class of service by their exponential smoothing value
    end if
    for all clients in this class of service do
        if CWF is enabled then
            if bitrate index of current client = number of classes of service − 1 − index of current
cos then
                UpdateBitrateChanges() then exit algorithm
            end if
        else
            if bitrate index of current client = bitrate index of last bitrate level then
                UpdateBitrateChanges() then exit algorithm
            end if
        end if
        if next bitrate − current bitrate + consumed bandwidth > total bandwidth then
            UpdateBitrateChanges() then exit algorithm
        else
            Increment consumed bandwidth
            Increment the bitrate of the current client
        end if
    end for
end for
UpdateBitrateChanges()
```

Algorithm 2 Update Bitrate Changes

```
if BCO is enabled then
    OptimizeBitrateChanges()
end if
for all Classes of service do
    for all clients in this class of service do
        Update number of bitrate changes for current client
        Update average bitrate for current client
    end for
    Set total allocated bandwidth for current class of service
    Set total bitrate used for current class of service
end for
```

Algorithm 3 Optimize Bitrate Changes

```
for all classes of service do
    for all clients in this class of service that had a bitrate change in the current round do
        c1 = current client
        temp_br = absolute value (curr_bitrate[c1] − prev_bitrate[c1])
        if curr_bitrate[c1] − prev_bitrate[c1] > 0 then
            final_bitrate[c1] = prev_bitrate[c1]
        else
            if consumedBandwidth + temp_br ≤ total bandwidth then
                final_bitrate[c1] = prev_bitrate[c1]
            else
                for all clients in higher classes of service than the one c1 belongs to do
                    c2 = current client
                    if curr_bitrate[c2] − prev_bitrate[c2] = prev_bitrate[c1] − curr_bitrate[c1] then
                        final_bitrate[c2] = prev_bitrate[c2]
                        final_bitrate[c1] = prev_bitrate[c1]
                        break
                    end if
                end for
            end if
        end if
    end for
end for
```

2.3 Benefits

The algorithm runs continuously and at every run it redistributes the bandwidth to all clients, as much as possible, taking into account the number of bitrate changes of each client. In this way the

algorithm will quickly adapt to changes in the network. A change in the network could be in the form of more or less overall available bandwidth due to clients leaving or joining the network. At every algorithm cycle, the new clients are detected and assigned any available bandwidth based on the class of the service they belong to as well as on the overall bandwidth availability. When is enough available bandwidth to assign maximum bitrate to all clients, then the algorithm doesn't have any benefits, except for forcing CWF when it is enabled and assigning equal bitrates to all clients when CWF is not enabled. However, especially when the network is over-congested (i.e. not enough bandwidth to service all clients), there are several benefits to using CWF, ESV, and BCO. Here are the immediate benefits of the proposed algorithm.

2.3.1 Minimizing of Bitrate Changes

The exponential smoothed value for the number of bitrate changes for each client is calculated at every round of the algorithm and is used to determine the client that is most suitable for a bitrate change when a bitrate change is needed. The more recent bitrate changes a client had in the past, the higher its exponential smoothing value, and therefore the less likely it is chosen by the algorithm to alter its bitrate in cases where there is not enough bandwidth to give equally to all clients in the same class of service. In addition, the bitrate changes optimization technique will revisit the bitrate assignment and attempt to reduce the number of bitrate changes while ensuring that every client from a higher class of service gets equal or higher bitrate than clients in the lower class of service, but not less. In this case, the algorithm reduces the number of bitrate changes and therefore minimizes the amount of noticeable disruptions of the continuous playback.

2.3.2 High Utilization

The algorithm maximizes the bandwidth utilization by iterating through all the clients and incrementing the bitrate encoding level of each until there is no more bandwidth or until a certain client could not get any more bandwidth based on the class of service they are in. In the latter case, all clients would have received the maximum amount of bandwidth they can have while still preserving the rule of class-weighted fairness, and therefore, the remaining unallocated bandwidth could not

be used. However, when bitrate changes optimization technique is being used, the algorithm will reduce the overall number of bitrate changes at the expense of reducing the overall bandwidth utilization. In low bandwidth situations, reduced bandwidth usage provides better elasticity for new clients entering the network.

2.3.3 Adapting to bandwidth changes

The algorithm runs periodically, and at every round, attempts to efficiently and quickly adapt to bandwidth changes caused by an increase or decrease in the total number of clients streaming from the same server on the network. In the next run of the algorithm, and assuming there's enough bandwidth to give all clients at least the minimum bitrate level, the algorithm redistributes the available bandwidth among the new set of clients while attempting to prevent new clients from waiting for bandwidth and at the same time allowing existing clients to continue the video streaming without any noticeable playback interruption (due to the increase of the number of clients in the network). By using the bitrate changes optimization technique, the algorithm will prevent the re-evaluation from constantly increasing then decreasing the bitrate encoding for clients in a given class of service.

2.3.4 Class-weighted fairness

The algorithm will ensure that any client in a higher class of service gets a higher bitrate encoding than any client in the lower class of service. This rule is enforced at all times and during all conditions of the network, except when the bitrate change optimization technique is being used. In that case, the algorithm will ensure that any client in a higher class of service gets a higher, or equal, bitrate encoding then any client in the lower class of service. This exception applies only for the clients that were affected by the bitrate change optimization technique and all other clients will still respect their class of service priority. Class-weighted fairness is a common requirement in real world deployment of video streaming, where a client who paid more for their service should always get a better service than clients who paid less for their service. However, other video streaming technologies might prefer to assign equal bitrate to clients in different classes of service when the

network is highly congested, as opposed to letting clients wait for bandwidth.

Chapter 3

Experiment

3.1 Setup

Our experiments utilize a bandwidth allocation management simulator developed in Java. The purpose of the experiments is to show the number of bitrate changes, the average bitrate per class of service, and the utilization of each class of service. These measures are shown using different combinations of the features Class-Weighted Fairness, Exponential Smoothing Value of number of previous bitrate changes, and Bitrate Changes Optimization technique.

Our simulations assume four classes of service. Each class of service contains different numbers of clients. We experimented with the following data: 20 clients in the first class of service (highest priority), 25 clients in the second class of service, 30 clients in the third class of service, and 35 clients in the fourth class of service. The total video duration used in this experiment is 25 timesteps, whereas the duration of the playback for each client is 18 timesteps. Each timestep represents an instance of the periodic execution of the algorithm as it happens in a real-world scenario. For example, the algorithm could be designed to run every 10 seconds, representing a single step of the 25 step execution used in the experiments. In addition, to simulate the process of clients joining and leaving the network, a random generator is used to place each client in a different 18 step window in the 25 steps the algorithm is run. The simulator doesn't support a client joining the network, leaving it, and then joining it back again. All clients will enter the network, then leave it only once. We chose 18 to be the playback duration and 25 to be the total video duration for this experiment. The algorithm will run with different amounts of total available bandwidth. The data used for the available bandwidth for each experiment is split into three categories. The first

category of experiments was done using 10,000 kbps to 70,000 kbps with 10 kbps increment. The second category was done using 80,000 kbps to 140,000 kbps with 10 kbps increment. And the third category was done using 150,000 kbps to 450,000 kbps with 50,000 kbps increment. The reason for experimenting with the three categories (10k-70k, 80k-140k, and 150k-450k) is to show the results of our algorithm in a highly-congested, moderately-congested, and lightly-congested scenarios, respectively. The available bitrate levels used are 0, 200, 400, 600, 1200, and 3500 kbps, based on Apple's Recommended Encoding Settings [1]. Each version of the algorithm is run 100 times and an average is calculated. The constant, α, that is used to calculate the exponential smoothing value for each client is set to 0.1.

3.2 Measures

The algorithm has a set of features that can be turned on or off. In addition to Ma's algorithm, we compare results from 5 different versions of the algorithm. The versions of the algorithm consist of the following rules:

- Ma's Algorithm
- Algorithm Version 1: Rule 2 (ESV Only)
- Algorithm Version 2: Rule 2 + Rule 3 (ESV + BCO)
- Algorithm Version 3: Rule 1 (CWF Only)
- Algorithm Version 4: Rule 1 + Rule 2 (CWF + ESV)
- Algorithm Version 5: Rule 1 + Rule 2 + Rule 3 (CWF + ESV + BCO)

We used five different evaluation methods to measure the quality of experience provided to the clients.

- Average Bitrate Changes

  We track the number of bitrate changes per client and calculate the average per available bandwidth. We show and compare the *average bitrate changes* for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm.

We measure the maximum number of bitrate changes among all clients and compare the results with the proposed algorithm versions 1 to 5 as well as with Ma's algorithm.

We calculate the average difference and average ESV of bitrate changes and compare the results between algorithm versions 1 to 5 and Ma's algorithm.

- Average Bandwidth Utilization

  We calculate the overall bandwidth utilization per available bandwidth for the proposed algorithm versions 1 to 5 as well as Ma's algorithm and show how it is distributed among the classes of service.

- Difference in Average Allocated Bandwidth

  We calculate the *Average Allocated Bandwidth per Class of Service* as an indication of how bandwidth is distributed over all clients in every class of service. We compare Ma's algorithm with the proposed algorithm versions 2 and 5. We don't compare with algorithm version 1 since the latter is ESV only and ESV by itself shows minimal impact on the average allocated bandwidth.

  We calculate the *Difference in Average Allocated Bandwidth among all Classes of Service* as an indication of the average difference in allocated bandwidth between the different classes of service. The function used to calculate the average difference is $C(x) = \frac{\sum_{i=1}^{n}(b(i-1)-b(i))}{n-1}$, where x represents the total available bandwidth in kbps, i represents the index of the current class of service, n represents the total number of classes of service, and $b(i)$ represents the average bitrate of class of service index i in kbps. We compare Ma's algorithm with the proposed algorithm versions 1 to 5.

- Tradeoff between Bandwidth Utilization and Bitrate Changes

  We calculate the trade-off between the average bandwidth utilization and the average bitrate changes. We aggregate the results from all the available bandwidths to produce the mean and standard deviation for the average bandwidth utilization and average bitrate changes.

- CWF Violations

  We calculate the *average number of clients with CWF violations* for algorithm versions 1 to 5 as well as for Ma's algorithm. A CWF violation is when the CWF rule — a client from a higher class of service must have higher bitrate than a client from a lower class of service — is violated. In other words, a CWF violation is when a client from a higher class of service have equal bitrate to a client from a lower class of service.

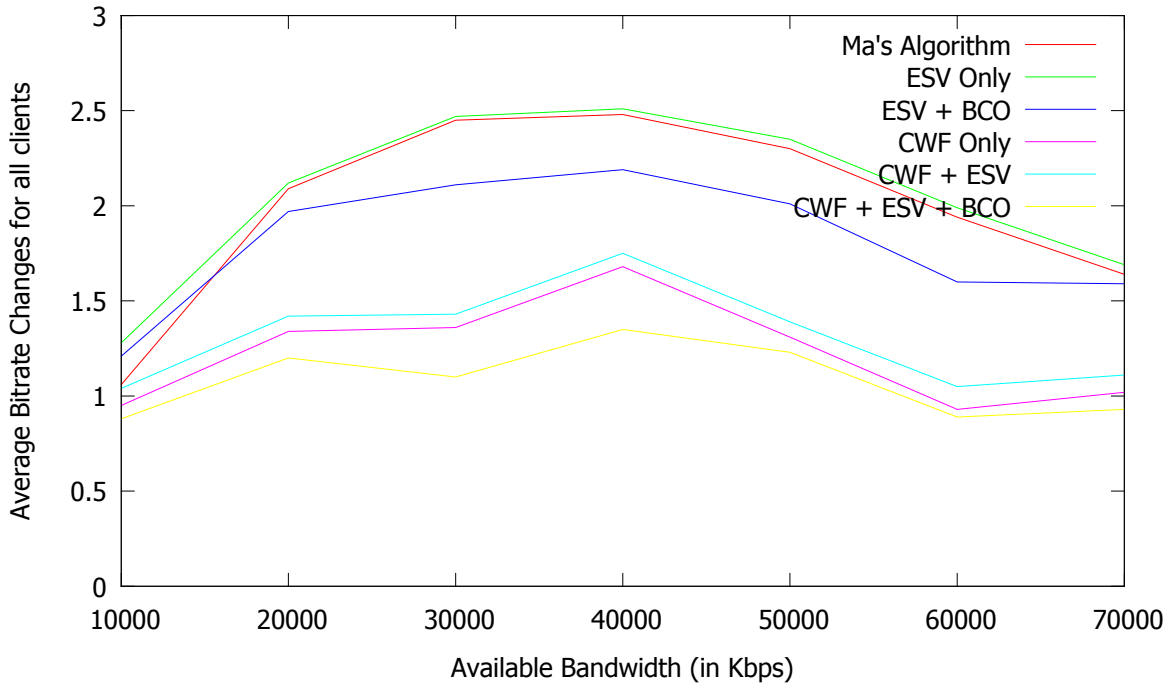

Figure 3-1: Average Bitrate Changes per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

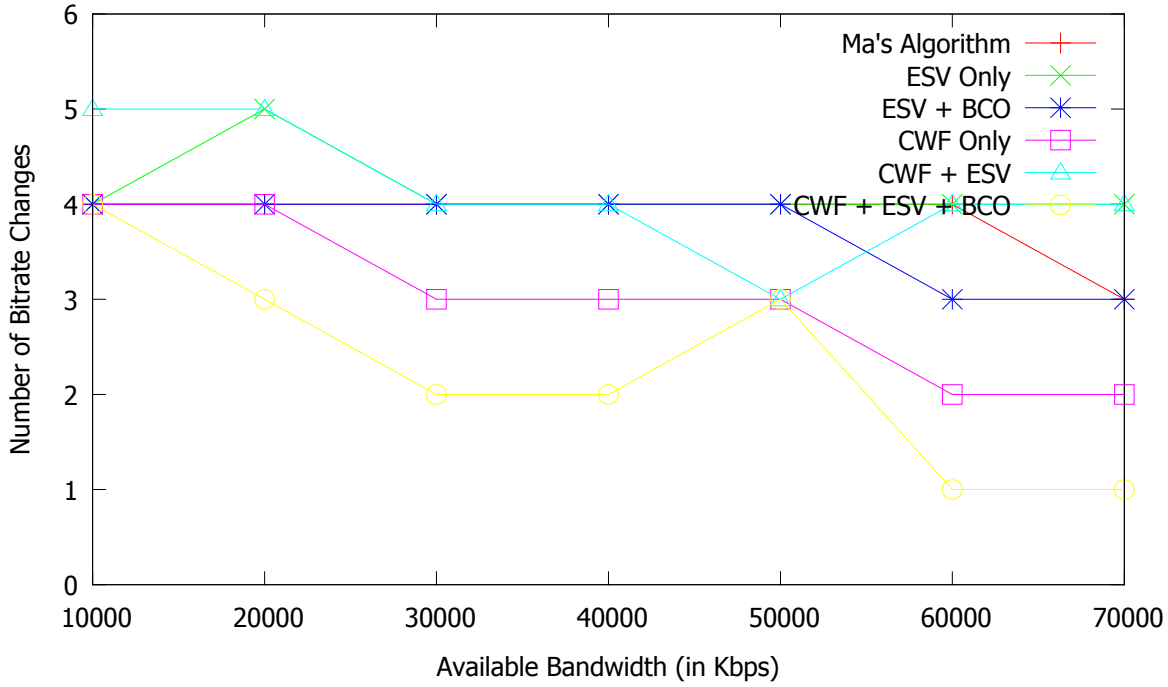

Figure 3-2: Maximum Number of Bitrate Changes per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

3.3 Results

3.3.1 10k to 70k graphs

Average Bitrate Changes

Figure 3-1 shows the average bitrate changes, for each available bandwidth of 10,000 kbps to 70,000 kbps with 10,000 kbps increment, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. The graph shows that the number of bitrate changes for Ma's algorithm is higher than all of the proposed algorithm versions, except for the almost identical results compared to proposed

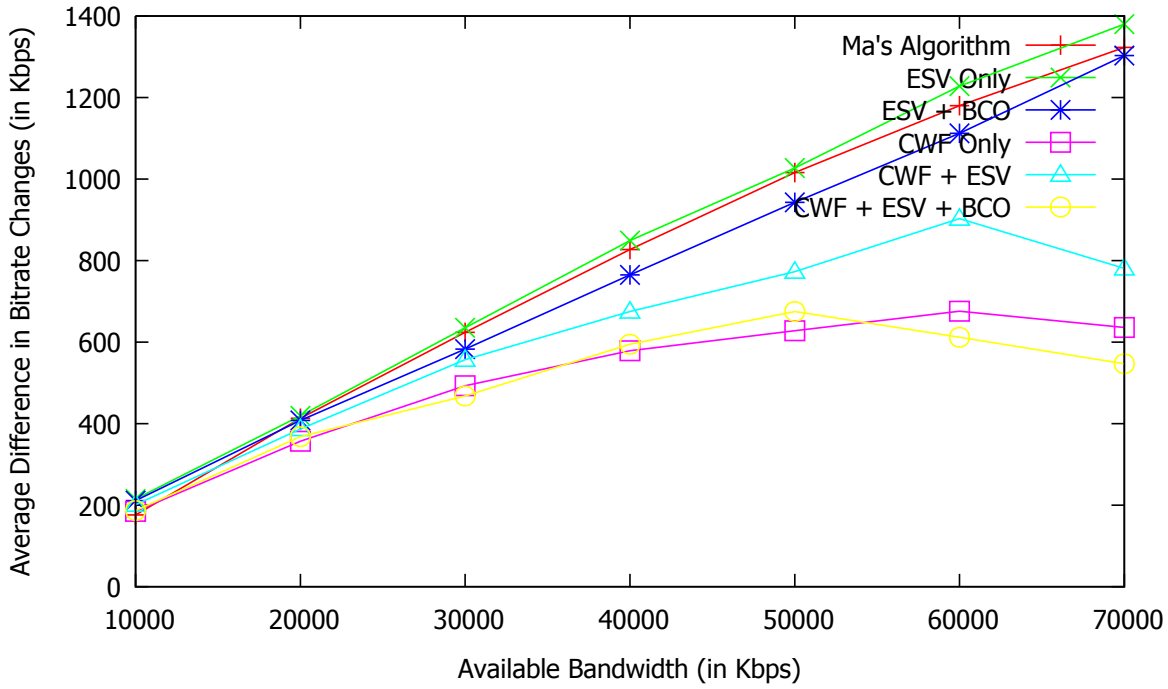

Figure 3-3: Average Difference in Bitrate Changes per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

algorithm version 1. This is because CWF is used in proposed algorithm versions 3, 4, and 5 which results in more clients waiting for bandwidth when the amount of available bandwidth is low. Naturally, when there are fewer clients in the network, the overall number of bitrate changes will be smaller than when there are more clients in the network. Proposed algorithm version 5 shows the smallest average number of bitrate changes because it uses BCO, which adds another layer of bitrate change optimization to the algorithm than just ESV. Algorithm version 3 shows a close average number of bitrate changes when compared to algorithm version 4 since ESV doesn't impact the overall number of bitrate changes, instead ESV works on individual clients to try to

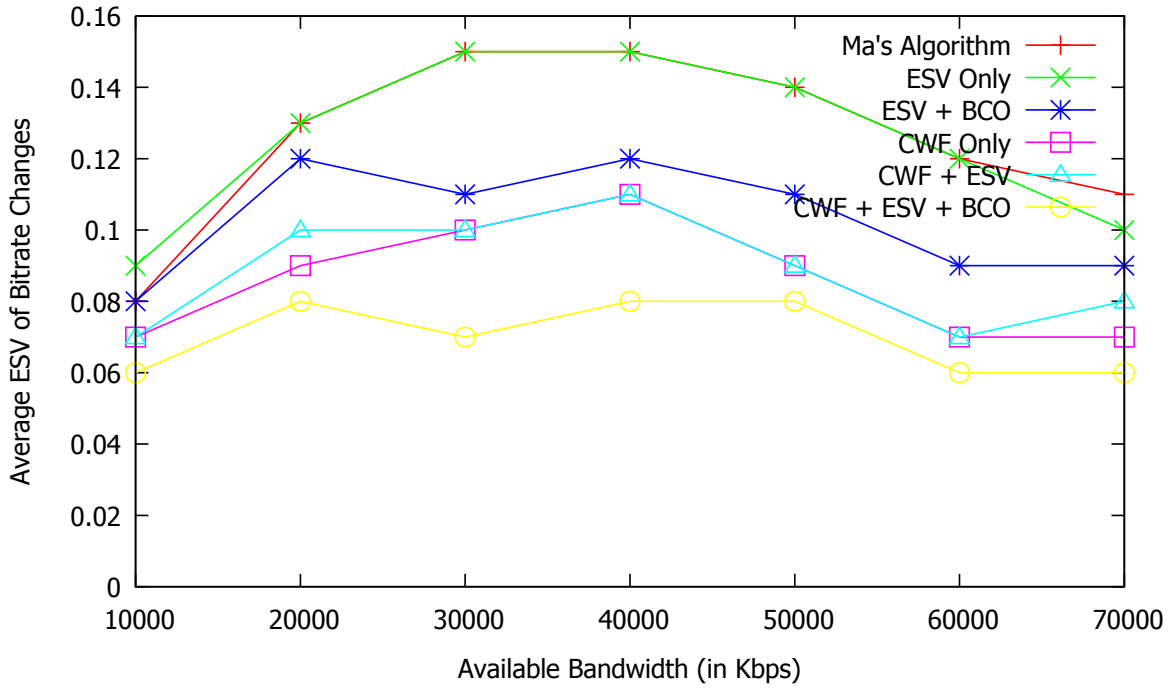

Figure 3-4: Average ESV of Bitrate Changes per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

distribute the bitrate changes as equally as possible among all clients in each class of service. Figure 3-2 shows the maximum number of bitrate changes among all clients for each available bandwidth. The reason that proposed algorithm version 1 shows the highest maximum number of bitrate changes among all available bandwidth is because, similar to Ma's algorithm, that algorithm doesn't use CWF or BCO, and therefore more clients are in the network than would be when CWF is used and no attempt is made to reduce the number of bitrate changes, other than ESV which reduces the difference of bitrate changes between clients of each class of service. We see that algorithm version 5 shows the lowest maximum number of bitrate changes. What that shows is

that algorithm version 5, using exponential smoothing value for the number of bitrate changes and bitrate changes optimization techniques, reduce the number of bitrate changes for all clients and provide a smooth and fair streaming experience more than all other versions of the algorithm. In Figure 3-3 and Figure 3-4 we show the average range of bitrate switches and the average frequency of bitrate changes, respectively. Naturally, the bigger the range of changes in bitrate levels, the more noticeable it is to the end user. In Figure 3-3, we see that proposed algorithm version 5 has the smallest range compared to the CWF algorithm versions 3 and 4, except at 50,000 kbps. Similarly, proposed algorithm version 2 has the smallest range compared to the non-CWF algorithm version 1 and Ma's algorithm. This is mainly due to the use of BCO, that reduces the number of bitrate changes and therefore has a direct effect on the average range of bitrate changes. Also we notice that non-CWF algorithm versions have mostly higher average difference in bitrate changes compared to CWF algorithm versions; that is because non-CWF algorithm versions can stream to more clients in a highly-congested scenario than CWF algorithm versions. Figure 3-4 show the frequency of the bitrate changes through calculating the average exponential smoothing value of the bitrate changes. The lower the exponential smoothing value, the less frequent are the bitrate change. We see that proposed algorithm version 5 has the least frequent bitrate changes of all other algorithm versions, which in turn has a positive impact on the quality of experience.

Average Bandwidth Utilization

Figure 3-5 shows the average bandwidth utilization, for each available bandwidth of 10,000 kbps to 70,000 kbps with 10,000 kbps increment, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. Ma's algorithm appears to have the highest average bandwidth utilization compared to proposed algorithm versions 1 to 5, with the exception of proposed algorithm version 1, where the results are almost identical. This is because Ma's algorithm doesn't consider class-weighted fairness nor does it attempt to reduce the overall number of bitrate changes. Proposed algorithm version 5 shows the lowest bandwidth utilization mainly because it takes into account both rules, class-weighted fairness and bitrate changes optimization. Proposed algorithm versions 3 and 4 show almost identical results as well, which tells us that, as expected, ESV does not have

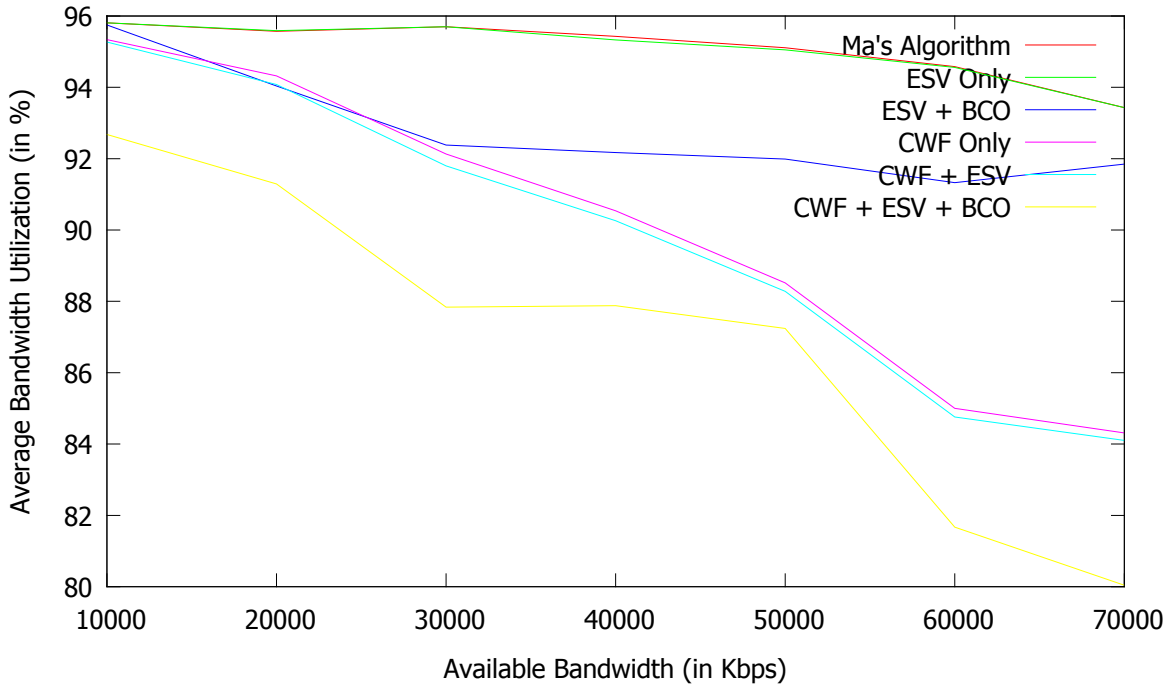

Figure 3-5: Average Bandwidth Utilization per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

an effect on overall bandwidth utilization. Proposed algorithm version 2 has significantly higher utilization than proposed algorithm version 5 which tells us that CWF has a negative impact on the overall bandwidth utilization. The reason that the average bandwidth utilization decreases with more available bandwidth is because when all clients have been assigned the maximum bitrate encoding level, then the remaining available bandwidth will not be used, and therefore will decrease the overall utilization.

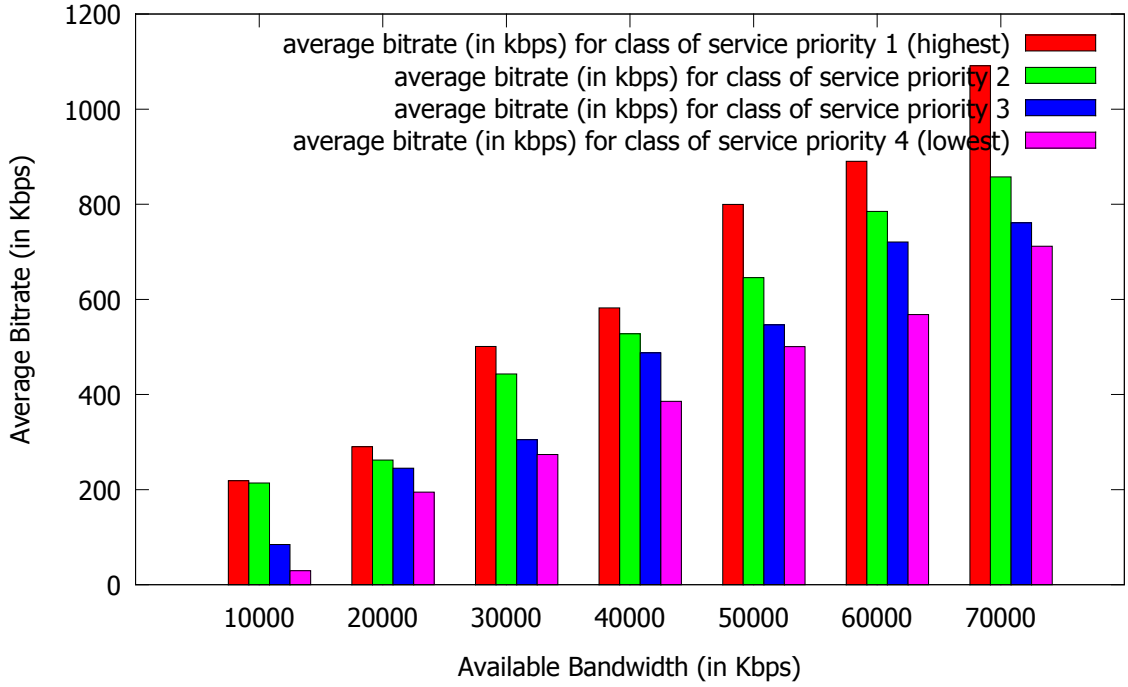

Figure 3-6: Average Allocated Bandwidth Per Class of Service per Available Bandwidth (10k to 70k with 10k increments)

Ma's Algorithm

Difference in Average Allocated Bandwidth

Figure 3-6 shows the average allocated bandwidth per class of service for each available bandwidth of 10,000 kbps to 70,000 kbps with 10,000 kbps increment, for Ma's algorithm. We see that all classes of service have close average allocated bandwidth, especially for the available bandwidth tests 10,000 kbps until 40,000 kbps. That is because Ma's algorithm doesn't apply the class weighted fairness rule when distributing bandwidth as opposed to proposed algorithm version 5, as seen in Figure 3-8. Figure 3-8 shows the distribution of bandwidth between the classes of service for proposed algorithm version 5. We see that since the class-weighted fairness rule is applied, the gap between the classes of service is bigger than Ma's algorithm in Figure 3-6. However, in Figure 3-7, for algorithm version 2, we see similar results to Ma's algorithm since the class-weighted fairness is not applied. Figure 3-9 shows the difference in average allocated bandwidth between all the

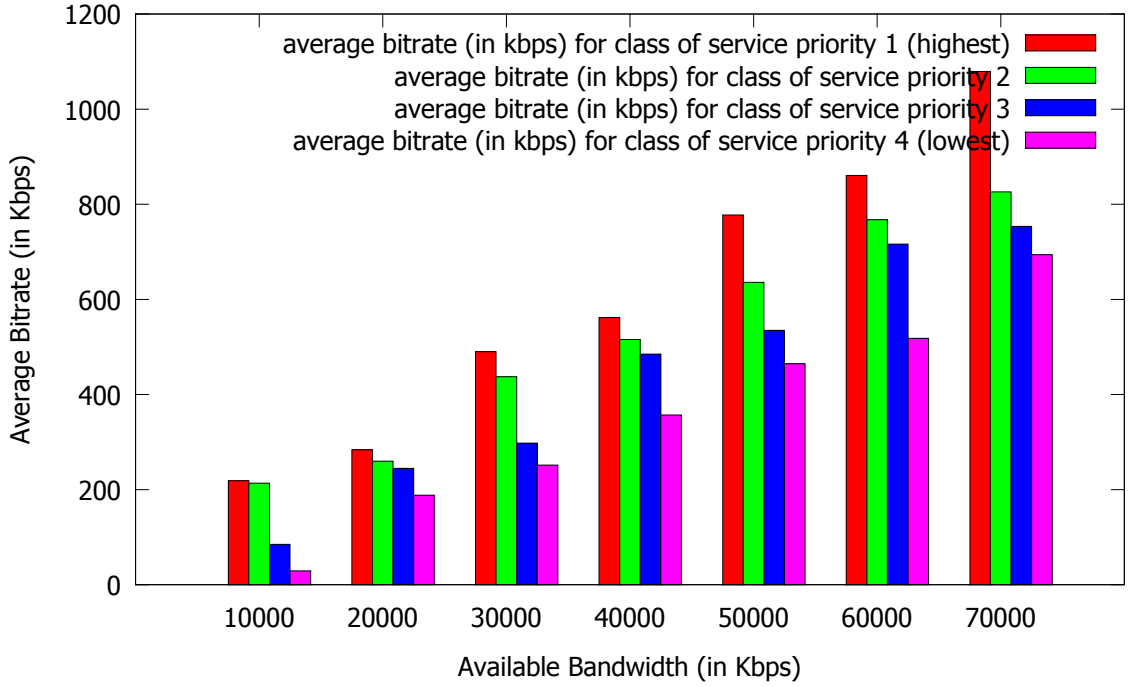

Figure 3-7: Average Allocated Bandwidth Per Class of Service per Available Bandwidth (10k to 70k with 10k increments)

Proposed Algorithm Version 2

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

classes of service for Ma's algorithm compared to proposed algorithm version 1 to 5. We see that Ma's algorithm and proposed algorithm versions 1 and 2 overlap, and that proposed algorithm versions 3, 4, and 5 have significantly higher difference in average allocated bandwidth between the different classes of service. This is because proposed algorithm versions 3, 4, and 5 uses CWF whereas proposed algorithm versions 1 and 2 do not. We see that proposed algorithm version 5 has slightly less difference in average allocated bandwidth than proposed algorithm version 4 because the former uses BCO which is designed to violate the CWF rule for certain clients in each round.

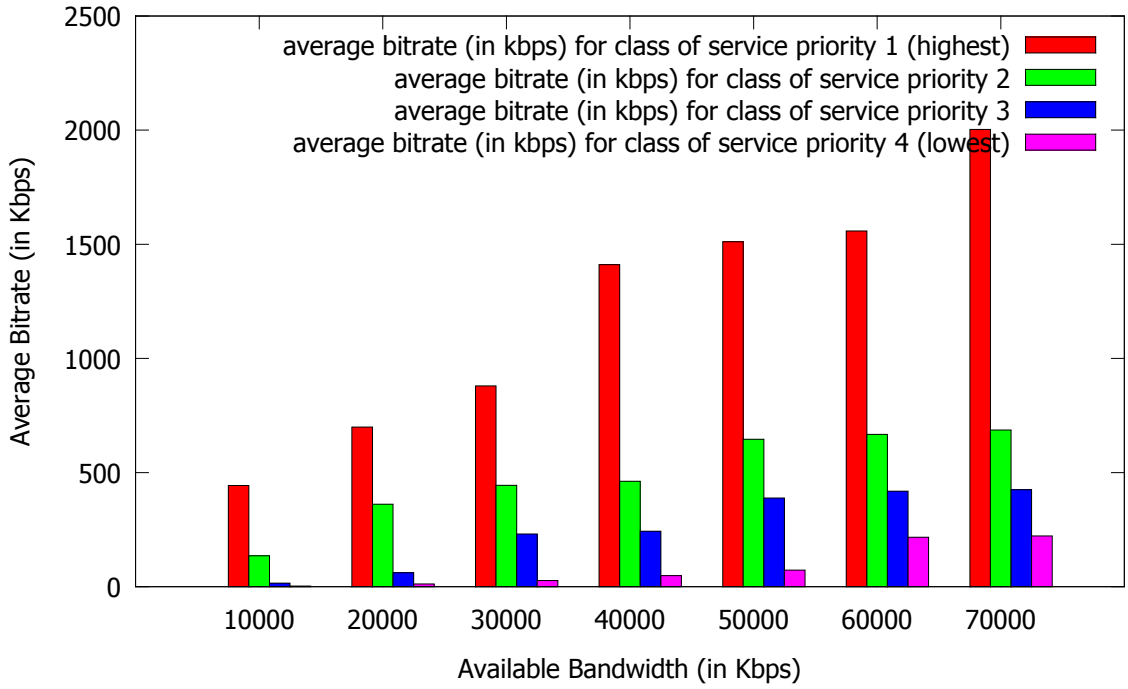

Figure 3-8: Average Allocated Bandwidth Per Class of Service per Available Bandwidth (10k to 70k with 10k increments)

Proposed Algorithm Version 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Tradeoff between Bitrate Changes and Bandwidth Utilization

Figure 3-10 shows the trade-off between average number of bitrate changes and the average bandwidth utilization, in aggregate, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. We see that proposed algorithm versions 3 and 5 have fewer average bitrate changes and less average bandwidth utilization than the remaining non-CWF and CWF versions at a cost of lower network utilization. Algorithm versions 3 and 4 are mostly overlapping, since ESV, by itself, doesn't have an impact over the overall number of bitrate changes or the average bandwidth utilization. The same goes for algorithm version 2 compared to Ma's algorithm since the only differ-

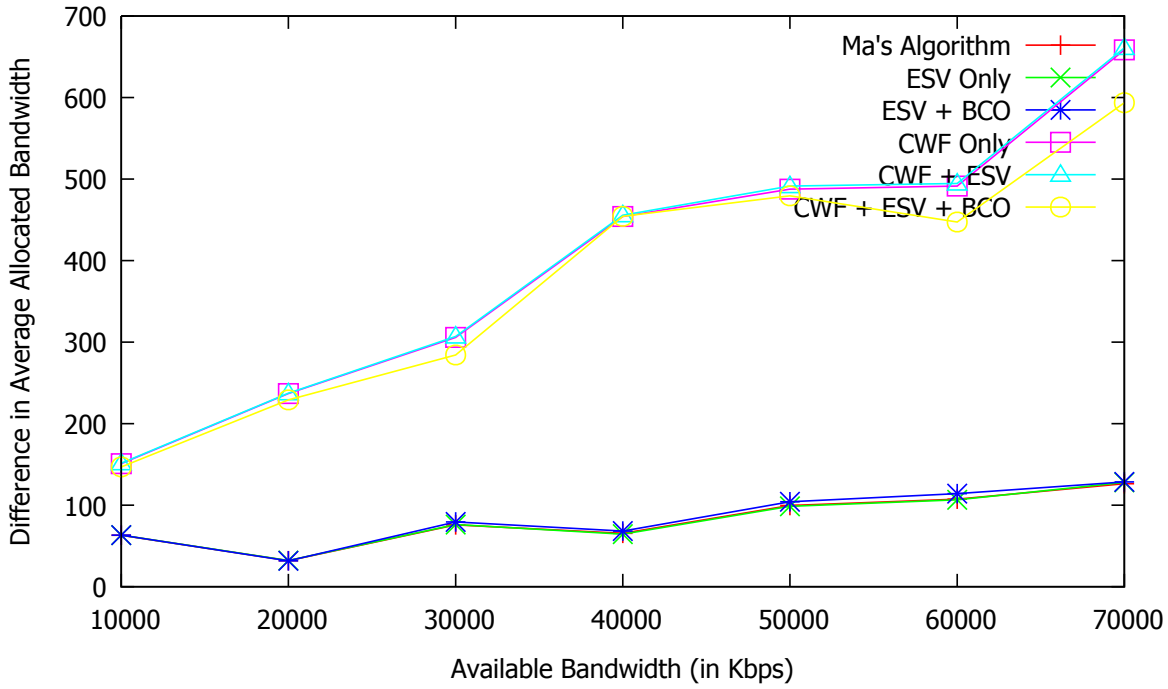

Figure 3-9: Difference in Average Allocated Bandwidth among all Classes of Service per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

ence between the two is ESV. Looking at this graph, adopters of this algorithm can make informed choices and decide what combination of feature works best for their needs in the highly-congested scenario.

CWF Violations

Figure 3-11 shows the average number of clients with CWF violations for the available bandwidth of 10,000 kbps to 70,000 kbps with 10,000 kbps increments, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. We see that non-CWF algorithms have the highest CWF

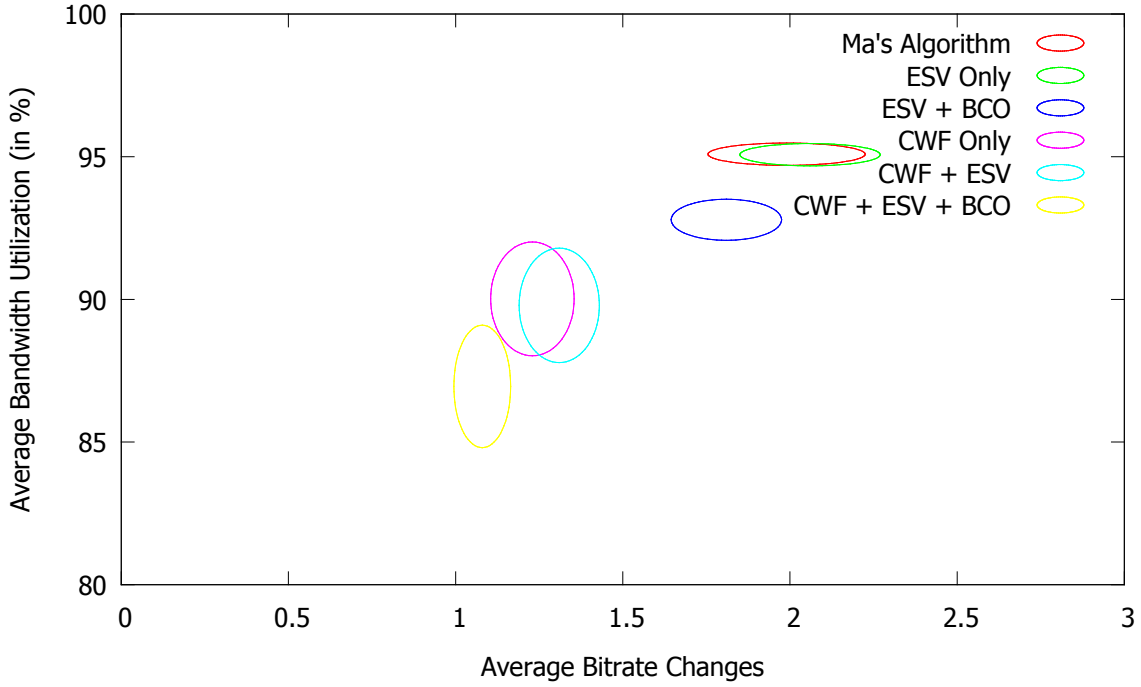

Figure 3-10: Tradeoff between Average Number of Bitrate Changes and Average Bandwidth Utilization - Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

violations compared to CWF algorithm versions, which is expected since CWF is not enforced. We see that algorithm version 2 have less clients CWF violations compared with Ma's algorithm and algorithm version 1. This is because BCO's role is to revert a bitrate switch in certain cases, which in turn could potentially revert a client back to a higher bitrate than another client in a lower class of service, and therefore preventing a CWF violation. For CWF algorithm versions, we see that proposed algorithm version 5 have around 10 clients violating CWF for the 40,000 kbps available bandwidth even when CWF is enforced. This tells us that when CWF is enforced, only a few clients will violate CWF which ultimately leads to a lower number of bitrate changes. Since BCO is not used in proposed algorithm versions 3 and 4, the number of clients with CWF violations is 0 for

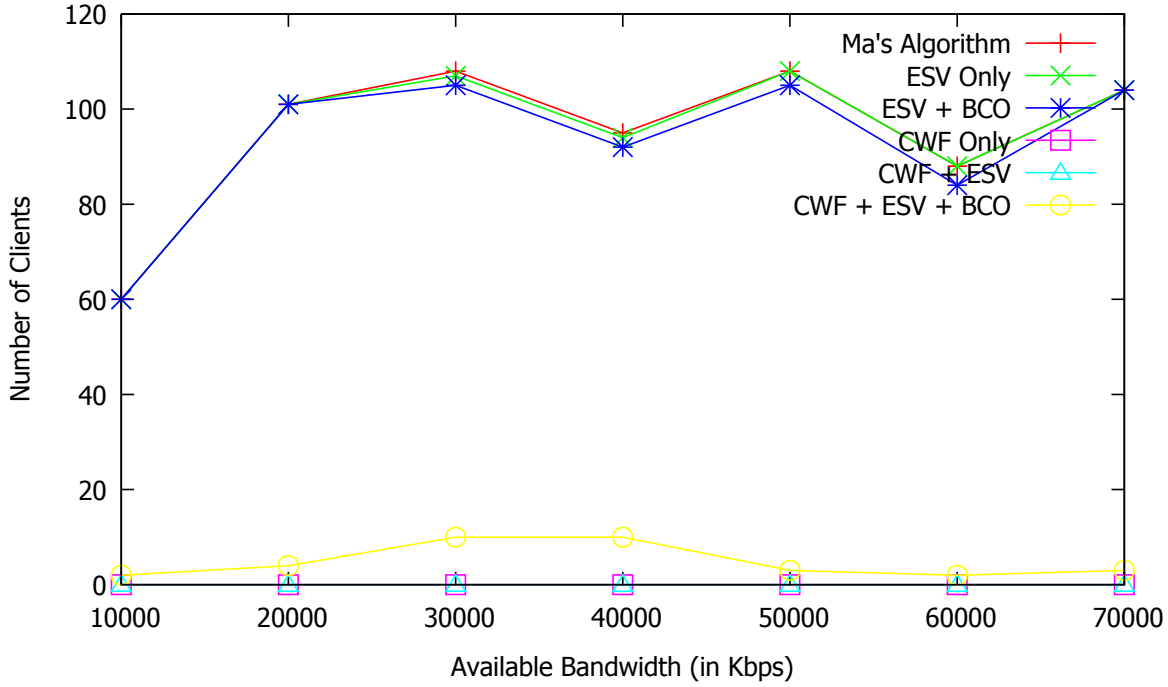

Figure 3-11: Average Number of Clients with CWF Violations per Available Bandwidth (10k to 70k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

both algorithm versions at all the available bandwidth used in this experiment.

3.3.2 80k to 140k graphs

Average Bitrate Changes

Figure 3-12 shows the average bitrate changes, for the available bandwidths of 80,000 kbps to 140,000 kbps with 10,000 kbps increments, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. Proposed algorithm version 1 performs very similarly to Ma's algorithm. That's because the only difference between proposed algorithm 1 and Ma's algorithm is that the former

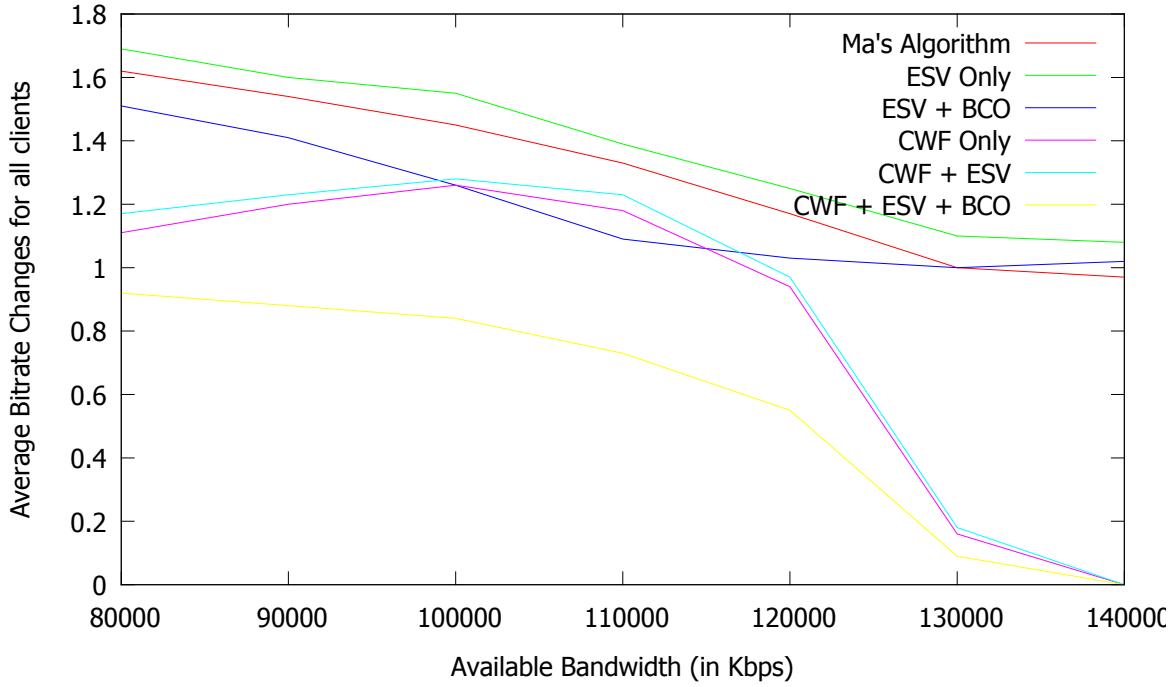

Figure 3-12: Average Bitrate Changes per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

uses ESV. Since ESV works on distributing the bitrate changes between clients in the same class of service, the average bitrate changes seem to be only slightly affected. However, as we see in proposed algorithm 2, the average of bitrate changes is for the most part lower than proposed algorithm 1 and Ma's algorithm, since BCO reduces the overall number of bitrate changes. Similarly, when we compare proposed algorithm versions 3 and 4, we see that the average bitrate change is almost identical; but we see BCO improves that in proposed algorithm version 5. Another observation in Figure 3-12 is that for CWF algorithms (proposed algorithm versions 3, 4, and 5), average bitrate change decreases significantly after 120,000 kpbs. The sudden decrease at 120,000 kbps is the result of using CWF which will cause clients to stream without needing to change bitrate when

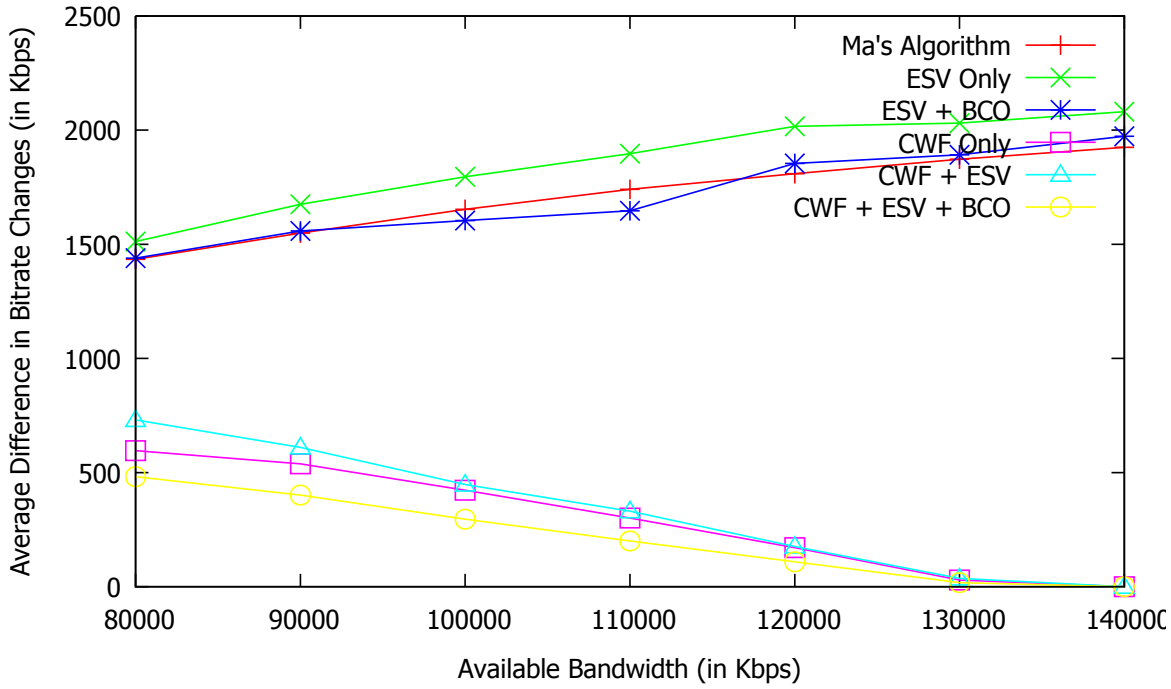

Figure 3-13: Average Difference in Bitrate Changes per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

there is enough bandwidth to assign the maximum bitrate to all clients in all classes of service. That is not the case for non-CWF algorithms, since with non-CWF, the algorithm can assign equal bitrate levels to clients from different classes of service, and therefore more than 120,000 kbps available bandwidth is needed to cover the maximum bitrate level for all clients in all classes of service. In Figure 3-13, we see the average range in bitrate changes for all clients. The CWF algorithm versions decrease with more available bandwidth until reaching almost zero at 130,000 kbps available bandwidth, since at that stage, all clients will be assigned the maximum bitrate level and there is no need for any bitrate changes. Non-CWF algorithm versions seem to have

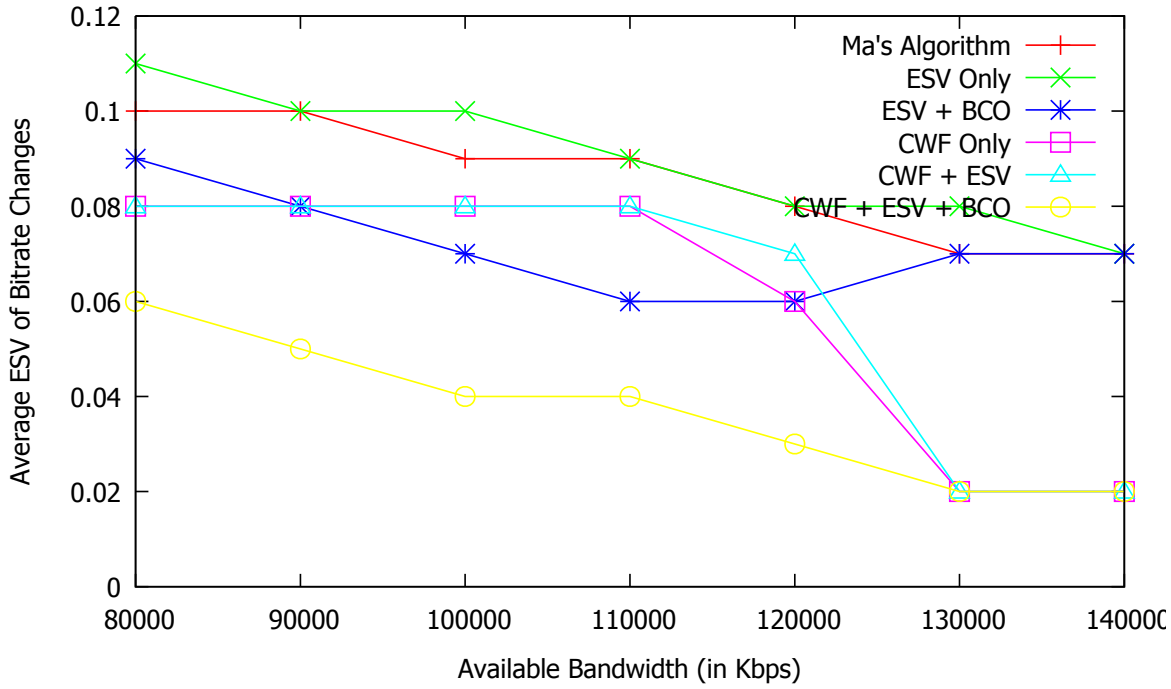

Figure 3-14: Average ESV of Bitrate Changes per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

greater average difference in bitrate levels in this moderately-congested scenario compared to CWF algorithm versions since CWF algorithm versions are serving many more clients with maximum bitrate level for their respective class of service which in turn will cause less bitrate changes when close to 130,000 kbps. Figure 3-14 shows the average ESV of bitrate changes for all clients for the different algorithm versions as well as for Ma's algorithm. We see that proposed algorithm version 5 has the least frequent bitrate changes, except at 130,000 kbps and 140,000 kbps, which correlates to the average number of bitrate changes that we saw in Figure 3-12. CWF algorithm versions 3, 4, and 5 reach average ESV of 0.02 at 130,000 kbps and 140,000 kbps since ESV is calculated the

first time a client has been assigned bitrate, and since at 130,000 kbps there is excess bandwidth, CWF clients do not need to change bitrate level after the first bitrate gets assigned.

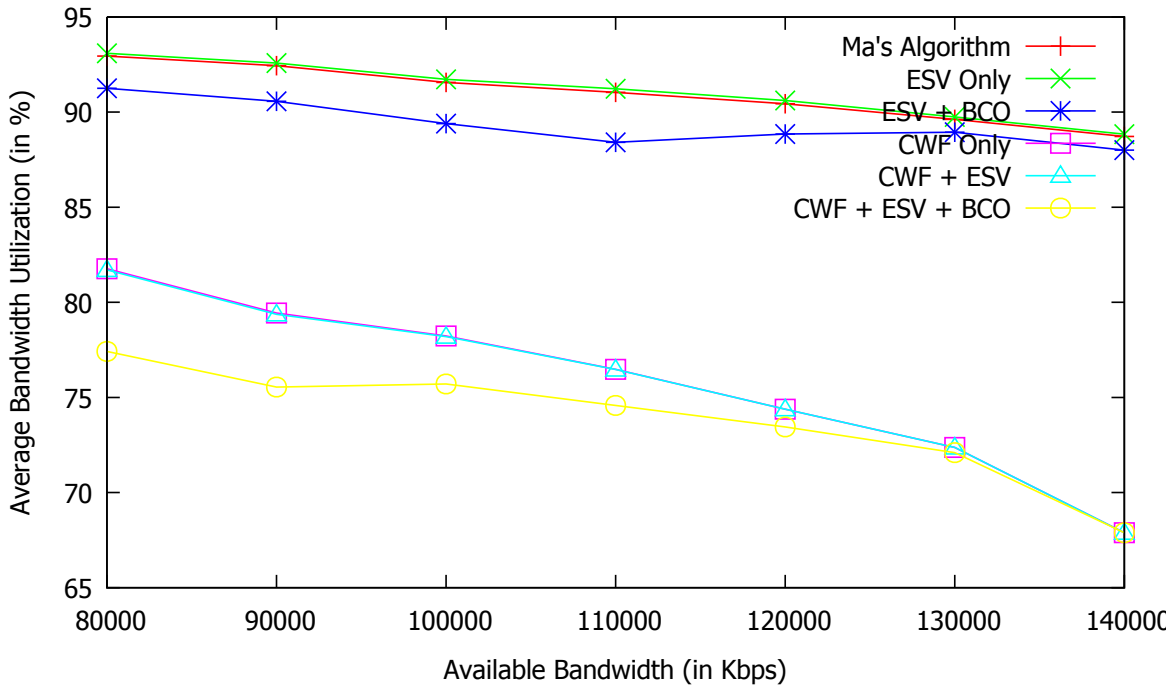

Figure 3-15: Average Bandwidth Utilization Per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Average Bandwidth Utilization

Figure 3-15 shows the average bandwidth utilization, for the available bandwidth of 80,000 kbps to 140,000 kbps with 10,000 kbps increments, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. The results of this experiment are similar to the results in Figure 3-5 in

that CWF algorithm version 5, using BCO, shows lower average bandwidth utilization than CWF algorithm versions 3 and 4, not using BCO. Similarly for non-CWF algorithm versions, using BCO causes lower average bandwidth utilization than algorithm version 1 and Ma's algorithm. The other difference in Figure 3-15 compared to Figure 3-5 is that the average bandwidth utilization is lower for all algorithm versions and that is because in Figure 3-15 there is more available bandwidth and therefore clients can be served more bandwidth and less bandwidth will be wasted.

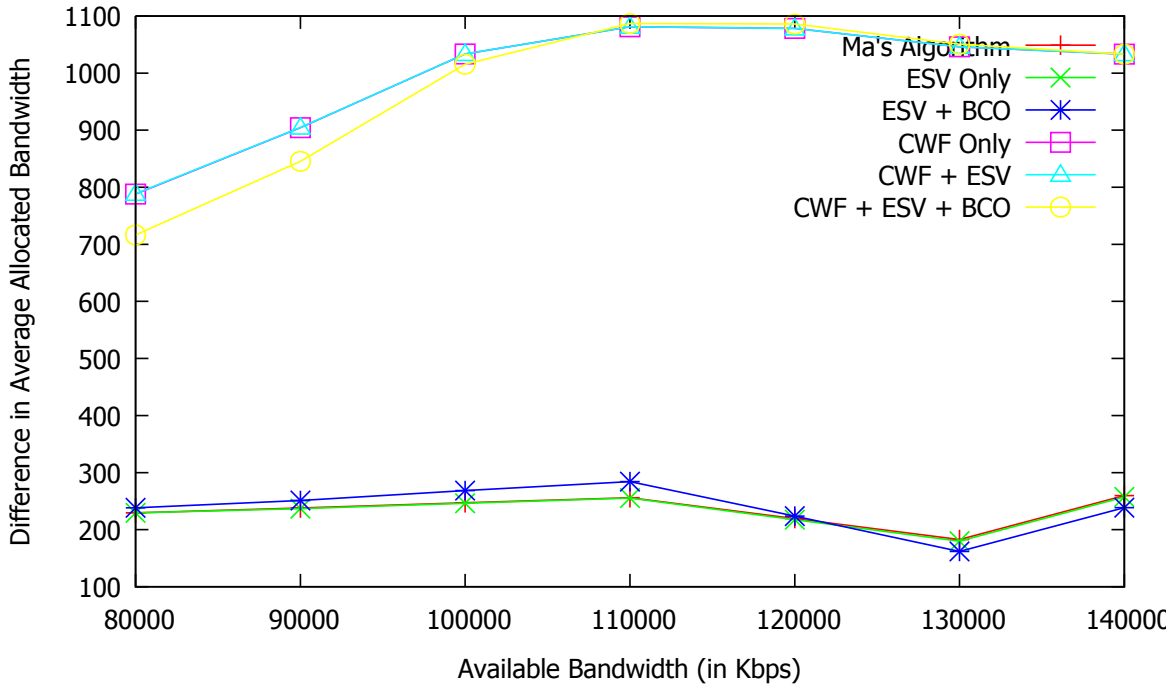

Figure 3-16: Difference in Average Allocated Bandwidth among all Classes of Service per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Difference in Average Allocated Bandwidth

Figure 3-16 shows the difference in average allocated bandwidth between all classes of service, for the available bandwidth of 80,000 kbps to 140,000 kbps with 10,000 kbps increments of the different classes of service for Ma's algorithm compared to proposed algorithm version 1 to 5. We see that Ma's algorithm and proposed algorithm versions 1 and 2 overlap, and that proposed algorithm versions 3, 4, and 5 have significantly higher difference in average allocated bandwidth between the different classes of service. This is because proposed algorithm versions 3, 4, and 5 uses CWF whereas proposed algorithm versions 1 and 2 do not. We see that proposed algorithm version 5 has slightly less difference in average allocated bandwidth than proposed algorithm version 4 because the former uses BCO which, by design, violates the CWF rule for certain clients in each round, to reduce the total number of bitrate changes.

Tradeoff between Bitrate Changes and Bandwidth Utilization

Figure 3-17 shows the trade-off between the average number of bitrate changes and the average bandwidth utilization, in aggregate, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. Here we see similar results to the Figure 3-10. As expected, CWF algorithm versions 3, 4, and 5, show significantly fewer average bitrate changes and less average bandwidth utilization than non-CWF algorithm versions 1, 2, and Ma's algorithm. Also, we see that when BCO is used, the average bitrate changes and average bandwidth utilization is even less than when BCO is not used. ESV, on the other hand, doesn't seem to have any visible effect on the the average bitrate changes or the average bandwidth utilization. So far, from the experiments shown in Figure 3-10 and Figure 3-17, we can conclude that with highly-congested (10k-70k) and moderately-congested (80k-140k) scenarios, the use of BCO seems to lower the average number of bitrate changes compared to non-BCO algorithm versions.

CWF Violations

Figure 3-18 shows the average number of clients with CWF violations for the available bandwidth of 80,000 kbps to 140,000 kbps with 10,000 kbps increments for the proposed algorithm versions 1

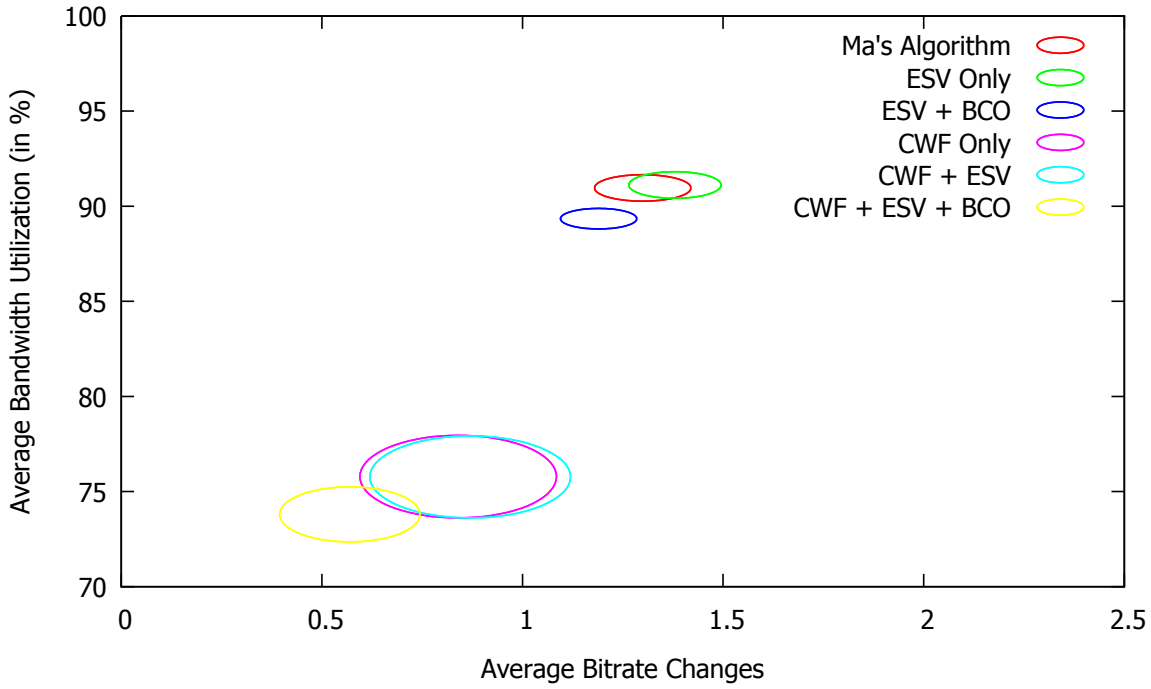

Figure 3-17: Tradeoff between Average Number of Bitrate Changes and Average Bandwidth Utilization - Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

to 5 as well as for Ma's algorithm. The results of this experiment are similar to the results of the experiment in Figure 3-11. In this experiment, we can see the magnitude of CWF violations for non-CWF algorithm versions compared to CWF algorithm versions. That is one of the trade-offs of non-CWF algorithms, which is that too many clients from one class of service could potentially receive the same bandwidth as clients from another class of service.

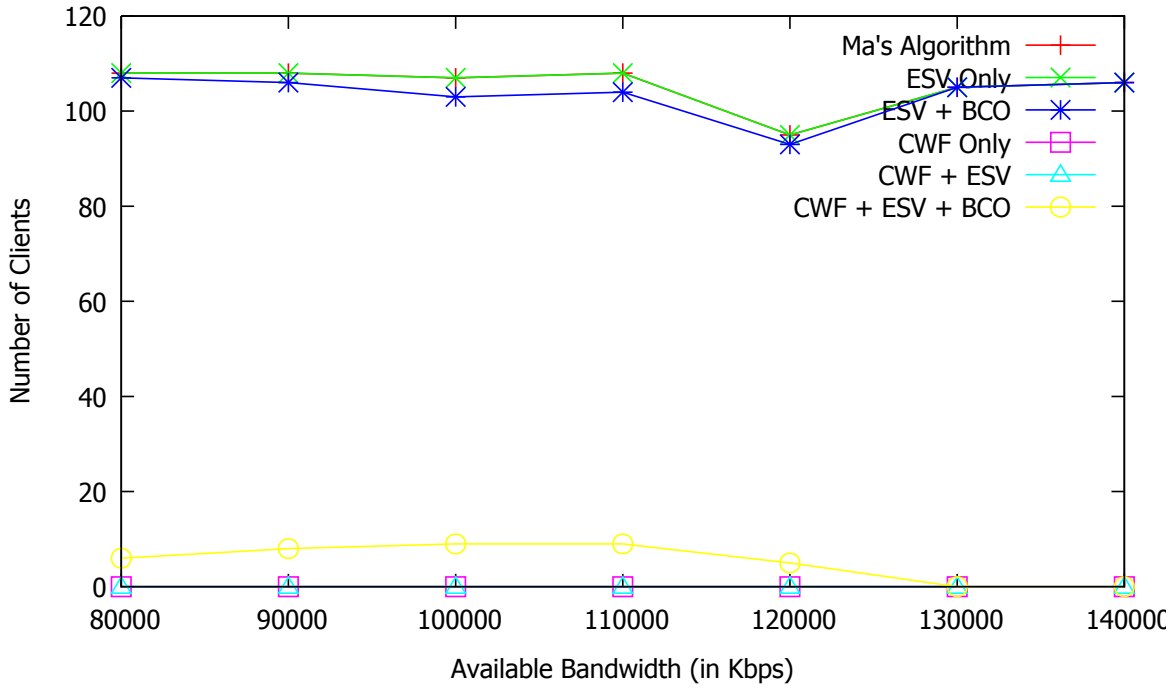

Figure 3-18: Average Number of Clients with CWF Violations per Available Bandwidth (80k to 140k with 10k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

3.3.3 150k to 450k graphs

Average Bitrate Changes

Figure 3-19 shows the average bitrate changes, for the available bandwidths 150,000 kbps to 450,000 kbps with 50,000 kbps increments, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. The graph shows that the number of bitrate changes for Ma's algorithm is higher than all of the proposed algorithm versions, except for the almost identical results compared to proposed algorithm version 1. The reason there are no bitrate switches for proposed algorithm versions 3, 4, and 5 is because there is enough bandwidth to give each of the CWF classes their max bitrates.

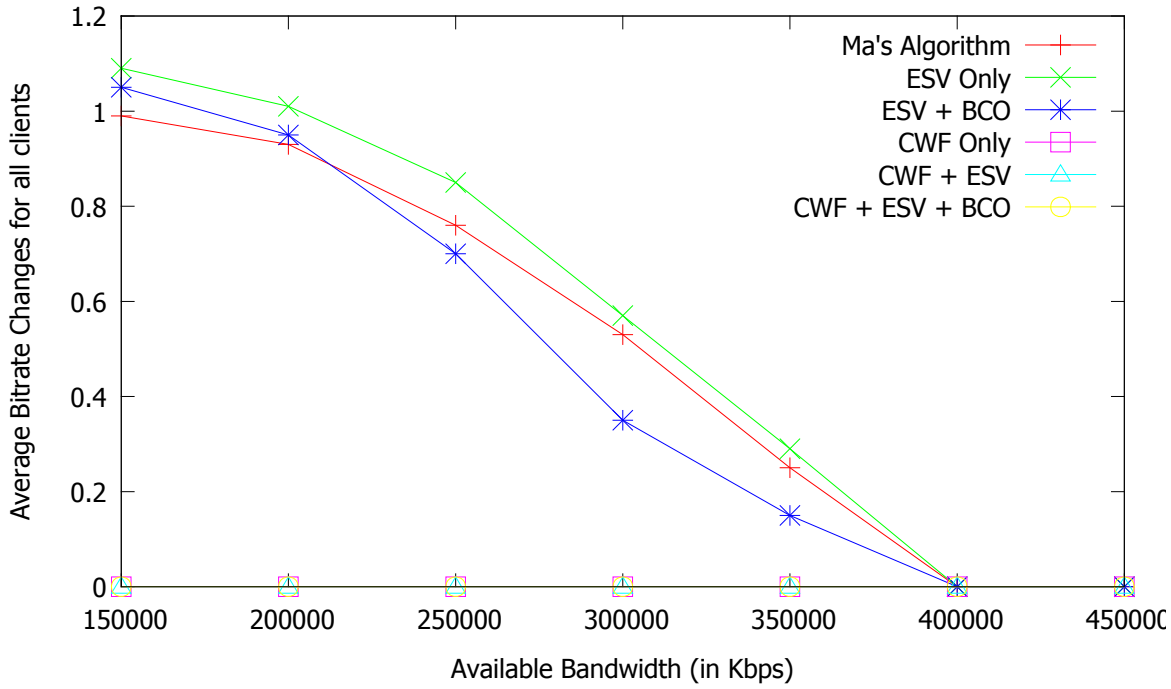

Figure 3-19: Average Bitrate Changes per Available Bandwidth (150k to 450k with 50k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Figure 3-20 shows the average difference in bitrate changes for all clients. We see that all CWF algorithm versions have 0, since all their clients will get the maximum bitrate level before even reaching 150,000 kbps. We see that non-CWF algorithm versions decrease with more available bandwidth since with more bandwidth more clients are served maximum bitrate levels and fewer clients require bitrate change.

Average Bandwidth Utilization

Figure 3-21 shows the average bandwidth utilization, for the available bandwidths 150,000 kbps to 450,000 kbps with 50,000 kbps increments for the proposed algorithm versions 1 to 5 as well

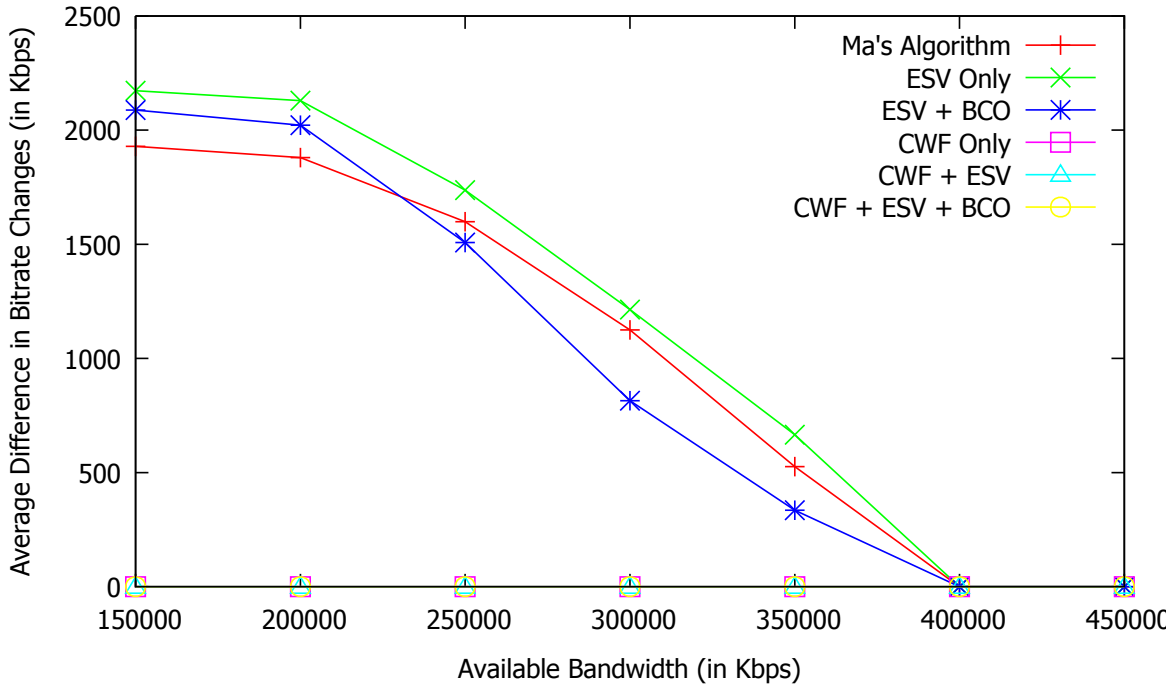

Figure 3-20: Average Difference in Bitrate Changes per Available Bandwidth (150k to 450k with 50k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

as for Ma's algorithm. Ma's algorithm appears to have the highest average bandwidth utilization compared to proposed algorithm versions 1 to 5, with the exception of proposed algorithm version 3, where the results are almost identical. This is because Ma's algorithm doesn't consider class-weighted fairness nor does it attempt to reduce the overall number of bitrate changes. Proposed algorithm version 5 shows the lowest bandwidth utilization mainly because it takes into account both rules, class-weighted fairness and bitrate changes optimization. Proposed algorithm versions 3 and 4 show almost identical results as well, which tells us that, as expected, ESV does not have an effect on overall bandwidth utilization. Proposed algorithm version 2 have significantly higher

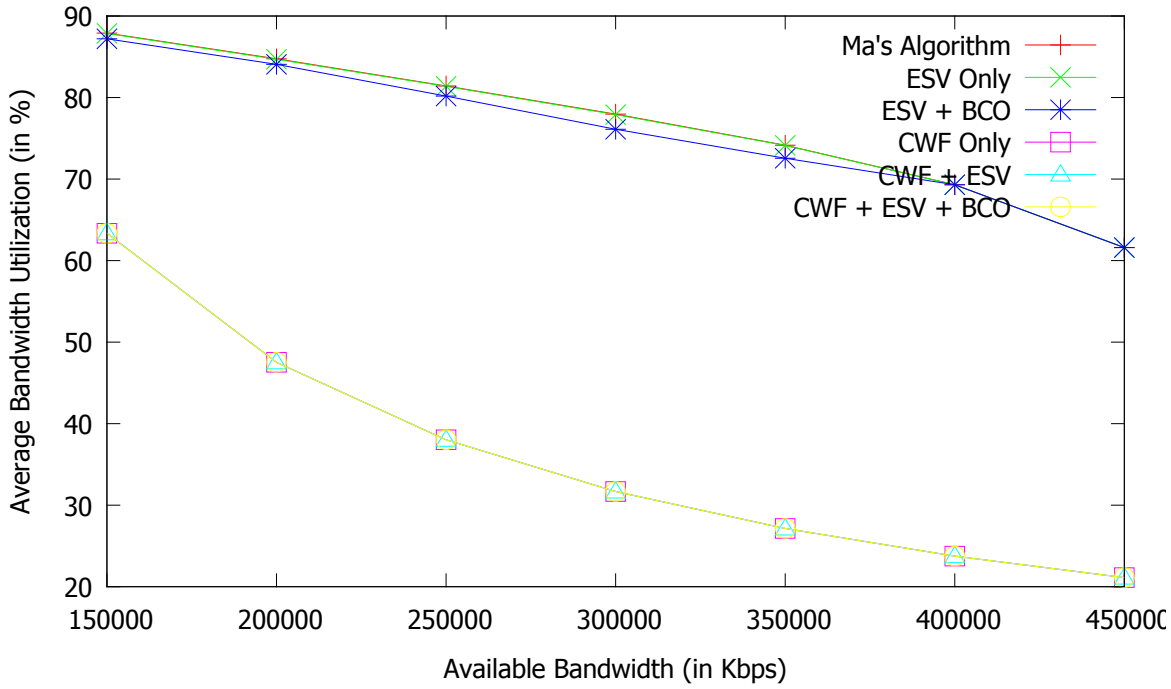

Figure 3-21: Average Bandwidth Utilization Per Available Bandwidth (150k to 450k with 50k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

utilization than proposed algorithm version 5 which tells us that CWF has a negative impact on the overall bandwidth utilization. The reason that the average bandwidth utilization decreases with more available bandwidth is because when all clients have been assigned the maximum bitrate encoding level, then the remaining available bandwidth will not be used, and therefore decreases the overall utilization.

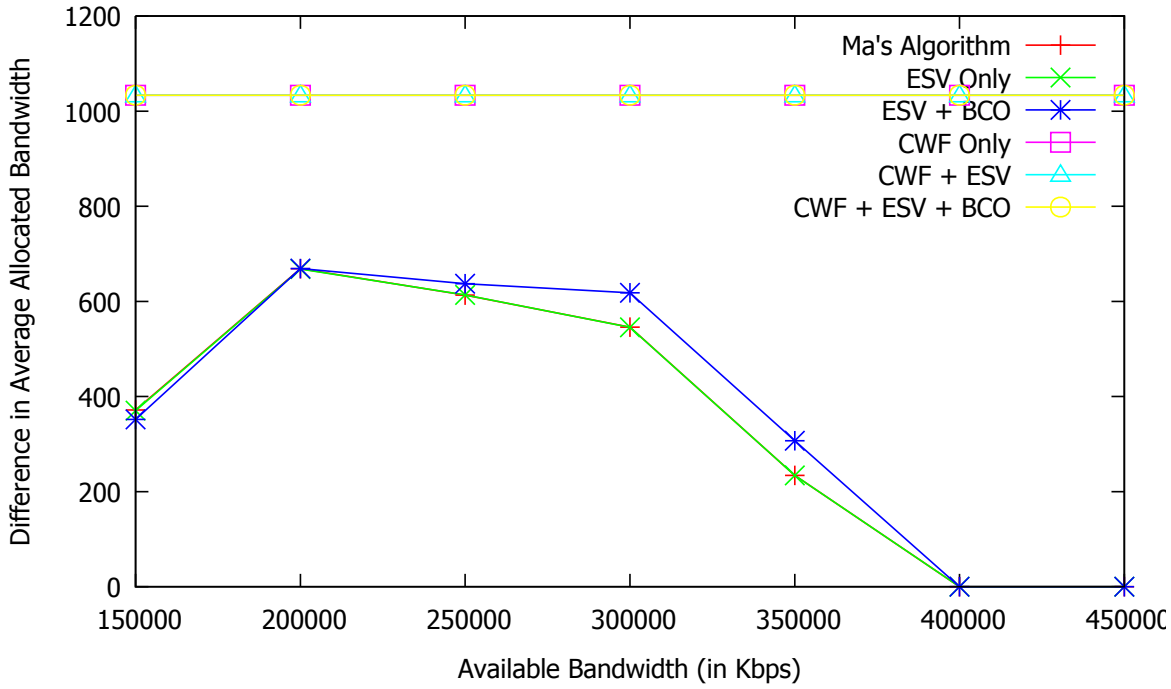

Figure 3-22: Difference in Average Allocated Bandwidth among all Classes of Service per Available Bandwidth (150k to 450k with 50k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Difference in Average Allocated Bandwidth

Figure 3-22 shows the difference in average allocated bandwidth between all classes of service, for the available bandwidths 150,000 kbps to 450,000 kbps with 50,000 kbps increments of the different classes of service for Ma's algorithm compared to proposed algorithm versions 1 to 5. We see that Ma's algorithm and proposed algorithm versions 1 and 2 overlap, and that proposed algorithm versions 3, 4, and 5 have significantly higher difference in average allocated bandwidth between the different classes of service. This is because proposed algorithm versions 3, 4, and 5 uses CWF whereas proposed algorithm versions 1 and 2 do not. Proposed algorithm versions 3, 4, and 5

overlap with each other since after 132,000 kbps, there is enough bandwidth to assign all CWF clients the maximum bitrate.

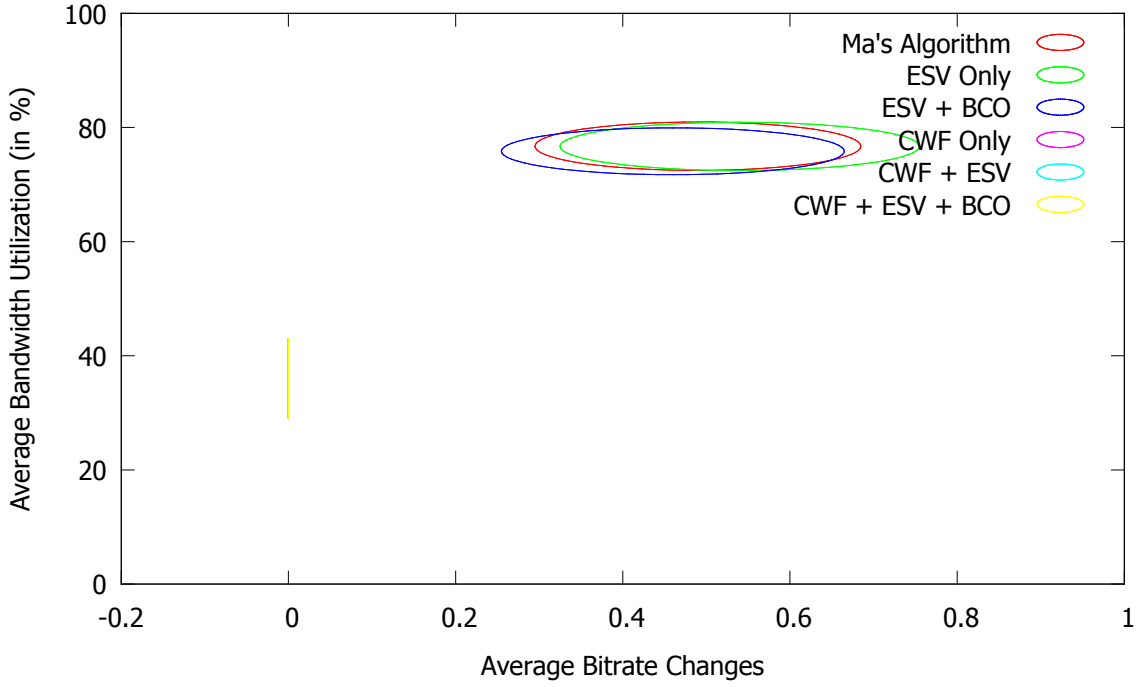

Figure 3-23: Tradeoff between Average Number of Bitrate Changes and Average Bandwidth Utilization (150k to 450k with 50k increments)

Ma's vs. Proposed Algorithm Versions 1 to 5

CWF - Class Weighted Fairness (Rule 1)

ESV - Exponential Smoothing Value of Bitrate Changes (Rule 2)

BCO - Bitrate Changes Optimization (Rule 3)

Tradeoff between Bitrate Changes and Bandwidth Utilization

Figure 3-23 shows the trade-off between the average number of bitrate changes and the average bandwidth utilization, in aggregate, for the proposed algorithm versions 1 to 5 as well as for Ma's algorithm. From this experiemnt, we see that all the CWF algorithm versions have 0 average

number of bitrate changes and low average bandwidth utilization. This is expected for the average bitrate changes, since only 132,000 kbps of available bandwidth is needed to assign the maximum bitrate levels for all the 110 clients used in this experiment. The minimum available bandwidth used in Figure 3-23 is 150,000 kbps, and therefore there wouldn't be any bitrate changes needed. Proposed algorithm versions 3, 4, and 5 overlap with each other which is the reason algorithm version 3 and 4 are not visible on the graph. As for the average bandwidth utilization, it is fairly low compared to the CWF algorithm versions 3, 4, and 5 in Figure 3-10 and Figure 3-17 and that is because a lot of available bandwidth is unused after all clients gets assigned the maximum bitrate levels in Figure 3-23.

Chapter 4

Conclusion and Future

4.1 Summary

In this project, we have outlined the characteristics of existing bandwidth allocation management technologies. As we saw, the number of bitrate changes was never taken into account when allocating bandwidth. We used Ma's algorithm [5] as a baseline, and developed a bandwidth allocation algorithm that consists of a set of three features (Class-Weighted Fairness, Exponential Smoothing Value, and Bitrate Change Optimization) that can each be enabled or disabled for each experiment. The Exponential Smoothing Value for the number of bitrate changes for each client and the Bitrate Changes Optimization techniques are used to evenly distribute the number of bitrate changes among clients as well as to reduce the average number of bitrate changes while distributing the bandwidth among all clients in the network. We show how enforcing Class-Weighted Fairness among the clients in the network affects the overall utilization at the expense of having a separation between clients from different classes of service at all time. The goal of the developed algorithm and its features is to minimize the number of bitrate changes while, if Class-Weighted Fairness is enabled, respecting the class of service priority of each client with few exceptions when the Bitrate Change Optimization feature is enabled. We plot different versions of the algorithm as well as Ma's algorithm to show how the use of Class-Weighted Fairness, Exponential Smoothing Value of the number of bitrate changes, as well as Bitrate Change Optimization affect the overall number of bitrate changes, bandwidth utilization, and average bitrate. Our experiments showed that enabling Class-Weighted Fairness, Exponential Smoothing Value, and Bitrate Change Optimization results in the lowest number of bitrate changes while negatively, but slightly, affecting the overall

bandwidth utilization. We also showed a trade-off between number of bitrate changes and bandwidth utilization for each algorithm, including Ma's algorithm. The experiments were based on three scenarios: highly-congested networks, moderately-congested networks, and lightly-congested networks. More experiments are needed to cover many other scenarios, but in the current set of experiments, we have demonstrated that our approach can reduce the overall number of bitrate changes, which has been shown by Robinson et al. to improve the overall Quality of Experience [8].

4.2 Future Work

In the future, the work done in this project can be expanded to study how modifying the simulator to support clients joining the network then leaving it multiple times in the same run could affect the overall number of bitrate changes and the average bandwidth utilization. In addition to that, we can experiment with different exponential smoothing constant values, as well as with different numbers of clients in each class of service. We can develop more experiments to study difference between low-to-high vs. high-to-low bitrate change on the overall number of bitrate changes. We can also study how limiting the bandwidth utilization to a pre-set threshold could affect the number of bitrate per individual client as well as per average. After we thoroughly study the behavior of the algorithms with various settings and variables, we can apply the developed bandwidth allocation algorithms on real devices. To do so, we will crowd-source our experiments to generate mobile data traffic from a real-world devices in a real-world scenario.

Bibliography

[1] Apple. *Best Practices for Creating and Deploying HTTP Live Streaming Media for the iPhone and iPad*, February 2014.

[2] Apple. *HTTP Live Streaming Overview*, April 2015.

[3] D. Bulichenko. HTTP Dynamic Streaming with Flash Access Protection. Technical report, Adobe, April 2011.

[4] Cisco. *The Zettabyte Era - Trends and Analysis*, May 2015.

[5] K. J. Ma and R. Bartos. HTTP live streaming bandwidth management using intelligent segment selection. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5, December 2011.

[6] K. J. Ma and R. Bartos. CoS enforcement for HTTP adaptive streaming. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1373–1377, December 2012.

[7] R. Pantos. HTTP live streaming. Internet-Draft Version 16 (draft-pantos-http-live-streaming-16), IETF Secretariat, April 2015, work in progress.

[8] D. C. Robinson, Y. Jutras, and V. Craciun. Subjective video quality assessment of HTTP adaptive streaming technologies. *Bell Labs Technical Journal*, pages 5–23, March 2012.

[9] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE Multimedia Magazine*, 18(4):62–67, 2011.

[10] N. Staelens, J. D. Meulenaere, M. Claeys, G. V. Wallendael, W. V. den Broeck, J. D. Cock, R. V. de Walle, P. Demeester, and F. D. Turck. Subjective quality assessment of longer duration video sequences delivered over HTTP adaptive streaming to tablet devices. *IEEE Transactions on Broadcasting*, 60(4):707–714, December 2014.

[11] T. Stockhammer. HTTP live streaming bandwidth management using intelligent segment selection. In *MMSys '11 Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, February 2011.

[12] J.-K. Sungsu Kim; Sin-seok Seo; Joon-Myung Kang; Pujolle, G.; Hong. Autonomic resource allocation for video streaming services in content delivery networks. In *Global Information Infrastructure and Networking Symposium (GIIS), 2012*, pages 1–4, December 2012.

[13] A. Zambelli. IIS smooth streaming technical overview. Technical report, Microsoft Corporation, March 2009.