End-to-End Video Adaptation using Frame Rate Optimization and TCP-Friendly Rate Control

By
Evan Li Shen Tan

A thesis submitted for the degree of
Doctor of Philosophy

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY, AUSTRALIA

School of Computer Science and Engineering,
The University of New South Wales.

September 2011
Abstract 350 words maximum: (PLEASE TYPE)

In this research, we focus on two areas of video adaptation: video rate control and frame rate control. The first part is on video rate control, which aims to adjust video bitrate to meet the network constraint. To that end, we propose a joint source rate and congestion control scheme called video TCP-friendly rate control (VTFRC) that incorporates the video bit rate characteristic into the TFRC rate. VTFRC uses a frame complexity measure and the rate gap between the TCP and TFRC rates to opportunistically encode the video at a higher rate. Experiments show that VTFRC improves video quality over existing scheme while maintaining TCP-friendliness.

However, VTFRC needs the encoder meet a target bitrate and provide a frame complexity measure. To do this, we propose a complexity-based rate control scheme using edge energy. We show that this scheme can describe the individual complexities of the frames without needing any information on the whole video. Experiments show that the scheme produces a video stream that is closer to the target bitrate while improving on its video quality over existing schemes.

The second part of this thesis is on frame rate control, which aims to maximize the video quality and continuity using the frame generation rates. To approach this, we propose a frame rate optimization framework that characterizes the frame rate control problem and use Lyapunov optimization to systematically derive decoupled optimization policies. Results show that the framework reduces the prebuffering requirements significantly with a modest tradeoff in video quality.

We then examine different ways of improving the frame rate optimization framework. Firstly, we reformulated the framework based on a discontinuity penalty virtual buffer, which is the cumulative difference between the receiving interval and playout interval. We show that this discontinuity penalty correlates to the discontinuity of the video and enables a wider range of frame quality functions to be used with the framework. Secondly, we introduce a constraint on the playout slowdowns using a virtual buffer that records the cumulative delays. Results show that this provides a superior tradeoff between the video quality and the delay introduced compared to the existing scheme. Lastly, we show how the optimization policies can be derived in the presence of feedback delays. Analyses show that the delayed feedbacks had a minimal impact on the optimization policies.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or hereafter known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

Signature  Witness  Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY

Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS
COPYRIGHT STATEMENT

'I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or hereafter known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstract International (this is applicable to doctoral theses only). I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.'

Signed .................................................................

Date .................................................................

12/09/2011

AUTHENTICITY STATEMENT

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed .................................................................

Date .................................................................

12/09/2011
ORIGINALLITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed

Date: 12/09/2011
Abstract

In this research, we focus on two areas of video adaptation: video rate control and frame rate control. The first part is on video rate control, which aims to adjust video bit rate to meet the network constraint. To that end, we propose a joint source rate and congestion control scheme called video TCP-friendly rate control (VTFRC) that incorporates the video bit rate characteristic into the TFRC rate. VTFRC uses a frame complexity measure and the rate gap between the TCP and TFRC rates to opportunistically encode the video at a higher rate. Experiments show that VTFRC improves video quality over existing scheme while maintaining TCP-friendliness.

However, VTFRC needs the encoder meet a target bitrate and provide a frame complexity measure. To do this, we propose a complexity-based rate control scheme using edge energy. We show that this scheme can describe the individual complexities of the frames without needing any information on the whole video. Experiments show that the scheme produces a video stream that is closer to the target bitrate while improving on its video quality over existing schemes.

The second part of this thesis is on frame rate control, which aims to maximize the video quality and continuity using the frame generation rates. To approach this, we propose a frame rate optimization framework that characterizes the frame rate control problem and use Lyapunov optimization to systematically derive decoupled optimization policies. Results show that the framework reduces the prebuffering requirements significantly with a modest tradeoff in video quality.

We then examine different ways of improving the frame rate optimization framework. Firstly, we reformulated the framework based on a discontinuity penalty virtual buffer, which is the cumulative difference between the receiving interval and playout interval. We show that this discontinuity penalty correlates to the discontinuity of the video and enables a wider range of frame quality functions to be used with the framework. Secondly, we introduce a constraint on the playout slowdowns using a virtual buffer that records the cumulative delays. Results show that this provides a superior tradeoff between the video quality and the delay introduced compared to the existing scheme.
Lastly, we show how the optimization policies can be derived in the presence of feedback delays. Analyses show that the delayed feedbacks had a minimal impact on the optimization policies.
In memory of my grandmother,
Kwek Seok Yan
1918-2009
Acknowledgements

Firstly, I am grateful to my supervisor Chun Tung Chou for his research guidance and support. The discussions we had and his critiques of my work have not only helped to considerably improve the quality of my research, but also helped me gain a deeper appreciation of the research process. I would also like to thank Jing Chen for his supervision in the first two and a half years of my PhD candidature. He helped me set the theme and lay the ground work for my research.

I also appreciate the research discussions I had with Reiji Mathew, Jian Zhang, Sebestien Ardon and Emmanuel Lochin. Furthermore, I would like to extend my thanks to the people in the multimedia group in NICTA for their insights and comments: Getian Ye, Yang Wang, Sijun Lu and Jack Yu.

A big thank you to the colleagues whom I had the pleasure of interacting with: Gunawan, Jie Xu, Paul, Matt, Nobu, John, Tuan, Luo Cheng, Weihong, Zhidong Li, Yut, Worapan, Pranam, Alpa, Joo Ghee and Rajan.

I am also grateful to the people in Microsoft Research Asia for making my three months stay there a memorable one. In particular, my mentor Chong Luo and my research manager Feng Wu, both of whom helped to support and guide my work there. And a warm thank you to the great friends I made there, Hao Cui, Wei Pu and Zhaotai Pan.

Finally, I would not have even started on my PhD without the support and love of my parents. They supported me when I first came to study in Australia years ago right until now and even when their health was letting them down. I owe thanks to my in-laws for the support and encouragement they have given me. I also owe a debt of gratitude to my loving wife Jue Wang, she made a large non-technical contribution to this thesis :).

My PhD candidature has been supported by the Australian Postgraduate Research Award, Faculty of Engineering Supplementary Award and the NICTA Research Project Award.
Contents

Abstract i

Acknowledgements iv

Chapter 1. Introduction 1
  1.1. Thesis Goals 2
  1.2. Contributions 3
  1.3. Thesis Outline 4

Chapter 2. Literature Review 5
  2.1. Introduction 5
  2.2. H.264/AVC Rate Control Background 5
  2.3. H.264/AVC Rate Control Schemes 12
  2.4. TCP-Friendly Rate Control 14
  2.5. Video Rate Control Over Networks 17
  2.6. Prebuffering Video 19
  2.7. Frame Rate Control 20
  2.8. Lyapunov Optimization 22

Chapter 3. Video TCP Friendly Rate Control 29
  3.1. Introduction 29
  3.2. Design Overview 31
  3.3. Rate Negotiation 35
  3.4. Experimental Results 37
  3.5. Conclusion 44

Chapter 4. Complexity Based H.264/AVC Rate Control 45
  4.1. Introduction 45
  4.2. Process Overview 47
  4.3. Extracting Edge Energy 48
  4.4. Frame Complexity Measure 50
  4.5. Bit Allocation 51
  4.6. Quantization Parameter Selection 53
  4.7. Experimental Results 54
  4.8. Conclusion 63
CHAPTER 1

Introduction

Video streaming has become more pervasive since it was touted as the next killer application for the Internet \[1, 2\]. It is now common to stream television programs, on demand, into computers, set-top boxes and mobile devices. Live event streaming is also now commonplace, one example is the 2008 Beijing Olympics that generated 5000 hours of online video content with more than 80 million viewers worldwide \[3\]. Video streaming has also been used in applications as diverse as blogging \[4\] and gaming \[5\]. However, video streaming brings along a myriad of issues that impacts on the video quality, possibly making it unacceptable to the users.

One class of techniques that help make video streaming acceptable to the user is video adaptation. Video adaptation is defined by Chang and Vetro \[6\] as a tool or system that adapts one or more video programs to generate a new presentation with a video or multimedia format to meet user needs in customized situations. Video adaptation is needed to ensure that the video adapts to the time varying network conditions, meets the network resource constraints, support heterogeneous terminals and satisfy user tasks. Video adaptation is a broad research area, in this thesis, we focus on two areas: video rate control and frame rate control.

The goal of video rate control is to adjust the encoder parameters such that the resulting video bitrate meets the channel or network constraints. Typically, the video quality needs to be traded off to ensure that the video bitrate meets the network available bandwidth. A popular approach is to integrate the video application with the transport control protocol (TCP) for rate control. This is evident in streaming technologies such as the widely used Adobe Flash video \[7\] and Microsoft’s Smooth Streaming \[8\]. The advantages of doing this is that TCP inherently performs congestion control. This is important as it helps the network to avoid a congestion collapse \[9\] if there is too much traffic in the network. Moreover, given that TCP is the de facto protocol being used, the integration of video application with TCP for video rate control ensures that it will compete fairly with the majority of the Internet traffic. However, TCP provides a guaranteed delivery of packets using a packet retransmission mechanism. This behaviour is difficult to control and potentially causes unwanted
additional delays into the streaming application. Furthermore, perhaps more seriously, TCP’s congestion control mechanism works by halving the sending rate in response to a single congestion event. This causes severe rate fluctuations and is known to have an adverse impact on the viewer perceived video quality [10].

The goal of frame rate control is to adjust the frame generation rate of the sender and receiver such that the video quality and video continuity are maximized. Where the frame generation rate refers to the number of frames being output by the sender for video transmission and by the receiver for playing out the video to the viewer. Video continuity is defined in this thesis as the length of time the video is played without interruptions due to buffer underflow. A low video continuity results in a stop-start video that is known to impact the viewer perceived video quality [11]. A widely used approach to improve video continuity is to prebuffer the video at the viewer end. Prebuffering is a simple mechanism that is guaranteed to provide video continuity against sudden bandwidth fluctuations and network jitter. However, it introduces additional delay to the streaming system and potentially causes a long initial delay. A long initial delay will cause the video to take longer to load as well as prevent the viewer from switching the channel rapidly to preview the channel content. These are known to impact on the viewer perceived quality of the video [12]. Furthermore, it has been shown that the prebuffering required to provide maximum video continuity for a video is directly proportional to the length of the video [13]. So the longer the video, the larger the amount of prebuffering required.

The aim of this thesis is to address the above-mentioned problems in video rate control and frame rate control.

1.1. Thesis Goals

The main goals of this thesis are:

(1) Design a video rate control scheme that improves video quality while ensuring a TCP-friendly video bitrate. The idea of TCP-friendliness is that a TCP-friendly stream behaves in a similar manner to a TCP stream in identical conditions. Since TCP is the de facto protocol used in most networks, being TCP-friendly will allow the stream to compete fairly with the majority of the network traffic.

(2) Design a frame rate control scheme that finds the optimal trade-off between video quality and video continuity. The optimal scheme would also lower the prebuffering requirements for the video application.
1.2. Contributions

The main contributions of this thesis are:

- A joint source rate and congestion control scheme called video TCP-friendly rate control (VTFRC) that incorporates the video bit rate characteristic into the calculation of the TCP-friendly transmission rate. The video bit rate characteristic can be determined more accurately by using a frame complexity measure. We show that using the frame complexity measure allows VTFRC to identify and opportunistically encode the more complex parts of the video at a higher rate while maintaining TCP-friendliness. We then show through experiments that VTFRC streams improves video quality while competing fairly and in a non-aggressive manner with other TCP traffic when compared to the existing scheme.

- A complexity based encoder rate control scheme that complements VTFRC by ensuring that the encoded video meets the required target bitrate. We demonstrate how the edge energy of a frame can be used to describe its frame complexity. We then show that this scheme does not require any information on the whole video sequence which the existing scheme needs. We finally show that the proposed scheme improves over existing schemes by producing a video stream that is closer to the required target bitrate with a better video quality.

- A frame rate optimization framework that performs frame rate control to maximize the video quality and video continuity. We characterize the frame rate control problem and use Lyapunov optimization to systematically derive the optimization policies. We show that these policies can be decoupled into separate encoder and decoder optimization policies with feedback, thus allowing a distributed implementation. We show that these policies provide maximum video continuity with a modest tradeoff in video quality. Moreover, the prebuffering requirements are reduced significantly.

- A study on the improvements that can be made on the proposed frame rate optimization framework. The following improvements to the framework were made:
  - A re-formulation of the frame rate control problem using virtual buffer concept [14]. We show that this new approach to the same problem allows a broader class of video quality functions to be supported within the framework.
  - Constraining the delay introduced by the framework. We show
that a delay constraint can be added into the framework using the virtual buffer mechanism. We then demonstrate that the delay constrained framework provides a superior tradeoff between the video quality and the delay introduced compared to the existing approach.

- An analysis on the effect of delayed feedbacks on the derived optimization policies. We show that the delayed feedbacks had a minimal impact on the optimization policies.

1.3. Thesis Outline

The following lists the outline of this thesis:

- Chapter 2 provides an overview of the current literature that are closely related to our work.

- Chapter 3 presents the video TCP-friendly rate control (VTFRC) scheme that is proposed for video rate control.

- Chapter 4 presents the complexity based encoder rate control. In this chapter, we look into a more suitable frame complexity measure that could be used by VTFRC.

- Chapter 5 describes the frame rate optimization framework.

- Chapter 6 extends the work in chapter 5 by proposing various improvements to the frame rate optimization framework.

- Chapter 7 concludes this thesis.
CHAPTER 2

Literature Review

2.1. Introduction

In this chapter, we review the concepts from the literature that are used in this thesis as well as discuss related work. This chapter is organized as follows: Section 2.2 covers the concepts behind the encoder rate control scheme that was adopted into the H.264/AVC video coding standard, which is used in chapter 4. Section 2.3 reviews the current H.264/AVC rate control techniques. Section 2.4 covers the concepts behind TFRC, a congestion control protocol alternative to TCP, this is used in chapter 3. Section 2.5 reviews rate control techniques used in computer networks. Section 2.6 reviews the prebuffering techniques used. Section 2.7 reviews current frame rate control techniques. Section 2.8 covers the basics behind Lyapunov optimization which is based on Lyapunov drift analysis, chapters 5 and 6 use the concepts here.

2.2. H.264/AVC Rate Control Background

H.264/AVC is the latest video coding standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG). It is widely supported in many Internet video streaming sources.

The aim of the H.264/AVC rate controller is to select a certain quantization parameter (QP) to ensure that the output video from the encoder meets a certain target bitrate. The QP will determine the amount of compression the video goes through. A rate controller would normally choose the minimal compression required for the video to meet the target bitrate.

2.2.1. Rate Control Overview

![Figure 2.1. A video encoding flow.](image-url)
A video rate controller is a part of the encoder that ensures that the output video stream meets a certain target sequence bitrate, where the target sequence bitrate could be set by the user or determined by the maximum bitrate the channel can support. Note that this target sequence bitrate may vary over time if the channel has a time varying bandwidth. Fig. 2.1 illustrates the role of a rate controller. The video rate controller performs its task by adjusting the encoding parameters within the encoder. One typical parameter to adjust is the quantization parameter (QP), this is because QP significantly affects the resulting bitrate of a video stream.

There are numerous video rate control schemes proposed, here we specifically review the H.264/AVC rate control scheme proposed by Li et al. Li et al’s rate control scheme has been adopted into the non-normative section of the H.264/AVC standard.

In Li et al’s rate control scheme, the QPs of intra frames (I-frames), B-frames and the first P-frame in the group-of-pictures (GOP) are heuristically calculated using the QPs of neighbouring frames (fig. 2.2 illustrates the GOP concept). In this section, we focus on explaining the concepts behind the calculation of QPs for P-frames after the first one in a GOP.

For these frames, the scheme has two main steps, bit allocation and QP selection. Bit allocation determines how the target sequence bitrate is allocated among a certain number of frames in the video. While QP selection uses the rate determined in the bit allocation step to determine the QP, this is done using a rate-quantization model proposed by Chiang and Zhang.

The rate control concepts presented here are used in chapter 4. The reference literature for H.264/AVC rate control are Li et al, Lee et al and Chiang and Zhang.

Figure 2.2. Group-of-pictures (GOP). A GOP begins with an intra frame (I-frame) and ends before the next I-frame. The frames in between can be a mix of P- and B-frames. In the illustration, the GOP size is 10 and GOP structure is IPPP.

\[1\] For example, the transport layer can feedback the available network bandwidth as the target sequence rate.
2.2.2. Bit Allocation

\[
\frac{R(i,k)}{R_F} \rightarrow \text{Buffer} \rightarrow \frac{R(i,k)}{R_F}
\]

**Figure 2.3.** The general fluid flow model. Each timeslot corresponds to a frame interval. \( R(i,k) \) is the target sequence bitrate rate as at GOP \( i \) frame \( k \), \( R_F \) is the sequence frame rate.

The H.264/AVC non-normative rate control scheme uses two mechanisms to determine the bit allocation: GOP-based bit allocation and simulated buffer\(^2\), both of these mechanisms make use of the fluid flow model.

Fig. 2.3 illustrates a fluid flow model. Given that \( R(i,k) \) is the target sequence bitrate rate at the \( i \)'th GOP in timeslot/frame \( k \) and \( R_F \) is the sequence frame rate. In this chapter, frame index \( k \) is relative to the first frame of each GOP, so every time the encoder progresses to a new GOP, \( k \) will be reset to zero at the beginning of the GOP. The sequence frame rate is the frame rate at which the sequence is captured in, this is typically a constant term in the lifetime of the video. In the fluid flow model, each time slot corresponds to a frame interval, i.e. the delay between two frames, and in each time slot \( \frac{R(i,k)}{R_F} \) bits arrives and departs the encoder buffer. \( \frac{R(i,k)}{R_F} \) bits represents the average frame size of the video. This is because when each frame of the video is encoded in \( \frac{R(i,k)}{R_F} \) bits and pumped out at the sequence frame rate \( R_F \), the target sequence bitrate \( R(i,k) \) will be met.

In this section, we look at how the H.264/AVC non-normative rate control scheme allocates bits for each frame (i.e. frame-layer rate control). We define the number of bits to allocate a frame as the frame target bits.

2.2.2.1. Simulated Buffer

\[
b(i, k - 1) \rightarrow \text{Buffer} \rightarrow \frac{R(i,k)}{R_F}
\]

**Figure 2.4.** The simulated buffer used. \( b(i, k - 1) \) is the actual bits used by the previously encoded frame, \( \frac{R(i,k)}{R_F} \) is the average departure bitrate.

The simulated buffer used in Li et al.’s scheme determines the frame target bits by estimating the amount of video data in the encoder buffer, this is done by assuming an average departure rate of \( \frac{R(i,k)}{R_F} \). The encoder buffer stores video data that has just been encoded but not yet transmitted through the channel.

\(^2\)What we define here as “simulated buffer” is referred to as “virtual buffer” in [15]. We chose “simulated buffer” to distinguish it from the virtual buffer concept used in the frame rate optimization chapters (chapters 5 and 6).
At the beginning of the $i$'th GOP, the simulated buffer level $V(i, k)$ is initialized as:

\[
V(i, 0) = \begin{cases} 
0 & i = 0 \\
V(i - 1, N_{i-1} - 1) & \text{otherwise}
\end{cases}
\]

where $N_{i-1}$ is the number of frames in the $(i-1)$'th GOP and the frame index $k = 0$ at the beginning of each GOP. After a frame has been encoded and before encoding the next frame, the simulated buffer $V(i, k)$ will update as:

\[
V(i, k) = V(i, k - 1) + b(i, k - 1) - \frac{R(i, k - 1)}{R_F} \quad k \in [2, N_i]
\]

where $b(i, k - 1)$ is the actual bits used by the previously encoded frame. To determine the bit allocation for a frame, Li et al make use of a target buffer level $S(i, k)$ that is initialized after the encoding of the first P-frame in the GOP as:

\[
S(i, 1) = V(i, 1)
\]

The target buffer level for each subsequent P-frame in the $i$'th GOP is then updated as:

\[
S(i, k + 1) = S(i, k) - \frac{S(i, 1)}{N_p(i) - 1}
\]

where $N_p(i)$ is the total number of P-frames in the $i$'th GOP. The frame target bits for frame $k$ using the simulated buffer mechanism is then defined as:

\[
\tilde{T}(i, k) = \frac{R(i, k)}{R_F} + \gamma \times (S(i, k) - V(i, k))
\]

where $\gamma$ is a constant that is typically set to 0.5. The above equation can be intuitively seen as the average frame size $\frac{R(i, k)}{R_F}$ adjusted by the difference between the target buffer level $S(i, k)$ and the current actual buffer level that is estimated using the simulated buffer $V(i, k)$. 
2.2.2.2. GOP-Based Bit Allocation

The GOP-based bit allocation mechanism of Li et al’s scheme is based on assuming each frame in the GOP takes up an average of $\frac{R(i,k)}{R_F}$ bits. So the $i$’th GOP would take up an average of $\frac{R(i,k)}{R_F} \times N_i$ bits. The available bits $B(i,k)$ are calculated on a per GOP basis, it is updated as:

$$B(i,k) = \begin{cases} \frac{R(i,k)}{R_F} \times N_i - V(i,k) & k = 0 \\ B(i,k - 1) + \frac{R(i,k) - R(i,k-1)}{R_F} \times (N_i - k) - b(i,k-1) & k \in [1, N_i - 1] \end{cases}$$

From the above equation, $B(i,k)$ is initialized with the average number of bits in a GOP $\frac{R(i,k)}{R_F} \times N_i$ with the estimated current encoder buffer level $V(i,k)$ subtracted from it. The available bits $B(i,k)$ is then reduced based on the actual bits used after encoding the previous frame ($b(i,k-1)$). The term $\frac{R(i,k) - R(i,k-1)}{R_F} \times (N_i - k)$ is used to capture any changes to the target sequence bitrate that might occur (e.g. bandwidth changes in a variable bitrate channel).

The frame target bits for frame $k$ using the GOP-based bit allocation mechanism is calculated as:

$$\hat{T}(i,k) = \frac{W_p(i,k-1) \times B(i,k)}{W_p(i,k-1) \times N_{p,r}(i)}$$

where $N_{p,r}(i)$ is the remaining number of P-frames in the $i$’th GOP and $W_p(i,k)$ is the estimated average complexity of P-frames and is calculated as:

$$W_p(i,k) = b(i,k) \times QP_p(i,k)$$

where $QP_p(i,k)$ is the QP chosen for the encoded frame $k$ in the $i$’th GOP.

2.2.2.3. Putting It All Together

Li et al’s scheme calculates the frame bit allocation using the simulated buffer mechanism, which yields $\hat{T}(i,k)$, as well as using the GOP-based bit allocation mechanism, which yields $\hat{T}(i,k)$. To determine the final frame bit allocation, Li et al used a weighted combination of both, calculated as:
(2.2.9) \[ T(i, k) = \beta \times \hat{T}(i, k) + (1 - \beta) \times \tilde{T}(i, k) \]

where \( \beta \) is a constant typically set to 0.5.

2.2.3. Quantization Parameter Selection

Li et al.’s QP selection step is largely based on MPEG-4’s scalable rate control proposed by Lee et al [17], which uses the rate-quantization model (R-Q model) proposed by Chiang and Zhang [16]. The idea of the R-Q model is that given a certain target number of bits as input, the model will determine a QP to use. If the target number of bits input is the frame target bits, e.g. calculated using (2.2.9), then the QP determined by the R-Q model is for a frame.

The R-Q model proposed by Chiang and Zhang [16] is based on the closed form solution for the rate distortion function derived in [18]:

\begin{equation}
R(D) = \ln \left( \frac{1}{\alpha D} \right)
\end{equation}

where \( R \) is the rate, \( D \) is the distortion and:

\[ D_{\text{min}} = 0 \quad D_{\text{max}} = \frac{1}{\alpha} \quad 0 < D < \frac{1}{\alpha} \]

Expanding (2.2.10) into a second order Taylor series yields:

\begin{equation}
R(D) = \left( \frac{1}{\alpha D} - 1 \right) - \frac{1}{2} \left( \frac{1}{\alpha D} - 1 \right)^2 + \ldots \\
= -\frac{3}{2} + \frac{2}{\alpha} D^{-1} - \frac{1}{2\alpha^2} D^{-2} + \ldots
\end{equation}

For notational convenience, we shorten \( T(i, k) \) to \( T(k) \). Then, using a similar form as the above equation, the R-Q model is estimated as:

\begin{equation}
T(k) = \frac{a_1}{QP(k)} + \frac{a_2}{QP^2(k)}
\end{equation}

where \( a_1 \) and \( a_2 \) are the first and second order model coefficients. \( a_1 \) and \( a_2 \) are calculated using a linear regression of the past \( n \) samples of \( T(k) \) and \( QP(k) \). That is, let the matrix \( X \) be:

\begin{equation}
X = \begin{bmatrix}
1 & \frac{1}{QP(0)} \\
1 & \frac{1}{QP(1)} \\
\vdots & \vdots \\
1 & \frac{1}{QP(n-1)}
\end{bmatrix}
\end{equation}
and the vector $Y$ be:

$$
(2.2.14) \quad Y = \begin{bmatrix}
QP(0) \times T(0) \\
\vdots \\
QP(n-1) \times T(n-1)
\end{bmatrix}
$$

where $n$ is the number of selected data samples taken from the past $n$ frames. Then the first and second order model coefficients $a_1$ and $a_2$ can be calculated as:

$$
(2.2.15) \quad \begin{bmatrix}
a_1 \\
a_2
\end{bmatrix} = (X^T X)^{-1} X^T Y
$$

Lee et al \cite{17} extended the original R-Q model by:

$$
(2.2.16) \quad \frac{T(k) - h(k)}{MAD(k)} = \frac{a_1}{QP(k)} + \frac{a_2}{QP^2(k)}
$$

where $h(k)$ is the bits used for headers and motion vectors for frame $k$ and $MAD(k)$ is a frame statistic that represents the mean absolute difference (MAD) of the motion compensated residual for the luminance component ($Y$ component). Excluding the header bits from the allocated frame bits $T(k)$ is done because it does not represent the actual pixel content of the frame. While the introduction of the frame statistic $MAD(k)$ into the R-Q model allows it to scale accordingly to the complexity of the frame. For example, a high complexity frame will increase the left-hand-side (LHS) of (2.2.16), causing the R-Q model to choose a larger QP, thus compressing the frame more. This behaviour is required as high complexity frames tend to use more bits to encode, so to ensure that the frame target bits are met, the R-Q model chooses to compress the frame more.

2.2.3.1. The Chicken and Egg Dilemma

One problem with H.264/AVC is that the MAD information is not available when rate control is being performed on the current frame $k$, this is coined the “chicken and egg” dilemma by Li et al \cite{19}. This occurs because the MAD information is not available to the rate controller until after the INTER and INTRA mode selection phase\footnote{Also known as the rate-distortion optimization (RDO) phase in the H.264/AVC standard \cite{19}.} which occurs after the QP has been determined by the rate controller\footnote{The rate control phase is done after the the mode selection phase in MPEG-4 \cite{17}.}. Furthermore, the mode selection phase itself requires...
the QP from the rate control phase. This creates an interdependency between
the rate control and mode selection phase.

Li et al circumvented this problem by predicting the MAD of the current
frame using linear regression. The predicted MAD for the current frame \( k \) is
calculated based on the following model:

\[
(2.2.17) \quad PMAD(k) = b_1 \times MAD(k - 1) + b_2
\]

where \( b_1 \) and \( b_2 \) are model coefficients predicted using a linear regression
of past MAD information. To incorporate the predicted MAD into the R-Q
model, Li et al modified (2.2.16) to:

\[
(2.2.18) \quad \frac{T(k) - h(k)}{PMAD(k)} = \frac{a_1}{QP(k)} + \frac{a_2}{QP^2(k)}
\]

2.3. H.264/AVC Rate Control Schemes

There are a number of schemes proposed to deal with the “chicken and egg”
dilemma discussed in the previous section, the most popular classes of schemes
are: predictive rate control, two pass rate control and image processing based
rate control.

2.3.1. Predictive Rate Control

Predictive rate control schemes make use of neighbouring frames to estimate
the current frame statistics. This is because neighbouring frames tend to have
a high correlation with the current frame.

Li et al [15, 19] falls into this category. However, their scheme has prob-
lems handling scene changes within a video [20]. Furthermore, the rate control
algorithm is only performed on P-frames, while I-frames and B-frames have
their QPs estimated based on the QPs calculated for P-frames without consid-
eration of the actual characteristics of the I-frames and B-frames [21]. This
makes it only ideal for videos with IPPP group-of-pictures (GOP) structure.

Leontaris and Tourapis [21] have attempted to fix this problem by in-
roducing a complexity measure for P-frames and made use of complexity
ratio parameters to determine the complexity of I-frames and B-frames. This
technique made the adopted rate control algorithm much more compliant to
the bit-rate constraints. However, sudden complexity changes such as scene
changes are not handled implicitly. Furthermore, the performance of the im-
proved algorithm is strongly correlated to the fixed parameters introduced,
which has to be tuned for various videos. Their approach also assumed the
total size of the sequence is available at the start, which may not always be
the case (e.g. for real-time video communication).

Other predictive schemes include the use of MAD ratio \[22\], which is the
ratio of the current predicted MAD with the average MAD of the previously
encoded frames, peak-signal-to-noise-ratio (PSNR) based prediction \[23\], both
PSNR and MAD ratio \[24\], the actual bits used in the previous frames \[25\]
and motion vector information of previous frames \[26\]. Additionally, Yuan
et al \[27\] and Dong and Ling \[28\] investigated on the choices of choosing the
neighbouring statistic for prediction. All these schemes tend to have a low
complexity, however, the performance suffers when there is a sudden change
in scene complexity.

2.3.2. Two Pass Rate Control

In two pass rate control schemes, the rate controller will encode a portion
of the video twice. The first encoding is used to gather preliminary frame
statistics for input into the rate controller while the second encoding is the
actual encoding process.

Liu et al \[20\] made use of the first encoding pass to estimate the MAD
information of the frame. The first pass fixes the INTER/INTRA modes to
INTER16x16 and INTRA16x16, which results in a coarse encoding and ap-
proximates the MAD for the current frame. At the second pass, it chooses
to use either the approximated MAD or the linearly predicted MAD from the
previous frame \[15\] depending on how close they estimated the actual MAD
in the past frame. After the estimated MAD is chosen, the scheme proceeds to
choose a QP. He and Wu \[29\] also follows this approach of using the first pass
to estimate certain frame statistics and the second pass to use the gathered
statistics in a model.

Ma et al \[30\] proposed using a two step QP update process at the mac-
roblock level. In the first pass, the QP is predicted based on the QP from the
previous frame and used as input into the mode selection phase. At the second
pass, the QP is updated by the rate controller. The scheme then determines
whether to use the updated QP based on its impact on the mode selection
phase calculated as a cost. If the cost is large, mode selection is performed
again, else it is skipped.

Two pass rate control schemes have the advantage of obtaining possibly
more accurate current frame statistics. This allows it to handle video sequences
with sudden large changes in scene complexity. Thus, it tends to produce
a more accurate bitrate output in general. However, two pass rate control
schemes tends to be more computationally expensive compared to single pass
schemes. Furthermore, the second pass cannot be performed in parallel as it requires information from the first pass. This introduces additional delay into the system.

2.3.3. Image Processing Based Rate Control

Image processing based rate control schemes make use of the raw pixel values of the frames to estimate the complexity of the frame.

Jing and Chau [31] made use of gradient information to determine the complexity of the frames and showed that there is a linear relationship between edge information and the bits required for coding the frame.

Lee et al [32] made use of the frame entropy to determine the complexity. Zhou et al [33] used the histogram of pixel differences (HOD) between two frames while Jing et al [34] made use of both HOD and the difference of pixel histograms.

All these approaches have shown that scene change can be detected reasonably well, however, none of them account for the differences between I-frames, P-frames and B-frames in their complexity measures. Normally, I-frames uses more bits than P-frames which in turn uses more bits than B-frames, but this relationship might change in a high motion video sequence due to the increased intra-coded macroblocks (MB) being introduced into the frame. A proper complexity measurement scheme should take into account of all these, which is discussed in chapter 4.

2.4. TCP-Friendly Rate Control

TCP-friendly rate control (TFRC) is an equation based congestion control protocol in that it uses a control equation to determine the maximum acceptable sending rate. TFRC is based on two main concepts: TCP-friendliness and smoothed rate changes. TFRC is designed to coexist with other TCP streams in the network by not being overly aggressive in competing for bandwidth.

The TFRC concepts presented here are used in chapter 3. The reference literature for TFRC are RFC 3448 [35] and Floyd et al [36].

2.4.1. TCP-Friendliness

TCP-friendliness is a protocol whose behaviour is similar to TCP. This is so that the protocol will be reasonably fair when competing with TCP streams for bandwidth, where “reasonably fair” is defined in RFC 3448 [35] as a sending rate within a factor of two of the sending rate of a TCP stream in identical conditions.
2.4. TCP-FRIENDLY RATE CONTROL

2.4.1. Slowstart
Like TCP, TFRC goes through a rate doubling slowstart phase that terminates once a congestion event occurs. To emulate TCP’s slowstart, it is noted that TCP’s ACK-clock mechanism limits TCP’s bandwidth overshoot to at most twice the bottleneck link bandwidth. Thus, TFRC limits its maximum slowstart increase rate, by setting its sending rate $T$ to:

$$T = \min(2T_{t-1}, 2T_{recv})$$

where $T_{t-1}$ is the previous sending rate and $T_{recv}$ is the measured throughput at the receiver-end. Equation (2.4.1) ensures that TFRC bandwidth overshoot is not worse than TCP’s overshoot in the slow-start phase.

2.4.1.2. Sending Rate Calculation
After the slowstart phase, TFRC’s sending rate is determined by a TCP response function [37]:

$$T = \frac{s}{RTT \sqrt{\frac{2p}{3}} + RTO \left(3\sqrt{\frac{3p}{8}}\right) p(1 + 32p^2)}$$

where $s$ is the packet size, $RTT$ is the mean round trip time, $RTO$ is the TCP retransmit timeout value and $p$ is the steady state loss event rate. A loss event is defined as one or more lost or marked packets within a single round trip time, where a marked packet refers to a congestion indication from Explicit Congestion Notification (ECN).

Using loss events instead of individual packet losses will cause TFRC to reduce its sending rate at most once for congestion notifications in one window of data. This emulates the behaviour of a typical TCP implementation as most TCP implementations tend to halve their congestion window once in response to several losses in a window.

2.4.2. Rate Smoothing
2.4.2.1. Loss Interval Calculation
The main mechanism TFRC uses to smooth rate fluctuations and determine its responsiveness to congestion changes is the average loss interval algorithm [36] used in calculating the loss event rate. The average loss interval algorithm calculates a weighted average of the last $n$ loss intervals, where a loss interval is defined as the number of packets between loss events. This algorithm reduces sudden changes in the calculated rate that could result from unrepresentative loss intervals and thus maintain a smoother rate transition. On the other
hand, the average loss interval algorithm is reasonably reactive to changes to
the available bandwidth.

The average loss interval \( s_{avg} \) is calculated from the previous \( n \) loss intervals as follows:

\[
s_{avg} = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i}
\]

(2.4.3)

where \( s_i \) represents the number of packets in the last \( i \)'th loss interval and the weights \( w_i \) are defined as:

\[
w_i = \begin{cases} 
1, & 1 \leq i \leq \frac{n}{2} \\
1 - \frac{i-n/2}{n/2+1}, & \frac{n}{2} < i \leq n
\end{cases}
\]

(2.4.4)

RFC 3448 recommends \( n = 8 \) \[35\] in practice. The loss event rate \( p \) is then calculated as:

\[
p = \frac{1}{s_{avg}}
\]

(2.4.5)

2.4.2.2. Round Trip Time

To avoid a strong reaction to changes in the round trip time (RTT), TFRC adopts a exponentially weighted moving average (EWMA) with a small weight (such as 0.1) to calculate its mean RTT. That is:

\[
RTT = \alpha R_0 + (1 - \alpha) RTT_{t-1}
\]

(2.4.6)

where \( \alpha \) is the EWMA weight, \( RTT_{t-1} \) is the previous RTT sample and \( R_0 \) is the most recent RTT sample. However, only doing EWMA with a small weight on the RTT calculation will cause oscillations in the sending rate. This is because, due to the slower reaction to changes in RTT, it causes TFRC to overshoot the available bandwidth, backoff and increase rate. To rectify this problem, TFRC further adjusts the sending rate calculated in (2.4.2) by setting the inter-packet interval as:

\[
t_{inter-packet} = \frac{s \sqrt{R_0}}{T \cdot M}
\]

(2.4.7)

where \( M \) is the average of the square-roots of the RTTs, this is calculated using EWMA with the same EWMA weight as (2.4.6).
2.4.2.3. Maximum Sending Rate Increase

TFRC caps the maximum sending rate increase after slowstart to:

\[ \delta_T = 1.2 \left( \sqrt{s_{\text{avg}}} + w_1 1.2 \sqrt{s_{\text{avg}}} - \sqrt{s_{\text{avg}}} \right) \]

where \( s_{\text{avg}} \) is from (2.4.3) and \( w_1 \) is calculated from (2.4.4). The above expression is derived from a simpler deterministic version of the TCP response function \[ T = \frac{\sqrt{1.5}}{\text{RTT} \sqrt{p}}. \]

2.5. Video Rate Control Over Networks

Video rate control over networks needs to consider both the network constraints and video quality. The main challenges are:

- Ensuring that the video is sent at a fair rate. The video streams may share the network with other traffic. A video stream using more than its fair share of rate may affect other streams. While this may result in an increase in video quality, it may lead to a decrease in the overall quality of the network.
- Adjusting to the time varying network bandwidth. The network bandwidth tends to fluctuate frequently. This can be due to traffic entering or exiting the network or existing traffic increasing or decreasing their rate. The schemes need to constantly adapt to the changing network conditions. Transmitting at a rate higher than available network bandwidth runs the risk of increasing the network congestion and rate unfairness. On the other hand, transmitting at a rate lower than the available bandwidth could lead to an underutilization of the network and it does not maximize the video quality.

There are three main classes of approaches to video rate control over networks: bandwidth probing, congestion modelling and TCP-friendly.

2.5.1. Bandwidth Probing Approaches

Bandwidth probing is a proactive way of estimating the available network bandwidth. Probe packets are sent from the sender and the bandwidth estimated using the receiving intervals of the probe packets.

Li et al [38] proposed a unique rate adaptation scheme that make use of network probing to build a network model using a Markov model. This network model is used to determine the video streaming rate as well as the amount of prebuffered video that will reduce the probability of an underflow event.
Huang et al. [39] uses one way jitter to determine the available bandwidth. To minimize the probing traffic, Huang et al uses the queuing delay propagation to predict whether the bottleneck is congested during the measured period and the captured traffic ratio which describes the relationship between the probing rate and available bandwidth. Other examples of approaches which attempts to manage probing traffic are [40, 41].

Balk et al. [42] proposed a video transport protocol which conservatively adapts the video based on the probed bandwidth. The authors also showed that their scheme competes fairly with TCP traffic depending on a certain choice of parameter. Setton et al. [43] used probing information to find reliable paths. Path diversity is then used to transmit the video to improve its error resiliency.

Bandwidth probing determines the available bandwidth fairly accurately. However, two open issues exists with these approaches:

- Operating fairly with other network traffic. Obtaining the available bandwidth does not mean the scheme will adapt to the rate fairly. How to determine a fair rate is still an open issue.
- Probe traffic management. The time varying nature of the network bandwidth means that probing needs to be done at regular intervals. Probing traffic tend to be additional overheads and need to be managed properly.

2.5.2. Congestion Model Based

Video rate control is also done in conjunction with congestion modeling. A congestion model can provide a rate to transmit the video at without causing network congestion. This rate is optimized against a video distortion function to acquire a distortion optimized rate for the video. It is shown that using this distortion optimized rate would result in distortion fairness for all the users.

Some of the congestion models studied in the literature are M/M/1 models [44], network utility maximization [45, 46, 47] and game theory [48, 49].

2.5.3. TCP-Friendly Approaches

TCP-friendly rate control approaches all attempt to produce a long term rate that is close enough to TCP traffic in similar conditions. As majority of the traffic are TCP-based, TCP-friendliness would ensure that the video stream would be able to compete fairly with other TCP flows. One such TCP-friendly approach is TCP-friendly Rate Control (TFRC) [35]. TFRC is an equation based congestion control technique for best effort networks that provides a
smoother throughput variation over time, making it more suitable for streaming multimedia applications. A number of TFRC-like approaches for streaming multimedia applications have emerged.

One TFRC variant by Tan and Zakhor \[50\] directly used the TFRC throughput in their rate control method to adjust the video source rate as did Zhang et al \[51\] and Shih et al \[52\].

Vieron and Guillemot \[53\] takes into account of the variable video packet sizes in the congestion calculation and synchronized the receiver feedback to the video frame rate. Kim et al \[54\] proposed another variant of TFRC that tradeoffs the smoothness of rate variation and network responsiveness by making use of weighted round-trip time (RTT) and retransmission time out (RTO).

Zhu et al \[55\], in particular, takes into account of buffer underflow by using a virtual buffer model to detect possible buffer underflow events. Should a possible buffer underflow event be detected, a higher amount of video data will be transmitted and thus temporarily violating TCP-friendliness. The TCP-friendliness violation would be compensated by sending the video at a rate lower than the TCP-friendly rate in the future. The idea of maintaining long term TCP-friendliness while possibly violating short term TCP-friendliness is also proposed by Yan et al \[56\].

However, none of these techniques factored in the video bit rate characteristic for the calculation of the TCP-friendly transmission rate. Furthermore, the aforementioned techniques tend to favor a smoothed rate more than a responsive rate, even though the difference between the two rates could possibly be exploited. This rate gap is discussed in chapter 3.

2.6. Prebuffering Video

Prebuffering video is the technique of delaying the initial video playout while the video is being streamed into the buffer. After a certain buffer level is reached, the video is then played. The goal of prebuffering is to maintain the video continuity when bandwidth fluctuates.

Prebuffering video data has been studied in \[57\] \[58\] \[59\]. These techniques are focused on calculating the correct amount of prebuffered data based on a mathematical model to avoid the occurrence of a buffer underflow once the video playout has started. However, these techniques do not take into account of the varying playout rates and the resulting reduction in prebuffering into their models. Furthermore, prebuffering only introduces unwanted delay into the system and is known to have an impact on the user perceived quality of the video \[60\].
2.7. Frame Rate Control

Frame rate control aims to adjust the frame generation rate of the sender and receiver such that the video quality and video continuity are maximized. In this thesis, the frame generation rate of the encoder is termed the encoder frame generation rate while the frame generation rate of the receiver is termed the playout frame rate. The two main classes of approaches in frame rate control are the prioritized media drop and adaptive media playout.

2.7.1. Prioritized Media Drop

Selectively dropping frames or specific video units from the video based on the network conditions forms the basis of prioritized media drop techniques. Initial work on prioritized media drop focuses on prioritizing video units to drop [61, 62, 63, 64, 65]. In particular, the seminal work by Krasic et al. [61] introduce the concept of mapping video units to certain priorities. When streaming the video, low priority units are dropped until the video rate meets the network rate. Tu et al. [62] extended this priority drop concept by implementing them on the intermediate nodes. For each stream, a rate vector and distortion matrix is sent. These are used along with a Lagrangian cost function together with the current buffer fullness level to find the optimum dropping pattern that fits the current network bandwidth. The proposed technique is shown to be of a low complexity and significant video quality improvements are reported.

More recent work on prioritized media drop is on packet scheduling. The main idea is to determine at each transmission opportunity at the sender whether to:

- send the current packet.
- discard the current packet to meet the current network bandwidth.
- send redundancy, either through retransmission of a previously transmitted packet or using forward error correction codes.

The decision is typically computed using a Markov decision process which attempts to minimize the video distortion by considering the cost of each action at each transmission opportunity. Examples of packet scheduling techniques are [66, 67, 68, 69, 70, 71].

The discussed prioritized media drop approaches all attempt to ensure that the video meets a certain target bitrate which is often derived from the network bandwidth. In contrast, for our proposal in chapters 5 and 6, we adjust the encoding frame rate to accelerate the number of frames being received at the decoder buffer when it is in danger of an underflow. The advantages of doing this is that it allows our scheme to be more adaptive to any sudden variations in the network conditions and at the same time it will not contribute
to any possible congestion in the network as it does not use up more network bandwidth. Furthermore, none of the prioritized media drop approaches jointly optimized the encoding frame rate and the playout frame rate to achieve video continuity.

2.7.2. **Adaptive Media Playout**

Adaptive media playout (AMP) is another class of techniques that is used to improve video continuity. The main idea behind AMP is to reduce the playout rate of the received media. This reduces the departure rate of the media in the decoder buffer and potentially allows the buffer to be filled up to safer levels.

AMP has been studied extensively for audio applications, \[72, 73, 74, 75, 76, 77\]. These audio AMP techniques depend on audio scaling techniques such as WSOLA \[78\], which allows audio to be scaled without changing its pitch. A recent work by Damnjanovic et al has shown that audio and video scaling can be done in real-time \[79\].

AMP for video involves scaling the frame intervals to slowdown or speedup the playout rate. The slowdown or speedup is triggered by a threshold set on the buffer occupancy. Once the buffer occupancy exceeds the threshold, AMP will slow down the playout rate and vice versa. There have been studies conducted on the dynamic adjustment of this threshold \[80, 81, 82, 83\]. The adjustment is normally based on the network condition, the poorer the network condition, the higher the threshold. These techniques mainly based the threshold on the buffer occupancy.

In chapter 3, we used the buffer occupancy as a penalty term in the optimization problem, which can be seen as equivalent to soft thresholding. In chapter 6, we attempt to improve over chapter 3 by recording the difference between the receiving frame rate and the playout rate into a virtual buffer and treat it as a penalty. We believe that using the rate differences is a better indication of the network condition compared to the buffer occupancy.

AMP has also been integrated into the design of packet schedulers \[84, 85, 86, 87, 88\]. These techniques tend to slowdown the playout rate for important video packets. This ensures that the more important video packets have a higher probability of meeting its deadline and thus avoid being dropped. We do not focus on packet scheduling in this thesis and our proposed framework could complement any packet scheduling scheme.

Another aspect of AMP being studied is the smoothness of transition between playout rate adjustment \[89, 90, 91\]. The goal of these approaches is to ensure that adjustments made to the playout rate is done as smoothly as possible so as to reduce any noticeable effects to the viewers. We do not focus
2. LITERATURE REVIEW

on rate smoothing in the current paper and again any smoothing scheme can be used within the framework.

Steinbach et al [92] and Kalman et al [13] both examined the trade-off between delay and buffer underflow using two state Markov models. Kalman et al further proposed AMPInitial, AMP-Robust and AMP-Live. AMP-Initial slows down the playout rate until the buffer reaches a certain target level, which produces a lower perceived initial buffering delay to the viewer. AMP-Robust slowdowns the playout rate if the current buffer occupancy falls below the target buffer level while AMP-Live slowdowns or speedups the playout rate to maintain the target buffer level.

These AMP approaches mainly examined only the effects of adjusting the playout rate, while our proposed approach aims to examine the effects on video quality and viewing delay based on the adjustment of both the encoder frame generation rate and the playout frame rate. This is discussed further in chapters [5] and [6].

2.8. Lyapunov Optimization

Lyapunov drift analysis is a mathematical tool that is used to provide methods for queue stability in networks. Recent works by Neely et al [93, 94] have extended Lyapunov drift analysis to provide performance optimization in addition to queue stability. With a buffer or queue to stabilize and some utility functions to maximize, one can make use of Neely et al’s methods to achieve that. We call Neely et al’s extended Lyapunov drift analysis as Lyapunov optimization in this thesis.

The Lyapunov optimization concepts presented here are used in chapters [5] and [6]. The reference literature for Lyapunov optimization are [93, 94, 14, 95]. In this section, we demonstrate some of the fundamental concepts in Lyapunov optimization.

2.8.1. System Model

\[
x(t) \xrightarrow{\text{buffer}} U(t) \xrightarrow{\text{output}} y(t)
\]

**Figure 2.6.** Illustration of the buffer model used.

To start off, one needs a system that works in discrete time slots and a buffer. Fig. 2.6 illustrates the buffer model we shall use in this section. At a certain time slot \( t \), the buffer level will be \( U(t) \) with \( x(t) \) units of data arriving and
y(t) units of data departing from the buffer. It can be seen that at each time slot, the buffer is updated as follows:

\[ U(t + 1) = [U(t) - y(t)]^+ + x(t) \]

where \([z]^+ = \max(z, 0)\), that is the buffer level \(U(t)\) is always non-negative. One of the goals of Lyapunov optimization is to ensure that the buffer stabilizes. The definition of stability used in this thesis is the one introduced by Neely et al [94]:

\[ \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E} \{ U(\tau) \} \leq \infty \]

That is, the buffer \(U(t)\) is stable if its long term average level is some finite value.

In addition to buffer stability, we also want to maximize a certain utility function \(g(y(t))\) of the departure rate \(y(t)\) over the time slots, specifically:

\[ \text{Maximize: } \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E} \{ g(y(\tau)) \} \]

The above equation intuitively means that we want to maximize the long term average of the utility function \(g(y(t))\).

2.8.2. One-Step Lyapunov Drift

The following lemma is an intermediate step towards proving the stability of the buffer \(U(t)\).

**Lemma 2.1.** Given that the buffer \(U(t)\) is updated as:

\[ U(t + 1) = [U(t) - y(t)]^+ + x(t) \]

where buffer arrival process \(x(t)\) is i.i.d at every timeslot \(t\) and \(y(t)\) is the buffer departure rate. Furthermore, both \(x(t)\) and \(y(t)\) are upper bounded as \(x(t) \leq x_{\max}\) and \(y(t) \leq y_{\max}\) respectively. Let a Lyapunov function be defined for \(U(t)\) as:

\[ L(U(t)) = \frac{U^2(t)}{2} \]

Let a one-step Lyapunov drift be defined as:
(2.8.6) \[ \Delta(U(t)) = \mathbb{E} \{ L(U(t+1)) - L(U(t)) | U(t) \} \]

And let \( g(y(t)) \) be a utility function of \( y(t) \), then the conditional Lyapunov drift satisfies:

(2.8.7) \[ \Delta(U(t)) - V \mathbb{E} \{ g(x(t)) | U(t) \} \leq B - U(t) \mathbb{E} \{ y(t) - x(t) | U(t) \} - V \mathbb{E} \{ g(y(t)) | U(t) \} \]

Where \( B = \frac{1}{2}(x_{\text{max}}^2 + y_{\text{max}}^2) \).

**Proof.** By squaring (2.8.4), dividing it by 2 and rearranging its terms, we get:

\[
\frac{U^2(t+1)}{2} - \frac{U^2(t)}{2} \leq \frac{x^2(t) + y^2(t)}{2} - U(t)(y(t) - x(t)) \\
\leq \frac{x_{\text{max}}^2 + y_{\text{max}}^2}{2} - U(t)(y(t) - x(t)) \\
\leq B - U(t)(y(t) - x(t))
\]

(2.8.8)

By taking conditional expectations of (2.8.8) with respect to \( U(t) \), then applying (2.8.5) and (2.8.6) in (2.8.8) yields:

(2.8.9) \[ \Delta(U(t)) \leq B - U(t) \mathbb{E} \{ y(t) - x(t) | U(t) \} \]

Subtracting the term \(-V \mathbb{E} \{ g(y(t)) | U(t) \}\) from both sides of (2.8.9) would produce (2.8.7).

Lemma 2.1 introduces the Lyapunov function and the one-step Lyapunov drift into the buffer update equation (2.8.4). The Lyapunov function (2.8.5) is a non-negative function that can be viewed as the 'energy' function of the buffer \( U(t) \). While the one-step Lyapunov drift can be seen as the change in energy in one time slot.

The key idea of lemma 2.1 is to manipulate the buffer update equation (2.8.4) such that its left-hand side (LHS) represents the one-step conditional drift (2.8.7) and the right-hand side (RHS) represents the upper bound of the drift.

The subtraction of the term \(-V \mathbb{E} \{ g(y(t)) | U(t) \}\) from both sides enables performance optimization. This is because buffer stability and utility maximization can be achieved by designing policies that can minimize the RHS of (2.8.7). One way to obtain such a policy is to group the terms with \( y(t) \) together on the RHS of (2.8.7). Doing this converts (2.8.7) to:
2.8. LYAPUNOV OPTIMIZATION

\[ \Delta(U(t)) - V \mathbb{E} \{ g(x(t)) | U(t) \} \leq B + U(t) \mathbb{E} \{ x(t) | U(t) \} - \mathbb{E} \{ V g(y(t)) + y(t) U(t) | U(t) \} \]

The policy to minimize is the last term in the RHS of the above equation: \(- \mathbb{E} \{ V g(y(t)) + y(t) U(t) | U(t) \}\). Note that minimizing this is the same as maximizing \( V g(y(t)) + y(t) U(t) \) for all time slots \( t \).

2.8.3. Performance Bounds

After (2.8.7) is obtained, we can prove the stability of the buffer \( U(t) \) by showing that there exists a stationary randomized policy that ensures \( \mathbb{E} \{ y(t) - x(t) | U(t) \} = \epsilon \), where \( \epsilon \) is some positive constant. One proposed approach by Neely et al [94] is to make use of a policy that is designed in such a way that chooses the departure rate such that \( \mathbb{E} \{ y(t) | U(t) \} \geq x(t) + \epsilon \). Once this is shown, the following lemma proves the stability of \( U(t) \) as well as the performance bounds of the utility function.

**Lemma 2.2.** Let \( V, B \) and \( \epsilon \) be some positive constants, \( y(t) \) be some variable to be optimized and \( g(y(t)) \) is some concave utility function of \( y(t) \). Assume that utility function is bounded as \( g(y(t)) \leq G_{\text{max}} \) and the optimal utility of \( g(y(t)) \) is \( g^* \). If the 1-step conditional Lyapunov drift (define in (2.8.7)) for all timeslots \( t \) satisfies:

\[ \Delta(U(t)) - V \mathbb{E} \{ g(y(t)) | U(t) \} \leq B - \epsilon U(t) - V g^* \]

then the system is stable and the time average of buffer \( U(t) \) is bounded as:

\[ \limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E} \{ U(\tau) \} \leq \frac{B + V G_{\text{max}}}{\epsilon} \]

and the time average utility is bounded as:

\[ \liminf_{M \to \infty} g \left( \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E} \{ y(\tau) \} \right) \geq g^* - \frac{B}{V} \]

**Proof.** Assume that (2.8.10) holds. By using the drift definition (2.8.6), taking expectations and using the law of iterated expectations on (2.8.10) would result in:

\[ \mathbb{E} \{ L(U(t+1)) \} - \mathbb{E} \{ L(U(t)) \} - V \mathbb{E} \{ g(y(t)) \} \leq B - \epsilon \mathbb{E} \{ U(t) \} - V g^* \]
By summing over the timeslots \( \tau \in [0..M-1] \), using telescoping series and the non-negativity of \( L(U(t)) \) which allows us to drop the term \( \mathbb{E}\{L(U(M))\} \), we get:

\[
(2.8.14) \quad -\mathbb{E}\{L(U(0))\} - V \sum_{\tau=0}^{M-1} \mathbb{E}\{g(y(\tau))\} \leq MB - \epsilon \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} - MV g^*
\]

We first prove the buffer bound \( (2.8.11) \). Using the utility function bound \( g(x(t)) \leq G_{\text{max}} \) in \( (2.8.14) \), dividing it by \( M\epsilon \) and rearranging the terms produces:

\[
(2.8.15) \quad \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \leq \frac{B + VG_{\text{max}}}{\epsilon} + \frac{\mathbb{E}\{L(U(0))\}}{M\epsilon}
\]

Taking the limits of the inequality as \( M \to \infty \) would yield the buffer bound \( (2.8.11) \).

We next prove the utility bound \( (2.8.12) \). We divide \( (2.8.14) \) by \( MV \) and rearrange its terms:

\[
(2.8.16) \quad \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(y(\tau))\} \geq g^* - \frac{B}{V} - \frac{\mathbb{E}\{L(U(0))\}}{MV}
\]

Note that the concavity of the utility function \( g(y(t)) \) together with Jensen’s inequality would mean that:

\[
(2.8.17) \quad \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(y(\tau))\} \leq g\left(\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{y(\tau)\}\right)
\]

Using \( (2.8.17) \) in \( (2.8.16) \) and taking the limits as \( M \to \infty \) would yield the utility bound \( (2.8.12) \). \( \square \)

Additionally, Neely et al also show that with buffer stability and an upper bound on the departure rate, the long term average of the departure rate \( y(t) \) will be the same as or more than the departure rate \( x(t) \). This is proved in the following lemma.

**Lemma 2.3.** If the buffer \( U(t) \) is strongly stable with a buffer arrival process \( x(t) \) and a buffer departure rate \( y(t) \) that is upper bounded as \( y(t) \leq \bar{y}_{\text{max}} \) for all time slots \( t \), then:

\[
(2.8.18) \quad \liminf_{t \to \infty} (\bar{y}(t) - \bar{x}(t)) \geq 0
\]
where:

$$\bar{y}(t) = \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\}$$

$$\bar{x}(t) = \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{x(\tau)\}$$

**Proof.** Suppose that the following inequality holds:

(2.8.19) \[ \limsup_{t \to \infty} \frac{\mathbb{E}\{U(t)\}}{t} = 0 \]

Note that for any time \( t \), we have:

(2.8.20) \[ U(t) \geq U(0) + \sum_{\tau=0}^{t-1} x(\tau) - \sum_{\tau=0}^{t-1} y(\tau) \]

Dividing (2.8.20) by \( t \) and taking expectation yields:

(2.8.21) \[ \frac{\mathbb{E}\{U(t)\}}{t} \geq \frac{\mathbb{E}\{U(0)\}}{t} + \bar{x}(t) - \bar{y}(t) \]

Taking the limsup of both sides and using (2.8.19) yields:

(2.8.22) \[ 0 \geq \limsup_{t \to \infty} (\bar{x}(t) - \bar{y}(t)) \]

Thus, it suffices to prove that (2.8.19) is satisfied whenever the conditions of the lemma hold. To show this, suppose (2.8.19) does not hold, so there exists a value \( \epsilon > 0 \) and a subsequence of times \( \{t_n\} \) such that \( t_n \to \infty \) and \( \mathbb{E}\{U(t)\}/t_n \geq \epsilon \). This is a contradiction as it implies that (2.8.2) is not met and \( U(t) \) is not stable.

Let \( M = \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U(t)\} \), now choose any arbitrarily large constant \( V \) such that \( V > M \) and let \( T_n \) denote the number of timeslots after time \( t_n \) until \( \mathbb{E}\{U(t)\} \) crosses below the \( V \) threshold. If \( \mathbb{E}\{U(t)\} < V \), then we define \( T_n \) to be 0. Note that \( T_n \) is finite for all \( n \), otherwise \( \mathbb{E}\{U(t)\} \geq V \) for all \( t \geq t_n \), which contradicts the fact that the limsup since average expected value of \( U(t) \) is equal to \( M \). Because the buffer departure rate \( y(t) \) is bounded by \( y_{\text{max}} \), we have for any time \( t > t_n \):
\[ \mathbb{E} \{ U(t) \} \geq \mathbb{E} \{ U(t_n) \} - y_{\text{max}}(t - t_n) \]

(2.8.23) \[ \geq \epsilon t_n - y_{\text{max}}(t - t_n) \]

Hence, \( \mathbb{E} \{ U(t) \} \geq V \) whenever \((t - t_n) \leq (\epsilon - V)/y_{\text{max}}\). It follows that \( T_n \geq (\epsilon t_n - V)/y_{\text{max}} \). Therefore:

(2.8.24) \[ \frac{T_n}{t_n} \geq \frac{\epsilon}{y_{\text{max}}} - \frac{V}{t_n y_{\text{max}}} \]

This means that \( \liminf_{t \to \infty} T_n/t_n \geq \epsilon/y_{\text{max}} \), which implies that \( \limsup_{t \to \infty} t_n/T_n \leq y_{\text{max}}/\epsilon \). However, note that by definition:

(2.8.25) \[ \frac{1}{t_n + T_n} \sum_{\tau=0}^{t_n + T_n - 1} \mathbb{E} \{ U(\tau) \} \geq V \frac{T_n}{t_n + T_n} \]

(2.8.26) \[ = V \frac{1}{t_n/T_n + 1} \]

Taking the limits of the above inequality, we get:

(2.8.27) \[ M \geq V \frac{1}{y_{\text{max}}/\epsilon + 1} \]

However, this inequality holds for arbitrarily large values of \( V \), contradicting the fact that \( M \) is finite. \( \square \)
CHAPTER 3

Video TCP Friendly Rate Control

3.1. Introduction

In this chapter, we address the video rate control problem by designing a joint source rate and congestion control method that makes use of H.264/AVC rate control to improve the frame quality of the video, while ensuring the video stream is TCP-friendly via interactions with the transport layer congestion control. A TCP-friendly stream behaves in a similar manner to a TCP stream in identical conditions but tends to adapt its rate more smoothly. This allows the stream to compete fairly for bandwidth with other TCP clients in the network.

We focus on real-time video streaming here, this is where the video is encoded just before it is streamed. Before the video is encoded, the encoder needs to set a target bitrate and ensure that the video is compressed to meet that target bitrate. A typical approach to this is to set the target bitrate to the sending rate returned by the transport layer [96].

This means that the video bitrate characteristics need to be similar to the sending rate characteristics. However, this may have a detrimental effect on the video quality. A reason for this is that the video bit rate tends to vary according to the complexity of the frame data in a small time scale, for example an I-frame would be more complex compared to a P-frame as it results in more bits after compression. The same also applies to scene changes and high motion scenes in a video sequence as they tend to incur a higher prediction error which results in a lower compression efficiency. Thus, a typical video bit rate tends to have occasional ‘pulses’ in a smaller time scale, see fig. 3.1. Furthermore, current TCP-friendly video streaming approaches [55, 50, 53, 54, 52, 51] do not factor in this video bit rate characteristic for the calculation of the TCP-friendly sending rate and it has been shown that the TCP-friendly rate could affect the video quality [56].
3. VIDEO TCP FRIENDLY RATE CONTROL

Figure 3.1. Frame sizes of the foreman sequence encoded using H.264. Group-of-pictures (GOP) size of 50 frames, GOP structure IPPP.

We approach this by proposing a joint source rate and congestion control method called video TCP-friendly rate control (VTFRC) that makes use of the frame complexity to determine a TCP-friendly sending rate. The advantage of this approach is that the frame complexity models the video bit rate characteristics more accurately \cite{20, 15, 32, 31}. Furthermore, by incorporating the video bit rate characteristics into the TCP-friendly sending rate, the video quality can be improved while maintaining TCP-friendliness.

The contribution of this chapter is the design of a joint source rate and congestion control method that incorporates the video bit rate characteristic using frame complexity into the TCP-friendly rate calculation. We show that VTFRC improves video quality while competing fairly and in a non-aggressive manner with other TCP traffic when compared to the existing scheme.

This chapter is organised as follows:

Section 3.2 discusses the design choices of our joint source rate and congestion control method. Section 3.2.3.1 explains how the rate gap is calculated. Section 3.3 shows how to incorporate the frame complexity into the sending rate. Section 3.4 evaluates the performance of our method. Section 3.5 concludes this chapter.
3.2. Design Overview

3.2.1. Congestion Control Choice

To produce a TCP-friendly bit stream, the congestion control technique in the transport layer needs to be TCP-friendly. A widely used congestion control technique is additive increase, multiplicative decrease (AIMD) that is provided by the TCP [9, 36]. AIMD works by halving the sending rate in response to a single congestion event. This causes abrupt rate changes and is shown to have a negative impact on the user perceived quality of the video stream [10]. In contrast, TCP-friendly rate control (TFRC) adopts an equation based congestion control that provides sending rate based on the history of loss intervals. Where a loss interval is defined as the number of packets between loss events while a loss event consists one or more packets dropped within a single round-trip time (RTT) [35]. This causes congestion responses from TFRC to produce smoother rate changes. Furthermore, TFRC rate equation is derived from the TCP response function, which helps to ensure that the TFRC stream is TCP-friendly [36]. These characteristics makes TFRC more suitable than TCP as a congestion control mechanism for video streaming.

3.2.2. Cross-Layer Interaction in VTFRC

To ensure a TCP-friendly stream as well as to improve the frame quality of the video, in our proposed approach, VTFRC incorporates the video bit rate characteristic into the calculation of the TCP-friendly sending rate by using the frame complexity. As was discussed above, this is because the video bit rate tends to vary according to the complexity of the frame data.

To approach this, we first observe that the TFRC rate returned from the TFRC protocol is less responsive to the congestion signals in the network due to its smooth changing properties. This possibly means that the TFRC rate may be less than the current TCP rate since it takes some time to adapt to a higher rate [36]. We define the current TCP rate as the instantaneous TCP rate in this chapter. Therefore, a possible rate gap exists between the TFRC rate and the instantaneous TCP rate.

Using the frame complexity from the rate controller and the rate gap, VTFRC will opportunistically encode a frame at a higher bit rate and at the same time perform a rate negotiation with the TFRC protocol to possibly transmit at a higher bit rate in order to match closely to the video bit rate characteristic. This helps to improve the frame quality of the video while ensuring that the video stream is TCP-friendly.

1See also section 2.4 of chapter 2 for more details.
3.2.3. Architecture of VTFRC

Fig. 3.2 shows the protocol stack of VTFRC. The three main components used in VTFRC are: the encoder, the TFRC protocol and the rate negotiator. The encoder in the application layer calculates the current frame complexity and sends it to the rate negotiator before the current video frame is encoded. Simultaneously, the TFRC protocol at the transport layer determines the TFRC rate as well as the instantaneous TCP rate and send it to the rate negotiator.

Based on these inputs from the encoder and the TFRC protocol, the rate negotiator will first calculate the rate gap using the TFRC rate and instantaneous TCP rate. Then, with the rate gap and frame complexity, the rate negotiator will calculate the VTFRC rate. The VTFRC rate will be feedback to both the encoder and the TFRC protocol. The encoder will use the VTFRC rate as its target bitrate for encoding the frame, while the TFRC protocol will use the VTFRC rate as the new sending rate.

3.2.3.1. Rate Gap Calculation

Fig. 3.3 illustrates how the rate gap is calculated, it can be seen that the rate gap is essentially the difference that might exist between the TFRC rate and
the instantaneous rate. The TFRC rate $R_{tfrc}(t)$ at time $t$ is calculated as per RFC 3448\(^2\) with the average packet size used in the calculation as the video packets tend to be of variable sizes.

The instantaneous TCP rate $R_{inst}(t)$ is a rate that is more indicative of the current congestion situation than the TFRC rate, thus, it is more responsive to the congestion signals in the network. To calculate the instantaneous TCP rate $R_{inst}(t)$, we look at the theoretical TCP upper-bound \(^3\):

$$R_{inst}(t) = \frac{s_{pkt}}{RTT(t)\sqrt{\frac{2p(t)}{3}} + RTO\left(3\sqrt{\frac{3p(t)}{8}}\right)p(t)(1 + 32p(t)^2)}$$ \(^{(3.2.1)}\)

Where $s_{pkt}$ is the average packet size, $RTT(t)$ is the receiver reported round-trip time (RTT) as at time $t$, $RTO$ is the TCP retransmit timeout value and $p(t)$ is the loss event rate. We first note that the value for $RTT(t)$ used in the TFRC rate is smoothed using the exponentially weighted moving average (EWMA) \(^3\) for the instantaneous TCP rate, $RTT(t)$ represents the current measured RTT.

We also observe that much of the smoothing effect of TFRC comes from the average loss interval calculation for the loss event rate $p(t)$ \(^3\). The main idea of the average loss interval calculation is to take a weighted average of the past history of loss intervals to determine a smoothed loss event rate \(^4\). For the instantaneous TCP rate, we reduce the number of loss intervals used for calculation to two, compared to eight recommended in \(^3\), so the average loss interval for time $t$ becomes:

$$s_{avg}(t) = \frac{1}{2} \sum_{i=1}^{2} s(t - i)$$ \(^{(3.2.2)}\)

Where $s(t)$ is the loss interval at time $t$. The corresponding loss event rate $p(t)$ is then calculated as:

$$p(t) = \frac{1}{s_{avg}(t)}$$ \(^{(3.2.3)}\)

To summarize the discussions thus far, the TFRC rate $R_{tfrc}(t)$ is the rate returned from the TFRC protocol. The instantaneous TCP rate $R_{inst}(t)$ like the TFRC rate, is derived from the TCP response function \(^{(3.2.1)}\). However,

\(^2\)See section 2.4.1.2 in chapter 2 for more details on how the TFRC rate is calculated.

\(^3\)See also section 2.4.2.2 of chapter 2.

\(^4\)see also section 2.4.2 of chapter 2.
3. VIDEO TCP FRIENDLY RATE CONTROL

to obtain a more responsive rate, the instantaneous TCP rate reduces the smoothing effect used in the TFRC rate. This is done by: 1. using the current measured RTT sample for $RTT(t)$ and 2. calculating the loss event rate $p(t)$ using only the two most recent loss interval samples.

The rate gap $R_{gap}(t)$ is the possible difference between the TFRC rate $R_{tfrc}(t)$ and the instantaneous TCP rate $R_{inst}(t)$, calculated as:

\[
R_{gap}(t) = R_{inst}(t) - R_{tfrc}(t)
\]

$R_{gap}(t)$ exhibits two properties:

- It may be positive or negative depending on the congestion situation in the network. Thus, VTFRC is opportunistic in the sense that when it detects there is a positive $R_{gap}(t)$, it will attempt to make use of it to incorporate the video bit rate characteristic into the sending rate.
- It is defined using $R_{tfrc}(t)$ and $R_{inst}(t)$, these are both defined using the TCP response function (3.2.1). This makes any rate that falls within the rate gap region to be TCP-friendly as well.

3.2.4. VTFRC Design Disadvantages

The cross-layer nature of VTFRC brings about some disadvantages to its design, namely:

1. VTFRC is closely tied to the TFRC protocol. This would mean that VTFRC will suffer any problems that an equation based congestion control protocol would have, such as those suggested by Rhee and Xu [97].

2. The TFRC protocol needs to be modified. Modifying TFRC to implement VTFRC requires modifying both the sender and receiver transport layer algorithm. This may be difficult especially since TFRC is adopted into a standard [98]. However, a possible way to circumvent this is to implement the entire VTFRC as an application on top of UDP.

3. The encoder needs to be modified to calculate the frame complexity. This could be easily done as the encoder sits on the application layer and can be modified in a straightforward way. Also, the frame complexity can be easily determined as there are a number of ways proposed to do this, see for example the complexity based rate control scheme proposed in chapter [4].
3.3. Rate Negotiation

3.3.1. Frame Complexity

We now look at how the encoder can calculate the frame complexity. In general, the more complex the video content is, the higher the bit rate required to encode the video. The frame complexity should ideally give an accurate estimate of how much more bit rate is required to encode the frame.

Frame complexity is typically used in encoder rate control schemes to ensure the video meets the target bit rate [20, 15, 32, 31]. One type of frame complexity used is the mean absolute difference (MAD) of the frame, where MAD is defined here as the difference between the encoded frame and the original frame.

However, for H.264/AVC encoding, the MAD of a frame is difficult to obtain before the frame is encoded. This issue has been dealt with by Li et al [15] by estimating the MAD of the current frame $k$ based on a linear prediction of the MAD of the previous frame $k-1$. That is:

$$MAD_{pred}(k) = a_1 \cdot MAD_{actual}(k-1) + a_2$$

where $a_1$ and $a_2$ are the linear model parameters that are updated via linear regression as described in [15], and $MAD_{actual}(k)$ is the MAD of frame $k$ after it has been encoded. To determine the frame complexity from its MAD value, we make use of Jiang et al’s MAD ratio [22], so the the complexity value of frame $k$ is defined as:

$$C(k) = \frac{MAD_{pred}(k)}{\frac{1}{n} \sum_{i=1}^{n} MAD_{actual}(k-i)} - 1$$

Equation (3.3.2) can be intuitively seen as comparing how much more complex the current frame is compared with the past $n$ frames. Jiang et al has shown that (3.3.2) could be use to measure the frame complexity reasonably well [22]. The number of past frames $n$ is updated based on a sliding-window data-point selection as described in [17] after the frame complexity $C(k)$ has been computed using (3.3.2), algorithm 1 describes this.

**Algorithm 1** Sliding-window calculation.

```markdown
if C(k-1) > C(k)
    n = floor (C(k)/C(k-1) * MAX_SLIDING_WINDOW)
else
    n = floor (C(k-1)/C(k) * MAX_SLIDING_WINDOW)
```
The constant MAX_SLIDING_WINDOW is set to 20 in our implementation, this is taken from [17]. Intuitively, the sliding-window data-point selection determines the window size based on the frame complexity trend. If the trend suggests a large change in frame complexities then a smaller more recent set of data points will be used. Otherwise, if the video displays very little frame complexity variation then a larger set of data points will be used.

### 3.3.2. VTFRC Rate Calculation

At this point, we have shown how to calculate the rate gap \( R_{\text{gap}}(t) \) using the TFRC rate \( R_{\text{tfrc}}(t) \) and the instantaneous TCP rate \( R_{\text{inst}}(t) \). We show that the rate gap \( R_{\text{gap}}(t) \) can be opportunistically exploited when it is positive and any rate that falls within the rate gap region is TCP-friendly. We also show how the frame complexity can be calculated before the frame is encoded. We show that this can be achieve by linearly predicting the MAD of the previous frame and using the MAD ratio to determine the frame complexity.

The next step is to determine how much of the rate gap to use. Using the whole rate gap only ensures that the sending rate follows a TCP rate characteristic not the video bit rate characteristic since it does not consider the video content. Furthermore, VTFRC opportunistically uses the additional rate in the rate gap, if the whole rate gap is used all the time the long term average sending rate will increase. This makes VTFRC more aggressive sender and has a knock-on effect of causing more network congestion. This does not happen to TCP since it halves its sending rate once a congestion signal is detected, thus ensuring its long term average sending rate is not aggressive.

We approach this issue by using the frame complexity to determine the amount of rate gap to use. This also allows us to determine the target bit rate for the current frame and the VTFRC sending rate. Specifically, given that frame \( k \) is about to be encoded at time \( t \), the additional rate \( R_{\text{add}}(k,t) \) that can possibly be used for sending and encoding is:

\[
(3.3.3) \quad R_{\text{add}}(k,t) = C(k) R_{\text{gap}}(t)
\]

Note that \( R_{\text{add}}(k) \) here is updated at the start of every frame period (i.e. just before the frame is being encoded) and thus \( R_{\text{gap}}(t) \) here is the rate gap at the start of the frame period. Given that the encoder is currently beginning to encode frame \( k \) at time \( t \), the target bit rate \( R_{\text{tgt}}(k,t) \) for the current frame in the encoder is then calculated by:

\[
(3.3.4) \quad R_{\text{tgt}}(k,t) = R_{\text{tfrc}}(t) + R_{\text{add}}(k,t)
\]
3.4. EXPERIMENTAL RESULTS

The above equation is calculated at the encoder, so $R_{tfrc}(t)$ refers to the TFRC rate feedback from the TFRC protocol in the transport layer. Simultaneously, at the transport layer, given the current time $t$ and that the last frame period for frame $k$ occurred at time $t - i$, the VTFRC rate to transmit at is:

$$R_{vtfrc}(t) = \min(R_{tfrc}(t) + R_{add}(k, t - i), R_{inst}(t))$$

The bound in (3.3.5) is to ensure that the VTFRC sending rate $R_{vtfrc}(t)$ does not exceed the current instantaneous TCP rate $R_{inst}(t)$ and maintain TCP-friendliness.

The idea behind of this scheme is that the frame complexity can provide an estimate of how much more rate is required, which helps to incorporate the video bit rate characteristics into the sending rate. If the frame is more complex, this would result in a higher target bit rate set by (3.3.4). This improves the quality of the frame. The VTFRC sending rate set in (3.3.5) will then increase correspondingly to ensure that the higher bit rate requirement of the frame is met.

3.4. Experimental Results

3.4.1. Setup

We implemented our proposed method in the H.264/AVC Joint-Module (JM) 12.2 reference software [99]. The encoder had rate-distortion (RD) optimizations disabled and the rate control mode set to RC_MODE_0, which is the original JM rate control method proposed by Li et al [15]. RC_MODE_0 provides the predicted MAD for (3.3.1). We set a group-of-pictures (GOP) size of 50 frames with a GOP structure of IPPP and the number of reference frames for motion estimation is set to 1.

The maximum packet size generated by the encoder is set by limiting the maximum slice size to 1400 bytes. The JM 12.2 reference software puts each slice into one packet and currently does not combine multiple slices to meet a fixed packet size, thus, the packet sizes generated are still highly variable (which is why the average packet size was used in the TFRC rate calculation). Five sequences were used for testing, they are: container, foreman, grandma, mother-daughter and salesman, table 3.1 shows the number of frames in each sequence. All of the sequences tested were of QCIF size (176x144). A frame rate of 20 frames per seconds is set for the experiments.

Our experiments were performed on a small test bed with two computers at the endpoints (fig. 3.4) emulating the senders and receivers. The middle
computer acts as the core of the network with dummynet pipes to emulate the bottleneck bandwidth and RTT. We set the bottleneck bandwidth to be 1.5Mbits/s and the RTT to be 160ms. The buffer overflow of the dummynet queue is set to 100 packets and the simulated loss rate is set to 0%, i.e. any loss generated will be due to queue buffer overflow.

An additional three TCP streams were generated at the sender using iperf with the receiver as the destination. These three streams therefore compete for buffer space at the core of the network. The TCP streams were given an initial 5 second delay before transmitting, while the encoder transmits right from the start.

Our proposed method (VTFRC) is compared to a setup of the original JM rate control method and the TFRC protocol which we call JM-TFRC here, fig. shows the protocol stack of JM-TFRC. In JM-TFRC, the target bit rate for each frame and the sending rate is set to the TFRC rate $R_{tfrc}(t)$. The differences between JM-TFRC and VTFRC is that JM-TFRC does not exploit any possible rate gap and it does not take the frame complexity into account.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>300</td>
</tr>
<tr>
<td>foreman</td>
<td>400</td>
</tr>
<tr>
<td>grandma</td>
<td>870</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>961</td>
</tr>
<tr>
<td>salesman</td>
<td>449</td>
</tr>
</tbody>
</table>

Table 3.1. Number of frames for each sequence.

Figure 3.4. Testbed used.
3.4. EXPERIMENTAL RESULTS

3.4.2. TCP-Friendliness Evaluation

We first check that VTFRC emits TCP-friendly flows. To do this, we used the TCP-friendliness ratio \( F \) metric by Padhye et al \[37\]. In our case, given the long term average throughput of the three competing TCP clients, \( T_{C1} \), \( T_{C2} \) and \( T_{C3} \). We calculate the average competing TCP client throughput by:

\[
T_C = \frac{1}{3} \sum_{i=1}^{3} T_{Ci}
\]

(3.4.1)

Given that the long term average throughput of VTFRC is \( T_V \), the TCP-friendliness ratio is:

\[
F = \frac{T_V}{T_C}
\]

(3.4.2)

The TCP-friendliness ratio \( F \) should ideally be one, i.e. the scheme has the same long term throughput as the TCP clients. Any \( F \) value less than two would mean the scheme is TCP-friendly. This is because RFC 3448 \[35\] states that a flow competing with TCP is reasonably fair when its sending rate is within a factor of two of the sending rate of a TCP flow in the same conditions. However, any \( F \) value more than one would mean that the scheme is more aggressive in sending than a typical TCP client.

Additionally, as was observed by Zhu et al \[55\], the multimedia flow here involves packets of variable sizes and the fairness of the flow is related to the packet size, thus, we calculate the sending rate in packets per seconds to remove this bias. The results for TCP-friendliness are shown in table 3.2. The throughputs over time for container and foreman are shown in fig. 3.6 and fig. 3.7 respectively.
3. VIDEO TCP FRIENDLY RATE CONTROL

<table>
<thead>
<tr>
<th>Sequence</th>
<th>VTFRC ($F$)</th>
<th>JM-TFRC ($F$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>0.92</td>
<td>0.80</td>
</tr>
<tr>
<td>foreman</td>
<td>1.07</td>
<td>0.97</td>
</tr>
<tr>
<td>grandma</td>
<td>0.78</td>
<td>0.57</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>1.02</td>
<td>0.80</td>
</tr>
<tr>
<td>saleman</td>
<td>1.02</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 3.2. Results of TCP-friendliness ratio ($F$) for VTFRC and JM-TFRC

**Figure 3.6.** Throughput over time for container.

**Figure 3.7.** Throughput over time for foreman.
3.4. EXPERIMENTAL RESULTS

It can be seen from the results that VTFRC is a slightly more aggressive transmitter compared to JM-TFRC as all its $F$ values are larger. This is because, VTFRC opportunistically uses any positive rate gaps to increase the bit rate of frames with high complexity. This results in an increased long term sending rate and, thus, a higher $F$ compared to JM-TFRC.

However, VTFRC is not significantly more aggressive than the TCP clients, as the $F$ values of VTFRC are all relatively close to 1. Furthermore, $F$ value is less than two which implies that it is TCP-friendly due to the definition of TCP-friendliness in [35]. This is expected as VTFRC only makes use of the TCP-friendly rate region to send its additional rate thus it still produces a TCP-friendly sending rate.

3.4.3. Video Frame Quality Evaluation

Next, we investigate the benefit of VTFRC in terms of video quality by measuring the peak signal-to-noise ratio (PSNR) values of the encoded video at the sender-end and the PSNR values of the decoded video at the receiver-end. Sender-end PSNR is used to show the frame quality of the video before it is transmitted. We compared the video quality of VTFRC and JM-TFRC by calculating the PSNR gain as the difference in the average PSNR produced by VTFRC and JM-TFRC, that is: $PSNR_{VTFRC} - PSNR_{JM-TFRC}$. The PSNR gain is positive if VTFRC produces an average PSNR that is higher than JM-TFRC. The PSNR gain is used to compare the schemes at both the sender-end and receiver-end.

We also take the difference between the PSNR of the encoded video prior to sending at the sender-end and the PSNR of the decoded video at the receiver-end, specifically: sender PSNR − receiver PSNR. The PSNR difference should ideally be zero, however, packet losses that occur during congestion will tend cause the PSNR difference to increase. Both the receiver-end PSNR and difference PSNR are used to show how the congestion affects the frame quality of the video.

The results are shown in table 3.3. The breakdown of the PSNR for each frame for container and foreman at the sender-end is shown in fig. 3.8 and fig. 3.9 respectively.
### Table 3.3. PSNR Comparisons

<table>
<thead>
<tr>
<th>Sequence</th>
<th>VTFRC Mean PSNR (dB)</th>
<th>JM-TFRC Mean PSNR (dB)</th>
<th>PSNR Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sender-End</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>container</td>
<td>45.49</td>
<td>44.97</td>
<td>+0.52</td>
</tr>
<tr>
<td>foreman</td>
<td>37.94</td>
<td>37.36</td>
<td>+0.58</td>
</tr>
<tr>
<td>grandma</td>
<td>45.29</td>
<td>44.31</td>
<td>+0.98</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>44.06</td>
<td>42.91</td>
<td>+1.15</td>
</tr>
<tr>
<td>salesman</td>
<td>45.45</td>
<td>44.73</td>
<td>+0.72</td>
</tr>
<tr>
<td><strong>Mean PSNR Gain:</strong></td>
<td></td>
<td></td>
<td>+0.79</td>
</tr>
<tr>
<td><strong>Receiver-End</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>container</td>
<td>43.37</td>
<td>42.25</td>
<td>+1.12</td>
</tr>
<tr>
<td>foreman</td>
<td>35.38</td>
<td>34.02</td>
<td>+1.35</td>
</tr>
<tr>
<td>grandma</td>
<td>43.37</td>
<td>43.02</td>
<td>+0.35</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>42.46</td>
<td>41.26</td>
<td>+1.20</td>
</tr>
<tr>
<td>salesman</td>
<td>41.36</td>
<td>39.01</td>
<td>+2.34</td>
</tr>
<tr>
<td><strong>Mean PSNR Gain:</strong></td>
<td></td>
<td></td>
<td>+1.27</td>
</tr>
</tbody>
</table>

**PSNR difference between sender and receiver**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Sender PSNR - Receiver PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>2.12</td>
</tr>
<tr>
<td>foreman</td>
<td>2.56</td>
</tr>
<tr>
<td>grandma</td>
<td>1.92</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>1.6</td>
</tr>
<tr>
<td>salesman</td>
<td>4.09</td>
</tr>
</tbody>
</table>

**Figure 3.8.** PSNR of each frame for container at the sender.
From the sender-end results, it is shown that even though VTFRC encodes a frame at a higher rate opportunistically, it still produced an overall mean PSNR gain of 0.79 dB over JM-TFRC. And even after losing video packets due to congestion, from the receiver-end results, VTFRC had an overall mean PSNR gain of 1.27 dB over JM-TFRC. The PSNR difference results suggests that even though VTFRC is in general more aggressive than JM-TFRC, it does not contribute much to the network congestion as the differences for VTFRC are lower to that of JM-TFRC.

To determine how much more VTFRC contributes to the network congestion, we measure the overall network throughput and overall packet losses. Table 3.4 shows the overall network throughput of VTFRC and JM-TFRC. It can be seen from the table that VTFRC does not exhibit any drop in throughput when compared to JM-TFRC. To further examine the congestion state of the network, we look at the packet loss results in table 3.5. Note that for this experiment, any packet losses is due to network congestion. The packet loss percentage in table 3.5 is calculated as the percentage of packets lost over the total number of packets sent. Together with the throughput results, the packet loss results shows that VTFRC does not increase the network congestion significantly.
3. VIDEO TCP FRIENDLY RATE CONTROL

<table>
<thead>
<tr>
<th>Sequence</th>
<th>VTFRC Throughput (MBits/s)</th>
<th>JM-TFRC Throughput (MBits/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>0.582</td>
<td>0.434</td>
</tr>
<tr>
<td>foreman</td>
<td>0.743</td>
<td>0.609</td>
</tr>
<tr>
<td>grandma</td>
<td>0.982</td>
<td>0.957</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>0.942</td>
<td>0.905</td>
</tr>
<tr>
<td>salesman</td>
<td>0.681</td>
<td>0.674</td>
</tr>
</tbody>
</table>

Table 3.4. Overall network throughput comparisons.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>VTFRC Packet Losses (%)</th>
<th>JM-TFRC Packet Losses (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>0.53</td>
<td>3.55</td>
</tr>
<tr>
<td>foreman</td>
<td>2.58</td>
<td>2.74</td>
</tr>
<tr>
<td>grandma</td>
<td>2.46</td>
<td>1.97</td>
</tr>
<tr>
<td>mother-daughter</td>
<td>1.00</td>
<td>0.95</td>
</tr>
<tr>
<td>salesman</td>
<td>2.42</td>
<td>2.11</td>
</tr>
</tbody>
</table>

Table 3.5. Overall network packet losses comparisons.

3.5. Conclusion

This chapter has proposed a joint source rate and congestion control method that incorporates the video bit rate characteristic in the sending rate. This is done by opportunistically coding the video frames depending on the frame complexity and network conditions as interpreted by the TFRC protocol. Since only the TCP-friendly rate region is exploited to do this, the resulting flow is still TCP-friendly although results show that it is slightly more aggressive. However, the video quality gains and the comparable congestion control justify this extra aggressiveness.
CHAPTER 4

Complexity Based H.264/AVC Rate Control

4.1. Introduction

In the previous chapter, we proposed Video TFRC (VTFRC) which is a joint source rate control and congestion control method that incorporates the video bit rate characteristic into a TCP-friendly transmission rate using the frame complexity of the video. In order for VTFRC to function correctly, a requirement is that the encoder must be able to encode frames to a specified target sequence bit-rate, which is known as rate control in video coding. Unfortunately, the originally adopted H.264/AVC rate control scheme does not perform well. Note that we define the target sequence bit-rate as the target bit-rate that the sequence needs to meet after it has been encoded. The target sequence bit-rate may vary over time if it is set to the network bandwidth, e.g. based on the TFRC rate in the previous chapter.

In this chapter, we propose a complexity based H.264/AVC rate control scheme that calculates the frame complexity of a video using edge energy and make use of this to provide a more precise video rate control.

The encoder rate controller regulates the coded video bit-stream in order to meet the network bandwidth and buffer constraints as well as to maximize the video quality as much as possible. This makes rate control one of the key components of a video coder especially in video streaming applications. A typical rate controller first allocates a target number of bits for each frame based on a bit budget, we define this as the frame target bits. Then, for each frame, a quantization parameter (QP) is selected either for the whole frame (frame-layer) or for each macroblock (MB layer) based on some rate-quantization (R-Q) model in order to meet the specified frame target bits.

Recall that one of the main issue with rate control in H.264/AVC is that the bit allocation and QP selection are conducted before the selection of INTER and INTRA modes. This means that various vital frame statistics, such as the mean absolute difference (MAD), is not readily available to the rate controller\textsuperscript{1}. This makes it harder for the rate controller to choose QPs that ensures that the video stream output meets a certain target sequence bit-rate.

\textsuperscript{1}See section 2.2.3.1 of chapter 2
Current approaches \cite{33,32,31,15,22,102,26,27,28,25,21} estimate these frame statistics by using a frame complexity measure that typically can be calculated before the frame is encoded. However, of these approaches, only Leontaris and Tourapis \cite{21} account for the differences between I-frames, P-frames and B-frames in their complexity measures. Recall from the previous chapter that I-frames uses more bits than P-frames, which in turn uses more bits than B-frames, but this relationship might change in a high motion video sequence due to the increased intra-coded macroblocks (MB) being introduced into the frame. These properties need to be considered in a complexity measurement scheme. Although the scheme proposed by Leontaris and Tourapis \cite{21} takes these into account, their scheme depends on parameters which are required to be specified before encoding. These parameters are fixed for the lifetime of the encoding and requires statistics on the whole video sequence to estimate. Furthermore, none of these approaches investigated on the possibility of using edge energy to perform rate control.

We address these issues by firstly, calculating the edge energy of each frame as a form of perceptual complexity measure and using the edge energy to define different complexity measures for I-frames, P-frames and B-frames. We then make use of these complexity measures to handle bit-rate variations due to scene change as well as to allocate bits to a frame with respect to the complexities of other frames within a time scale. We also define a bit-rate traffic model to ensure that the bit allocations by the rate controller meet the target sequence bit-rate in a smaller time scale. Finally, we modify the linear quadratic R-Q model to account for the different bit allocations for different frames. Our scheme estimates the frame statistics while the video is being encoded and does not require any information on the whole video sequence. This makes our scheme suitable for real-time video.

The contributions of this chapter are:

1. Using edge energy to define different complexity measures for I-frames, P-frames and B-frames.
2. The proposed scheme does not require any information on the whole video sequence.
3. We show that the proposed scheme improves over existing schemes by producing a video stream that is closer to the required target bitrate with a better video quality.

This chapter is organised as follows: Section 4.2 describes an overview of our proposed scheme. Section 4.3 shows how the edge energy are calculated from the video frames. Section 4.4 derives the frame complexity measures based on
the edge energy. Section 4.5 describes how the bits are allocated for each frame via the bit-rate traffic model. Section 4.6 describes the quantization selection process using the R-Q model. Section 4.7 demonstrates the performance of our scheme. Section 4.8 concludes this chapter.

4.2. Process Overview

The main goal of a H.264/AVC rate control scheme is to choose a quantization parameter (QP) that will maximize the frame quality of the video while ensuring that the encoded video meets a certain target sequence bit-rate. Most H.264/AVC rate control schemes do this by constructing a rate-quantization (R-Q) model [16]. Then, by using a target number of bits as input, the R-Q model will determine a QP. This process is called QP selection.

In our proposed scheme, we assign a QP to each frame, this is also known as frame layer rate control. This approach was chosen because frame layer rate control requires less computations compared to a macroblock (MB) based QP assignment, also known as MB layer rate control. Since the QP is assigned on a per frame basis, the target number of bits input to the R-Q model needs to represent the frame target bits. Calculating the frame target bits is a process known as bit allocation.

Proper bit allocation requires information on the frame statistics. We estimate the frame statistics by extracting edge energy from each frame. This edge energy is used to determine the frame complexity that is, in turn, used to determine the bits to be allocated to the frame.

To summarize, the processes involved in our proposed H.264/AVC rate control scheme are performed in the following order:

1. Edge energy extraction
2. Bit allocation for each frame based on a bit-rate traffic model
3. QP selection
4.3. Extracting Edge Energy

To calculate the edge energy of a frame, we make use of the edge extraction technique by Won et al \[103\]. Won et al \[103\] make use of edge filters that are applied to non-overlapping blocks of image. We shall make use of these edge filters in our edge extraction algorithm to calculate the total edge energy of a frame. Our modified algorithm starts by linearly quantizing the Y component of the frame into 128 levels as a way of noise removal. The frame is then divided up into 8x8 blocks called image blocks where each image block consists of four 4x4 sub-blocks. Fig. 4.1 illustrates this.

![Figure 4.1. Illustration of pixel groupings in a frame.](image)

To obtain the edge energy for an image block, we apply the following filters to each of its 4x4 sub-blocks:

- **Vertical filters**:
  
  \[
  \begin{array}{cc}
  1 & -1 \\
  1 & -1 \\
  \end{array}
  \]

- **Horizontal filters**:
  
  \[
  \begin{array}{cc}
  1 & 1 \\
  -1 & -1 \\
  \end{array}
  \]

- **45° diagonal filters**:
  
  \[
  \begin{array}{cc}
  \sqrt{2} & 0 \\
  0 & -\sqrt{2} \\
  \end{array}
  \]

- **135° diagonal filters**:
  
  \[
  \begin{array}{cc}
  0 & \sqrt{2} \\
  -\sqrt{2} & 0 \\
  \end{array}
  \]

- **Non-directional filters**:
  
  \[
  \begin{array}{cc}
  2 & -2 \\
  -2 & 2 \\
  \end{array}
  \]

![Figure 4.2. Filter coefficients for edge detection, these are applied to each image block.](image)
The pixel intensities of each sub-block are first averaged and then digital filters are applied to onto the four averaged pixel intensities of each image block, fig. 4.2 shows these filters. To determine the edge energy of an image block, the edge magnitudes are calculated for each possible edge direction. Where the edge directions are classified as vertical, horizontal, 45° diagonal, 135° diagonal, non-directional and no-edge. If all the edge magnitudes for the image block is less than a predefined threshold \( T_{edge} \) (set to 11 as in [103]), the image block is classified as a no-edge block and its edge energy is not counted towards the frame edge energy. Otherwise, the largest edge magnitude, out of the 5 possible edge directions, is chosen as the edge energy of the image block. The edge energy of the frame is then the sum of the edge energies of its image blocks.

Formally, let \( a_n \) denote the averaged pixel intensity of the sub-block and \( a_0 \) to \( a_3 \) are positioned as illustrated in fig. 4.3. The edge magnitudes for each possible edge direction are then calculated for the \( i \)’th image block on frame \( k \), where \( k \in [0..K] \) and \( K \) is the total number of frames in the video, using the digital filters (see fig. 4.2) as:

\[
\begin{align*}
\text{(4.3.1)} & \quad m_v(i, k) = a_0 - a_1 + a_2 - a_3 \\
\text{(4.3.2)} & \quad m_h(i, k) = a_0 + a_1 - a_2 - a_3 \\
\text{(4.3.3)} & \quad m_{d45}(i, k) = \sqrt{2}a_0 - \sqrt{2}a_3 \\
\text{(4.3.4)} & \quad m_{d135}(i, k) = \sqrt{2}a_1 - \sqrt{2}a_2 \\
\text{(4.3.5)} & \quad m_{nd}(i, k) = 2a_0 - 2a_1 - 2a_2 + 2a_3
\end{align*}
\]

where \( m_v(i, k) \) represents the edge energy of the \( i \)’th image block in the \( k \)’th frame in the vertical direction, \( m_h(i, k) \) represents the horizontal direction edge energy, \( m_{d45}(i, k) \) represents the 45° diagonal direction edge energy, \( m_{d135}(i, k) \) represents the 135° diagonal direction edge energy, and \( m_{nd}(i, k) \) represents the non-directional edge energy. The edge energy is determined for the \( i \)’th image block on frame \( k \) as:
(4.3.6) $B_{\text{max}}(i, k) = \max(m_v(i, k), m_h(i, k), m_{d45}(i, k), m_{d135}(i, k), m_{nd}(i, k))$

(4.3.7) $B_E(i, k) = \begin{cases} B_{\text{max}}(i, k) & B_{\text{max}}(i, k) \geq T_{\text{edge}} \\ 0 & \text{otherwise} \end{cases}$

Given a frame with $N$ image blocks, the edge energy of the frame $k$ is:

(4.3.8) $E(k) = \sum_{i=0}^{N-1} B_E(i, k)$

4.4. Frame Complexity Measure

In our proposed method, we made use of the edge energy extracted from a frame to calculate the frame complexity. This is because the AC coefficients of a compressed video represent edge information, so a frame with higher edge energy would tend to contain more AC coefficients which typically means that the I-frame would end up using more bits. Furthermore, motion information can be represented by the localized differences of edge energy between frames, as edge energy tends to change more when there is high motion. We now show how the edge energy is calculated for an I-frame, P-frame and B-frame. The edge energy for an I-frame with $N$ image blocks is:

(4.4.1) $E_I(k) = \sum_{i=0}^{N-1} B_E(i, k)$

Let $\rho$ be the previous anchor frame, then the edge energy of the current P-frame $k$ is:

(4.4.2) $E_P(k) = \sum_{i=0}^{N-1} |B_E(i, k) - B_E(i, \rho)|$

Let $\eta$ be the next anchor frame, then the edge energy of a B-frame is:

(4.4.3) $E_B(k) = \frac{1}{2} \sum_{i=0}^{N-1} |B_E(i, k) - B_E(i, \rho)| + |B_E(i, k) - B_E(i, \eta)|$

Finally, the complexity measure of frame $k$ is calculated as:

(4.4.4) $C_x(k) = \frac{E_x(k)}{\frac{1}{k-1} \sum_{x=0}^{k-2} E_x(l_x)} \quad x \in I, P, B$
Equation (4.4.4) is calculated for each group of I-frames, P-frames and B-frames. The denominator determines the average edge energy of the past \((k - 1)\) I/P/B-frames.

4.5. Bit Allocation

4.5.1. Bit-Rate Traffic Model

To determine the bit allocation for a frame, we make use of a bit-rate traffic model. The bit-rate traffic model specifically tries to calculate and update the current available bits at a certain time scale. The available bits are then used to determine the bits to allocate each frame.

Recall that \(K\) is the total number of frames in the whole video to be encoded. Then, we divide the \(K\) frames of the video into segments with \(R_F\) frames in each segment. Each of these segments is defined as a bit-rate period. We now define \(R_B(k)\) as the target sequence bit-rate at the current \(k\)’th frame, where \(k \in [0..K]\). If \(R_F\) is set to the sequence frame rate with \(R_B(k)\) as the target sequence bit-rate, it means that a bit-rate period need to be encoded to \(R_B(k)\) bits or less, in order to meet the target sequence bit-rate. The sequence frame rate here refers to the natural frame rate of the video. For example, suppose the sequence frame rate is 10 fps and the target sequence bit-rate \(R_B(k) = 100\) kbps. This means that for a rate controller to meet the target sequence bit-rate, the frames need to be an average size of 10 kilobits. In this scenario, when \(R_F = 10\) frames, then every bit-rate period of 10 frames can at most be encoded to 100 kilobits.

Let \(A(t,k)\) be the available bits for allocation in the \(t\)’th bit-rate period just before frame \(k\) is encoded, frame \(s_t\) be the frame at the start of the \(t\)’th bit-rate period and \(b(k)\) be the actual bits used by frame \(k\) after it is encoded. Fig. 4.4 illustrates these definitions.

The bit-rate traffic model then can be seen as a buffer of \(A(t,k)\) available bits. At the beginning of every bit-rate period, the available bits \(A(t,s_t)\) will increase by \(R_B(s_t)\) bits and decrease by the total number of actuals bits used to encode the previous bit-rate period \(\sum_{i=s_t-1}^{s_t-R_F} b(i)\), the available bits \(A(t,s_t)\)

\[2R_B(k)\] might change from frame to frame in this model.
is then updated every bit-rate period, fig. 4.5 illustrates this. In practice, the
bit-rate traffic model updates the available bits \( A(t, k) \) after every frame has
been encoded to capture any changes in \( R_B(k) \) from one frame to the next.
This is done in addition to the update that occurs at every bit rate period

\[
R_B(s_t) \rightarrow \sum_{i=s_t-1}^{s_t-R_F} b(i) \rightarrow A(t, s_t)
\]

**Figure 4.5.** Illustration of the buffer concept of the bit-rate
traffic model updated at every bit-rate period.

Formally, at the start of the video, the available bits \( A(0, 0) \) before frame
0 is encoded in bit-rate period 0 (the first bit-rate period) is:

\[
A(0, 0) = R_B(0)
\]

Just before frame \( k \) in the \( t \)’th bit-rate period is encoded, where \( k > 0 \), the
bits left for frame allocation \( A(t, k) \) are:

\[
A(t, k) = A(t, k-1) - b(k-1) + (R_B(k) - R_B(k-1))
\]

The last term of (4.5.2) updates the available bits \( A(t, k) \) based on any
changes on the target sequence bit-rate \( R_B(k) \) between frame \( k-1 \) and \( k \).

After bit-rate period 0, at the start of the \( t \)’th bit-rate period, just before
frame \( s_t \) is encoded, the available bits \( A(t, s_t) \) are updated as:

\[
A(t, s_t) = A(t, s_t-1) - b(s_t-1) + R_B(s_t)
\]

**4.5.2. Frame Layer Bit Allocation**

To determine the bits allocated to a frame, we make use of the frame complexity
\( C_x(k) \) in (4.4.4). The frame target bits for frame \( k, T(k) \), in the \( t \)’th bit-rate
period is then calculated by:

\[
T(k) = \frac{C_x(k)}{N_{rI}C_I + N_{rP}C_P + N_{rB}C_B} \cdot A(t, k) \quad x \in I, P, B
\]

where \( C_I, C_P \) and \( C_B \) are the mean complexities for I-frames, P-frames
and B-frames respectively. The mean complexities are calculated based on all
the past frames that have been encoded. \( N_{rI}, N_{rP}, N_{rB} \) are the remaining
number of I-frames, P-frames and B-frames left to code in the bit-rate period
respectively.
The idea of (4.5.4) is to estimate the amount of bits to allocate the current frame $k$ based on its complexity in relation to the average complexity of the frames remaining to be encoded in the bit-rate period. As the complexity of the remaining frames to be encoded in the bit period is not known beforehand, we estimate it by using the mean complexity value of each I/P/B-frame. We estimate $C_I$, $C_P$ and $C_B$ by maintaining a sliding window of the complexity values of the past frames for each I/P/B-frame, and averaging the values within the window. The sliding window is not restarted for each bit rate period. In our experiments, the sliding window sizes were set to 2 for I-frame complexity and 3 for both P and B frame complexity. This sliding window technique is used to make the system more responsive to bit allocation changes due to scene change.

For the first I, P and B-frames in the video, we determine their QPs directly, this will be discussed further in the next section.

4.6. Quantization Parameter Selection

QP selection is conducted using the quadratic R-Q model [16]. Three quadratic models are used for each I-frame, P-frame and B-frame respectively as the linear prediction model is different for each frame type. For I-frames, the model is:

$$(4.6.1) \quad \frac{T(k) - h(k)}{C_I(k)} = \frac{a_1}{QP(k)} + \frac{a_2}{QP^2(k)}$$

While the model for P-frames as well as the model for B-frames is:

$$(4.6.2) \quad \frac{T(k) - h(k)}{\alpha \cdot PMAD(k) + (1 - \alpha) \cdot C_x(k)} = \frac{a_1}{QP(k)} + \frac{a_2}{QP^2(k)} \quad x \in P, B$$

Where $h(k)$ is the amount of bits used for the header information of the frame, $a_1$ and $a_2$ are the first and second order model coefficients respectively, $QP(k)$ is the quantization level for frame $k$, $PMAD(k)$ is the linearly predicted mean absolute value (MAD) for frame $k$ as defined in [15] and $\alpha$ is a weighting factor (set to 0.5 in our experiments). The model coefficients are estimated using the linear regression technique based on the actual bits and QPs of previously encoded frames as described in [16]. Using the R-Q models (4.6.1) and (4.6.2), the QP for the frame can be determined based on the technique described in [16].

\textsuperscript{3}See also section 2.2.3 of chapter 2
For the first I, P, B-frames in the video, we determine their QPs based on the initial QP calculation scheme proposed in [15]. For the first I and P frames in the video, their QPs are estimated by:

\[
QP_1 = \begin{cases} 
    40 & \text{bpp} \leq l_1 \\
    30 & l_1 < \text{bpp} \leq l_2 \\
    20 & l_2 < \text{bpp} \leq l_3 \\
    10 & \text{bpp} > l_3 
\end{cases}
\]

where \( l_1 = 0.15, l_2 = 0.45 \) and \( l_3 = 0.9 \) is recommended for QCIF/CIF size videos, and \( l_1 = 0.6, l_2 = 1.4 \) and \( l_3 = 2.4 \) is recommended for videos larger than CIF. bpp stands for bits per pixel and, with \( N_{\text{pixel}} \) as the number of pixels in the frame, is calculated as:

\[
bpp = \frac{R_B(0)}{R_F \times N_{\text{pixel}}}
\]

Let \( j \) represent the frame number of the first B-frame, \( j - P_1 \) be the first P-frame before B-frame \( j \), \( j + P_2 \) be the first P-frame after B-frame \( j \). Then, the QP of the first B-frame \( j \) is estimated as:

\[
QP(j) = \begin{cases} 
    \frac{QP(j-P_1)+QP(j+P_2)+2}{2} & \text{if } QP(j - P_1) \neq QP(j + P_2) \\
    QP(j - P_1) + 2 & \text{otherwise}
\end{cases}
\]

4.7. Experimental Results

4.7.1. Setup

We tested our proposed method on seven different video sequences of CIF size (352x288), comprising of both high and low motion contents. The H.264/AVC reference software JM12.2 was used to conduct our simulations. RD optimizations were turned off in the software and the group-of-pictures (GOP) structure was specified as I-B-B-P with a GOP size of 10.

We ran our simulations with a frame rate of 15 fps with no frame skipping and at a constant bit-rate. The bit-rate period is set to one second. We then compared our proposed method, named here as RC4, with the original adopted H.264/AVC rate control scheme by Li et al [15], called RC0 here, as well as the modified H.264/AVC rate control scheme by Leontaris and Tourapis [21],

4B-frames are calculated (or predicted in video coding terms) based on its neighbouring P-frames. In this case, B-frame \( j \) is calculated from both P-frames \( j - P_1 \) and \( j + P_2 \).

5That is, one GOP will be: IBBPBBPBBP.
called RC3 here. The frame-layer rate control was enabled for RC0 and RC3. The parameters for RC3 were fixed with RCISliceBitRatio set to 1, RCB-SliceBitRatio0 set to 0.5, RCBoverPRatio set to 0.45 and RCloverPRatio set to 3.8. These values were recommended in [21] and determine the complexity relationship of I-frames, P-frames and B-frames. Hierarchial coding was disabled.

4.7.2. Satisfying the Bit-Rate Constraint

To check if the method meets the target sequence bit-rate constraint at the given frame rate, we averaged the actual bits used by the frames in a bit-rate period (i.e. the actual bits used by every set of 15 frames in this experiment). Note that the target sequence bit-rate is a constant bit-rate. The actual mean bit-rate is computed along with the percentage deviation from the target sequence bit-rate (error) for each method as shown in Table 4.1. The breakdown of the actual bits used for each bit-rate period for foreman is shown in fig. 4.6 while fig. 4.7 shows the same results without RC0.

RC0 shows a large deviation, almost four times on average, from the target sequence bit-rate. This highlights the inability of RC0 to do a proper rate control on I-frames and B-frames as the QPs for I-frames and B-frames are determined from previously encoded frames and may not meet the target sequence bit-rate constraint [21]. In contrast, RC3 shows a much better conformance to the bit-rate and frame rate constraints compared to RC0. However, RC4 still outperforms RC3 by a fair amount. The main reason for this is that RC3 tends to allocate more bits than allowed within a bit-rate period. Also the larger bit-rate variation of RC3 due to frequent scene changes is evident in the results for high motion sequences (i.e. football and stefan), while our proposed method shows a much smaller bit-rate variation in the same high-motion sequences.

To further examine how variable the methods are we look at their variances and mean square error to the target bit rate, these are shown in table 4.2 and table 4.3 respectively. The lower variability of RC4 confirms our above observation.

It can be seen from Table 4.1 that RC3 tends to encode the sequence at a slightly higher average bit-rate than RC4. This seems to suggest that the PSNR performance of RC3 will be higher since a higher bit-rate correlates to a higher PSNR. However, fig. 4.7 shows that RC3 obtains a higher bit-rate by allocating more bits to the frames at the early part of the sequence. This over-allocation of bits is then heavily penalized towards end of the sequence where a sharp dip to half of the target bits can be seen. This uneven bit
allocation will cause fluctuations in the sequence quality. In the next section, we examine the PSNR performance of RC3 and RC4.

**Figure 4.6.** The actual bits used for each bit-rate period for foreman.
Figure 4.7. The actual bits used for each bit-rate period for foreman. \textit{RC0} is not shown here as the excessive bit-rates it generates skew the graph.
<table>
<thead>
<tr>
<th>Seq</th>
<th>Target Bit-Rate (kbps)</th>
<th>RC0</th>
<th>Error (%)</th>
<th>Actual Bit-Rate (kbps)</th>
<th>Error (%)</th>
<th>Actual Bit-Rate (kbps)</th>
<th>Error (%)</th>
<th>Actual Bit-Rate (kbps)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>500</td>
<td>3460</td>
<td>+592</td>
<td>513</td>
<td>-2.6</td>
<td>497</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>3987</td>
<td>+470</td>
<td>704</td>
<td>+0.6</td>
<td>696</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>4149</td>
<td>+361</td>
<td>903</td>
<td>+0.3</td>
<td>897</td>
<td>-0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>4682</td>
<td>+290</td>
<td>1208</td>
<td>+0.7</td>
<td>1193</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>4823</td>
<td>+222</td>
<td>1524</td>
<td>+1.6</td>
<td>1493</td>
<td>-0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>football</td>
<td>500</td>
<td>1122</td>
<td>+124.4</td>
<td>528</td>
<td>+5.6</td>
<td>493</td>
<td>-1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>1122</td>
<td>+60.3</td>
<td>730</td>
<td>+4.3</td>
<td>693</td>
<td>-1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>1122</td>
<td>+24.7</td>
<td>947</td>
<td>+5.2</td>
<td>891</td>
<td>-1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>1963</td>
<td>+63.6</td>
<td>1284</td>
<td>+7.0</td>
<td>1195</td>
<td>-0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>1963</td>
<td>+30.9</td>
<td>1603</td>
<td>+6.9</td>
<td>1497</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>foreman</td>
<td>500</td>
<td>4551</td>
<td>+810.2</td>
<td>515</td>
<td>+3.0</td>
<td>501</td>
<td>+0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>4771</td>
<td>+581.6</td>
<td>722</td>
<td>+3.1</td>
<td>700</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>5182</td>
<td>+475.8</td>
<td>928</td>
<td>+3.1</td>
<td>900</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>5618</td>
<td>+368.2</td>
<td>1238</td>
<td>+3.2</td>
<td>1198</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>5998</td>
<td>+299.9</td>
<td>1545</td>
<td>+3.0</td>
<td>1498</td>
<td>-0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mobile</td>
<td>500</td>
<td>7249</td>
<td>+1350.8</td>
<td>515</td>
<td>+3.0</td>
<td>499</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>7249</td>
<td>+935.6</td>
<td>718</td>
<td>+2.6</td>
<td>700</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>7249</td>
<td>+705.4</td>
<td>927</td>
<td>+3.0</td>
<td>900</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>8945</td>
<td>+645.4</td>
<td>1234</td>
<td>+2.8</td>
<td>1198</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>8945</td>
<td>+496.3</td>
<td>1538</td>
<td>+2.5</td>
<td>1499</td>
<td>-0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>news</td>
<td>500</td>
<td>2720</td>
<td>+444.0</td>
<td>508</td>
<td>+1.6</td>
<td>497</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>2925</td>
<td>+317.9</td>
<td>711</td>
<td>+1.6</td>
<td>697</td>
<td>-0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>3015</td>
<td>+235.0</td>
<td>914</td>
<td>+1.6</td>
<td>896</td>
<td>-0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>3415</td>
<td>+184.6</td>
<td>1226</td>
<td>+2.2</td>
<td>1193</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>3474</td>
<td>+131.6</td>
<td>1536</td>
<td>+2.4</td>
<td>1491</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesman</td>
<td>500</td>
<td>4178</td>
<td>+735.6</td>
<td>512</td>
<td>+2.4</td>
<td>500</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>4636</td>
<td>+562.3</td>
<td>717</td>
<td>+2.4</td>
<td>699</td>
<td>-0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>4742</td>
<td>+426.9</td>
<td>917</td>
<td>+1.9</td>
<td>899</td>
<td>-0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>5088</td>
<td>+324.0</td>
<td>1227</td>
<td>+2.3</td>
<td>1197</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>5180</td>
<td>+245.3</td>
<td>1524</td>
<td>+1.6</td>
<td>1497</td>
<td>-0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stefan</td>
<td>500</td>
<td>1834</td>
<td>+266.8</td>
<td>547</td>
<td>+9.4</td>
<td>503</td>
<td>+0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>1834</td>
<td>+162.0</td>
<td>764</td>
<td>+9.1</td>
<td>696</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>1834</td>
<td>+103.8</td>
<td>980</td>
<td>+8.9</td>
<td>895</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>2922</td>
<td>+143.5</td>
<td>1304</td>
<td>+8.7</td>
<td>1195</td>
<td>-0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>2922</td>
<td>+94.8</td>
<td>1584</td>
<td>+5.6</td>
<td>1491</td>
<td>-0.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.1.** The actual mean bit-rates and error of the proposed method (RC4), the original H.264/AVC method (RC0) and the modified H.264/AVC method (RC3).
<table>
<thead>
<tr>
<th>Sequence</th>
<th>Target Bit-Rate (kbps)</th>
<th>RC0</th>
<th>RC3</th>
<th>RC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>500</td>
<td>16837797</td>
<td>271257</td>
<td>274788</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>20551177</td>
<td>499989</td>
<td>510854</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>21762306</td>
<td>817759</td>
<td>835382</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>25527631</td>
<td>1477851</td>
<td>1462838</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>26680471</td>
<td>2346001</td>
<td>2302121</td>
</tr>
<tr>
<td>football</td>
<td>500</td>
<td>1371687</td>
<td>296388</td>
<td>244149</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>1371687</td>
<td>546280</td>
<td>481049</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>1371687</td>
<td>914272</td>
<td>795444</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>4140684</td>
<td>1702229</td>
<td>1429767</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>4140684</td>
<td>2618730</td>
<td>224638</td>
</tr>
<tr>
<td>foreman</td>
<td>500</td>
<td>30701152</td>
<td>272425</td>
<td>252897</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>32447751</td>
<td>538555</td>
<td>492755</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>35873346</td>
<td>890246</td>
<td>813372</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>39428396</td>
<td>1591300</td>
<td>1444181</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>42879891</td>
<td>2462749</td>
<td>2255018</td>
</tr>
<tr>
<td>mobile</td>
<td>500</td>
<td>65330395</td>
<td>300151</td>
<td>250918</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>65330395</td>
<td>557375</td>
<td>557866</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>65330395</td>
<td>914808</td>
<td>868747</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>91655202</td>
<td>1626542</td>
<td>1441855</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>91655202</td>
<td>2501926</td>
<td>2282019</td>
</tr>
<tr>
<td>news</td>
<td>500</td>
<td>9457107</td>
<td>263919</td>
<td>254063</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>10545721</td>
<td>516351</td>
<td>493418</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>11063849</td>
<td>852718</td>
<td>812326</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>13147771</td>
<td>1533195</td>
<td>1443856</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>13553977</td>
<td>2420504</td>
<td>2262313</td>
</tr>
<tr>
<td>salesman</td>
<td>500</td>
<td>22384030</td>
<td>274349</td>
<td>279693</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>25521336</td>
<td>538213</td>
<td>519614</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>26285815</td>
<td>867893</td>
<td>859362</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>28694304</td>
<td>1548219</td>
<td>1483937</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>29434459</td>
<td>2394552</td>
<td>2307697</td>
</tr>
<tr>
<td>stefan</td>
<td>500</td>
<td>3526987</td>
<td>317862</td>
<td>259622</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>3526987</td>
<td>611010</td>
<td>485562</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>3526987</td>
<td>1000322</td>
<td>806736</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>8897518</td>
<td>1753213</td>
<td>1435067</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>8897518</td>
<td>2566792</td>
<td>2233773</td>
</tr>
</tbody>
</table>

**Mean variance:** 25795780 1160170 1062220

Table 4.2. The variances of the proposed method (RC4), the original H.264/AVC method (RC0) and the modified H.264/AVC method (RC3).
<table>
<thead>
<tr>
<th>Sequence</th>
<th>Target Bit-Rate (kbps)</th>
<th>RC0</th>
<th>RC3</th>
<th>RC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>500</td>
<td>13630790</td>
<td>8371</td>
<td>27995</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>15463520</td>
<td>5772</td>
<td>26772</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>15108597</td>
<td>2809</td>
<td>31693</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>15735274</td>
<td>19572</td>
<td>40096</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>14466622</td>
<td>25320</td>
<td>73835</td>
</tr>
<tr>
<td>football</td>
<td>500</td>
<td>500668</td>
<td>18518</td>
<td>1411</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>291812</td>
<td>14728</td>
<td>1864</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>162956</td>
<td>21238</td>
<td>2650</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>871650</td>
<td>62727</td>
<td>3292</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>503901</td>
<td>60556</td>
<td>5687</td>
</tr>
<tr>
<td>foreman</td>
<td>500</td>
<td>26404664</td>
<td>8181</td>
<td>2623</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>26262507</td>
<td>18331</td>
<td>3712</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>27360094</td>
<td>30691</td>
<td>5151</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>27391702</td>
<td>60813</td>
<td>9660</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>27141405</td>
<td>79651</td>
<td>11670</td>
</tr>
<tr>
<td>mobile</td>
<td>500</td>
<td>58339009</td>
<td>35315</td>
<td>2169</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>55679556</td>
<td>43178</td>
<td>69125</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>53100102</td>
<td>56490</td>
<td>60471</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>71636969</td>
<td>106579</td>
<td>8013</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>67080175</td>
<td>138791</td>
<td>35854</td>
</tr>
<tr>
<td>news</td>
<td>500</td>
<td>6989757</td>
<td>6137</td>
<td>7080</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>6943648</td>
<td>11053</td>
<td>8829</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>6450648</td>
<td>18208</td>
<td>10920</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>6395758</td>
<td>31880</td>
<td>21642</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>5386521</td>
<td>65211</td>
<td>39805</td>
</tr>
<tr>
<td>salesman</td>
<td>500</td>
<td>18460493</td>
<td>13337</td>
<td>30588</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>19524908</td>
<td>24593</td>
<td>31845</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>18565799</td>
<td>28806</td>
<td>52185</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>17929237</td>
<td>45215</td>
<td>52819</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>16149523</td>
<td>74142</td>
<td>68383</td>
</tr>
<tr>
<td>stefan</td>
<td>500</td>
<td>1945156</td>
<td>21651</td>
<td>7349</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>1451691</td>
<td>32163</td>
<td>2402</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>1038225</td>
<td>46698</td>
<td>5872</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>3327264</td>
<td>64825</td>
<td>9448</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>2383970</td>
<td>66328</td>
<td>11727</td>
</tr>
</tbody>
</table>

**Table 4.3.** The mean square errors of the proposed method (RC4), the original H.264/AVC method (RC0) and the modified H.264/AVC method (RC3).

4.7.3. **Video Frame Quality Test**

We did a comparison on the decoded video quality output to show that our proposed method do not compromise heavily on the quality in order to meet
4.7. EXPERIMENTAL RESULTS

the bit-rate and frame rate. We chose not to include RC0 in this test due to the fact that it deviates far too much from the target sequence bit-rate to make a fair comparison on the decoded video quality. We calculate the mean output PSNR of the Y-component of the frames and the PSNR gain of our proposed method compared to RC3 as shown in table 4.4. The breakdown of the Y PSNR for each frame for foreman is shown in fig. 4.8.

The results show that our proposed method not only did not perform worse than RC3, but in general performed better by a fair amount in almost all cases with a mean PSNR gain of 0.98 dB. RC3 requires its parameters to be tuned for each sequence and this is difficult to do in general. Using fixed parameter values causes RC3 to perform badly as shown in the results. Moreover, RC3 tends to allocate a smaller amount of bits for the I-frames, doing this may sometimes degrade the quality on subsequent P-frames and B-frames. Our proposed method, on the other hand, allocates bits purely based on the derived complexity measure of the frame, avoiding the issue of choosing parameters by providing an accurate estimated weighting to I-frames, P-frames and B-frames.
### Table 4.4. PSNR of the proposed method (RC4) and the modified H.264/AVC method (RC3)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Target Rate (kbps)</th>
<th>RC3 Mean PSNR (dB)</th>
<th>RC4 Mean PSNR (dB)</th>
<th>PSNR Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>500</td>
<td>38.84</td>
<td>38.38</td>
<td>-0.46</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>40.27</td>
<td>40.98</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>41.36</td>
<td>42.29</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>43.04</td>
<td>43.99</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>44.38</td>
<td>45.21</td>
<td>0.83</td>
</tr>
<tr>
<td>football</td>
<td>500</td>
<td>35.59</td>
<td>35.67</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>37.15</td>
<td>37.33</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>38.43</td>
<td>38.77</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>39.89</td>
<td>40.34</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>41.25</td>
<td>41.68</td>
<td>0.43</td>
</tr>
<tr>
<td>foreman</td>
<td>500</td>
<td>35.41</td>
<td>36.5</td>
<td>1.09</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>36.74</td>
<td>37.78</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>37.82</td>
<td>39.17</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>39.2</td>
<td>40.7</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>40.5</td>
<td>41.92</td>
<td>1.42</td>
</tr>
<tr>
<td>mobile</td>
<td>500</td>
<td>25.39</td>
<td>26.44</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>26.8</td>
<td>27.75</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>28.25</td>
<td>29.35</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>30.1</td>
<td>31.68</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>31.69</td>
<td>33.08</td>
<td>1.39</td>
</tr>
<tr>
<td>news</td>
<td>500</td>
<td>41.34</td>
<td>42.13</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>43.16</td>
<td>44.1</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>44.59</td>
<td>45.56</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>46.11</td>
<td>47.77</td>
<td>1.66</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>47.89</td>
<td>49.28</td>
<td>1.39</td>
</tr>
<tr>
<td>salesman</td>
<td>500</td>
<td>38.78</td>
<td>39.22</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>40.09</td>
<td>41.3</td>
<td>1.21</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>41.36</td>
<td>42.76</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>42.87</td>
<td>44.96</td>
<td>2.09</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>43.82</td>
<td>46.15</td>
<td>2.33</td>
</tr>
<tr>
<td>stefan</td>
<td>500</td>
<td>29.32</td>
<td>29.7</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>700</td>
<td>31.02</td>
<td>32</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>900</td>
<td>32.44</td>
<td>33.37</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>34.41</td>
<td>35.23</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>36.19</td>
<td>37.28</td>
<td>1.09</td>
</tr>
</tbody>
</table>

Mean Gain: 0.98
4.8. Conclusion

In this chapter, we proposed a novel rate control scheme by using the edge energy of a frame to estimate the frame complexity and integrated it into the R-Q model. We also proposed a bit-rate traffic model to determine the bit allocation. Results showed that using edge energy to define different complexity measures for I, P and B frames coupled with the bit-rate traffic model provides a more stringent adherence to the target sequence bit-rates while improving on the frame quality of the video output. Additionally, our proposed method does not assume that any information on the whole video sequence is available, making it suitable for real-time video applications.

Figure 4.8. Y PSNR of each frame for foreman
CHAPTER 5

Frame Rate Optimization Framework For Video Continuity

5.1. Introduction

In previous chapters, we proposed VTFRC, a joint source rate control and congestion control method that incorporates the video bit rate characteristic into the sending rate. We then proposed a complexity based H.264/AVC rate control scheme that complements VTFRC by calculating the frame complexity based on edge energy and using it to ensure that the encoded video meets the required target bitrate. These aim to address the issues in video rate control. In this chapter, we focus on frame rate control by building a framework that maximizes the video quality and video continuity.

A typical way to improve video continuity is to prebuffer video at the client-end (i.e. the decoder). Video can be prebuffered by delaying the initial video playout, this allows for a smoother video quality at the expense of an initial viewing delay. From a network’s perspective, it is desirable to have a large prebuffer to deal with the large variations in the network condition. This reduces the probability of a buffer underflow occurrence. However, from a user’s perspective, larger prebuffering translates to longer wait times for the video to begin playing which may impact on the user’s perceived video quality. Indeed, many user tend to rapidly switch between channels to preview the channel content before settling down on a channel to watch, thus, channel switching can have an impact on the user’s experience [12]. Having a long prebuffering time between channel switching would discourage the users from using the video application.

However, reducing the prebuffering size has the consequence of increasing the chance of buffer underflows which will impact on the video continuity. Here, we assume a small prebuffer and deal with the continuity of the video in a reactive manner. Current approaches handle this by using frame rate control to reduce the frame generation rate at the decoder. This is known as adaptive media playout (AMP) [80, 92, 13]. We define the playout frame rate as the frame generation rate at the decoder. AMP has been shown to reduce prebuffering [92, 13]. However, it introduces playout distortion to
5.1. INTRODUCTION

the viewers, this is because the video is being played slower than its natural playout frame rate. The playout distortion can be reduced by limiting the playout adjustments, but it potentially affects the video continuity.

We preserve the continuity of the video by using frame rate control to ensure that the buffer maintains at a certain level of occupancy. As each bandwidth drop or packet loss event lowers the buffer level and increases the possibility of a buffer underflow, we send more video frames to compensate the dips in the buffer level. We name this process buffer compensation. This can be done by adjusting the frame generation rate at the encoder so that more video frames will be transmitted to the decoder buffer. Simultaneously, the decoder also slows down the playout frame rate of the video to extend the existing video content in the decoder buffer. We define the encoder frame generation rate as the frame generation rate at the encoder.

Performing a joint adjustment of the encoder frame generation rate and the playout frame rate requires finding the optimal trade-off. To do this, we formulate and systematically derive optimization policies using Lyapunov optimization. These optimization policies form the frame rate optimization framework.

To summarize, the contributions of this chapter are:

1. We perform frame rate control by a joint adjustment of the encoder frame generation rate and the playout frame rate.
2. We characterize the frame rate control problem and use Lyapunov optimization to systematically derive the optimization policies. We show that these policies can be decoupled into separate encoder and decoder optimization policies with feedback between the two. This allows for a distributed implementation of our framework.
3. We show that our framework maximizes the video continuity with a modest tradeoff in video quality. We also show that this significantly reduces the prebuffering requirements of the video application.

This chapter is organized as follows: Section 5.2 describes how the encoder frame generation rate and playout frame rate will be adjusted and the trade-offs that result. Section 5.3 describes the model and definitions used in this chapter. Section 5.4 presents a formulation of the frame rate control problem based on the encoder frame generation rate and playout frame rates. Section 5.5 demonstrates how the optimization policies are derived using Lyapunov optimization.

---

\(^1\)This is also known as the capture frame rate of the video. We assume that the natural playout frame rate of the video is a constant in this thesis.

\(^2\)Recall that the frame generation rate is the number of frames being output by either the encoder or decoder.
optimization as well as the normalizations used. We also show that the resulting policies can be decoupled after the derivations. Section 5.6 explores some video quality functions that can be used with the optimization policies. Section 5.7 shows the performance bounds for the optimization policies. Section 5.9 shows the evaluation of our framework compared with the current approaches. Section 5.10 concludes this chapter.

5.2. Preserving Video Continuity

In this section, we discuss the mechanisms we use to preserve the continuity of the video playout, namely buffer compensation and adaptive media playout. We also discuss the trade-offs that result from adjusting these mechanisms.

5.2.1. Video Transmission Scenario

![Diagram of video transmission scenario]

Figure 5.1. Encoding-decoding flow with encoder frame generation rate $i(t)$, sending frame rate $\mu(t)$, receiving frame rate $\lambda(t)$ and playout frame rate $o(t)$. All rates are in frames per seconds.

Figure 5.1 shows a typical video transmission scenario. The encoder generates video frames at a rate of $i(t)$ frames per second (fps) at time $t$ into the encoder buffer. The network transport protocol then transmits the video data from the encoder buffer into the network at a rate of $\mu(t)$ fps. The transmitted video data will be received at the decoder at a rate of $\lambda(t)$ fps in its buffer. The decoder then proceeds to playout the received video data from the decoder buffer at a rate of $o(t)$ fps. Video data arriving after the playout deadline are assumed to be lost.

Notice that if $i(t)$ and $o(t)$ are fixed to the natural playout frame rate, then $\lambda(t) \leq o(t)$ due to the delays and losses that may occur while traversing the network (e.g. due to congestion). A side-effect of this is that whenever a sufficiently long series of consecutive network loss or delay occurs or when the bandwidth drops, the decoder’s buffer occupancy will drop. This makes the decoder sensitive to network jitters and further losses, and may cause it to underflow. Thus, there is a need for buffer compensation and AMP to handle this issue.
5.2. PRESERVING VIDEO CONTINUITY

5.2.2. Buffer Compensation Using Encoder Frame Generation Rate

We perform buffer compensation by adjusting the encoder frame generation rate \( i(t) \) at the encoder. In this chapter, the encoder frame generation rate \( i(t) \) is defined as the frames produced by the encoder into the sending buffer per second (see fig. 5.1 for an illustration).

Increasing the encoder frame generation rate \( i(t) \) is likely to cause the video bitrate to increase as more frames are produced per second by the encoder. This has an adverse effect of the video bitrate exceeding the available bandwidth when the encoder frame generation rate increases. To ensure that the video bitrate does not exceed the available bandwidth, we introduce additional compression to the video such that the higher the encoder frame generation rate \( i(t) \), the higher the compression applied. Introducing compression in such a way will affect the frame quality of the video, we will show in later sections how the optimization policies trade-off the frame quality.

The additional compression is introduced by setting the input frame rate to the rate controller to the encoder frame generation rate \( i(t) \). To see how this works, assume that the rate controller adjusts the encoder such that the average frame size produced by the encoder is:

\[
(5.2.1) \quad \text{average frame size} = r(i(t)) = \frac{ABR(t)}{i(t)}
\]

where \( ABR(t) \) is the available bandwidth at time \( t \) and \( i(t) \) is the encoder frame generation rate \( i(t) \) that is input into the rate controller. Equation (5.2.1) is valid in most popular rate control schemes as they use (5.2.1) as part of their fluid flow model \[17, 15, 104\]. Since \( i(t) \) frames are produced per second, using (5.2.1) means that the bits produced per second is \( \frac{ABR(t)}{i(t)} \times i(t) = ABR(t) \).

Thus, it can be seen from (5.2.1) that the video bitrate will not exceed the available bandwidth for different values of the encoder frame generation rate \( i(t) \). Also, the average frame size decreases when \( i(t) \) is increased which would tend to reduce the frame quality. Finally, (5.2.1) also ensures that more frames will be sent to the decoder and thus allows us to perform buffer compensation.

5.2.3. Decoder Playout Slowdown

At the decoder, the video playout frame rate \( o(t) \) will be slowed down in a process known as adaptive media playout (AMP) \[92, 80, 13\]. The idea is that the slower video playout allows time for the buffer to fill up to the required level without the need to stop the video for rebuffering. This has
been shown to allow the continuity of the video to be preserved longer and reduce the amount of prebuffering required [13].

Although slowing down the video playout can help us with the continuity problem, it may potentially affect the viewer’s perception of the video, we call this *playout distortion*. However, informal subjective tests have shown that slowing the playout by up to 25% is mostly unnoticeable to the viewers and that it is preferable to halting the playout [105, 106]. The accompanying audio could be slowed down as well. Studies have shown that audio could be scaled by up to two times while preserving the pitch [78]. In this thesis, we focus on handling video playout slowdowns.

5.2.4. **Design Considerations**

We want the framework to adjust the encoder frame generation rate and playout frame rate so that the following objectives can ideally all be achieved:

1. **Maximize Frame Quality.** The video frame quality can be maximized by choosing a low encoder frame generation rate, which is due to (5.2.1).

2. **Minimize Playout Distortion.** The playout distortion can be minimized by setting the playout frame rate to the natural playout frame rate of the video, e.g. 30 fps is a common natural playout rate. It then follows that the playout distortion can be minimized by setting the playout frame rate as close to natural playout frame rate as possible.

3. **Steady Decoder Buffer Occupancy.** Having a steady decoder buffer occupancy reduces the probability of buffer underflow and thus preserves the video continuity. Additionally, actively maintaining a certain level of decoder buffer occupancy helps reduce the prebuffering required [13]. A steady decoder buffer occupancy can be achieved by adjusting the encoder frame generation rate and playout frame rate accordingly. For example, if the decoder buffer occupancy drops, the encoder frame generation rate can be increased to compensate the decoder buffer while the playout frame rate can be slowed down to ensure that the decoder buffer occupancy does not drop too fast and underflow.

These three objectives have possible conflicts with each other. As an example, if the bandwidth drops, we need to increase the encoder frame generation rate and reduce the playout frame rate to ensure a steady buffer occupancy. However, doing this will reduce the frame quality and increase playout distortion. Therefore, we can only settle for an optimal trade-off between these three objectives.
To find the optimal trade-off between these three objectives, we first formulate an optimization problem based on the encoder frame generation rate and playout frame rate as well as the three objectives.

We then derive optimization policies using Lyapunov optimization \[^{[93]}\]. These optimization policies will form the frame rate optimization framework. We also show that these policies can be decoupled to encoder and decoder optimization policies. That is, the encoder will adjust the encoder frame generation rate while the decoder independently adjusts its playout frame rate. The only information needed to be passed from the decoder to the encoder is the decoder buffer level and the available network bandwidth. We also show that under appropriate conditions, the decoupled problems are convex. We finally show that the derived optimization policies obtains a close to optimal trade-off between the three objectives while adapting to the time-varying network bandwidth.

5.3. System Model

The encoder frame generation rate \(i(t)\) and the video playout frame rate \(o(t)\) are assumed to be bounded as \(0 \leq i(t) \leq i_{\text{max}}\) and \(0 \leq o(t) \leq o_{\text{max}}\). It is further assumed that \(\lambda_{\text{max}} \leq i_{\text{max}}\), i.e. the maximum arriving frame rate at the decoder is assumed to be bounded by the encoder frame generation rate.

We define \(D(t)\) as the decoder buffer occupancy at time \(t\). We now introduce a concept of buffer stability as:

\[
(5.3.1) \quad \mathbb{E}\{D\} \triangleq \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{D(\tau)\} < \infty
\]

where \(\mathbb{E}\{D(.)\}\) denotes the expectation of the decoder buffer occupancy. That is the buffer is deemed stable if its long term averaged occupancy is finite. To aid in the formulation of the optimization problem, we simplify the model by assuming \(i(t) = \mu(t)\) and the arrival rate \(\lambda(t)\) is a random variable that depends on \(\mu(t)\). We also assume that there is an network throughput function \(W(\mu(t))\)\(^4\) that provides the arrival rate \(\lambda(t)\) based on a certain \(\mu(t)\), formally:

\[
(5.3.2) \quad W(\mu(t)) = \lambda(t)
\]

Since we assumed \(i(t) = \mu(t)\), which would mean (5.3.2) becomes:

\[
(5.3.3) \quad W(i(t)) = \lambda(t)
\]

\(^4\)Note that \(W(.)\) is not a deterministic function but a random function.
The resulting throughput function from (5.3.3) is assumed to be a continuous i.i.d random function. However, in this chapter, we specify:

$$W(i(t)) = l(t) \times i(t)$$

where $l(t)$ is the throughput scale factor. We further assume that $\lambda(t)$ is conditionally independent of the past history given the current $i(t)$ and $l(t)$. We estimate $l(t)$ at the decoder by:

$$l(t) = \frac{\lambda(t)}{i(t)}$$

$l(t)$ is then feedbacked to the encoder. Finally, we assume that the encoder, decoder and network works in slotted time where each time slot corresponds to the time needed to process a single frame, that is $t \in \{0, 1, 2, \ldots\}$.

5.4. Optimization Problem

Following from our above discussion, we now formulate an optimization problem to capture the aforementioned design considerations. Now let the functions $g(i(t))$ and $h(o(t))$ represent the frame quality and playout distortion respectively. $g(i(t))$ is a decreasing function of $i(t)$ while $h(o(t))$ is a decreasing function of $o(t)$. We will discuss specific choices for $g(i(t))$ and $h(o(t))$ in a section 5.6. Then, the optimization problem would be:

Maximize: $$\Psi(t) = (\alpha_2 W(i(t)) - \kappa o(t)) + \alpha_1 g(i(t)) - h(o(t))$$

Subject to:

$$0 \leq o(t) \leq o_{\text{max}}$$
$$0 \leq i(t) \leq i_{\text{max}}$$

where $\alpha_1$, $\alpha_2$ and $\kappa$ are some positive constants. The first term of $\Psi(t)$ maximizes the buffer occupancy by maximizing the difference between the rate of frames received $W(i(t))$ and the playout frame rate $o(t)$, which helps to minimize the probability of a buffer underflow. It can be seen that the first term can only be maximized by having a high $i(t)$ and low $o(t)$. However, a high $i(t)$ would result in a lower utility from $g(i(t))$, and $h(o(t))$ would also give a lower utility due to a low $o(t)$. Thus, the optimization policy would need to find the proper trade-offs to maximize (5.4.1).

To obtain an optimal solution that can dynamically adapt to the stochastic changes in network state and still provide stability would require Lyapunov

---

5Note that (5.3.4) is only needed to prove the analytical properties of the algorithm.
optimization. We will discuss in the next section how we construct a frame rate optimization policies using Lyapunov optimization.

5.5. Policy Derivation

In this section, we show how we derive the optimization policies using Lyapunov optimization for a joint buffer compensation and adaptive media playout system. We show that these policies can be decoupled into separate encoder and decoder policies. We also show how we normalize these policies.

5.5.1. Lyapunov Optimization

We first show how we convert the optimization problem presented in section 5.4 into a separate encoder and decoder optimization policies using Lyapunov optimization.

We assume that the encoder, decoder and network works in slotted time where each time slot corresponds to the time needed to process a single frame, that is $t \in \{0, 1, 2, \ldots\}$.

The decoder buffer dynamics at each slot will then be:

\[
D(t + 1) = [D(t) - o(t)]^+ + \lambda(t)
\]

We use the following Lyapunov function in this chapter:

\[
L(D(t)) \triangleq \frac{1}{2} D^2(t)
\]

We then define the one-step conditional Lyapunov drift $\Delta(D(t))$ as:

\[
\Delta(D(t)) \triangleq \mathbb{E}\{L(D(t + 1)) - L(D(t)) | D(t)\}
\]

By squaring the buffer dynamics equation (5.5.1) and taking expectations of the result we will get the following expression:

\[
\Delta(D(t)) \leq A - D(t) \mathbb{E}\{o(t) - \lambda(t) | D(t)\}
\]

Where:

\[
A = \frac{1}{2} (i_{\text{max}}^2 + o_{\text{max}}^2)
\]

\[\text{[5.5.5]}\]

\[\text{[5.5.4]}\]

\[\text{[5.5.3]}\]

\[\text{[5.5.2]}\]

\[\text{[5.5.1]}\]
By subtracting from both sides, the term $V \mathbb{E}\{\Psi(t)|D(t)\}$, which is the expectation of (5.4.1) scaled by a positive constant $V > 0$, and by rearranging the terms. We get:

$$\Delta(D(t)) - V \mathbb{E}\{\Psi(t)|D(t)\} \leq A - D(t)\mathbb{E}\{o(t) - \lambda(t)|D(t)\} - V \mathbb{E}\{\Psi(t)|D(t)\}$$

(5.5.6)

$$\leq A - \mathbb{E}\{\alpha_1 V g(i(t)) + \alpha_2 V \lambda(t) - D(t)\lambda(t)|D(t)\} - \mathbb{E}\{D(t)o(t) - V h(o(t)) - \kappa V o(t)|D(t)\}$$

By applying (5.3.3) and (5.3.4) to (5.5.6), we obtain:

$$\Delta(D(t)) - V \mathbb{E}\{\Psi(t)|D(t)\} \leq A - \mathbb{E}\{\alpha_1 V g(i(t)) + \alpha_2 V W(i(t)) - D(t)W(i(t))|D(t)\}$$

(5.5.7)

$$\leq A - \mathbb{E}\{D(t)o(t) - V h(o(t)) - \kappa V o(t)|D(t)\}$$

The key idea of the Lyapunov optimisation is to keep the left-hand-side (LHS) of (5.5.7) small. Note that (5.5.7) is the sum of Lyapunov drift $\Delta(D(t))$ and $-V$ times the expectation of the objective function (5.4.1). The minimization of Lyapunov drift will ensure that the buffer overflow will not likely occur. The minimization of $-V$ times the expectation of the objective (5.4.1) will ensure that the objective (5.4.1) is maximized. Since the minimization of the LHS of (5.5.7) is hard, Lyapunov optimisation chooses to minimize the upper bound of the LHS of (5.5.7), which is given by the right-hand-side (RHS) of (5.5.7).

Also note that in (5.5.7), the terms are rearranged such that the second RHS term is only a function of $i(t)$ while the last term is a function of $o(t)$ only. Thus, it can be seen from (5.5.7) that the last RHS term represents the decoder objective function and minimizing this term would provide an ideal playout frame rate. Likewise, the second RHS term represents the encoder objective function and minimizing this term would provide an ideal encoder frame generation rate. Notice that the frame adjustment policy is decoupled into separate optimization subproblems for playout frame rate $o(t)$ and encoder frame generation rate $i(t)$. Furthermore, under appropriate conditions, the decoupled problems are convex. This makes the problem easier and more flexible to solve.
5.5. POLICY DERIVATION

5.5.2. Frame Rate Optimization Policies

We will now state the frame adjustment policy based on the derivations from the previous section.

5.5.2.1. Decoder Policy

At each time slot, the decoder will observe the current buffer occupancy \( D(t) \) and choose \( o(t) \) as the solution of the following optimization:

\[
\begin{align*}
\text{Maximize:} & \quad D(t) o(t) - V h(o(t)) - \kappa V o(t) \\
\text{Subject to:} & \quad 0 \leq o(t) \leq o_{\text{max}}
\end{align*}
\]  
(5.5.8)

Note that the objective of (5.5.8) is directly taken from the third term of (5.5.7). The decoder will then adjust its playout frame rate to \( o(t) \). It can also be seen from (5.5.8) that the choice of \( o(t) \) depends mainly on the playout quality function \( h(o(t)) \) and the current decoder buffer occupancy \( D(t) \).

5.5.2.2. Encoder Policy

At each time slot, given the decoder buffer occupancy \( D(t) \) and the network loss probability, the encoder will choose \( i(t) \) as the solution of the following optimization:

\[
\begin{align*}
\text{Maximize:} & \quad \alpha_1 V g(i(t)) + \alpha_2 VW(i(t)) - D(t) W(i(t)) \\
\text{Subject to:} & \quad 0 \leq i(t) \leq i_{\text{max}}
\end{align*}
\]  
(5.5.9)

It can be seen that the objective of (5.5.9) is directly taken from the second term of (5.5.7). \( D(t) \) and the network loss probability can be obtained through regular feedbacks from the decoder. From (5.5.9), the choice of \( i(t) \) is based on the video quality function and the current decoder buffer occupancy \( D(t) \).

Notice that \( D(t) \) plays a different role in each subproblem. A low \( D(t) \) would cause \( i(t) \) to increase due to it being a negative term in (5.5.9) and decrease \( o(t) \) as it is a positive term in (5.5.8), while the opposite will happen when there is a high \( D(t) \). This is precisely the kind of behavior we would like to have as a frame rate adjustment technique.

5.5.3. Normalization

The optimization policies deal with different quantities such as the playout distortion function \( h(o(t)) \) and the decoder buffer \( D(t) \). These quantities need to be normalized in order to ensure more effective policies.
5. FRAME RATE OPTIMIZATION FRAMEWORK FOR VIDEO CONTINUITY

5.5.3.1. Decoder Policy Normalization

To normalize the decoder policy, we introduce a normalization term $\kappa$ such that (5.5.8) becomes:

$$\begin{align*}
\text{Maximize:} & \quad D(t)o(t) - Vh(o(t)) - \kappa Vo(t) \\
\text{Subject to:} & \quad 0 \leq o(t) \leq o_{max}
\end{align*}$$

(5.5.10)

We then take the first derivative of (5.5.10) with respect to $o(t)$ and set it to zero, we get:

$$D(t) - V\kappa - Vh'(o(t)) = 0$$

(5.5.11)

Next, we chose $D(t) = D_b$ and $o(t) = o_{max}$ so that (5.5.11) becomes:

$$D_b - V\kappa - Vh'(o_{max}) = 0$$

(5.5.12)

The intuition of (5.5.12) is that whenever the decoder buffer level reaches or exceeds $D_b$, the decoder policy will choose $o(t) = o_{max}$. $o_{max}$ is set to 30 fps (the natural playout frame rate) in this chapter, so $D_b$ can be seen as the ideal buffer level. Rearranging (5.5.12) to solve for $\kappa$ will produce:

$$\kappa = \frac{D_b - Vh'(o_{max})}{V}$$

(5.5.13)

5.5.3.2. Encoder Policy Normalization

We introduce the normalization terms $\alpha_1$ and $\alpha_2$ into (5.5.9) such that:

$$\begin{align*}
\text{Maximize:} & \quad \alpha_1 Vg(i(t)) + \alpha_2 Vl(i(t)) - D(t)l(i(t)) \\
\text{Subject to:} & \quad 0 \leq i(t) \leq i_{max}
\end{align*}$$

(5.5.14)

Taking the first derivative of (5.5.14) with respect to $i(t)$ and using (5.3.4) would give:

$$\alpha_1 Vg'(i(t)) + \alpha_2 Vl(t) - D(t)l(t) = 0$$

(5.5.15)

where $l(t)$ is defined in (5.3.5). We assume that $l(t) = 1$ for the derivations of $\alpha_1$ and $\alpha_2$, that is there are no variations on the throughput and, from (5.3.4), $W(i(t)) = i(t)$. This assumption is made to simplify the derivations.

Since we have two normalization terms, we need at least two equations to obtain a solution. We chose two cases:
5.6. VIDEO QUALITY FUNCTIONS

(1) When the decoder buffer reaches or exceed an ideal level $D_b$ (from (5.5.13)), we want the policy to choose $i(t) = i_{\min}$.

(2) When the decoder buffer declines to or goes below a minimum level $D_s$, we want the policy to choose $i(t) = i_{\max}$.

These two cases produces the following:

\[(5.5.16) \quad \alpha_1 V g'(i_{\min}) + \alpha_2 V l(t) - D_b l(t) = 0 \]
\[(5.5.17) \quad \alpha_1 V g'(i_{\max}) + \alpha_2 V l(t) - D_s l(t) = 0 \]

Rearranging (5.5.16), setting $l(t) = 1$ and solving for $\alpha_2$ yields:

\[(5.5.18) \quad \alpha_2 = \frac{D_b l(t) - \alpha_1 V g'(i_{\min})}{V} \]

Subtracting (5.5.17) from (5.5.16), setting $l(t) = 1$ and solving for $\alpha_1$ gives:

\[(5.5.19) \quad \alpha_1 = \frac{(D_b - D_s)}{g'(i_{\min}) - g'(i_{\max})} \]

5.6. Video Quality Functions

In this section, we examine suitable video quality functions to use in the frame rate optimization framework.
5. FRAME RATE OPTIMIZATION FRAMEWORK FOR VIDEO CONTINUITY

5.6.1. Frame Quality Function

![Frame utility function for football sequence.](image)

The frame utility function \( g(i(t)) \) should show the frame quality decreasing as the encoder frame generation rate \( i(t) \) increases. This is because a higher \( i(t) \) would result in lower average frame sizes, which will result in a lower average frame quality at a given available bit rate.

To begin with, it is well known that PSNR can be used to represent frame quality and can be modelled as [107]:

\[
PSNR(i(t)) = a \log(r(i(t))) + c
\]

Where \( a \geq 0 \) and \( c \) are the model coefficients, while \( r(i(t)) \) is the frame size defined in (5.2.1). However, (5.6.1) is a non-concave function due to (5.2.1) since by the second derivative test,

\[
\frac{\partial^2 PSNR}{\partial i^2} = \frac{a}{i^2(t)}
\]

Given that \( a \geq 0 \) and \( i(t) \geq 0 \), which means that (5.6.2) is always positive and (5.6.1) is convex, making it difficult to maximize. To try obtain a concave function, we make use of Ou et al’s frame quality function [108]. We then get our proposed frame utility function (see fig. 5.2):
(5.6.3) \[ g(i(t)) = 1 - \frac{1}{1 + e^{q(PSNR(i(t)) - s)}} \]

where \( q > 0 \) and \( s > 0 \) are some positive model constants, and with \( PSNR(t) \) defined in (5.6.1). Eqn (5.6.3) is still not yet a concave function as it is a sigmoidal function. However, it can be shown that (5.6.3) is concave if \( i(t) \) obeys the following condition:

(5.6.4) \[ i(t) \leq ABR(t) \cdot e^{-\frac{1}{2}(w+c+s)} \]

Where:

\[ w = \frac{1}{q} \cdot \log \left( \frac{1 + qa}{qa - 1} \right) \]

This upper bound on \( i(t) \) can intuitively be viewed as the minimum acceptable frame quality. With this bound, we can obtain a concave frame utility function, proposition C.1 in appendix C proves this.

![Figure 5.3. Upper bounds for i(t) over different ABR(t) for crew and football CIF sequence.](image)

To determine whether (5.6.4) produces bounds that are too low, we plotted the bounds over different \( ABR(t) \). Fig. 5.3 shows the bounds for \( i(t) \) calculated using (5.6.4) for two high motion CIF (352x288) sized sequences. Given that the sequences are in CIF resolution, the bounds produced are not too restrictive. While the bound of less than 15 fps at 200 kbps may seem too low, consider that the recommended maximum frame rate at 192 kbps is 7.5 fps for CIF sequences in the H.264/AVC standard [109]. Therefore, the bound is reasonably high enough and not too restrictive.
5. FRAME RATE OPTIMIZATION FRAMEWORK FOR VIDEO CONTINUITY

5.6.2. Playout Distortion Function

![Figure 5.4. Playout distortion function for football sequence.]

\[ h(o(t)) = m_s(o_n - o(t))^2 \]

Where \( m_s \) is the motion complexity of scene \( s \) and \( o_n \) is the natural playout frame rate of the sequence (i.e. the frame rate at which the sequence was filmed at). To determine the motion complexity we made use of the normalized frame difference describe by Ou et al \[110\], specifically:

\[ m_s = \frac{MAD}{Contrast} \]

Where \( MAD \) is the mean absolute difference between the current frame being encoded and the previous frame, \( Contrast \) is the mean standard deviation of the pixel values in each frame. The convexity of (5.6.5) can be verified by proposition D.1 in appendix D.

5.7. Performance Analysis

In this section, we derive the performance bounds for the frame rate optimization framework and prove its stability. To begin with, we assume that:
(5.7.1) \[ o_{\text{max}} \geq i_{\text{max}} \]

Equation (5.7.1) ensures that buffer stability is achievable. This is because \( \lambda_{\text{max}} \leq i_{\text{max}} \leq o_{\text{max}} \), so for every arriving frame rate \( \lambda(t) \) at the decoder, there is a playout frame rate \( o(t) \) that can be chosen to ensure that the buffer \( D(t) \) does not grow for all time slots. This property then enables the existence of at least one stationary randomized policy that chooses \( i(t) \) and \( o(t) \) over the time slots such that the buffer stability condition (5.3.1) can be satisfied. Note that (5.7.1) is only needed to prove the analytical properties of the algorithm.

To derive the performance bounds, we make use of Neely’s technique of introducing an \( \epsilon \)-optimal policy \[ \text{Idea} \], where \( \epsilon \) is some positive constant, i.e. \( \epsilon > 0 \). The \( \epsilon \)-optimal policy is designed in such a way that when \( \epsilon \to 0 \), the \( \epsilon \)-optimal policy converges to the optimal policy. Neely has shown that performing a Lyapunov drift analysis on the \( \epsilon \)-optimal policy allows lemma 2.2 in chapter 2 to be used. This, in turn, produces the performance bounds. The following theorem shows this.

**Theorem 5.8.** If the network throughput function \( W(i(t)) \) is i.i.d over the timeslots and the objectives are assumed to be bounded as \( \Psi(t) \leq \Psi_{\text{max}} \). Then implementing the optimization policies (5.5.9) and (5.5.8) in each timeslot stabilizes the decoder buffer \( D(t) \) and satisfies the following performance bounds:

\[
\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{D(\tau)\} \leq \frac{A + V \Psi_{\text{max}}}{o_{\text{max}}}
\]

\[
\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{\Psi(\tau)\} \geq \Psi^* - \frac{A}{V}
\]

Where \( V > 0 \) and \( \Psi^* \) maximizes (5.4.1) subject to the constraints (5.4.2) and (5.4.3), and \( A \) is defined as in (5.5.5).

**Proof.** Let the long term frame generation rate be \( \bar{i} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{i(t)\} \), the long term receiving frame rate be \( \bar{\lambda} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\lambda(t)\} \). Additionally, let there be a long term throughput scale factor \( \bar{l} \) such that, using (5.3.4) and (5.3.3), we can define the following:

\[
W(\bar{i}) = \bar{l} \times \bar{i} = \bar{\lambda}
\]

\[ \text{This is also known as a “near optimal policy” in Neely’s work.} \]
Suppose now that a stable $\epsilon$-optimal policy selects the long term playout frame rate $\bar{o}_\epsilon$ as:

$$\bar{o}_\epsilon \geq \bar{\lambda} + \epsilon$$

(5.8.4)

$$\epsilon \leq \bar{o}_\epsilon \leq o_{\max}$$

where $\epsilon > 0$. The above equation is valid because (5.7.1) ensures that $o_{\max} \geq \lambda_{\max}$ so there exists a policy that can choose the playout frame rate $o(t)$ over time such that (5.8.4) holds. Furthermore, (5.8.4) ensures that the stability condition (5.3.1) can be met. The $\epsilon$-optimal policy can be as a policy which stabilizes decoder buffer $D(t)$ and optimizes the following problem:

Maximize: $\Psi(t)$

Subject to: $\epsilon \leq o(t) \leq o_{\max}$

(5.8.5)

with $\Psi(t)$ defined in (5.4.1), then $\bar{o}_\epsilon$ forms part of the solution to (5.8.5).

Let the long term frame generation rate $i^*$ and long term playout frame rate $o^*$ be the solutions to the optimization problem (5.4.1) whose maximum is $\Psi^*$. Notice that due to (5.8.4), $\bar{o}_\epsilon$ is bounded as $\epsilon \leq \bar{o}_\epsilon \leq o_{\max}$ while $o^*$ is bounded as $0 \leq o^* \leq o_{\max}$. Furthermore, it can be shown that $o^* \leq \bar{o}_\epsilon$, because $o^*$ and $\bar{o}_\epsilon$ are respectively the solutions to the optimization problems (5.4.1) and (5.8.5) where these two optimization problems are identical except that $o(t)$ is lower bounded by $\epsilon$ in the $\epsilon$-optimal policy problem (5.8.5). This means any policy that produces $\bar{o}_\epsilon$ will have its long term average playout frame rate at least as high as a policy that produces $o^*$ as its long term average playout frame rate. Furthermore, it can be shown that $\bar{o}_\epsilon \to o^*$ as $\epsilon \to 0$. This is because (14):

$$\bar{o}_\epsilon \geq \left(1 - \frac{\epsilon}{o_{\max}}\right) o^* + \frac{\epsilon}{o_{\max}} \bar{o}_\epsilon \geq o^*$$

(5.8.6)

The middle term of (5.8.6) can be seen as an example of a mixed policy that picks $o^*$ with $1 - \frac{\epsilon}{o_{\max}}$ probability and $\bar{o}_\epsilon$ with $\frac{\epsilon}{o_{\max}}$ probability.

If $\epsilon \to 0$ and $\bar{o}_\epsilon \to o^*$, which implies that $i \to i^*$. This is because the playout frame rate $o(t)$ are bounded as $\epsilon \leq o(t) \leq o_{\max}$ in the $\epsilon$-optimal policy, so as $\bar{o}_\epsilon \to o^*$ the values for $o(t)$ become less restricted as $\epsilon$ gets smaller. Therefore, policies that can produce the solution $\Psi^*$ with the unrestricted $o^*$,
will produce $i^*$. Using the above properties of $\bar{o}_\epsilon$ and $\bar{i}$, we now attempt to derive the performance bounds. Based on (5.5.7), the Lyapunov one step drift bound of the $\epsilon$-optimal policy, with the long term playout rate $\bar{o}_\epsilon$ and long term frame generation rate $\bar{i}$, would be:

$$\Delta(D(t)) - V\mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq A - [\alpha_1 Vg(\bar{i}) + \alpha_2 VW(\bar{i}) - Vh(\bar{o}_\epsilon) - \kappa V\bar{o}_\epsilon]$$

$$- [D(t)\bar{o}_\epsilon - D(t)W(\bar{i})]$$

(5.8.7) $$\Delta(D(t)) - V\mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq A - \Psi_\epsilon - D(t) [\bar{o}_\epsilon - W(\bar{i})]$$

where:

(5.8.8) $$\Psi_\epsilon = \alpha_1 Vg(\bar{i}) + \alpha_2 VW(\bar{i}) - Vh(\bar{o}_\epsilon) - \kappa V\bar{o}_\epsilon$$

Based on (5.8.4), it means that the $\epsilon$-optimal policy can choose a particular $\bar{o}_\epsilon = \bar{\lambda} + \epsilon \{14\}$. Using this together with $W(\bar{i}) = \bar{\lambda}$ in (5.8.7) yields:

$$\Delta(D(t)) - V\mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq A - \Psi_\epsilon - D(t) [(W(\bar{i}) + \epsilon) - W(\bar{i})]$$

(5.8.9) $$\Delta(D(t)) - V\mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq A - \Psi_\epsilon - \epsilon D(t)$$

Since (5.8.9) is of a similar form to (2.8.10), we apply lemma 2.2 to (5.8.9). By taking expectations, summing over the timeslots $\tau \in [0..M-1]$ and using the non-negativity of $L(D(t))$ to drop the term $\mathbb{E}\{L(D(M-1))\}$, we get:

$$-\mathbb{E}\{L(D(0))\} - V \sum_{\tau=0}^{M-1} \mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq MA - \epsilon \sum_{\tau=0}^{M-1} \mathbb{E}\{D(\tau)\} - VM \Psi_\epsilon$$

(5.8.10) $$-\mathbb{E}\{L(D(0))\} - V \sum_{\tau=0}^{M-1} \mathbb{E}\{\Psi(t)\,|\,D(t)\}\leq MA - \epsilon \sum_{\tau=0}^{M-1} \mathbb{E}\{D(\tau)\} - VM \Psi_\epsilon$$

By using $\Psi(t) \leq \Psi_{\text{max}}$, dividing (5.8.10) by $M\epsilon$ and rearranging, we obtain:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{D(\tau)\} \leq \frac{A + V\Psi_{\text{max}}}{\epsilon} + \frac{\mathbb{E}\{L(D(0))\}}{M\epsilon}$$

(5.8.11) $$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{D(\tau)\} \leq \frac{A + V\Psi_{\text{max}}}{\epsilon} + \frac{\mathbb{E}\{L(D(0))\}}{M\epsilon}$$

Taking the limits of (5.8.11) as $M \to \infty$ and choosing $\epsilon = \alpha_{\text{max}}$ yields the decoder buffer bound (5.8.1). We chose $\epsilon = \alpha_{\text{max}}$ as it minimizes the
above bound. Furthermore, any particular choice of $\epsilon$ only affects the bound calculations and will not affect the optimization policies in any way.

To prove the objectives bound, we divide (5.8.10) by $MV$ and rearrange to get:

\begin{equation}
\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{\Psi(t)\} \geq \Psi_\epsilon - \frac{A}{V} + \frac{\mathbb{E}\{L(D(0))\}}{MV}
\end{equation}

Taking the limits of (5.8.12) as $M \rightarrow \infty$ would give us:

\begin{equation}
\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{\Psi(t)\} \geq \Psi_\epsilon - \frac{A}{V}
\end{equation}

The objectives bound (5.8.2) is then obtained by taking the limit of the above bound as $\epsilon \rightarrow 0$. This because $\bar{o}_\epsilon \rightarrow \bar{o}^*$ and $\bar{i} \rightarrow i^*$ as $\epsilon \rightarrow 0$ which implies that $\Psi_\epsilon \rightarrow \Psi^*$ as $\epsilon \rightarrow 0$. \hfill \Box

It can be seen from the bounds (5.8.1) and (5.8.2) that how close to optimality the solution would reach will depend on the parameter $V$. With the solution reaching close to the optimum as $V \rightarrow \infty$. However, a larger $V$ would result in a larger decoder buffer occupancy $D(t)$.

5.9. Performance Evaluation

5.9.1. Experiment Setup

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{dumbbell_topology}
\caption{Dumbbell topology with a 5 Mbits/s bottleneck link. A circle is an end-node.}
\end{figure}

We made use of ns-2 \cite{112} to simulate a network with time-varying data bandwidths. We implemented our framework into a x264 \cite{113} encoder, an open source multi-threaded H.264/AVC encoder. x264 is one of the more popular H.264/AVC encoders that is currently being used to encode videos for web services such as Youtube, Facebook and Vimeo. Our implementation in x264 simulates the network transmission using network traces. The decoder buffer evolution is simulated by tracking the arrivals of video frames from
5.9. PERFORMANCE EVALUATION

the simulated network and the removal of video frames from the buffer for playout. Everytime a frame is encoded, the framework will make a decision on the encoder frame generation rate and, at the simulated decoder, the playout frame rate. This is to simulate the real-time adjustment of parameters as the video is being encoded for transmission. Network traces obtained from ns-2 are used to simulate the network within the encoder. Note that our framework could easily be adapted to multiple pre-encoded copies of the video.

For the network simulations, we made use of a dumbell network topology with the bottleneck bandwidth set to 5 Mbits/s, see fig. 5.5 Video is streamed between two nodes in the network using the TFRC transport protocol [35], this ensures that the video streams are TCP-friendly. To generate background traffic, random webpage sessions were used for the other pairs of nodes. All the random sessions go through the bottleneck link. The webpage sessions are configured as, number of sessions is set to 400, inter-session distribution is an exponential distribution with its average set to 1, session size is set to 250, inter-page distribution is an exponential distribution with its average set to 15, page size is a Pareto distribution with its average set to 5 and shape 1.2, inter-object distribution is an exponential distribution with its average set to 0.01 and object size is a Pareto distribution with average 12 and shape 1.2.

The test sequence used is a concatenation of football, city, crew and akiyo sequences in that order. This is to ensure a mix of high and low motion within the sequence. Concatenating the sequences also allows us to minimize the experiment repetitions. A 16 minute test sequence is obtained by repeating the concatenated sequence. The parameters of (5.6.1) are determined via curve fitting as: $a = 4.91$ and $c = -1.424$. As in [108], we set $q = 0.34$. $s$ is calculated by averaging the $s$ of each sequence in [108] and found to be $s = 28.125$. Recall that $q$ and $s$ are model constants in (5.6.3).

For the parameters in the optimization policies, we empirically set: $V = 1000$, $D_b = 1500$ and $D_s = 500$. The video quality functions proposed in section 5.6 allows the encoder policy (5.5.9) and decoder policy (5.5.8) to be concave. To find the optimal solution, Newton’s method is implemented at the encoder and decoder to solve (5.5.9) and (5.5.8) respectively. While Newton’s method is an iterative search algorithm, the simulations have shown that it requires only a maximum of 4 iterations to obtain the optimal solution.

We also define a metric to measure the continuity of video delivery:

\[
(5.9.1) \quad \text{Continuity} = 1 - \frac{\text{rebuffering time}}{\text{total sequence time}}
\]
The rebuffering time is the time needed to refill the buffer to a certain threshold after an underflow event, which is a typical behavior of a video player [114]. The rebuffering threshold is set to half of the prebuffer amount. The amount of prebuffering will vary in our simulations to test the performance of each scheme. We name our framework \textit{FROpt}. We compare the our framework with a typical setup of x264 with its target frame rate and its playout rate set to a constant 30 fps, we label this scheme \textit{Norm}. We also compared our framework with the setup proposed in [13] by implementing a combination of AMP-Initial and AMP-Robust. We empirically chose the smallest slowdown factor of AMP such that it achieves a continuity of 99% or higher in each test case. This effectively simulates a close to optimal adaptive AMP scheme, we label this as \textit{AMP}. We also added in the results of \textit{AMP} with a constant slowdown factor of 25% (used in [13]) as a reference, which we label \textit{AMP25}.

5.9.2. Results

We compare the continuity of each scheme based on the amount of prebuffering provided. Continuity is calculated as in (5.9.1). The continuity results are shown in fig. 5.6. It can be seen that \textit{AMP} and \textit{FROpt} requires very little prebuffering compared to the rest. \textit{AMP} has its slowdown factor manually tuned for each test case to achieve 99% or higher continuity, so it can be seen as a close to the optimal performance possible for all adaptive media playout schemes. The optimization policies allow \textit{FROpt} to obtain 100% continuity without any manual tuning. The performance disparity between \textit{AMP25} and \textit{Norm} suggest that adaptive media playout schemes can help reduce the prebuffering by a fair amount.
Figure 5.6. Tradeoff between prebuffering delay and continuity. Prebuffer delay is on a log base 10 scale.

Figure 5.7. Tradeoff between prebuffering delay and playout distortion. Prebuffer delay is on a log base 10 scale.
Figure 5.8. $i(t)$ and $o(t)$ over the time slots for FROpt with 8.5s prebuffer delay. Natural playout frame rate set to 30 fps.

Figure 5.9. $i(t)$ and $o(t)$ over the time slots for AMP with 8.5s prebuffer delay. Natural playout frame rate set to 30 fps.
We next look at the playout distortion performance of each scheme. Playout distortion is calculated and averaged for each scheme using (5.6.5) \((o_n = 30 \text{ fps})\). Fig. 5.7 shows the results for this. Since playout distortion is created only when the playout frame rate is adjusted, \(Norm\) has zero playout distortion in all the test cases. \(FROpt\) produces a lower playout distortion than \(AMP\) in almost all cases. This is because by performing a joint buffer compensation and adaptive media playout adjustment, \(FROpt\) can choose to reduce the playout
slowdowns by increasing the buffer compensation. Figures 5.8, 5.9, 5.10 and 5.11 shows the evolution of the encoder frame generation rate \( i(t) \) and the playout frame rate \( o(t) \) over the time slots. Figures 5.8 and 5.9 shows \( FROpt \) and \( AMP \) respectively with 8.5s prebuffer delay. Figures 5.10 and 5.11 shows \( FROpt \) and \( AMP \) respectively with 34s prebuffer delay. From 8.5s to 34s prebuffer delays, \( FROpt \) showed a reduction in playout distortion in fig. 5.7. This can be seen from figs. 5.8 and 5.10 as \( o(t) \) shows a reduced amount of deviation from the natural playout frame rate (30 fps). Figs. 5.9 and 5.11 also show that \( AMP \) tends to introduce playout distortion for a longer period of time.

Increasing buffer compensation comes at a cost of frame quality. We compare the PSNR of the encoded video before transmission, the results of which is shown in fig. 5.12. \( AMP \) have the same PSNR characteristic as \( Norm \) and is not shown on the graph, this is because \( AMP \) does not its encoding strategy. The encoded PSNR was used since it shows how the encoder optimization policy affects the compressed video before any drops in PSNR occurring due to transmission. The PSNR results show that the maximum PSNR loss in the test cases was less than 0.3 dB. The loss is considered negligible when compared to the gains in continuity and playout distortion. We also compared the PSNR at the receiver in fig. 5.13, again \( AMP \) have the same PSNR characteristic as \( Norm \) and is not shown. The receiver PSNR results show that the maximum PSNR gain of \( Norm \) over \( FROpt \) has increased to more than 0.4 dB. This is because \( FROpt \) packs more video data into each packet when it is attempting buffer compensation, thus each packet loss will have a greater impact on the frame quality of the video. However, the PSNR gain of \( Norm \) is still comparatively negligible when compared to the gains in continuity and playout distortion.

The frame quality function (5.6.3) is derived based on the mean opinion scores (MOS) from the viewers [108]. Since it is laborious to get MOS for all sequences that are streamed, the parameters for (5.6.3) are likely to be estimated. A question that can be asked is: how does the frame rate optimization framework performs if the parameters to (5.6.3) are slightly wrong?

To answer this question, we first note that the parameter \( q \) from (5.6.3) is shown to be fixed for different sequences [108]. Also, since \( a \) and \( c \) from (5.6.1) are estimated via curve fitting on a rate-PSNR graph, it allows them to be calculated directly from the sequence. This leaves parameter \( s \) from (5.6.3) to be estimated. To examine the performance of a slightly wrong \( s \), we tested \( FROpt \) using \( s = 30.65 \) and \( s = 25.9 \) as opposed to the originally used \( s = 28.125 \). \( s = 30.65 \) and \( s = 25.9 \) correspond to the \( s \) parameters of
the akiyo and football respectively [108]. Note that akiyo and football are concatenated as part of the larger test sequence.

Figures (5.14), (5.15) and (5.16) shows the comparisons of \textit{FROpt} with different $s$ parameters. It can be seen that there is very little difference in performances. This shows that the frame rate optimization framework is robust enough for an estimated $s$ parameter.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{sender_psnr.png}
\caption{Sender side PSNR. Note that \textit{AMP} has the same PSNR characteristic as \textit{Norm} and thus, is not shown here.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{receiver_psnr.png}
\caption{Receiver side PSNR. Note that \textit{AMP} has the same PSNR characteristic as \textit{Norm} and thus, is not shown here.}
\end{figure}
Figure 5.14. Continuity vs prebuffer delay graph of FROpt with different $s$ parameter.

Figure 5.15. Playout distortion vs prebuffer delay graph of FROpt with different $s$ parameter.
5.10. Conclusion

In this chapter, we look at preserving video continuity with small prebuffering. We approach this by a joint adjustment of encoder frame generation rate and playout frame rate. We showed that doing a joint adjustment amounts to finding the optimal trade-off between frame quality, playout distortion and steady buffer occupancy. We then formulated an optimization problem that describes this trade-off and made use of Lyapunov optimization to derive optimization policies, thus creating the frame rate optimization framework. We show that these policies can be decoupled into separate encoder and decoder optimization policies with feedback between the two. Results shows that the framework improves over current approaches in terms of prebuffering delay required and playout distortion with a minor trade-off in PSNR.

Note that this framework mainly requires the encoder and decoder to be modified and since these sit at the application layer, the modifications can be implemented in a straightforward manner. The framework does not require any changes to the network protocols other than a need to receive feedback on the current network bandwidth from the transport layer. The framework could potentially be implemented on an existing video streaming system via modifications to the streaming server and patches to the clients of the system.

However, the frame rate optimization framework still have some unresolved issues. One possible issue in this chapter that might arise is that the frame quality function is tied to the quality metric proposed by Ou et al [108] (see eq. (5.6.3)). This means that popular PSNR-based frame quality functions

---

**Figure 5.16.** PSNR vs prebuffer delay graph of FROpt with different $s$ parameter.
such as the rate distortion models \cite{45,115} and the PSNR utility model \cite{107} cannot be used. This problem came about because we used the encoder frame generation rate as input to the frame quality function (see eq. (5.2.1)). Using the encoder frame generation rate converts a PSNR-based frame quality function into a problem of maximizing a convex function, which is hard. Ou et al’s quality metric also requires a bound on the encoder frame generation rate to remain as a concave function. Note that from (5.6.4), one of the conditions for concavity is $qa - 1 > 0$. Since $q$ is assumed by Ou et al to be fixed to 0.34 \cite{108}. This means that if a sequence happens to have $a \leq \frac{1}{0.34}$, the frame quality function will be convex regardless of other conditions.

Another issue in this chapter is the impact of network delay on the optimization policies. Since the encoder relies on information from the decoder to perform its optimization, network delay may have an effect on the optimization policies.

A final issue is that playout slowdowns contributes to the viewing latency of the system. The viewing latency might need to be bounded to be a certain latency requirement. This is especially so for delayed broadcast video applications where the provider wants a certain maximum viewing latency for the viewers. This means that the playout slowdowns might need to be constrained to meet the viewing latency requirements.

In the next chapter, we look into improving the framework by resolving the above issues.
Improvements To The Frame Rate Optimization Framework

6.1. Introduction

In the previous chapter, we proposed a frame rate optimization framework that performs frame rate control based on Lyapunov optimization. At its conclusion, we highlighted several existing issues with the framework, namely:

- The frame quality function is restricted to the function proposed by Ou et al. [108] because the encoder frame generation rate has an inverse relationship with the frame quality metric (see (5.6.1) in chapter 5). This is a problem because of the following:
  - some of the more popular frame quality functions cannot be used.
  - certain bounds on the constants and variable need to be met at all times to ensure the concavity of the frame quality function.

- Playout slowdowns contribute to the viewing latency of the streaming system. This is not ideal when the system has a delay budget to adhere to.

- The assumption of no network delay when formulating the optimization policies. The impact of network delay on the optimization policies is not studied, particularly the feedback delay between the receiver and sender.

In this chapter, we look into resolving the above issues of the framework. To solve the inverse relationship issue, we reformulated the optimization problem to make use of the inverse encoder frame generation rate and the inverse playout frame rate. We call the inverse encoder frame generation rate as the encoder frame generation interval and the inverse playout frame rate as the playout frame interval in this chapter. We then introduce a virtual buffer that describes the dynamics of the encoder frame generation interval and the playout frame interval. We also introduce a delay constraint into the framework in an attempt to reduce the accumulated delay caused by playout slowdowns. We then study the impact of delayed feedback on the optimization policies and derive policies that deal with the delayed feedback between the receiver to the

93
sender. Finally, we evaluate the effectiveness of our approach and prove that using our approaches would yield certain performance bounds.

The contributions of this chapter are:

1. Reformulation of the frame rate optimization problem to allow the more popular frame quality functions to be used. We then show that the reformulated framework produces a very low playout distortion in addition to a significant reduction in the prebuffering requirements compared to existing approaches.

2. A delay constraint that reduces the accumulated delay from playout slowdowns. We then show that the delay constrained framework provides a superior tradeoff between the video quality and the delay introduced compared to the existing approach.

3. An analysis of the impact of delayed feedback between the receiver and the sender. We show that the delayed feedbacks have a minimal impact on the optimization policies.

This chapter is organized as follows:

Section 6.2 discusses the reformulation of the frame rate optimization problem. Section 6.3 demonstrates how a delay constraint can be introduced into the framework. Section 6.4 analyzes the delayed feedback on the optimization policies. Section 6.5 describes the video quality functions that can be used with this framework. Section 6.6 analyzes the performance bound for the optimization policies. Section 6.8 evaluates the performance of the proposed improved framework. Section 6.9 concludes this chapter.

6.2. Discontinuity Penalty For Frame Rate Optimization

In this section, we look into solving the inverse relationship between the encoder frame generation rate and the frame quality metric in \((5.6.1)\). We approach this by inverting the encoder frame generation rate. While this solves the inverse relationship problem in \((5.6.1)\), the system model presented in chapter 5 section 5.3 no longer makes sense. This suggests that a new system model and, consequently, a reformulation is required.

Therefore, we propose to invert both the encoder frame generation rate and the playout frame rate to obtain the encoder frame generation interval and playout frame delay interval respectively, and redesign the buffer dynamics to describe the processes in terms of frame intervals.

The new buffer dynamics makes use of a virtual buffer, which we call the discontinuity penalty. This is essentially a value representing the accumulated difference between the receiving frame interval and the playout delay interval,
6.2. DISCONTINUITY PENALTY FOR FRAME RATE OPTIMIZATION

and can be seen as an indication of the underflow probability. Lyapunov optimization is then used to derive optimization policies to ensure that buffer underflows are minimized while the video quality is maximized.

6.2.1. Buffering Criteria

\[ b(t) \geq \sum_{\tau=t}^{T} o(\tau) - \sum_{\tau=t}^{T} \lambda(\tau) \quad \forall T \geq t \]

Equation (6.2.1) intuitively means that, to avoid a buffer underflow, the cumulative buffer drainage for the rest of the sequence playing time should not exceed the current buffer occupancy \( b(t) \).

Now we refine the above model to make use of frame intervals, fig. [6.2] illustrates this. This is done to build up a system model using frame intervals. Basically to make use of the frame intervals, we set \( \lambda(t) = \frac{1}{r(t)} \) and \( o(t) = \frac{1}{p(t)} \). Equation (6.2.1) then becomes:
\[ b(t) \geq \sum_{\tau=t}^{T} \frac{1}{p(\tau)} - \sum_{\tau=t}^{T} \frac{1}{r(\tau)} = \sum_{\tau=t}^{T} \left( \frac{1}{p(\tau)} - \frac{1}{r(\tau)} \right) = \sum_{\tau=t}^{T} \frac{r(\tau) - p(\tau)}{r(\tau)p(\tau)} \] (6.2.2)

We assume that \( r_{\text{min}} \leq r(t) \leq r_{\text{max}} \) and \( p_{\text{min}} \leq p(t) \leq p_{\text{max}} \), i.e. both \( r(t) \) and \( p(t) \) are bounded. That would mean that we can approximate (6.2.2) as:

\[ b(t) \geq \sum_{\tau=t}^{T} \frac{r(\tau) - p(\tau)}{r_{\text{max}}p_{\text{max}}} \] (6.2.3)

If we divide (6.2.3) by \( T - t \), the remaining time slots left, we can estimate the buffer underflow bound per time slot. This will result in:

\[ \frac{b(t)r_{\text{max}}p_{\text{max}}}{T - t} \geq \hat{r} - \hat{p} \] (6.2.4)

Where \( \hat{r} \) and \( \hat{p} \) are the averages of \( r(t) \) and \( p(t) \) respectively. Equation (6.2.4) provides us with a way to design the optimization policy to avoid buffer underflow in a time slot. Since Lyapunov optimization works on a per time slot basis, we prefer (6.2.4) over (6.2.3). Essentially, we need to design a policy that produces a receiving frame interval \( r(t) \) and a playout frame interval \( p(t) \) in such a way that the above bound is met.

6.2.2. System Model

![System model showing the frame intervals.](image)

We now show how \( r(t) \) and \( p(t) \) are produced in the complete system model. Fig. 6.3 illustrates the system model. In a time slot \( t \), the encoder at the sender produces video frames at intervals of \( f(t) \) and stores them into the
sender buffer. The sender sends the video frames in its buffer at intervals of \( s(t) \). Note that \( f(t) \) and \( s(t) \) are the encoder frame generation interval and the sending frame interval respectively. Simultaneously, the receiver receives video frames from the sender at intervals of \( r(t) \) and puts it into the decoder buffer. The decoder in the receiver plays out the video frames at intervals of \( p(t) \) to the viewer, \( p(t) \) is the playout frame interval. The network will also produce a forward delay of \( d_f \) and a backward delay of \( d_b \).

The goal of our framework is to jointly adjust the encoding frame interval \( f(t) \) and the playout frame interval \( p(t) \) to maintain video continuity. Adjustment of \( p(t) \) at the decoder can be done by setting the playout interval of the frames to \( p(t) \). \( f(t) \) can be adjusted at the encoder by setting the encoding frame rate for the rate controller to \( \frac{1}{f(t)} \). Thus, for a smaller \( f(t) \) at the same available bitrate \( ABR(t) \), the rate controller will compress the frames more and the sender can then send more frames to the receiver.

To simplify the model, we assume that \( f(t) = s(t) \) and \( f(t) \) is bounded within the range \([f_{\text{min}}, f_{\text{max}}]\). We also assume that \( r(t) \) is defined by a network delay variation function \( F(s(t)) \):

\[
\text{(6.2.5)} \quad r(t) = F(s(t))
\]

\( F(s(t)) \) basically tries to represent the delay variation produced by the network. By using \( f(t) = s(t) \), (6.2.5) becomes:

\[
\text{(6.2.6)} \quad r(t) = F(f(t))
\]

This means that we can represent the delay variation based on the encoding frame interval \( f(t) \). In this chapter, we specify \( F(f(t)) \) as:

\[
\text{(6.2.7)} \quad F(f(t)) = e(t) \times f(t)
\]

Where \( e(t) \) is the frame interval scaling factor due to delay variations from the network. In practice, we estimate \( e(t) \) at the receiver by:

\[
\text{(6.2.8)} \quad e(t) = \frac{r(t)}{f(t - d_f)}
\]

Where \( d_f \) is the forward delay between the sender and receiver. \( e(t) \) in (6.2.8) is essentially the inverse counterpart of the throughput scale factor \( l(t) \) described in chapter 5 in (5.3.5). Note that if there is no delay (i.e. \( d_b = d_f = 0 \)), it will mean that \( F(f(t)) = r(t) \).
6.2.3. General Optimization Problem

There are three main objectives that we want to optimize: 1. frame quality, 2. playout distortion and 3. continuity. Since we are adjusting both \( f(t) \) and \( p(t) \), we want to ensure that the frame quality does not degrade too much and the playout distortion does not get too high as both have an impact on the viewer’s perception of the video. We ensure continuity by providing sufficient video data at the receiver for playing at all times.

With the system model described in the previous section, we now formulate a general optimization problem:

\[
\begin{align*}
\text{Maximize:} & \quad w_1 g(f(t)) - w_2 h(p(t)) \\
\text{Subject to:} & \quad \mathbb{E}\{F(f(t)) - p(t)\} < \beta(t) \\
\beta(t) & = \frac{b(t) r_{\text{max}} p_{\text{max}}}{T - t} \\
f_{\text{min}} & \leq f(t) \leq f_{\text{max}} \\
p_{\text{min}} & \leq p(t) \leq p_{\text{max}}
\end{align*}
\]

Where \( w_1 > 0 \) and \( w_2 > 0 \) are positive constants, \( g(f(t)) \) is the frame quality function and \( h(p(t)) \) is the playout distortion function. Both \( g(f(t)) \) and \( h(p(t)) \) are non-negative functions, i.e. \( g(f(t)) \geq 0 \) and \( h(p(t)) \geq 0 \). \( g(f(t)) \) is an increasing function of \( f(t) \) while \( h(p(t)) \) is a convex function of \( p(t) \). We will suggest a specific form for \( g(f(t)) \) and \( h(p(t)) \) later in section 6.5. Maximizing the objective (6.2.9) would maximize the frame quality function \( g(f(t)) \) while minimizing the playout distortion function \( h(p(t)) \). The constraint (6.2.10) is the continuity constraint derived using (6.2.4). Constraints (6.2.12) and (6.2.13) are the limits set on \( f(t) \) and \( p(t) \) respectively.

The continuity constraint (6.2.10) can be satisfied by either decreasing \( f(t) \) or increasing \( p(t) \). However, decreasing \( f(t) \) would result in a lower frame quality given by \( g(f(t)) \) and increasing \( p(t) \) would result in a higher playout distortion given by \( h(p(t)) \). The optimization policy would need to handle these tradeoffs.

To solve this optimization problem, we make use of the concepts of virtual buffer and Lyapunov optimization.

6.2.4. Virtual Buffer And Stability

Virtual buffers are a concept introduced by Neely et al. \[14\] to replace certain constraints of an optimization problem. To determine a suitable virtual buffer for our problem, we make an initial virtual buffer design to represent the
continuity of the video. The virtual buffer will be updated at every time slot \( t \), and is updated as:

\[
U(t+1) = [U(t) - p(t)]^+ + F(f(t))
\]

Where \( U(0) = 0 \). The virtual buffer \( U(t) \) is lower bounded by 0 (i.e. always non-negative). This virtual buffer represents the difference between the receiving frame interval \( F(f(t)) \) and the playout frame interval \( p(t) \), and represents the discontinuity penalty. If \( F(f(t)) \) is higher than \( p(t) \), it means that the network throughput is lower than the playout rate, i.e. the rate of video frames received is slower than the amount of video frames being played out. Thus, the discontinuity penalty \( U(t) \) accumulates the difference of \( F(f(t)) - p(t) \) as a penalty. If the network subsequently improves and \( F(f(t)) \) is lower than \( p(t) \), the penalty in the buffer then reduces by \( p(t) - F(f(t)) \). The higher \( U(t) \) becomes, the higher the possibility of a buffer underflow. In an ideal network, where \( F(f(t)) \leq p(t) \) at all times, \( U(t) \) will never accumulate any penalty.

To ensure that \( U(t) \) does not keep increasing, we want \( U(t) \) to stabilize. We will show later that by stabilizing \( U(t) \), we can ensure that the video continuity is preserved. The definition of stability used here is:

\[
\mathbb{E}\{U\} \triangleq \lim_{t \to \infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U(\tau)\} < \infty
\]

Equation (6.2.15) intuitively means that the buffer is stable if it does not grow infinitely large over time. We also like the virtual buffer to meet the continuity constraint (6.2.10) when it stabilizes. To do that, we extend the initial virtual buffer (6.2.14) as:

\[
U(t+1) = [U(t) - p(t) - \beta(t)]^+ + F(f(t))
\]

To see how it works, notice that for the discontinuity penalty \( U(t) \) to grow infinitely, the following condition needs to be met: \( F(f(t)) - p(t) > \beta(t) \). Therefore, in order for \( U(t) \) to stabilize the following condition needs to be true: \( F(f(t)) - p(t) \leq \beta(t) \). This is the continuity constraint as defined in (6.2.10), so when \( U(t) \) stabilizes, the continuity constraint will be met. The idea of using a virtual buffer to satisfy a constraint was introduced by Neely et al [14]. The discontinuity penalty \( U(t) \) is maintained at the receiver in our design.
Thus, we can restate the optimization problem presented in section 6.2.3 as:

\[
\begin{align*}
\text{Maximize:} & \quad w_1 g(f(t)) - w_2 h(p(t)) \\
\text{Subject to:} & \quad U(t) \text{ is stable} \\
& \quad f_{\text{min}} \leq f(t) \leq f_{\text{max}} \\
& \quad p_{\text{min}} \leq p(t) \leq p_{\text{max}}
\end{align*}
\] (6.2.17) (6.2.18) (6.2.19) (6.2.20)

where \( U(t) \) is updated using (6.2.16), the continuity constraint (6.2.10) is replaced with the constraint that \( U(t) \) stabilizes.

6.2.5. Lyapunov Optimization Derivation

We show here how we convert the optimization problem presented in section 6.2.4 into a separate sender and receiver optimization policies using Lyapunov optimization.

We assume that there is no network delay between the sender and receiver (i.e. \( d_f = d_b = 0 \)). This is to simplify the analysis presented here. We will relax this assumption in the next section.

We define a Lyapunov function \( L(U(t)) \) to represent the "energy" of the discontinuity penalty \( U(t) \) at time \( t \). We use the following Lyapunov function:

\[ L(U(t)) \triangleq \frac{U^2(t)}{2} \] (6.2.21)

We then define the one-step conditional Lyapunov drift \( \Delta(U(t)) \) as:

\[ \Delta(U(t)) \triangleq \mathbb{E}\{L(U(t + 1)) - L(U(t)) | U(t)\} \] (6.2.22)

Equation (6.2.22) can be understood as the expected change in the energy in one time slot step. The goal of Lyapunov optimization is to show that this energy reduces or stays the same at each slot (i.e. (6.2.22) produces a negative or zero drift). This would ensure the stability of the buffer, which in turn enforces the continuity of the video playout. To show that \( U(t) \) stabilizes, we need to convert the buffer update equation (6.2.16) into a one-step drift (6.2.22).

To do that, we square the buffer update equation (6.2.16), divide it by two, and take expectations of the result we will get the following expression:\footnote{Detailed derivations in appendix B.2, see also lemma 2.1 in chapter 2}
\[ \Delta(U(t)) \leq B - U(t)\mathbb{E}\{\beta(t) + p(t) - F(f(t))\} \]

Where \( B \) is a constant defined as:

\[ B = \frac{1}{2}\left( r_{\text{max}}^2 + \left( p_{\text{max}} + \frac{T}{T-t} r_{\text{max}} p_{\text{max}} \right)^2 \right) \]

It can be seen that (6.2.23) is of a similar form to (2.8.10) in chapter 2. As a result, we can apply lemma 2.2 to (6.2.23) to show that \( U(t) \) stabilizes, which will be proven later in theorem 6.7 in section 6.6. Once \( U(t) \) can be proven to stabilize, the continuity constraint (6.2.10) can be satisfied by using lemma 2.3 in chapter 2. However, to optimize the frame quality utility \( g(f(t)) \) and the playout distortion \( h(p(t)) \), we need to massage the equation more. By subtracting from both sides, the term \( \mathbb{E}\{w_1 g(f(t)) - w_2 h(p(t))\} \), which is the expectation of (6.2.17) scaled by a positive constant \( V > 0 \), and by rearranging the terms. We get:

\[ \Delta(U(t)) - V\mathbb{E}\{w_1 g(f(t)) - w_2 h(p(t))\} \leq B - \mathbb{E}\{U(t)\beta(t)\} - \mathbb{E}\{V w_1 g(f(t)) - U(t) F(f(t))\} - \mathbb{E}\{U(t) p(t) - V w_2 h(p(t))\} \]

The third RHS term of (6.2.25) is a function of the encoder frame generation interval \( f(t) \) only and represents the sender optimization policy. The last term of (6.2.25) is a function of the playout frame interval \( p(t) \) and represents the receiver optimization policy.

To summarize:

6.2.5.1. Encoder optimization policy

From the third term of (6.2.25), based on the frame interval scaling factor \( e(t) \) and discontinuity penalty \( U(t) \) feedback from the receiver. The encoder in the sender will calculate \( F(f(t)) \) using (6.2.7), and it will choose \( f(t) \) at each time slot as the solution of the following optimization:

Maximize: \[ V w_1 g(f(t)) - U(t) F(f(t)) \]

Subject to: \[ f_{\text{min}} \leq f(t) \leq f_{\text{max}} \]
6.2.5.2. Decoder optimization policy

From the last term of (6.2.25), the decoder in the receiver will observe the current discontinuity penalty $U(t)$ and choose $p(t)$ at each time slot as the solution of the following optimization:

Maximize: $U(t)p(t) - Vwzh(p(t))$
Subject to: $p_{\text{min}} \leq p(t) \leq p_{\text{max}}$

(6.2.27)

Notice that the optimization policies are decoupled into separate optimization subproblems for playout frame and encoding frame intervals. Furthermore, under appropriate conditions, the decoupled problems are convex\(^2\). This makes the problem easier and more flexible to solve.

6.3. Delay Constrained Frame Rate Optimization

While slowing down the video playout allows us to reduce the occurrences of buffer underflows and preserve the video continuity, it introduces extra viewing latency to the viewers. This becomes an issue when the system has a delay budget, as frequent playout slowdowns might cause the delay budget to be exceeded.

In this section, we examine the problem when there is a constraint imposed on the viewing latency. We focus on constraining the additional playout latency generated by AMP. More specifically, we want to impose a constraint on how often playout slowdowns occur and how much the playout can be slowed down. This is done by introducing another virtual buffer, called the delay accumulator, into the problem to represent the constraint on the accumulated delay due to playout slowdowns. We then use Lyapunov optimization to ensure that this constraint is met.

6.3.1. Delay Constrained Policy Design

The delay constraint can be described as constraining the accumulated playout slowdowns used over the lifetime of the whole video sequence. Specifically, let $\theta$ be the maximum playout slowdown delay tolerable for the video application and $p_n$ represent the natural playout interval of the video. Then, the constraint could be written as:

(6.3.1) $\sum_{\tau=0}^{T-1} (p(\tau) - p_n) \leq \theta$

\(^2\)This will be discussed further in section 6.5 of this chapter.
Recall that $T$ is the length of the sequence in time slots. Since Lyapunov optimization works on a per time slot basis, we need to express the constraint (6.3.1) as a constraint for each time slot. To do this, Note that for the current timeslot $t$, the averaged maximum playout slowdown delay tolerable will be:

\[(6.3.2)\]

\[t_d = \frac{\theta}{T-t}\]

where $T-t$ represents the number of time slots remaining for the sequence. Using (6.3.2), we can rewrite the delay constraint (6.3.1) as:

\[(6.3.3)\]

\[E\{p(t) - p_n\} \leq t_d\]

Then, what remains to be done is to design a policy that ensures that constraint (6.3.3) is met at each time slot. It can be seen that by adding in constraint (6.3.3) to the optimization problem presented in section 6.2.4 would yield:

\[(6.3.4)\]

Maximize: $w_1g(f(t)) - w_2h(p(t))$

\[(6.3.5)\]

Subject to: $U(t)$ is stable

\[(6.3.6)\]

$E\{p(t) - p_n\} \leq t_d$

\[(6.3.7)\]

$f_{\text{min}} \leq f(t) \leq f_{\text{max}}$

\[(6.3.8)\]

$p_{\text{min}} \leq p(t) \leq p_{\text{max}}$

6.3.2. Delay Accumulator Virtual Buffer

To apply the delay constraint (6.3.3) into the framework using Lyapunov optimization, we again make use of the virtual buffer concept. We introduce another virtual buffer named as the delay accumulator. The delay accumulator is a virtual buffer that keeps track of the accumulated delay caused by playout slowdowns. Everytime a playout slowdown is perform, the delay accumulator will increase correspondingly. However, the delay accumulator will reduce when playout speed up is perform. Formally, the buffer dynamics to describe the delay accumulator for each time slot would be:

\[(6.3.9)\]

\[X(t+1) = [X(t) - p_n - t_d]^+ + p(t)\]

Note that the terms $p_n$, $p(t)$ and $t_d$ are taken from constraint (6.3.3) above. Therefore, by showing that the delay accumulator $X(t)$ stabilizes and using lemma 2.3 from chapter 2, we can show that the delay constraint (6.3.3) can
be satisfied. This also means that the optimization problem presented in the
previous section can be rewritten as:

\[
\begin{align*}
\text{(6.3.10)} & \quad \text{Maximize: } w_1 g(f(t)) - w_2 h(p(t)) \\
\text{(6.3.11)} & \quad \text{Subject to: } U(t) \text{ is stable} \\
\text{(6.3.12)} & \quad X(t) \text{ is stable} \\
\text{(6.3.13)} & \quad f_{\min} \leq f(t) \leq f_{\max} \\
\text{(6.3.14)} & \quad p_{\min} \leq p(t) \leq p_{\max}
\end{align*}
\]

We now show how Lyapunov optimization can be used to derive optimization
policies to solve the above problem.

6.3.3. Lyapunov Optimization Derivation

Note that there are two buffers in the problem, the discontinuity penalty \( U(t) \)
and the delay accumulator \( X(t) \). To stabilize both of these simultaneously, we
first redefine the Lyapunov function to be:

\[
\text{(6.3.15)} \quad L(U(t), X(t)) \triangleq U^2(t) + w_3 X^2(t)
\]

where \( w_3 \) is some positive constant. The one step conditional drift also
needs to consider both buffers and is defined as:

\[
\Delta(U(t), X(t)) \triangleq \mathbb{E}\{L(U(t + 1), X(t + 1)) - L(U(t), X(t)) | U(t), X(t)\}
\]

To shorten the formulas, we use \( \Delta, U, X, p \) and \( f \) to represent \( \Delta(U(t), X(t)), U(t), X(t), p(t) \) and \( f(t) \) respectively. By squaring \( \text{(6.2.16)} \) and \( \text{(6.3.9)} \), multiply by \( w_3 \), taking expectations and dividing by 2\(^3\) we get:

\[
\Delta(U, X) \leq B + C - U \mathbb{E}\{p - F(f) | U, X\} - w_3 X \mathbb{E}\{p_n + t_d - p | U, X\}
\]

Where \( B \) is defined as in \( \text{(6.2.24)} \) and \( C \) is:

\[
\text{(6.3.18)} \quad C = \frac{w_3}{2} \left( p_{\max}^2 + (p_n + t_d)^2 \right)
\]

\(^3\text{Detailed derivations in appendix B.3 see also lemma 2.1 in chapter 2}\)
It can be proven that \((6.3.17)\) results in stability for both the discontinuity \(U\) and the delay accumulator \(X\) by applying lemma 2.2 from chapter 2 to \((6.4.20)\), which will be proven later in corollary 6.1. Furthermore, it can be proven that stabilization of \(X\) implies that the constraint \((6.3.3)\) can be met.

To optimize the frame quality and the playout distortion, we subtract \(\{w_1 g(f(t)) - w_2 h(p(t))|U(t)\}\) from both sides of \((6.3.17)\) to obtain:

\[
\Delta(U, X) - \mathbb{E}\{w_1 g(f) - w_2 h(p)|U, X\} \\
\leq B + C - w_3 X\mathbb{E}\{p_n + t_d|U, X\} \\
- \mathbb{E}\{V w_1 g(f) - UF(f)|U, X\} \\
- \mathbb{E}\{Up - V w_2 h(p) - w_3 Xp|U, X\}
\]

\((6.3.19)\)

Note that the encoder optimization policy (second last term) remains the same as \((6.2.26)\). The last term shows that decoder optimization policy \((6.2.27)\) has an additional penalty term \(-w_3 Xp\). This will mean that as \(X\) increases, the decoder will get penalized more for high playout interval \(p\) values. This will encourage the decoder to choose a lower \(p\) whenever the accumulated delay in \(X\) is high.

6.4. Network Delay Impact

In sections 6.2 and 6.3, we showed how Lyapunov optimization can derive optimization policies that help ensure the video continuity and enforce a delay constraint. However, the assumption made in those sections for the derivations is that there is no network delay. Specifically, we assumed that the feedback delay from the receiver to the sender is non-existent. In this section, we relax this network delay assumption and analyze the impact of network delay on the optimization policies.

6.4.1. Discontinuity Penalty Lyapunov Optimization Rederivation

Recall that the network generates a forward delay of \(d_f\) and a backward delay of \(d_b\) from the system model discussed in section 6.2.2. With network delay, the discontinuity penalty \(U(t)\) updating in the receiver is performed as:

\[
(6.4.1) \\
U(t + 1) = [U(t) - \gamma(t)]^+ + F(f(t - d_f))
\]

where \(\gamma(t) = p(t) + \beta(t)\) and recall that \(\beta(t) = \frac{b(t) r_{\max} p_{\max}}{t - t}\). Note that the difference between the previously presented discontinuity penalty buffer dynamics in \((6.2.14)\) and above is that \((6.4.1)\) is based on the forward delayed

\[\text{see lemma 2.3 in chapter 2}\]
encoder frame generation interval $f(t - d_f)$. Furthermore, the sender relies on feedback from the receiver, so it chooses $f(t)$ based on a backward delayed $U(t - d_b)$. There are two possible issues that might impact on the optimization policies when delays are present:

1. the delayed discontinuity penalty $U(t)$ feedback from the receiver to the sender means the encoder optimization policy will need to make use of $U(t - d_b)$.

2. the current choice of $f(t)$ at the sender would only affect receiver $d_f$ time slots later.

What we need to do is to derive optimization policies that take the above two issues into account and show that these policies stabilizes the discontinuity penalty $U(t)$.

To begin, note that (6.4.1) can be represented recursively as:

$$U(t + 1) = \left[ U(t) - \gamma(t) \right]^+ + F(f(t - d_f))$$

$$U(t) = \left[ U(t - 1) - \gamma(t - 1) \right]^+ + F(f(t - d_f - 1))$$

$$\vdots$$

(6.4.2) 

$$U(t - d_b + 1) = \left[ U(t - d_b) - \gamma(t - d_b) \right]^+ + F(f(t - d_f - d_b))$$

This implies that $U(t)$ can be recursively defined as:

(6.4.3) 

$$U(t) \leq \left[ U(t - d_b) - \sum_{\tau=t-d_b}^{t-1} \gamma(\tau) \right]^+ + \sum_{\tau=t-d_b-d_f}^{t-d_f-1} F(f(\tau))$$

Issue 2 suggests that we need to predict $d_f$ time slots in the future. To do that, we change the buffer updating equation (6.4.1) into a $d_f$ slot update:

(6.4.4) 

$$U(t + d_f + 1) \leq \left[ U(t) - \sum_{\tau=t}^{t+d_f} \gamma(\tau) \right]^+ + \sum_{\tau=t-d_f}^{t} F(f(\tau))$$

What (6.4.4) means is that the receiver updates the discontinuity penalty $U(t)$ not only based on the known current values $\gamma(t)$ and $f(t - d_f)$ but also using the predicted values $d_f$ slots in the future, $[\gamma(t + 1) \ldots \gamma(t + d_f)]$ and $[f(t - d_f + 1) \ldots f(t)]$. Note that $F(f(\tau))$ in (6.4.4) is calculated from $d_f$ time steps in the past. This is because the current $f(t)$ only affects the discontinuity penalty $d_f$ time slots later which will be $U(t + d_f + 1)$. The $d_f$ step buffer dynamics can be proved to obtain stability by using the $T$-slot Lyapunov drift.
To show how the T-slot Lyapunov drift achieves stability, we first convert the 1-step Lyapunov drift in \((6.2.22)\) into a \(d_f\)-step drift:

\[
\Delta(U(t)) \triangleq \mathbb{E}\{L(U(t + df + 1)) - L(U(t))|U(t)\}
\]

We now use the following shortened notations to simplify the equations:

\[
U \triangleq U(t) \quad U_{db} \triangleq U(t - db)
\]

\[
\gamma \triangleq \sum_{\tau = t - db}^{t - df - 1} \gamma(\tau) \quad \gamma_{df} \triangleq \sum_{\tau = t}^{t + df} \gamma(\tau)
\]

\[
F \triangleq \sum_{\tau = t - df - db}^{t - df - 1} F(f(\tau)) \quad F_{df} \triangleq \sum_{\tau = t}^{t + df} F(f(\tau))
\]

Using the same Lyapunov function \((6.2.21)\) as in the previous section and drift definition \((6.4.5)\). If we square \((6.4.4)\), divide it by two and take expectations\(^5\), we get:

\[
\Delta(U) \leq B' - \mathbb{E}\{U\gamma_{df}\} + \mathbb{E}\{UF_{df}\}
\]

Where:

\[
B' = \frac{df}{2} \left( r_{\text{max}}^2 + \left( p_{\text{max}} + \frac{T r_{\text{max}} p_{\text{max}}}{T - t} \right)^2 \right)
\]

Equation \((6.4.9)\) can be shown to achieve stability \([111]\), effectively settling issue 2. However, to deal with issue 1, we need to show how the feedbacked discontinuity penalty \(U(t - db)\) affects \((6.4.9)\). To do that, we substitute the recursively defined \(U(t)\) \((6.4.3)\) into \((6.4.9)\):

\[
\Delta(U) \leq B' - \mathbb{E}\{U\gamma_{df}\} + \mathbb{E}\{([U_{db} - \gamma]^+ + F) F_{df}\}
\]

Recall from \((6.4.8)\) on the definitions of \(F\) and \(F_{df}\), and given that \(F(\cdot)\) is an increasing function of \(f(t)\) (see \((6.2.7)\)), which implies that \(F\) and \(F_{df}\) can be bounded as \(F \leq db r_{\text{max}}\) and \(F_{df} \leq (df + 1) r_{\text{max}}\) respectively. It then follows that \(FF_{df}\) can be bounded as:

\(^5\)Detailed derivations in appendix B.4
(6.4.12) \[ FF_{df} \leq d_b(d_f + 1)r_{max}^2 \]

Note that \([U_{db} - \gamma] \leq U_{db}\). Thus, if we use (6.4.12) in (6.4.11), we get:

\[ \Delta(U) \leq B'' - U\mathbb{E}\{\gamma_{df}|U\} - \mathbb{E}\{-U_{db}F_{df}|U\} \]

(6.4.13)

where \(B'' = B' + d_b(d_f + 1) r_{max}^2\) with \(B'\) from (6.4.10). Equation (6.4.13) in that form has been proven to stabilize \([111]\), thus settling issue 1.

As in section 6.2.5, to cater for utility optimization, we subtract from both sides, the term \(V\mathbb{E}\{w_1g(f(t)) - w_2h(p(t))|U\}\) and rearrange the terms to obtain:

\[
\Delta(U) - V\mathbb{E}\{w_1g(f(t)) - w_2h(p(t))|U\} \\
\leq B'' - \mathbb{E}\{U\gamma_{df} - Vw_2h(p(t))|U\} \\
- \mathbb{E}\{Vw_1g(f(t)) - U_{db}F_{df}|U\}
\]

(6.4.14)

Using the definitions of \(\gamma_{df}\) and \(F_{df}\) from (6.4.8) and that \(\gamma(t) = p(t) + \beta(t)\). Equation (6.4.14) can be rewritten as:

\[
\Delta(U) - V\mathbb{E}\{w_1g(f(t)) - w_2h(p(t))|U\} \\
\leq B'' - U\mathbb{E}\left\{\sum_{\tau=t+1}^{t+d_f} p(\tau) + \sum_{\tau=t}^{t+d_f} \beta(\tau)|U\right\} \\
+ U_{db}\mathbb{E}\left\{\sum_{\tau=t-d_f}^{t-1} F(f(\tau))|U\right\} \\
- \mathbb{E}\{Up(t) - Vw_2h(p(t))|U\} \\
- \mathbb{E}\{Vw_1g(f(t)) - U_{db}F(f(t))|U\}
\]

(6.4.15)

It can be seen from (6.4.15) that the last two terms represent the decoder and encoder optimization policies respectively. To summarize, we obtain the following policies after taking the network delay into consideration:

6.4.1.1. Encoder policy with network delay

Based on the \(U(t)\) feedback from the receiver, the encoder in the sender will choose \(f(t)\) at each time slot as the solution of the following optimization:

\[ \text{Proof is also detailed in appendix E.} \]
Maximize: \( Vw_1g(f(t)) - U(t - d_b)F(f(t)) \)

Subject to: \( f_{\text{min}} \leq f(t) \leq f_{\text{max}} \)  

(6.4.16)

6.4.1.2. Decoder policy with network delay

The decoder in the receiver will observe the current discontinuity penalty \( U(t) \) and choose \( p(t) \) at each time slot as the solution of the following optimization:

Maximize: \( U(t)p(t) - Vw_2h(p(t)) \)

Subject to: \( p_{\text{min}} \leq p(t) \leq p_{\text{max}} \)  

(6.4.17)

In summary, we show that the decoder policy is not affected by the network delay while the only change to the encoder policy is to make use of \( U(t - d_b) \) fed back from the decoder.

6.4.2. Delay Constrained Lyapunov Optimization Rederivation

By observation, since the decoder policy (6.4.23) is not affected by the network delay and the delay accumulator \( X(t) \) is updated locally within the decoder. This would imply that the delay constrained decoder policy (6.5.13) would not be affected. To show this, we first convert the updating mechanics for the delay accumulator \( X(t) \) to predict \( d_f \) steps into the future:

\[
X(t + d_f + 1) \leq X(t) + \sum_{\tau=t}^{t+d_f} (p_n + t_d) + \sum_{\tau=t}^{t+d_f} p(\tau)
\]

(6.4.18)

Correspondingly, the one step conditional drift discussed in section 6.3 is turned into a \( d_f \)-step drift:

\[
\Delta(U(t), X(t)) \triangleq \mathbb{E}\{L(U(t + d_f + 1), X(t + d_f + 1)) - L(U(t), X(t))|U(t), X(t)\}
\]

(6.4.19)

where the same Lyapunov function \( L(U(t), X(t)) \) defined in (6.3.15) is used as before. Using the same shortened notations as in section 6.4.1 as well as the following:

\[
X \triangleq X(t) \\
p \triangleq d_f(p_n + t_d) \\
p_{d_f} \triangleq \sum_{\tau=t}^{t+d_f} p(\tau)
\]
By squaring $[6.4.18]$, multiplying it by $w_3$, divide it by 2, taking expectations and adding it with the results of $[6.4.13]$, we get:

$$\Delta(U, X) \leq B'' + C' - U \mathbb{E}\{\gamma_{df}|U, X\} + \mathbb{E}\{U_d F_{df}|U, X\} - w_3 X \mathbb{E}\{\rho - p_{df}|U, X\}$$

(6.4.20)

Where $C'$ is:

$$C' = \frac{w_3 d_f}{2} \left( p_{max}^2 + (p_n + t_d)^2 \right)$$

(6.4.21)

When we subtract from both sides, the term $V \mathbb{E}\{w_1 g(f) - w_2 h(p)|U, X\}$ and rearrange the terms, we get the following two policies:

6.4.2.1. **Encoder policy with network delay**

Based on the $U(t)$ feedback from the receiver, the encoder in the sender will choose $f(t)$ at each time slot as the solution of the following optimization:

Maximize: $V w_1 g(f(t)) - U(t-d_b) F(f(t))$

Subject to: $f_{min} \leq f(t) \leq f_{max}$

(6.4.22)

6.4.2.2. **Decoder policy with network delay**

The decoder in the receiver will observe the current discontinuity penalty $U(t)$ and choose $p(t)$ at each time slot as the solution of the following optimization:

Maximize: $U(t)p(t) - V w_2 h(p(t)) - w_3 X(t)p(t)$

Subject to: $p_{min} \leq p(t) \leq p_{max}$

(6.4.23)

6.4.3. **Alternate Network Delay Impact Analysis**

In the discussions delayed feedback analysis thus far, we demonstrated a derivation of optimization policies that take delayed feedbacks into consideration. However, this is not the only way to derive the optimization policies with delayed feedback consideration. Here, we present an alternative way to derive the optimization policies to deal with the network feedback delay.

Notice that we can rewrite $[6.4.4]$ as:
(6.4.24) \[ U(t + d_f + 1) \leq \left[ U(t - d_b) - \sum_{\tau=t-d_b}^{t+d_f} \gamma(\tau) \right]^+ + \sum_{\tau=t-d_b-d_f}^{t} F(f(\tau)) \]

As before, (6.4.24) solves issue 2 (on page 106) by predicting \( d_f \) steps into the future. However, it solves issue 1 by ensuring both the sender and receiver updates their buffer based on \( U(t - d_b) \). This will result in a more consistent buffer updating between the sender and receiver, although the receiver updates using out-dated information.

Using the same shortened notations as in section 6.4.1 as well as the following:

\[ F_{rtt} \triangleq \sum_{\tau=t-d_b-d_f}^{t} F(f(\tau)) \]
\[ \gamma_{rtt} \triangleq \sum_{\tau=t-d_b}^{t} \gamma(\tau) \]

Using the Lyapunov function (6.2.21) and drift definition (6.4.5). If we square (6.4.24), divide it by two and take expectations, we get:

(6.4.25) \[ \Delta(U) \leq B' - U_{db} E\left\{ \gamma_{rtt} - F_{rtt}\right\} \]

Where \( B' \) is defined in (6.4.10). Equation (6.4.25) is shown to be stabilizable too [111]. To cater for utility optimization, we again subtract from both sides of (6.4.25), the term \( V E\{w_1 g(f(t)) - w_2 h(p(t))\} \) and rearrange the terms.

We then get the following two policies:

6.4.3.1. Encoder policy with network delay
Based on the \( U(t) \) feedback from the receiver, the encoder in the sender will choose \( f(t) \) at each time slot as the solution of the following optimization:

Maximize: \[ V w_1 g(f(t)) - U(t - d_b) F(f(t)) \]
Subject to: \( f_{min} \leq f(t) \leq f_{max} \)

(6.4.26)

6.4.3.2. Decoder policy with network delay
The decoder in the receiver will observe the current discontinuity penalty \( U(t) \) and the delay accumulator \( X(t) \), and choose \( p(t) \) at each time slot as the solution of the following optimization:
Maximize: \[ U(t)p(t) - Vw_2h(p(t)) - w_3X(t)p(t) \]

(6.4.27) Subject to: \[ p_{\text{min}} \leq p(t) \leq p_{\text{max}} \]

The main difference with the policies derived before is that the decoder policy (6.4.27) makes use of the virtual buffer state at time \( t - d_b \), i.e. \( U(t - d_b) \), to perform its optimization.

The performance of feedback delay analysis and the alternate feedback delay analysis will be further investigated later in section 6.8.4.

6.5. Video Quality Functions and Normalization

In the previous sections, we showed how using the discontinuity penalty virtual buffer in the system model allows us to express the processes as frame intervals. We then used Lyapunov optimization to derive optimization policies that stabilizes the discontinuity penalty virtual buffer and showed that this helps to maintain the video continuity. We also demonstrated how to add a delay constraint into a framework by using the delay accumulator virtual buffer. By deriving optimization policies that stabilize the delay accumulator virtual buffer, we showed that the delay constraint can be met. Finally, we studied the impact of network delay on the optimization policies and showed two different ways of deriving optimization policies with network delay consideration.

What is lacking thus far is a discussion on the specific choice of frame quality function \( g(f(t)) \) and playout distortion function \( h(p(t)) \). In this section, we shall examine the specific forms for \( g(f(t)) \) and \( h(p(t)) \). We will also examine a way to normalize the policies using these video quality functions, specifically we look into a particular choice for the constants \( w_1, w_2 \) and \( w_3 \).
6.5.1. **Frame Quality Function**

**Figure 6.4.** PSNR versus average frame size. The sequence used is a concatenation of football, city, crew and akiyo in that order.

We first look at an appropriate frame quality function for \( g(f(t)) \), \( g(f(t)) \) should ideally be concave so that a solution can be easily found for the encoder policy (6.4.16). An and Nguyen [107] has been shown that PSNR can be represented using a log function of the bitrate. We make use of their result and fit PSNR to the average frame size of the sequence:

(6.5.1) \[
PSNR(f(t)) = a \log(ABR(t)f(t)) + c
\]

Where \( a \) and \( c \) are the modeling coefficients, and \( ABR(t) \) is the current available network bandwidth. As in the previous chapter \([5]\), we make use of all the available bandwidth to transmit the video frames from the sender. Fig. 6.4 shows the fitted curve. The video sequence used for fitting is a concatenation of football, city, crew and akiyo sequences in that order. This is done to ensure that the sequence contains several subsequence of different coding complexity. Notice that if we use the encoder frame generation rate \( i(t) \) as the input variable, (6.5.1) becomes:

(6.5.2) \[
PSNR(i(t)) = a \log \left( \frac{ABR(t)}{i(t)} \right) + c
\]

Where \( i(t) = \frac{1}{f(t)} \). The resulting PSNR function would no longer be a concave function of \( i(t) \) and would make the optimization problem more difficult.
to solve. This is the reason why we use the encoder frame generation interval \( f(t) \).

Since \( f(t) \) is upper bounded by \( f_{\text{max}} \), we fix the maxima of \( \text{PSNR}(f(t)) \) to \( f_{\text{max}} \). This is done to make the PSNR function differentiable in the interval \([f_{\text{min}}, f_{\text{max}}]\) and make the maximization of it easy to calculate. We do this by subtracting \( \frac{a}{f_{\text{max}}} \) from the first derivative of (6.5.1) \( \text{PSNR}'(f(t)) \):

\[
(6.5.3) \quad \text{PSNR}'(f(t)) - \frac{a}{f_{\text{max}}} = \frac{a}{f(t)} - \frac{a}{f_{\text{max}}}
\]

Integrating (6.5.3) will give us the desired \( g(f(t)) \):

\[
(6.5.4) \quad g(f(t)) = a \log(ABR(t) f(t)) + c - \frac{a f(t)}{f_{\text{max}}}
\]

Note that \( g(f(t)) \) is concave between the range \([f_{\text{min}}, f_{\text{max}}]\) (where \( f_{\text{min}} > 0 \)).

6.5.2. Playout Distortion Function

![Figure 6.5. Playout distortion function. \( p(t) \) is in milliseconds, \( m = 1 \) and \( p_n = \frac{1}{30} \) ms.](image)

In this section, we choose an appropriate playout distortion function for \( h(p(t)) \). We modified the version of playout distortion function used in [SS]:

\[
(6.5.5) \quad h(p(t)) = m \cdot (p_n - p(t))^2
\]
Where $m$ is the motion intensity of the sequence and $p_n$ is the natural playout interval. Fig. 6.5 shows the playout distortion function $h(p(t))$. $h(p(t))$ is convex in the range $[p_{\text{min}}, p_{\text{max}}]$.

6.5.3. Choice of $w_1$ and $w_2$

We now examine how the constants $w_1$ and $w_2$ can be chosen. The choice of these constants is important as it is needed to help scale the three different quantities involved in the encoder policy (6.2.26) and decoder policy (6.2.27), namely $U(t)$, $g(f(t))$ and $h(p(t))$, correctly.

We want to choose $w_1$ and $w_2$ such that when the discontinuity penalty $U(t)$ increases beyond a maximum tolerable level $U_n$, the policies would choose $f(t) = f_{\text{min}}$ and $p(t) = p_{\text{max}}$ to reduce the chance of a buffer underflow and increase the probability of maintaining video continuity.

We observed that such an effect can be achieved by setting the maxima of the encoder optimization policy (6.2.26) and decoder optimization policy (6.2.27) at $f_{\text{min}}$ and $p_{\text{max}}$ respectively when $U(t) = U_n$. The maxima of the policies can be analyzed by looking into the first derivatives of the policies.

We also assume that there is no network delay variation in our derivations here. That is the frame interval scaling factor $e(t) = 1$ and therefore, by the definition of the network delay variation function (6.2.7), $F(f(t)) = f(t)$. This assumption is made to simplify the derivation of $w_1$. In practice, the $F(f(t))$ in the encoder policy (6.2.26) will scale the choice of $f(t)$ according the network delay variation.

To determine $w_1$, we look into the derivative of the encoder objective (6.2.26) and use the $g(f(t))$ specified in (6.5.4):

$$
\frac{d}{df} \left( Vw_1g(f(t)) - U(t)F(f(t)) \right) = Vw_1 \left( \frac{a}{f(t)} - \frac{a}{f_{\text{max}}} \right) - U(t)e(t)
$$

Setting (6.5.6) to zero and solving for $f(t)$ for the interval $[f_{\text{min}}, f_{\text{max}}]$ gives us:

$$
f(t) = \frac{aVw_1f_{\text{max}}}{aVw_1 + U(t)e(t)f_{\text{max}}}
$$

We can now observe from (6.5.7) that $f(t)$ is inversely proportional to both the discontinuity penalty $U(t)$ and the frame interval scaling factor $e(t)$. Subsequently, if we set $e(t) = 1$ in (6.5.7) and solve for $w_1$, we will get:

$$
w_1 = \frac{U(t)f(t)f_{\text{max}}}{aV(f_{\text{max}} - f(t))}
$$
Now, we want the maxima to occur at $f_{\text{min}}$ when $U_n$ is reached, this is done by setting $f(t) = f_{\text{min}}$ and $U(t) = U_n$ to obtain:

$$w_1 = \frac{U_n f_{\text{min}} f_{\text{max}}}{aV(f_{\text{max}} - f_{\text{min}})}$$  \hspace{1cm} (6.5.9)

Substituting (6.5.9) into (6.5.7) would mean:

1. If $U(t) = U_n$ and $e(t) = 1$, then $f(t) = f_{\text{min}}$.
2. If $U(t) > U_n$ and $e(t) = 1$, then $f(t) < f_{\text{min}}$ as $f(t)$ is inversely proportional to $U(t)$. However, since $f(t)$ is bounded as $f(t) \geq f_{\text{min}}$, which means that $f(t) = f_{\text{min}}$.

Therefore, by setting $w_1$ using (6.5.9), then for any $U(t) \geq U_n$, the encoder will set $f(t) = f_{\text{min}}$.

Likewise, for the decoder objective (6.2.27), taking its derivative and using the playout distortion function (6.5.5) will result in:

$$\frac{d}{dp} (U(t)p(t) - V w_2 h(p(t))) = U(t) - 2V w_2 m(p(t) - p_n)$$  \hspace{1cm} (6.5.10)

Setting (6.5.10) to zero and solving for $p(t)$ for the interval $[p_{\text{min}}, p_{\text{max}}]$:

$$p(t) = \frac{U(t)}{2V w_2 m} + p_n$$  \hspace{1cm} (6.5.11)

Note that $p(t)$ varies linearly with $U(t)$. We want the maxima to occur at $p_{\text{max}}$ when $U(t) = U_n$. To achieve this effect, we set (6.5.10) to zero, $U(t) = U_n$, $p(t) = p_{\text{max}}$ and solve for $w_2$, we get:

$$w_2 = \frac{U_n}{2V m(p_{\text{max}} - p_n)}$$  \hspace{1cm} (6.5.12)

Using a similiar argument as above, it then follows that when $w_2$ is set using (6.5.12), then for any $U(t) \geq U_n$, the decoder will set $p(t) = p_{\text{max}}$.

To summarize, if $w_1$ is set using (6.5.9) and $w_2$ is set using (6.5.12). When $U(t)$ exceeds the maximum tolerable discontinuity penalty, that is $U(t) \geq U_n$. The policies will set the encoding frame interval to the minimum possible ($f(t) = f_{\text{min}}$) to speed up the encoding rate and set the playout frame interval to the maximum possible ($p(t) = p_{\text{max}}$) to slow down the playout rate. These are done to help reduce the discontinuity penalty $U(t)$.

6.5.4. Choice of $w_3$

We also need to normalize the penalty term $-Xp$. To do this, we make use of the normalization factor $w_3$. Recall from (6.3.19) that the delay constrained
decoder optimization policy is:

Maximize: $U(t)p(t) - Vw_2h(p(t)) - w_3X(t)p(t)$

Subject to: $p_{min} \leq p(t) \leq p_{max}$

Equation (6.5.13)

We want to choose $w_3$ such that when the delay accumulator $X(t)$ reaches a maximum tolerable level $X_n$. The decoder policy will choose $p(t) = p_{min}$ to speed up the playout in order to reduce the accumulated delay. This can be done by setting the maximum at $p_{min}$ when $X(t) = X_n$.

We based our calculations of $w_3$ on the assumption that the discontinuity penalty is zero, i.e. $U(t) = 0$ and the calculation of $w_2$ is done on the assumption that there are no accumulated delay (i.e. $X(t) = 0$). This is done to keep the calculations of $w_2$ and $w_3$ simpler and independent of each other.

$$\frac{d}{dp}(U(t)p(t) - Vw_2h(p(t)) - w_3X(t)p(t)) =$$

Equation (6.5.14)

$$U(t) - 2Vw_2m(p(t) - p_n) - w_3X(t)$$

Setting (6.5.14) to zero and solving for $p(t)$ for the interval $[p_{min}, p_{max}]$ yields:

Equation (6.5.15)

$$p(t) = \frac{U(t) - w_3X(t)}{2Vw_2m} + p_n$$

It can be seen from (6.5.15) that as the delay accumulator $X(t)$ increases, $p(t)$ decreases. Note that the previous normalization factor $w_2$ is still defined as in (6.5.12).

To set the maximum at $p_{min}$ when $X(t) = X_n$, we set (6.5.14) to zero, $U(t) = 0$, $p(t) = p_{min}$, $X(t) = X_n$, and solve for $w_3$:

Equation (6.5.16)

$$w_3 = \frac{2Vw_2m(p_n - p_{min})}{X_n}$$

Using a similar argument as in section 6.5.3, it follows that when $w_3$ is set using (6.5.16). Whenever $U(t) = 0$ and $X(t) \geq X_n$, the decoder will attempt to reduce $X(t)$ by setting $p(t) = p_{min}$.

6.6. Performance Analysis

In this section, we prove the performance bounds for the optimization policies presented.
6.6.1. Discontinuity Penalty Optimization Performance Bounds

**Theorem 6.7.** Let the long term receiving frame interval \( \bar{r} = \lim_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathbb{E} \{ r(t) \} \).

If the network delay variation \( r(t) \) is i.i.d over the timeslots, the receiving frame interval is lower bounded by the encoder frame generation interval as \( r(t) \geq f(t) \), the frame quality function is bounded as \( g(f(t)) \leq G_{\max} \) and the playout distortion function is bounded as \( h(p(t)) \geq H_{\min} \). Then implementing the optimization policies (6.2.26) and (6.2.27) in each timeslot stabilizes the discontinuity penalty \( U(t) \) (using the stability definition (6.2.15)) and satisfies the following performance bounds:

\[
\begin{align*}
(6.7.1) & \quad \limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E} \{ U(\tau) \} \leq B + V \left( w_1 G_{\max} - w_2 H_{\min} \right) / r_{\max} \\
(6.7.2) & \quad \liminf_{M \to \infty} g(\bar{f}) \geq \frac{(w_1 g^* - w_2 h^*) + w_2 H_{\min} - B}{w_1} \\
(6.7.3) & \quad \limsup_{M \to \infty} h(\bar{p}) \leq \frac{B - (w_1 g^* - w_2 h^*) - w_1 G_{\max}}{w_2}
\end{align*}
\]

where \( V \) is some positive constant (i.e. \( V > 0 \)), \( g^* \) and \( h^* \) are the specific values of \( g(.) \) and \( h(.) \) respectively that maximizes the objective (6.2.17) subject to the constraints (6.2.18), (6.2.19) and (6.2.20), \( B \) is defined as in (6.2.24) and:

\[
\begin{align*}
(6.7.4) & \quad \bar{f} = \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E} \{ f(\tau) \} \\
(6.7.5) & \quad \bar{p} = \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E} \{ p(\tau) \}
\end{align*}
\]

**Proof.** We first introduce \( \Lambda \) as a set of receiving frame intervals that stabilizes the discontinuity penalty \( U(t) \). \( \Lambda \) is bounded by \( r_{\max} \). Using \( \Lambda \) assumes a complete knowledge of future, but this is only required for the analysis on the performance bounds. With \( \Lambda \), the optimization problem in section 6.2.4 can be restated as:

\footnote{With i.i.d processes, the steady state is exactly achieve every timeslot. This allows us to use Lyapunov drift analysis on a per time slot basis. However, Neely has shown that i.i.d processes provides all of the intuition needed to treat general processes. For more details on this, see chapter 4 of [111].}
Maximize: \( w_1 g(f(t)) - w_2 h(p(t)) \)

Subject to: \( F(f(t)) = r(t) \)

\[ \bar{r} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{r(t)\} \]

\[ \bar{r} \in \Lambda \]

\[ f_{\min} \leq f(t) \leq f_{\max} \]

\[ p_{\min} \leq p(t) \leq p_{\max} \]

From the above optimization problem, \( \Lambda \) can be intuitively seen as a range of values that \( r(t) \) can take that allows the existence of at least one stationary randomized policy that can stabilize \( U(t) \). Constraint (6.7.7) is taken from (6.2.6) on page 97. Let \( f^* \) and \( p^* \) form the solution that maximizes the objective (6.7.6). This means that an optimal stable policy would ensure that the following is met:

\[ F(f^*) = \bar{r} \leq p^* \]

\( F(f^*) = \bar{r} \) comes from constraint (6.7.7), while the inequality \( \bar{r} \leq p^* \) is the result of lemma 2.3. Suppose now an \( \epsilon \)-optimal stable policy causes the long term receiving frame interval \( \bar{r} \) to become \( \bar{r}_\epsilon \) such that:

\[ F(f^*_\epsilon) = \bar{r}_\epsilon \leq p^* - \epsilon \]

\[ \bar{r}_\epsilon \in \Lambda_\epsilon \]

\[ \bar{r}_\epsilon + \epsilon \in \Lambda \]

where \( \epsilon > 0 \) is a positive constant and the receiving frame intervals set \( \Lambda_\epsilon \subset \Lambda \). \( \Lambda_\epsilon \) can be seen as \( \Lambda \) that is reduced by \( \epsilon \). Then, the \( \epsilon \)-optimal policy can be seen as optimizing the following problem:

\( \epsilon \)-optimal policy

Here \( r(t) \) is the arrival process while \( p(t) \) is the departure rate.
Maximize:  \[ w_1 g(f(t)) - w_2 h(p(t)) \]

Subject to:  \[ F(f(t)) = r(t) \]
\[ r(t) \in \Lambda_\epsilon \]
\[ f_{min} \leq f(t) \leq f_{max} \]
\[ p_{min} \leq p(t) \leq p_{max} \]  

(6.7.14)

Note that the long term encoder frame generation interval \( f_\epsilon^* \) forms part of the solution that maximizes the above problem. Equations (6.7.12) and (6.7.13) together implies that:

(6.7.15)  \[ \bar{r}_\epsilon \leq \bar{r} \]

This is because the \( \epsilon \)-optimal policy maximizes the same objective as (6.7.6) but with its long term receiving frame interval \( \bar{r}_\epsilon \) limited by \( \epsilon \). Using (6.2.6) would mean the above equation becomes:

(6.7.16)  \[ F(f_\epsilon^*) \leq F(f^*) \]

Using (6.2.7) and assuming there exist the long term frame interval scaling factors \( e_\epsilon \) and \( e^* \), such that \( F(f_\epsilon^*) = e_\epsilon f_\epsilon^* = \bar{r}_\epsilon \) and \( F(f^*) = e^* f^* = \bar{r} \). Then, applying these properties to the above equation yields:

(6.7.17)  \[ e_\epsilon f_\epsilon^* \leq e^* f^* \]

Since we made the assumption that \( r(t) \geq f(t) \), which can be rewritten as \( e(t) f(t) \geq f(t) \). This implies that \( e_\epsilon \geq 1 \) and \( e^* \geq 1 \), which means that:

(6.7.18)  \[ f_\epsilon^* \leq f^* \]

This is because \( f_\epsilon^* \) is chosen by the \( \epsilon \)-optimal policy to maximize the objective in (6.7.14). Since the frame quality function \( g(f(t)) \) in the objective is an increasing function of \( f(t) \), which implies that the \( \epsilon \)-optimal policy will choose \( f_\epsilon^* \) as large as possible. However, as \( f_\epsilon^* \) is upper bounded by \( \bar{r}_\epsilon \), which is in turn limited by \( \epsilon \), which means that \( f_\epsilon^* \) will be at most as large as \( f^* \). Furthermore, \( f_\epsilon^* \to f^* \) as \( \epsilon \to 0 \). This is because [14]:
\[ f^* \geq \left( 1 - \frac{r}{r_{\text{max}}} \right) f^* + \frac{r}{r_{\text{max}}} f_{\epsilon}^* \geq f_{\epsilon}^* \]  

The middle term of (6.7.19) can be seen as an example of a mixed policy that picks \( f^* \) with \( 1 - \frac{r}{r_{\text{max}}} \) probability and \( f_{\epsilon}^* \) with \( \frac{r}{r_{\text{max}}} \) probability.

If the long term receiving frame interval is \( \bar{r}_\epsilon \), then there exists a stationary randomized policy that stabilizes \( U(t) \) by setting the long term playout interval to \( p^* \) \[111\]. That is with \( \bar{r}_\epsilon \) as the long term average arrival rate into the discontinuity penalty virtual buffer \( U(t) \). If the policy chooses the playout interval \( p(t) \) over time such that the long term receiving frame interval is \( p^* \), then \( U(t) \) can stabilize since \( p^* > \bar{r}_\epsilon \). Thus, it will not grow infinitely large over time. Under such a policy, the Lyapunov one step drift bound (6.2.25) would be calculated using (6.7.13) as:

\[
\Delta(U(t)) - V \mathbb{E}\{w_1 g(f(t)) - w_2 h(p(t))|U(t)\} \\
\leq B - V \mathbb{E}\{U(t)\beta(t)|U(t)\} \\
- V w_1 g(f_{\epsilon}^*) + U(t)\bar{r}_\epsilon \\
- U(t)p^* + V w_2 h(p^*) \\
\leq B - V \mathbb{E}\{U(t)\beta(t)|U(t)\} \\
- V w_1 g(f_{\epsilon}^*) + U(t)(p^* - \epsilon) \\
\leq U(t)p^* + V w_2 h(p^*) \\
\tag{6.7.20}
\]

By rearranging the terms:

\[
\Delta(U(t)) - V \mathbb{E}\{w_1 g(f(t)) - w_2 h(p(t))|U(t)\} \\
\leq B - V (w_1 g(f_{\epsilon}^*) - w_2 h(p^*)) - \epsilon U(t) \\
\tag{6.7.21}
\]

Equation (6.7.21) is of a similar form to (2.8.10). Thus, we apply the derivations of lemma \[2.2\] to (6.7.21). By taking expectations, summing over the timeslots \( \tau \in [0..M-1] \) and using the non-negativity of \( L(U(t)) \) to drop
the term $\mathbb{E}\{L(U(M-1))\}$, we get:

$$-\mathbb{E}\{L(U(0))\} - V \sum_{\tau=0}^{M-1} \mathbb{E}\{w_1g(f(\tau)) - w_2h(p(\tau))\}$$

(6.7.23) \quad \leq MB - \epsilon \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} - VM(w_1g(f^*_\epsilon) - w_2h^*)$$

To prove the discontinuity penalty bound (6.7.1), we divide (6.7.23) by $M\epsilon$ and rearrange its terms to obtain:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \leq B + V(w_1G_{max} - w_2H_{min})$$

(6.7.24) \quad + \frac{\mathbb{E}\{L(U(0))\}}{M\epsilon}$$

Taking the limits of (6.7.24) as $M \to \infty$ and setting $\epsilon = r_{max}$ yields (6.7.1). Setting $\epsilon = r_{max}$ is done to minimize the bound, as a particular choice for $\epsilon$ will only affect the bound calculation and will not affect the policies in any way.

To prove the utility bounds, note that the concavity of the frame quality function $g(f(t))$ and the convexity of the playout distortion function $h(p(t))$ together with Jensen’s inequality implies the following:

(6.7.25) \quad \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(f(\tau))\} \leq g\left(\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{f(\tau)\}\right)$$

(6.7.26) \quad \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{h(p(\tau))\} \geq h\left(\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{p(\tau)\}\right)$$

If we divide (6.7.23) by $MV$ and rearrange it, we obtain:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{w_1g(f(\tau))\} - \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{w_2h(p(\tau))\}$$

(6.7.27) \quad \geq (w_1g(f^*_\epsilon) - w_2h^*) - \frac{B}{V} - \frac{\mathbb{E}\{L(U(0))\}}{MV}$$

To obtain the frame quality bound, we use the fact that $h(p(t)) \geq H_{min}$ in (6.7.27) and rearrange to get:
\[ \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{w_1 g(f(\tau))\} \]

(6.7.28) \[ \geq (w_1 g(f^*_1) - w_2 h^*) + w_2 H_{min} - \frac{B}{V} - \frac{\mathbb{E}\{L(U(0))\}}{MV} \]

Using (6.7.25) and taking the limits of (6.7.28) as \( M \to \infty \) yields:

(6.7.29) \[ \lim_{M \to \infty} \inf g(\bar{f}) \geq \frac{(w_1 g(f^*_1) - w_2 h^*) + w_2 H_{min} - \frac{B}{V}}{w_1} \]

Again, the above bound can be maximized by taking the limit as \( \epsilon \to 0 \).

This produces the frame quality bound (6.7.2).

To obtain the playout distortion bound, we use the fact that \( g(x(t)) \leq G_{max} \) in (6.7.27) and rearrange to get:

(6.7.30) \[ \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{w_2 h(p(\tau))\} \]

(6.7.31) \[ \leq \frac{B}{V} - (w_1 g(f^*_1) - w_2 h^*) - w_1 G_{max} + \frac{\mathbb{E}\{L(U(0))\}}{MV} \]

Using (6.7.26) and taking the limits of (6.7.30) as \( M \to \infty \) yields:

(6.7.32) \[ \lim_{M \to \infty} \sup h(\bar{p}) \leq \frac{B}{w_2} - (w_1 g(f^*_1) - w_2 h^*) - \frac{w_1}{w_2} G_{max} \]

Taking the limit as \( \epsilon \to 0 \) maximizes the above bound and produces the frame quality bound (6.7.3).

6.7.1. Delay Constrained Optimization Performance Bounds

To obtain the performance bound for the delay constrained Lyapunov optimization policies (6.4.22) and (6.4.23), we develop a policy to enforce a limit on the maximum operating size of the delay accumulator. To determine such a policy, we observe that for (6.4.23) to be non-negative, we need:

(6.7.33) \[ U(t)p(t) - V w_2 h(p(t)) - X(t)p(t) \geq 0 \]

(6.7.34) \[ \Rightarrow U(t)p(t) - X(t)p(t) \geq 0 \]

(6.7.35) \[ \Rightarrow X(t) \leq U(t) \]

Therefore, we introduce the following policy:
If \( X(t) \leq U \)

solve \( p(t) \) by maximizing (6.4.23)

else

\[
p(t) = p_n + t_d
\]

Where \( U \) is a positive constant that represents how easily the maximum operating size of \( X(t) \) gets enforced, \( U \) is set to 100 in our experiments. (6.7.36) together with (6.3.9) ensures that \( X(t) \) does not accumulate anymore in subsequent timeslots. With (6.7.36), we introduce the following corollary:

**Corollary 6.1.** If the network delay variation \( r(t) \) is i.i.d over the timeslots, the frame quality function is bounded as \( g(f(t)) \leq G_{\text{max}} \) and the playout distortion function is bounded as \( h(p(t)) \geq H_{\text{min}} \). Then implementing the optimization policies (6.4.22) and (6.4.23) in each timeslot stabilizes the discontinuity penalty \( U(t) \) and satisfies the following performance bounds:

\[
\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \leq \frac{B + \tilde{C} + V(w_1G_{\text{max}} - w_2H_{\text{min}})}{p_{\text{max}}}
\]

(6.7.37)

\[
\liminf_{M \to \infty} g(\bar{f}) \geq \frac{(w_1g^* - w_2h^*) + w_2H_{\text{min}} - \frac{B + \tilde{C}}{V}}{w_1}
\]

(6.7.38)

\[
\limsup_{M \to \infty} h(\bar{p}) \leq \frac{B + \tilde{C}}{V} - \frac{(w_1g^* - w_2h^*) - w_1G_{\text{max}}}{w_2}
\]

(6.7.39)

Where \( V > 0 \), \( g^* \), \( r^* \), \( f \) and \( p \) are defined as in theorem 6.7. \( B \) is defined as in (6.2.24) and \( \tilde{C} \) is defined using \( C \) (6.3.18) as:

\[
\tilde{C} = C + p_{\text{max}}(U + p_{\text{max}})
\]

(6.7.40)

Furthermore, implementing limit enforcing policy (6.7.36) on the delay accumulator \( X(t) \) would result in it deterministically upper bounded for all timeslots \( t \) as:

\[
X(t) \leq U + p_{\text{max}}
\]

(6.7.41)

**Proof.** Bound (6.7.41) is proved by induction. It can be easily seen that (6.7.41) is satisfied at time 0. Assume that (6.7.41) holds at the current time
t \geq 0$, then we need to prove that $X(t + 1) \leq U + p_{\text{max}}$ in the next timeslot $t + 1$. We have two cases:

1. $X(t) \leq U$: Here $X(t + 1) \leq U + p_{\text{max}}$. Since from (6.3.9), the maximum delay added to $X(t)$ in one timeslot is $p_{\text{max}}$.

2. $X(t) > U$: In this case, the limiting policy (6.7.36) will be triggered and $X(t)$ will not increase in time $t + 1$ resulting in $X(t + 1) \leq X(t) \leq U + p_{\text{max}}$.

This proves (6.7.41). To prove (6.7.37), (6.7.38) and (6.7.39), observe that using (6.7.41):

$$X(t)E\{p_n + t_d - p(t)|U(t), X(t)\} \geq -X(t)E\{p(t)|U(t), X(t)\} \geq -(U + p_{\text{max}})p_{\text{max}}$$

(6.7.42)

Using (6.7.42) and (6.7.40) in (6.3.19) will yield:

$$\Delta(U, X) - V E\{w_1g(f) - w_2h(p)|U, X\} \leq B + \tilde{C} - E\{Vw_1g(f) - UF(f)|U, X\} - E\{Up - Vw_2h(p) - Xp|U, X\}$$

(6.7.43)

The proof then proceeds exactly as in theorem 6.7.

6.8. Performance Evaluation

6.8.1. Experiment Setup

![Dumbbell topology with a 5 Mbits/s bottleneck link. A circle is an end-node.](image)

We made use of ns-2 [112] to simulate a network with time-varying data bandwidths. We implemented our framework into a x264 [113] encoder. Network traces obtained from ns-2 are used to simulate the network within the
encoder, thereby allowing the encoder to simulate on-the-fly encoding with delayed feedback. Note that our framework could easily be adapted to multiple pre-encoded copies of the video.

For the network simulations, we made use of a dumbell network topology with the bottleneck bandwidth set to 5 Mbits/s, see fig. 6.6. Video is streamed between two nodes in the network using the TFRC transport protocol. This ensures that the video streams are TCP-friendly. To generate background traffic, random webpage sessions were used for the other pairs of nodes. All the random sessions go through the bottleneck link. Fig. 6.7 shows the throughput profile at the receiver.

![Figure 6.7. Throughput at the receiver over time.](image)

The encoder receives feedback from the decoder and solves the optimization problem to determine the encoder frame generation interval $f(t)$. The x264 encoder makes use of this encoder frame generation interval $f(t)$ by setting the target frame rate of the rate controller to encoder frame generation rate $\frac{1}{f(t)}$. The rate controller’s target frame rate is used to compute the quantization level for each frame. The lower the target frame rate, the lower the quantization level and, as a result, bigger frame sizes. More details on x264’s rate control can be found in [104].

The test sequence used is a concatenation of football, city, crew and akiyo sequences in that order. This is to ensure a mix of high and low motion within the sequence. A 16 minute test sequence is obtained by repeating the concatenated sequence.
The model coefficient $a$ from (6.5.4) is found by curve fitting to be 4.91. The constant $V$ is set to 1 and $U_n$ is set to 10. The maximum tolerable accumulated delay is set as $X_n = 0.5$, although, $X_n$ will be determined by the provider based on the overall delay budget on the streaming system in practice. In our experiments, we tested the continuity of the video, which is defined here as the amount of time spent playing video, specifically:

\begin{equation}
(6.8.1) \quad \text{Playout Continuity} = 1 - \frac{\text{rebuffering time}}{\text{total sequence time}}
\end{equation}

where the rebuffering time is the time needed to refill the buffer to a certain threshold after an underflow event, which is a typical behaviour of a video player [114]. The rebuffering threshold is set to half of the prebuffer amount. The prebuffer amount is varied for each run to determine the performance of each scheme. At the end of each run, we will calculate the playout continuity using (6.8.1) for each scheme and make a comparison.

6.8.2. Discontinuity Penalty Lyapunov Optimization Results

![Figure 6.8. Correlation between average discontinuity penalty and continuity.](image-url)
Figure 6.9. Tradeoff between prebuffering delay and continuity. Prebuffer delay is on a log base 10 scale.

Figure 6.10. Tradeoff between prebuffering delay and playout distortion. Prebuffer delay is on a log base 10 scale.
We first examine the correlation between the discontinuity penalty $U(t)$ and the continuity of the video (6.8.1), fig. 6.8 plots this. It can be seen from the graph that as the discontinuity penalty increases, the continuity drops due to the longer rebuffering time caused by an increased number of buffer underflows. This shows that the discontinuity penalty $U(t)$ can be used as an indicator of a lack of video continuity.

We next present the results of the Lyapunov optimization framework described in section 6.2, labelled as $LOpt$ here. We compare our framework with a typical setup of x264 with its target frame rate and its playout rate set to a constant 30 fps, we label this scheme $Norm$. We also compared our framework with the AMP scheme [13]. We implemented a combination of AMP-Initial and AMP-Robust. AMP-Initial slows down the playout rate by a predetermined slowdown factor when the video starts playing until a certain buffer level has been reached. AMP-Robust slows down the playout rate of the video by a predetermined slowdown factor when the buffer level falls below a certain level. In our implementation, AMP-Initial is used in conjunction with AMP-Robust. We empirically chose the smallest slowdown factor of AMP such that it achieves a continuity of 99% or higher in each test case. This effectively simulates a close to optimal adaptive AMP scheme, we label this as $AMP$. We also added in the results of $AMP$ with a constant slowdown factor of 25% (used in [13]) as a reference, which we label $AMP25$.

To examine the performance of each scheme, we compare the continuity of each scheme based on the amount of prebuffering provided. Continuity is
calculated as in (6.8.1).

The continuity results are shown in fig. 6.9. It can be seen that $LOpt$ and $AMP$ achieves similar results. This is expected as $AMP$ was tuned to achieve a high continuity. The performance disparity between $AMP$ and $AMP25$ shows that this simulation requires a greater amount of playout slowdown at the decoder in order to reduce the occurrences of buffer underflows. However, both $LOpt$ and $AMP$ require about a 100 times less prebuffering compared to $Norm$ to provide similar continuity. $AMP25$ too requires about 7 times less prebuffering than $Norm$ but still requires about 50 times more prebuffering than $LOpt$ and $AMP25$. This suggests that using some form of playout slowdown would reduce the prebuffering requirements of a video application significantly.

We next measure the playout distortion of each scheme using (6.5.5) ($p_n = 1/30$). Note that while (6.5.5) will only produce a non-zero value when the playout deviates from the natural playout frame interval, playout interruptions due to buffer underflows are not factored into the playout distortion. Fig. 6.10 shows the playout distortion results. $Norm$ does not have any playout distortion since it has a constant 30 fps playout rate, but suffers from playout discontinuity as discussed earlier. $LOpt$ has a very similar playout distortion characteristic when compared to $AMP25$. In contrast, $AMP$ for most cases produces twice the amount of playout distortions compared to $LOpt$ and $AMP25$. This is mainly because $AMP$ requires a higher slowdown factor to obtain a better video continuity and, as a consequence, this results in a higher playout distortion.

$LOpt$’s comparatively low playout distortion is due to the joint adjustment of both the encoding frame rate and playout rate. By increasing the encoding frame rate, the receiving frame at the decoder increases. This provides a higher buffer occupancy and reduces the need to slowdown the playout rate, thus reducing the playout distortion. This comes at the expense of frame quality. To examine $LOpt$’s effect on frame quality, we compare the PSNR of the encoded video before transmission. This is done to eliminate any possible drops in PSNR due to transmission. $Norm$, $AMP25$ and $AMP$ have a constant encoding rate of 30 fps, which means all produce an encoded video of the same PSNR. Thus, we only compared $LOpt$ with $Norm$. From fig. 6.11 it is shown that the drop in PSNR is about 0.6 dB for $LOpt$. This is a reasonably small tradeoff in frame quality given the improvements in playout distortion and continuity.
6.8. PERFORMANCE EVALUATION

6.8.3. Delay Constrained Lyapunov Optimization Results

Figure 6.12. Tradeoff between playout delay and continuity. Playout delay is on a log base 10 scale.

Figure 6.13. Tradeoff between playout delay and playout distortion. Playout delay is on a log base 10 scale.
To evaluate the performance of our delay constrained Lyapunov optimization framework, which we call $DLOpt$ here, we compare our scheme with AMP-Live \[13\], labelled as $AMPL$. AMP-Live maintains the buffer level by slowing down or speeding up the playout based on a predetermined scale factor. The scale factor of $AMPL$ is set to 40\% in our experiments, this value was found to provide the best overall performance for $AMPL$. The playout delay of each scheme is measured by:

\[
\text{Total playout delay} = \sum_{\tau=0}^{T} (p(\tau) - p_n)
\]

Recall that $p_n$ represents the natural playout frame interval of the sequence. So every playout slowdown will cause $p(t)$ to be larger than $p_n$, thus accumulating playout delay. To reduce the total playout delay, the scheme needs to find the proper moment to increase the playout rate.

We compared the schemes by varying the delay constraints from 4.27 seconds to 273.07 seconds by doubling the constraint at each run. At the end of each run, we plot the total playout delay and the performance metric of each scheme. The performance metric we examined are playout continuity, playout distortion and PSNR.

We first look at the continuity results with respect to the total playout delay (fig. 6.12). It can be seen that $DLOpt$ achieves maximum continuity regardless of the delay constraint, while $AMPL$ requires more than 50 times the amount of total playout delay compared to $DLOpt$ to reach maximum
continuity. AMPL manages the total playout delay constraint by maintaining a certain buffer level \[13\], thus a tighter delay constraint will result in a lower buffer level which has higher probabilities of buffer underflows. On the other hand, DLopt makes a trade-off between the encoding frame rate and playout rate to satisfy the continuity goal and delay constraint.

We next look at the distortion results in fig. 6.13. Again, DLopt achieves a much lower playout distortion compared to AMPL. Note that the lower the total playout delay constraints the lower the playout distortion. This is because a low playout constraint causes DLopt to adjust the playout rate a lot less, thus causing a low playout distortion. AMPL has an almost constant playout distortion. This is because, as mentioned before, AMPL only tries to maintain a certain buffer level and does not constrain its playout rate adjustment in anyway.

Lastly, we examine the PSNR trade-offs made, see fig. 6.14. As before we compared the PSNR of the encoded video of both schemes prior to transmission. It can be seen that DLopt sacrifices about 1dB of PSNR, which is lower than LOpt in the previous section. The main cause of this is the additional delay constraint imposed through the last term of the objective function of (6.5.13) and regulated by the virtual buffer (6.4.18). This results a lower PSNR due to the need to satisfy the delay constraint, however, this allows the large gains in continuity and distortion.

6.8.4. Comparison Of Schemes For Delayed Feedback

![Figure 6.15. Feedback delay vs playout distortion.](image-url)
In sections 6.4.1 and 6.4.3, we derived two different schemes to deal the delayed feedback. We now study the performance differences between the two. We name the set of optimization policies derived in section 6.4.1 $ND$. While the set of policies derived in section 6.4.3 is named $AltND$.

To compare these schemes, we gradually increased the feedback delay by increasing the backward delay $d_b$, and measured the averaged playout distortion as well as the averaged PSNR that resulted. Again, the PSNR of the encoded video was used for comparison. The continuity results of these two schemes is not shown here as both maintained 100% continuity regardless of the feedback delay tested.

Fig. 6.15 shows the playout distortion comparison. It can be seen that the playout distortion of $AltND$ gets larger as the feedback delay increases. The PSNR results in fig. 6.16 also show $AltND$ performing with a lower PSNR than $ND$ as the feedback delay increases. This is because, in $AltND$, the receiver uses the outdated buffer state $U(t - d_b)$ for its optimization policy. As a consequence, the receiver using $AltND$ is not as reactive to the current network conditions compared to $ND$. Thus, the performance of $AltND$ decreases as the feedback delay increases. Therefore, the chosen scheme to use for our main experiment in sections 6.8.2 and 6.8.3 is $ND$.

6.9. Conclusion

In this chapter, we look at resolving some of the issues in the frame rate optimization framework. What we achieved is:
• A reformulation of the frame rate optimization problem using frame intervals and the discontinuity penalty virtual buffer. We first showed that this reformulation solves the inverse relationship between the encoder frame generation rate and the frame quality metric, thus allowing the more popular frame quality functions to be used. We also showed through experiments that the proposed discontinuity penalty based on the virtual buffer is correlated to the video continuity. Furthermore, our analysis using Lyapunov optimization shows that stabilizing the discontinuity penalty virtual buffer allows the video to maintain continuity. Finally, simulation results demonstrates the effectiveness of the discontinuity penalty virtual buffer approach.

• A delay constraint imposed on the accumulated delay from playout slowdowns. We introduced the delay constraint into the framework using the delay accumulator virtual buffer. We showed, using Lyapunov optimization analysis, that by stabilizing the delay accumulator virtual buffer, the delay constraint would be satisfied. Simulation results showed a superior playout continuity and playout distortion performance with a reasonable tradeoff in PSNR.

• An analysis of the impact of delayed feedback from receiver to sender. We derived two different analyses, the first analysis showed very little impact on the optimization polices. The alternate analysis showed that the decoder needed to use an outdated buffer state. Simulation results demonstrated that using the first analysis results in a better performance.

• The choice of video quality functions presented in this chapter allows the solution to be determined from a closed form solution that is determined through the first derivative of the encoder optimization policy and the decoder optimization policy. This results in a potentially lower computational overhead than the original framework proposed in chapter 5, which requires an iterative search for the solution.
CHAPTER 7

Conclusion

This thesis focuses on the video rate control and the frame rate control part of video adaptation. Video rate control aims to adjust the encoder parameters such that the resulting video bitrate meets the channel or network constraints. We approached this by proposing VTFRC, a joint source rate and congestion control scheme, in chapter 3. VTFRC incorporates the video bit rate characteristic into the calculation of the TCP-friendly transmission rate. The video bit rate characteristic is estimated using a frame complexity measure as it can describe the video bit rate characteristic more accurately. VTFRC also calculates the rate gap based on the instantaneous TCP rate and the TFRC rate. VTFRC then made use of both the frame complexity measure and the rate gap to identify and opportunistically encode the more complex parts of the video at a higher rate. We showed that VTFRC improves video quality while achieving TCP-friendliness over the existing scheme. We then looked at ways to determine the frame complexity to aid in the video rate control. To that end, we proposed a complexity-based encoder rate control scheme in chapter 4. The frame complexity measure is derived based on the edge energy of the frame. We showed that the edge energy can be used to describe the complexities of the I-, P- and B-frames. Furthermore, the proposed scheme does not assume that any information on the whole video sequence is available. We showed that the proposed encoder rate control scheme produces a video bit stream that is closer to the target bitrate while improving on its video quality over existing schemes. Frame rate control aims to adjust the frame generation rate of the sender and receiver such that the video quality and video continuity are maximized. We attempted to achieve this by proposing a frame rate optimization framework in chapter 5. The frame rate optimization framework poses the frame rate control problem as an optimization problem that maximizes the visual quality and the decoder buffer occupancy, minimize the playout distortion. Since these three objectives have conflicts with each other, we attempted to find the optimal tradeoff. Using Lyapunov optimization, we showed that the optimization policies can be systematically derived and that they can be decoupled into encoder and decoder policies with feedback. Simulation results showed that the framework provided maximum continuity with modest trade-
offs in visual quality. It is also shown that the prebuffering required is reduced significantly compared to current approaches.

Despite promising results, the frame rate optimization framework still had some issues, namely:

- popular frame quality functions cannot be used with the framework due to the expression of frame quality as the inverse of the encoder frame generation rate.
- playout slowdowns introduce additional viewing latency into the system and may need to be constrained to meet a certain delay budget.
- the impact of feedback delays on the optimization policies have not been investigated.

To deal with these issues, we proposed various improvements on the framework in chapter 6, in particular:

- a re-formulation of the optimization problem using the discontinuity penalty virtual buffer. We showed that this circumvents the inversion issue by formulating the variables as frame intervals. This then allows allows a broader class of video quality functions to be supported within the framework. We also showed that stabilizing the discontinuity penalty would ensure that the continuity of the video is preserved.
- a delay constraint was introduced into a framework using the delay accumulator virtual buffer. We showed that stabilizing the delay accumulator would mean the delay constraint imposed is met. We also showed that the delay constrained framework provides a superior tradeoff between the video quality and the delay introduced compared to the existing approach.
- we performed an analysis on the effect of delayed feedbacks on the derived optimization policies. We showed that the impact on the optimization policies is minimal.

7.1. Future Works

The work on video adaptation in this thesis is by no means conclusive. Research, as they say, goes on. To this end, we identified the following possible research issues:

- A study on the suitability of the complexity based H.264/AVC rate control proposed in chapter 4 for use in VTFRC proposed in chapter 3. The proposed complexity based rate control scheme makes use of edge energy to determine the frame complexity. This could be used as the frame complexity measure in VTFRC. Furthermore, the higher
performance of the proposed complexity based rate control scheme may help increase the performance of VTFRC.

- A playout distortion function that represents the user perceived quality. The objective playout distortion functions used by Kalman et al [54], Li et al [88] and in chapters 5 and 6 do not have any user experiments to back up. In fact, to the author’s knowledge, there is no known published results on the playout distortion function. It is anticipated that a playout distortion function would play a greater role in the future as high frame rate television comes into play. Douglas Trumbull had shown that the emotional response of people peaked at 72 fps with a 35mm film [116]. Studies from BBC research have shown that a high capture and display frame rate can significantly improve the perceived quality [117]. An accurate playout distortion function would allow a high frame rate video to be optimized more perceptually.

- Packet scheduling with frame rate optimization. The scheduling of packets becomes more critical as buffer compensation is performed. This is because possibly more video data will be sent in one packet and a loss of such a packet will have a greater impact on the video. The scheduling framework will prioritize the packets based not only on its content, network conditions and packet deadlines, but also the amount of buffer compensation being done. The scheduling framework should also protect the higher priority packets using some redundancy mechanism such as forward error correction or packet retransmission.

- Frame rate optimization framework using a movement model. It is anticipated that the frame rate optimization framework could work with some movement model to determine when to increase the number of frames in the decoder buffer. For example, suppose video is being streamed to a bus and suppose the bus is about to enter a tunnel that has no network connections. The movement model should detect this and alert the sender and receiver. The sender and receiver should then determine the amount of frame increase needed to ensure video continuity while the bus is in the tunnel, and adjust the encoder frame generation rate and playout rate accordingly.
APPENDIX A

Publications and Patents List

A.1. Publications Related to this Thesis

• Evan Tan, Jing Chen, Sebastien Ardon and Emmanuel Lochin, "Video TFRC," IEEE International Conference on Communications 2008. ICC '08, pp.1767-1771, 19-23 May 2008

A.2. Other Publications


A.3. Patent

B.1. Derivation of Decoder Buffer Lyapunov Drift Bound

We first square the buffer dynamics equation (5.5.1) to get:

\[ D(t + 1) \leq [D(t) - o(t)]^+ + \lambda(t) \]  

(B.1.1)

\[ D(t + 1)^2 \leq D(t)^2 + o(t)^2 + \lambda(t)^2 - 2D(t)o(t) + 2D(t)\lambda(t) - 2o(t)\lambda(t) \]

We then drop the last term \(-2o(t)\lambda(t)\) and divide the above equation by 2 to get:

(B.1.2)

\[ \frac{D(t + 1)^2}{2} - \frac{D(t)^2}{2} \leq \frac{o(t)^2 + \lambda(t)^2}{2} - D(t) [o(t) - \lambda(t)] \]

Note that the left-hand-side (LHS) of the above now composes of the difference between two Lyapunov functions as defined in (5.5.2). Furthermore, since \(o(t) \leq o_{\text{max}}\) and \(\lambda(t) \leq \lambda_{\text{max}} \leq i_{\text{max}}\), we can obtain:

\[
L(D(t + 1)) - L(D(t)) \leq \frac{o_{\text{max}}^2 + i_{\text{max}}^2}{2} - D(t) [o(t) - \lambda(t)]
\]

(B.1.3)

where:

(B.1.4)

\[ A = \frac{o_{\text{max}}^2 + i_{\text{max}}^2}{2} \]

If we take conditional expectations of (B.1.3) with respect to \(D(t)\), we get:

(B.1.5)

\[ \mathbb{E}\{L(D(t + 1)) - L(D(t))|D(t)\} \leq A - D(t)\mathbb{E}\{o(t) - \lambda(t)|D(t)\} \]

The LHS of the above becomes the one-step conditional Lyapunov drift \(\Delta(D(t))\) as defined in (5.5.3), so the above can be rewritten as:

(B.1.6)

\[ \Delta(D(t)) \leq A - D(t)\mathbb{E}\{o(t) - \lambda(t)|D(t)\} \]

B.2. Derivation of Discontinuity Penalty Lyapunov Drift Bound

We square the buffer dynamics equation (6.2.16), divide it by two and rearrange its terms to get:
B.2. DERIVATION OF DISCONTINUITY PENALTY LYAPUNOV DRIFT BOUND

\[ U(t+1) \leq [U(t) - p(t) - \beta(t)]^+ + F(f(t)) \]
\[ U(t+1)^2 \leq U(t)^2 + (p(t) + \beta(t))^2 + F(f(t))^2 \]  
(B.2.1)

\[-2U(t)[p(t) + \beta(t) - F(f(t))]\]

Dividing the above equation by two and rearranging, we get:

\[ \frac{U(t+1)^2}{2} - \frac{U(t)^2}{2} \leq \frac{(p(t) + \beta(t))^2 + F(f(t))^2}{2} - U(t)[p(t) + \beta(t) - F(f(t))] \]  
(B.2.2)

Using the definition of the Lyapunov function (6.2.21) in the above equation yields:

\[ L(U(t+1) - L(U(t)) \leq \frac{(p(t) + \beta(t))^2 + F(f(t))^2}{2} \]
\[-U(t)[p(t) + \beta(t) - F(f(t))] \]
(B.2.3)

Recall that \( \beta(t) \) is defined in (6.2.11) as:

\[ \beta(t) = \frac{b(t)r_{\text{max}}p_{\text{max}}}{T-t} \]
(B.2.4)

where \( b(t) \) is the amount of buffered video frames in the buffer at time \( t \) and \( T \) is the length of the whole sequence in time slots. Since a time slot corresponds to a frame in our framework, it can be seen that \( b(t) \leq T \). This is because the maximum amount of video content that can exist in the buffer is the whole sequence. This then implies that:

\[ \beta(t) \leq \frac{Tr_{\text{max}}p_{\text{max}}}{T-t} \]  
(B.2.5)

Using the above bound in (B.2.3), we obtain:

\[ L(U(t+1) - L(U(t)) \leq \frac{1}{2} \left[ \left(p(t) + \frac{Tr_{\text{max}}p_{\text{max}}}{T-t}\right)^2 + F(f(t))^2 \right] \]
\[-U(t)[p(t) + \beta(t) - F(f(t))] \]
(B.2.6)

Using (6.2.6) and \( r(t) \leq r_{\text{max}} \), we get \( F(f(t)) \leq r_{\text{max}} \). Since \( p(t) \leq p_{\text{max}} \), using these two bounds into the above equation yields:
\[ L(U(t+1) - L(U(t)) \leq \frac{1}{2} \left[ \left( p_{\text{max}} + \frac{T r_{\text{max}} p_{\text{max}}}{T - t} \right)^2 + r_{\text{max}}^2 \right] - U(t) [p(t) + \beta(t) - F(f(t))] \]

(B.2.7) \[ = B - U(t) [p(t) + \beta(t) - F(f(t))] \]

where:

(B.2.8) \[ B = \frac{1}{2} \left( r_{\text{max}}^2 + \left( p_{\text{max}} + \frac{T r_{\text{max}} p_{\text{max}}}{T - t} \right)^2 \right) \]

If we take conditional expectations of (B.2.7) with respect to \( U(t) \) and use the definition of the one-step conditional Lyapunov drift (6.2.22), we get:

(B.2.9) \[ \Delta(U(t)) \leq B - U(t) \mathbb{E}\{\beta(t) + p(t) - F(f(t))|U(t)\} \]

B.3. Derivation of Delay Constrained Lyapunov Drift Bound

We first a Lyapunov function \( L(X(t)) \) based on \( X(t) \) as:

(B.3.1) \[ L(X(t)) \triangleq \frac{w_3 X^2(t)}{2} \]

Recall for the discontinuity penalty \( U(t) \), its Lyapunov function is:

(B.3.2) \[ L(U(t)) \triangleq \frac{U^2(t)}{2} \]

Then, it can be seen that (6.3.15) can be defined as:

\[
L(U(t), X(t)) = \frac{U^2(t) + w_3 X^2(t)}{2} = \frac{U^2(t)}{2} + \frac{w_3 X^2(t)}{2}
\]

(B.3.3) \[ = L(U(t)) + L(X(t)) \]

Now, we square the buffer dynamics equation (6.3.9) to obtain:

\[
X(t + 1) \leq [X(t) - p_n - t_d]^+ + p(t)
\]

(B.3.4) \[ X(t + 1)^2 \leq X(t)^2 + p(t)^2 + (p_n + t_d)^2 - 2X(t) [p_n + t_d - p(t)] \]

Dividing the above equation by two and rearranging yields:
(B.3.5) \( \frac{X(t+1)^2}{2} - \frac{X(t)^2}{2} \leq \frac{p(t)^2 + (p_n + t_d)^2}{2} - X(t) [p_n + t_d - p(t)] \)

Multiplying the above by \( w_3 \), we get:

(B.3.6) \( \frac{w_3 X(t+1)^2}{2} - \frac{w_3 X(t)^2}{2} \leq \frac{w_3 (p(t)^2 + (p_n + t_d)^2)}{2} - w_3 X(t) [p_n + t_d - p(t)] \)

Using \( p(t) \leq p_{max} \) and (B.3.1) in the above equation yields:

\[ L(X(t+1)) - L(X(t)) \leq \frac{w_3}{2} (p(t)^2 + (p_n + t_d)^2) - w_3 X(t) [p_n + t_d - p(t)] \]

(B.3.7) \[ = C - w_3 X(t) [p_n + t_d - p(t)] \]

where:

(B.3.8) \[ C = \frac{w_3}{2} \left( p_{max}^2 + (p_n + t_d)^2 \right) \]

By adding (B.3.7) and (B.2.7) together, we get:

\[ L(X(t+1)) - L(X(t)) + L(U(t+1)) - L(U(t)) \leq \]

(B.3.9) \[ C - w_3 X(t) [p_n + t_d - p(t)] + B - U(t) [p(t) + \beta(t) - F(f(t))] \]

Using (B.3.3) and rearranging the above yields:

\[ L(U(t+1), X(t+1)) - L(U(t), X(t)) \leq B + C \]

\[ - U(t) [p(t) + \beta(t) - F(f(t))] \]

(B.3.10) \[ - w_3 X(t) [p_n + t_d - p(t)] \]

Taking conditional expectations of the above with respects to \( U(t) \) and \( X(t) \), and using (6.3.16) yields:

\[ \Delta(U(t), X(t)) \leq B + C \]

\[ - U(t) \mathbb{E}\{p(t) - F(f(t))|U(t), X(t)\} \]

(B.3.11) \[ - w_3 X(t) \mathbb{E}\{p_n + t_d - p(t)|U(t), X(t)\} \]
B.4. Derivation of Network Delayed Lyapunov Drift Bound

Note that, in this section, we use the same shortened notations defined in (6.4.8), additionally we also define $U_{df} = U(t + df + 1)$. This means that (6.4.4) can be rewritten as:

\[(B.4.1)\]

\[U_{df} \leq [U - \gamma_{df}]^+ + F_{df}\]

By squaring the above equation and dividing it by two, we obtain:

\[\frac{U_{df}^2}{2} \leq \frac{U^2}{2} + \frac{\gamma_{df}^2 + F_{df}^2}{2} - U \left[\gamma_{df} - F_{df}\right]\]

Using the definition of the Lyapunov function (6.2.21) on the above equation and rearranging:

\[L(U_{df}) - L(U) \leq \frac{\gamma_{df}^2 + F_{df}^2}{2} - U \left[\gamma_{df} - F_{df}\right]\]

Using $\gamma(t) = p(t) + \beta(t)$, $F(f(t)) \leq r_{\text{max}}^1$, $p(t) \leq p_{\text{max}}$ and (B.2.5) on the above equation yields:

\[L(U_{df}) - L(U) \leq \frac{df}{2} \left(r_{\text{max}}^2 + \left(p_{\text{max}} + \frac{T r_{\text{max}} p_{\text{max}}}{T - t}\right)^2\right)\]

\[= B' - U \left[\gamma_{df} - F_{df}\right]\]

where:

\[B' = \frac{df}{2} \left(r_{\text{max}}^2 + \left(p_{\text{max}} + \frac{T r_{\text{max}} p_{\text{max}}}{T - t}\right)^2\right)\]

Taking conditional expectations of (B.4.4) with respects to $U$ and using (6.4.5) yields:

\[\Delta(U) \leq B' - \mathbb{E}\{U \gamma_{df} \mid U\} + \mathbb{E}\{U F_{df} \mid U\}\]

APPENDIX C

Concavity of Frame Utility

\[\text{See Appendix B.2}\]
Proposition C.1. The frame utility function \( g(i(t)) \) is a concave function if \((5.6.4)\) is satisfied.

Proof. In this proof, for notational convenience, we set \( i \triangleq i(t) \) and \( ABR \triangleq ABR(t) \). Now looking into the second order derivative of \((5.6.3)\):

\[
\frac{\partial^2 g}{\partial i^2} = -\frac{2q^2a^2\xi^2}{(1 + \xi)^3i^2} + \frac{qa\xi}{(1 + \xi)^2i^2} + \frac{q^2a^2\xi}{(1 + \xi)^2i^2}
\]

Where:

\[
\xi \triangleq e^\theta \left( a \log \left( \frac{ABR}{i} \right) - c - s \right)
\]

Equation \((C.0.7)\) needs to be negative for concavity, thus, the following needs to be satisfied:

\[
\frac{2q^2a^2\xi^2}{(1 + \xi)^3i^2} \geq \frac{qa\xi}{(1 + \xi)^2i^2} + \frac{q^2a^2\xi}{(1 + \xi)^2i^2}
\]

\[
\frac{2qa\xi}{(1 + \xi)} \geq 1 + qa
\]

\[
\xi \geq \frac{1 + qa}{qa - 1}
\]

Then substituting \((C.0.8)\) into \((C.0.9)\) and solving for \( i \) would yield \((5.6.4)\). □

Appendix D

Convexity of Playout Distortion Function

Proposition D.1. \( h(o(t)) \) as defined in \((5.6.5)\) is a convex function.

Proof. The convexity of \( h(o(t)) \) can be proven by the second derivative test:

\[
\frac{\partial^2 h}{\partial o^2} = 2m_s
\]

Given that \( m_s \geq 0 \). It can be seen that \((D.0.10)\) is always positive and thus \((5.6.5)\) is a convex function. □
APPENDIX E

D. CONVEXITY OF PLAYOUT DISTORTION FUNCTION

Stability of Policies with Network Delays

**Corollary E.1.** If the network delay variation \( r(t) \) is i.i.d over the timeslots, the frame quality function is bounded as \( g(f(t)) \leq G_{\text{max}} \) and the playout distortion function is bounded as \( h(p(t)) \geq H_{\text{min}} \). Then implementing the optimization policies (6.4.16) and (6.4.17) in each timeslot stabilizes the discontinuity penalty \( U(t) \) and satisfies the following performance bounds:

\[
\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \leq \frac{B'' + V(w_1 G_{\text{max}} - w_2 H_{\text{min}})}{p_{\text{max}}}
\]

\[
\liminf_{M \to \infty} g(\bar{f}) \geq \frac{(w_1 g^* - w_2 h^*) + w_2 H_{\text{min}} - \frac{B''}{V}}{w_1}
\]

\[
\limsup_{M \to \infty} h(\bar{p}) \leq \frac{B''}{V} - \frac{(w_1 g^* - w_2 h^*) - w_1 G_{\text{max}}}{w_2}
\]

Where \( V > 0, g^*, r^*, \bar{f} \) and \( \bar{p} \) are defined as in theorem 6.7. \( B'' \) is defined as:

\[
B'' = B' + d_b (d_f + 1) (r_{\text{max}}^2 + r_{\text{max}} p_{\text{max}})
\]

with \( B' \) from (6.4.10).

**Proof.** We first look at (6.4.13). Note that with the recursive definition of \( U(t) \) (6.4.3), (6.4.13) can be rewritten as:

\[
\Delta(U) \leq B'' - U \mathbb{E}\{\gamma_{d_f} \mid U\} - \mathbb{E}\{-U_{d_b} F_{d_f} \mid U\}
\]

\[
\leq B'' - \mathbb{E}\{(\gamma_{d_f} U_{d_b} - \gamma)^+ + F_{\gamma_{d_f}} \mid U\} - \mathbb{E}\{-U_{d_b} F_{d_f} \mid U\}
\]

Since \([U_{d_b} - \gamma]^+ \leq U_{d_b}\), the above equation becomes:

\[
\Delta(U) \leq B'' - \mathbb{E}\{(\gamma_{d_f} U_{d_b} + F_{\gamma_{d_f}}) \mid U\} - \mathbb{E}\{-U_{d_b} F_{d_f} \mid U\}
\]

Note that the term \( F_{\gamma_{d_f}} \) can be upper bounded as:

\[
F_{\gamma_{d_f}} \leq d_b (d_f + r_{\text{max}} p_{\text{max}})
\]

Using (E.0.17) in (E.0.16) yields:
\[ \Delta(U) \leq B''' \mathbb{E} \left\{ \gamma_{df} U_{db} - U_{db} F_{df} \middle| U \right\} \]

with \( B''' \) defined in (E.0.14). This then implies that (6.4.14) can be bounded as:

\[
\Delta(U) - V \mathbb{E} \{ w_1 g(f(t)) - w_2 h(p(t)) \middle| U \} \\
\leq B'' - \mathbb{E} \left\{ U \gamma_{df} - U_{db} F_{df} \middle| U \right\} \\
- \mathbb{E} \left\{ V w_1 g(f(t)) - V w_2 h(p(t)) \middle| U \right\} \\
\leq B''' - \mathbb{E} \left\{ U_{db} \gamma_{df} - U_{db} F_{df} \middle| U \right\} \\
- \mathbb{E} \left\{ V w_1 g(f(t)) - V w_2 h(p(t)) \middle| U \right\}
\]

(E.0.19)

The proof then proceeds as in theorem 6.7
Bibliography


