OPTIMAL STREAMING OF RATE ADAPTABLE VIDEO

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EREN GÜRSES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2006

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Nail Akar
Co-Supervisor

Assoc. Prof. Dr. Gözde Bozdağı Akar
Supervisor

Examining Committee Members

Prof. Dr. Semih Bilgen (METU,EE) _____

Assoc. Prof. Dr. Gözde Bozdağı Akar (METU,EE) _____

Prof. Dr. Hitay Özbay (Bilkent University,EE) _____

Prof. Dr. Buyurman Baykal (METU,EE) _____

Prof. Dr. Reha Civanlar (Koç University,CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required, I have fully cited and referenced all material and results that are not original to this work.

Name Lastname : Eren GÜRSES

Signature :

ABSTRACT

OPTIMAL STREAMING OF RATE ADAPTABLE VIDEO

Gürses, Eren

Ph.D., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Gözde Bozdağı Akar

Co-Supervisor: Assoc. Prof. Dr. Nail Akar

June 2006, 93 pages

In this study, we study the dynamics of network adaptive video streaming and propose novel algorithms for rate distortion control in video streaming. While doing so, we maintain inter-protocol fairness with TCP (Transmission Control Protocol) that is the dominant transport protocol in the current Internet. The proposed algorithms are retransmission-based and necessitate the use of play-back buffers in order to tolerate the extra latency introduced by retransmissions. In the first part, we propose a practical network-adaptive streaming scheme based on TCP transport and the idea of Selective Frame Discarding (SFD) that makes use of two-layer temporally scalable video. The efficacy of the SFD scheme is validated for playout buffer times in the order of seconds and therefore makes it suitable more for delay tolerant streaming applications.

In the second part of the thesis, we propose an application layer rate-distortion control algorithm which provides Optimal Scheduling and Rate Control (OSRC) policies in the average reward sense in order to achieve efficient streaming of video. The Optimal Scheduling (OS) we propose maximizes the probability of successfully on time delivery according to a prespecified set of rate constraints, and different channel conditions by using Markov Decision Process (MDP) models. On the other hand optimal rate control (RC) is achieved by calculating the optimal rate constraint which minimizes the average distortion of a video streaming session by making use of the video distortion model derived for lossy channels and achievable success probabilities provided by the set of optimal schedules. For numerical examples, we focus on an equation-based TCP friendly rate control (TFRC) protocol where transport layer retransmissions are disabled and Fine Granular Scalable (FGS) coded video is used for improved rate adaptation capabilities but with an additional rate distortion penalty. The efficacy of the proposed OSRC algorithm is demonstrated by means of both analytical results and ns-2 simulations.

Keywords: video streaming, rate adaptable video, Markov decision processes, TCP-friendly rate control

ÖZ

AYARLANABİLİR HIZLI VİDEONUN EN İYİ İLETİMİ

Gürses, Eren

Doktora, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Gözde Bozdağı Akar

Ortak Tez Yöneticisi: Doç. Dr. Nail Akar

Haziran 2006, 93 sayfa

Bu çalışmada ağ durumuna göre ayarlanabilir video iletiminin dinamikleri analiz edilmiş ve elde edilen sonuçlar doğrultusunda hız bozunum kontrolü yapabilen ve TCP (İletim Kontrol Protokolü) trafiği ile denkserlik sağlayan video iletim algoritmaları önerilmiştir. Önerilen algoritmalar yeniden iletime dayalı olarak çalışmakta ve bu sebeple yeniden iletimlerden kaynaklanan gecikmeyi tolere edecek izleme önbelleği kullanımını gerekli kılmaktadır. İlk olarak TCP iletim protokolü üzerinde koşan ve uyguladığı seçici çerçeve düşürme (SFD) algoritması ile ağ koşullarına kendini uyarlayabilen, basit, etkili ve özgün bir yöntem önerilmiştir. Zamansal olarak hızı ayarlanabilir iki katmanlı video kullanımıyla gerçeklenen SFD ancak saniyeler mertebesinde ilk izleme önbelleği kullanılması durumunda başarılı sonuçlar vermiştir.

Tezin ikinci bölümünde ise uygulama katmanında yapılacak En iyi paket iletim Politikalarını ve en iyi Hız Kontolünü (OSRC) ortalama ödül kriterine göre bulan bir hız bozunum kontrolü algoritması önerilmiştir. En iyi paket iletim Politikaları (OS) Markov Karar Süreçlerini (MDP) kullanarak verilen kanal koşulları ve hız kısıtı altında paketlerin zamanında başarılı olarak yerine ulaşması olasılığını ortalama ödül kriterine göre en büyüten politiklar olarak bulunmaktadır. Diğer taraftan en iyi Hız Kontrolü (RC) ise video bozunum modeli ve OS ile bulunan ortalama başarılı iletim olasılığı bilgisini kullanarak video bozunumunu en küçülten en iyi video hız kısıtının bulunması ile yapılmaktadır. Sayısal örnekler için, yeniden iletim yapmayan TCP-dostu hız kontrolü (TFRC) protokolü ve hız bozunum kaybına rağmen yüksek çözünürlüklü hız ayarlamalı (FGS) kodlanmış video kullanılmıştır. Önerilen algoritmanın etkinliği analitik sonuçlar ve ns-2 simülasyonları gösterilmiştir.

Anahtar Kelimeler: video iletimi, ayarlanabilir hızlı video, Markov karar süreçleri, TCP-dostu hız kontrolü

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Assoc. Prof. Dr. Gözde Bozdağı Akar and co-supervisor Assoc. Prof. Dr. Nail Akar for their guidance, advice, criticism, encouragements and insight throughout the research. Furthermore the author wants to give his special thanks to his wife Ms. Elif Aytek Gürses for her advices, patience, and lifetime support which all play an important role in the completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AF	Assured Forwarding
AIMD	Additive Increase Multiplicative Decrease
ARQ	Automatic Repeat reQuest
ASFD	Adaptive Selective Frame Discard
BCC	Binomial Congestion Control
DCCP	Datagram Congestion Control Protocol
ECN	Explicit Congestion Notification
FEC	Forward Error Correction
FGS	Fine Granular Scalable
GOP	Group of Pictures
IR	Incremental Redundancy
ISA	Iterative Sensitivity Adaptation
LP	Linear Programming
MD	Multiple Description
MDP	Markov Decision Process
MTU	Maximum Transmission Unit
OSRC	Optimal packet Scheduling and Rate Control
PHB	Per Hop Behavior
PSNR	Peak Signal to Noise Ratio
QoS	Quality of Service
RaDiO	RAte DIstortion Optimized
RARQ	Retransmission based Analytical Rate and Quality controller
RAP	Rate Adaptation Protocol
RED	Random Early Detect
SCTP	Stream Control Transmission Protocol
SFD	Selective Frame Discard
SLA	Service Level Agreement
TFRC	TCP Friendly Rate Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

CHAPTER I

INTRODUCTION

The transmission of high quality video over the Internet has become commonplace due to recent advances in video compression and networking fields, development of efficient video coders/decoders and increasing interest in applications such as live video, video on demand, videophone, and video conferencing. However, the Internet is inherently a lossy packet network and the available bandwidth to a certain user changes in all time scales because of its very dynamic nature. These characteristics of the Internet led to the rise of network-adaptive video applications so as to provide best end-to-end delivery while adapting to the network conditions such as delay, error and bandwidth, which can be gathers by means of network feedback. Network adaptive streaming applications adapt themselves to the observed channel conditions by means of implementing their own rate distortion control mechanisms in order to provide best end-to-end delivery while stipulating inter-protocol fairness with TCP traffic which is the transport protocol of choice for most applications in the current Internet.

Video streaming applications require a playout buffer at the receiver which is

drained by the decoder once it fills up. This buffer is used to provide a smoother display at the receiver and its size is determined by the jitter and the delay of the network, the error correction algorithm employed at the receiver, and application's latency tolerance. Nevertheless, this playout buffer introduces a latency and streaming applications are classified as non-interactive and interactive video depending on their latency requirements. In the current Internet, non-interactive video streaming applications such as live video and on demand video distribution play an important role and constitute the majority of existing video traffic. These applications require timely delivery of their data but have looser latency constraints as compared to the stringent delay requirements of the interactive video streaming applications such as videophone, video conferencing and interactive games. In order to be more concrete, the maximum allowed latency for non-interactive video may extend from a few hundred milliseconds to seconds or even minutes, whereas in the interactive case this is only limited to on the order of 150 msec. [60]. However, despite the delay tolerant nature of non-interactive applications, generally video streaming applications that require long buffering are not preferred since users should wait for the same buffering time in every repositioning action within a stream (i.e. forward and rewind). Hence, especially in high latency channels, forward error correction (FEC) [46],[44],[43] or multiple description (MD) [45] coding techniques are used in both non-interactive and interactive video streaming applications. These schemes avoid the drawbacks of startup latency since they remove the need for retransmissions, however they suffer from a rate distortion penalty as a result of introduced coding redundancy.

Hence in both interactive and non-interactive cases, decision about redundancy and video bit rate is given by the *rate distortion control* mechanisms of network adaptive streaming.

I.1 Rate-Distortion Control

Rate-distortion control is a mechanism aiming to calculate optimal redundancy injection rate into the network in addition to its scheduling policy, while adapting the video bit rate accordingly in order to match with the available bandwidth estimate. Redundancy may be generated by means of either retransmissions, multiple description codes or forward error correction codes and this redundancy is used to minimize the average distortion resulting from network losses during a streaming session.

As opposed to retransmission-based schemes, use of channel codes or multiple description codes results in a rate distortion penalty due to the constant injection of redundancy into the bit stream irrespective of whether frames in transit are lost or not. Hence this rate distortion penalty becomes significant in applications that allow a sufficiently long initial playout buffering time, denoted by T_p, as given in Fig. I.1. Furthermore, FEC codes provide little benefit in case of burst losses [40],[41], although it handles isolated losses quite well. Disadvantages of FEC and even MD coding led to an increased interest in using intelligent ARQ (Automatic Repeat Request) [47],[48],[50],[57] hybrid FEC/ARQ [51],[52] and incremental redundancy (IR) transmission [56] based streaming techniques for non-interactive applications. In [47], an intelligent ARQ scheme is proposed

which uses packet deadlines as a priority metric, nevertheless it is heuristic-based and an optimization is not sought. The pioneering work in the literature that makes use of the Markov Decision Processes (MDP) for the calculation of optimal packets (re)transmission schedules for a two layer stream is [25], however it is far from practicality. Later in Chou's seminal paper [50] a very flexible and practical MDP framework based on the idea of transmitting packets in a rate distortion optimized (RaDiO) manner. To the best of our knowledge, it has been one of the most popular algorithms in the literature and the rate distortion optimized decisions are given by minimizing a Lagrangian cost function that is calculated by using rate and distortion information, channel statistics, packet deadlines and transmission history. However it suffers from the high complexity of the utilized iterative multivariate minimization solution, therefore some low complexity and suboptimal versions are proposed [53]. In [55], a path diversity streaming scheme is developed by using the RaDiO framework whereas in [56], a hybrid FEC/ARQ scheme which is known as incremental redundancy (IR) is developed by using a framework similar to the RaDiO framework. Performance of RaDiO based algorithms depend on the observability of the system state which is supplied by ACKs. In [57], the sender estimates the receiver status by using a probability distribution and calculates optimal policy using a partially observable Markov decision process (POMDP) model.

The redundancy injection schemes constitute one important dimension of rate distortion control by means of minimizing the distortion of a streaming session over lossy channels. However, these schemes assume that video rate is

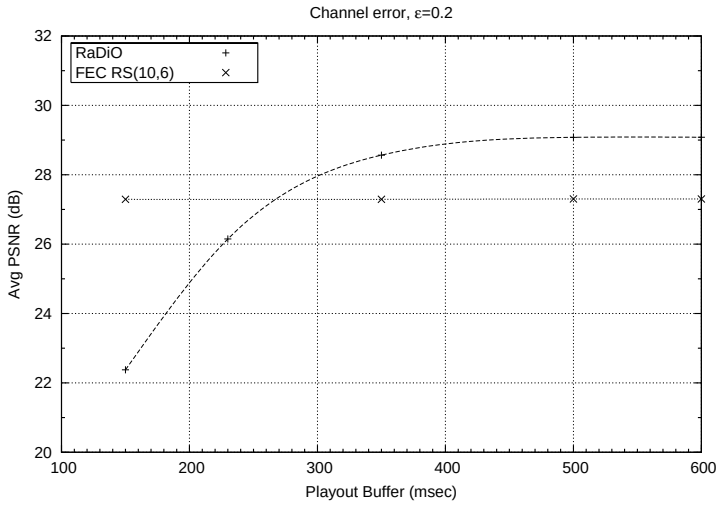

Figure I.1: Retransmission based RaDiO streaming vs. static FEC based streaming for varying T_p

adaptable in the sense that it can be scaled up or down in order to fit the video transmission rate to the remaining available bandwidth after taking the amount of injected redundancy into consideration. Several mechanisms are proposed for rate adaptation including stream switching as in the SureStream technology provided by RealSystem G2 [6],[7], rate-adaptive video encoding/transcoding [8], or joint use of scalable coding (i.e., layered coding) and rate shaping via server-side selective frame discard [9] as discussed in Chapter II.

I.2 Inter-Protocol Fairness

Besides network adaptivity, another challenging issue for the stored video streaming problem over the Internet is to provide inter-protocol fairness. TCP (Transmission Control Protocol) is the de-facto transport protocol for data in the current Internet. TCP is designed to offer a fully reliable service which is suitable

for applications like file transfers, e-mail, etc. On the other hand, the alternative transport protocol UDP (User Datagram Protocol) used by many current streaming applications does not possess congestion control. Consequently, when UDP and TCP flows share the same link, TCP flows reduce their rates in case of a packet drop. This leaves most of the available bandwidth to unresponsive UDP flows leading to starvation of TCP traffic in case of substantial UDP load. Some believe that the current trend in using UDP as the transport layer without congestion control can lead to a congestion collapse of the Internet due to the rapid growth of such applications like Internet telephony, streaming video, and on-line games [3]. Taking into consideration the dominance of TCP in today's Internet traffic, it is therefore desirable that the throughput of a video streaming session be similar to that of a TCP flow under the same network circumstances (i.e., two sessions simultaneously using the same network path). Such a mechanism is called TCP-friendly and TCP friendly schemes need to be designed to be cooperative with TCP flows by appropriately reacting to congestion [3]. There are a number of TCP-friendly congestion control algorithms which have recently been proposed, such as the rate-based RAP (Rate Adaptation Protocol) [16], equation-based TFRC (TCP-Friendly Rate Control) [4],[5], and window-based SCTP (Stream Control Transport Protocol) [21] or BCC (Binomial Congestion Control) [17]. The transmission rates of the proposed TCP-friendly algorithms are generally smoother than that of TCP under stationary conditions at the expense of reduced responsiveness to changes in the network state (e.g., a new session arrival/departure to/from the bottleneck link) [18]. Moreover, these

TCP-friendly mechanisms do not provide reliable transfer as TCP does, making them more suitable for real-time applications. DCCP, the Datagram Congestion Control Protocol, is a new transport protocol being developed by the IETF that provides a congestion-controlled flow of unreliable datagrams [19]. TCP-like congestion control without reliability and the equation-based TFRC [5] form the basis for the two congestion control profiles ID 2 and ID 3, respectively, in the DCCP protocol suite [23], [22].

I.3 Scope and Outline of the Thesis

In this thesis, we address the problem of network adaptive video streaming. Our aim is to achieve rate-distortion control while providing inter-protocol fairness with TCP flows.

In Chapter II we give a brief summary of the techniques used for video rate adaptation.

In Chapter III, we introduce a simple and effective algorithm [39] which is proposed to be used with TCP transport. The proposed algorithm is novel in terms of making prioritization between base layer (BL) and enhancement layer (EL) frames by means of a Selective Frame Discard (SFD) based input buffer management scheme which makes use of frame display time estimates. Even though the use of TCP as a transport protocol has some unique advantages due to its transparency to firewalls, ubiquity in all computer systems, and its identical response to congestion with TCP traffic that dominate the current Internet, it inevitably introduces high latency due to its inherent retransmission

mechanisms which aim to provide data integrity rather than efficient and on time delivery. Therefore from Chapter IV on, we focus on algorithms that provide application layer retransmissions that work in cooperation with the congestion control mechanism of transport layer where transport layer retransmissions are disabled.

In Chapter IV, we propose an application layer analytical rate and quality fluctuation control mechanism which applies a timer driven ARQ mechanism for retransmissions (RARQ). The timer based ARQ streaming system is modeled as a Markov decision process (MDP) and solved by Linear Programming (LP) in order to obtain the optimal policy. The results are compared with RaDiO streaming algorithm. The distortion control capability of this scheme is determined by the initial playout buffer time and network delay. From this fact the RARQ model cannot adapt to changes in channel delay and error in order to provide a better distortion control, unless the playout buffer time is increased during a streaming session by means of adaptive playout techniques. On the other hand this model does not scale well with the increasing playout buffer time. Therefore we propose a new packet scheduling algorithm in Chapter V which is robust and adaptable to changing conditions.

The proposed algorithm in Chapter V uses MDP models for finding optimal policies for different channel statistics. However this algorithm is novel since it conducts a joint optimization procedure for the calculation of optimal packet schedules (policies) and optimal rate control parameters λ^*. Similar to OSRC, RaDiO finds optimal policies using an online and costly optimization algorithm

(ISA) however it is not designed to calculate an optimal video rate $B^* = C/\lambda^*$ for a channel with given statistics and capacity C. Therefore we do not present any comparisons with RaDiO in this thesis, instead we concentrate on the algorithm's TCP friendliness, policy switching capability and the resulting performance for dynamically changing network conditions by means of using an ns-2 simulator.

Finally we conclude in Chapter VI.

CHAPTER II

RATE ADAPTABLE VIDEO

Several mechanisms are proposed for rate adaptation including stream switching as in the SureStream technology provided by RealSystem G2 [6],[7], rate-adaptive video encoding/transcoding [8], or joint use of scalable coding (i.e., layered coding) and rate shaping via server-side selective frame discard [9]

One common way of providing rate adaptivity is making real-time video transcoding [8]. In this method the video is not stored in rate adaptable format, however the video rate is matched to the network resources by means of changing coding parameters appropriately on the fly while still using the information provided by some components like DCT or motion estimation from the original bit stream. Although these schemes do not re-encode the bit stream from scratch, they are still computationally complex and not feasible to support hundreds or thousands of streaming sessions at the same time.

Another popular video rate adaptation scheme is bit stream switching [6] which has low computational complexity. This scheme is used in real world systems, however bit stream switching has some disadvantages due to its extra

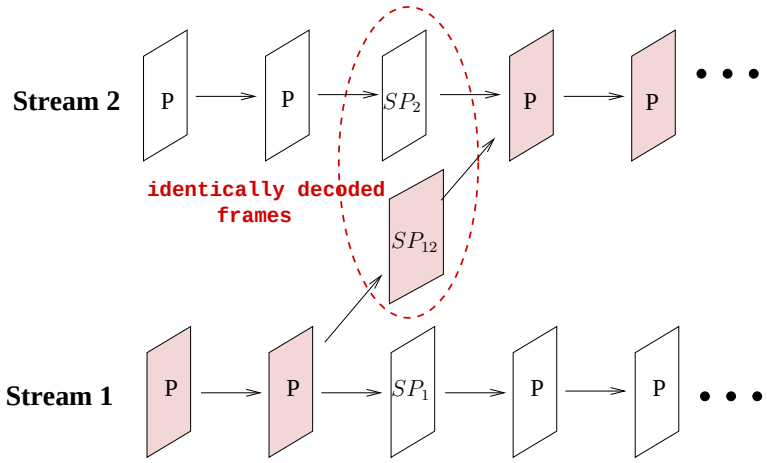

Figure II.1: Bit stream switching using an SP frame

storage requirement for multiple bit streams encoded at different rates and limited rate adaptation capability. In the new H.264 AVC standard, the concept of SI and SP frames [62] are introduced in order to facilitate bit stream switching. In Figure II.1 bit stream switching is illustrated where darker frames indicate the transmitted ones as a result of bit stream switching.

Other techniques presented in this chapter are related to the scalable representation of video signals. The main goal of all these video bit rate adaptation techniques, including transcoding, bit stream switching and scalable coding, is to flexibly support a heterogeneous set of receivers with different access bandwidths and display capabilities. On the other hand scalable coding yields a bit stream with different priority levels which in turn becomes amenable to prioritized transmission [63],[44],[43],[39]. However despite its advantages, scalable coding introduces a rate distortion penalty. Several scalable video-coding techniques have been proposed over the past few years for real-time Internet applications in the form of several video compression standards such as MPEG-2/4 and

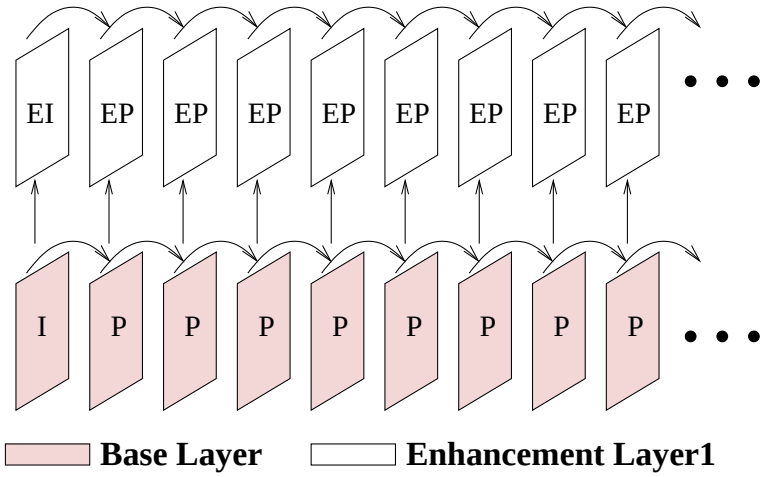

Figure II.2: Base and enhancement layers in a two layer SNR scalability mode

H.263/H.264 [10],[11],[12],[13],[58]. Popular and well known scalability profiles are spatial, SNR and temporal scalability [15]. Spatial scalability corresponds to the representation of video signal at different spatial dimensions and only limited number of work is based on that profile. However SNR scalability is a more frequently referred scalability profile, where the original video signal is represented as layers and these layers correspond to different SNR levels. An illustration of the SNR scalable coding is given in Figure II.2 with two layers where the layers are represented as base layer (BL), enhancement layer1 (EL1) and etc. In SNR scalability it is common to make predictive coding in the enhancement layers.

Temporal scalability is provided by means of either I-B-P coding of MPEG-2 or Reference Picture Selection (RPS) mode of H.263+. In Figure II.3 a schematic diagram of RPS mode is given such that Intra (I) and anchor P (predicted) frames constitute the base layer (BL) whereas the remaining P frames are denoted as the enhancement layer (EL). From the prioritization point of view in a two layer scalable video base layer frames are denoted by H (High-priority),

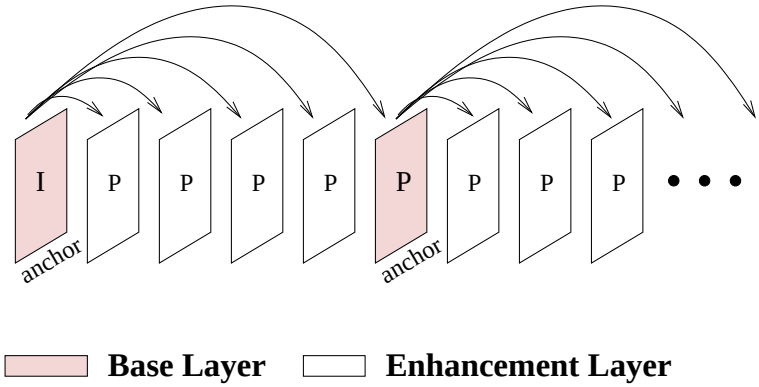

Figure II.3: Base and enhancement layers in temporal scalability mode

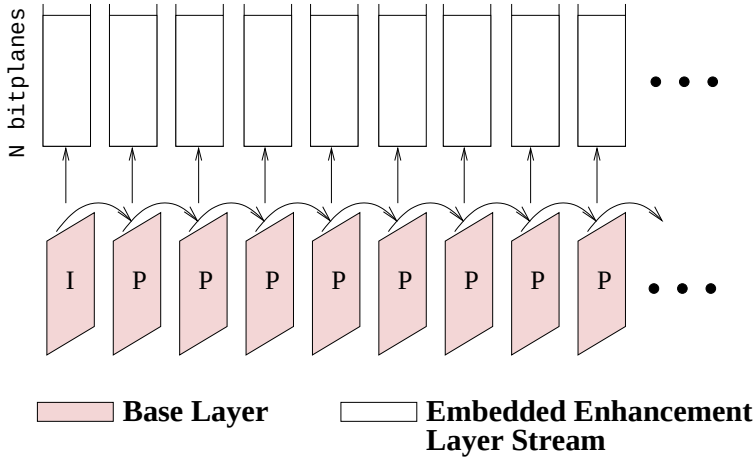

Figure II.4: Base and enhancement layers bit streams in FGS mode

whereas the enhancement layer frames by L (Low-priority).

The more recent Fine Grain Scalability (FGS) coding [14] allows the enhancement frame to be encoded independently with an arbitrary number of bits and the bit rate can thus be adjusted at transmission time for finer granularity. Despite its perfect rate adaptation capability, FGS introduces a high rate-distortion penalty due to removal of predictive coding between the enhancement layers. A schematic diagram of FGS coding is demonstrated in Figure II.4 where the embedded enhancement layer bit stream can be cut from anywhere which is labeled as *N bitplanes*.

Bit stream switching does not offer a fine granularity since there are only a few bit streams available among which the streaming server can switch. Rate-adaptive encoding/transcoding is more appropriate for live video streaming or interactive video applications as opposed to the stored video streaming problem we address in this thesis. We therefore focus on rate adaptation using scalable encoded bit streams.

CHAPTER III

SERVER-SIDE ADAPTIVE FRAME DISCARD

This chapter addresses the problem of TCP-friendly on-demand streaming of temporally scalable stored video over the Internet using server-side adaptive frame discarding. In a stored video-on-demand system, the server prestores the encoded video and transmits it on demand to a client for playout in real time. The client buffers the data and starts playout after a short delay in the order of seconds (called the playout delay and denoted by T_p). We assume a fixed T_p throughout this work as opposed to the adaptive playout schemes where the client buffering delay is varied with respect to the network conditions [1],[2]. It is this tolerability to larger playout delays that distinguishes the stored video streaming problem from other video networking applications like videophone, video conferencing, and live video streaming. It is also very desirable that once the playout begins, it should be able to playout without any interruption (i.e., smooth playout) until the end of the video streaming session. Moreover, such a transmission strategy should not jeopardize the data flows on the same network

path which use TCP as their transport protocol, which is referred to as the "TCP-friendliness" requirement [3],[4],[5].

The stored video streaming problem over resource constrained networks, like the Internet, has attracted the attention of many researchers. Given network bandwidth and client buffer constraints, a dynamic programming algorithm with reportedly significant computational complexity is developed for the optimal selective frame discard problem in [9] as well as several heuristic algorithms. However, this study is unable to accommodate the bandwidth variability patterns of the Internet since the network bandwidth is assumed to be fixed and a-priori known. On a similar ground, rate-distortion optimization-based video streaming algorithms have been developed in [25] and [50] that obtain scheduling policy for both new and retransmitted frames using stochastic control principles but the proposed methods are relatively complex and their feasibility remain to be seen. The reference [26] considers a practical frame dropping algorithm for MPEG streams over best-effort networks but they neither use a TCP-friendly congestion control algorithm nor they take into account the deadlines of frames. In [27], a dynamic frame dropping filter for MPEG streams is proposed in a network environment where the available bandwidth changes dynamically but this work also lacks the TCP-friendliness component. A number of studies focus on streaming video using new TCP-friendly transport protocols [16],[5] while others employing TCP itself [28],[29],[30],[31]. One common objection to use of TCP for streaming applications is the fully reliable service model of TCP through retransmissions [30]. While delays due to retransmissions may not be

tolerable for interactive applications, the service model for TCP may not be problematic for video on demand applications, which is the scope of this chapter [30]. Moreover, the use of ECN (Explicit Congestion Notification) allows TCP [20] or other ECN conformant congestion control mechanisms like SCTP [21], DCCP CCID-2 [22], TFRC and TFRC-like DCCP CCID-3 [23], to perform congestion avoidance without losses, limiting further the potential adverse effect of the underlying transport layer service model.

In this chapter, we propose a stored video streaming system architecture which consists of an input buffer at the server side coupled with the congestion control scheme of TCP at the transport layer, for efficiently streaming stored video over the best-effort Internet. The proposed method can be made to work with other transport protocols including DCCP but our choice of TCP in this chapter as the underlying transport protocol stems from the following reasons:

- Slowly-responding algorithms like RAP [16] or some BCC [17] profiles perform reasonably well in terms of video goodput in stationary conditions. However, responsiveness is especially critical in the core of the Internet today which appears to be operating in the transient rather than in the stationary regime due to the large session arrival and/or departure rates to/from the network. On the other hand, TCP congestion control has a well-established responsiveness to changing network state and may be more appropriate in rapidly changing environments.

- TCP with its original congestion control but with its full reliability fea-

ture replaced with selective reliability would be a more appropriate fit as a transport protocol for the underlying problem but the standards in this direction have not finalized and are still evolving [19],[22]. We note that TCP's insistence on reliable delivery without timing considerations would adversely affect the performance of the system under packet losses especially for (near) real-time applications (e.g., applications requiring short playout delays). In this chapter, we study the regimes for which TCP performance for stored video streaming is acceptable but also identify regimes for which TCP performs poorly and a new transport protocol would be needed.

- TCP is currently used for streaming applications in order to get through some firewalls that block UDP traffic.

- The choice of either TCP or other transport protocols that react to congestion, eliminates the unnecessary burden on the application-level designer by providing congestion control at the transport layer.

- Another key advantage related to providing congestion control at the transport layer (i.e., TCP or other ECN conformant congestion control [24]) rather than "above UDP" is that the proposed scheme can make use of the services provided by the standard-based Explicit Congestion Notification (ECN) mechanism [32] which provides a means of explicitly sending a "congestion experienced" signal towards the TCP sender in TCP acknowledgment packets. We note that explicit feedback significantly reduces the

losses in the network and is therefore particularly useful in scenarios such as video streaming where the frequency of retransmissions due to losses is to be kept at a minimum.

In our proposed architecture, the buffer management scheme selectively discards low priority frames from its head-end which otherwise would jeopardize the successful playout of high priority frames. Moreover, the proposed discarding policy is adaptive to changes in the bandwidth available to the video stream. Contrary to many of the previously proposed adaptive transmission algorithms, the proposed Selective Frame Discard (SFD) strategy is simple and is easily implementable at the application layer by allowing additional information exchange between the transport layer and the application layer. Moreover, our proposed server-side frame discarding algorithm only needs to know the playout delay T_p and several network-related variables which are made available by using the services of TCP and the playout buffer occupancy does not need to fed back to the server in this proposed scheme. Our simulation results demonstrate that scalable stored video can efficiently be streamed over TCP with the proposed adaptive frame discarding strategy if the client playout delay is large enough to absorb the fluctuations in the TCP estimation of the available bandwidth. We also study the impact using Explicit Congestion Notification (ECN) in the network in terms of attained video quality. Finally, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) Assured Forwarding (AF) Per-Hop-Behavior (PHB) architecture (see [33]) in the context of stored video streaming and identify regimes in which the

former architecture outperforms the latter.

The rest of this chapter is organized as follows. In Section III.2, the proposed selective frame discard architecture is presented where the utilized two layer temporal scalability scheme is described in Chapter III.1 The simulation platform and the numerical results are given in Section III.3 and concluded in the final section.

III.1 Scalable Video Coding for SFD

In this work, we assume that the stored video is encoded into temporally scalable video composed of two layers, the BL and the EL, using the Reference Picture Selection mode of H.263 version 2 [12],[13]. In this structure, the BL is composed of Intra (I) and anchor P (predicted) frames whereas the EL is composed of the remaining P frames as illustrated in Figure II.3. P frames in the EL are estimated using the anchor P frames or I frames in the BL where anchor P frames are chosen using the Reference Picture Selection mode. Throughout the rest of this chapter, we will denote the base layer frames by H (High-priority), and enhancement layer frames as L (Low-priority). A schematic diagram of the employed temporally scalable video coding structure is shown in in Figure II.3 of Chapter II.

III.2 Selective Frame Discarding

As stated in the previous section, we assume that video encoders generate H- and L-frames. If the available network bandwidth cannot accommodate the

transmission of all frames, then it would be desirable to discard some of the L-frames on behalf of the H-frames. While making a L-frame discarding decision, our goal is to maximize the number of transported L-frames subject to the constraint that the loss rate for the H-frames would be minimal. In this definition, a loss refers to a missed frame at the client either because the frame is not transmitted by the server or is transmitted but partially/completely lost in the network or the frame is received by the client but after its deadline. For this purpose, we propose an input buffer implemented at the application layer of the sender which dynamically and intelligently discards L-frames from its headend and this scheme is depicted in Fig. III.1.

We use the RTP/TCP/IP protocols stack in this chapter. We propose in this architecture that the stored video frames arrive at the input buffer at a frequency $f = 1/T$ frames per second, which is the frame generation rate of the underlying video session. These frames wait in the input buffer until they reach the headend of the buffer and a decision is then made by the Selective Frame Discard (SFD) block whether the corresponding frame should be passed

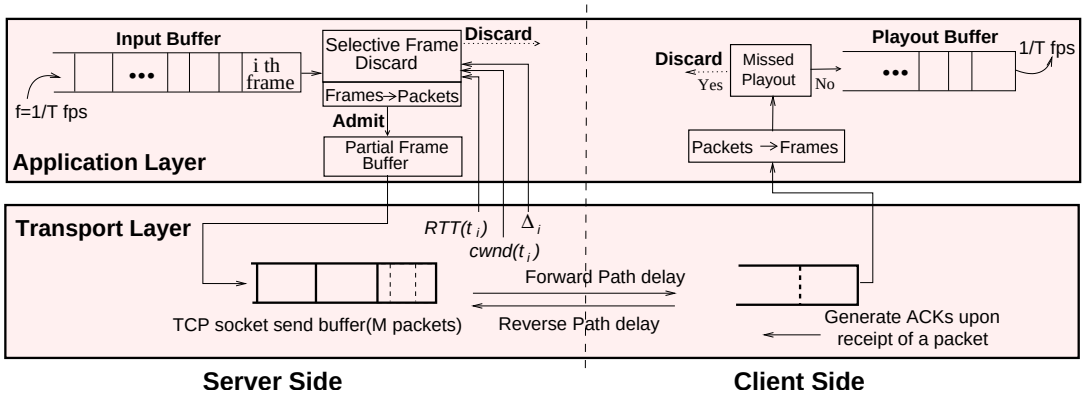

Figure III.1: Proposed stored video streaming architecture

towards the transport layer or is simply discarded. In cases of discard, the SFD block will make subsequent discard decisions until an acceptance decision is made. When a frame is accepted by the SFD module, it is segmented into video packets (or RTP packets) of length at most L where we fix L to 1 Kbytes in this study. In our simulation studies, QCIF videos are encoded at around 30 dB quality and a typical video packet can carry 1-3 P-frames depending on the compression efficiency of the frame (i.e. high/low motion) and a typical I-frame can be transported by 2-3 video packets. Video packets of accepted frames are first placed in the partial frame buffer which is then drained by the TCP layer. We suggest that whenever a TCP packet begins to take its first journey towards the network, the TCP layer immediately retrieves a packet from the partial frame buffer if the buffer is nonempty. Otherwise, it queries the SFD module to make an acceptance/rejection decision on the head-end frame.

The acceptance/rejection decision is made as follows: The decision epoch for the ith frame is denoted by t_i irrespective of the outcome of the decision. The waiting time or the shaping delay in the input buffer for frame i, denoted by $D_{i,S}$, is the difference between t_i and the injection time for the i^{th} frame to the input buffer. Let $D_{i,N}$ denote the network delay for the ith frame injected into the input buffer. Recalling that frames are generated by the encoder at integer multiples of T, the injection time for the ith frame to the input buffer will be $t_0 + iT$, where t_0 is the injection time of the 0th frame. The ith frame will then wait in the input buffer for $D_{i,S}$ seconds and the SFD module will make an admit/discard decision for the ith frame at time epoch $t_i \stackrel{\triangle}{=} t_0 + iT + D_{i,S}$. If

the ith frame is admitted by the SFD module into the transport layer then that frame will be delayed an additional $D_{i,TCP}$ and $D_{i,N}$ seconds in the TCP buffer and in the network, respectively. It is clear that the ith frame must arrive at the receiver before its playout time $t_0 + D_{0,N} + T_p + iT$ where T_p is the initial buffering time of the playout buffer which starts accumulation as soon as the frame 0 arrives. So the following inequality should be satisfied for every accepted frame $i > 0$ for its successful playout:

$$D_{i,S} \leq T_p - (D_{i,N} - D_{0,N}) - D_{i,TCP} \tag{III.1}$$

In the above inequality, $D_{i,S}$ and T_p are known to the SFD module, however one needs to find estimates for the last two terms on the right hand side of the inequality. In this study, we suggest to estimate the one-way network delay difference $\Delta_i = D_{i,N} - D_{0,N}$ using the TCP Timestamps option (TSopt) in TCP headers [34]. In the TCP Timestamps Option, while transmitting packet m, the sender puts the transmission instant timestamp in the *TSval* (Timestamp Value) field. After receiving packet m, the receiver generates an acknowledgment packet denoted by ack m, by setting its *TSval* field with the current time of the receiver and by copying the *TSval* field of packet m to the *TSecr* (Timestamp Echo Reply) field of ack m. In this way, the SFD module will have an estimate of the one-way network delay difference using the TCP timestamp option for the last acknowledged TCP packet before time t_i when it needs to make a decision for frame i. On the other hand, the last term $D_{i,TCP}$ is not known in advance but is relatively small compared to T_p unless there are TCP losses because of the mechanism described for initiating a data transfer from the application layer

into the TCP layer. We therefore introduce a safety parameter α, $0 < \alpha < 1$ to account for the errors due to inaccuracies due to estimations to be used in the inequality (III.1) as follows. In order for an admission decision for frame i to take place, the following new inequality should be checked by the SFD block:

$$D_{i,S} \leq \alpha(T_p - \Delta_i) \tag{III.2}$$

The inequality (III.2) can be used to select which frames to discard for non-scalable video but it needs to be modified for layered video. This modification is studied next.

III.2.1 Static and Adaptive Selective Frame Discard Algorithms

We propose to use two different safety parameters α_L and α_H for the L-frames and the H-frames, respectively, for preferential treatment for H-frames. Such a treatment is possible by choosing $\alpha_L < \alpha_H$. This choice makes α_L not only a safety parameter but also a prioritization instrument. We summarize the general SFD algorithm at decision epoch t_i in Table III.1.

The choice of the algorithm parameters α_L and α_H are key to the success of the proposed architecture. In Static SFD (SSFD), fixed α_L and α_H values are

Table III.1: The pseudo-code for the SFD algorithm at time t_i

```
if ((frame i == L-frame) && (D_{i,S} < α_L(T_p − Δ_i) ) {
   Admit();
} else if ((frame i == H-frame) && (D_{i,S} < α_H(T_p − Δ_i) ) {
   Admit();
} else Discard();
```

used throughout the video streaming session. However, such a fixed policy may not work well in all possible traffic scenarios. For example in cases where the instantaneous available bandwidth is close to the the BL rate then the L-frames should aggressively be discarded (i.e., $\alpha_L \rightarrow 0$) in order to minimize the loss probability of the BL frames. On the other hand, if the available bandwidth happens to be close to or exceeds the total rate of the BL and the EL frames, then the L-frames should conservatively be discarded (i.e. $\alpha_L \rightarrow \alpha_H$) . The very dynamic nature of the Internet may lead to significant variations in the available bandwidth even during the lifetime of a video session. Moreover the instantaneous BL and EL rates for VBR encoded video may substantially deviate from their long-run average values. These observations lead us to an adaptive version of the SFD algorithm. For this purpose, we define $C(t)$ as a smoothed estimate of the bandwidth available to the session at time t, where $C_i = C(t_i)$ is simply the weighted average of C_{i-1} and the instantaneous rate of TCP which is found by $cwnd_i/RTT_i$. Also we let $R_L(t)$ and $R_H(t)$ be the smoothed estimates of the EL and the BL, respectively, by monitoring the frame arrivals to the input buffer. We also let C, R_L and R_H denote the time averages of of the waveforms $C(t)$, $R_H(t)$, and $R_L(t)$, respectively. We then propose the simple Adaptive SFD (ASFD) scheme depicted in Fig. III.2. We fix α_H and use it only as a safety parameter (α_H set to 0.7 in this study). The choice of α_L is less straightforward: α_L is zero when $C(t) < R_H(t)$, α_L equals α_H when $C(t) > R_H(t) + R_L(t)$ and it changes linearly within between these two end regimes. The notation SSFD(x) denotes the SSFD algorithm with $\alpha_H = 0.7$ and α_L set to x.

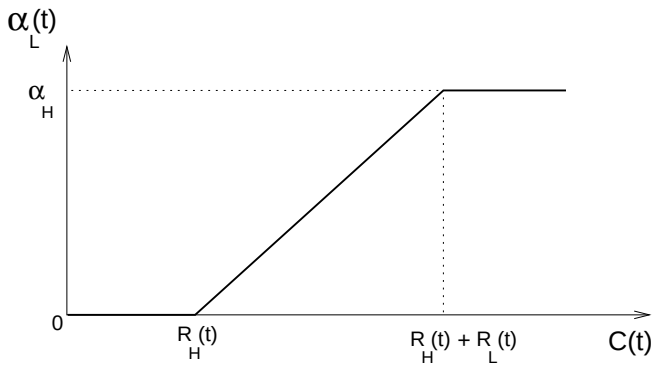

Figure III.2: Adaptive choice of α_L in the ASFD algorithm

III.3 Simulation Results

In this section, we study the performance of the proposed stored video streaming architecture using simulation. We use ns-2 [35] for simulations with a number of enhancements required for the video streaming architecture given in Fig. III.1. We use the single bottleneck topology in Fig. III.3 for all the simulation experiments. In all simulations, N video sessions (of length 780 seconds) share a single bottleneck link with capacity C_{tot} (set to 1 Mbps), where N will be varied to account for the variability of the available bandwidth to each user. The buffer management mechanism for the bottleneck link is assumed to be RED (Random Early Detect). Motivated by [36], we use the RED parameters $(min_{th}, max_{th}, max_p) = (20, 60, 0.1)$ and the RED smoothing parameter set to 0.002 unless otherwise stated.

The first $N/2$ sessions are sinked at $dest_1$ and the remaining ones at $dest_2$. Each video source employs TCP Reno with the same set of parameters and options and each source streams the same video clip. There is one tagged source we monitor among the N sources for PSNR (Peak Signal-Noise Ratio) plots.

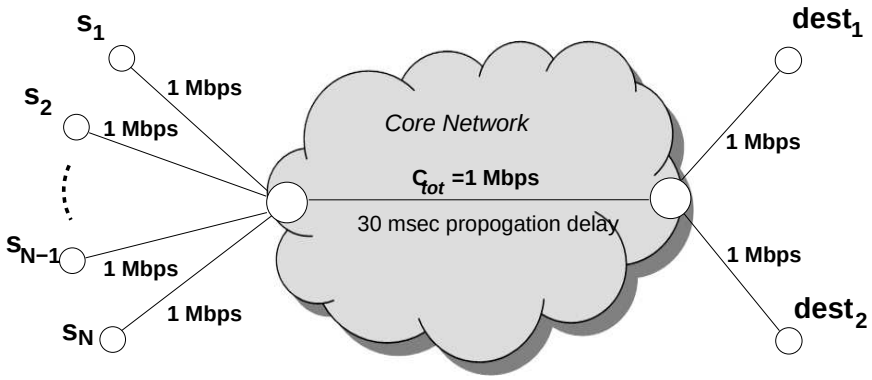

Figure III.3: The network topology used in the simulation studies

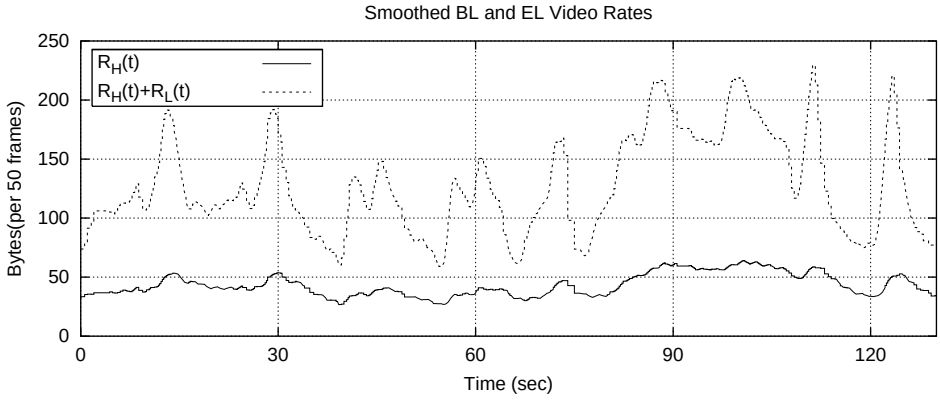

Figure III.4: Smoothed bit rates for the BL and EL for the layered video used in the simulations

Each source starts streaming at random points in the video clip in order to prevent synchronization among the sources. Throughout the simulations, the bit rate of the VBR encoded video has substantial oscillations while the average rates are $R_L \approx 82.6$ kbps and $R_H \approx 35.0$ kbps (see Figure III.4). Given that the original video frequency is $f = 25$ frames/sec, the two layer temporally scalable video which is illustrated in Fig. II.3 is composed of a single I and 9 anchor-P frames as the base layer for each two-seconds interval (i.e., Group of Pictures (GOP) duration). On the other hand, the remaining 40 are plain P frames that constitute the enhancement layer. In our simulations, the average PSNR is used as the performance metric. For lost (either discarded by sender or received after decoding deadline by receiver) frames the concealment is done at the receiver by replicating the most recently decoded frame. Since we are using a temporally scalable bit stream, the PSNR of the received frames reflects the degradation in system performance due to losses only in the BL. Using PSNR for both received and lost frames enables us to see the degradation in the system performance caused by both the L-frame and H-frame losses. In all of our experiments, the bottleneck link with capacity C_{tot} is shared among N sources where $N \in \{6, .., 40\}$ and the expected fair bandwidth share per flow, which is $C \approx C_{tot}/N$, changes in the range $\{25, \ldots, 166\}$ kbps.

In our first experiment, we compare and contrast the performance of the ASFD algorithm with the SSFD algorithm with three settings for $\alpha_L \in \{0.05, 0.4, 0.7\}$. For this purpose, we vary the number of video sessions N and thus change the fair share of each session $C \approx C_{tot}/N$ and obtain the corresponding PSNR value

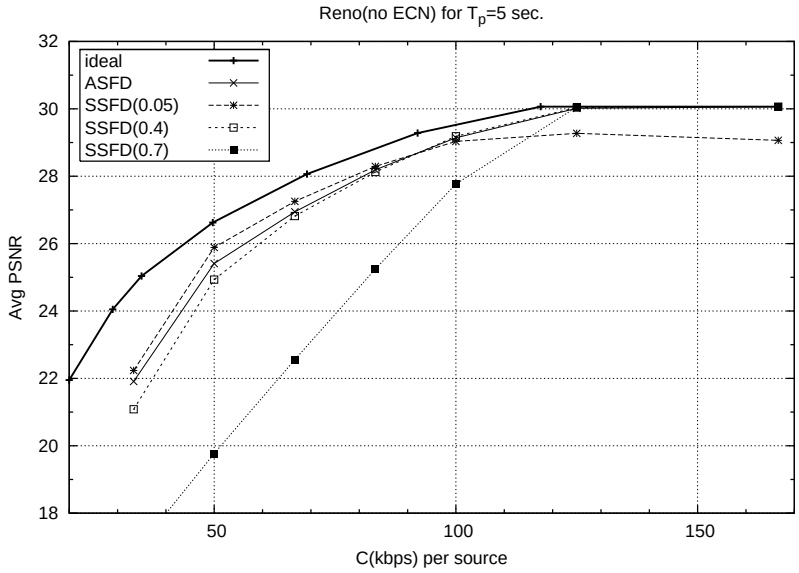

Figure III.5: Comparison of SSFD vs ASFD for the case $T_p = 5$ sec

for the SSFD and ASFD algorithms. The playout delay T_p is set to 5 sec in this study. The results are depicted in Fig. III.5. The ideal curve is obtained by allowing the system to transmit and play all the scheduled frames, in other words for a given bandwidth it is assumed that there is enough playout buffering to tolerate the latency due to retransmissions and the video bit rate is properly matched to the constant available bandwidth in the network so that the scheduled frames never miss their playout times. In our simulations, the EL and/or BL frames are discarded sequentially for the computation of the ideal curve and the corresponding bit rate is calculated. The sequence used for discarding is the same for each GOP. The selection of a conservative SSFD policy (i.e., SSFD(0.05)) gives the best results for the heavy load case (i.e., $C < 100$ kbps) when compared to all other schemes. However, in the light load case when C gets close to or beyond $R_L + R_H$, the PSNR performance of SSFD(0.05) degrades substantially

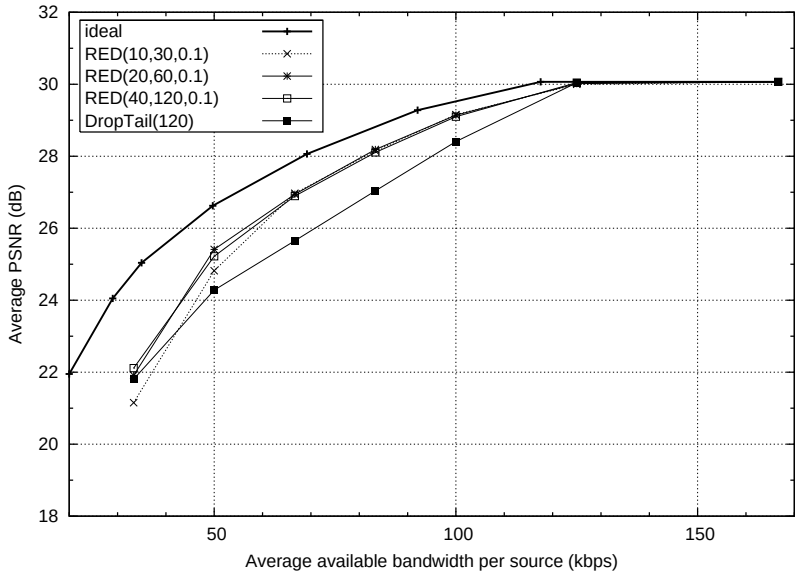

Figure III.6: Effect of RED parameters on ASFD performance with $T_p = 5$ sec.

compared to the less conservative policies SSFD(0.4) and SSFD(0.7). On the other hand, the adaptive version ASFD is robust with respect to the changes in the available bandwidth per user and it compares reasonably well with the best performing static policy in each case. The advantage of the ASFD is that the video server can find a policy very close to the optimal frame discarding policy using local measurements even when the available bandwidth per user changes significantly during the lifetime of the video session. This behavior can definitely not be obtained with static policies.

In our second simulation experiment, we study the impact of the RED parameters on the ASFD performance. The results are given in Fig. III.6. The cases with three different RED configurations outperformed the drop-tail policy with the buffer size set to 120 packets. This observation can be explained by the fact that drop-tail buffer management causes synchronized losses and the resulting

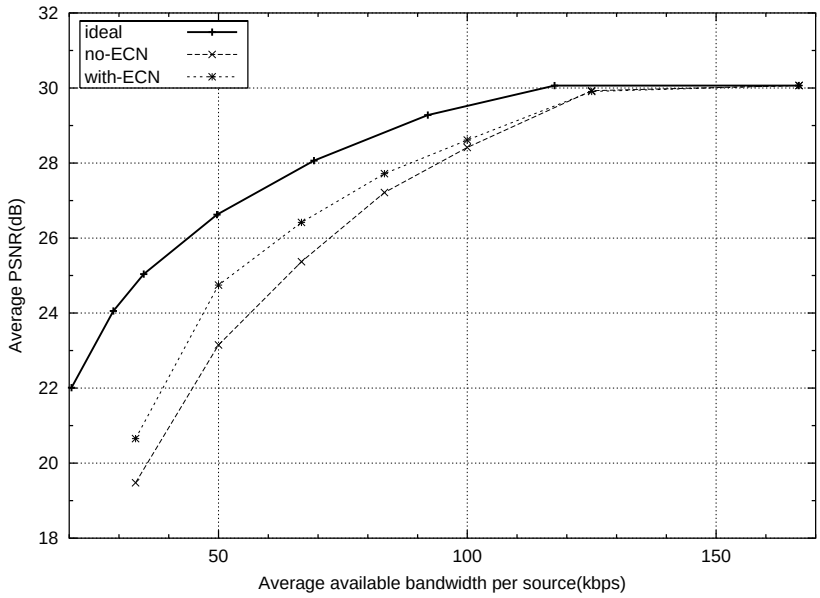

Figure III.7: Impact of ECN on streaming performance for ASFD with $T_p = 2$ sec.

overshoots and undershoots in the resulting buffer occupancy yield substantial performance degradation relative to that of RED. We generally obtained quite robust results with RED but we also observed performance degradation with RED(10,30,0.1) in the heavy load case compared to the other two RED systems. This degradation is due to the relatively conservative choice of min_{th} and max_{th} in this system when a fairly large number of sources are multiplexed.

In the third simulation experiment, we study the impact of using ECN for which the RED module at the bottleneck link marks the packets with the corresponding probabilities as opposed to discarding them. This congestion information is then fed back in the TCP acknowledgments via which the TCP sources adjust their window sizes. Since all TCP senders are using ECN and all respond to congestion before actually loosing a packet they tend to experience less the

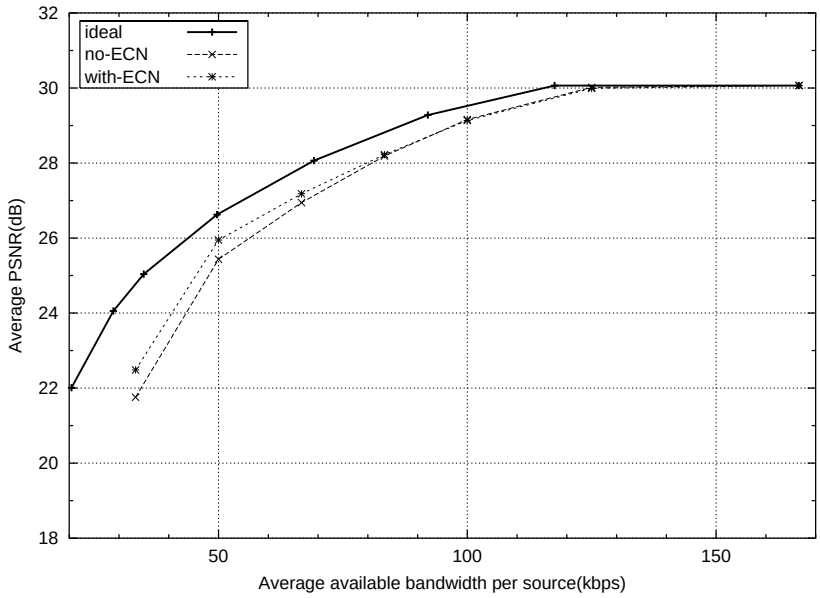

Figure III.8: Effect of ECN on streaming performance for ASFD with $T_p = 5$ sec.

undesired data or timer driven loss recovery phases of TCP. This behavior, as one might expect, leads to a significant performance improvement especially in congested network scenarios and for small initial playout delays. This situation is depicted in Fig. III.7 in which T_p is set to 2 sec. and the performance of using TCP Reno without ECN and TCP Reno with ECN are shown in terms of the average PSNR values for varying C. For the heavy load case, the performance gain with ECN is remarkable (up to 2 db). The $T_p = 5$ sec. case is depicted in Fig. III.8 for which the ECN gains are smaller compared to the $T_p = 2$ sec. case. For small playout delays, it is more likely that a larger percentage of the TCP's retransmissions arrive at the receiver later than their corresponding deadlines. With ECN, losses in the network are reduced and so are retransmissions. This is why the performance gain of ECN is more significant in cases with small

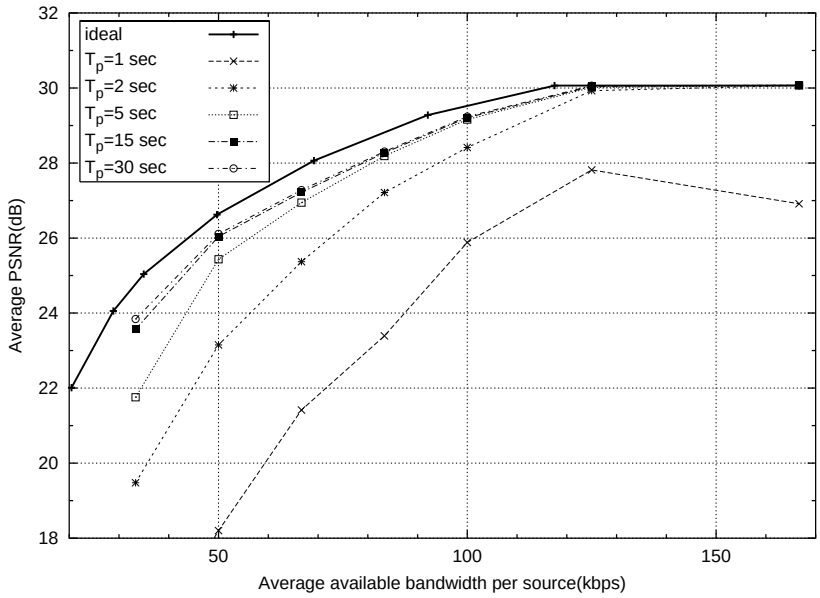

Figure III.9: Impact of T_p on average PSNR for ASFD algorithm

playout delays. As shown in Fig. III.7, $T_p = 2$ sec. of buffering cannot tolerate the timer driven retransmissions occurring in TCP, therefore a significant PSNR degradation is observed if ECN is not employed as compared to the $T_p = 5$ sec. case.

In the fourth experiment, we study the impact of the playout delay T_p which is used in order to compensate for the oscillations in the video bit rate and available network bandwidth per user. The playout delay T_p is varied from 1 sec. to 30 sec. and the corresponding PSNR values are plotted with respect to varying C in Fig. III.9. The PSNR curves saturate at around $T_p = 15$ sec. beyond which buffering only slightly improves the PSNR performance. For small T_p (i.e., $T_p =$ 1 or 2 sec.), the playout delay is comparable to the delays encountered in TCP's data/timer driven retransmissions and a larger percentage of the network losses result in missed playouts and thus reduced PSNRs. With TCP, increasing T_p

from 2 to 5 sec. increases the streaming performance substantially by up to 3 dB.

Up to now, we assumed a best-effort Internet and we proposed intelligent frame scheduling and discarding techniques at the edge (i.e., at the application layer) which operates in harmony with the underlying transport protocol TCP. A network-based alternative for frame discrimination is the IETF (Internet Engineering Task Force) Differentiated Services (Diffserv) architecture [37]. Diffserv defines different service classes for applications with different Quality of Service (QoS) requirements. An end-to-end service differentiation is obtained by concatenation of per-domain services and Service Level Agreements (SLAs) between adjoining domains. Per domain services are realized by traffic conditioning including classification, metering, policing, shaping at the edge and simple differentiated forwarding mechanisms at the core of the network. One of the popular proposed forwarding mechanisms is Assured Forwarding (AF) Per Hop Behavior (PHB) [33]. The AF PHB defines four AF (Assured Forwarding) classes: AF1-4. Each class is assigned a specific amount of buffer space and bandwidth. Within each AF class, one can specify three drop precedence values: 1, 2, and 3. In the notation AFxy, x denotes the AF class number ($x = 1, \ldots, 4$) and y denotes the drop precedence ($y = 1, \ldots, 3$).

In our final simulation experiment, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) Assured Forwarding (AF) Per-Hop-Behavior (PHB) architecture in the context of stored video streaming and identify regimes in which the former

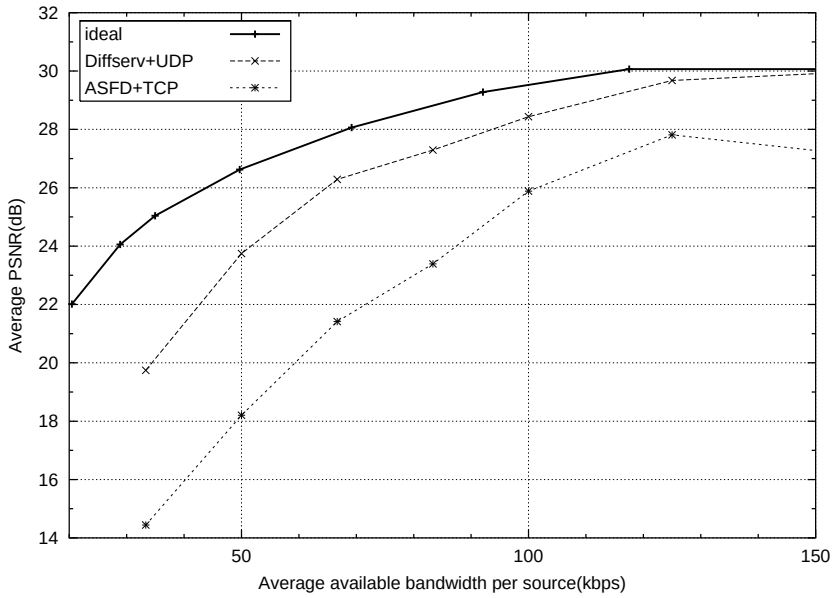

Figure III.10: PSNR plots using Diffserv+UDP and ASFD+TCP scheme for $T_p = 1sec.$ scenario

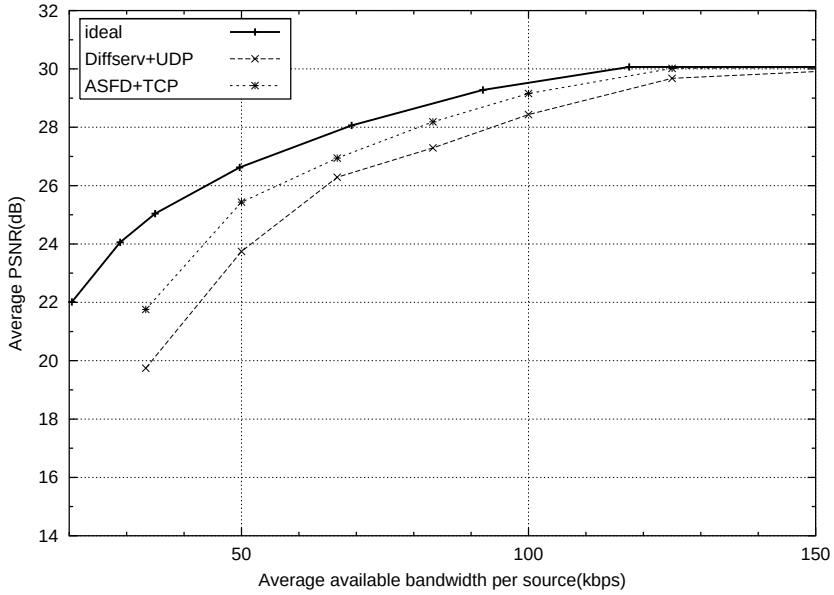

Figure III.11: PSNR plots using Diffserv+UDP and ASFD+TCP scheme for $T_p = 5sec.$ scenario

architecture outperforms the latter. For the Diffserv scenario, we mark packets belonging to H-frames as AF11 and those of L-frames as AF12. We use Weighted RED (WRED) with the RED parameters (20, 60, 0.1) and (10, 30, 0.25) for AF11 and AF12, respectively [38]. We do not impose the use of any traffic conditioner in this experiment but we make use of only the differentiated forwarding paradigm of Diffserv. We use UDP (User Datagram Protocol) for the transport layer for this scenario. We will refer to the combined scheme as Diffserv+UDP. The number of video sources sharing the bottleneck link are varied and PSNR values are plotted in Fig. III.10 for the case $T_p = 1$ sec. which demonstrates that when the client playout delay T_p is small and comparable to one Round Trip Time (RTT), the Diffserv+UDP solution outperforms the proposed ASFD+TCP approach. However, when T_p is increased to 5 sec., then the ASFD+TCP solution gives better results than that of the Diffserv+UDP solution (see Figure III.11). The reason for this behavior is that when the client playout delay is large enough then the TCP sender can retransmit not acknowledged packets without them missing their deadlines (as opposed to the $T_p = 1$ sec. case). Moreover, it is the application layer that intelligently decides on which frames to discard in ASFD+TCP by taking into consideration their playout deadlines. We're led to believe that when the playout delays are sufficiently large (i.e., $T_p > 5$ sec.) then the proposed edge-based adaptive approach is superior to the network-based Diffserv+UDP scheme which is static in its parameter settings and which is not aware of the playout deadlines.

The simulation results presented in this chapter are obtained for TCP traffic

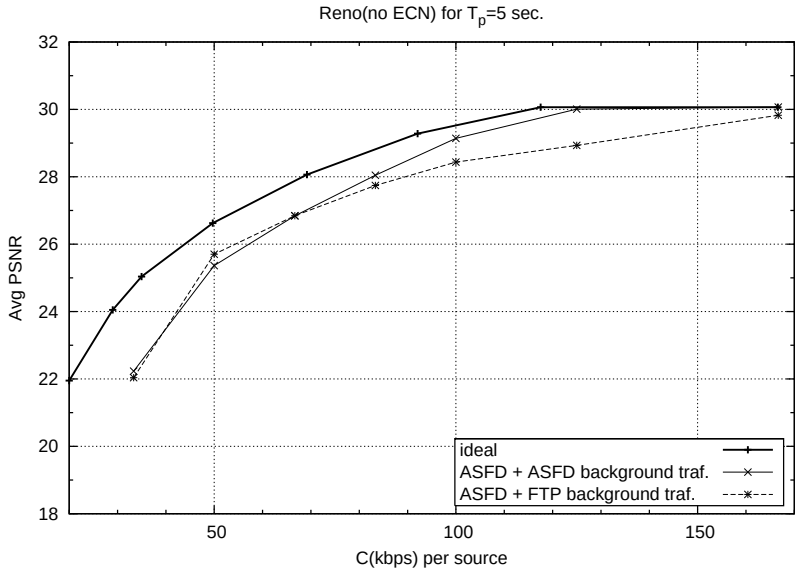

Figure III.12: Comparison on the performance of ASFD with $T_p = 5$ sec while using all ASFD and all FTP sources as the background traffic

generated by N video sources using ASFD (i.e. ASFD+TCP), which means that background traffic is ASFD+TCP. This situation may yield misleading results since video sources are self regulatory and prevent the TCP rate to exceed the maximum rate of that video. In real life the TCP sources contending for the bandwidth would not be all video, rather they would mostly behave like an infinite FTP source. For this reason we designed a new experiment in order to observe the performance degradation in ASFD+TCP when it is contending with infinite FTP+TCP sources. As expected, the simulation results in Figure III.12 shows that at higher rates which are limited by video's maximum bit rate, share of the ASFD+TCP share will decrease.

III.4 Conclusions

Motivated by the extensive operation experience behind TCP, we propose in this chapter an easily implementable stored video streaming system using TCP transport. The proposed system consists of an input buffer implemented at the application layer of the server coupled with the congestion control scheme of TCP at the transport layer. The proposed frame discarding strategy dynamically and intelligently discards low priority frames from its head-end. Moreover, it is adaptive to changes in the bandwidth available to the video stream. Our simulation results demonstrate that scalable stored video can efficiently be streamed over TCP with the proposed adaptive frame discarding strategy if the client playout delay is large enough to absorb the fluctuations in the TCP estimation of the available bandwidth. As expected, the use of Explicit Congestion Notification (ECN) in the network is shown to slightly improve the throughput especially in congested network scenarios and for small initial playout delays. Finally, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) AF PHB architecture and identify regimes in which the former architecture outperforms the latter. We show through a number of simulations that if the playout delay is sufficiently long (i.e., $T_p > 5$ sec.) then the proposed edge-based solution outperforms the core-based Diffserv solution whereas this relationship is reversed otherwise.

However using TCP transport in streaming has some advantages due to its ubiquity, inherent congestion control algorithm, transparency to firewalls, ability to use transport layer signals (i.e. ECN) like other ECN capable protocols

and almost identical behaviour with TCP flows behaving as infinite sources, its disadvantages such as variation in throughput and delay, head of line blocking at the receiver and loss intolerant behavior, have catastrophic effects on a streaming system performance. Our results reveal that it is not possible to use TCP in a streaming system, unless large playout buffers are allowed in order to tolerate these adverse effects. In the rest of this work we replaced TCP transport and two layer temporal scalability in order to minimize initial playout delay which both have adverse effects on streaming performance, by TCP-friendly rate control (TFRC) and fine granular scalability (FGS) schemes respectively.

CHAPTER IV

RETRANSMISSION BASED ANALYTICAL RATE AND QUALITY CONTROLLER

In this chapter we propose a non-interactive video streaming application which relies on application layer retransmissions initiated by timer driven ARQs. However the proposed RARQ (Retransmission Based Analytical Rate and Quality) controller suffers from its limited application domain due to its lack of adaptiveness and limited distortion control capability. Therefore the RARQ controller should be interpreted as a transitional work, on the other hand the analysis used in this section and the ideas from the RaDiO framework constitute the basis of the OSRC (Optimal Scheduling and Rate Control) framework described in the next chapter.

In RARQ, the startup latency T_p is fixed and rate information is assumed to be provided by an underlying congestion control scheme such as TCP-friendly rate control (TFRC) [5]. The streaming system gets C bytes/sec from the scalable bit stream and divides it into fixed number of layers. At each transmission opportunity a decision is made only for the new frame by the RARQ controller

which specifies the number of layers that will be transmitted. However due to lossy behavior of the communication channel, some layers (packets) of previously transmitted frames are lost and immediately retransmitted by timer driven automatic repeat requests (ARQs), where the timeout values of these retransmission timers are known as retransmission timeout (RTO) and calculated from channel statistics. The channel is modeled as a shifted Gamma distribution as in [50],[61] and this model is later used for calculating parameters of the the optimization framework described in Section IV.1. The proposed streaming scheme is modeled as a Markov decision process (MDP) and solved by using linear programming (LP) formulation. Solution of this MDP model yields the optimal policy which maximizes an objective function (i.e. minimize expected distortion) while conforming to rate and quality constraints which respectively correspond to the estimate of available network bandwidth and expected PSNR change between consecutive frames. Proposed algorithm is compared to the results of RaDiO (Rate Distortion Optimized) packet transmission framework [50]. Provided that acceptable amount of startup latency is allowed, our algorithm yields nearly the same results with RaDiO for error rates < 0.2 that is typical in the current Internet. Furthermore as opposed to the RaDiO framework, our algorithm is capable of reducing the quality fluctuations between consecutive frames which may be very disturbing for the viewer.

In Section IV.1 we presented the MDP model of RARQ controller which regulates the delivery of rate adaptable video under specified rate and quality constraints. Simulation results are given in section IV.2 and concluded in

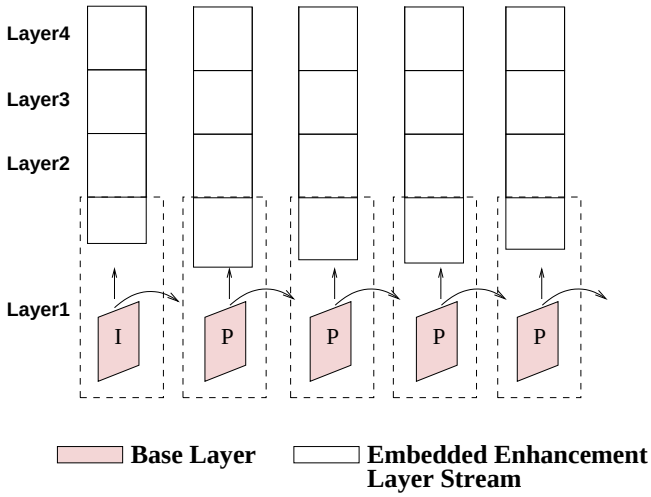

Figure IV.1: Rate adaptable, FGS video coding and packetization

section IV.3.

IV.1 Retransmission Based Analytical Rate Quality Controller

In proposed video streaming framework, bit stream is assumed to be encoded in a rate adaptable manner using H.264 [58] and MPEG-4/FGS [14] as base layer (BL) and enhancement layer (EL) streams respectively. Packetization is then achieved by dividing the overall data of each frame (BL+EL) into equal size packets as given in Figure IV.1 and can be perfectly adapted to the estimated available bandwidth C. Despite the fact that video rate is adapted to channel bandwidth, a rate increase will occur due to retransmission of packets that are lost in the channel. In the retransmission based streaming system, once a layer of a frame is allowed to the system, it is retransmitted until it's decoding deadline. The problem is finding the optimal policy that minimizes total distortion by means of maximizing the number of transmitted layers under given channel

bandwidth and quality fluctuation constraints. Once the system is modeled as a Markovian decision process (MDP) where the state information is specified by the number of outstanding packets (layers) in the system, it can be solved by using linear programming under given probabilistic constraints.

The underlying Markov chain is given as $\mathbf{X} = \{\mathbf{X_n} : \mathbf{n} \in \mathcal{Z}^+, \mathbf{X_n} \in \mathcal{I}\}$ where $\mathcal{I}$ represents the set of all possible states. X_n denotes the system state at the nth decision making epoch which corresponds to generation time of nth frame which is an integer multiple of frame inter arrival time, ΔT. At each decision making epoch n, the system observes state X_n where $X_n = i \in \mathcal{I}$, takes action $A_n = a \in \mathcal{A}$, observes new state X_{n+1} where $X_{n+1} = j \in \mathcal{I}$ with probability $P_{ij}(a)$. For a state action pair $(X_n = i, A_n = a)$ reward function, $r(i,a) = r(X_n, A_n)$ is defined to be the reduction in distortion introduced by the layers of frame n received before their decoding deadline. In addition to the reward function we define two cost functions *rate* and *quality fluctuation*, which will specify the constraints of our optimization framework. First we will verbally define rate and quality fluctuation functions here and their formulation will be respectively presented later in this section in Eqn. IV.13 and Eqn. IV.11 after introducing the state variables and calculating state transition probabilities. Rate constraint puts a bound (N packets) on the long run average of rate function $c(i,a)$ in Eqn. IV.1. More specifically $c(i,a)$ represents the expected number of transmitted/retransmitted packets during state transition $i \rightarrow j$. On the other hand, the constraint on the quality fluctuation function $v(i,a)$ specifies the variation in the number of scheduled layers for consecutive frames, and

this constraint puts a bound σ on the long run average of quality fluctuation function $v(i,a)$ in Eqn. IV.1. The quality fluctuation function is expected to be a function of previous and current actions,A_{n-1},A_n and the current state X_n. However in the MDP model, states are defined such that previous action is included as a state variable of X_n, so explicit relation to A_{n-1} may be removed in the notation such that $v(i,a) = v(X_n, A_n) = v(X_n, A_n, A_{n-1})$ can be used interchangeably. Expected quality fluctuation, rate and reward functions will be introduced in detail later in this section in Eqn. IV.11,IV.13,IV.17. Hence, maximization of long run average reward under given constraints will result in a policy π^* which will be optimal in terms of minimizing distortion under rate and quality fluctuation constraints as given in Eqn. IV.1.

$$\lim_{m\to\infty} \frac{1}{m} E_\pi\{\sum_{n=1}^{m} r(X_n, A_n)\}$$

$$s.t. \begin{cases} \lim_{m\to\infty} \frac{1}{m} E_\pi\{\sum_{n=1}^{m} c(X_n, A_n)\} \leq N \\ \lim_{m\to\infty} \frac{1}{m} E_\pi\{\sum_{n=1}^{m} v(X_n, A_n)\} \leq \sigma \end{cases} \tag{IV.1}$$

The maximization problem in Eqn. IV.1 can be solved by means of linear programming as formulated in Eqn. IV.2 where details are given in [59]. In the linear programming formulation x_{ia}'s are variables of the linear programming formulation and can be interpreted as the steady state frequency of visiting that state action pair (i,a). The imposed constraints on rate and quality fluctuation are represented by N and σ respectively, and since these constraints are imposed on state visiting frequencies, x_{ia}, they are called *probabilistic constraints*. Under

probabilistic constraints, a policy π is called to be a *randomized* policy and described by a probability distribution $\{\pi_a(i), a \in A(i)\}$ for each state $i \in \mathcal{I}$. For some policy π action $a \in A(i)$ is chosen with probability $\pi_a(i)$ whenever the process is in state i.

Step 1. Using simplex algorithm compute optimal basic solution x^*_{ia} to the linear program

$$\max \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} r(i,a)x_{ia}$$

$$s.t. \begin{cases} \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} c(i,a)x_{ia} \leq N \\ \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} v(i,a)x_{ia} \leq \sigma \\ \sum_{a\in\mathcal{A}(j)} x_{ja} - \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} P_{ij}(a)x_{ia} = 0, j \in \mathcal{I} \\ \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} x_{ia} = 1 \\ x_{ia} \geq 0, \text{ for all } i, a \end{cases} \tag{IV.2}$$

Step 2. Start with nonempty set $I_0 = \left\{ i \,\middle|\, \sum_{a\in A(i)} x^*_{ia} > 0 \right\}$ and for any state $i \in I_0$, set the decision

$$\pi_a(i)^* = \begin{cases} x^*_{ia} / \sum_{a\in A(i)} x^*_{ia}, & a \in A(i) \text{ and } i \in I_0 \\ \text{arbitrary}, & \text{otherwise} \end{cases} \tag{IV.3}$$

In our retransmission based MDP framework, frames become available for transmission at integer multiples of frame inter arrival time ΔT and upon their availability a layers of that frame are immediately transmitted according to some optimal policy determined by the state of the controlled Markov chain. System state $i = X_n$ is represented by the transmission history $(s^i_1, f^i_1, f^i_2, ..., f^i_M)$ where

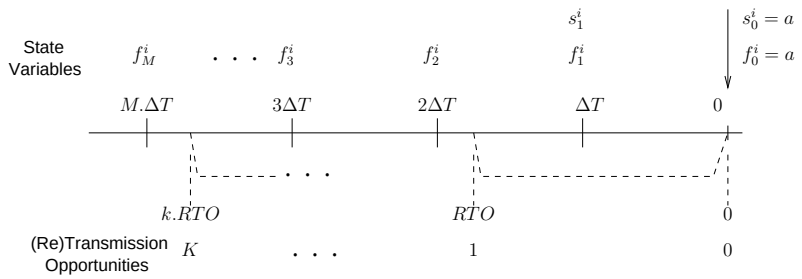

Figure IV.2: State variables $(s^i_1, f^i_1, f^i_2, ..., f^i_M)$ of state-i and transmission opportunities

s^i_1 is the number of layers sent in the previous action and f_m with $m, \in \{1, ..., M\}$ corresponds to the on-the-fly layers of m previous frame in the transmission history. System state is shown in Figure IV.2 for system with transmission history of M frames where this number is determined by the initial playout buffer time, T_p and ΔT, such that $M = \lfloor T_p/\Delta T \rfloor$. Furthermore relation between frame rate, F, and frame inter arrival time is given by $\Delta T = 1/F$. On the other hand initial transmission and subsequent retransmissions of a frame occur at a time scale independent of ΔT and is determined by the retransmission timeout (RTO) value chosen according to the channel statistics. Each frame will obtain transmission opportunities $k = 0, 1, ..., K$ where 0 is the initial transmission and remaining K are the retransmissions of that frame as illustrated in Figure IV.2. Total number of transmission opportunities of a frame including the very first transmission is simply $K+1$ where K is a function of T_p and RTO and calculated as $K = \lfloor T_p/RTO \rfloor$.

Given that f^i_m for $m \in \{1, ..., M\}$ to be the state variables of system state i, for the completeness of the notation below we should define $f^i_0 = a$ to be the action taken at state i. In state transition $i \rightarrow j$ probability of transiting from

f_m^i to f_{m+1}^j for m^{th} frame in the transmission history of $m \in \{0, .., M-1\}$ while considering a set of $0, .., k$ transmissions of that frame is defined as $P_{ij}(m,k)$ in Eqn. IV.4 and can be calculated as in Eqn. IV.5.

$$P_{ij}(m,k) = P\{f_{m+1}^j \text{ layers of } m^{th} \text{ frame not received considering tx. } \leq k \mid f_m^i \text{ layers not rcvd.}\} \tag{IV.4}$$

$$P_{ij}(m,k) = \begin{cases} \sum_{l=f_{m+1}^j}^{f_m^i} P'_{ij}(m,k,l), & m\Delta T \leq kRTO < (m+1)\Delta T \\ \binom{f_m^i}{f_{m+1}^j} \epsilon(m,k)^{f_{m+1}^j} (1-\epsilon(m,k))^{f_m^i - f_{m+1}^j}, & k.RTO < m\Delta T \\ \text{not defined }, otherwise \end{cases} \tag{IV.5}$$

IV.1.1 Case $m\Delta T \leq kRTO \leq (m+1)\Delta T$

This condition in Eqn. IV.5 is the case that k^{th} transmission opportunity occur during the transition of the m^{th} frame from f_m^i to f_{m+1}^j as illustrated in Figure IV.3. $P'(m,k,l)$ is defined for the m^{th} frame in transmission history as the probability of starting with f_m^i at current state i and having f_{m+1}^j at next state j while passing the condition of retransmitting l layers at the k-th transmission opportunity where l can definitely take values $f_{m+1}^j \leq l \leq f_m^i$ (Eqn. IV.6).

$$P'_{ij}(m,k,l) = \binom{f_m^i}{l} \epsilon_p(m,k)^l (1-\epsilon_p(m,k))^{f_m^i - l} \times \binom{l}{f_{m+1}^j} \epsilon_c(m,k)^{f_{m+1}^j} (1-\epsilon_c(m,k))^{l - f_{m+1}^j} \tag{IV.6}$$

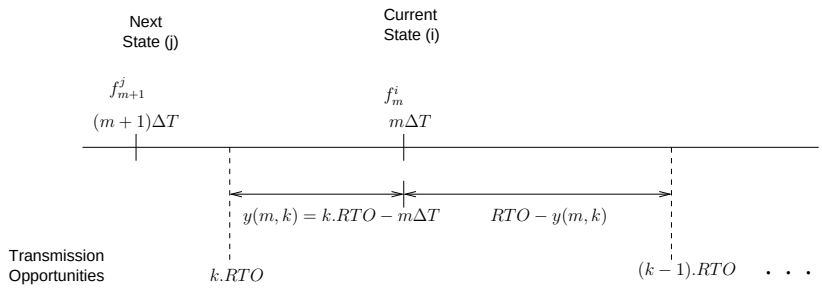

Figure IV.3: Case $m\Delta T \leq kRTO \leq (m+1)\Delta T$: State transition for m previous frame

In order to calculate $P'_{ij}(m,k,l)$ for $m\Delta T \leq kRTO \leq (m+1)\Delta T$, we have to consider the k^{th} transmission opportunity that occurs in between the transition f^i_m and f^j_{m+1} as given in Figure IV.3. Hence we define $\epsilon_p(m,k)$, $\epsilon_c(m,k)$ as error probabilities of a layer from m^{th} frame which are respectively calculated by considering transmissions *prior* to k or considering prior transmissions by including k as the *current* transmission. In Figure IV.3, $y(m,k) \triangleq k.RTO - m\Delta T$ is defined in order to simplify the notation in Eqn. IV.7 and IV.8. In Eqn. IV.7 $\epsilon_p(m,k)$ is defined to be the probability of not receiving a layer from the m^{th} frame *prior* to $k.RTO$ while considering $0,...,k-1$ transmissions given that the layer of frame m has not been received before $m\Delta T$. Similarly in Eqn. IV.8 $\epsilon_c(m,k)$ is defined to be the probability of not receiving a layer until $(m+1)\Delta T$ at the next state considering all transmissions $0,..,k$ of that frame including the *current* (i.e. k^{th}) given that the layer is not received until $k.RTO$.

$$\epsilon_p(m,k) = \prod_{r=0}^{k} P\{RTT > (k-r)RTO \mid RTT > (k-r)RTO - y(m,k)\} \quad \text{(IV.7)}$$

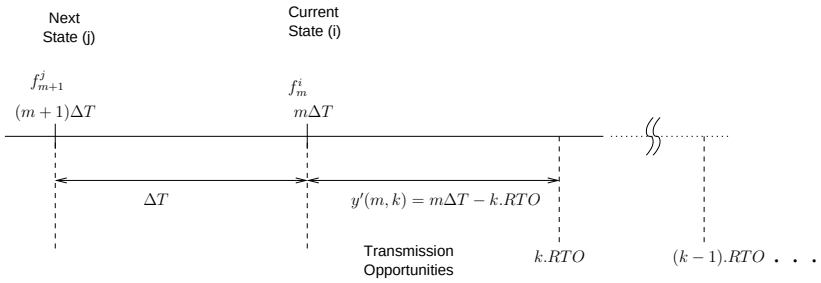

Figure IV.4: Case $kRTO \leq m\Delta T$: State transition for m^{th} frame

$$\epsilon_c(m,k) = \prod_{r=0}^{k} P\{RTT > (k-r)RTO + (\Delta T - y(m,k)) \mid RTT > (k-r)RTO\} \quad \text{(IV.8)}$$

IV.1.2 Case $kRTO \leq m\Delta T$

In this case no retransmission opportunity exist while transiting from f^i_m to f^j_{m+1} for the m^{th} frame. Hence the latest retransmission k occurs sometime before $m\Delta T$ such that $k.RTO < m\Delta T$. This case is illustrated in Figure IV.4 where $y'(m,k) = m\Delta T - k.RTO$ is defined in order to write error probability, $\epsilon(m,k)$ for the m^{th} frame. In Eqn. IV.9 $\epsilon(m,k)$ is defined to be the probability of not receiving a layer from the m^{th} frame until $(m+1)\Delta T$ while considering all transmissions before and equal to k given that it is not received until $m\Delta T$.

$$\epsilon(m,k) = \prod_{r=0}^{k} P\{RTT > (k-r)RTO + y'(m,k) + \Delta T \mid RTT > (k-r)RTO + y'(m,k)\} \quad \text{(IV.9)}$$

Analysis of regions given in Eqn. IV.5 is finished in previous sections, and we are now equipped with the expression $P_{ij}(m,k)$. Consider the m^{th} frame at state

i, total number of transmissions (opportunities) of a layer from that frame until it becomes the $m+1^{st}$ frame at next state j, is defined to be $g(m)+1$ and can be calculated as $g(m) = \left\lfloor \frac{(m+1)\Delta T}{RTO} \right\rfloor$. After setting $k = g(m)$ in $P(m,k)$, total state transition probability $P_{ij}(a)$ under action a can be calculated as the product of transition probabilities for all frames $m = 0, 1, .., M-1$ in transmission history as given in Eqn IV.10 (including the action for the current frame in $f_0^i = a$).

$$P_{ij}(a) = \prod_{l=0}^{M-1} P_{ij}(l, g(l)), \text{ where } f_0^i = a \tag{IV.10}$$

After calculating the state transition probabilities, we can focus on the quality fluctuation $v(i,a)$ and rate $c(i,a)$ functions. Definition of quality fluctuation is simpler and given as the difference between scheduled layers of consecutive frames as in Eqn. IV.11

$$v(i,a) = |s_1^i - a| \tag{IV.11}$$

On the other hand in order to calculate cost (packets transmitted), $E_{ij}\{n\}$, the expected number of packets transmitted during $i \to j$ should be evaluated. Using expected cost in Eqn. IV.12 during state transition and the current action total cost is found as in Eqn. IV.13

$$E_{ij}\{n\} = \sum_{\substack{m \in g^{-1}(k) \\ \forall k \in \{1,\ldots,K\}}} \sum_{l=f_{m+1}^j}^{f_m^i} \frac{P'(m, g(m), l)}{P(m, g(m))} \times l \tag{IV.12}$$

$$c(i,a) = a + \sum_{j \in I} P_{ij}(a) E_{ij}\{n\} \tag{IV.13}$$

The last step is to calculate the reward obtained at state i, $r(i,a)$. For this purpose we first write the success probability $P_s(k)$ of a layer from a frame. Success probability of a packet is not a function of the optimal policy that we are seeking for, and it can be interpreted as the channel's correction capability by means of retransmissions. $P_s(k), k \in 0, ..., K-1$ is defined to be the probability of successfully receiving a packet (layer) before its deadline at the client and its ACK at the server while consider all transmission prior to and including the transmission k. Note that for the last transmission opportunity we only include the probability of receiving the data packet at the client. The successfully reception probability of a packet considering all K transmissions is denoted as $P_s(K)$ and can be calculated as given in Eqn IV.14 by using the error term considering K transmission in Eqn. IV.9.

$$P_s(K) = 1 - P\{K\text{th tx. occurs and packet is lost}\} \tag{IV.14}$$

If we define $P(k)$ to be the probability of making kth (re)transmission, and $P\{\text{packet is lost} \mid K\text{th tx. occurs}\}$ to be the probability of not receiving that packet after Kth retransmission until it's decoding deadline, following equation for $P\{\text{packet is lost} \mid K\text{th tx. occurs}\}$ is written from Bayes rule.

$$P\{K\text{th tx. occurs and packet is lost}\} =$$
$$P(k).P\{\text{packet is lost} \mid K\text{th tx. occurs}\}$$

The terms $P(k)$ and $P\{\text{packet is lost} \mid K\text{th tx. occurs}\}$ is defined in Eqn. IV.15

and Eqn. IV.16 respectively. Note that in Eqn. IV.15 obviously $P(0) = 1$, hence when a packet is determined for transmission it is surely transmitted at the first transmission opportunity. On the other hand in Eqn. IV.16 it is such defined that $\Delta \stackrel{\triangle}{=} T_p - K.RTO$.

$$P(k)=\prod_{k'=0}^{k-1}\prod_{l=0}^{k'} P\{RTT > (k'-l+1)RTO \mid RTT > (k'-l).RTO\}), \text{ for } k \in 1,..K \quad \text{(IV.15)}$$

$$P\{\text{packet is lost} \mid K\text{th tx. occurs}\} = \prod_{l=0}^{K} P\{FTT > \Delta + (K-l)RTO | RTT > \Delta + (K-l).RTO\}) \quad \text{(IV.16)}$$

Assuming that system is in state $X_n = i$, reduction in distortion $r(i, a)$ in state $i = X_n$ for frame n under action a is simply calculated by using distortion $D_i(l)$ for lth layer of that frame. Reward $r(i, a)$ obtained at state i (for frame n) is denoted by the reduction in distortion as given in Eqn. IV.17.

$$r(i,a) = \sum_{l=1}^{a} D_i(l).(P_s(K))^l \quad \text{(IV.17)}$$

IV.2 Simulation Results

In order to evaluate our proposed algorithm and compare it with other streaming schemes, we first implemented a simple channel simulator which applies error and delay according to the prespecified probabilistic channel model. Throughout the simulations we used a Gamma distributed channel delay model as given

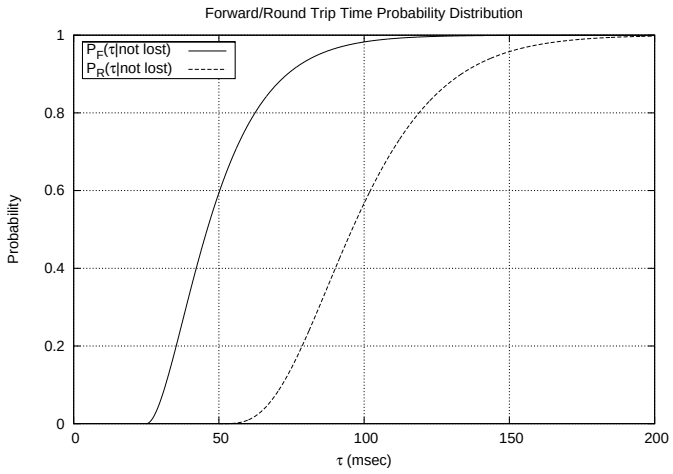

Figure IV.5: Delay distribution of the forward (F) and round trip (R) channels

in Figure IV.5 and assumed symmetric error probabilities in both forward and backward channels. On top of the channel simulator we used an FGS coded video where base layer is coded by a standard H.264 compressor where the the residual error is encoded by using MPEG-4/FGS encoder into a fine granular scalable embedded bit stream. The rate adaptable bit stream is generated by encoding a sequence of 1088 frames at 10 fps obtained from FOREMAN, CARPHONE, LION, COASTGUARD AND TENNIS test sequences which corresponds to a video of 109 seconds.

In all of our experiment setups we fixed the startup playout buffering time to be $T_p = 500$ msec. and used the channel delay model given in Figure IV.5. We set the retransmission timer expiry to be $RTO = 180$ msec which causes only a minimal number of faulty retransmissions. A transmission is named as faulty when an acknowledgment (ACK) packet transmitted by the receiver for a successful data packet is not received by the sender within a period of RTO due

to either loss or lateness of that ACK packet. Selecting a larger RTO will further reduce the number of faulty retransmissions whereas such an increase will reduce the number of retransmission opportunities within a given T_p. In this chapter we do not propose any explicit method for the selection of RTO value, however when limitations on startup latency in media streaming are considered it may not be preferable to select safe values as TCP does for the estimation of RTO, such that $RTO = \mu_{RTT} + 4.\sigma_{RTT}$.

In the following experiments we evaluated the performance of RARQ for a wide range of channel error values and compared it with the results of streaming schemes that use RaDiO scheduling and Reed-Solomon FEC codes of $RS(n, k)$. FEC code rates are assumed to be static throughout a session in our simulations. Figure IV.6 shows the video performance observed at the decoder for FEC, RaDiO and RARQ algorithms. Provided that sufficient startup latency is allowed retransmission based schemes demonstrate their superiority as compared to FEC based schemes since the latter approach suffers from a rate-distortion penalty caused by the coding redundancy. On the other hand when retransmission based schemes are compared, RaDiO and RARQ perform very similarly at low error rates whereas RaDiO works best at all error rates as shown in Fig. IV.6. However when a typical error range (< 0.2) of a properly managed IP network is considered, RaDiO offers only a marginal improvement or not at all. Furthermore, since RARQ algorithm takes quality fluctuation as a constraint of the problem, resulting policy will bound the quality fluctuation in received video to the specified level. Fig. IV.7 shows that RARQ algorithm may lower

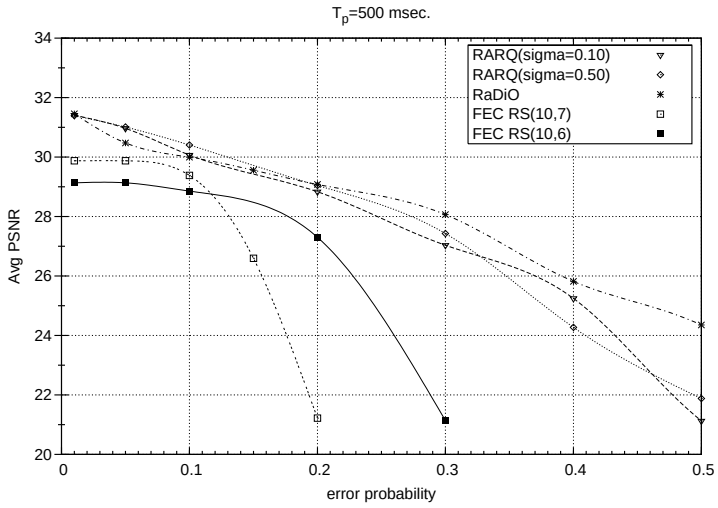

Figure IV.6: Expected PSNR for RaDiO, static FEC and our framework

the expected PSNR difference between consecutive frames nearly up to 1.5 dB as compared to RaDiO.

IV.3 Conclusion

In this chapter we proposed a retransmission based analytical rate and quality (RARQ) controller for non-interactive video streaming applications. The timer based ARQ streaming system is modeled as a Markov decision process (MDP) and solved by Linear Programming (LP) in order to obtain the optimal policy. The optimal RARQ policy set , π^*, decides the amount of layers to transmit that will minimize the total distortion while matching the network *rate* and *quality fluctuation* constraints. The proposed RARQ framework yields nearly identical results with RaDiO at low error rates that is typical in the Internet, whereas the proposed algorithm may reduce the quality fluctuation up to 1.5 dB as compared to RaDiO. However due to RARQ's lack of adaptiveness and limited distortion

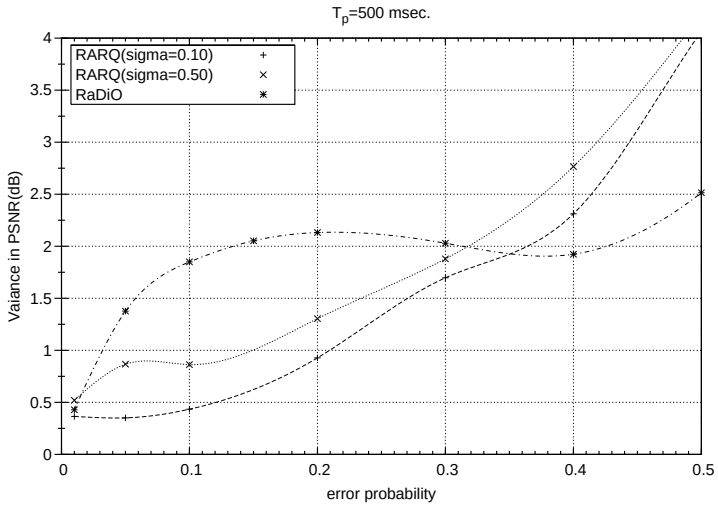

Figure IV.7: Expected inter frame PSNR variance at decoder

control capability its application domain would be very limited, for this reason our efforts on RARQ is limited to that chapter.

CHAPTER V

AVERAGE REWARD OPTIMAL PACKET SCHEDULING AND RATE CONTROL

In this chapter, we propose a new retransmission based optimal video streaming framework. This framework minimizes the average distortion of the observed video at the receiver by joint use of the proposed optimal packet scheduling (OS) and rate control (RC) algorithms which is named OSRC. The OSRC framework is capable of making intelligent and adaptive retransmissions, such that in case of an increase in channel error or delay it is capable to control the distortion at an acceptable level. This is achieved by switching to the optimal policy schedule $\pi^*(\lambda)$ and rate constraint λ^* for new channel conditions, which increases the number of transmissions and reduces the video bit rate in order to obey the rate constraint λ^*.

The OSRC framework is composed of two algorithms namely OS and RC. In the OS part by using an MDP analysis optimal schedule (policy) $\pi^*(\lambda)$ that maximizes the success probability of packets and conform to rate constraint λ are calculated such that the resulting optimal policy will yield the optimal

success probability $P^*_{suc}(\lambda)$. This step is done a priori in an off-line manner for different values of channel error, round trip time and rate constraint λ and the resulting optimal policies and success probabilities for $\lambda \in \Lambda$ are stored for later use of the RC algorithm. On the other hand, RC is an online algorithm and corresponds to the optimal rate control scheme which calculates the optimal λ^* by using the optimal success probabilities $P^*_{suc}(\lambda)$, the distortion model of the video, channel statistics and the packetization information of a frame where a frame can be fragmented into N_p network packets.

Some major features of the OSRC algorithm are listed as follows:

- Low streaming complexity due to direct use of optimal policy $\pi^*(\lambda^*)$ selected from a set of a priori calculated optimal policies. However it requires more space to store off-line calculated policies.

- Increased granularity for redundancy selection such that $\lambda^* \in \Lambda$.

- Capable of handling fragmentation of large sized video frames into small sized network packets imposed by the MTU (Maximum Transmission Unit) size.

- No extra quality fluctuation is introduced by the distortion control and rate adaptation algorithms of the OSRC.

- Utilizes a more realistic distortion model which also takes the effect of the simple error concealment scheme into account. However an analytical model for the calculation of video distortion is required.

The details of OSRC framework are given in In Section V.2. However prior to Section V.2 channel model is introduced in Section V.1 which is necessary to calculate state transition probabilities. Simulation results obtained with both our analytical simulator and ns-2 [35] are provided in Section V.3 and finally concluded in Section IV.3.

V.1 Channel Model

In this work we used the same channel model in [50] which models the network as an independent time-invariant packet erasure channel that introduce random delays. The network is assumed to be composed of forward and backward channel pairs such that data packets are sent on the forward channel whereas acknowledgment (ACK) packets received from the backward channel. Given that a data or an ACK packet is not lost, probability density functions of the random variables forward (F) and backward (B) delay are denoted by $p_F(\tau|\text{not lost})$ and $p_B(\tau|\text{not lost})$ respectively. Round trip delay (R) is also a random variable and is given by $R = F + B$, where its density function is given as the convolution of densities of F and B under independence assumption, such that $p_R(\tau|\text{not lost}) = p_F(\tau|\text{not lost}) * p_B(\tau|\text{not lost})$. On the other hand the overall error rate of the round trip channel is found to be as $\epsilon_R = \epsilon_F + (1 - \epsilon_F)\epsilon_B$. Combining packet loss probability and packet delay density into a single probability measure simplifies the calculations [50]. Channel errors are introduced into delay density function as an impulse at ∞ as in Figure V.1 which results in scaled down versions of densities such that $p_F(\tau) = (1 - \epsilon_F).p_F(\tau|\text{not lost})$ and

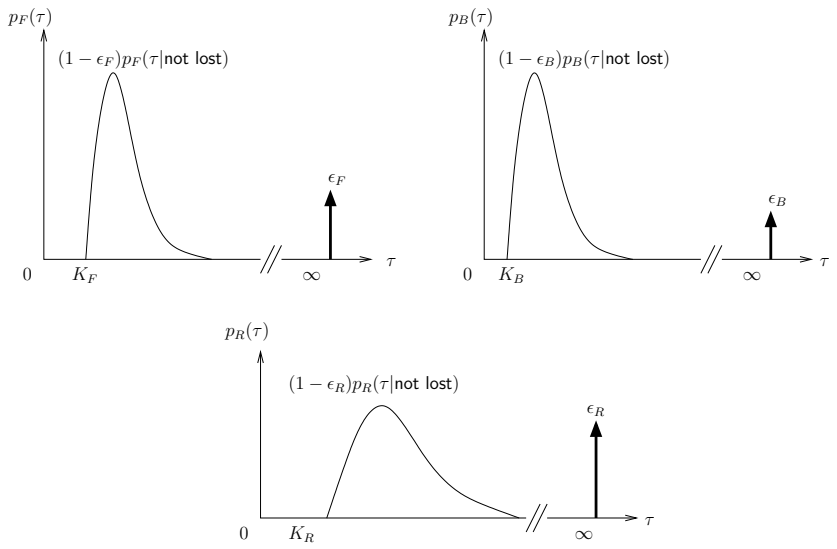

Figure V.1: Illustration of the forward (F), backward (B) and round trip (R) channel models

$p_R(\tau) = (1-\epsilon_R).p_R(\tau|\text{not lost})$.

The above channel model do not assume particular form for the densities $p_F(\tau|\text{not lost})$, $p_B(\tau|\text{not lost})$ and $p_R(\tau|\text{not lost})$ However for the sake of concreteness throughout the simulations we used a shifted Gamma distributed channel delay model as in [50],[61]. Gamma distribution is capable of modeling the channel delay of a path with n routers where delay at each router i is a random variable with exponential distribution having the same α parameter, such that $f(\tau) = \alpha e^{-\alpha\tau}, \tau \geq 0$. The probability density function of the total queueing delay in n routers is calculated by the n-fold convolution of $f(\tau)$ which is simply the Gamma distribution in Eqn V.1 for $\tau \geq 0$ where the gamma function is defined to be as $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ for $x \in \mathbf{R}^+$.

$$p(\tau|\text{not lost}) = \frac{\alpha}{\Gamma(n)}(\alpha\tau)^{n-1}e^{-\alpha\tau} \tag{V.1}$$

κ_F is a constant value and represents the end-to-end propagation delay in

the forward path. Note that Eqn V.2 is κ_F units shifted version of Eqn V.1 after replacing $\alpha = \alpha_F$, $n = n_F$ and defined for $\tau \geq \kappa_F$. The resulting channel model is capable of modeling both propagation and queueing delay.

$$p_F(\tau|\text{not lost}) = \frac{\alpha_F}{\Gamma(n_F)}(\alpha_F(\tau - \kappa_F))^{n_F-1}e^{-\alpha_F(\tau-\kappa_F)} \tag{V.2}$$

By knowing the mean μ_F and the variance σ_F^2 of the forward delay F which is modeled as a gamma distribution in Eqn V.2, we can calculate the α_F and n_F parameters by assuming an M/M/1 queueing at each node. M/M/1 queues introduce mean waiting time $1/\alpha_F$ with a variance of $1/\alpha_F^2$. So the mean and variance of the overall forward delay over n_F routers are $\mu_F = \kappa_F + n_F/\alpha_F$ and $\sigma_F^2 = n_F/\alpha_F^2$ respectively. Hence by simple manipulations α_F and n_F are found as in Eqn V.3 .

$$\begin{aligned} n_F &= (\mu_F - \kappa_F)\alpha_F \\ \alpha_F &= (\mu_F - \kappa_F)/\sigma_F^2 \end{aligned} \tag{V.3}$$

V.2 System Architecture

In the following Sections V.2.1-V.2.2 the theoretical framework used for the calculation of the optimal packet schedules (OS) and optimal rate control (RC) parameter is presented. The main goal of the OSRC algorithm is to calculate and apply the optimal transmission policy $\pi^*(\lambda^*)$ which minimize the average distortion while regulating the video bit rate B^* by using the rate constraint

$\lambda^* = C/B^* \geq 1$. Hence when the policy $\pi^*(\lambda^*)$ is used the remaining bandwidth $C - B^*$ is then utilized by packet retransmissions in order to control the distortion in case of packet losses.

In this framework we assume that the initial playout buffer time T_p and video frame rate $1/\Delta T$ is fixed. The product of these two entities yields the number of frames M' in the transmission window, however we use only $M \leq M'$ of them in MDP analysis in order to work with tractable state spaces. We also assume a close interaction with the transport layer such that the estimate of bandwidth C, packet loss rate ϵ_F, ϵ_B and round trip time RTT are made available by transport layer algorithms which is the case in TFRC (TCP Friendly Rate Control) [5]. Furthermore we also assume that video bit rate is adaptable to the rate B^* dictated by the OSRC algorithm by means of either encoding it in real time or consuming from a rate adaptable encoded bit stream. We obtained our results by using a rate adaptable (scalable) bit stream in order save time by removing the need for re-encoding the bit stream for each simulation.

Before continuing with the theoretical details of the OSRC, we give an illustration of a typical OSRC based video streaming server in Figure V.2. The OS algorithm captures channel information including C, ϵ_F, ϵ_B and μ_R from TFRC and retrieves the list of $P^*_{suc}(\lambda), \lambda \in \Lambda$ immediately. As soon as it receives success probabilities, it passes this information together with available bandwidth estimate C to the RC module in order to obtain the optimal rate control parameter λ^*. Using $B^* = C/\lambda^*$ and MTU size N_p is recalculated and the RC algorithm iterates until the convergence of λ^*. After the completion of this loop,

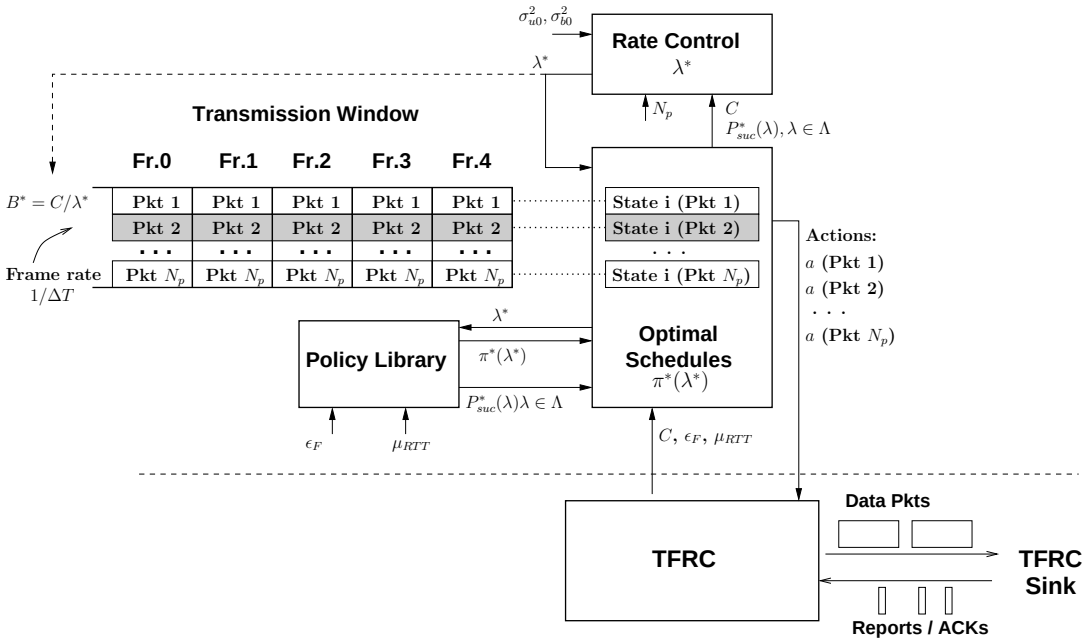

Figure V.2: Illustration of a typical video streaming server using OSRC

the resulting optimal RC parameter λ^* will be used both for calculating the bit rate $B^* = C/\lambda^*$ of newly generated frames and for the selection of new optimal transmission policy $\pi^*(\lambda^*)$ from the *Policy Library* in Figure V.2.

The OSRC framework handles fragmentation of video frames into packets while transmitting them over the packet network. Calculation of the optimal packet schedules (OS) in Section V.2.1 are independent of whether the transmitted units are either frames or packets and this feature enables the OSRC to handle the frame fragmentation scenario easily. However for the sake of generality we will assume in the rest of this chapter that units transmitted will be packets rather then frames. Then the success probabilities $P^*_{suc}(\lambda)$ calculated by the OS part corresponds to the success probability of packets as illustrated in Figure V.2, and the optimal rate control (RC) algorithm in Section V.2.2 will convert it to frame loss rate in order to use its analytical distortion model. In

the RC algorithm we would expect that the fragments of a frame should be fully received by the receiver's transport layer in order to provide the integrity of the transmitted data.

V.2.1 Optimal Packet Scheduling (OS) Problem, $\pi^*(\lambda)$

The problem of finding average reward optimal packet schedules (OS) is defined as the calculation of an optimal policy $\pi^*(\lambda)$ for a given channel statistics that maximizes the average successful on time delivery probability $P_{suc}(\lambda)$ of transmitted packets while conforming to the rate constraint $\lambda \in \Lambda$. The optimal policy is calculated by first developing an MDP model which captures the retransmissions in a video streaming system, and then solving it by means of constrained optimization techniques which maximize a reward function while conforming to a constraint. In our model we provide a constraint set Λ which is obtained by quantizing a feasible range of rate constraint values for example $[1, 2]$ and the corresponding set of optimal policies $\pi^*(\lambda), \lambda \in \Lambda$ and success probabilities $P^*_{suc}(\lambda)$ are calculated off-line and stored for later use of the optimal rate control (RC) algorithm in Section V.2.2 which will later calculate optimal rate control (constraint) parameter λ^*.

Before giving the details of the proposed MDP model and its optimal solution, for the sake of generality we address the fragmentation of a frame into MTUs where the MTU size is determined by the network which is generally the case in a typical packet network. Each frame is assumed to be fragmented into N_p packets where this number is determined by the optimal video rate B^* pro-

vided by the RC algorithm and the MTU size. However since proposed MDP model is independent from the inter-frame relations, we can separately apply this model and its solution to these fragments (packets) originating from M different frames in the transmission window. For example in Figure V.2 darker packets show Pkt2 (i.e. fragment 2) from $M = 5$ different frames for which the optimal schedule $\pi^*(\lambda)$ will be calculated. Hence the MDP model can be applied to any of these fragments separately, in the rest of this section in order to keep the notation simple we will develop the MDP model by only considering the single fragment of M different frames.

In our MDP model, we denote the underlying Markov chain as $\mathbf{X} = \{X_n : n \in \mathcal{Z}^+, X_n \in \mathcal{I}\}$. At each decision making epoch n, the system observes state X_n where $X_n = i \in \mathcal{I}$, takes action $A_n = a \in \mathcal{A}$, and observes new state $X_{n+1} = j \in \mathcal{I}$ with probability $P_{ij}(a)$. In the following notation, system state $i = X_n$ is assumed to represent the current state of the transmission window corresponding to a specific fragment similar to the illustration of Pkt2 in Figure V.2. For a specific fragment the transmission window is composed of M packets from M most recent consecutive frames. Hence in our formulation i denotes the overall system state of the transmission window corresponding to that specific fragment and is given by the combination of the independent state variables $s_k^{(i)}$ where $k \in \{0, .., M-1\}$ such that $i = (s_0^{(i)}, s_1^{(i)}, ..., s_{M-1}^{(i)})$. The state variable $s_k^{(i)}$ represents the transmission and the observation history of the k previous packet in the transmission window as illustrated in Figure V.3. The labels 1 and 0 on the arrows respectively indicate that whether packet is

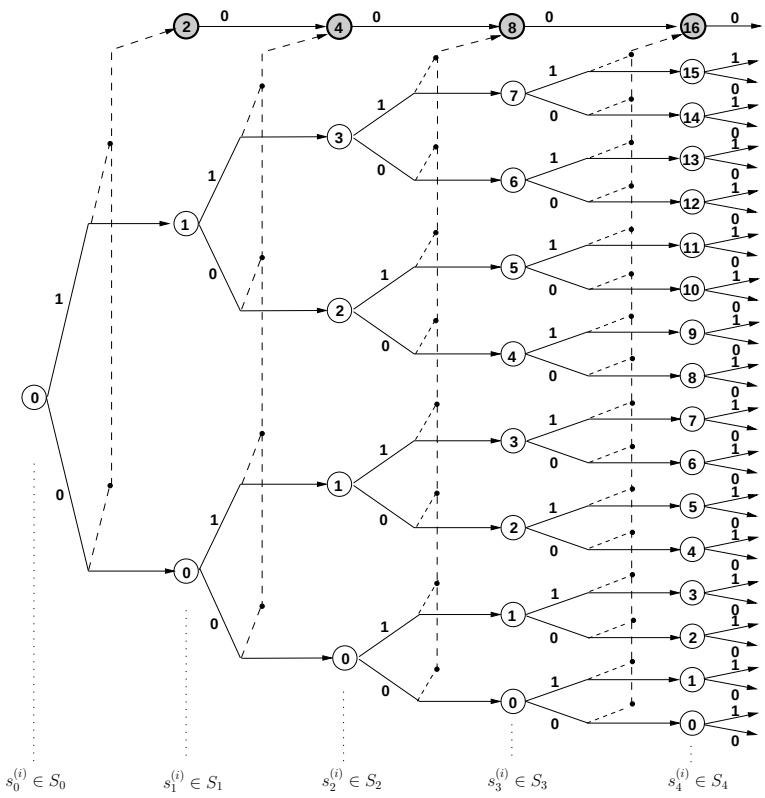

Figure V.3: State variables $(s_0^{(i)}, s_1^{(i)}, ..., s_{M-1}^{(i)})$ for $M = 5$

transmitted or not at that transmission opportunity, and by following back to the initial state on the left of the trellis the transmission history of a packet can be pointed out. On the other hand, hollow and darker circles represent non-final and final states respectively where a final state means an acknowledgement is already received for that packet. Each state variable $s_k^{(i)}$ of state i can be selected from a state space S_k as illustrated in Figure V.3 for $M = 5$ and namely $S_0 = \{0\}$, $S_1 = \{0, 1, 2\}$, $S_2 = \{0, .., 4\}$, $S_3 = \{0, .., 8\}$, and $S_4 = \{0, .., 16\}$. Hence the overall state space can be defined as $i \in \mathcal{I} = S_0 \times S_1 \times ...S_{M-1}$. On the other hand the action taken at state i is denoted as $a = [a_0, ..., a_{M-1}]$ where $a_k \in \{0, 1\}$ is the action for the packet of the kth frame in transmission window and as for labels of the arrows 1 indicate a packet transmission and 0 otherwise.

Given a state action pair (i, a), the reward function $r(i, a)$ is defined as the probability of successfully receiving the packet belonging to the oldest frame (i.e. the $M-1$ th) in the transmission window until its decoding deadline as given by Eqn. V.4.

$$r(i,a) = 1 - \epsilon_{s_{M-1}^{(i)}}(a, M-1), \tag{V.4}$$

Note that $\epsilon_{s_k^{(i)}}(a,k)$ is calculated as in Eqn. V.5 and represents the error probability of a packet from the kth frame of the transmission window given that its state is i and the current action is a.

$$\epsilon_{s_k^{(i)}}(a,k) = \begin{cases} \prod_{\substack{m=0, \\ q_{s_k^{(i)}}(a,m)=1}}^{k} P\{FTT > T_p - m\Delta T | RTT > (k-m)\Delta T\} \\ \qquad\qquad , \ s_k^{(i)} \text{ non-final} \\ 0 \qquad\qquad , \ s_k^{(i)} \text{ final} \end{cases} \tag{V.5}$$

The state transition probability $P_{ij}(a) = \prod_{k=0}^{M-1} P_{ij}(a,k)$ is the product of per packet state transition probabilities $P_{ij}(a,k)$ as in Eqn. V.6

$$P_{ij}(a,k) = \begin{cases} 1 & , \ s_k^{(i)} \text{ final} \\ 1 - p_{s_k^{(i)}}(a,k), & s_k^{(i)} \text{ non-final}, \ s_{k+1}^{(j)} \text{ final} \\ p_{s_k^{(i)}}(a,k) & , \ s_k^{(i)} \text{ non-final}, \ s_{k+1}^{(j)} \text{ non-final} \\ 0 & , \ s_{k+1}^{(j)} \text{ not reachable from } s_k^{(i)} \end{cases} \tag{V.6}$$

where $p_{s_k^{(i)}}(a,k)$ is the probability of not reaching a final state for any of the packets from frame k during state transition under action a, provided that $s_k^{(i)}$ is not a final state and given by

$$p_{s_k^{(i)}}(a,k) \triangleq \prod_{\substack{m=0,\\ q_{s_k^{(i)}}(a,m)=1}}^{k} P\{RTT > (k-m+1)\Delta T | RTT > (k-m)\Delta T\} \tag{V.7}$$

In Eqn. V.7, $q_{s_k^{(i)}}(a,m)$ is a variable that merges the transmission (action) history $h_{s_k^{(i)}}(m)$ and current action a as

$$[q_{s_k^{(i)}}(a,0)\ldots q_{s_k^{(i)}}(a,k)] \triangleq \begin{cases} [h_{s_k^{(i)}}(0)\ldots h_{s_k^{(i)}}(k-1), a_k], s_k^{(i)} \text{ is non-final} \\ \text{not defined} \qquad\qquad\qquad\qquad , \text{otherwise} \end{cases} \tag{V.8}$$

The cost function is defined to be the number of transmitted/retransmitted packets at each transmission opportunity during a state transition under action a and is denoted by $c(i,a) = \sum_{k=0}^{M-1} a_k$. In our constrained optimization framework, the rate constraint $\lambda = C/B$ puts a bound on the long run average value of the cost function as expressed in Eqn. V.9.

Provided that the channel statistics, reward function, initial playout buffer time and rate constraint λ are given, the long run average reward function that is subject to the rate constraint λ is written as in Eqn. V.9. Hence the maximization of the average reward subject to the constraint λ will result in policy $\pi^*(\lambda)$

that is optimal in the average reward sense and this policy maximizes the average successful on time delivery probability $P^*_{suc}(\lambda)$ of packets while conforming to the rate constraint.

$$\lim_{m\to\infty} \frac{1}{m} E_\pi\{\sum_{n=1}^{m} r(X_n, A_n)\}$$
$$s.t. \quad \lim_{m\to\infty} \tfrac{1}{m} E_\pi\{\textstyle\sum_{n=1}^{m} c(X_n, A_n)\} \leq \lambda \tag{V.9}$$

The average reward optimal policy $\pi^*(\lambda)$ to the Eqn. V.9 is calculated by using a linear programming (LP) solution presented in equations V.10 and V.11 where further details can be found in [59]. In these equations $\pi^*_a(i)$ denotes the probability of selecting action $a \in A(i)$ in state i and x_{ia}'s represent the steady state frequency (probability) of visiting that state action pair (i, a). Note that the optimal value of the objective function in Eqn. V.10 corresponds to the average successful on time delivery probability of a packet $P^*_{suc}(\lambda) = \sum_{i\in\mathcal{I}} \sum_{a\in A(i)} x^*_{ia} r(i, a)$.

Step 1. Calculate optimal basic solution x^*_{ia} using simplex algorithm

$$\max \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} r(i, a) x_{ia}$$
$$s.t. \begin{cases} \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} c(i, a) x_{ia} \leq \lambda \\ \sum_{a\in\mathcal{A}(j)} x_{ja} - \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} P_{ij}(a) x_{ia} = 0, j \in \mathcal{I} \\ \sum_{i\in\mathcal{I}} \sum_{a\in\mathcal{A}(i)} x_{ia} = 1 \\ x_{ia} \geq 0, \text{ for all } i, a \end{cases} \tag{V.10}$$

Step 2. For any state $i \in I_0 = \left\{ i \,\middle|\, \sum_{a \in A(i)} x^*_{ia} > 0 \right\}$ set decision

$$\pi_a(i)^* = \begin{cases} x^*_{ia} / \sum_{a' \in A(i)} x^*_{ia'}, & a \in A(i) \text{ and } i \in I_0 \\ \text{arbitrary}, & \text{otherwise} \end{cases} \tag{V.11}$$

V.2.2 Optimal Rate Control (RC) Problem, λ^*

Optimal rate control problem is denoted as the selection of the optimal rate constraint parameter $\lambda^* \in \Lambda$ which minimizes the average video distortion by using a priori calculated optimal policies $\pi^*(\lambda)$ for $\lambda \in \Lambda$ and corresponding average packet success rates $P^*_{suc}(\lambda)$ in Section V.2.1. Furthermore according to the assumption on fragmentation of frames into N_p network packets the effective frame success rate should be defined by using the packet success rate as follows, $P^*_{fsuc}(\lambda) = P^*_{suc}(\lambda)^{N_p}$ and used in the distortion calculations conducted in this chapter. This relation between frame and packet success rate is acceptable since the transport layer discards the partially received transport layer packets due to fragmentation in order to preserve its data integrity. However more advanced distortion calculations that remove this constraint can also be performed. In the rest of this section we will use the video distortion model for lossy packet networks in [49] which utilizes the average frame success rate $P^*_{fsuc}(\lambda)$ that is defined above.

The total distortion D_{tot} for video over lossy channels can be expressed as the sum of distortions D_e and D_v which are introduced by signal compression and packet losses respectively. In this framework, we use a simple error concealment

technique at the decoder which replaces the corrupted frame with the previous one. Hence, in case of losses D_v can be calculated by using the the mean squared error (MSE) $\sigma_{u0}^2(C,\lambda)$ which quantifies the distortion introduced by the first erroneous frame after applying error concealment. In the case of streaming FGS coded video we should elaborate the model in [49] and split this initial distortion term into two parts such as $\sigma_{u0}^2(C,\lambda) = \sigma_{b0}^2 + \sigma_{e0}^2(C,\lambda)$ as illustrated in Figure V.4. The terms σ_{b0}^2 and $\sigma_{u0}^2(C,\lambda)$ can be precalculated for FGS encoded videos where they respectively correspond to the MSE of base layer (BL) and BL+EL when simple error concealment scheme is applied. Hence the remaining term is simply calculated as $\sigma_{e0}^2(C,\lambda) = \sigma_{u0}^2(C,\lambda) - \sigma_{b0}^2$. In case of a packet loss base layer distortion σ_{b0}^2 appears at the base layer as a result of error concealment process. Since base layer is compressed with a predictive encoder, this base layer distortion σ_{b0}^2, propagates to the subsequent frames, whereas the remaining term $\sigma_{e0}^2(C,\lambda)$ that appear due to the concealment of FGS data is constrained only to the lost frame. As a result of this fact the distortion signal due to the loss of Kth frame is represented as $\sigma_v^2[k-K]$ from Eqn V.12 with a slight modification of the original version in [49] by adding an impulse term $\delta[k]$ in order to represent the extra distortion introduced by the concealment of the FGS part of the video.

In [49] given the initial value of concealment distortion as σ_{b0}^2, the power of the propagated error signal in an INTRA refresh period T is denoted by $\sigma_b^2[k] = \sigma_{b0}^2(1 - P_{fsuc}^*(\lambda))\frac{1-\gamma k}{1-\beta k}$ as given by the second term of the summation in Eqn V.12 for $k \in [0, T-1]$ where $1 - P_{fsuc}^*(\lambda)$ is the expected frame loss rate which is provided by optimal packet scheduler for $\lambda \in \Lambda$.

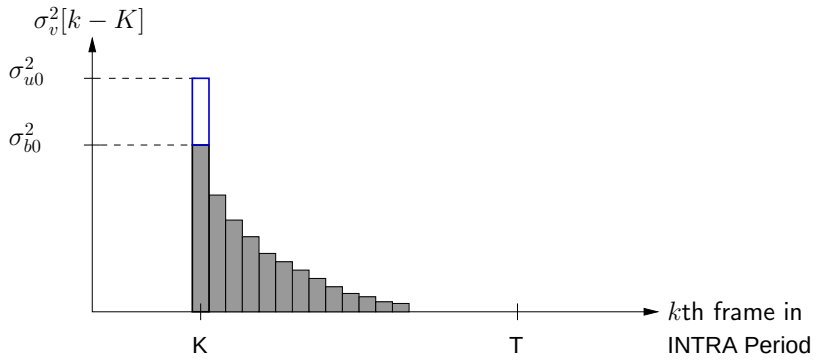

Figure V.4: Illustration of the modified distortion model for FGS video

$$\sigma_v^2[k] = \sigma_{e0}^2(C,\lambda)(1-P_{fsuc}^*(\lambda))\delta[k] + \sigma_{b0}^2(1-P_{fsuc}^*(\lambda))\frac{1-\gamma k}{1-\beta k} \text{ , for } k \in [0, T-1] \tag{V.12}$$

The other parameters $\beta = 1/T$ and γ correspond to the intra frame rate and leakage factor, respectively, where $\beta, \gamma \in [0,1]$. The leakage factor γ describes the efficiency of loop filtering and is not straightforward to obtain. By conducting numerical minimization for D_v and β values that are obtained experimentally, we approximately calculate the leakage factor to be $\gamma = 0.55$. In Eqn V.13, $D_v(C,\lambda)$ can be obtained as the sum of two distortion terms that appear in Eqn V.12 which respectively stands for the distortion due to FGS and base layers streams. Note that the second term, which corresponds to the base layer distortion is obtained by the superposition of T error signals that are shifted in time under the assumption that the decoder is linear and superimposed error signals are uncorrelated.

$$D_v(C,\lambda) = \sigma_{e0}^2(C,\lambda)(1-P_{fsuc}^*(\lambda)) + \sigma_{b0}^2(1-P_{fsuc}^*(\lambda))\sum_{t=0}^{T-1}\frac{1-\gamma t}{1-\beta t} \tag{V.13}$$

On the other hand, the relation between the coding distortion $D_e(C,\lambda)$ and the encoding rate B is trivial and video bit rate is simply represented as $B = C/\lambda$ in the OSRC framework. Knowing these quantities it is possible to define the PSNR (Peak Signal to Noise Ratio) metric which is widely used in measuring the quality of multimedia. It is simply calculated by using the distortion terms $D_{tot}(C,\lambda) = D_e(C,\lambda) + D_v(C,\lambda)$ as in Eqn V.14.

$$PSNR(C,\lambda) = 10log(\frac{255^2}{(D_e(C,\lambda) + D_v(C,\lambda))}) \tag{V.14}$$

Finally, the optimal rate control (RC) is simply achieved by finding the optimal rate constraint λ^* which minimizes the total distortion $D_{tot}(C,\lambda)$ or maximizes the $PSNR(C,\lambda)$ as given in Eqn. V.15. However the number of fragments N_p is depending on the $B^* = C/\lambda^*$, MTU size and frame rate. Since MTU size and frame rate is fixed for a streaming session and C is given, the value of N_p is determined only by the λ^*. Hence the minimization procedure of the RC algorithm in Eqn. V.15 is iterated until the value for λ^* converges.

$$\lambda^* = \arg\min_{\lambda\in\Lambda} D_e(C,\lambda) + D_v(C,\lambda) = \arg\max_{\lambda\in\Lambda} PSNR(C,\lambda) \tag{V.15}$$

V.3 Simulation Results

The simulation results for the OSRC framework is obtained by using both an analytical network simulator and standard ns-2 simulator. The analytical sim-

ulator is capable of applying a channel error and delay from a probabilistic channel model presented in Section V.1. Hence, the analytical simulator provides a controlled environment for making simulations and showing the relations easily. However it doesn't reflect the system dynamics of other flows like TCP and their effects properly as it is in the real world, for this reason in the second part of this section, we presented ns-2 simulator results in order to show the applicability of the proposed algorithm.

The optimal policies for the OS part of the algorithm are calculated for a set of values for parameters mean round trip time $\mu_R \in \{75, 100, 150, 200, 250, 300\}$ and channel error rates of $\epsilon_F = \epsilon_B = \{0.01, 0.05, 0.075, 0.1\}$ where the rate constraint λ is limited to the set of $\Lambda = \{1.00, 1.05, 1.10, 1.15, 1.20, \ldots, 2.00\}$. However we set the variance as in [50] such that $\sigma_R \approx 0.35(\mu_R - \kappa)$ and the resulting set of values would be $\sigma_R = \{8, 17, 34, 52, 70, 88\}$ where propagation delay would be $\kappa = 50$ msec. By selecting σ_R as a function of μ_R and κ, variance would no further be a new dimension in the problem. Hence, knowing the μ_R and σ_R the other parameters n and α of the gamma distribution $p_R(\tau|\text{not lost}) = \frac{\alpha}{\Gamma(n)}(\alpha(\tau - \kappa))^{n-1}e^{-\alpha(\tau-\kappa)}$ in Section V.1 can be calculated as in Eqn V.3.

The OSRC framework is simply a network adaptive video streaming scheme which makes use of video bit rate adaptation, either by means of real-time video coding or using a rate adaptable video like FGS. We conducted our simulations using FGS video due to its practicality however it should be noted that scalable coding schemes introduce a rate-distortion penalty. The rate adaptable video bit stream is generated by using a standard H.264 encoder [58] as for the base layer

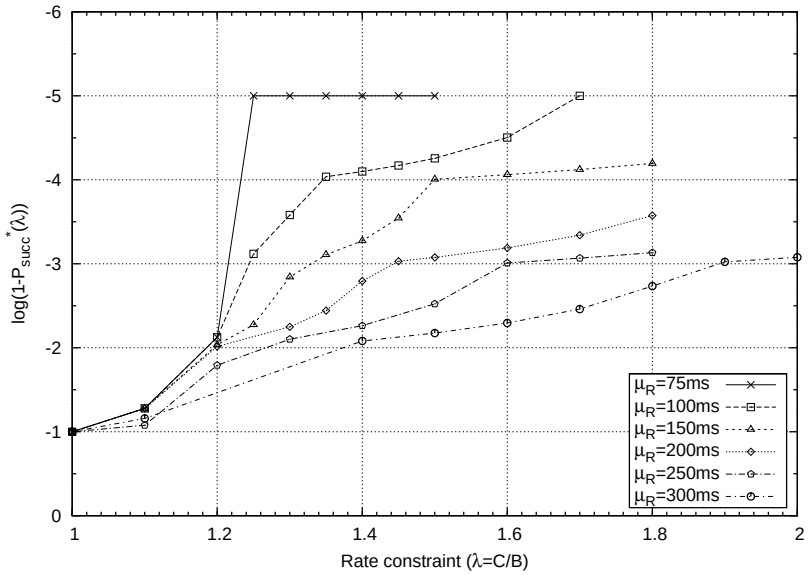

Figure V.5: $log(1 - P^*_{suc}(\lambda))$ for $T_p = 1000$ msec, $\epsilon_F = \epsilon_B = 0.1$

(BL) together with an MPEG-4 FGS encoder. Throughout the experiments we used I-P-P coding for the BL with an INTRA update period of $T = 50$ frames. the resulting video is composed of 1044 frames which are obtained by encoding the FOREMAN, CARPHONE sequences sampled at a frequency of 10 fps for four times. A streaming session using this video is extended to any duration by rewinding and starting from the beginning. In our simulations we conducted our experiments for a duration of 100 sec. which corresponds to a total of 1000 frames at 10 fps. In all of the experiments, the initial playout buffer time is fixed to $T_p = 1000$ msec which results in a number of $M' = 10$ frames in the transmission window where we used only $M = 5$ opportunities for our analysis in order to work with tractable state spaces.

In Figure V.5 we fixed the channel error to $\epsilon_F = \epsilon_B = 0.1$ and give the plot of the logarithm of error in other words the $log(1 - P^*_{suc}(\lambda))$ where $P^*_{suc}(\lambda)$

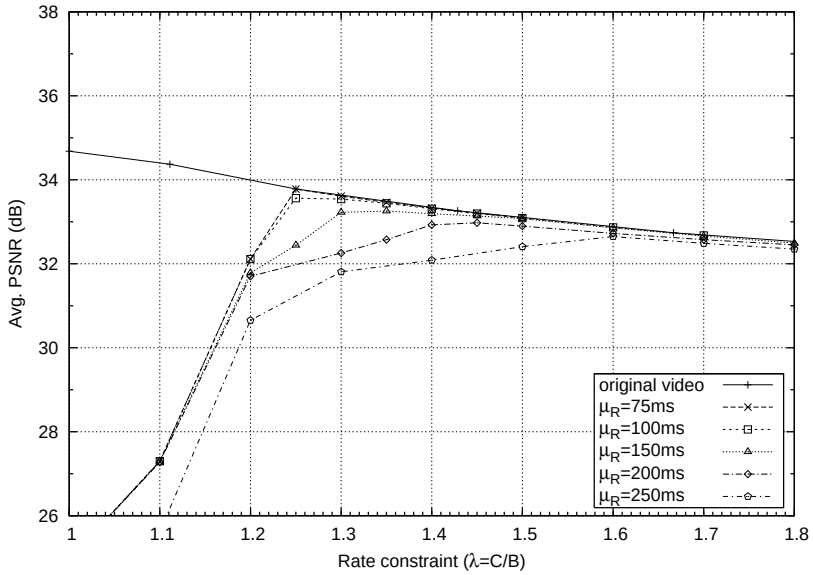

Figure V.6: PSNR for $T_p = 1000$ msec, $\epsilon_F = \epsilon_B = 0.1$ and $C = 20000$ bytes/sec

is the achievable success rate while using optimal policies $\pi^*(\lambda), \lambda \in \Lambda$ under rate constraint λ. Note that this value, N_p and C are provided to the RC module by the OS module in Figure V.2 for the purpose of calculation of the average distortion in order to find the optimal λ^* that minimize this distortion. Distortion is calculated at the RC module by using the information provided by OS module in addition to the N_p, σ_{u0}^2 and σ_{b0}^2 parameters and the corresponding PSNR in Eqn. V.14 is plotted as in Figure V.6 for $C = 20000$ bytes/sec. Finally RC module calculates the optimal rate control (RC) parameter λ^* that minimize the distortion as in Eqn V.15 and return it back to the OS module for its use in order to load and use the policy $\pi^*(\lambda^*)$ during streaming until next update.

Similarly in Figures V.7- V.8 we fixed the average round trip time to $\mu_R = 150$ msec and obtained plots for a set of rate constraints $\lambda \in \Lambda$. In Figure V.7 we present the logarithm of error $log(1 - P_{suc}^*(\lambda))$ when OS algorithm is applied

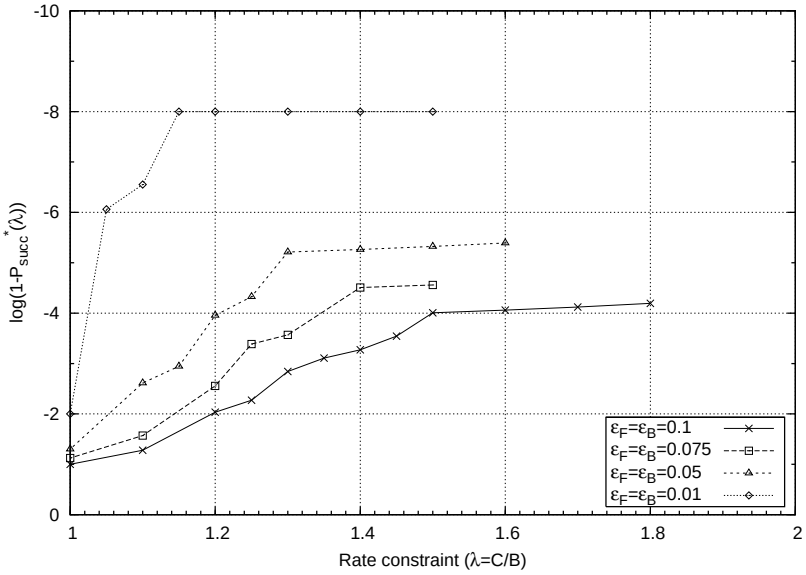

Figure V.7: $log(1 - P^*_{suc}(\lambda))$ for $T_p = 1000$ msec, $\mu_R = 150$ msec

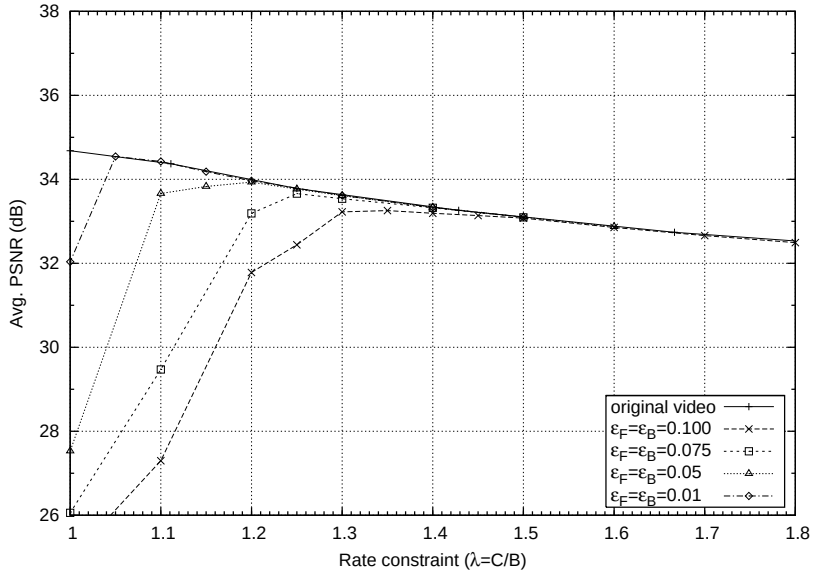

Figure V.8: PSNR for $T_p = 1000$ msec, $\mu_R = 150$ msec and $C = 20000$ bytes/sec

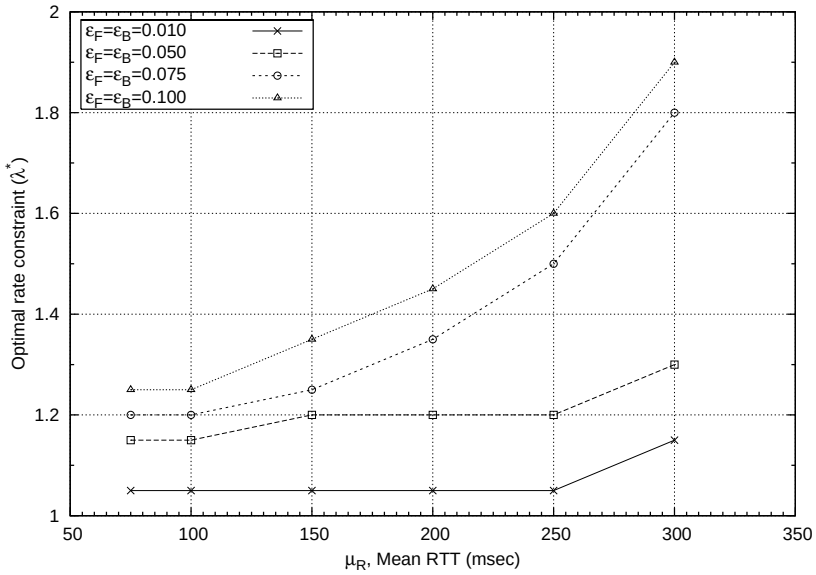

Figure V.9: PSNR for $T_p = 1000$ msec, $\epsilon_F = \epsilon_B = 0.1$ and $C = 20000$ bytes/sec

for $\lambda \in \Lambda$. On the other hand in Figure V.8 we plot the PSNR in Eqn V.14 calculated with the knowledge of $P^*_{suc}(\lambda)$, N_p and C. From the perspective of the RC algorithm by knowing these entities, the optimal rate constraint λ^* that minimize the distortion is calculated as in Eqn V.15.

In Figures V.5- V.8 two sets of numerical results are given where each set gives the plot of residual error probability $log(1 - P^*_{suc}(\lambda))$ and corresponding PSNR value at C. The results are taken for varying $\lambda \in \Lambda$ where channel error rate $\epsilon_F = \epsilon_B = 0.1$ is fixed in the first set and mean RTT $\mu_R = 150$ at the second. However in these plots the OSRC algorithm is interested in the point where $\lambda = \lambda^*$ rather than the whole range. Fig. V.9 illustrates the optimal behaviour of OSRC where λ^* values are calculated by the optimal rate control (RC) algorithm for different mean channel delay μ_R and channel error ϵ_F, ϵ_B values. With increasing channel delay μ_R, the system observability decreases

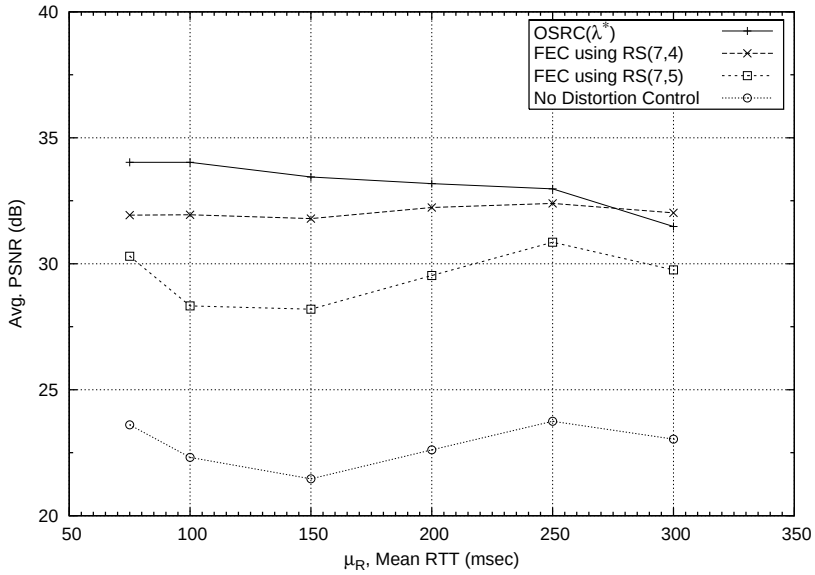

Figure V.10: PSNR for $T_p = 1000$ msec, $\epsilon_F = \epsilon_B = 0.1$ and $C = 20000$ bytes/sec

which results in an increase in λ^* in order to provide distortion control. On the other hand, with increasing channel error rates more retransmissions are needed to provide distortion control which again results in an increase in the value of λ^* as demonstrated in Fig. V.9.

After providing the above figure which aims to describe the behaviour of the OSRC algorithm, we may give some comparisons in order to specify the efficiency of the algorithm. In Fig. V.10 we compare the results of three different approaches for a streaming session of approximately 100 sec. long. First, we apply no distortion control which does not make any retransmissions or inject any redundancy. This case should be interpreted as the limiting case for the distortion while fixing the channel statistics and the available bandwidth. The other two approaches are the static FEC using Reed-Solomon codes as in [51] and the proposed optimal streaming scheme. We first show that these two

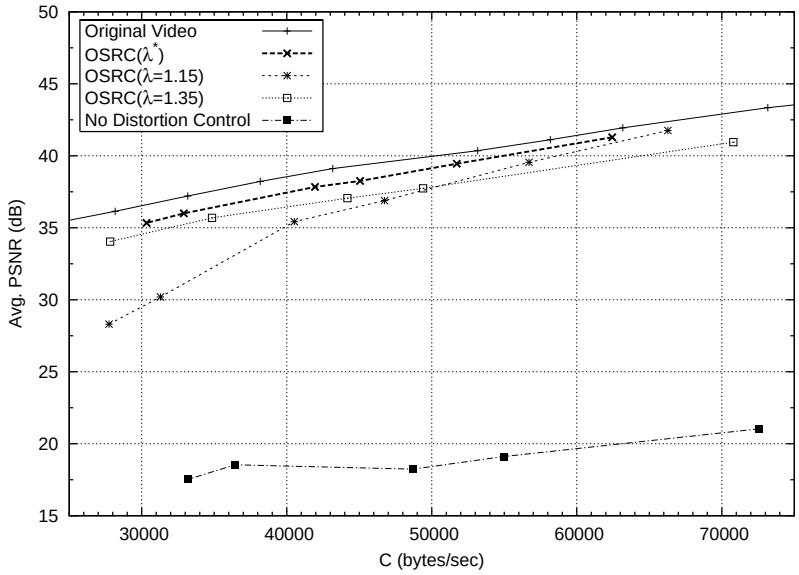

Figure V.11: Comparison of OSRC with policy switching to OSRC with fixed λ and no distortion control scenarios

approaches both outperform the no distortion control case. Our results also show that if the channel mean RTT is low as compared to the time between transmission opportunities, intelligent ARQ schemes outperform FEC. However as the channel delay increases, FEC becomes advantageous.

In Figure V.11 we present the validation of the OSRC algorithm by conduct-

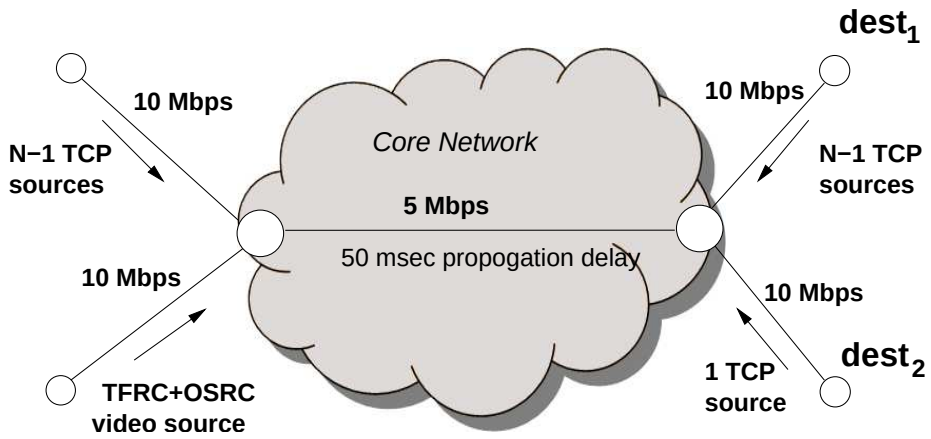

Figure V.12: Topology used in ns-2 simulations

ing more realistic simulations using ns-2. In our ns-2 simulations we created the following simple topology in Figure V.12 in order to validate the OSRC algorithm. Note that $N \in \{10, 15, 20, 25, 30, 35, 40\}$ corresponds to the total number of traffic sources used in one direction. However in the forward direction we use $N-1$ infinite FTP sources using TCP (i.e. FTP+TCP) contending together with a single OSRC+TFRC source. In the simulations we presented the no distortion control case as a reference point that indicates the lowest possible performance. We used two fixed OSRC policies that makes use of the $\lambda = 1.15, 1.35$ where the low and high λ correspond to the low and high protection cases respectively. Since they do not have the capability to adapt their policies, it gives a performance similar to that of the static FEC protection. High protection case OSRC($\lambda = 1.35$) successfully recovers from packet drops however it suffers from a rate distortion penalty due to unnecessary introduction of redundancy. Therefore it is nearly parallel to the original video but with an offset due this rate-distortion penalty. On the other hand low-protection case, OSRC($\lambda = 1.15$) gets close to the original video when there is excessive bandwidth (i.e. low number of packet drops), nevertheless low-protection case fails in recovering from packet losses when error rates are higher. The OSRC algorithm performs better then the fixed λ schemes on each end (i.e. low or high error rate). Furthermore the policy switch profile due to calculation of a new λ^* can be observed in Figure V.13, which is illustrated for $N = 40$ where the same topology in Figure V.12 is used.

Finally we demonstrate the effect of using OSRC based video streaming

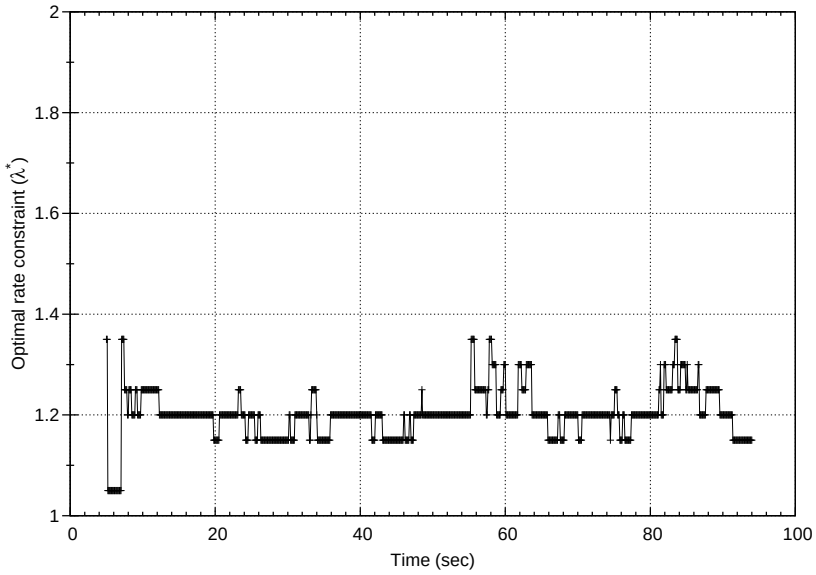

Figure V.13: Switching between policies and optimal rate control λ^* parameter

on the TCP-friendliness of the underlying TFRC transport protocol. For this purpose we compare the bandwidth share taken by an OSRC+TFRC source with the bandwidth share of an infinite TFRC source. Results are obtained by by replacing the video source with an FTP+TFRC source while keeping the background TCP traffic same by selecting $N \in \{10, 15, 20, 25, 30, 35, 40\}$ in Figure V.12. Figure V.14 reveals the similarity between the bandwidth share of a standard TFRC source (i.e. FTP+TFRC) and the OSRC algorithm using TFRC.

Finally we compare the bandwidth share of an OSRC+TFRC source directly with an infinite TCP source (i.e. FTP+TCP). In this experiment, rather than fixing the capacity of the *Core Network* to 5 Mbps. as given in Figure V.12 and varying N, we fixed $N = 2$ and changed the link's capacity for values $\{0.750, 0.875, 1.0, 1.125, 1.250, 1.375, 1.5\}$ Mbps. The fairness between the

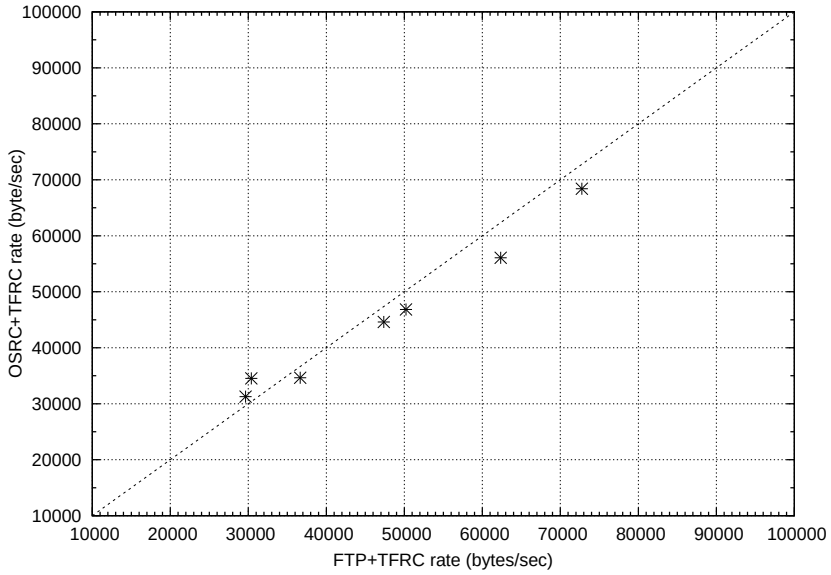

Figure V.14: Using TCP as background traffic, fairness comparison between TFRC sources carrying either FTP or OSRC data

sources OSRC+TFRC and FTP+TCP is demonstrated in Figure V.15.

V.4 Conclusion

In this chapter given the channel statistics and video distortion model we introduce an optimal packet scheduling and rate control (OSRC) scheme. This scheme is capable of adapting itself to the changing network conditions by means of policy switching in order to minimize the average distortion at the receiver. These policies $\pi^*(\lambda), \lambda \in \Lambda$ are calculated a priori by the OS algorithm and optimal in terms maximizing the average on time successful delivery probability $P_{suc}(\lambda)$ of packets and depending on the optimal rate constraint λ^* provided by the optimal rate control (RC) algorithm. On the other hand, the RC algorithm utilizes the analytical video distortion model and the $P^*_{suc}(\lambda), \lambda \in \Lambda$ in order

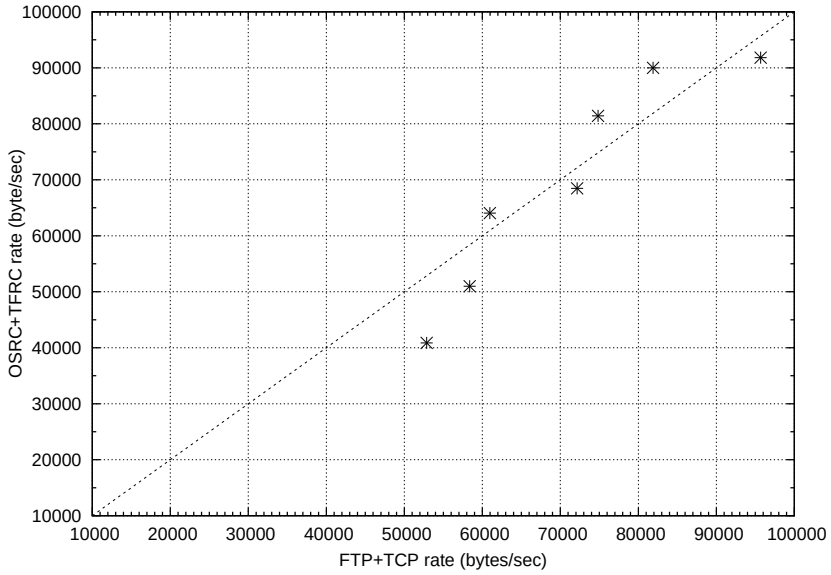

Figure V.15: Fairness comparison between the OSRC+TFRC and the FTP+TCP sources sharing the same link

to calculate optimal rate constraint λ^*. Finally, until network statistics change, the optimal policy $\pi^*(\lambda^*)$ is used by the OS algorithm throughout the streaming session.

CHAPTER VI

CONCLUSION

In this thesis we propose two novel network adaptive video streaming schemes which are both making rate-distortion control by means of retransmissions and friendly to TCP flows, where the first algorithm SFD (Selective Frame Discard) is TCP based and necessitates the use of a large playout buffer, while on the other hand the second algorithm OSRC (Optimal Scheduling and Rate Control) is TFRC based and enables the use of a smaller playout buffer as compared to the SFD case.

Selective Frame Discard (SFD) algorithm is a network adaptive video streaming scheme which is capable of delivering a two layer temporally scalable video by conducting a heuristic based input buffer management scheme using frame display time estimates and using TCP transport. In this streaming scheme frames are delayed in the input buffer due to a mismatch between video bit rate and TCP's rate estimate. The SFD algorithm manages the input buffer delay by selectively discarding frames from the enhancement layer by prioritizing the base layer frames in order to match the estimated rate. Our simulation results

reveal that with the proper analysis of video streaming dynamics and the use of a large initial playout buffer in the order of seconds, it is possible to make video streaming over TCP transport. Furthermore the use of TCP transport provides some unique advantages such as, ubiquity in all computer systems, conformance with other TCP flows and transparency to firewalls. Besides, similar to TCP-friendly and ECN conformant congestion control algorithms, TCP provides an inherent congestion control algorithm and ability to use transport layer signals (i.e. ECN). The proposed SFD algorithm over TCP transport is simple, effective and easily implementable due to the ubiquity of TCP, however it necessitates a large playout buffer in order to tolerate the extra latency introduced by fluctuations in TCP's rate estimation and can easily be used as a streaming system for delay tolerant non-interactive applications. Even though non-interactive video streaming applications are delay tolerant, long playout buffering is not preferable since user should wait for the same buffering time in every repositioning action within a stream (i.e. forward and rewind). In the second algorithm in order to minimize initial playout delay, we replace TCP transport and two layer temporal scalability which both have adverse effects on streaming performance, by TCP-friendly rate control (TFRC) and fine granular scalability (FGS) schemes respectively.

Optimal packet scheduling and rate control (OSRC) scheme is a network adaptive rate-distortion control algorithm implemented at the application layer that makes use of the FGS (Fine Granular Scalable) coded video and TFRC transport. The proposed algorithm is novel in terms of conducting a joint op-

timization procedure which calculates optimal packet schedules (policies) $\pi^*(\lambda)$ for $\lambda \in \Lambda$ while making optimal rate control(adaptation) determined by λ^*, yet many research in the literature focus these problems separately. Since the OSRC algorithm calculates the optimal bit rate besides the calculation of optimal policies, it is applicable to streaming of both stored scalable and real-time coded/transcoded video. In addition to that, OSRC is a low complexity algorithm with a cost of extra need for storing a priori calculated optimal policies while providing a higher granularity for redundancy selection. Hence the OSRC algorithm is applicable for video-on-demand and live video applications where channel delay is low as compared to $M.\Delta T$ and in such cases provide superior performance as compared to the algorithms that make use of the forward error correcting (FEC) erasure codes. Furthermore OSRC scheme yields better rate adaptation capability than FEC based schemes with small number of source blocks, due to its high granularity in redundancy injection.

CHAPTER VII

PUBLICATION LIST

VII.1 Refereed National Conferences

1. S. İnce, E. Gurses, G. Bozdagi Akar, "Seri kanallar için gerçek zamanlı çoğulortam haberleşmesi uygulaması," in *Sinyal İşleme Uygulamaları Kurultayı (SİU)*, June 2002, Ankara, Türkiye .

2. E. Gurses, G. Bozdagi Akar, N. Akar, "TCP/IP Ağlarında kesintisiz görüntü iletimi için katmanlı duraksız video iletimi," in *Sinyal İşleme Uygulamaları Kurultayı (SİU)*, June 2003, İstanbul, Türkiye.

3. E. Gurses, G. Bozdagi Akar and N. Akar, "Optimal Video Streaming Under Rate Constraints," in *IEEE Signal Processing and Communications Applications Conference (SİU)*, April 2006, Antalya, Turkey.

VII.2 Refereed International Conferences/Journals

1. E. Gurses, G. Bozdagi Akar and N. Akar, "Impact of Scalability in Video Transmission in Promotion Capable Differentiated Services Networks," in *Proceedings of ICIP* Vol. 3, pp.753-756, September 2002

2. E. Gurses, G. Bozdagi Akar and N. Akar, "A novel architecture for layered video streaming over TCP/IP," in *Proceedings of 4th COST 276 Workshop on Transmitting, Processing and Watermarking Multimedia Contents*, pp.77-81, Bordeaux, France, 2003.

3. E. Gurses, G. Bozdagi Akar and N. Akar, "Selective Frame Discarding for Video Streaming in TCP/IP Networks," in *Packet Video Workshop*, Nantes, France, April 2003,

4. E. Gurses, G. B. Akar and N. Akar, "A simple and effective mechanism for stored video streaming with TCP transport and adaptive frame discard," in *Computer Networks*, vol. 48, pp. 489-501, 2005.

5. E. Gurses, O. B. Akan, "Multimedia communication in wireless sensor networks," in *Annals of Telecommunications*, vol. 60, no. 7-8, pp. 872-900, July-August, 2005.

REFERENCES

[1] N. Laoutaris and I. Stavrakakis, "Intrastream synchronization for continuous media streams: A survey of playout schedulers," in *IEEE Network Magazine*, vol. 16, no. 3, pp. 30-40, May 2002.

[2] M. Kalman and E. Steinbach and B. Girod, "Rate-distortion optimized video streaming with adaptive playout," in *Proc. of IEEE ICIP*, vol. 3, pp. 189-192, Rochester, NY, June 2002.

[3] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," in *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458-472, August 1999.

[4] J.Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," in *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133-145, April 2000.

[5] S. Floyd, M. Handley, J. Padhye and Joerg Widmer, "Equation-based congestion control for unicast applications," in *ACM SIGCOMM*, pp. 43-56, Stockholm, Sweden, August 2000.

[6] A. Lippman, "Video coding for multiple target audiences," in *SPIE Conference on Visual Communications and Image Processing*, vol. 3653, pp. 780-782, January 1999.

[7] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman and Y. A. Reznik "Video coding for streaming media delivery on the Internet," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269-281, March 2001.

[8] D. Wu and Y. T. Hou and Y. Q. Zhang, "Transporting real-time video over the Internet: Challenges and Approaches," in *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1855-1875, December 2000.

[9] Z.-L Zhang, S. Nelakuditi, R. Aggarwal and R. P. Tsang, "Efficient selective frame discard algorithms for stored video delivery across resource constrained networks," in *Proceedings of INFOCOM*, vol. 2, pp. 472-479, 1999.

[10] B. G. Haskell and A. Puri and A. N. Netravali, "Digital video: An introduction to MPEG-2," Kluwer Academic Publishers, December 1996.

[11] A. Puri and T. Chen "Multimedia systems, standards, and networks," Marcel Dekker, New York/Basel, March 2000.

[12] ITU-T, "Video coding for low bit rate communication," *ITU-T Recommendation H.263*, February 1998.

[13] G. Cote, B. Erol, M. Gallant and F. Kossentini, "H.263+: Video coding at low bit rates," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7,pp. 849-866, 1998.

[14] H. Radha, M. van der Schaar and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," in *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53-68, March 2001.

[15] M.-T Sun and A. Reibman, Ed., "Compressed Video over Networks," Marcel Dekker, pp. 251-308, 2001.

[16] R. Rejaie, M. Handley and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proceedings of INFOCOM*, vol. 3, pp. 1337-1345, March 1999.

[17] D.Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *Proceedings of INFOCOM*, vol. 2, pp. 631-640, 2001.

[18] Y. Yang, M. Kim and S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proceedings of INFOCOM*, pp. 1716-1725, May 2001.

[19] E. Kohler, M. Handley and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," *IETF*, RFC 4340 , March 2006.

[20] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," in *IETF*, RFC 2581, Proposed Standard, April 1999.

[21] R. Stewart et al, "Stream Control Transmission Protocol," in *IETF*, RFC 2960, October 2000.

[22] S. Floyd and E. Kohler, "Profile for DCCP congestion control ID2: TCP-like congestion control," *IETF* RFC 4341, March 2006.

[23] S. Floyd and E. Kohler, "Profile for DCCP congestion control ID3: TCP-friendly congestion control TFRC," *IETF* RFC 4342, March 2006.

[24] S. Floyd, "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field," in *IETF Internet Draft*, draft-ietf-tsvwg-ecn-alternates-00.txt, January 2006.

[25] M. Podolsky, S. McCanne and M. Vetterli, "Soft ARQ for layered streaming media," Technical Report UCB/CSD-98-1024, University of California, Computer Science Division, Berkeley, November 1998.

[26] M. Hemy, U. Hengartner and P. Steenkiste, "MPEG systems in best-effort networks," in *Packet Video Workshop*, New York 1999.

[27] H. Cha, J. Oh and R. Ha, "Dynamic frame dropping for bandwidth control in MPEG streaming system," in *Multimedia Tools and Applications*, vol. 19, pp. 155-178, 2003.

[28] Y. Dong, R. Rakshe and Z-L. Zhang, "A Practical Technique to Support Controlled Quality Assurance in Video Streaming across the Internet," in *Packet Video Workshop*, Pittsburgh, Pennsylvania, April 2002.

[29] P. Mehra and A. Zakhor, "TCP-Based Video Streaming Using Receiver-Driven Bandwidth Sharing," in *Packet Video Workshop*, Nantes, France, April 2003.

[30] C. Krasic, K. Li and J. Walpole, "The Case for Streaming Multimedia with TCP," in *International Workshop on Interactive Distributed Multimedia Systems*, Lancaster, UK, 2001.

[31] I. V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman and J. Woods, "Integrated end-end buffer management and congestion control for scalable video communications," in *Proc. of IEEE ICIP*, vol. 3, pp. 257-260, 2003.

[32] S. Floyd, "TCP and Explicit Congestion Notification," in *ACM Computer Communication Review*, vol. 24, no.5, pp. 10-23, October 1994.

[33] J.Heinanen, F. Baker, W. Weiss and J. Wroclawski, "Assured forwarding PHB group," in *IETF*, RFC 2597, 1999.

[34] S. Floyd, "TCP Extensions for High Performance," in *IETF*, RFC 1323, May 1992.

[35] Information Sciences Institute (ISI), USC, "The Network Simulator - ns-2," *http://nsnam.isi.edu/nsnam/index.php/Main_Page*, July 2006 (last access 3 July 2006).

[36] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," in *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, August 1993.

[37] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Service," in *IETF*, RFC 2475, May 1999.

[38] U. Bodin, U. Schelen and S. Pink, "Load-tolerant differentiation with active queue management," in *ACM Computer Communication Review*, vol. 30, no. 1, January 2000.

[39] E. Gurses, G. B. Akar and N. Akar, "A simple and effective mechanism for stored video streaming with TCP transport and adaptive frame discard," in *Computer Networks*, vol. 48, pp. 489-501, 2005.

[40] M. Zorzi, "Performance of FEC and ARQ error control in bursty channels under delay constraints," in *Proc. of IEEE VTC*, Ottawa, Canada, May 1998.

[41] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: does burst-length matter?," in *Proc. IEEE ICASSP*, Hong Kong, April 2003.

[42] V. Paxson, "End-to-end Internet packet dynamics," in *IEEE Trans. on Networks*, vol. 7, no. 3, pp. 277-292 June 1999.

[43] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation over packet erasure channels through forward error correction," in *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 819-828, Apr. 2000.

[44] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," in *Signal Processing: Image Communications*, vol. 15, pp. 77-94, 1999.

[45] Y. Wang, A.R. Reibman, and S. Lin, "Multiple description coding for video delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 57-70, January 2005.

[46] M. Elaoud and P. Ramanathan, "Adaptive use of error-correcting codes for real-time communication in wireless networks," in *Proceedings of INFOCOM*, March 1998.

[47] S. H. Kang and A. Zakhor, "Packet scheduling algorithm for wireless video streaming," in *Packet Video 2002*, Pittsburgh, April 2002.

[48] Z. Miao and A. Ortega, "Expected run-time distortion based scheduling for delivery of scalable media," in *Packet Video 2002*, Pittsburgh, April 2002.

[49] K. Stuhlmüller, N. Färber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, June 2000.

[50] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *Microsoft Research Tech. Rep.*, MSR-TR-2001-35, Feb. 2001.

[51] P. Chou, A. E. Mohr, A. Wang and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," in *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 108-121, March 2001.

[52] W.-T. Tan and A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Packet Video Workshop*, New York, April 1999.

[53] J. Chakareski, J. Apostolopoulos, B. Girod, "Low-Complexity Rate-Distortion Optimized Video Streaming," *IEEE ICIP*, October 2004.

[54] J. Chakareski, P. Frossard, "Rate-Distortion Optimized Distributed Packet Scheduling of Multiple Video Streams Over Shared Communication Resources," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 207-218, March 2006.

[55] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Proc. IEEE Data Compression Conference*, Snowbird, UT, April 2003.

[56] J. Chakareski and P. A. Chou, "Application layer error correction coding for rate-distortion optimized streaming to wireless clients," in *IEEE Transactions on Communications*, vol.52, no.10, Oct. 2004.

[57] D. Tian, X. Li, G. Al-Regib, Y. Altunbasak and J. R. Jackson, "Optimal packet scheduling for wireless video streaming with error-prone feedback," in *Proc. IEEE WCNC*, Atlanta, Mar. 2004.

[58] ITU-T, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H.264*, May 2003.

[59] H. J. Tijms, "Stochastic Models: An Algorithmic Approach," John Wiley & Sons, April 1995.

[60] J. G. Apostolopoulos, W-t. Tan and S. J. Wee, "Video Streaming: Concepts, Algorithms, and Systems," *HP Laboratories Technical Report*, HPL-2002-260, September 2002.

[61] A. Mukherjee, "On the dynamics and significance of low frequency components of internet load." *Internetworking: Res. Experience*, vol. 5, pp. 163-205, Dec. 1994.

[62] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," in *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 637-644, July 2003.

[63] R. Puri, K.-W. Lee, K. Ramchandran and V. Bharghavan, "An Integrated Source Transcoding and Congestion Control Paradigm for Video Streaming in the Internet," in *IEEE Transactions on Multimedia*, Vol. 3, No. 1, pp. 18-32, March 2001.