Adaptation Strategies for Scalable Video Streaming

by

Burak Görkemli

A Dissertation Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy
in
Computer Engineering

Koç University

August 2010

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Burak Görkemli

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. A. Murat Tekalp (Advisor)

_____

Assoc. Prof. M. Oğuz Sunay

_____

Prof. M. Reha Civanlar

_____

Assoc. Prof. Öznur Özkasap

_____

Assist. Prof. Alptekin Küpçü

Date: _____

Işıl'a,

hayatımın ışığına,

beni 7 yıldır doktoramla paylaşan karıma,

artık seninim,

geriye ne kadar kaldıysa...

ABSTRACT

Traditionally, video transport has been realized over dedicated, fixed bandwidth channels, such as terrestrial, satellite, or cable. Hence, video coding standards, such as MPEG-2 and MPEG-4 AVC, encode video at a fixed target rate for a given resolution and application. With the advent of video over IP and WebTV, video transport must now be achieved over heterogeneous IP networks, including a variety of fixed and wireless links. It is well-known that achieving the best video quality in this heterogeneous environment requires an adaptive streaming framework that can most efficiently adapt the source video rate to the available network throughput.

Fundamental blocks of an adaptive streaming framework are a codec that can output video at multiple rates, a transport protocol that employs effective rate/congestion control, and an adaptation engine built on top of these. This work is on realizing the adaptation engine by supplying it with proper adaptation strategies to be used both in coding and transport blocks. For that purpose, an adaptive video streaming system, which employs SVC as the video codec and DCCP or TCP as the transport protocol, is implemented to evaluate various adaptation strategies in streaming scalable video. The system is also extended to wireless domain, proposing a solution to differentiate between wireless and congestion losses and therefore improve the performance of DCCP in wireless networks.

ÖZET

Geleneksel olarak video aktarımı, bant genişliğinin sabit olduğu karasal, uydu veya kablolu yayınlar üzerinden yapılmaktadır. Bu yüzden, MPEG-2 ve MPEG-4 AVC gibi video kodlama standartları istenen çözünürlük ve uygulamaya bağlı olarak görüntüyü sabit bir oranda sıkıştırırlar. Gelişmekte olan IP ve WebTV üzerinden video aktarımı çözümlerinde ise, aktarımın birbirinden farklı ve değişen kapasitelere sahip, kablolu veya kablosuz IP ağları üzerinden yapılması gerekmektedir. Bilindiği üzere, bu tarz değişen kapasitelere sahip hatlar üzerinden yapılan yayınlarda en iyi görüntü kalitesine ulaşabilmek için, videonun sıkıştırma oranını hat kapasitesine göre düzenleyen uyarlanabilir video aktarım çözümleri kullanılmalıdır.

Uyarlanabilir aktarım çözümleri temelde üç parçadan oluşur: Videoyu farklı oranlarda sıkıştırabilen bir kodlayıcı, hatta oluşabilecek tıkanıklığa duyarlı bir iletim protokolü ve bu ikisini kullanacak olan bir uyarlama motoru. Bu çalışma, hem iletim hem de kodlama sırasında uygun tekniklerle nasıl bir uyarlama motoru yapılabileceğini açıklamaktadır. Bu amaçla çalışmada, ölçeklenebilir SVC video kodlayıcısı ile DCCP ve TCP iletim protokolleri kullanılmıştır. Ayrıca sistem, DCCP protokolünün kablosuz ağlardaki performansını arttırmak için kablosuz paket kayıplarıyla tıkanıklık kayıplarını birbirinden ayıran bir mekanizma barındırmaktadır.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and profound respect to Prof. A. Murat Tekalp for his reliable guidance, inspiration, valuable support and everlasting patience over the past years. I have benefited tremendously from his wisdom and character. I also want to thank Assoc. Prof. M. Oğuz Sunay for his guidance, patience and financial support through his TÜBİTAK project. I am grateful to Prof. Reha Civanlar, who had been my first advisor in Ph.D., and I feel myself lucky to have had the opportunity to work with him.

I am also grateful to members of my thesis committee for their valuable comments. I would like to thank Assoc. Prof. Öznur Özkasap and Assist. Prof. Alptekin Küpçü for the critical reading of this thesis and serving on my defense committee.

Next, I would like to thank my brother Göktuğ, with whom I shared more time than my wife – which should also be correct for him. Gratitudes to the rest of the gang who made it unforgettable: Mali of the Dark Side, Görkem the tidy, Emrah the handsome, Ekin the magnificient problem, Emre the man, Çağdaş the wireless, and Erkan the whatever. Special thanks to Çağdaş, who helped me greatly in building the wireless test setup, and to Yalçın, with whom I worked in realizing the work in Chapter 2.

Also, I would like to thank my employer Argela Technologies, for the flexibility and understanding they provided during my Ph.D. study. No other company would have done the same.

My everlasting love to Işıl, to whom I dedicate this work, and Eylül, the second best that came to my life. And finally I would like to thank my parents Özden and Tevfik, without whom there would be none of this.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

MPEG	Moving Picture Experts Group
AVC	Advanced Video Coding
IP	Internet Protocol
SVC	Scalable Video Coding
DCCP	Datagram Congestion Control Protocol
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
CWND	Congestion Window
UDP	User Datagram Protocol
AIMD	Additive Increase Multiplicative Decrease
TFRC	TCP Friendly Rate Control
CCID	Congestion Control Identifier
SCTP	Stream Control Transmission Protocol
VTP	Video Transport Protocol
ARQ	Automatic Repeat request
MGS	Medium Grain Scalability
MAC	Medium Access Control
CGS	Coarse Grain Scalability
NAL	Network Abstraction Layer
MTU	Maximum Transmission Unit
GoP	Group of Pictures
JSVM	Joint Scalable Video Model
PSNR	Peak Signal-to-Noise Ratio
LAN	Local Area Network
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
CIF	Common Intermediate Format
FTP	File Transfer Protocol
RLC	Radio Link Control
BS	Base Station
ACK	Acknowledgment
SNR	Signal-to-Noise Ratio

Chapter 1

INTRODUCTION

1.1 Motivation and problem definition

Traditionally, video transport has been realized over dedicated, fixed bandwidth channels, such as terrestrial, satellite, or cable. Hence, video coding standards, such as MPEG-2 and MPEG-4 AVC, encode video at a fixed target rate for a given resolution and application. With the advent of video over IP and WebTV, video transport must now be achieved over heterogeneous IP networks including a variety of fixed and wireless links. It is well-known that achieving the best video quality in this heterogeneous environment requires an adaptive streaming framework that can most efficiently adapt the video encoding rate to the available network throughput. Adaptive streaming can be employed in both push-based (sender-controlled) and pull-based (receiver-controlled) applications. Traditional media streaming servers, where a central server is in charge of estimating the available bandwidth and adapting the video rate accordingly, are examples of push-based applications. The clients are mainly responsible for the flawless playout of the streamed media, and while they may aid the streaming server in bandwidth estimation, the adaptation logic resides on the server. On the other hand, in HTTP Live Streaming [1], which describes a protocol for pull-based media streaming, the adaptation logic is shifted to clients. The server is responsible for servicing incoming client requests, while the clients estimate the available network bandwidth and request appropriate chunks of media.

Fundamental blocks of an adaptive streaming framework are a codec that can output video at multiple rates, a transport protocol that employs effective rate/congestion control, and an adaptation engine built on top of them, as depicted in Figure 1.1.

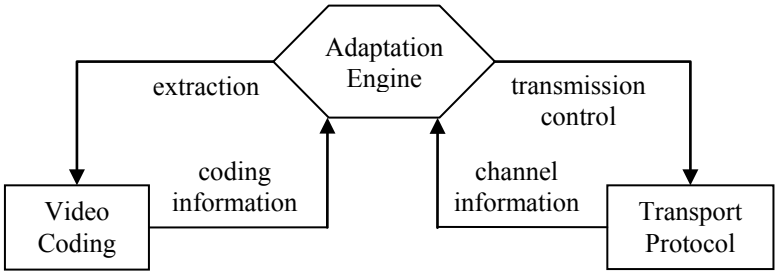


Figure 1.1 Main blocks of an adaptive streaming framework.

The video coding characteristics play an important role on the adaptation strategies to be employed. [2] gives an overview on the adaptation methods used for different encoding configurations. For instance, the video can be encoded in multiple streams with varying bitrates and the sender can switch between these bitstreams according to the network conditions [3]. In HTTP Live Streaming, the client selects from a number of alternate streams containing the same material encoded at a variety of data rates, allowing the streaming session to adapt to the available network rate. A more elegant solution is coding video with multiple layers in a scalable fashion and switching between these layers. In this respect, the Scalable Video Coding (SVC), which is an extension of the Advanced Video Coding (AVC) standard, is a promising video format which introduces spatial, temporal and quality scalability [4]. [5] presents some typical schemes used in adapting an SVC coded stream, such as spatial, temporal, SNR, priority-based, and ROI-based adaptation; while the spatial-temporal resolution of the scalable video is modified depending on the estimated bandwidth in [6]. Real-time encoding of the video is another video adaptation scheme, which can be employed both for non-scalable [7] and scalable [8] videos.

Being the *de facto* transport protocol of the Internet, the TCP is the first that comes to mind when transport protocols are considered. But it may be unsuitable to use TCP when streaming video, especially live video with strict end-to-end delay constraint, due to TCP's lack of control on delay (because of reliability and in order delivery) and its rapidly changing transmission rate. On the other hand, TCP is the easiest choice for streaming stored media, with its built-in congestion control, reliable transmission and firewall friendliness. Popular video distribution sites, such as YouTube, Vimeo and Metacafe, use HTTP over TCP to stream video to clients. Moreover, it has been shown in [9] that using TCP for streaming video provides good performance when the available network bandwidth is about twice the maximum video rate, with a few seconds pre-roll delay. [10] tries to minimize the latency caused by large TCP send buffers by tuning the send buffer size to follow the TCP congestion window (CWND).

An alternative to TCP is the UDP, which does not accommodate TCP's built-in congestion control and reliable, in order packet delivery, leaving their implementations to the application layer. Since congestion control is crucial for the stability of the Internet, a type of congestion control scheme should be realized over UDP, whose examples are listed in [11] and [12]. [11] classifies these schemes into four groups: TCP's Additive Increase Multiplicative Decrease (AIMD) based schemes, equation based schemes, methods using the bandwidth estimation scheme employed in the TCP Westwood and receiver buffer occupancy based schemes.

The TCP Friendly Rate Control (TFRC) is a well-known equation based congestion control mechanism designed for unicast flows, which uses the TCP throughput equation for calculating the allowed TCP friendly sending rate [13]. The Datagram Congestion Control Protocol (DCCP) [14] is a new transport protocol, implementing bi-directional unicast connections of congestion-controlled, unreliable datagrams, which uses TFRC as one of its selectable congestion control mechanisms. DCCP accommodates a choice of modular

congestion control mechanisms, to be selected at connection startup, which are the stated TFRC identified by Congestion Control Identifier 3 (CCID3) [15] in DCCP and the TCP-like Congestion Control identified by CCID2 [16]. TCP-like Congestion Control performs similar to TCP's AIMD mechanism, halving the congestion window in response to congestion.

DCCP is designed for media streaming applications, which do not prefer to use TCP due to lack of control on delay and which do not like to implement complex congestion control mechanisms required when using UDP. [17] compares the performance of streaming video over DCCP in heterogeneous networks with UDP and the Stream Control Transmission Protocol (SCTP), which is a reliable, message-oriented data transport protocol, and concludes that DCCP achieves better results than SCTP and UDP. Moreover, it is shown in [18] that using DCCP for streaming in wireless networks results in video with better quality than UDP. Congestion control schemes CCID2 and CCID3 of DCCP are compared to each other, as well as to TCP, UDP in [19] and it has been found that CCID2 achieves more throughput than CCID3, while increasing the end-to-end delay. However, no impact on the received video quality is given in the paper.

Video Transport Protocol (VTP) is proposed in [20], attempting to maximize the quality of real-time MPEG-4 video streams while simultaneously providing basic end-to-end congestion control. VTP behaves similar to AIMD in the sense that it performs additive increase but not multiplicative decrease: The protocol adjusts its transmission rate to the rate perceived by the receiver when there is congestion.

Whichever protocol is used for transport, a mechanism should exist to estimate the available network rate and adapt the video rate accordingly, in order to achieve better video quality. This can be performed by using the receiver buffer occupancy information to prevent any buffer underflow/overflow [11][21] or by combining the receiver buffer state with the bandwidth estimate [6]. [3] employs a virtual network buffer between the sender

and the receiver together with end-to-end delay constraints to adapt the video being transmitted, while the same virtual network buffer algorithm is also utilized by [12] in implementing source rate control and congestion control jointly. Packets may be sent depending on their rate distortion values, as in [22] and [23]. A similar approach is presented in [24], where the expected distortion is tried to be minimized through optimal selection of the packets to be transmitted. In case DCCP CCID3 is used, the TFRC rate calculated by DCCP can be utilized by the sender to estimate the available network rate, without requiring an additional estimation mechanism.

In this work, the methods of streaming SVC coded video over wired/wireless networks using DCCP or TCP are investigated and their results are compared in terms of received video quality and used network resources, based on our previous work [25]-[30]. UDP is not utilized in our tests, because it does not provide a congestion control scheme and it is outperformed by DCCP, as shown in [17]-[19].

1.2 Contributions

The contributions of the work can be listed as follows:

- To the best of our knowledge, this is the first work in streaming SVC video over DCCP on the real Internet with adaptation of the video rate to the actual throughput.
- An extensive set of MGS fragmentation configurations are compared in terms of their rate-distortion performance of extracted video streams using different extraction methods with byte-limited slice mode disabled/enabled.
- A new extraction method, called *priority-based hierarchical extraction*, which takes advantage of the hierarchical B-pictures structure and using the priority information computed during encoding, is proposed.
- The oscillatory nature of DCCP is discovered when the transmission rate is matched to the video extraction rate, which lowers the quality of the received video.

- An adaptive Automatic Repeat reQuest (ARQ) scheme is proposed, which can request all lost packets or only the base layer packets, depending on the decoder buffer state.
- A comparison of transport protocols, DCCP (CCID3, CCID2) and TCP, is presented.
- In order to improve the performance of DCCP in wireless networks, a solution based on limited checksum coverage is proposed to discriminate wireless losses from congestion losses. In this way, DCCP reacts only to congestion losses.

1.3 Organization

Chapter 2 describes the Scalable Video Coding (SVC), with key encoder parameters and various video extraction methods. Among several encoding configurations and extraction methods, the best performing combination is searched, in terms of resulting video quality and the required bitrate.

Chapter 3 introduces the adaptive video streaming system, implemented to test various adaptation strategies in streaming scalable video. Tests are done on wired networks to evaluate the adaptation strategies proposed.

Chapter 4 extends the adaptive video streaming system to wireless domain. A solution based on limited checksum coverage is proposed to distinguish wireless losses from congestion losses, and tests are performed using a wireless emulator based on the IS-856 standard (1xEV-DO).

Finally, the dissertation is concluded with a short summary of the performed study and future research work.

Chapter 2

SCALABLE VIDEO CODING AND EXTRACTION

2.1 Introduction

The Scalable Video Coding (SVC), extension of the Advanced Video Coding (AVC) standard, provides temporal scalability through use of hierarchical prediction structures, whereas spatial and quality scalability are supported by multilayer coding [4]. Quality scalability is supported under two modes: *Coarse-grained scalability* (CGS) and *medium-grained scalability* (MGS). CGS, also called *layer-based scalability*, is based on the multi-layer concept of SVC, meaning that only a small number of bitrates can be selected during adaptation. A CGS layer cannot be partially retained or removed; adaptation should be performed on complete layer basis. Additionally, switching between different CGS layers can only be done at defined points in the bit stream. However, the MGS concept allows any enhancement layer Network Abstraction Layer (NAL) unit to be discarded during bit stream extraction, thanks to modified high-level signaling, enabling packet-based scalability [31]. The MGS scheme allows splitting zig-zag scanned transform coefficients into a number of fragments and forming a separate sub-layer for each of the resulting fragments. In this way, an MGS layer can be split into between 2 to 16 sub-layers. It is also possible to put all the coefficients into a single layer, without any fragmentation.

2.2 Encoder configuration

In this study, MGS based quality scalability is used to produce the videos to be streamed adaptively, because of its rate-adaptation flexibility over CGS. Spatial and

temporal scalability are not employed, since their effect to the resulting quality may be subjective.

Having selected MGS for scalability exposes another configuration decision to be made: How to fragment the MGS layer into sub-layers? As an MGS layer is split into sub-layers, the number of available NAL unit combinations that can be selected in extraction increases, thus increasing the rate adaptation points. However, it should be noted that each sub-layer added introduces a cost in terms of rate-distortion. Hence, it is important to achieve a balance between adaptation and rate-distortion performance.

Slice mode is another important parameter for configuring the encoder, enabling a video frame to be split into slices to minimize the effect of packet loss on received video quality. It is highly probable for some video packets to be lost during streaming, no matter what retransmission or forward error correction techniques are deployed, when an unreliable transport protocol like UDP or DCCP is used. In such a scenario, dividing video frames into slices will limit the effect of packet loss to the corresponding slice, instead of the entire frame.

Slicing can be done by limiting either the number of macroblocks or the number of bytes carried by a slice. When streaming over UDP or DCCP, it is preferable to use slicing in byte-limitation mode and keep the slice sizes less than 1500 bytes, which is the conventional MTU length. In this way, each slice is carried in a single transport packet. On the other hand, when TCP is used, slicing does not improve the received video quality, if not decrease it due to slicing overhead, since the protocol does not preserve packet boundaries. Hence, when streaming over TCP, it is advised to keep one slice per frame and disable slice mode.

In addition to limiting the effects of packet loss, slicing also increases the granularity of an MGS layer; because each layer is carried in multiple NAL units, rather than being carried in a much bigger single NAL unit. Rate adaptation can then be performed by

selecting a set of slices belonging to a layer, in case it is not possible to extract that particular layer as a whole due to rate constraints. Since each slice data corresponds to a specific region in a frame, a quality enhancement gained by selecting a set of slices will only affect the corresponding regions, which may result in undesired quality variations over a frame.

2.3 Video extraction

While streaming SVC video adaptively over the Internet, it is a common practice to extract video on a Group of Pictures (GoP) basis. Each GoP of video is extracted independently from the others, such that the given rate constraint is satisfied. Selecting the NAL units to be streamed can be performed in various schemes, provided that the NAL units belonging to the base layer are completely extracted in all schemes. The simplest extraction method, called *flat-quality extraction*, distributes the rate budget among the frames in a GoP such that each frame will end up having the same number of quality layers, if the budget allows. Otherwise, some frames will be extracted in lesser quality than the others. In case of using hierarchical B-pictures, the budget distribution will be in the order of temporal layers for each quality layer. JSVM implements a version of this method, in which it is assured that all frames are extracted at the same quality layer.

Priority-based extraction is another extraction scheme having a better rate-distortion performance from flat-quality extraction for most of the MGS fragmentation configurations, which uses the priority identifiers for each NAL unit that can be inserted as a post-encoding process [32]. The priority identifier, ranging from 0 to 63 with increasing priority, represents the contribution of the corresponding NAL unit to the video quality. The extraction method starts with selecting the NAL units with the highest priority identifier and continues in decreasing order, until the rate constraint is exceeded. It is

evident that this extraction method requires the priority information to exist in the video stream.

We propose an improved priority-based extraction, called *priority-based hierarchical extraction*, which takes advantage of the hierarchical B-pictures structure together with the priority information. Differently from the priority-based extraction, this scheme starts with the lowest temporal layer t_0 to select the NAL units based on their priority identifiers and continues with the remaining temporal layers, t_1, t_2, and so on, as long as the rate budget allows. In case the priority information is not present in the video sequence, *hierarchical extraction* can be utilized in place of this extraction method, as proposed in [31], showing comparable performance.

The extraction methods listed above are not the only possible schemes available. Other strategies, like [33], may result in better rate-distortion performance. However, it is not in the scope of this study to find the best extraction method, but to compare various MGS fragmentation configurations using basic extraction schemes.

2.4 Results

In this section, we compare the performances of the listed extraction methods and various MGS fragmentation configurations without performing any video transmission, using four different sequences: Soccer, City, Bus and Harbour. Each of these sequences is encoded to have a base and an MGS layer (fragmented or not), having 289 frames in total (except for the Bus sequence that has 145 frames), with a GoP size of 16 frames. In order to observe the effects of slicing, we coded the videos both with byte-limited slice mode enabled (slices limited to 1400 bytes) and with slice mode disabled. We modified the JSVM software, version 9.19.3, to slice the MGS fragments properly in byte-limited mode. Without modifications, the byte-limited slice mode cannot be used when MGS fragmentation is performed.

After encoding, each video is extracted GoP-by-GoP with bandwidth constraints ranging from the rate of its base layer to that of its MGS layer, in 50 kbps increments. During an extraction, the same rate constraint is used for all the GoPs of the video and the extraction is done so that none of the extracted GoPs exceed the given constraint. After each extraction, the rate is incremented in 50 kbps and the process is repeated, until the maximum extractable video rate is met. Moreover, each extracted video is decoded and compared with the original sequence, to calculate its PSNR. Figure 2.1 and Figure 2.2 give the PSNR values of such extractions, averaged over a range of rate constraints for each video, which is from 200 kbps to 1200 kbps for the Soccer sequence, 200 kbps to 700 kbps for the City sequence, and 500 kbps to 2500 kbps for the Bus and Harbour sequences, all in 50 kbps increments. Figure 2.1 shows the average PSNR values for the slice mode disabled case, whereas the average PSNR values obtained when byte-limited slice mode is enabled is given in Figure 2.2. It is apparent from both of the figures that the proposed priority-based hierarchical extraction performs better than the other methods in most of the MGS configurations; while the flat-quality extraction, used by the JSVM software by default, performs the worst. However, the performance of flat-quality extraction improves as the MGS layer is fragmented into fewer sub-layers, and it results in nearly the same PSNR values with the priority-based hierarchical extraction at some configurations having the fewest MGS sub-layers. It is interesting to observe that the priority-based extraction results in lower PSNR values as the number of MGS sub-layers decreases, in contrast with the flat-quality extraction.

The MGS fragmentation configuration of (1,1,2,4,8) acts as a critical point in the results: There occurs a noticeable decrease in the average PSNR as the MGS layer is fragmented into more sub-layers at this configuration. Less number of MGS sub-layers results in similar PSNR values when priority-based hierarchical extraction is used, except for the City video with slice mode disabled. Actually, the City video is an interesting

sequence, hiding the penalty of MGS fragmentation, which performs better with more MGS sub-layers when the priority-based or priority-based hierarchical extraction methods are employed in the slice mode disabled case. Also, it is the only video showing that much of PSNR degradation for all of the extraction methods as the MGS layer is fragmented into fewer sub-layers, as seen in Figure 2.1.

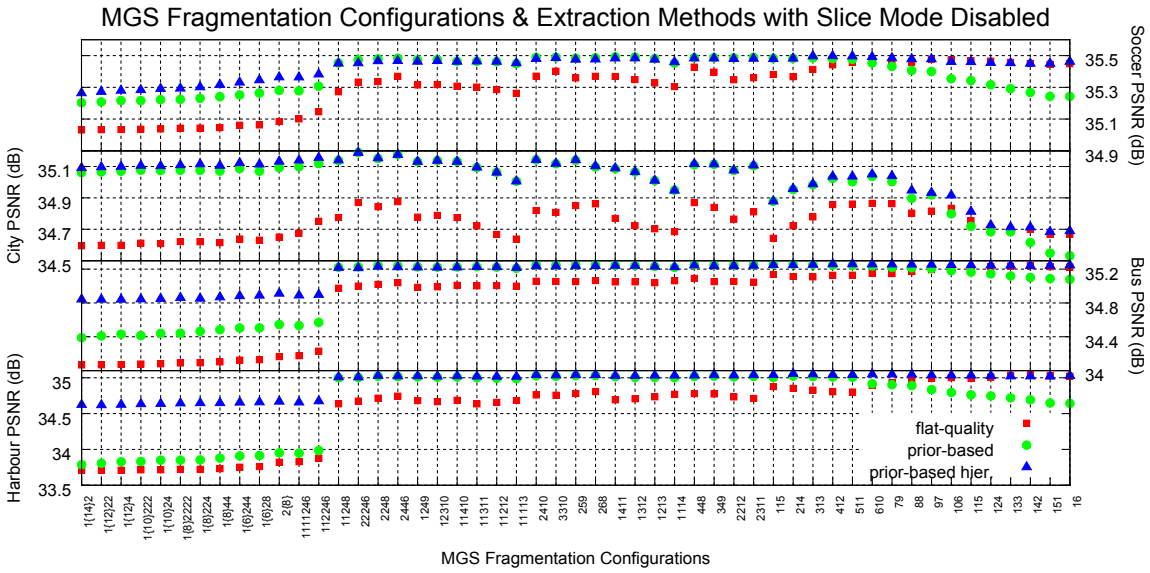


Figure 2.1 Average PSNR values of various MGS fragmentation configurations and extraction methods, when byte-limited slice mode is disabled. The x-axis shows the MGS configurations used to distribute the transform coefficients into MGS sub-layers, with decreasing n number of sub-layers from left to right, the last "16" value showing the non-fragmented case. The *n{m}* notation is employed for repetition, meaning that *n* is repeated for *m* times – 1{14}2 corresponds to 111111111111112.

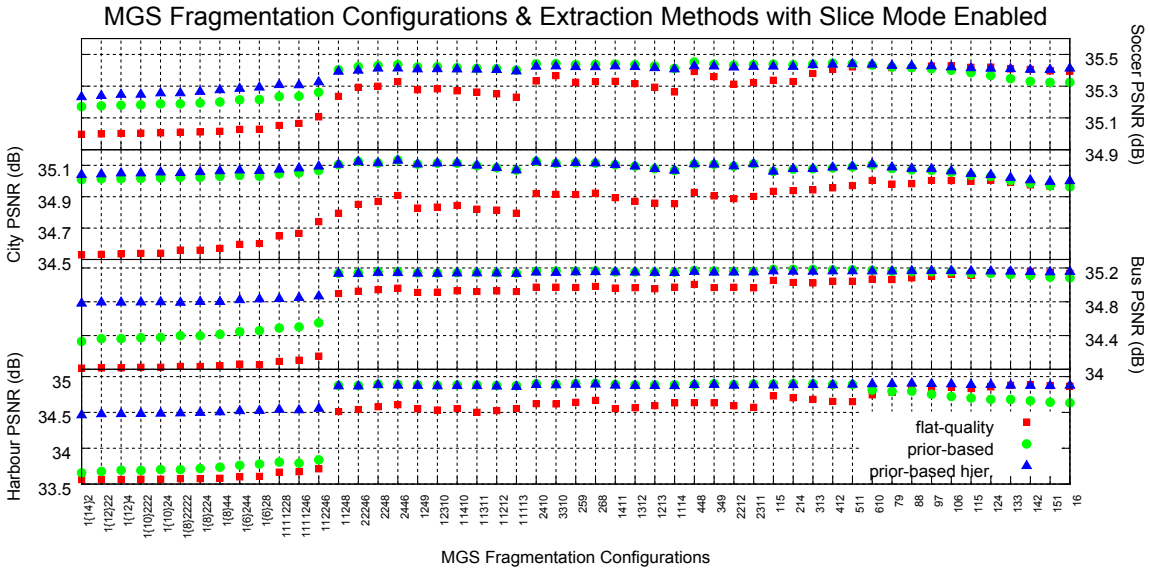


Figure 2.2 Average PSNR values of various MGS fragmentation configurations and extraction methods, when byte-limited slice mode is enabled.

The performance of the City sequence can be better observed in Figure 2.3, where the PSNR vs. bandwidth graphs for the slice mode disabled and enabled cases are compared to that of the Soccer sequence, generated using the priority-based hierarchical extraction. Different from Figure 2.1 and Figure 2.2, where the PSNR value of each extracted video is averaged over a range of bandwidth values to achieve a single average PSNR, Figure 2.3 shows the PSNR of each extraction that corresponds to a given bandwidth. Here, the poor extraction performance of the City sequence for the non-fragmented MGS configuration with slice mode disabled can be clearly seen. This performance decrease is due to the fact that the sizes of the key frame enhancement NAL units are too big to be extracted at low rates, since each layer is packetized in a single NAL unit. As these NAL units are fragmented to MGS sub-layers having smaller NAL units, the resulting PSNR values increase, because some NAL units belonging to these sub-layers can be extracted within the given bandwidth constraint. Hence, as the number of MGS fragments in the

configuration increases, the extraction performance improves at low bandwidth values. Similar performance gain can be achieved by enabling the slice mode, since large NAL units are split into smaller ones whose sizes do not exceed the given byte limit. Enabling slice mode for the City video increases the PSNR values at low rates, compared to the non-sliced case, except for the (1,1,1,1,1,1,1,1,1,1,1,1,1,1,2) configuration where most NAL units are already under the given byte limit without using slice mode.

MGS fragmentation has also a negative impact on the PSNR performance, due to the fragmentation overhead. This effect is outperformed at low rates by the PSNR gain achieved because of being able to select the small NAL units of the MGS sub-layers. At higher rates, as the rate budget gets big enough to select the higher priority NAL units, the fragmentation overhead starts to be observed. For the City and the Soccer sequence, the PSNR values of the (1,1,1,1,1,1,1,1,1,1,1,1,1,1,2) configuration is lower than the others after the bandwidth of 450 kbps. However, the MGS fragmentation penalty is much more observable for the Soccer video in Figure 2.3. This difference is caused by the fact that the sizes of the key frame enhancement NAL units of the Soccer sequence are smaller than that of the City sequence.

Enabling the byte-limited slice mode has a small overhead, as can be seen in Figure 2.4, when the NAL unit sizes are limited to 1400 bytes. However, this overhead will increase as the byte limit is lowered. Also, the unique performance of the City sequence can be observed in the figure, where employing slice mode increases the average PSNR values for the MGS configurations having fewer fragments, instead of introducing overhead.

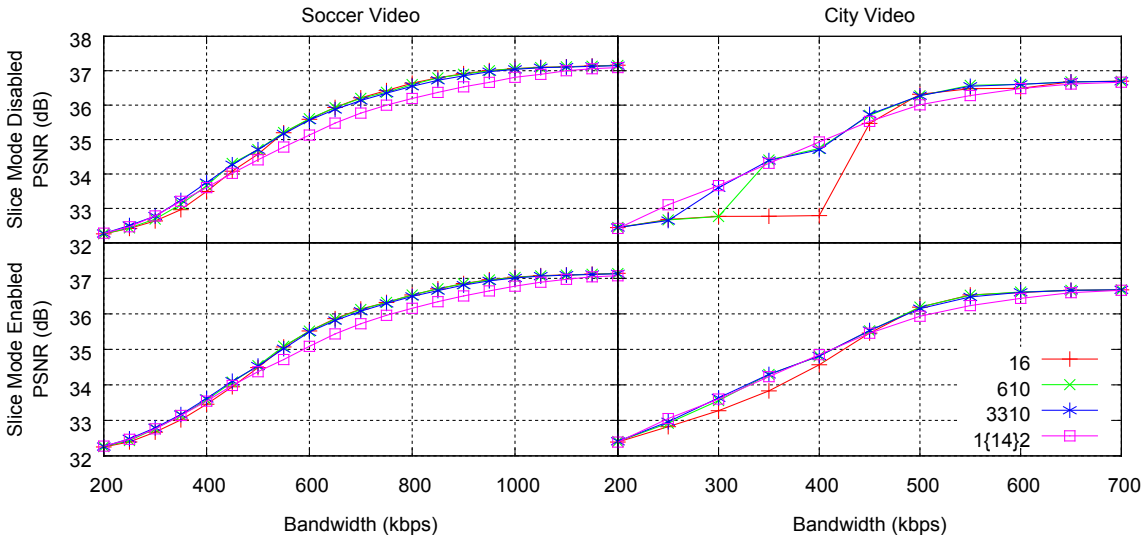


Figure 2.3 PSNR vs. bandwidth graphs for the Soccer and City videos, with four different MGS fragmentation configurations, when slice mode is disabled/enabled, generated using priority-based hierarchical extraction.

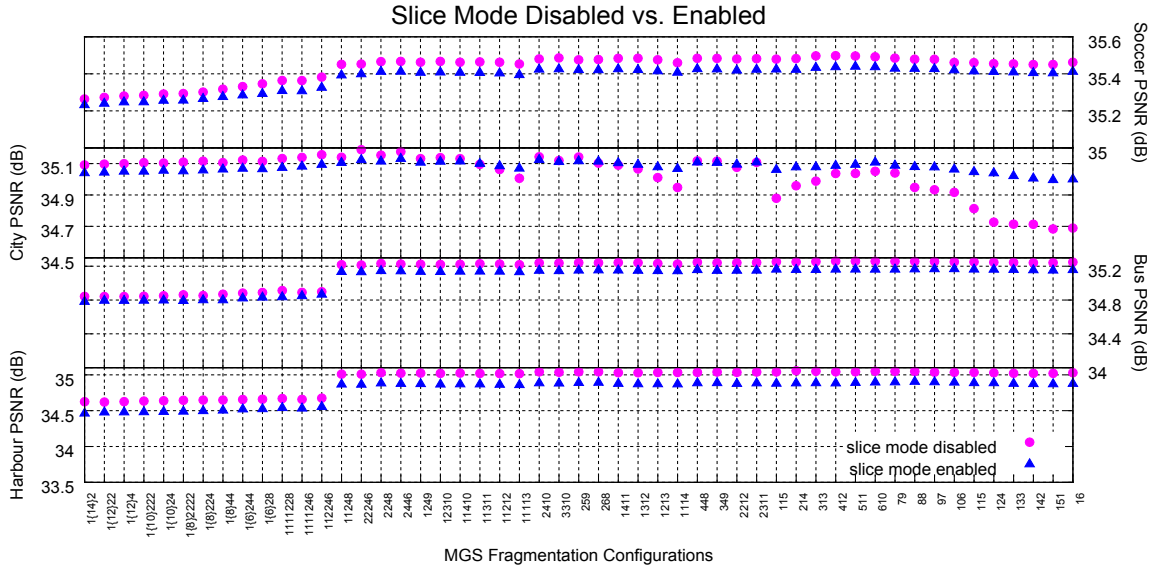


Figure 2.4 Average PSNR values of various MGS fragmentation configuration using the priority-based hierarchical extraction, when byte-limited slice mode is disabled/enabled. NAL unit sizes are limited to 1400 bytes, when slice mode is enabled.

The priority-based hierarchical extraction needs the priority information to be present in the stream that is to be extracted. In case no priority information is found in the stream, hierarchical extraction can be employed in place of priority-based hierarchical extraction, with minimal performance degradation as shown in Figure 2.5.

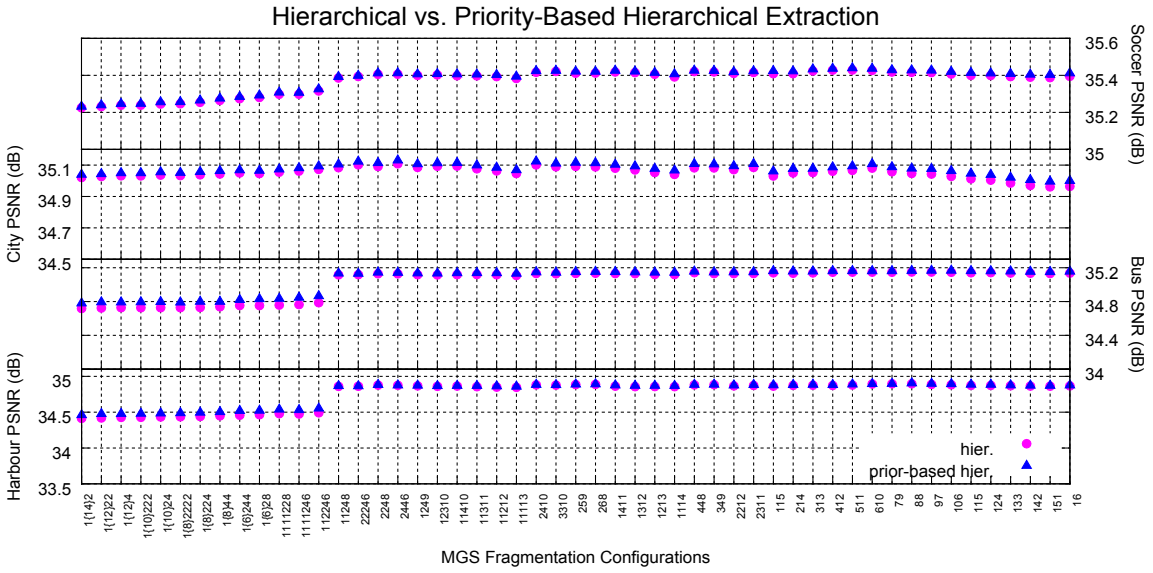


Figure 2.5 Comparison of the two extraction methods, hierarchical extraction and priority-based hierarchical extraction, when byte-limited slice mode is enabled. Hierarchical extraction can be utilized in case the priority information is not present in the video sequence.

PSNR vs. bandwidth curves of the extraction methods for the Soccer, City, Bus, and Harbour videos are given in Figure 2.6 – Figure 2.9, respectively. The figures clearly show the performances of the listed extraction methods over a range of bandwidth values for different MGS fragmentation configurations. Here, it is interesting to observe the PSNR values lower than that of the base layer, achieved by flat-quality extraction for the Bus and Harbour sequences when the MGS sub-layer count is high.

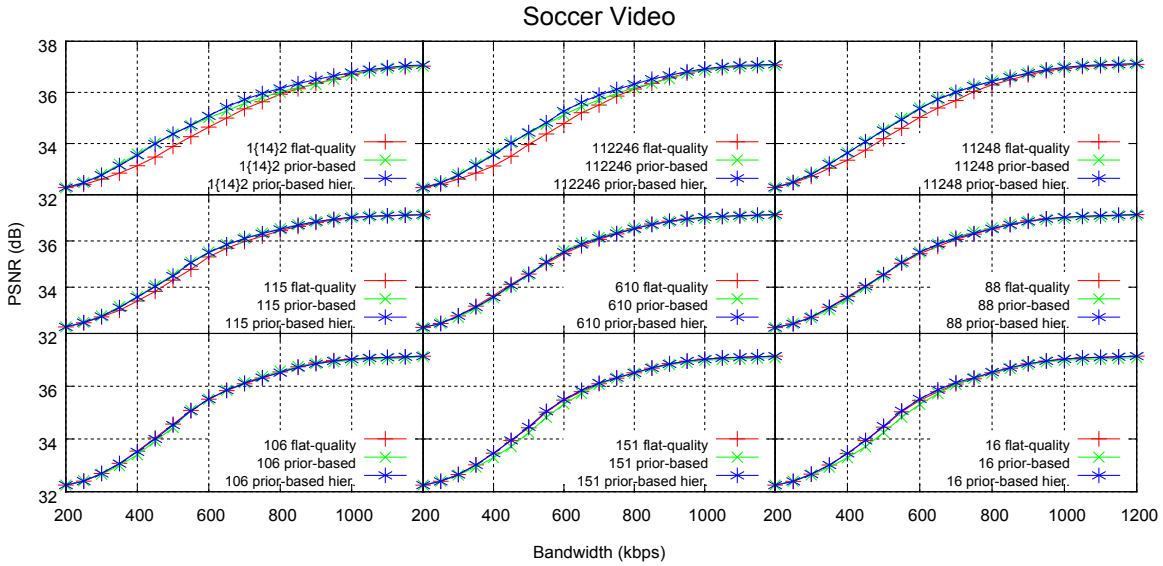


Figure 2.6 PSNR vs. bandwidth graphs of the extraction methods for the Soccer video at various MGS configurations, when byte-limited slice mode is enabled.

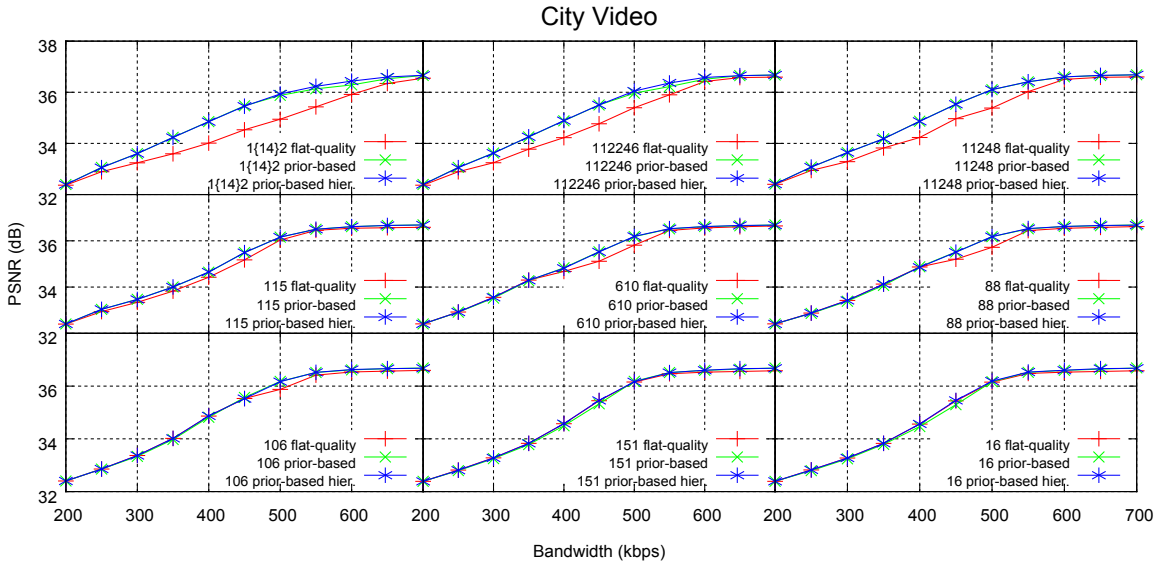


Figure 2.7 PSNR vs. bandwidth graphs of the extraction methods for the City video at various MGS configurations, when byte-limited slice mode is enabled.

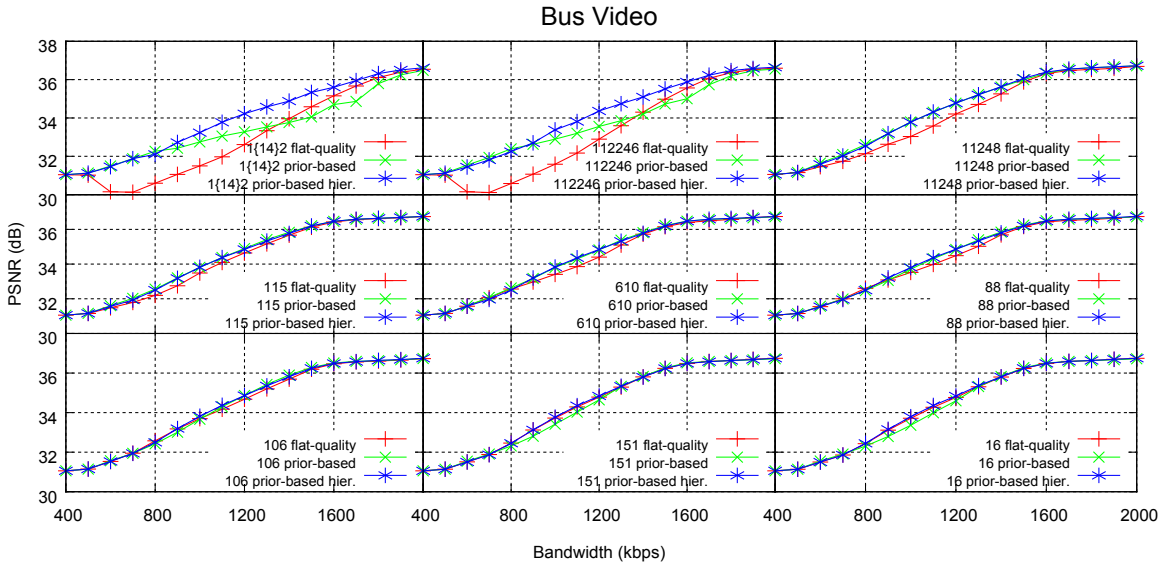


Figure 2.8 PSNR vs. bandwidth graphs of the extraction methods for the Bus video at various MGS configurations, when byte-limited slice mode is enabled.

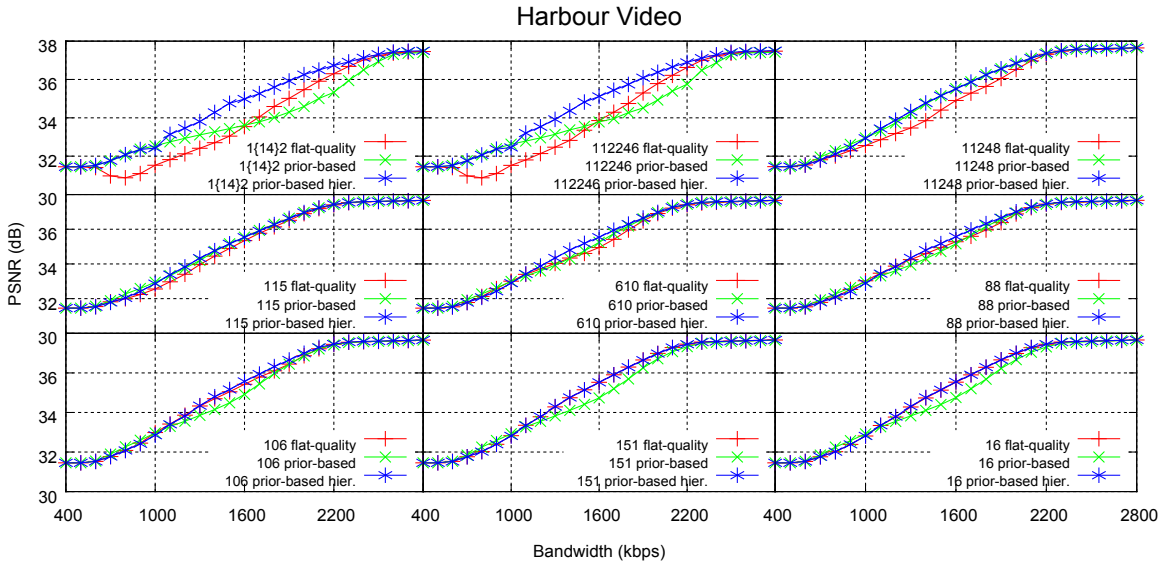


Figure 2.9 PSNR vs. bandwidth graphs of the extraction methods for the Harbour video at various MGS configurations, when byte-limited slice mode is enabled.

2.5 Conclusion

In this work, an extensive set of MGS fragmentation configurations of SVC are compared in terms of their PSNR performance, with the slice mode enabled or disabled, using several extraction methods. It is apparent from the results that the proposed priority-based hierarchical extraction performs better than the other methods in most of the MGS configurations; while the flat-quality extraction, used by the JSVM software by default, performs the worst. Also, results show that splitting the MGS layer into more than five fragments may result in noticeable decrease in the average PSNR.

For some videos whose key frame enhancement NAL units are relatively large, such as the City sequence, MGS fragmentation and/or slice mode have positive impact on the PSNR of the extracted video at low bitrates. While using slice mode without MGS fragmentation may address the PSNR performance problem at low rates, it may result in uneven video quality within frames due to varying quality of slices. Therefore, we recommend combined use of up to five MGS fragments and slice mode, especially for low bitrate video applications.

The results presented show that the performance of different encoding configurations and extraction methods may change with content. However, it is also apparent that a compromising solution is available that includes all of the videos tested, where the MGS layer is fragmented into two sub-layers with the fragmentation configuration of (6,10) and the byte-limited slice mode enabled (limit set as 1400 bytes). The priority-based hierarchical extraction method is employed for video rate adaptation in the solution.

Chapter 3

ADAPTATION STRATEGIES FOR SCALABLE VIDEO STREAMING

3.1 Introduction

This section introduces the adaptive video streaming system, implemented to test various adaptation strategies in streaming scalable video. First, key adaptation schemes are described and then corresponding test results are given. Tests are both run in a controlled LAN and on the actual Internet.

3.2 An adaptive video streaming system

We have implemented an adaptive video streaming system that works on IP networks and uses DCCP (both CCIDs) or TCP as the transport protocol, which is shown in Figure 3.1. The system, which consists of a sender and a receiver node, is adaptive in the sense that the rate – therefore the quality – of the video to be streamed is determined by the network conditions. The sender estimates the available network rate, extracts a GoP of video using the estimation and sends the extracted video packets to the receiver. As the packets arrive at the receiver, they are inserted to a decoder buffer large enough to filter out variations in the network and then decoded for display.

Parameters used by the streaming system are given in Table 3.1. The rest of the section covers the details of the video rate adaptation strategies utilized by the given system.

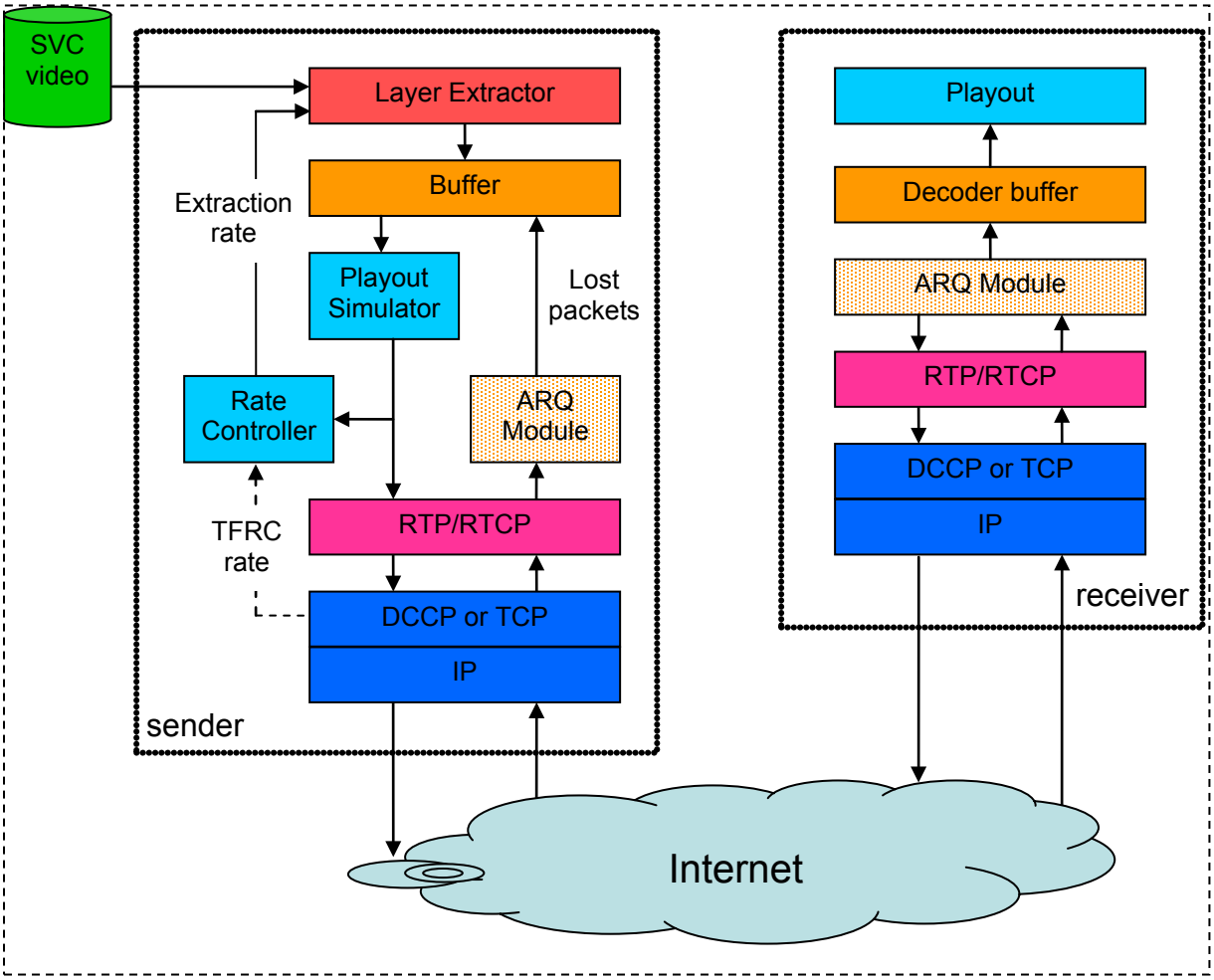


Figure 3.1 Block-diagram for a sender-driven adaptive video streaming system.

Table 3.1 Main parameters of the adaptive streaming system

Symbol	Parameter Description
r_{send}, R_{send}	The instantaneous and the average rate, respectively, at which the sender application deposits video packets to the transport protocol.
r_{resend}, R_{resend}	The instantaneous and the average rate, respectively, at which the sender application deposits lost video packets to the DCCP. Packet loss occurs only when DCCP is used for transport; hence retransmission rates are defined for DCCP, not for TCP. In case of TCP, these rates are zero.
$r_{extract}$, $R_{extract}$	The instantaneous and the average rate, respectively, at which the video data sent is extracted.
r_{tfrc}	The instantaneous TFRC rate calculated and reported by the DCCP module. In case of TCP, this rate is zero.
R_{tfrc}	The average of the instantaneous TFRC rate. In case of TCP, this rate is zero.
$T_{average}$	The length of the averaging window, in milliseconds.
T_{rc}	Time at when the rate controller measures rates, in milliseconds.
L_{play}	Playout start threshold, in GoPs.
$L_{overflow}$	Decoder buffer overflow threshold, in GoPs.

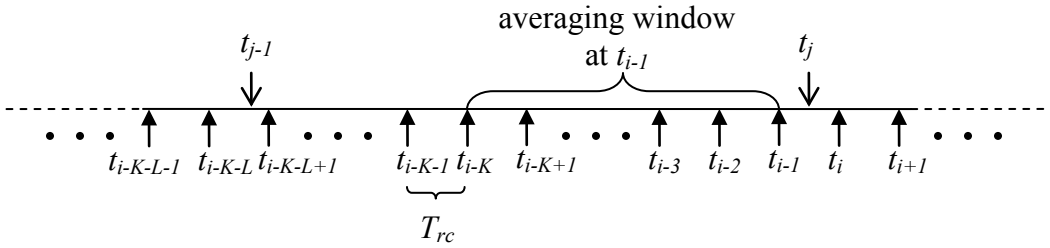
3.2.1 Determination of the video extraction rate

When DCCP CCID3 is utilized as the transport protocol, the DCCP module at the sender continuously calculates the TFRC rate using the feedbacks delivered by its peer at the receiver and employs this rate to transmit the packets in its queue. Apart from using it internally, the DCCP module can also expose the TFRC rate to requesting applications. Hence, a network adaptive application can utilize the TFRC to discover the network dynamics and adjust accordingly, without any implementation to estimate the current bandwidth. In our streaming system, the sender application periodically requests the TFRC rate from the DCCP and employs it to determine the quality of the video to transmit, in case DCCP CCID3 is used.

It is of interest to decide how to use the received TFRC rates in computing the rate of the scalable video to extract. Since network is changing instantaneously, it may be reasonable to average a series of TFRC rates over a time-window, to get rid of noise. The length of this time-window $T_{average}$ is a parameter that affects the performance of the streaming system: Longer the window, more resistant is the system to noise, but also less responsive to network dynamics.

TCP controls its transmission rate using window based congestion/flow control schemes but it does not expose any rate to its users in contrast to DCCP CCID3. The same also holds for DCCP CCID2, where TCP is mimicked. Therefore, the proposed system utilizes the packet deposit rate r_{send} averaged over a time-window with length $T_{average}$ to adapt the quality of the streamed video to the available bandwidth, unless DCCP CCID3 is used.

In our streaming system, the *rate controller* module in sender requests the TFRC rate r_{tfrc} from DCCP every T_{rc} milliseconds, when DCCP CCID3 is used. This module also measures packet deposit rates r_{send}, r_{resend} and the extraction rate of the packets deposited, $r_{extract}$ every T_{rc} milliseconds. The module then averages all these rates over a moving time-window of length $T_{average}$ and calculates the rate of the video to be extracted using (1)-(8), as shown in Figure 3.2.



..., t_{j-1}, t_j, t_{j+1}, ... : time when the next extraction rate is calculated
..., t_{i-1}, t_i, t_{i+1}, ... : time when rates are measured

Figure 3.2 Calculation of the next extraction rate

$$R_{send}(t_{i-1}) = \frac{1}{K}\sum_{i=1}^{K} r_{send}(t_{i-l}) \tag{1}$$

$$R_{resend}(t_{i-1}) = \begin{cases} \frac{1}{K}\sum_{i=1}^{K} r_{resend}(t_{i-l}) & \text{if DCCP CCID2 or CCID3 used} \\ 0 & \text{if TCP used} \end{cases} \tag{2}$$

$$R_{tfrc}(t_{i-1}) = \begin{cases} \frac{1}{K}\sum_{i=1}^{K} r_{tfrc}(t_{i-l}) & \text{if DCCP CCID3 used} \\ 0 & \text{if TCP or DCCP CCID2 used} \end{cases} \tag{3}$$

$$R_{extract}(t_{i-1}) = \frac{1}{K}\sum_{i=1}^{K} r_{extract}(t_{i-l}) \tag{4}$$

$$t_{i-1} - t_{i-K} \le T_{average} \qquad t_{i-1} - t_{i-K-1} > T_{average} \tag{5}$$

$$t_j - t_{i-K} < T_{rc} \tag{6}$$

$$R_{diff}(t_{i-1}) = R_{sent}(t_{i-l}) - R_{resent}(t_{i-l}) - R_{extract}(t_{i-l}) \tag{7}$$

$$R^{+}_{extract}(j) = \begin{cases} R_{tfrc}(t_{i-1}) + \min\left(R_{diff}(t_{i-1}),0\right) & \text{if DCCP CCID3 used} \\ R_{sent}(t_{i-1}) + \min\left(R_{diff}(t_{i-1}),0\right) & \text{if TCP or DCCP CCID2 used} \end{cases} \tag{8}$$

The equations in (1)-(4) average K instantaneous samples with equal weight. The number of samples to average, denoted by K, is determined by the averaging-window length $T_{average}$, as in (5). Moreover, (6) ensures that averaging starts with most recent samples. Then, the averaged rates are used to calculate the next extraction rate, as given by (7) and (8).

Given an extraction rate $R^{+}_{extract}$, the *layer extractor* module extracts a GoP of video satisfying the rate constraint, whose NAL units are inserted to the sender buffer to be kept until sent to the transport protocol queue. As the NAL units are fetched from the buffer, packetized and sent, the buffer starts to get empty. When the number of NAL units in the sender buffer decreases below QS_{min}, a new extraction rate is computed and fed to the layer extractor for a new GoP of video. The value of QS_{min} is selected such that it is small enough for using the most recent bandwidth information in computing the extraction rate $R^{+}_{extract}$. But, QS_{min} should also be big enough that sending QS_{min} NAL units take longer than the time taken by each extraction process and therefore the buffer gets filled with new extracted NAL units before being drained.

3.2.2 Sending rate control

During a video streaming session, the sender transmits packetized video data to the receiver. The actual sending operation is implemented by depositing video packets to the transport protocol queue at a rate of r_{send}. The transport protocol module at the sender transmits the packets in its queue to its peer at the receiver with a rate determined by the protocol. In case of DCCP CCID3, this rate is the calculated TFRC rate. If the sender application inserts packets to the transport protocol queue faster than the rate at which the packets in the queue are processed, the queue may become full due to being limited. In this case, sender will wait for the queue to become available, meaning that the fast sender will be slowed down by the transport protocol.

The sender cannot directly control the rate at which the transport protocol sends packets in its queue to the receiver. More specifically, the sender cannot send packets faster than the transport protocol allows. However, it can slow down the transmission by dispatching packets to the transport protocol queue at a lower rate than the allowed rate. This may be meaningful especially when the available bandwidth is more than the highest extraction

rate of the video streamed. It may be considered that streaming video at the extracted rate rather than the higher available bandwidth would result in lower loss, leaving more bandwidth to competing flows. In order to investigate the results of sending rate control, we have implemented three selectable modes of rate control, which are: i.) sending the extracted video at the available bandwidth rate (*send @ max. rate*); ii.) sending video at the extraction rate (*send @ ext. rate*); iii.) sending video at the available bandwidth rate but not more than the maximum extractable video rate (*send @ lim. max. rate*).

Moreover, the sender may have to slow itself down to prevent any receiver buffer overflow. In our streaming system, the *playout simulator* module estimates the receiver side decoder buffer status depending on the packets transmitted as well as the decoder buffer feedback sent by the receiver. Then, by using this estimate, the rate controller decreases packet dispatch rate in case of a decoder buffer overflow risk.

3.2.3 Retransmission of lost packets

Although video rate is adapted to the network, packet losses occur when DCCP is used. As the receiver detects missing packets with enough time for playout, it may request these from the sender using an ARQ scheme. Three different ARQ schemes exist in the streaming system, which are: i.) request all missing packets (*arq all*); ii.) request base layer missing packets only (*arq base*); iii.) request all or base layer missing packets only, depending on the decoder buffer occupancy (*arq adaptive*). In the first scheme (arq all), all of the missing packets are requested for retransmission, provided that their playout times have not expired. In the second scheme (arq base), only the base layer packets, which are mandatory for the error-free video decoding, are requested by the receiver. The third scheme makes this decision adaptively, based on the decoder buffer occupancy. If there is a decoder buffer overflow risk, all the missing packets are requested from sender. Otherwise, only base layer packets are asked for retransmission. Hence, when the bottleneck

bandwidth is more than the maximum extractable video rate, the remaining bandwidth is utilized for achieving the maximum received video quality. At other times, the available bandwidth is used for transmitting the mandatory data.

3.2.4 Number of GoPs extracted at a time

The implemented adaptive video streaming system extracts a GoP of video at a time by default. However, this can be increased to two, three, or more GoPs that are extracted together, using the same extraction rate. Increasing the number of GoPs extracted at a time introduces an extra averaging, which may smooth out the rate differences between consecutive GoPs. It should also be noted that increasing the extraction GoP size may cause the system to be less responsive to rapid changes in the network.

3.2.5 Decoder buffer size

The decoder buffer size is given in terms of GoP count, since the decoding is performed GoP by GoP in the proposed system. The default buffer size is set as 20 GoPs, which corresponds to 10 seconds of video approximately, for a 30 fps video with a GoP size of 16 frames. However, the size of the decoder buffer is not fixed; it can be altered depending on the preferences. The size of the decoder buffer determines the pre-buffering period, since the video playout starts as soon as half of the buffer is full. Also, the decoder buffer overflow threshold $L_{overflow}$, which is used to prevent decoder buffer overflow, changes with the buffer size.

3.2.6 Packetization

When DCCP is used, the extracted video data should be packetized prior to being sent. Maximum size of a DCCP packet payload, which is a NAL unit, is given by the path Maximum Transmission Unit (MTU) minus the packet header introduced. For example, if

the MTU is 1500 bytes, the maximum possible length of a NAL unit will be 1452 bytes, which is MTU (1500 bytes) minus RTP+DCCP+IP headers (12+16+20 bytes). In order to minimize the effect of packet loss on video quality, the video to be streamed is coded in slices so that each NAL unit can fit to a DCCP packet. Hence, each NAL unit is sent in a single DCCP packet, without being fragmented. Moreover, smaller NAL units are aggregated in decoding order to fit into a packet, for minimizing the packet header overhead. RTP packetization is done as per [34]. Moreover, the same packetization scheme is also used for TCP. In order to send the packetized NAL data over TCP, the packets are framed according to [35].

3.3 Results

This section covers video streaming tests performed using various transmission schemes and system parameters. The Soccer and City sequences at CIF resolution and Soccer at 4CIF resolution are utilized throughout the tests – Soccer at 4CIF is used only in the Internet tests. Each video is looped 10 times, so as to have a 2881 frame sequence longing for about 96 seconds. The looped sequences are encoded with JSVM version 9.19.3, to have a base and an MGS layer, with the quantization parameter equal to (38,28) for the Soccer video and (36,26) for the City video. The MGS layer is further divided into two sub-layers, using the MGS fragmentation configuration of (6,10), as per the results presented in Chapter 2. The GoP size is set as 16 frames, and the first frame of each GoP is coded as key picture so that error drift is constrained to a single GoP. Moreover, the each frame in the Soccer and City sequences is partitioned into multiple slices using the byte-limited slice mode (limit set as 1400 bytes) so that the maximum NAL unit length is less than the default MTU value of 1500 bytes, together with the RTP, DCCP and IP headers, guaranteeing none of the NAL units will be fragmented in the IP layer. For adapting the

video rate to the available network bandwidth, the priority-based hierarchical extraction scheme is utilized, depending on the results presented in Chapter 2.

The maximum extraction rates and the corresponding PSNR values over the whole video are given in Figure 3.3 for the Soccer and City sequences, achieved using the priority-based hierarchical extraction. It can be noted from the figure that the rate and PSNR values vary much more during the Soccer sequence, compared to the City video. This is because the Soccer video contains more movement and scene change than the City video, which is more or less static. Additionally, the minimum (base layer only) and the maximum (base + MGS enhancement) extractable rates (without packetization) of the sequences are listed in Table 3.2, together with the corresponding PSNR values. The videos can be extracted at almost any rate between these minimum and maximum values, thanks to the MGS fragmentation and multiple slices. It should be noted here that the values given in Table 3.2 are averaged over the whole sequence, while the ones given in Figure 3.3 are averaged over GoPs in each sequence.

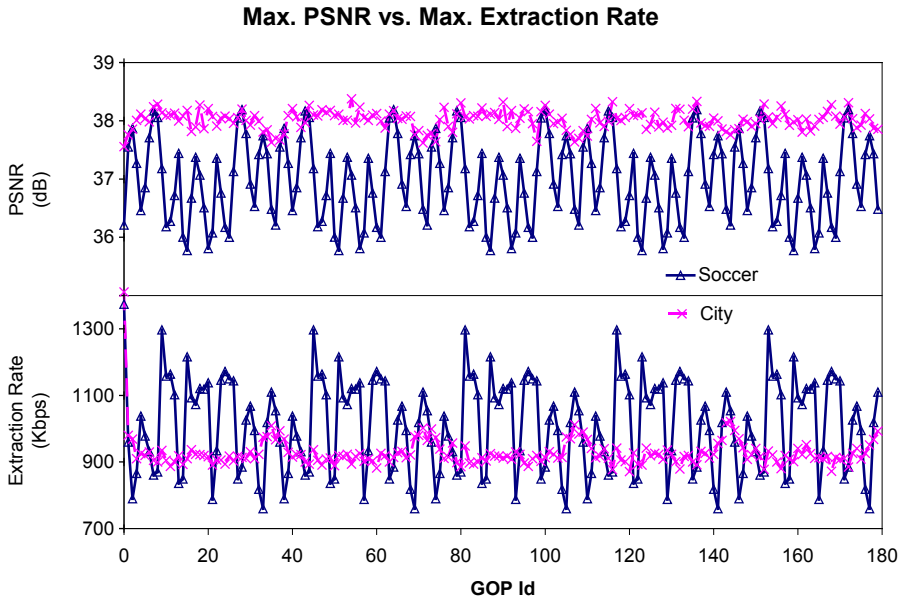


Figure 3.3 Maximum extraction rates of the videos and the corresponding PSNR values, using the priority-based hierarchical extraction.

Table 3.2 Min./max. average extraction rates (without packetization) and PSNR values

		Rate (kbps)	PSNR (dB)
Soccer	**Min**	316	32.20
	Max	945	36.98
City	**Min**	320	33.54
	Max	855	38.01

The tests are first run in a controlled LAN, in which the Internet traffic is simulated using the network simulator *ns2*. ns2 version 2.33 is used in emulation mode so that packets from sender to receiver pass through ns2, being subject to loss or delay. A setup of three nodes is used in the tests: one node running the sender application, one running the receiver application and the remaining node running ns2. The ns2 node is connected to both the sender and the receiver nodes, passing traffic from sender to receiver and vice versa, as shown in Figure 3.4. For cross traffic the PackMime-HTTP [36] module is utilized, generating web traffic, with the rate parameter set to 1, 5, and 10 HTTP requests per second for low, moderate, and high cross traffic, respectively. Tests are done with varying parameters, at different bottleneck bandwidths ranging from 500 kbps to 1500 kbps with a step size of 100 kbps (500, 600, …, 1500 kbps). For each set of parameters, 15 streaming tests are performed at each bottleneck bandwidth. The results shown in the figures are the averages of these 15 tests.

After tests in the controlled LAN, new tests are performed on the actual Internet, between the Koç University and Argela Technologies, both located in Istanbul. The setup for the Internet tests is given in Figure 3.5. The sender node resides in the Koç University, whereas the receiver is located in the Argela Technologies. The Internet speed between these two nodes is about 5 Mbps. During Internet tests, 15 tests are run for each test set, and the results are averaged prior to presentation.

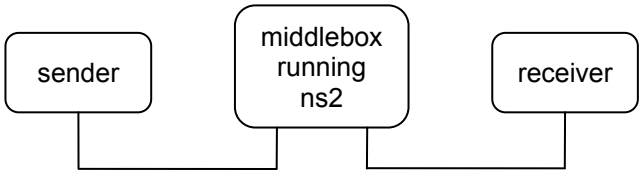


Figure 3.4 Controlled LAN test setup

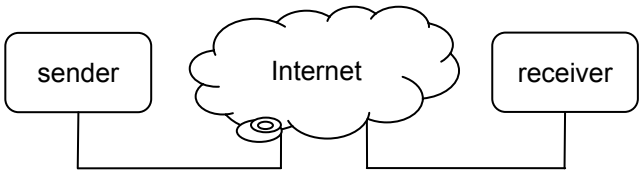


Figure 3.5 Internet test setup

As the DCCP implementation at the sender and receiver nodes, Linux with kernel version 2.6.35-rc1, which is pulled from Gerrit Renker's tree [37], is utilized.

3.3.1 Determination of the video extraction rate

Tests have been performed with two different averaging window lengths (1 and 5 seconds) using various transport protocols and their performances are compared with the results found without any averaging. Tests done using DCCP CCID3 show that averaging the calculated TFRC rate over a window improves the quality of the received video. Increasing the averaging window length from 0 (no averaging) to 1 second enhances the video quality by approximately 0.3 dB for the Soccer video and 0.9 dB for the City video at maximum, as given in Figure 3.6. Further increasing the window length to 5 seconds does not result in a significant difference, except for the high traffic case at 500 kbps when the City video is used, where extending the averaging window length to 5 seconds improves the video quality by 0.7 db by smoothing out TFRC rate oscillations.

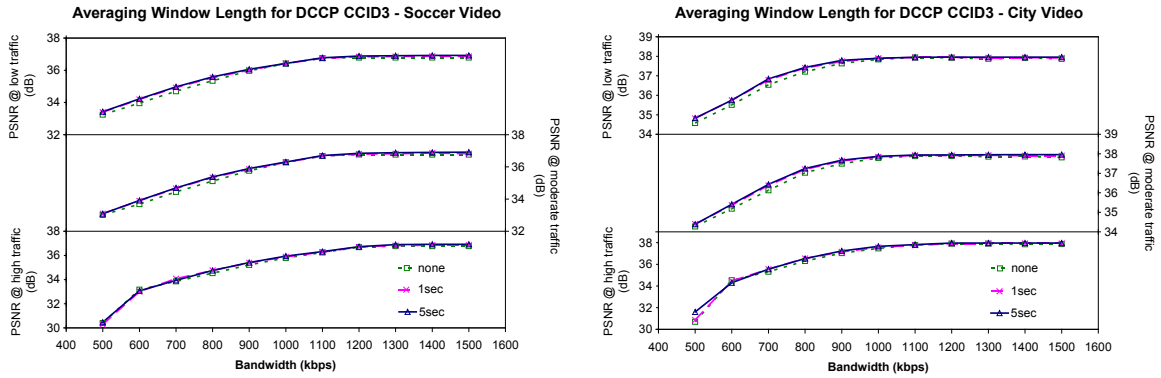


Figure 3.6 PSNR values for different averaging window lengths under varying cross traffic with DCCP CCID3

Averaging improves the received video quality much more than CCID3, when DCCP CCID2 is used, except for the highest congestion case (high traffic at 500 kbps). Figure 3.7 shows that the gain earned by performing averaging over a 1 second window is about 3 dB for the Soccer video and 2.5 dB for the City video at maximum, when compared with the no-averaging results. This significant difference is due to the fact that the packet sending rates are averaged and used for extraction in CCID2, as opposed to the TFRC rates in CCID3. Since the video packets are dispatched to the network in a discrete approach, measuring the sending rate instantaneously results in a noisy rate, which decreases the video quality when used for video extraction. High traffic case at 500 kbps is an exception to this, where only base layer video is sent because averaging is not applied and that is why the target extraction rate is mostly zero. This tells us that sending video at the base layer without any adaptation results in the best performance when the network is highly congested.

Increasing the averaging window length from 1 second to 5 seconds decreases the received video quality at low bandwidths: the difference is as much as 0.7 dB for the Soccer video and 0.5 dB for the City sequence at 500 kbps low traffic scenario. The quality

difference between these averaging lengths diminishes as the bottleneck bandwidth increase or more traffic is introduced into the network.

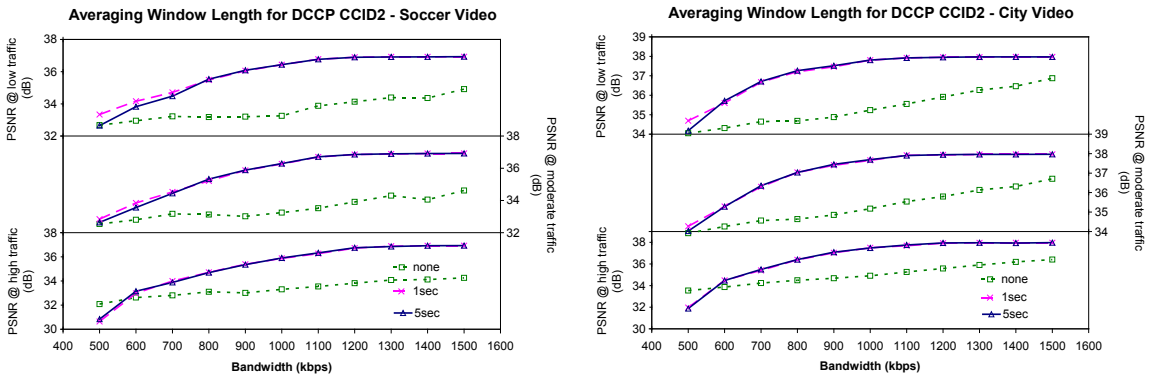


Figure 3.7 PSNR values for different averaging window lengths under varying cross traffic with DCCP CCID2

Packet send rate is used for calculating the video extraction rate also when TCP Reno is used as the transport protocol, as done with DCCP CCID2. Therefore, applying averaging over the packet send rate improves the received video quality significantly, as can be noted in Figure 3.8. The quality difference between the no-averaging case and 1 second averaging is as much as 14 dB for the Soccer sequence and 10 dB for the City sequence when the bottleneck bandwidth is low. Here, not averaging the packet send rate does not produce higher quality videos when the network is highly congested, as experienced with DCCP CCID2, because TCP sends packets in a more busty scheme.

As the bandwidth increases, the quality difference between the no-averaging and 1 second averaging scenarios drops down to approximately 2 dB. Extending the averaging window length to 5 seconds improves the results slightly at most cases, whereas there are also times when unnoticeable decreases are experienced.

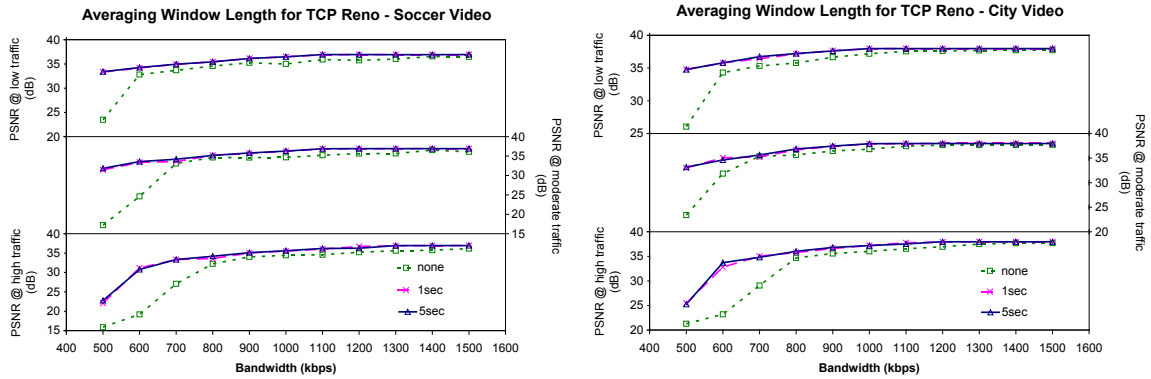


Figure 3.8 PSNR values for different averaging window lengths under varying cross traffic with TCP Reno

Results given in the above figures determine that the best performing averaging window length values are 5 seconds for the DCCP CCID3 and TCP Reno, while it is 1 second for the DCCP CCID2. These are the default values used for the averaging window length throughout the tests, unless otherwise stated.

3.3.2 Sending rate control

Tests have been performed under varying bottleneck bandwidth and cross traffic, using the three rate control schemes: i.) send @ max. rate; ii.) send @ ext. rate; iii.) send @ lim. max. rate. Results given in Figure 3.9 and Figure 3.10 reveal that sending video at the maximum or limited maximum rate produces videos with similar quality, whereas sending at the extraction rate generates lower quality videos, especially when there is cross traffic. Additionally, Figure 3.9 shows that limiting the video transmission rate (send @ lim. max. rate) results in less packet loss and therefore less retransmission traffic when the available bottleneck bandwidth is more than the maximum extractable video rate.

When DCCP CCID3 is used with no cross traffic, the three rate control schemes show similar performance up to the bottleneck bandwidth of 800 kbps for the Soccer video, as seen in Figure 3.9. At this rate, sending video at the extraction rate starts performing worse

than the other two schemes: The PSNR difference is about 0.7 dB at 800 kbps and 0.9 dB at 900 kbps. The reason of this quality decrease can be observed Figure 3.11, where the TFRC rate calculated by DCCP varies between 400 kbps and 1500 kbps when sending rate is kept at the video extraction rate; compared to the variation between 700 kbps and 900 kbps when video is sent at the limited maximum available rate. Keeping the transmission rate at the video extraction rate results in TFRC rate to oscillate. This is due to the nature of TFRC rate calculation: Packet loss is used to estimate the available bandwidth, along with RTT, and when a source transmits data at a rate lower than the calculated TFRC rate, less number of packets gets lost and DCCP overshoots the bandwidth estimate. Then the sender node in our streaming system uses this overshot estimate to extract new packets, many of which in turn will be lost. Hence this time DCCP will lower the TFRC rate, resulting in oscillations.

The oscillating behavior of TFRC starts to be experienced at bottleneck bandwidth value of 800 kbps for the Soccer video. Before that, the video is extracted at the bottleneck rate; hence there is no difference between sending at the extraction rate or the available rate. At 800 kbps, some parts of the video will be extracted below the available bandwidth, because their maximum extraction rates will be below 800 kbps, as shown in Figure 3.3. The same holds for 900 kbps, 1000 kbps and 1100 kbps, where there is difference between the extraction rate and the available bandwidth. However, after 1200 kbps, although the extraction rate differs from the bottleneck bandwidth, the three sending rate control schemes start to give similar results, because the bandwidth is over-provisioned and the oscillating behavior of TFRC is compensated.

City video results show that the performance of sending rate control schemes differ based on content. When there is no cross traffic, the three schemes perform similarly in terms of received video quality for the City sequence, as seen in Figure 3.9. Moreover, starting with the bottleneck bandwidth of 900 kbps, which is approximately the maximum

extractable rate of the City video, sending at the extraction rate results in less retransmission traffic than the other rate control schemes. After 1000 kbps, sending at the extraction rate results in no packet loss at all, while the other schemes, especially sending at the maximum available rate causes some packets to be dropped. However, as cross traffic is introduced into the network, sending at the extraction rate starts to perform worse than the other two schemes, similar to what is observed with the Soccer video.

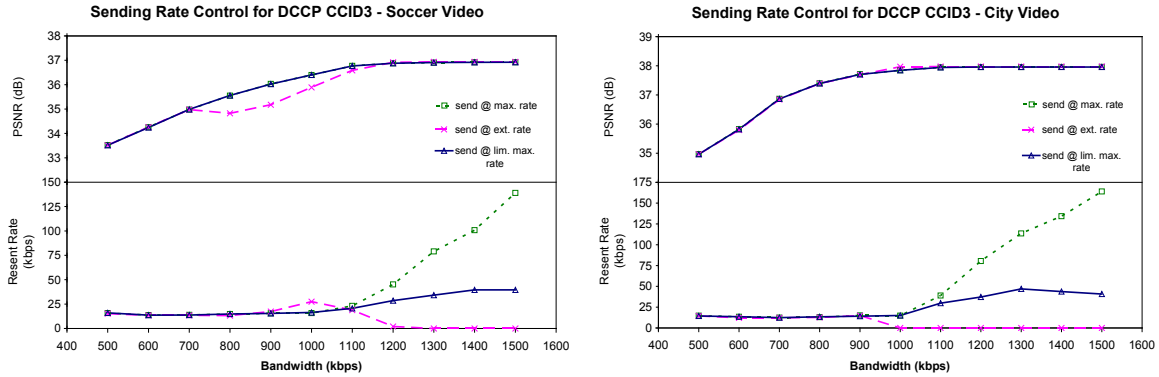


Figure 3.9 PSNR and resent rate values for the three sending rate control schemes using DCCP CCID3 without cross traffic

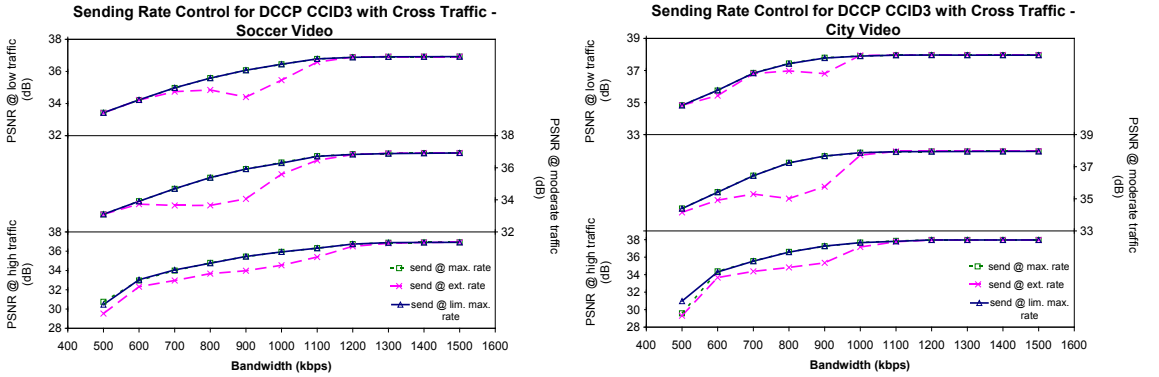


Figure 3.10 PSNR values for the three sending rate control schemes using DCCP CCID3 under varying cross traffic

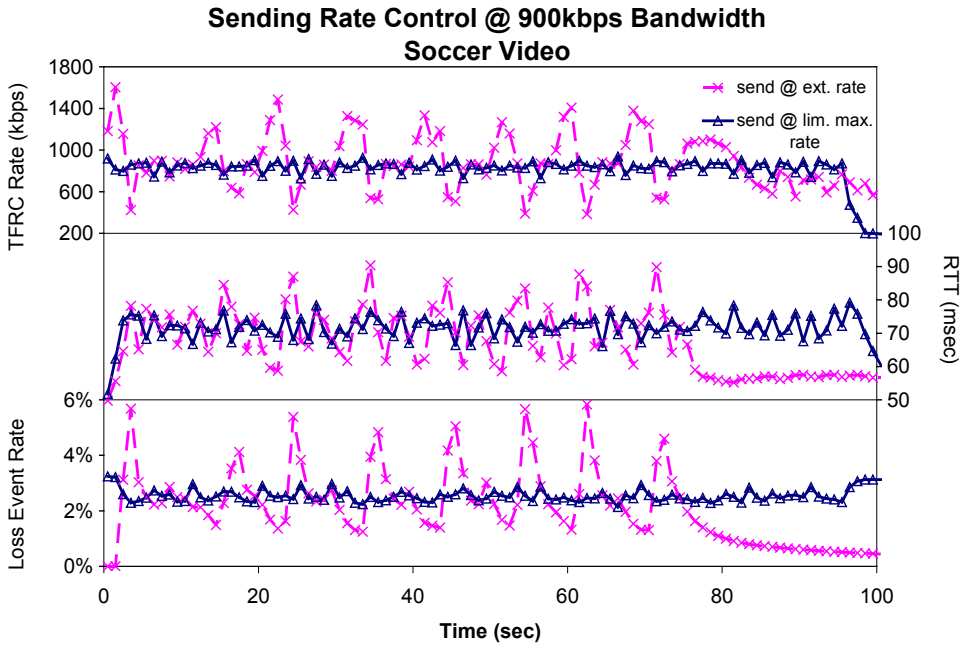


Figure 3.11 Soccer video TFRC, RTT and loss event rate values for the two sending rate control schemes at 900 kbps bottleneck bandwidth using DCCP CCID3

Tests performed with DCCP CCID2 and TCP Reno show that controlling the transmission rate results in poorer performance, as given by Figure 3.12 and Figure 3.13. This is due to the fact that the extraction rate is calculated depending on the transmission rate, when DCCP CCID2 and TCP Reno are used. Sending video at the extraction rate causes a bursty transmission pattern, decreasing the calculated extraction rate and hence the extracted video quality.

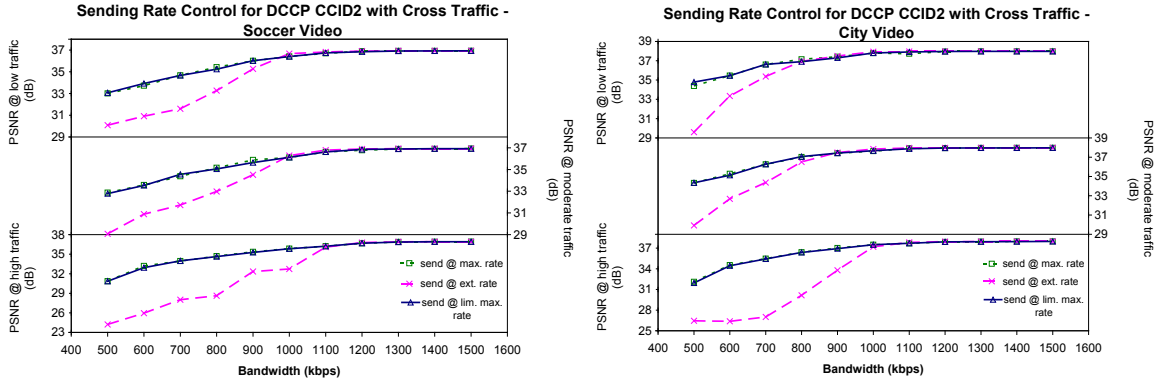


Figure 3.12 PSNR values for the three sending rate control schemes using DCCP CCID2 under varying cross traffic

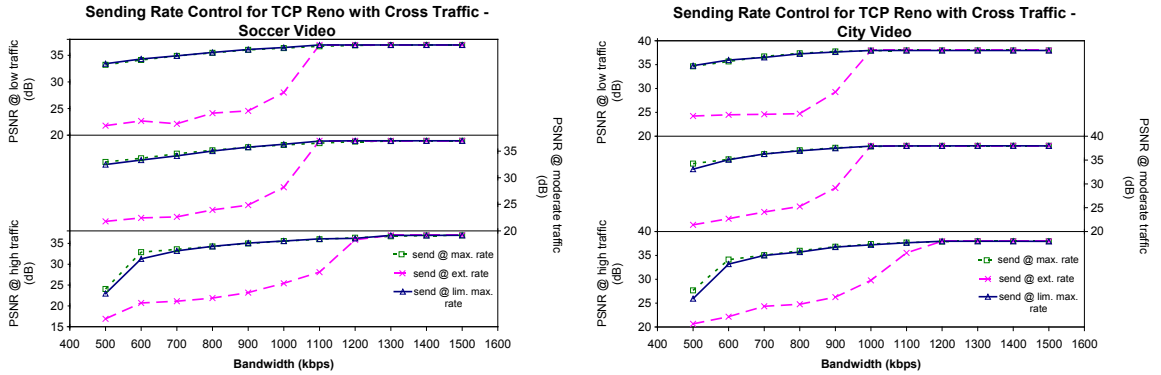


Figure 3.13 PSNR values for the three sending rate control schemes using TCP Reno under varying cross traffic

3.3.3 Retransmission of lost packets

Performances of the existing ARQ schemes using DCCP CCID3 are shown in Figure 3.14 and Figure 3.15, without any cross traffic and under high cross traffic, respectively. The results reveal that resending all missing packets does not produce a noticeable increase in the received video quality when the bandwidth is scarce. Under cross traffic, resending all packets even causes minor quality decrease, despite the increased retransmission traffic. In case the available network rate is high, however, the highest possible video quality cannot be achieved by requesting only base layer missing packets, because of the lost

enhancement layer packets. At this point, the proposed adaptive ARQ scheme aims to scale the retransmission traffic, requesting only base layer packets when the network rate is low and requesting all missing packets when the network allows. In this way, the unnecessary retransmission traffic can be avoided by 40% at best and the maximum video quality can be reached.

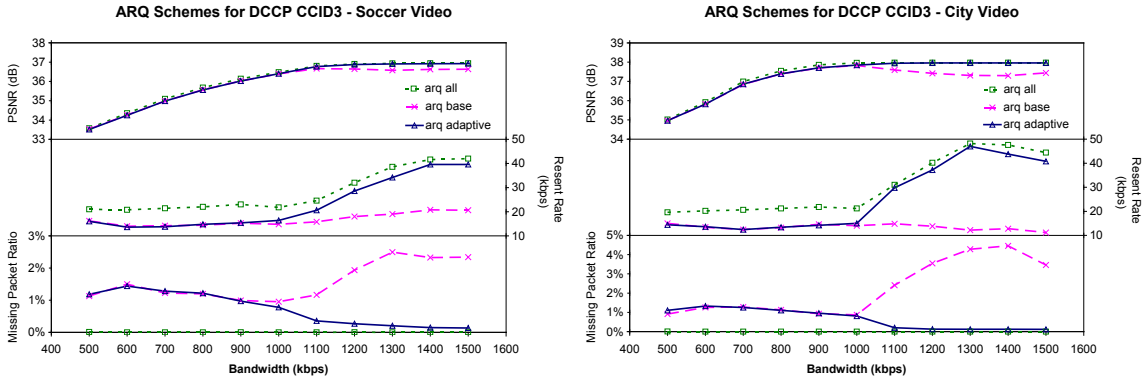


Figure 3.14 Results for different ARQ schemes with DCCP CCID3

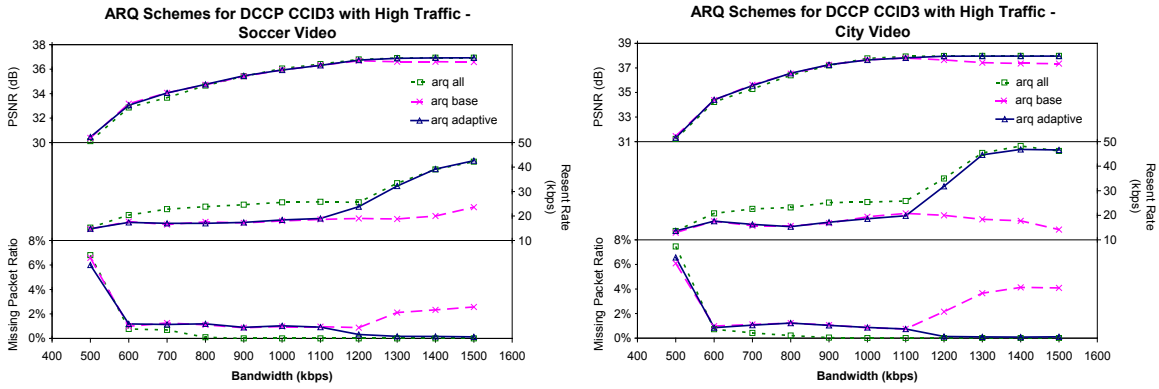


Figure 3.15 Results for different ARQ schemes under high cross traffic with DCCP CCID3

Existing ARQ schemes performs more or less the same in terms of received video quality, when DCCP CCID2 is utilized. Inability to reach to the maximum possible video

quality, experienced with CCID3, is not seen with CCID2, due to the difference between missing packet ratios, especially at high bandwidth. When DCCP CCID3 is used and only base layer missing packets are retransmitted, the missing packet ratio can be as high as 4%. However, this value is about 1% for DCCP CCID2 at high network rates, which does not result in a noticeable difference in video quality. This difference in missing packet ratios is due to CCID2's AIMD behavior, which acts faster to changes in network conditions than CCID3's TFRC scheme.

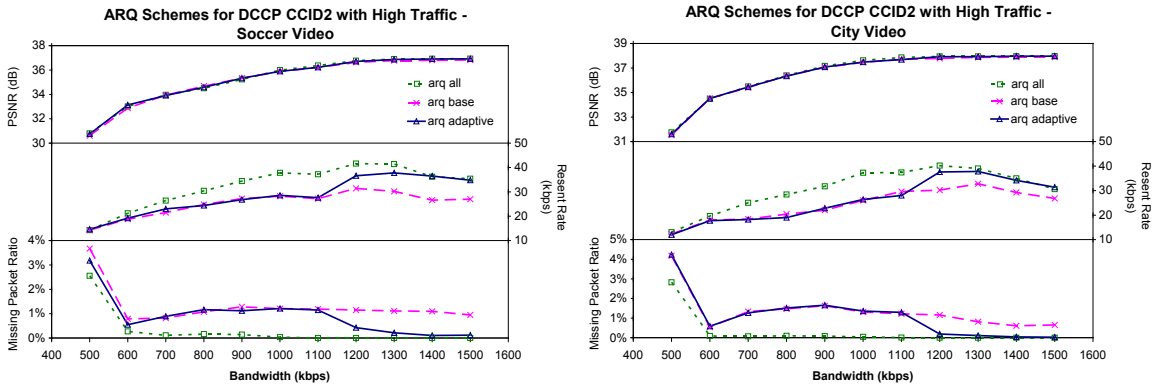


Figure 3.16 Results for different ARQ schemes under high cross traffic with DCCP CCID2

3.3.4 Number of GoPs extracted at a time

Figure 3.17 gives the received video quality values for different number of GoPs extracted at a time: one GoP, two GoPs and five GoPs. It is apparent from the figure that one GoP and two GoPs perform similar, while minimal quality decrease is experienced as the video extraction size is increased to five GoPs (approx. 2.7 secs), due to the decreased adaptability of the system. Hence, extraction size of one GoP is a reasonable value for adaptation to the available network rate. Similar results are achieved with DCCP CCID2 and TCP Reno, which are not shown here.

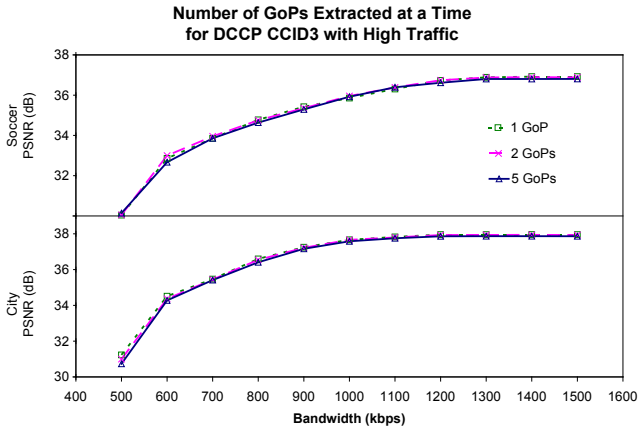


Figure 3.17 PSNR values for different number of GoPs extracted at a time under high cross traffic with DCCP CCID3

3.3.5 Decoder buffer size

Tests have been done with three different decoder buffer sizes: 40 GoPs , 20 GoPs, and 10 GoPs. Table 3.3 gives the buffer sizes and the corresponding $L_{overflow}$ and L_{play} threshold values, in terms of GoPs and seconds.

Table 3.3 Different decoder buffer size settings, given in terms of GoPs and seconds. Time values are calculated for a 30 fps video with a GoP size of 16 frames.

	GoP	**sec**	**GoP**	**sec**	**GoP**	**sec**
Buffer Size	40	21.3	20	10.7	10	5.3
$L_{overflow}$	30	16	15	8	7	3.7
L_{play}	20	10.7	10	5.3	5	2.7

Figure 3.18 gives the quality of the received videos and the length of the pre-buffering periods for different decoder buffer size settings. It is apparent from the figure that the default buffer size of 20 GoPs is a good compromise between the video quality and the pre-buffering period, which is 5.2 seconds in average – for the 96 seconds video. Buffer size of 40 GoPs results in a rather long pre-buffering period of 9.6 seconds in average; whereas buffer size of 10 GoPs produces low quality videos especially at low bandwidth values, while the pre-buffering period is as low as 2.6 seconds in average.

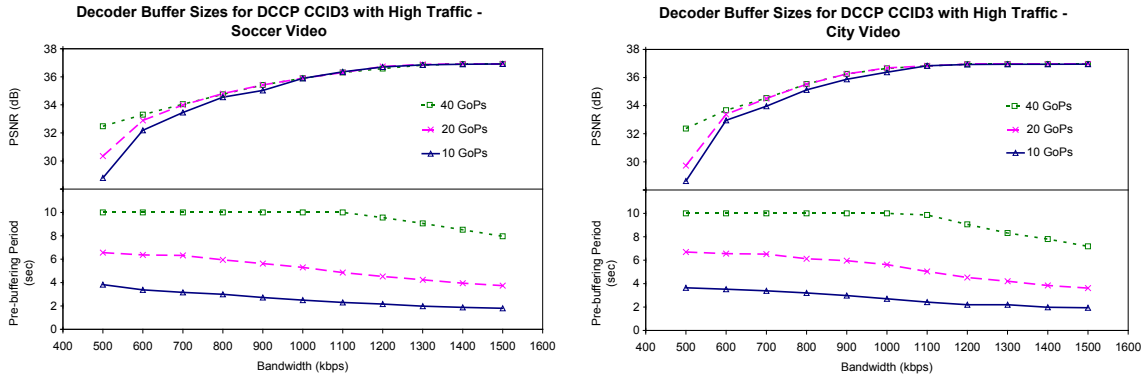


Figure 3.18 Results for different decoder buffer sizes under high cross traffic with DCCP CCID3

3.3.6 Comparison of protocols

This section compares the performances of DCCP CCID3, DCCP CCID2, and TCP Reno under varying cross traffic. Figure 3.19 gives the PSNR values of the received video for each protocol used, from which the TCP's retransmission penalty can be clearly seen under heavy congestion. On the other hand, the protocols show similar performance when there is plenty of available network capacity. Both CCIDs of DCCP behave alike under high cross traffic, with CCID2 performing slightly better at the highest congestion. At other times, CCID3 shows a marginally better performance compared to CCID2, about 0.3 dB at maximum.

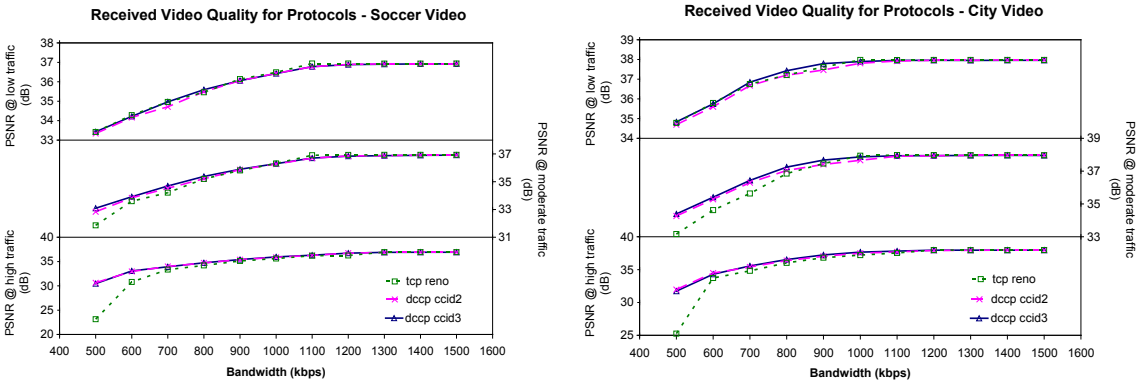


Figure 3.19 PSNR values for DCCP CCID3, DCCP CCID2 and TCP Reno under varying cross traffic

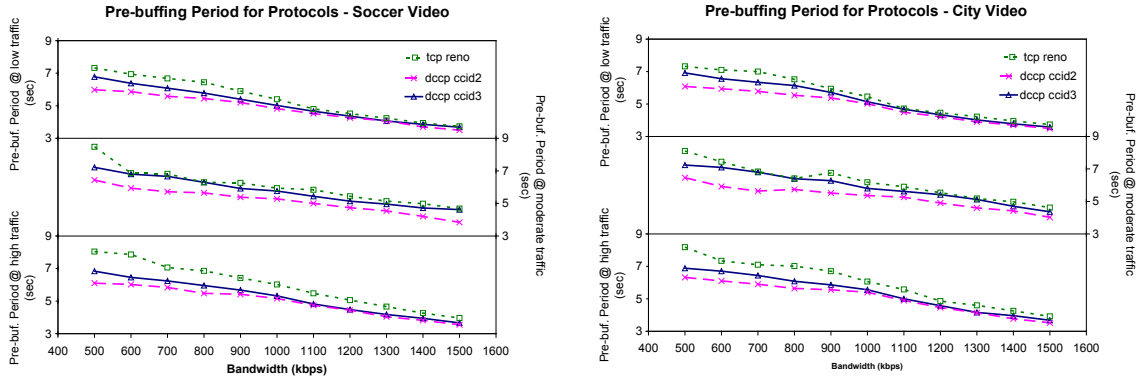


Figure 3.20 Pre-buffering period in seconds for DCCP CCID3, DCCP CCID2 and TCP Reno under varying cross traffic, when the decoder buffer size is 20 GoPs and the playout start threshold is 10 GoPs.

Figure 3.20 shows pre-buffering period lengths of the tests performed, whose PSNR values are given in the previous figure. The tests are done with a decoder buffer size of 20 GoPs, meaning that the pre-buffering lasts until 10 GoPs of video is received in the buffer. It is apparent from the figure that the shortest pre-buffering period is always achieved with DCCP CCID2, thanks to its AIMD behavior, with maximum differences of 1.2 and 2 seconds compared to DCCP CCID3 and TCP Reno, respectively. DCCP CCID3 results in second shortest pre-buffering period, whereas the longest pre-buffering is achieved with TCP Reno.

3.3.7 Internet tests

Streaming tests are performed using DCCP CCID3 and TCP Reno, between the Koç University and Argela Technologies, which are connected with a link of 5 Mbps approximately. DCCP tests are limited to CCID3, that is, DCCP CCID2 is not used in the Internet tests, since both CCIDs performed similar in the LAN tests.

Figure 3.21 shows the quality of the received videos averaged over each test set, which comprises twenty tests run sequentially in a close time-frame, using DCCP CCID3 and TCP Reno in alternating fashion. That is, a DCCP CCID3 test is followed by a TCP Reno

test, which is followed by CCID3, continuing until twenty tests are run, making ten for DCCP and TCP. The results of these fifteen tests are then averaged for each protocol, prior to being presented. Tests done without competing traffic between end-hosts reveal that DCCP and TCP perform similar when there is enough network capacity. Here, it should be noted that there still can be uncontrollable short-lived cross traffic between the sender-receiver hosts, since the tests are done in the open Internet. When competing traffic is started alongside with the video stream, which is basically an FTP flow, DCCP performs better than TCP, as much as 1.6 dB at maximum in terms of received video quality. Both of these results – with and without competing traffic – agree with the controlled LAN tests. Moreover, it is apparent in Figure 3.22 that TCP Reno leaves more bandwidth to the competing flow than DCCP CCID3, with DCCP still being fair to the competing flow. The average video-to-competing-flow ratio is 1.3 for DCCP, whereas it is 0.8 for TCP.

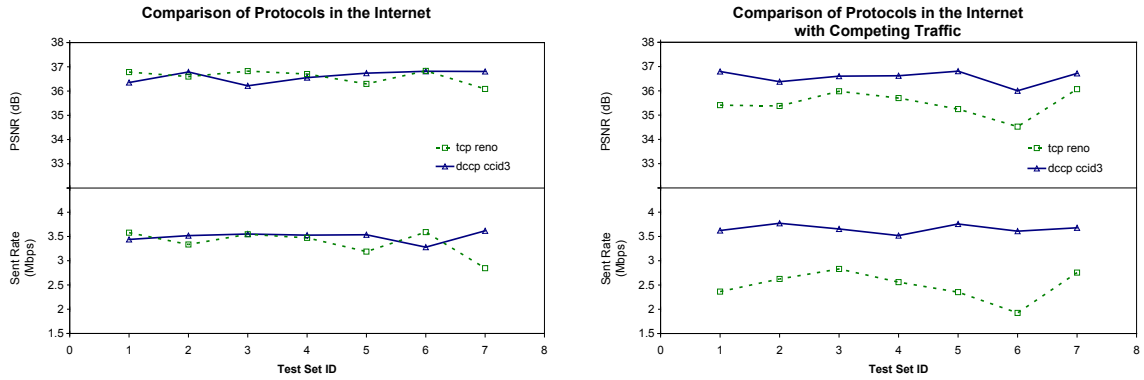


Figure 3.21 Results of Internet tests using DCCP CCID3 and TCP Reno, without and with competing traffic.

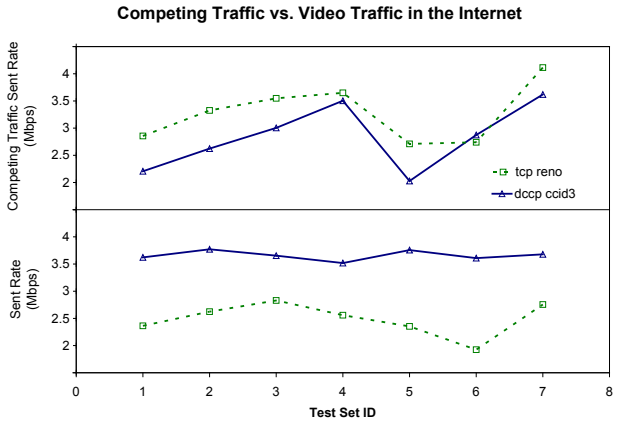


Figure 3.22 Rates of competing traffic and video stream, for DCCP CCID3 and TCP Reno.

3.4 Conclusion

In this work, an end-to-end adaptive video streaming system is introduced; covering key techniques in network adaptation and transmission. For each technique presented, extensive set of tests are performed, to find the optimal combination for streaming SVC video over the Internet.

Different bandwidth estimation techniques are implemented for different transport protocols, such as utilizing the TFRC rate when DCCP CCID3 is used or calculating the packet transmission rate otherwise. Tests performed using various averaging window lengths reveal that averaging improves the video quality considerably when DCCP CCID2 or TCP is utilized. Averaging the TFRC rate to estimate the available network rate when using DCCP CCID3 also increases the quality of the received video, but only to a limited extend. Hence, the packet transmission rate should be averaged over a moving window of at least one second long, to be used as a bandwidth estimate, in case DCCP CCID2 or TCP is utilized.

Controlling the transmission rate in the application layer exposed the oscillating nature of DCCP CCID3, decreasing the quality of the received video. A similar result is also

experienced with DCCP CCID2 and TCP Reno. Video should be sent at the maximum available network rate rather than sending at the extraction rate, as long as the receiver buffer allows. When the network capacity is high, the transmission rate may also be limited with the maximum extractable video rate, to decrease the retransmission traffic without affecting the received video quality. Hence, for application layer sending rate control, we recommend the following strategy: Unless there is receiver buffer overflow risk, send video at the maximum extractable video rate. Otherwise, send video at the current extraction rate.

When dealing with losses, retransmitting only base layer packets are sufficient, except for the case when the network is over-provisioned. A better approach is retransmitting base or all lost packets, depending on the available network rate. In this work, an adaptive ARQ scheme is proposed, which aims to scale the retransmission traffic by requesting only base layer packets when the network rate is low and requesting all missing packets otherwise, using the decoder buffer occupancy information.

The transport protocols used in the streaming tests are also compared in terms of received video quality and pre-buffering period length. It is evident from the figures that TCP performs worst under heavy congestion, due to its retransmission policy. At other times, when there is plenty of available network capacity, the protocols show similar performance, in terms of received video quality. This result is also observed during the actual Internet tests. Moreover, the pre-buffering tests reveal that the quickest playout start can be achieved with DCCP CCID2 and the longest pre-buffering is observed with TCP Reno. Hence, using DCCP, if available, is recommended. Otherwise, heavy congestion should be prevented in case of using TCP.

Chapter 4

ADAPTATION STRATEGIES FOR WIRELESS NETWORKS

4.1 Introduction

It is envisioned that access networks will be mostly wireless in the future. Hence, it is of interest to consider extensions of DCCP for wireless networks. DCCP, with its current congestion control schemes, mimics TCP in responding to congestion, hence inheriting the problems of TCP in wireless networks. It interprets packet loss as a sign of congestion and decreases its sending rate accordingly. This is fair in wired networks where packets get lost only due to congestion. However, in the wireless domain, data loss is often caused by wide variations in the quality of the wireless link over time, rather than congestion. These variations are due to multipath fading, shadowing, path loss as well as terminal mobility. Considering losses of all types as if they were due to congestion and acting accordingly results in inefficient use of the limited wireless link capacity.

There have been many efforts to solve the problems of TCP and TFRC in wireless networks [38]. These approaches either hide end-hosts from wireless losses and transform the wireless link to a lossless link with reduced bandwidth or provide tools for end-hosts to identify the existence of wireless hops to take counter measures. [39] and [40] uses cross-layer information provided in the radio link control layer (RLC) to differentiate between wireless and congestion losses.

In this work, we propose a solution to increase the efficiency of DCCP over wireless links for video streaming where only congestion losses are used in the congestion control scheme. To achieve this, we employ limited checksum coverage in DCCP packets so that

DCCP does not consider erroneous packets as congestion indication, treating such packets as error-free.

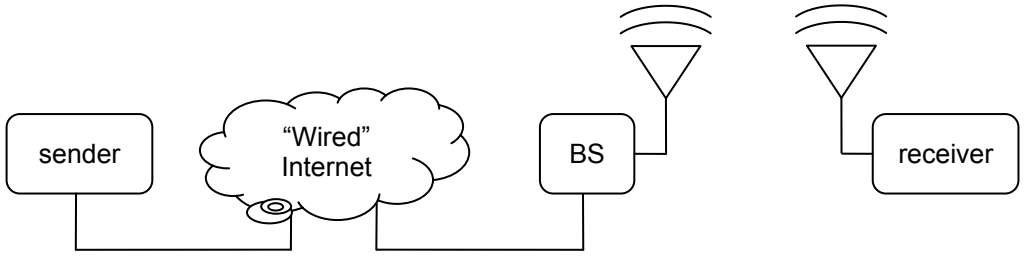


Figure 4.1 Heterogeneous network, covering both wired and wireless links.

This work extends the adaptive video streaming system, introduced in Chapter 3, to wireless domain. The same sender and receiver nodes are utilized, but this time connected over a heterogeneous network, covering both wired and wireless links, as depicted in Figure 4.1. In addition to employing limited checksum coverage to differentiate between wireless and congestion losses, several adaptation schemes are also tested. Furthermore, both CCIDs (CCID2: TCP-like and CCID3: TFRC) of DCCP are compared to each other in terms of video streaming performance.

4.2 Extending the adaptive video streaming system to wireless domain

4.2.1 Limited checksum coverage

In a typical wireless network, it is well-known that the physical layer packets are much smaller in size than the IP packets. An IP packet loss occurs when any physical layer packet belonging to the IP packet fails to be delivered. In the wireless system, if physical layer ARQ is available, retransmission of erroneous physical layer packets is possible. If the physical layer ARQ is not able to correct the packet, and the maximum retransmission time is achieved, a wireless error occurs.

Similar to the wireless TFRC framework given in [39] and [40], we propose using the MAC layer wireless loss information to distinguish between wireless and congestion losses, but in an indirect manner. Rather than realizing a cross-layer mechanism for information transfer, we make sure that the MAC layer delivers packets to DCCP although frame reassembly fails, indicating wireless loss. We also limit the checksum of DCCP packets to cover only header data, omitting payload, by using the header checksum coverage field in the DCCP generic packet header [14]. In this way, DCCP does not detect wireless errors occurring in the payload, passing such packets to the application layer without reducing the transmission rate. It is up to the application to detect corrupted data and use it accordingly.

The proposed "limited checksum coverage" solution is based on the following assumptions:

- *Wireless errors mostly occur in the payload.*

  Considering the ratio of protocol headers to the payload, this is what will mostly occur. In case of header data corruption, however rare, the corresponding packet will be considered as lost due to congestion.

- *The first physical layer packet of each IP packet should be transmitted without error.*

  Most of the time, IP packets are fragmented into multiple physical layer packets because they are much bigger in size than the physical layer packets. The first physical layer packet of an IP packet will carry the header data, with some part of the payload. Hence, this first packet should be received without error by the end terminal. Otherwise, the corresponding IP packet will be considered as lost due to congestion.

4.3 Results

This section covers video streaming tests performed over a network where the last node (receiver) is wireless, as shown in Figure 4.2. The tests are done with the limited checksum coverage as well as the original unlimited checksum – checksum covering both the header and the payload, and their results are compared to each other. Also, several adaptation strategies introduced in the previous chapter are tested in the wireless setup.

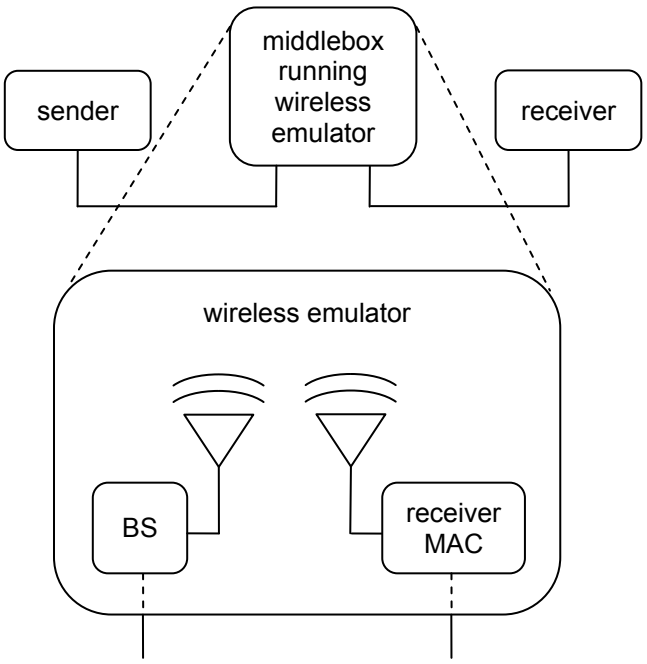


Figure 4.2 Wireless test setup, with the wireless emulator shown in detail.

We use the IS-856 standard Release 0 (1xEV-DO) [42] as the underlying wireless infrastructure. IS-856 is used due to its spectral efficiency and inherent physical layer ARQ capability. The wireless emulator, running on the middlebox node as depicted in Figure 4.2, is responsible for emulating the IS-856 network. The *Base Station* (BS) module inside the emulator is used for realizing the IS-856 wireless access medium, together with the MAC layer in the BS performing fragmentation/defragmentation of the packets sent/received.

The physical layer packets received from the BS are reassembled in the *receiver MAC* module, prior to being directed to the receiver node. The receiver MAC module is also responsible for fragmenting the IP packets to be sent to the BS. The sizes of the physical layer packets are determined by the observed SNR values of the mobile users at a given time. The BS module emulates the wireless medium by queuing received physical layer packets and sending them with data rates determined according to the observed SNR values for a 1% physical layer packet error rate. Simple ARQ is utilized by the MAC layer, where erroneous packets are retransmitted for a maximum resend count, which is either 0 (ARQ disabled), 1, 2, or 5, until received without error. *Proportional Fair Scheduler* is employed in the BS to service mobile users, with a moving average window of 1000 slots – 1.67 seconds.

Table 4.1 The minimum, the maximum and the average data rates of the five different rate sets, measured over 10 users.

Rate Set ID		***1***	***2***	***3***	***4***	***5***
Pedestrian	**Min**	860	823	886	841	795
Rates	**Max**	1464	1257	1344	1365	1321
(kbps)	**Avg**	1184	1046	1096	1127	1019
Vehicular	**Min**	647	652	650	655	656
Rates	**Max**	698	700	680	705	704
(kbps)	**Avg**	677	673	667	683	676

We emulate two ITU wireless channel scenarios: *Pedestrian A* and *Vehicular B* as in [41]. Five different set of SNR values, each having values for 10 different mobile users for a period of 180 seconds, are utilized during the tests. Table 4.1 gives the minimum, the maximum and the average data rates corresponding to the SNR values, measured over 10 users. Each of these rate sets are used by a test set, which consists of five video streaming

tests for each user, for a total of 10 users, making 50 streaming tests, each longing for 96 seconds. Rate set 1 is used by test set 1, rate set 2 is used by test set 2, and so on. The results of these 50 tests for each test set are then averaged and shown in the figures below. Besides the results averaged over all users, results corresponding to the users experiencing the minimum and the maximum data rates are also given.

We use the same Soccer sequence as in Chapter 3 in our tests. The encoding configuration is also kept the same, with a base and an MGS layer with two sub-layers, except for the quantization parameters. Two versions of the video are streamed, both of them at CIF resolution: a normal quality version with quantization parameter (38,28) – which is the one used in the previous chapter – for the vehicular channel and a higher quality version with quantization parameter (34,24) for the pedestrian channel. The minimum (base layer only) and the maximum (base + MGS enhancement) extractable rates (without packetization) of the sequences are listed in Table 4.2.

Table 4.2 Min./max. average extraction rates (without packetization) and PSNR values for the Soccer video at normal quality and high quality.

		Rate (kbps)	**PSNR (dB)**
Soccer @ NQ	**Min**	316	32.20
	Max	945	36.98
Soccer @ HQ	**Min**	518	34.33
	Max	1440	39.57

4.3.1 Limited checksum coverage

Figure 4.3**,** Figure 4.4 and Figure 4.5 compare the performances of the proposed limited checksum coverage solution with the original unlimited checksum when no MAC layer ARQ is employed. As can be seen in the figures, the proposed solution performs better than the original setup in all of the cases, but the difference is most notable when CCID3 is used

in the Pedestrian A channel, which can be as high as 1.2 dB. DCCP CCID2 seems to perform better than CCID3, which can be observed better in Figure 4.11.

The reason of the performance difference between the limited and the unlimited checksum can be better seen in Figure 4.6, where the loss event rate values of the original unlimited setup are about four times that of the limited checksum solution. This is because the loss event rate includes the packet losses due to wireless errors in the original setup. This difference results in a higher calculated TFRC rate, as can be noted in the figure.

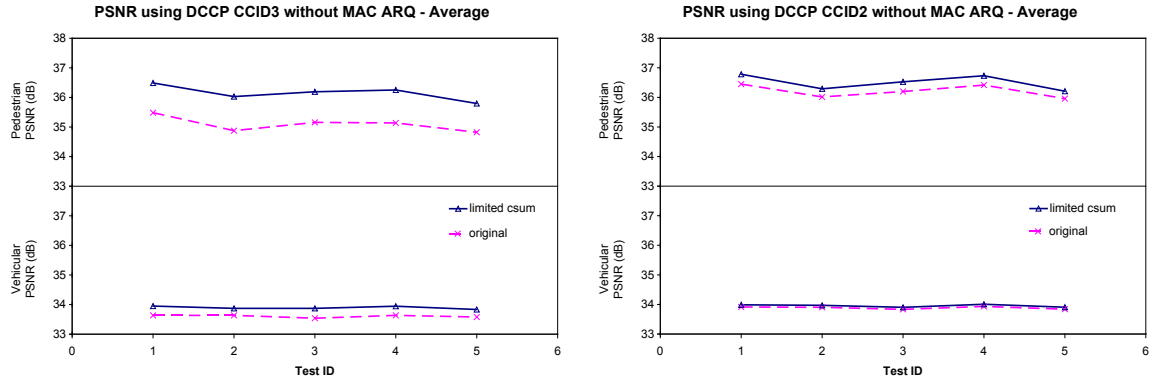


Figure 4.3 Comparison of the limited checksum coverage solution with the original unlimited checksum – checksum covering both the header and the payload, when no MAC layer ARQ is employed. The results are averaged over all users.

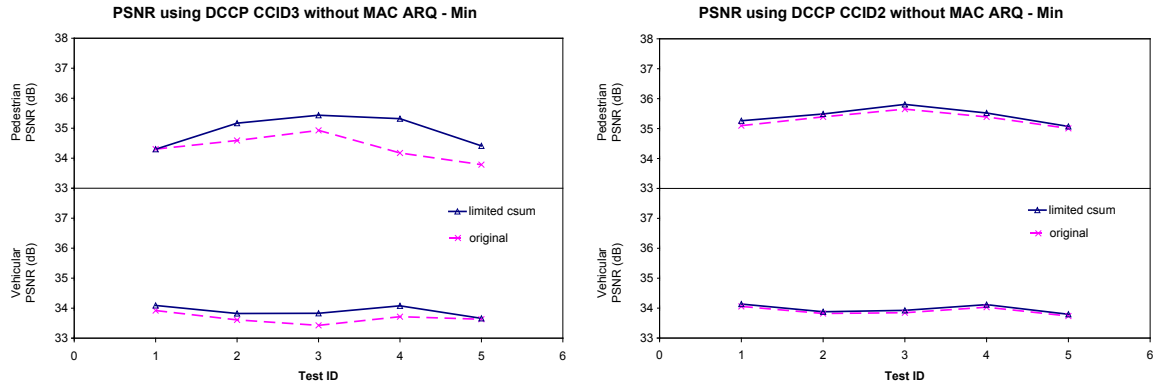


Figure 4.4 Comparison of the limited checksum coverage solution with the original unlimited checksum, when no MAC layer ARQ is employed. The results correspond to the user experiencing the minimum data rate.

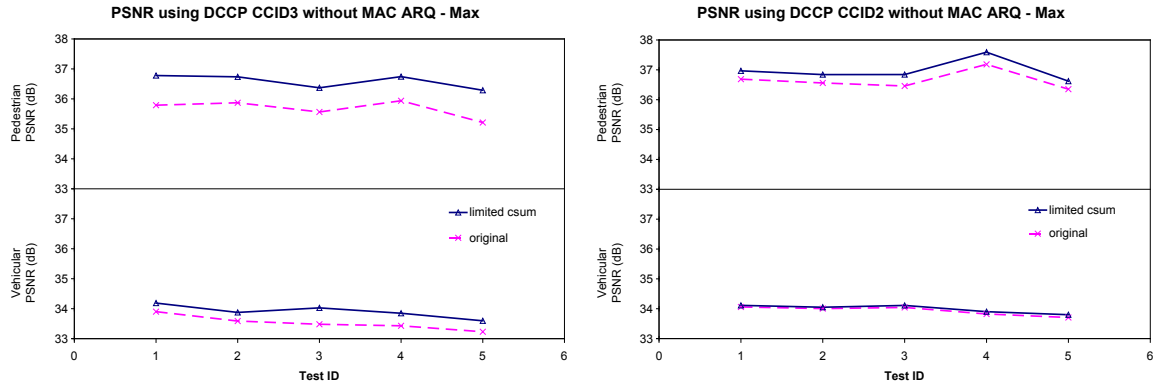


Figure 4.5 Comparison of the limited checksum coverage solution with the original unlimited checksum, when no MAC layer ARQ is employed. The results correspond to the user experiencing the maximum data rate.

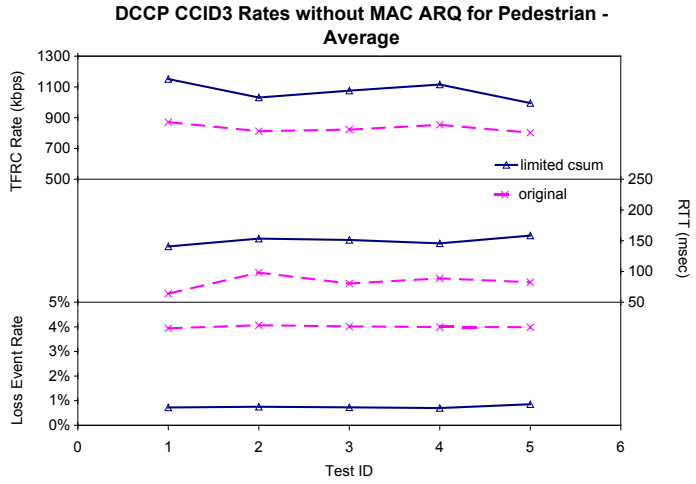


Figure 4.6 DCCP CCID3 rates for the Pedestrian A channel, when no MAC layer ARQ is used. The results are averaged over all users.

When MAC layer ARQ is used, however, the limited checksum solution behaves almost identical with the original setup, as seen in Figure 4.7. Even a single retransmission of a corrupted segment removes the performance difference between the limited and the unlimited checksum setup for most of the tests.

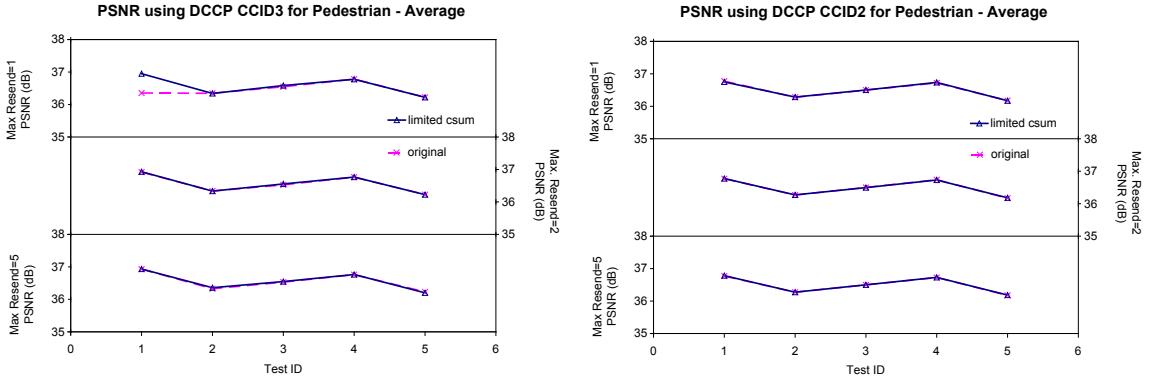


Figure 4.7 Comparison of the limited checksum coverage solution with the original unlimited checksum, when MAC layer ARQ is employed. The results are averaged over all users.

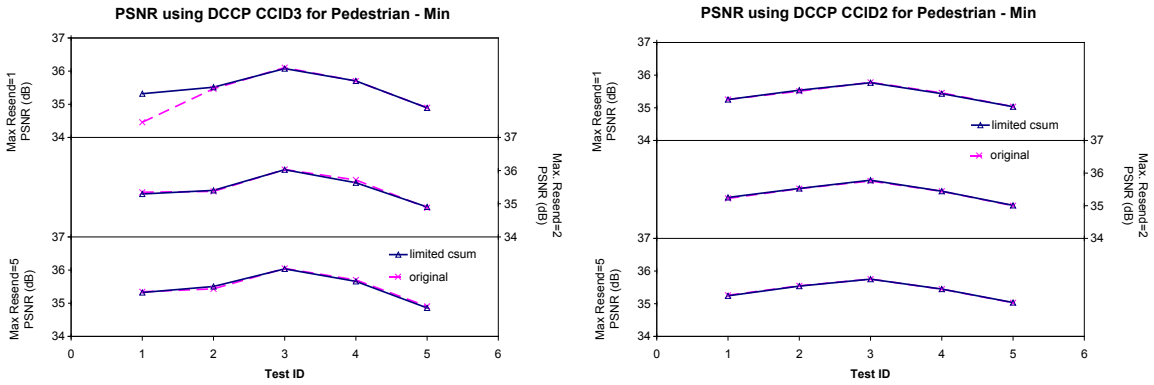


Figure 4.8 Comparison of the limited checksum coverage solution with the original unlimited checksum, when MAC layer ARQ is employed. The results correspond to the user experiencing the minimum data rate.

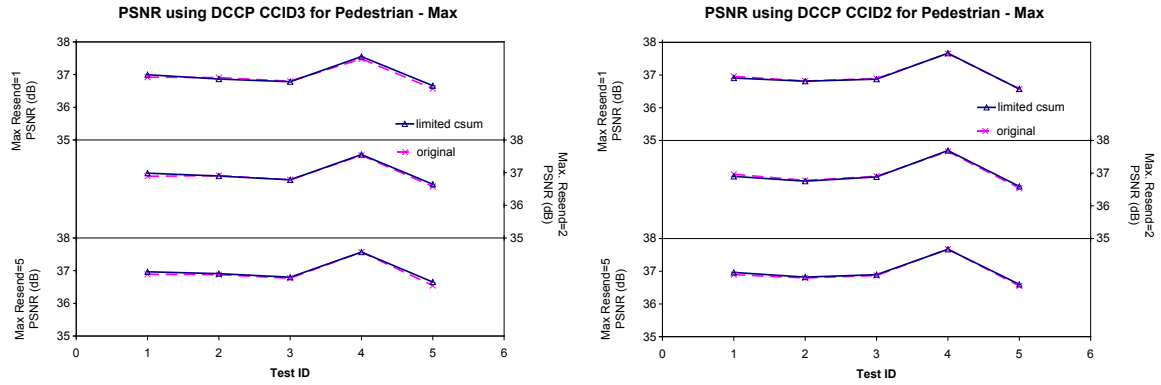


Figure 4.9 Comparison of the limited checksum coverage solution with the original unlimited checksum, when MAC layer ARQ is employed. The results correspond to the user experiencing the maximum data rate.

It is interesting to observe in Figure 4.10 that enabling MAC layer ARQ does not increase the received video quality in case of limited checksum coverage when DCCP CCID2 is utilized. This is because CCID2 does not take wireless losses as congestion indication and hence does not decrease its transmission rate, when limited checksum is used. Moreover, the application layer ARQ is able to recover the corrupted packets. The effect of wireless losses are noticeable when original setup is utilized with DCCP CCID2.

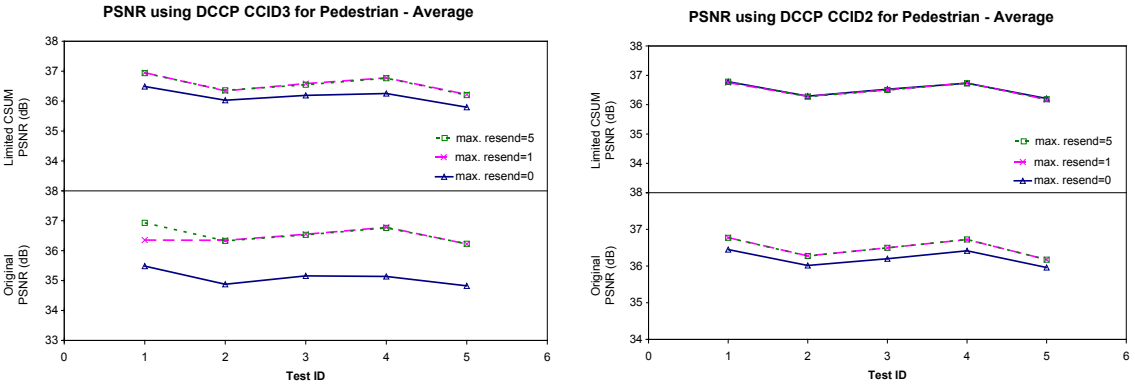


Figure 4.10 Comparing the PSNR results of various maximum MAC layer retransmission counts. The results are averaged over all users.

Different from CCID2, enabling MAC layer ARQ improves the received video quality even when limited checksum coverage is used with CCID3. This is due to the fact that the TFRC based CCID3 reacts to changing network conditions slower than the AIMD based CCID2. Although wireless losses are not taken as congestion indication, retransmission of corrupted packets decreases the effective bandwidth.

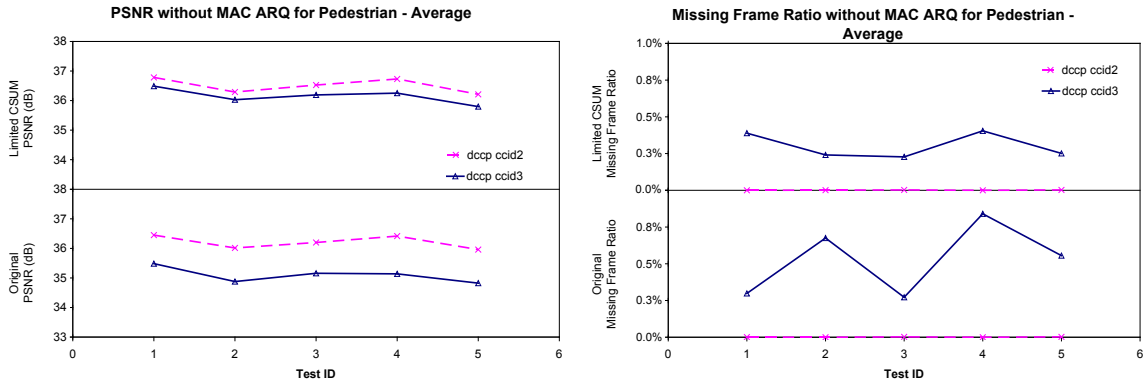


Figure 4.11 CCID2 vs. CCID3, with both the limited checksum coverage solution and the original unlimited checksum, when MAC layer ARQ is not employed. The results are averaged over all users.

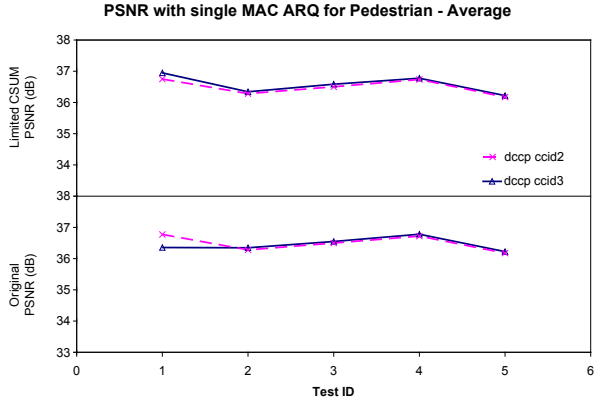


Figure 4.12 CCID2 vs. CCID3, with both the limited checksum coverage solution and the original unlimited checksum, when a single MAC layer ARQ is enabled. The results are averaged over all users.

It is evident from Figure 4.11 that DCCP CCID2 performs better than CCID3, when MAC layer ARQ is disabled. The difference is as much as 1.3 dB in the original setup and 0.5 dB when limited checksum coverage is used. This is different from the results achieved in the previous chapter, where CCID3 showed similar performance with CCID2, if not better. However, when MAC layer ARQ is enabled, the results agree with the ones in Chapter 3, as seen in Figure 4.12.

4.3.2 Sending rate control

Figure 4.13 compares the performances of the sending rate control schemes in terms of received video quality and retransmission traffic, when DCCP CCID3 is used with limited checksum coverage and MAC layer ARQ is disabled. The results agree with the ones given in the previous chapter: Sending video at the maximum or limited maximum rate produces videos with similar quality, whereas sending at the extraction rate results in videos with lower quality.

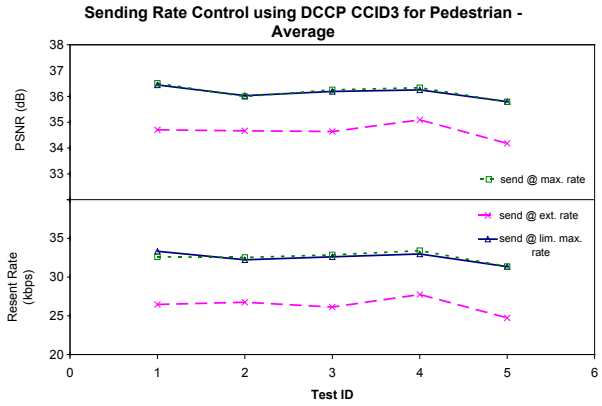


Figure 4.13 Comparison of application layer sending rate control schemes using DCCP CCID3 with limited checksum coverage, when MAC layer ARQ is disabled. The results are averaged over all users.

4.3.3 Retransmission of lost packets

The performances of the three application layer ARQ schemes are compared using DCCP CCID3 with limited checksum coverage and similar results are observed as in the previous chapter. *arq base* and *arq adaptive* schemes show parallel performance, whereas *arq all* results in a slight quality improvement at the cost of increased retransmission traffic, as seen in Figure 4.14.

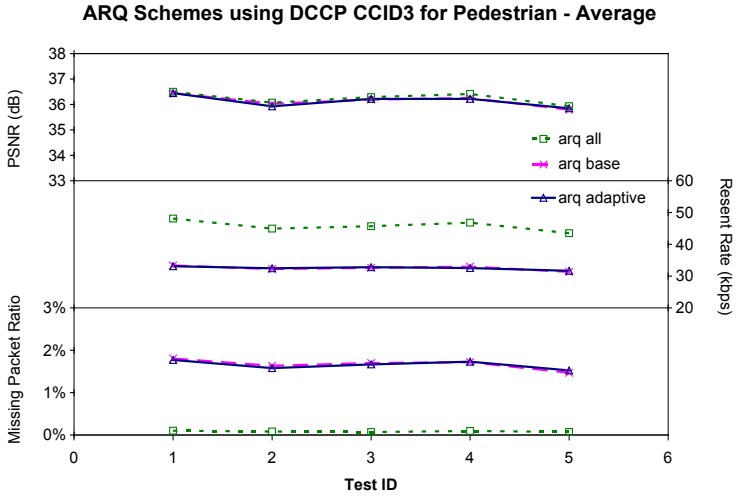


Figure 4.14 Comparison of application layer ARQ schemes using DCCP CCID3 with limited checksum coverage, when MAC layer ARQ is disabled. The results are averaged over all users.

4.4 Conclusion

The results presented show that differentiating between wireless and congestion losses improves the received video quality as much as 1.2 dB, when MAC layer ARQ is not employed. However, even a single ARQ removes the quality gain owed to the limited checksum coverage solution in most of the cases, when the physical layer packet error rate is set as 1%. Hence, when MAC layer ARQ is possible, there is no need to differentiate the wireless and congestion losses in the tested setup. But in cases where physical layer

retransmission is not applicable, the proposed solution results in considerable improvements in received video quality.

DCCP CCID2 performs better than CCID3 in terms of received video quality, both for limited and unlimited checksum coverage, when MAC layer ARQ is disabled. This is due to CCID2's fast reacting nature; it can tolerate wireless losses better than CCID3, which reacts slower than CCID2. However, when ARQ is enabled, CCID2 and CCID3 show similar performance.

Chapter 5

CONCLUSIONS

A successful video streaming system should employ adaptation strategies in both coding and transport domains. This work first focuses on video coding to determine the encoding configuration and the extraction method, resulting in optimal rate distortion values. After that, the research concentrates on transport, evaluating several adaptation strategies for streaming video over wired/wireless networks.

During the experiments to find the optimal encoding configuration, it is observed that splitting the MGS layer into more than five sub-layers may cause degradation in the video quality. Although the results show that performance of different encoding configurations and extraction methods may change with content, a compromising solution is available that includes all of the videos tested, where the MGS layer is fragmented into two sub-layers with the fragmentation configuration of (6,10) and the byte-limited slice mode enabled (limit set as 1400 bytes). It is apparent that limiting the maximum slice size by enabling slice mode has a positive impact on the PSNR of the extracted video at low bitrates, in addition to restricting the effects of packet loss. The proposed priority-based hierarchical extraction method is employed for video rate adaptation in the compromising solution.

Transport domain tests reveal that controlling the video transmission rate in the application layer may decrease the quality of the received video. Hence, video should be sent at the maximum available network rate rather than at the extraction rate, as long as the receiver buffer allows. Also, retransmitting only base layer or all missing packets

adaptively, depending on the available network rate, results in better video quality and better network utilization.

As the performances of TCP and DCCP are compared to each other, it is observed that TCP performs worst under heavy congestion, as expected. However, the transport protocols show similar performance when there is plenty of available network capacity.

The adaptive video streaming system is also extended to wireless domain, and a solution based on limited checksum coverage is proposed to improve the performance of DCCP in wireless networks. The solution can improve the received video quality as much as 1.2 dB, when the physical layer ARQ is disabled.

This work focuses on the server-client architecture where the clients are rather dumb and the server is solely responsible for estimating the available network rate and adapting the video encoding rate accordingly. However, HTTP Live Streaming proposes the opposite architecture, where the clients are responsible for adaptation and the server only acts as a web server. This new architecture, started to be used by Apple, Microsoft and others, seems to be promising as it allows service providers to use the current web server infrastructure. Hence, extending the adaptive video streaming system to this new architecture is planned as a future task.

BIBLIOGRAPHY

[1] R. Pantos, "HTTP live streaming," Internet-Draft, draft-pantos-http-live-streaming-04, Jun. 2010.

[2] H. Sun, A. Veto, and J. Xin, “An overview of scalable video streaming,” *Wireless Communications & Mobile Computing*, vol. 7, no. 2, pp. 159–172, Feb. 2007.

[3] B. Xie and W. Zeng, “Rate distortion optimized dynamic bitstream switching for scalable video streaming,” *IEEE Int. Conf.on Multimedia and Expo (ICME)*, vol. 2, pp. 1327–1330, Taipei, Taiwan, Jun. 2004.

[4] H. Schwarz, D. Marpe, T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[5] T. C. Thang, J.-G. Kim, J. W. Kang, and J.-J. Yoo, “SVC adaptation: Standard tools and supporting methods,” *Signal Processing: Image Communication*, vol. 24, no. 3, pp. 214-228, Mar. 2009.

[6] D. T. Nguyen and J. Ostermann, “Congestion control for scalable video streaming using the scalability extension of H.264/AVC”, *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 2, Aug. 2007.

[7] J. Vieron and C. Guillemot, “Real-time constrained TCP-compatible rate control for video over the Internet,” *IEEE Trans. Multimedia*, vol. 6, no. 3, pp. 634–646, Aug. 2004.

[8] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, P. Amon, “Real-time system for adaptive video streaming based on SVC,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol.17, no.9, pp. 1227–1237, Sep. 2007.

[9] B. Wang, J. F. Kurose, P. J. Shenoy, and D. F. Towsley, “Multimedia streaming via TCP: an analytic performance study,” *Proc. ACM Multimedia’04*, pp. 908–915, Oct. 2004.

[10] A. Goel, C. Krasic, and J. Walpole, “Low-latency adaptive streaming over TCP,” *ACM Trans. on Multimedia Computing, Comm., and Appl. (TOMCCAP)*, vol. 4, no. 3, Aug. 2008.

[11] S. Lee and K. Chung, “Buffer-driven adaptive video streaming with TCP-friendliness,” *Computer Communications,* vol. 31, no. 10, pp. 2621–2630, Jun. 2008.

[12] P. Zhu, W. Zeng, and C. Li, "Joint design of source rate control and QoS-aware congestion control for video streaming over the Internet," *IEEE Trans. on Multimedia,* vol. 9, no. 2, pp. 366-376, Feb. 2007.

[13] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," IETF RFC 5348, Sep. 2008.

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

[15] S. Floyd, E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)," IETF RFC 4342, Mar. 2006.

[16] S. Floyd, and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control," IETF RFC 4341, Mar. 2006.

[17] Y. B. Zikria, S. A. Malik, H. Ahmed, S. Nosheen, N. Z. Azeemi, and S. A. Khan, "Video transport over heterogeneous networks using SCTP and DCCP," *Wireless Networks, Information Processing and Systems, IMTIC*, pp. 180–190, Apr. 2008.

[18] S. Linck, E. Mory, J. Bourgeois, E. Dedu, and F. Spies, "Video quality estimation of DCCP streaming over wireless networks", *Parallel, Distributed, and Network-Based Processing (PDP),* Feb. 2006.

[19] M. A. Azad, R. Mahmood, and T. Mehmood, "A comparative analysis of DCCP variants (CCID2, CCID3), TCP and UDP for MPEG4 video applications," *Int. Conf. on Information and Communication Technologies (ICICT)*, pp. 40–45, Aug. 2009.

[20] A. Balk, M. Gerla, and M. Sanadidi, "Adaptive video streaming: Pre-encoded MPEG4 with bandwidth scaling," *Comput. Netw.,* vol. 44, no. 4, pp. 415-439, Mar. 2004.

[21] P. de Cuetos, P. Guillotel, K. W. Ross, and D. Thoreau, "Implementation of adaptive streaming of stored mpeg-4 FGS video over TCP," *IEEE Int. Conf. Multimedia and Expo (ICME)*, pp. 405–408, Aug. 2002.

[22] B. Girod, M. Kalman, Y.J. Liang, R. Zhang, "Advances in channel-adaptive video streaming," *IEEE Int. Conf. Image Processing (ICIP)*, New York, Dec. 2002.

[23] J. Chakareski, J.G. Apostolopoulos, S. Wee, W. Tan, and B. Girod, "Rate-distortion hint tracks for adaptive video streaming," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.15, no.10, pp. 1257–1269, Oct. 2005.

[24] A. Sehgal, O. Verscheure, and P. Frossard, “Distortion-buffer optimized TCP video streaming,” *IEEE Int. Conf. Image Processing (ICIP),* pp. 2083–2086, Oct. 2004.

[25] B. Gorkemli and A. M. Tekalp, “Adaptation strategies for streaming SVC video,” *IEEE Int. Conf. Image Processing (ICIP),* Hong Kong, Sep. 2010.

[26] B. Gorkemli, Y. Sadi, and A. M. Tekalp, “Effects of MGS fragmentation, slice mode and extraction strategies on the performance of SVC with medium-grained scalability,” *IEEE Int. Conf. Image Processing (ICIP),* Hong Kong, Sep. 2010.

[27] C. G. Gurler, B. Gorkemli, G. Saygili, and A. M. Tekalp, “Flexible transport of 3D video over IP,” in *Journal of Proc. IEEE*, 2010 [submitted].

[28] B. Gorkemli, M. O. Sunay, and A. M. Tekalp, “Video streaming over wireless DCCP,” *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 2028-2031, San Diego, CA, Oct. 2008.

[29] N. Ozbek, B. Gorkemli, A. M. Tekalp, and E. T. Tunali, “Adaptive streaming of scalable stereoscopic video over DCCP,” *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 489-492, San Antonio, TX, Sep. 2007.

[30] B. Gorkemli, M. R. Civanlar, “SVC coded video streaming over DCCP,” *IEEE Int. Symp. on Multimedia (ISM)*, pp. 437-441, San Diego, CA, Dec. 2006.

[31] H. Kirchhoffer, D. Marpe, H. Schwarz, and T. Wiegand, “A low-complexity approach for increasing the granularity of packet-based fidelity scalability in scalable video coding,” *Picture Coding Symposium (PCS)*, Lisbon, Portugal, Nov. 2007.

[32] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux, “Optimized rate-distortion extraction with quality layers,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1186–1193, Sep. 2007.

[33] E. Maani and A. K. Katsaggelos, “Optimized bit extraction using distortion modeling in the scalable extension of H.264/AVC,” *IEEE Trans. on Image Processing*, vol. 18, no. 9, pp. 2022–2029, Sep. 2009.

[34] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RTP payload format for H.264 video," IETF RFC 3984, Feb. 2005.

[35] J. Lazzaro, “Framing RTP and RTCP packets over connection-oriented transport,” IETF RFC 4571, Jul. 2006.

[36] J. Cao, W.S. Cleveland, Y. Gao, K. Jeffay, F.D. Smith, and M.C. Weigle, “Stochastic models for generating synthetic HTTP source traffic”, *IEEE INFOCOM*, Hong Kong, Mar. 2004.

[37] Available online: http://www.linuxfoundation.org/collaborate/workgroups/networking/dccp_testing#experimental_dccp_source_tree, last accessed at Aug 10, 2010.

[38] M. Chen and A. Zakhor, "Rate control for streaming video over wireless," *IEEE Wireless Communications*, pp. 32-41, Aug. 2005.

[39] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end TCP-friendly streaming protocol and bit allocation for scalable video over wireless Internet" *IEEE Journal on Selected Areas in Comm.*, vol. 22, no. 4, May 2004.

[40] Y. Fu, R. Hu, G. Tian, and Z. Wang, "TCP-friendly rate control for streaming service over 3G network," *WiCOM 2006*, pp. 1-4, Sep. 2006.

[41] C. Atici and M. O. Sunay, "High Data-Rate video broadcasting over 3G wireless systems," *IEEE Transactions on Broadcasting,* vol. 53, no. 1, pp. 212-223, Mar. 2007.

[42] TIA/EIA/IS-856, "cdma2000 high rate packet data air interface specification" 3GPP2, C.S0024, v4.0, Oct. 2002.