An Open-Standards Based Server-Client Model for Robust Streaming of Multi-View Content over the Internet

by

Cihat Göktuğ Gürler

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

Master of Science
in
Electrical and Computer Engineering

Koc University

September 2008

Koc University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Cihat Göktuğ Gürler

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)

Assist. Prof. Oğuz Sunay

Assist. Prof. Sibel Salman

Date:

To my wife

ABSTRACT

The Internet Protocol (IP) is the natural choice for 3D video transmission, if we considering the recent success of many voice over IP (VoIP), video over IP, and IPTV applications. However, the choice of IP for the network layer does not make the transmission application design straightforward, since IP only dictates the use of unreliable packet switched networks. Its flexible architecture allows use of various transmission protocols and policies over IP. Therefore, it is the task of the application to implement an efficient scheme that will most effectively manage the available channel throughput. This can only be achieved by understanding the characteristics of each layer in the protocol stack and implementing a cross-layer design approach.

This thesis proposes a novel framework for streaming of 3D video based on multi-view video (MVV) representations. The proposed streaming platform is completely open standards based, flexible, and backwards compatible for supporting monoscopic streaming to legacy clients. The MVV in the server is compressed using a simplified form of MVC with negligible loss of compression efficiency and streamed using RTSP, SDP and RTP to clients. Raptor Codes are used for fighting with the possible packet loses at the network layer. The clients can perform basic error concealment to reduce the effects of remaining packet losses and can decode MVC in real-time. The modular client can display decoded 3D content on a multitude of 3D display systems.

The proposed streaming platform has been extensively tested over the Internet to find answers to the following questions related to 3DTV transmission over IP:
i) What is the required bandwidth for 3D transmission, and what are the parameters that affact this ratio? ii) Is the cost of enabling slice mode justified for achieving better channel throughput? iii) What is the required rate of channel coding against what percentage of packet losses? iv) How does Raptor Codes perform when we compare its redundancy level against alternative strategies such as Multiple Description Coding? v) What are the required modifications over the standard streaming protocols for 3DTV transmission? vi) How can the design take advantage of the multi-core processors? vii) Are there any other parameters that have significant influence on packet loss ratio beside bit-rate of the stream? If there is, do they affect the packetization strategy?

ÖZET

İnternet protokolü (IP) üzerinden ses, video ve IPTV gibi uygulamaların başarısı göz önüne alındığında, 3BTV için iletiminin de IP üzerinden yapılması en makul seçimdir. Ancak bu seçimi yapmış olmak, iletimi gerçekleştirecek olan uygulamanın hazırlanmasını basit bir hale getirmez. Zira IP sadece ağ katmanında paket anahtarlamasına dayalı bir iletimin olmasını zorunlu kılar. Bunun dışındaki pek çok kararın, üstteki katmanlar tarafından alınmasına olanak sağlayan esnek bir yapıya sahiptir. Bu sebepten ötürü, eldeki kanalı en verimli şekilde kullanacak yapıyı oluşturmak uygulamanın sorumluluğundadır. Bu da ancak tüm katmanların özelliklerini göz önünde tutan bir yaklaşım ile mümkündür.

Bu tez de çoklu-görüntülü tasfirlere dayalı 3B videonun akıtılması için yeni bir yapı önerilmektedir. Önerilen bu yapı şimdiki standardlara dayalı, esnek ve önceki sistemlerle uyumlu olacak şekilde hazırlanmıştır. Bu sadece şu an kullanımda olan tek görüntülü aktarımlarla birlikte çalışabilmektedir. Çoklu-Görüntülü videonun (ÇGV) kodlaması sırasında basitleştirilmiş Çoklu-Görüntülü-Kodlama (ÇGK) sistemi kullanılmıştır ve basitleştirilme işlemi yüzünden kaybedilen kalite gözardı edilebilecek düzeydedir. Akıtım sırasında Gerçek Zamanlı Akıtım Prokolu (Real-Time Streaming Protocol) (RTSP), Oturum Betimleme Protokolü (Session Description Protokol) (SDP) ve Gerçek Zamanlı Taşıma Protokolü (Real-Time Transport Protocol) (RTP) kullanılmıştır. Ayrıca kaybedilen paketlerin yeniden oluşturulması için Raptor Kodlarından yararlanılmıştır. Buna ek olarak, geri kalan kayıpların neden olduğu görüntü bozulmalarının etkisini azaltabilmek için hata gizleme teknikleri kullanılmıştır. Sistem önceden ÇGK ile kodlanmış bir görüntünün, kod çözülmesi işlemini gerçek zamanda tamamlayıp, oluşan görüntüyü pek çok farklı ekranda gösterebilmektedir.

Bu tez ile aşağıdaki sorulara cevap vermeyi amaçlıyoruz. i) 3B görüntünün aktarılması için ne kadarlık bir bantgenişliğine ihtiyaç duyulur? Bunu etkileyen değişkenler nelerdir? ii) Dilimlemeyi aktif hale getirmenin getirdiği külfetleri, kazançları göz önüne alındığında gözardı edilebilir mi? iii) Ne kadarlık paket kayıp yüzdesi için, ne kadarlık kanal kodlamasına ihtiyaç duyulur? iv) Raptor kodlarının kullanılmasının gerektirdiği yük ile alternatif çözümlerinki kıyaslandığında hangisi daha iyidir. v) 3B görüntünün aktarılması için şu an kullanılan protokollere ne gibi değişiklikler gereklidir? vi) Kod-çözme ve görüntünün gösterilmesi işlemlerinde çok çekirdekli sistemlerden faydalanılabilir mi? vii) Gönderilmek istenen bilginin büyüklüğünden başka paket kayıplarını gözlemlenebilir ölçüde etkileyen bir değişen var mıdır? Eğer varsa bu paketleme seçimlerimizi etkiler mi?

ACKNOWLEDGEMENTS

I would like to thank to my advisor Prof. A. Murat Tekalp for his guidance and understanding. I would also like to thank to Prof. Reha Civanlar and Prof. İrşadi Aksun for beliving in me. I'm very thankful to Prof. Reha Civanlar and Prof. Serdar Taşıran for their teachings without which I could not have come this far.

Without my friends, it would be impossible for me to become the man I am now. Very special thanks to Burak Görkemli, Kagan Cetin, Umut Yıldırım, Gürkan Genç, Selçuk Erol, Mehmet Ali Yatbaz (a.k.a. MALİ), and many others at the Graduate School of Sciences and Engineering at Koç University.

Finally, I would like to thank to my family and my future wife. Knowing that there are people that will support me no matter what I do gave me the confidance, which I need to have to achieve this success.

TABLE OF CONTENTS

List of Tables ix

List of Figures x

Nomenclature xii

Chapter 1: Introduction 1

1.1 Background 1

1.2 Definition of Problem and Motivation 3

1.3 3D Perception and 3DTV. 4

1.3.1 3D Perception by Stereoscopy 5

1.3.2 3D Display Systems 5

1.4 Related Works 8

1.4.1 Monocular Streaming Applications 8

1.4.2 3D Streaming Projects 9

1.5 Contributions 10

Chapter 2: Multi-View Video Streaming Architecture 12

2.1 System Overview 12

2.2 3D Content Generation, Representation and Encoding/Decoding 16

2.3 Protocol Initialization 20

2.4 Video Streaming over UDP 26

2.5 Channel Coding 28

2.6 Multi-Threaded Decoding, Interzigging, and Display 29

Chapter 3: Experimental Results **37**

3.1 Encoding Results.. 37

3.1.1 Comparison of MVC to Simulcast Coding. 38

3.1.2 Cost of Enabling Slice Mode. 42

3.1.3 Effect of Encoding Scheme Set for Multi-Threaded Decoding . . . 43

3.2 Forward Error Correction Results.. 45

3.2.1 Packet Recovery 45

3.2.2 Redundancy Comparison between FEC and MDC. 47

3.3 Network Analysis 50

3.3.1 Packet Loss Behaviour 50

3.3.2 The Packet Loss Experiments 52

Chapter 4: Conclusion

4.1 Summary. 56

3.2 Future Work 58

Bibliography

Vita

LIST OF TABLES

Table 3.1: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Adile, PSNR 36.7) 41

Table 3.2: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Adile, PSNR 28.25) 41

Table 3.3: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Rena, PSNR 28.7) 41

Table 3.4: Overhead of Slice Mode 42

Table 3.5: Parameters for Forward Error Correction 46

Table 3.6: Possibility of Recovering a Lost Packet by FEC Decoding 47

Table 3.7: Streams in Redundancy Test 48

Table 3.8: Packet Loss Behavior During Low Bitrate Transmission 54

Table 3.9: Packet Loss Behavior During Comparable Bitrate Transmission 55

LIST OF FIGURES

Figure 1.1: Working Principle of Stereoscopy 4

Figure 1.2: Anaglyphic Picture 6

Figure 2.1: Modules of Proposed Streaming Architecture 13

Figure 2.2: MVC Standard Prediction 18

Figure 2.3: MVC Simplified Prediction 18

Figure 2.4: MVC Implemented Prediction 19

Figure 2.5: (a) Standard RTSP Message Exchange Order
(b) Modified RTSP Message Exchange Order 22

Figure 2.6: Convergance of TFRC rate to the available bandwdith 23

Figure 2.7: Segment of RTSP Message Exchange 25

Figure 2.8: Video Transmission over the Internet 29

Figure 2.9: Group of Frames (GOP) in simulcast encoding 30

Figure 2.10: Group of Frames (GOP) in MVC encoding 31

Figure 2.11: Required frames for watching stereo video. 32

Figure 2.12: Generation of independently decodable streams for the stereo case 33

Figure 2.13: Generation of independently decodable streams
for the multi-view case 33

Figure 2.14: Interzigging pattern for an 8 view display 35

Figure 2.15: An interzigged image 35

Figure 3.1: PSNR/Bitrate Comparison between MVC and
Simulcast Coding (Stereo) 39

Figure 3.2: PSNR/Bit-Rate comparison between MVC and
Simulcast Coding (8-View) 39

Figure 3.3: Multi-Threaded Decoding Scheme 43
Figure 3.4: Multi-Threaded Encoding Scheme 44
Figure 3.5: Internal structure of a Router 49
Figure 3.6: A Router in LAN Environment 51
Figure 3.7: Internal Structure of a Router 52
Figure 3.8: Packet Forwarding Model for the Internet 53
Figure 4.1: An Example on the Modularity of Live555 Library 59

NOMENCLATURE

3DTV	Three Dimesional Television
DCCP	Datagram Congestion Control Protocol
FEC	Forward Error Correction
HDTV	High Definition Television
JSVM	Joint Scalable Video Model
JVT	Joint Video Team
MCTF	Motion-Compensated Temporal Filtering
MD	Multiple Description
MDC	Multiple Description Coding
MVC	Multi-View Coding
MVV	Multi-View Video
NALU	Network Abstraction Layer Unit
QP	Quantization Parameter
RFC	Request for Comment
RTP	Real-time Transport protocol
SMDC	Scalable Multiple Description Coding
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
UDP	User Datagram Protocol
VPD	Video plus Depth

Chapter 1

INTRODUCTION

1.1 Background

The effect of the Internet over people's communication methods is undeniable. The earliest communication over the Internet was limited with the delivery of simple text messages. Later audio became a common component, and now it evolved to sessions where both audio and video are delivered. This progress was so rapid that the days when e-mail was perceived as an unreliable way of communication seem to be long forgotten. Now, people not only demand to be able to transfer multi-media, they also want it fast and smooth and apparently they will want it in 3D in the future.

Although 3-D illusions had people's imagination since 19th century, computers and links connecting them, recently became powerful enough accommodate 3D features. Developments in the last decade made it possible to implement a streaming architecture, which can actually meet the processing and storage demands of 3D video. One of the most critical improvements was the drastic decrease in the cost of computing and storage capacities. In 1950s, a single transistor cost around \$5 whereas it was even less then 2.5 millionth of it in 2004 [1]. Similarly in 1993, cost of 1 gigabyte storage was about \$550. In 2008, you can easily find it as low as \$0.2 in the market [2]. The amount of data that has to be processed for generating multi-view scene is much larger than its 2D counterpart, so the raise in capacity of processors and storage devices was a necessity. Meanwhile, the bandwidth of an average Internet connection increased significantly as well. In 1995, a typical connection was 33600bps modem over telephony lines. Today, 4Mbps (4194304bps) link is becoming standard connection. All those developments in the hardware made it possible to transmit and process multi-view video (MVV).

Developments in software were at least as crucial as the developments in hardware. A new understanding named as Application Layer Framing (ALF) had emerged for transmitting data over unreliable channel [3]. ALF involves abandoning the cumbersome and uncontrollable transmission policies of TCP and forming a policy based on the actual requirements of the application. With ALF, an application can interfere with packetization of the transmitted data. It creates independently decodable data chunks so that loss of a single packet does not make another packet, which probably arrived, useless. The application can also choose between retransmitting a lost packet or focus on degrading its effects by error concealment or correction algorithms. It can even ignore packet losses, if their effects are already negligible. Following the idea of ALF, streaming protocols were developed such as RTP [4], RTSP [5], SDP [6], and RTCP [4]. Also, a new protocol named as Datagram Congestion Control Protocol (DCCP) was proposed for standardization in March 2006 to cope with the rising risk of congestion due to increase interest in multi-media delivery [7]. Besides those developments in protocols, there were also improvements in encoding technology. In May 2003, ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) completed the standards of the H.264 video encoder [8]. Video encoders compress raw video data by exploiting the spatial and temporal redundancies in order to decrease the bit rate of the content. Size of a single uncompressed frame with 640 by 480 resolution is larger than 7Mbit. A 3D video with several views at 30 fps can easily require more than one Gbps to transfer. Since bandwidth is the most common bottleneck in a common streaming application the importance of video encoders should be obvious.

1.2 Definition of Problem and Motivation

Even with the developments at hand, there are still challenges that must be encountered in order to establish a successful multi-view streaming architecture. The key issue is to notice that, simply streaming multiple 2D views is not a good way of streaming 3D content. Unlike 2D case, there are multiple methods for representing 3D content. Protocols of the proposed architecture must be flexible and extendable to accommodate those different representations. It should also be able to support new representations without major modification.

Backward compatibility is another important aspect. Although the number of available 3D content is increasing, it is still at an incomparable level with its 2D counterpart. Demand for 3D displays depends on the number of available content. Supply of 3D content depends on the number of people who has 3D display and can view 3D content. So, if the backward compatibility is ignored, this vicious circle cannot be broken easily. However, if the system is backward compatible and people who do not have 3D displays can watch 2D version of the content, then the adaptation of 3D system can be easier. [9]

The visual quality evaluation more complicated when the delivered content is 3D [10-12]. In 3D, we have multiple parameters that define the overall quality of visual experience. The first is the SNR quality of video just like in the case of 2D content. Second and newly added parameter is the sense of depth. There has been a lot of research going on about quantifying the quality of 3D visual systems and they had two important observations regarding SNR quality and sense of depth. First one is that, if there is a difference between the SNR qualities of views, then the perceived SNR quality is decided based on the view that has higher quality. Secondly, the sense of depth depends more on the disparity views rather then their quality. The combination of these finding has critical impact over the rate allocation strategies for 3D video streaming. If we have a stereo video, it is better to dedicate larger bandwidth to one view instead of splitting it in half. By this way, we perceive both good SNR quality and sense of depth. [12]

Our motivation for this thesis is to build a standard based architecture that pays attention to concepts such as extendibility, flexibility, backward and human perception. It must be stated that as well as the theoretical research, a good level of programming skills is put into this study.

1.3 3D Perception and 3DTV

The goal of this thesis is to propose a streaming architecture for 3D content. So the reader should know what a 3D content is and how human can perceive it. This section is dedicated to explaining those questions and introducing 3D display technologies that are available today.

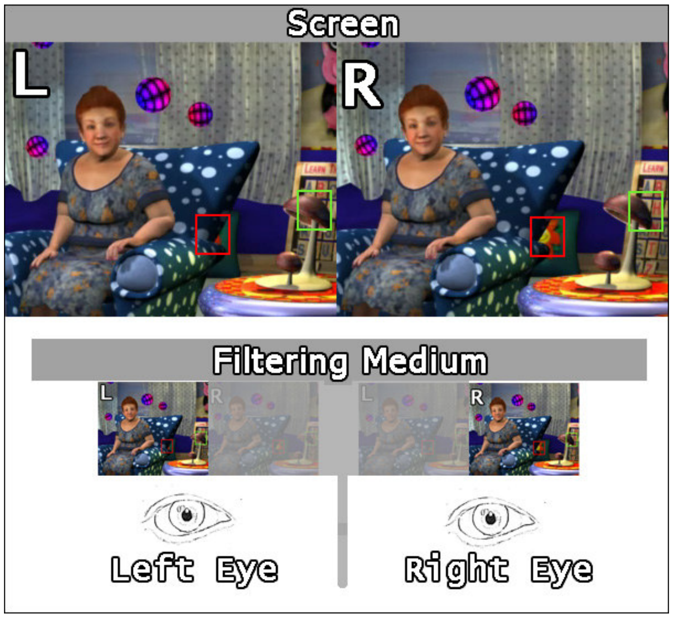


Figure 1.1: Working Principle of Stereoscopy

1.3.1 3D Perception by Stereoscopy

Most common understanding of 3DTV is a holographic image, which appears in the mid-air like in the movie Star Wars or Star Trek. Even though this can be the ultimate way of displaying a scene, currently we are far from this technology in terms of sharpness of the image. Today, the most common way of achieving 3D experiences is done by a method called stereoscopy. In stereoscopy, you have two distinct views of a scene recorded for each eye. The cameras that record this scene are calibrated to imitate the orientation of our eyes. Commonly, a filtering medium such as special glasses or on-screen lenses are used to direct only the correct view to the eye of the observer. If the calibration of the cameras is done correctly, then human mind can merge those images in a way that the viewer can experience sense of depth. An example of stereoscopic views is shown in Figure 1.1. Although these two views look very similar, red and green boxes outline some noticeable differences. When these two separate views are presented to their corresponding eye a sense of depth is perceived by the observer.

1.3.2 3D Display Systems

Although there is a rising interest in 3D entertainment recently, the work on 3D devices is older than expected. Method of stereoscopy is dates back to 1838 when Sir Charles Wheatstone, a British researcher and inventor, developed a device that could display two different views of a scene by using mirrors [13]. In 1903, the first short movie in 3D was displayed in Paris, with modified stereoscope allowing only one person to watch at a time. Later in 1922, first full length 3D movie was demonstrated to the audience with anaglyphic glasses. However, the introduction of 3D to broadcast was delayed until 1950s and never gained much popularity. Neither 3D cinema nor 3D broadcast was successful, due to the poor visual quality of anaglyphic display systems and side effects such as headache and eyestrain [14, 15]. In Figure 1.2, you can see an example of a 3D image, which can be viewed by red/blue anaglyphic glasses. You should notice that due to color filtering, the original color information of the scene is lost.

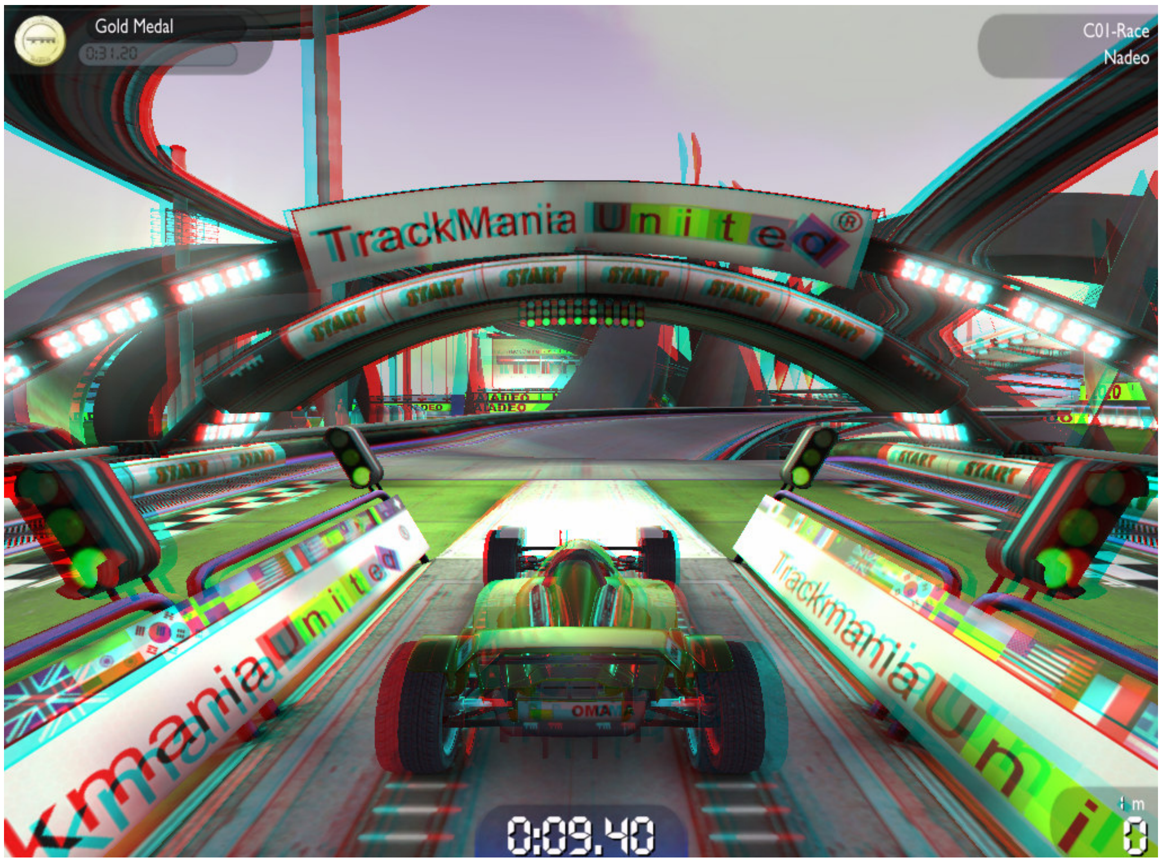


Figure 1.2: Anaglyphic Picture

The recent rise in the interest in 3D entertainment systems is probably due to the increase in offered visual quality. One of the newly introduced methodologies for stereoscopy is based on using polarization filtering. Special material polarize the light from emitted from two projector in different orientations. Then the polarized lights are projected on a dielectric screen that reflects the light without losing its polarization. The user wears glasses with special filters allowing only one view per eye. Since the separation of views is not done by color filtering, the color information is conserved. Filtering polarized radiation is currently the most commonly used technology in 3D theatres.

Another 3D system uses stereo parallax barrier stripes over LCD displays. The parallax barriers block the light from certain pixels to certain directions. This allows the user to watch the stereo content without wearing glasses but requires the observer to stay at a particular point in order to receive the content correctly. The frames from each view are reordered based on the location of the parallax barrier. This operation is called as interdigitizing and implemented at sub-pixel level.

Yet another 3D display technology depends on lens array that divert the light instead of blocking it. The lens array is called lenticular sheet and can be either fixed on LCD display during manufacturing or used as an external device. Similar to the parallax barrier technology, displays with lenticular sheets does not require the glasses but they should be watch from a certain point. In order to distribute the pixels based on the lenticular sheet, the interdigiting operation is performed over the frames with a different pattern.

Using only two views of a scene, the viewer can only experience 3D for a single point. Multi-view displays allow the viewer to see the scene from different directions by moving around. This time lens arrays are used to reflect lights to a particular direction. By this way the user can move around and receive different consecutive views of the scene.

Lastly, head mounted systems and head tracking systems based on the stereoscopy principle are also used. They track the movement of the head and then display views correspondingly. They provide free-view interactivity with real time responsiveness. [16]

1.4 Related Works

1.4.1 Monocular Streaming Applications

It has been a while since the introduction of ALF. Many streaming applications emerged since then. While they can be differentiated based on their transmission policies, all of them are developed for purpose of to playing video on the fly. Currently the most famous video-on-demand sites can be listed as YouTube [17], Google Video [18], and AolVideo [19]. Although the software used in these servers, can be categorized as video streaming applications, they use TCP as their transport layer protocol. Although TCP barely meet the standards of ALF, it is the only practical solution that can be used with all types of connections. UDP traffic is blocked by network administrators occasionally. Also the users behind Network Address Translator (NAT) cannot receive UDP packets even if the traffic is allowed due to the connectionless nature of UDP. Therefore, people that use a shared connection in a network would have difficulties in receiving UDP streams. As a results, in a system that use TCP the viewers experience frozen screens more frequently compared to UDP based transmission. This is due to the lossless transmission policy implemented by TCP protocol, which requires the retransmission of all lost packets even though some of them can be discarded with negligible loss in visual quality.

Apple developed a product named as Darwin Streaming Server [20], which uses RTP over UDP and RTSP protocols for data transmission and signaling respectively. Similarly, VideoLAN is another streaming application, which can serve as both sender and receiver for a streaming environment [21]. VideoLAN uses Live555 as network library for implementing protocols such as RTP, RTSP, and RTCP. Live555 is another open-source project that we use in our streaming system.

1.4.2 3D Streaming Projects

Capture, representation and rendering real-life scenes are hot topics, which are addressed by various research programs such as ATTEST project[22] and 3DTV NoE[23] in Europe or FTV[24] project in Japan. These research programs have already produced mature results. However, there has been relatively little research on the transmission aspects of 3D video. In the following paragraphs, we will present the current state of the research projects about the transmission of MVV.

Mitsubishi Electric Research Laboratories (MERL) initiated a 3DTV project in 2004 [25]. They were able to transmit high resolution (1024x768) videos of 16 cameras over gigabit Ethernet. They had used an array of cameras and PC for real-time accusation and encoding. At the receiver side, they had decoded and displayed the views for multiple types of display [26,27]. Although at first glance their system seems to be complete, there are critical missing parts, which make it impossible to implement over the Internet. It is very difficult to encode high resolution video in real time. This is the main reason of using Gbit Ethernet and local area network. Secondly, in LAN, they had no need of error concealment or error correction feature, which is another key aspect for a complete streaming system over the Internet. Lastly, the handshaking protocol seems to be a custom one making it difficult to interact with the current applications that are in use. Nevertheless, their system forms a basis for broadband 3D transmission application

Electronics and Telecommunications Research Institute (ETRI) developed another ambitious project about 3D HDTV over ATM [28]. The system was fairly similar to the MERL’s implementation but it was composed of more advanced equipments. They used polarized projector system, since currently there is no lenticular display or parallax barrier, which can accommodate HD resolution. The HD content was streamed over an ATM connection at 155 Mbps. Projects of ETRI and MERL had similar objectives such as transmitting 3D content in case of abundant available bandwidth. Neither of them was intended for the common user over the Internet.

Video Coding and Architectures (VCA) group at the Eindhoven University of Technology has proposed a streaming architecture for 3D-IPTV. They have designed and implemented a stereoscopic and multiple-perspective video streaming system [29]. However, their work was at the elementary level.

Lastly, there are two other FP7 projects. FP7 MOBILE3DTV project addresses the research towards the delivery of 3DTV over DVB-H [30]. The project intended for developing the optimal representation and coding formats for stereo video and its robust transmission over DVB-H utilizing novel, stereo-video dedicated UEP schemes. 3DTV broadcast solutions are studied in the 3D4YOU project [31]. It favors the "video+depth" representation format as the most broadcast-friendly one and aims at developing format conversion tools and carrying out broadcast demonstrations.

1.5 Contributions

The contributions of this thesis can be summarized as follows;

- Complete streaming architecture is proposed and implemented. It is demonstrated in several high-profile events, such as the IBC Amsterdam in September 2007. The system was one of the most successful demonstrations of NoE, 3DTV project.
- We had investigated with encoding schemes to determine the required bandwidth to transmit MVV. We focused the performance of MVC coding in particular and compared it with legacy H264 encoders.
- We had measured the cost of enabling slice mode during the encoding process. We evaluated the gain in the bit-rate by disabling slice mode versus introducing FEC when slice mode is enabled.

- We performed tests on Raptor Codes and determined the amount of channel coding required under variety of network conditions.
- We had compared the redundancy of Raptor Codes with a proposed MDC model.
- We framed the required modifications upon the current streaming protocols to accommodate the extended data exchange required for multi-view streaming.
- A novel module is proposed for allowing multi-threaded decoding with decoders that are designed for single thread. This is especially important for streaming MVV in real-time.
- We had defined the effect of packet rate as the second most important parameter for the packet loss rate.

Chapter 2

MULTI-VIEW VIDEO STREAMING ARCHITECTURE

2.1 System Overview

Modules that compose 2D and 3D streaming system have similar objectives. However, almost each one of them is more complicated in the 3D case. In Figure 2.1, presents the order and the list of these modules. In this section, we will first describe the overall working principles of the implemented 3D streaming system. Then, we will give detailed information about each module.

The initial task is to generate or capture the content. A variety of methods is present in the literature for acquiring 3D content [25], [32-34]. The first one is capturing the scene from multiple viewpoints. For this purpose, you can either use an array of cameras or a single camera with multiple objectives such as Bumblebee [35]. Another option is to generate 3D content using standard 2D video. A very successful algorithm for a static scene captured by a moving camera is already proposed and demonstrated in HHI headquarters [36]. Yet another method depends on the estimation of the depth values of pixels and then generating artificial views. Beside the special equipments, which provide the depth-map, one can also use multiple cameras and estimate depth-map by protection. Lastly, you can generate 3D content by using computer graphics technology. Taking snapshots from different locations is a straightforward procedure when you have the 3D model.

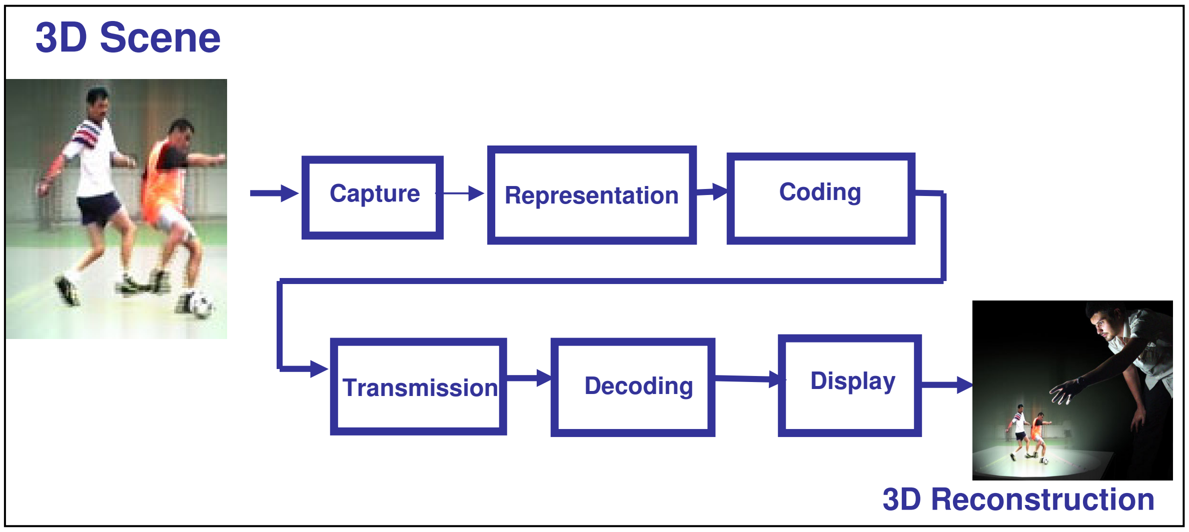


Figure 2.1: Modules of Proposed Streaming Architecture

Once the content is available, a representation format is required to store the visual data. Two common formats are present for storing raw video. In the first one, each view is stored in a separate stream. This allows accessing every captured frame without any information loss but requires high storage capacity. In video-plus-depth format (VPD) in addition to the video stream, the depth map information is stored. The depth map requires less storage area and utilizes generation of artificial views. Therefore, it is not necessary to store every view with depth-map representation. However, it is not possible to recover all of the frames exactly. The occlusions and semi-transparent object cause difficulties in generating correct artificial views. As a solution, a new method that uses *N* video-plus-depth streams to generate M views, is proposed (M > N). There is an ongoing study for the standardization of this representation as well [37].

The next step is to compress the 3D data before transmitting over the Internet. There are two fundamental solutions for encoding MVV. In simulcast encoding, each video stream is considered independent video stream and encoded one by one. In standard monocular video encoding, spatial redundancy and temporal redundancy are targeted to decrease the bitrate of the content. The current state of art encoding standard is H264 Advanced Video Coding (H264/AVC) defined in [38]. MVV introduces interview redundancy, which is due to high correlation between views. In simulcast coding, interview redundancy is not exploited and independently decodable video streams are generated. Multi-View Coding (MVC) [39] is and extension of H264/AVC and it is developed to take advantage of the interview redundancies. MVC takes advantage of similarities between different views by referring frames not only in time axis but also in views. These dependencies, decreases the overall size of the content, but makes the decoding process more complicated. A frame in a dependent stream cannot be decoded without receiving all of the referred frames in other views. As a result, MVC decoding mandates buffering mechanisms to feed the decoder in correct order.

A compressed video stream is packetized using ALF conventions and streamed via well-known protocols such as, RTP, RTSP and RTCP. RTSP is an out of band protocol for signaling and handshaking purposes. RTSP runs over TCP and offers remote control abilities such as play, pause, rewind, and stop. It also grants handshaking mechanism by getting the description of the video using SDP. Handshaking allows the receiver to understand whether the content is suitable or not before actual data transfer. A video content may not be compatible with the client due to variety of reasons such as missing codec. In 3D, the display requirements of the content can be another reason for incompatibility. Once the handshaking is over, RTP is responsible for the actual data transmission. As a control mechanism, RTCP handles the report exchange between server and client. The server may adjust to network conditions based on the information available in these reports. In section 2.2, we introduced our proposed modifications for RTSP and SDP protocols.

Once the handshaking is over, the server initiates transmission and the receiver can start decoding the incoming packets. During the video transmission, the available bandwidth is divided between source coding and channel coding. Channel coding enables the receive to recover lost packets using FEC mechanisms. The optimum rate for channel coding depends on the characteristics of the link. As an example, every bit dedicated to channel coding is a waste in a lossless medium. However, experiments show that, packets are lost due to early discard mechanism employed by the routers even if the transmission does not exceed the available bandwidth. Therefore, dedicating a reasonable rate for channel coding improves the overall quality of the transmission.

As the packets arrive at the client, they are stored at the buffer and forwarded to decoder when all the required packets received. Generation of 3D scene is the last part of the streaming. In the 2D case, basic operation is to display the decoded frames at a steady rate. In 3D, both the content representation format and the features of the display have affects on the scene generation. In video-plus-depth representation, we need to generate the artificial view(s) before any other operation. If the frames are ready, we may project them on screen immediately, like in the case of projector based display systems. In some display systems, we have to merge all frames at a time instant generate single image for an instance. Due to the difference in the orientation of lens array, the pattern for merging (Interdigitizing) views differs from one display to another. Once the interdigitizing operation is over, we simply display frames based on their timestamps.

In the paragraphs above, we have summarized the modules that compose a complete streaming architecture. During the implementation, we focus on both individual performance of each module and the efficiency of the interfaces between them. In theory, the increase in performance in any module should increase the overall performance. However, if the interfaces between modules are not efficient, it may not be possible to observe the expected performance gain. Therefore, a successful implementation should consider collaborative efficiency of modules as well as their stand-alone performance. When compared with 2D systems, structure of each module is more complex In order to meet the demands of such a rapid growing technology, the modularity of design is crucial. Otherwise, the streaming software can become obsolete with in a short period.

2.2 3D Content Generation, Representation and Encoding/Decoding

In our tests, we used contents created by several methods in order to investigate the encoding performance and visual quality of 3D experience. The first one is a computer generated artificial model of Adile Naşit by Momentum, a company that was also member in the 3DTV project. The second content, labeled as Iceberg, was an example of 2D to 3D conversion. BBC did the original shooting and the artificial views generated by Sebastian Knorr at the Technical University of Berlin [36]. In the scene, there are no mobile objects, so that it was possible to generate an artificial view based on the motion of the camera. Third content was captured in Japan with a 1D-camera array that composed of 100 cameras. Finally, Microsoft break-dance record was the last content and used for testing the depth map feature of the application.

The features of content like the amount of motion and the ratio of high frequency components affect the encoding performance of a codec. The introduction of multiple views adds even more complexities such as white balance and focus settings of the recording camera setup. Naturally, it is expected that, encoding artificially created content by 2D-3D conversion yields better results compared to an actually captured video due to absence of difference between cameras. Even better results can be achieved in a completely artificially generated computer graphics.

Encoding structure of MVC may need more explanation. In Figure 2.2, you can see the proposed MVC prediction algorithms. Horizontal axis represents the flow of time and the vertical axis is the view id. Each arrow in the figure corresponds to dependency. The arrows mean that the target frame is taken as a reference by the source frame. As a result, dependencies generated since target frame is require decoding the source frame. Figure 2.2

describes full dependency between frames of multi-content. In this architecture, each frame is dependent to all the frames in adjacent views in addition to the frames in the same view. While it may increase the performance encoding, it introduces complexities for the decoding process. Yet another side effect is the propagation of distortion due to packet losses. In Figure 2.3, a simplified model is proposed. In the simplified model, the interview dependencies are restricted to only I-Frames for each GOP. By definition, an I-Frame is independently decodable. Once an I-Frame refers to frames in another view, it is no longer decodable independently. We label such frames as key-frame. Figure 2.4 displays the dependency structure that is adopted in our implementation. At the Figure 2.3 taken from [40], you can see the comparison of encoding performance of those models. Notice that the simplified model performs very close to the complex model. Considering the side effects of complex model, simplified case forms the basis of our model. In both standard and simplified prediction, the view 0 is independently decodable. In our model however, the independent view is in the middle. This scheme introduces simpler dependencies for stereo displays.

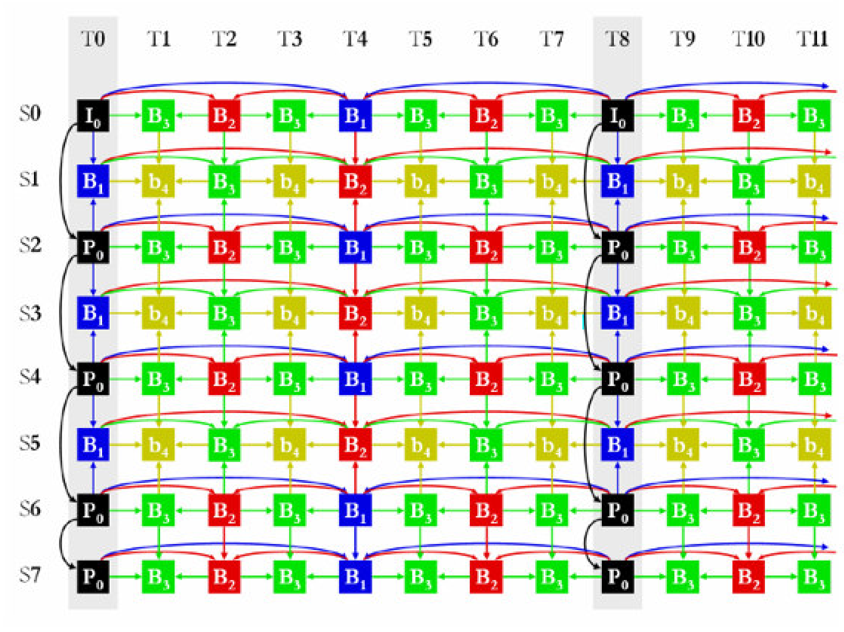


Figure 2.2: MVC Standard Prediction

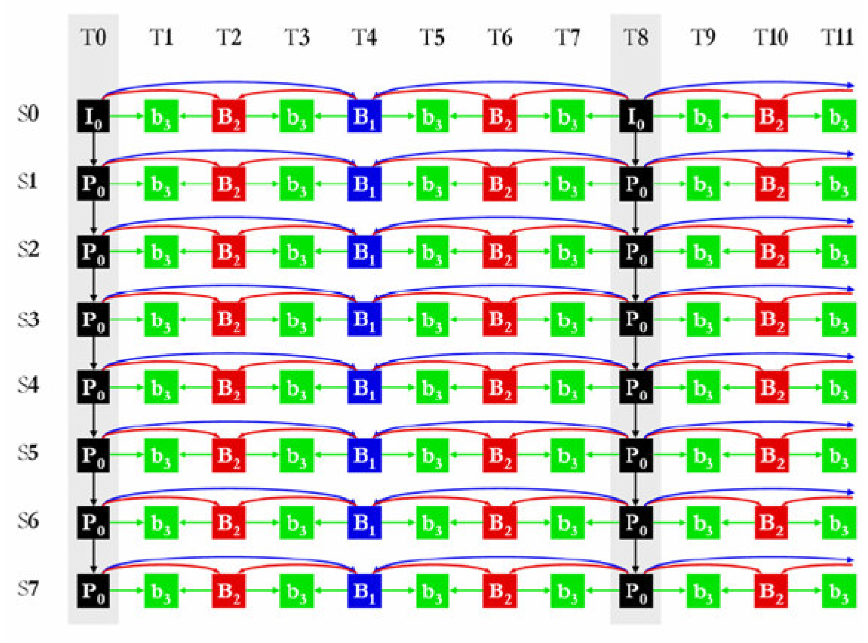


Figure 2.3: Implemented Prediction Model

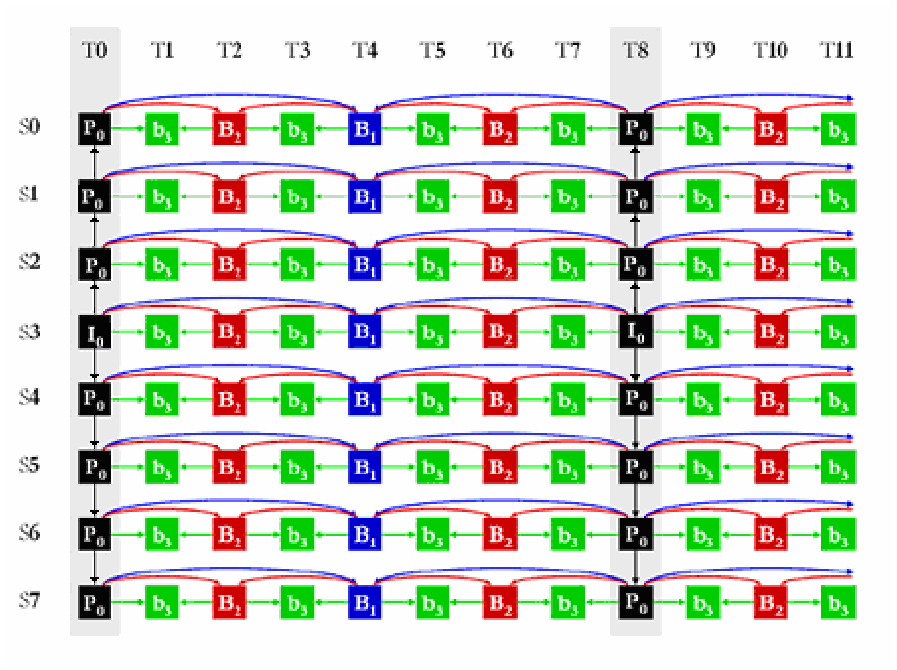


Figure 2.4: MVC Implemented Prediction

2.3 Protocol Initialization

We use client driven Real Timer Streaming Protocol (RTSP) defined in [5] for the handshaking and initial setup. RTSP is a text-based, extendible protocol and runs over TCP for lossless data exchange. RTSP is not intended for actual data transfer, instead it allows remote control capabilities to the client such as play, pause, rewind, and stop. RTSP is not a connection-oriented protocol meaning; A client can use distinct TCP connections to issue RTSP requests during a single session. However, the server does keep session information in order to respond the requests correctly. The diagram in Figure 2.5 exemplifies a typical RTSP message traffic. In Figure 2.5 (a), you can see the standard RTSP while in (b) you can see modified version that we use in our proposed architecture. In the rest of this section, we present our proposed modifications for handshaking procedure and session setup.

The first expected message in an RTSP session is describe. The client interprets the response to understand whether it is suitable to play or not. The response is in the format defined by Session Description Protocol. Our proposed modifications for RTSP and SDP protocols are as follows;

SDP Modification: In the proposed system, each view contained in the multi-view content is declared as a separate media-level section in the SDP announcement. This allows the receivers to select the views they would like to receive. Although the output of MVC is a single compressed file that includes NAL units of all views, we propose to break this single file into separate streams for each view. While this has no effect on the total file size, it facilitates fast disk access to the NAL units belonging to different views. Obviously, most of these streams will not be independently decodable due to the inter-view dependencies generated by MVC compression. Therefore, we extend the SDP standard by adding a new media-level attribute called x-ref in order to declare dependencies between streams. In accordance with [6],this new attribute is used as a=x-ref:REF LIST, where REF LIST is a space separated list of rtpmap values for the streams referenced by this stream, Rtpmap is a dynamically assigned RTP payload type identifier for a H.264/AVC stream and is unique for each view. Therefore, REF LIST defines all the required views for which the REF LIST belongs. The client should go through the list of references in a recursive fashion and initiate streaming for all streams in the dependency hierarchy. If REF LIST is an empty string or there are no x-ref attributes for a media-level section, it means that the stream in question does not depend on any other streams and can be decoded independently. That completes our modification for inter-view dependency.

The second modification for SDP was the use of depth maps that are required to generate artificial views. In case depth map streams are present, we tag those streams in order to distinguish them from the video streams. We propose to accomplish that using another SDP attribute in addition to x-ref. Let us call this boolean attribute x-depthmap. For any depth map stream both x-depthmap and x-ref need to be present. x-depthmap declares the stream as a depth map stream and x-ref denotes the stream that depthmap is indented for. Obviously, the client has to assume the SDP announcement generated by the server is correct in the semantics as well as well in syntax, i.e. correct depth streams need to be associated with the correct video streams in order to achieve meaningful rendering at the client side. This method of declaring dependencies, is completely backwards compatible. In accordance with original SDP specification, a client, which does not understand the proposed SDP attributes, ignores them. However, such a client might request and fail to decode dependent streams due to its inability to make sense of dependency information. Similarly, a 3-D client can still display in 2-D if it is connected to a conventional video server with no support for the proposed SDP attributes.

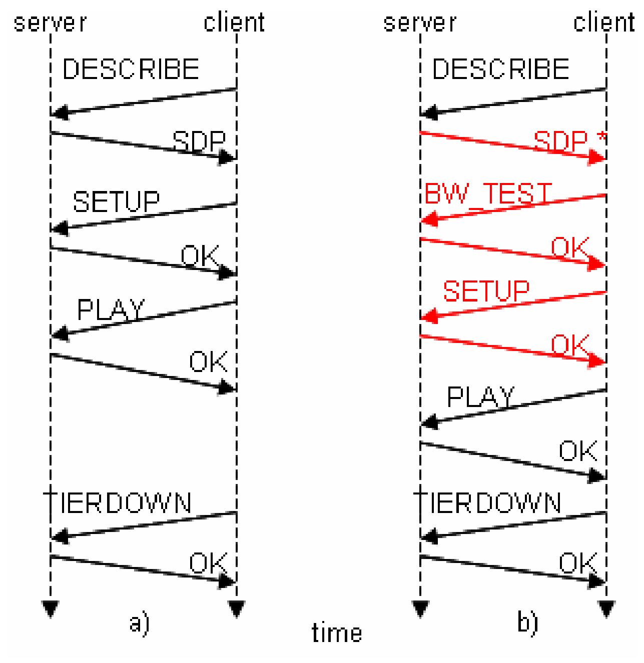


Figure 2.5
(a) Standard RTSP Message Exchange Order
(b) Modified RTSP Message Exchange Order

RTSP Modification: Amount of bandwidth requirements of 3D Video, especially in eight or nine views, can be overwhelming for client's connection. It is best if the server can give feedback on the connection quality of the client, and state whether it is adequate to play the content or not. For this purpose, we include a new phase in the RTSP message exchange list in which the server will try to estimate the bandwidth currently available to the client. In this optional phase, we use the TFRC rate adaptation formula, defined in [41], for measuring the available bandwidth between server and client. This procedure is initialized by a request from the client, containing both the list of streams that are about to be played and the duration of the test. Figure 2.6**,** generosity of Burak Görkemli, displays the average time elapsed until the calculated rate converges to the actually available bandwidth. The figure shows that TFRC rapidly converges to the available bandwidth. Duration of five to ten seconds is adequate for measuring the bandwidth of the client. Server replies with a recommendation based on the total bandwidth requirement of the requested contents and the calculated available bandwidth. It is the decision of the client to follow or ignore the recommendation of the server.

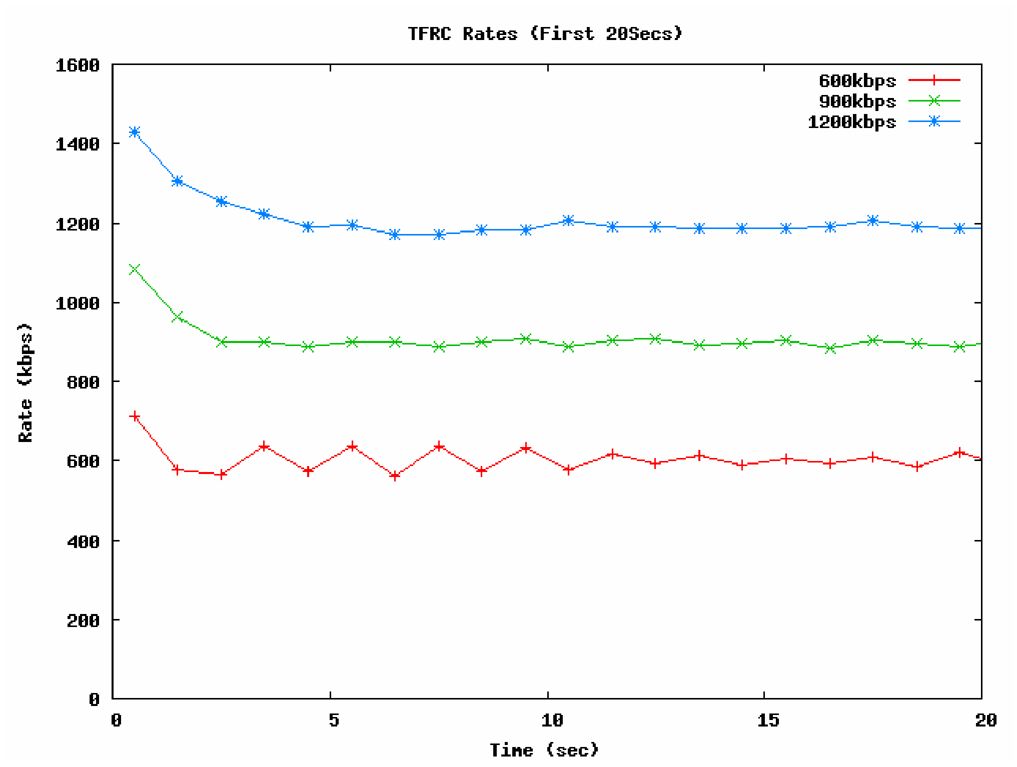


Figure 2.6: Convergance of TFRC rate to the available bandwdith

Once the client interprets the messages in SDP and decides to play the content, it advances to the next phase of RTSP. In this setup phase, the client and server declare communication parameters such as port number of both actual data transfer (RTP) and reporting (RTCP). In our architecture, the actual data transfer is done over RTP/UDP/IP protocol stack, which works on best effort principle. In order to struggle with possible packet losses we use Raptor Codes as Forward Error Correction mechanism. The forward error correction layer can be present above and below RTP in the protocol stack. In our case, once the initial parameters are set, the sequence number field on the RTP header is adequate for tracing the FEC packets. Therefore, if FEC packets are payload for RTP is possible to use Raptor Codes without additional header. However, a modification in SDP is required for setting the initial parameters of raptor decoding operation.

In Figure 2.6, you can see a typical SETUP request and response during an RTSP message exchange. The bold texts are due to the modifications we propose in the setup phase. In our architecture, the client can determine the rate of protection for each stream separately, allowing unequal protection between video streams. The independent stream is clearly the most crucial for the decoding process. Therefore, the client may wish higher level of protection for the independent stream and less for the ones that depend on it. **fecRate:numProtectionSymbols-numTotalSymbols** syntax fully describes the rate allocation between source and channel coding. The term block refers to a group of source packets along with the protection packets generated to protect the source packets. As the name suggests, **numTotalSymbols** is the total number of packets in a block. The **numProtectionSymbols** is the number of protection packets present per single block. The higher the ratio of protection symbols to the total symbols means higher level of protection for the stream. Section 2.5 describes the effect of those parameters.

```
Client ➔ Server
SETUP rtsp://88.255.97.55:8555/adileLong2/track1 RTSP/1.0
CSeq: 2
Transport: RTP/AVP;unicast;client_port=3740-3741;fecRate:30-400
User-Agent: adileLong2 (LIVE555 Streaming Media v2006.12.08)

Server ➔ Client
RTSP/1.0 200 OK
CSeq: 3
Date: Tue, Sep 09 2008 12:03:29 GMT
Transport: RTP/AVP;unicast;destination=127.0.0.1;client_port=3740-
3741;server_port=5002-5003;fecRate:30-400;fecSize:1430
Session: 2
```

Figure 2.7: Segment of RTSP Message Exchange

2.4 Video Streaming over UDP

We generate a byte-stream by encoding raw video. It is composed of data chunks, named as macro-blocks, which include information about a limited number of pixels in a frame. The term slice is a group of macro-blocks and header for the common parameters of the macro-blocks. As we decode slices, we get portion of a frame. The word "streaming" refers to delivering of those slices in timely fashion.

RTP header wraps slices and includes that includes sequence number and timestamps. Based on the sequence number, receiver can detect losses and may trigger retransmission. It may also run error correction algorithms to recover that lost packets if channel coding is present. It may also choose to ignore the packet loss event. The next field in the RTP header is timestamp, which defines the instant that the payload is intended to play.

TCP, UDP, and DCCP are the transport layer protocols that are commonly used network applications. DCCP is a protocol still under development and is not implemented by all OS. TCP can provide reliable data transfer over unreliable channels. However, streaming applications prefer UDP for its customizable policies. At this point, it may be unclear why an unreliable transport is preferred over a reliable one. The fundamental benefits using UDP during video transmission are summarized as follows;

- **Application Layer Framing**: In TCP as the transport protocol, the application cannot participate in packetization. It is not possible to select the exact bitstream within each packet. Therefore, it is very likely that the packets will contain data that can be nullified when a particular packet is lost. This policy is a clear violation of ALF.
- **Retransmission Policy:** TCP is designed for lossless data transmission for files that get corrupt in case of information loss. However, multi-media streams are tolerable to losses. Consider a TV show display at 30 frames in a second. Loss of a single frame is most likely to be unnoticed by the audience. Since UDP does not enforce retransmission of lost packets, it is possible to ignore that lost frame and keep the video playing. However, TCP will try to resend that lost frame and halt the stream if unable to do. As a side effect, the video is paused and the quality of the visual experience is degraded. UDP allows the application to choose the action in case of packet loss. The application may choose to ask the server to retransmit the packet, or try to run error correction algorithms or do nothing and try to hide its side effects.
- **Steady Transfer Rate**: TCP is designed for transmission of bulky data and it is under the influence of two driving forces, bandwidth utilization and congestion control. The former one dictates indefinite increase in transfer rate to increase the throughput of the channel. However, this eventually forces a packet loss at some point since the channels have a limited capacity. On the other hand, the congestion control policy try to avoid congestion by demanding drastic reduction of transfer rate in case of a packet loss, which is unavoidable due to first policy. Under the influence of these forces, rate of data transfer displays

fluctuations. Unlike bulky data transmission, a streaming application prefers steady transmission rate, thus UDP is more preferable.

We propose to stream multi-view content over RTP/UDP/IP in a similar fashion to [14]. In the case of simulcast streaming, views are independent of each other so it is not different then well-known 2D case. For the case of MVC encoding however, we have a single file that contains slices from all views. As stated previously, we propose to split this file, to multiple streams, each containing slices from single view. Then the generated streams are compatible with RTP/UDP/IP stack and the receiver can identify them based of the dynamic payload type in the RTP header.

2.5 Channel Coding

In the previous section, we explained the transmission of video slices. Now we will describe the Forward Error Correction mechanism, which initiated when a data packet is lost. The term `forward` stands to the fact that, we generate the protection packets before actual data transfer and obviously before the occurrence of any packet losses.

XOR operation forms the basis of packet generation in almost all FEC algorithms. For the simplest case, assume that we have data packets A and B. We try to transmit A, B and C where C is A XOR B. The client can reconstruct A and B as long as it receives two out of three packets in any combination. This simple scheme has the two fundamental problems. First, the protection system is useless in case of two consecutive packet losses and secondly the protection packets introduce 50% bandwidth overhead. Therefore, advanced FEC techniques are developed.

Among many alternatives, we used Raptor Codes with is an implementation of Fountain coding. Fountain coding is a novel technique that became recently popular as Error Correction algorithms for streaming applications [42-44]. Some practical implementations are can be listed as Online codes [45], LT codes [46], and Raptor codes [45]. Fountain codes require byte chunks, named as symbols, at fixed length. The slices generated by video encoders are good candidates for this requirement. However, due to the nature of video encoding size of slices are not fixed. Nevertheless, we can overcome this problem simply by padding zeros at the end of NAL units.

A raptor code is composed of two parts. The encoder resides at the server side and generates the protection symbols in real-time once the data symbol are available. The decoder, resides at the client side, and tries to recover lost data symbols using the information available in the protection packet. During the transmission, we feed a predefined number of data symbols to the FEC encoder and receive protection symbols. The data symbols, together with the corresponding protection symbols are called a block. It is the task of the server to deliver the symbol blocks to the client. Upon receipt of a single symbol block, the client has does the following operations; if no data symbols is lost then it simply ignores the protection symbols. If one or more of the data symbols are lost, it initiates FEC decoding process and tries to reconstruct (decode) as many data symbols as possible. Figure 2.7 summarizes the FEC encoding and decoding process.

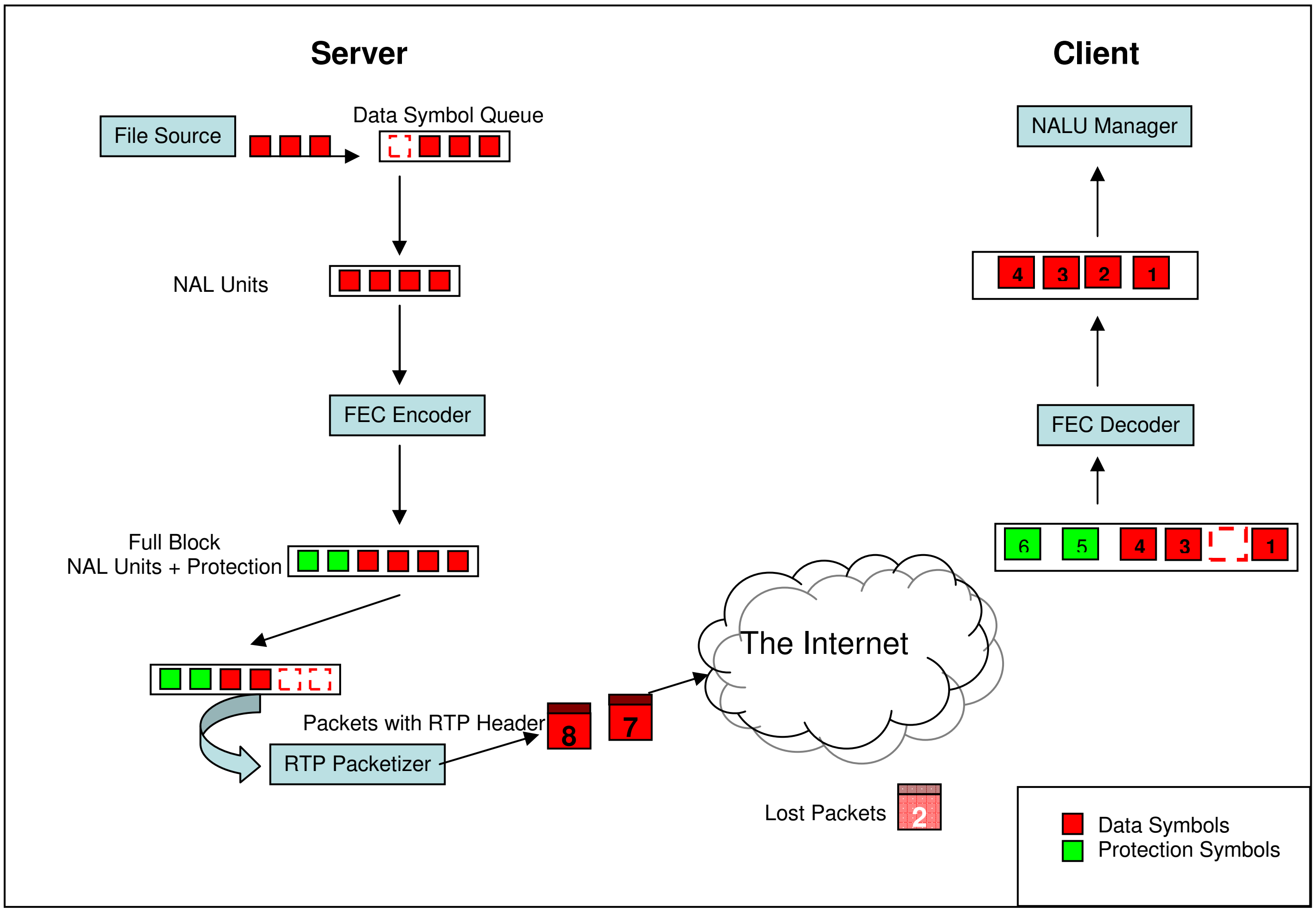


Figure 2.8: Video Transmission over the Internet

2.6 Multi-Threaded Decoding, Interzigging, and Display

Once the channel coding is over, we strip the NAL Units from their RTP header and forward them to NALU Manager. NALU Manager is the module that is responsible for queuing and decoding operations. It has multiple threads for performing operations simultaneously if multiple cores are present. Once the stream is processed by NALU

Manager, the final task is to display the frames in a timely fashion. Therefore, it is correct to say that NALU Manager is the final link step of the multi-view streaming system.

2.6.1 NALU Buffering

Queuing is required to synchronize the frames of separate views, which get more or less unsynchronized due to the nature of the packet transferring mechanism in the Internet. Queuing also utilizes the decoding operation if inter-view dependencies introduced by the MVC encoder.

We insert intra-coded frames (I-Frames) periodically during simulcast encoding. An I-Frame does not depend on any other frame therefore, it is independently decodable. The frames between two consecutive I-Frames are called Group of Pictures (GOP). A frame within one GOP can not refer to another frame in another GOP. Therefore I-Frames also work as a boundary point in the dependency hierarchy. Yet another meaning is that in simulcast coding GOP is independently decodable as long as I-Frames a received.

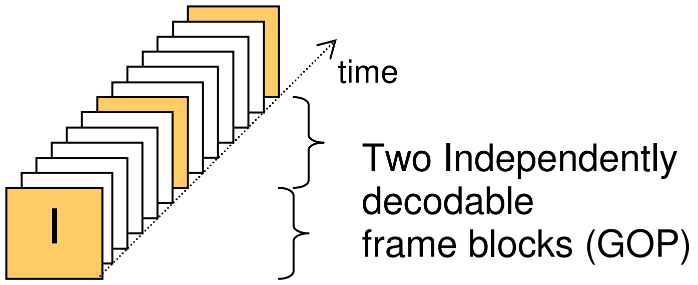


Figure 2.9: Group of frames (GOP) in simulcast encoding

However, MVC introduces inter-view dependencies defined for I-frames, which are now called key frames. As a result, in order to achieve independently decodable block all dependent frames should be received beforehand. The dependency hierarchy is summarized in Figure 2.9. Based on this hierarchy, in order to decode GOP in view 2, I-Frames in view

3 should be received. Likewise, in order to decode the GOP in view 5, key frames in view 4 and I-Frames in view 3 should have arrived.

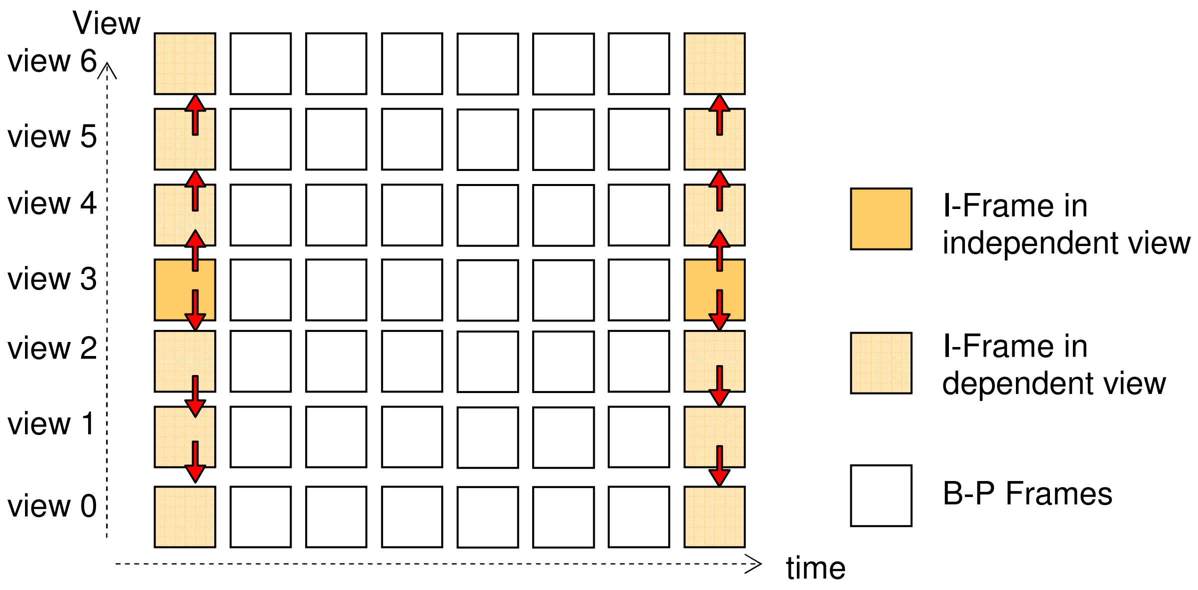


Figure 2.10: Group of Frames (GOP) in MVC Encoding

If the client requests less then the available views, we can use the dependency hierarchy to stream only the required frames. The ability to stream the content selectively allows the client to adjust the disparity, which is the difference between left and right views due to orientation of cameras. This is especially important if we consider that, the optimum disparity depends on the display and may differ from person to person. Figure 2.10 specifies the requires frames for decoding the GOP in view 0 and 5.

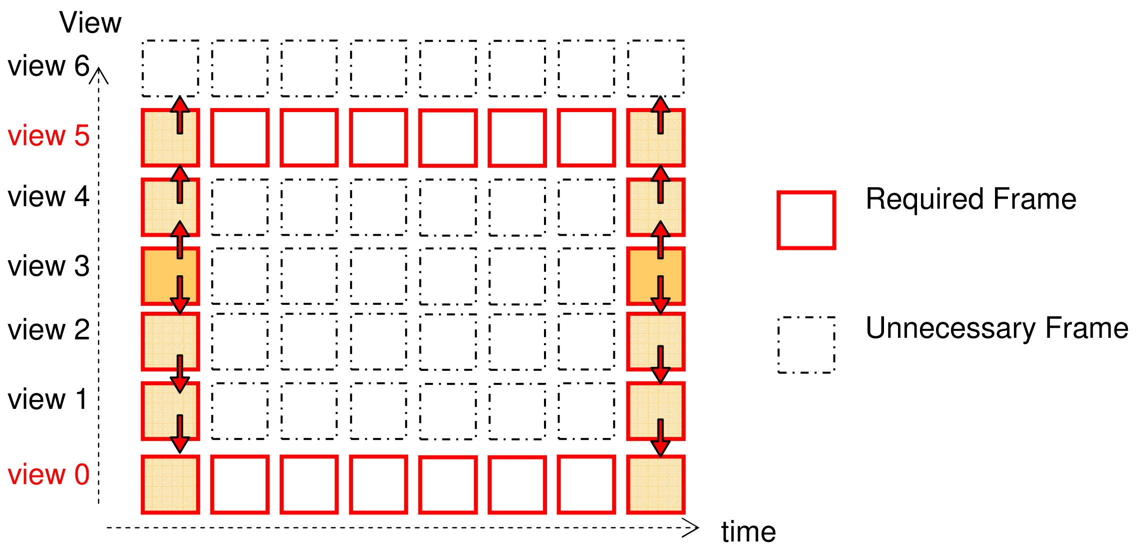


Figure 2.10: Required frames for watching stereo video.

2.6.2 Multi-Threaded Decoding

In our streaming system, we dedicate most of the computational resources for the decoding operation of NAL Units. During the transmission for 9-view, clients has to decode 270 frames per second in order to reach the 30fps limit. Due to the structure of the decoder software, it cannot take advantage of multi-core processors automatically. Therefore, we propose a systematic methodology for simultaneous decoding operation.

As a way of implementing multi-threaded decoding, we propose to initiate multiple instances of decoder objects. Once we create the instances, what remains is feed them with independently decodable byte-streams. The highlighted frames In Figure 2.10 describe two independently decodable streams. If these frames are fed into the decoder, then simultaneous decoding operation can be performed.

The task of creating multiple instances of decoders and multiple threads is a straightforward implementation. However, creating the independently decodable byte-streams requires special attention. For this purpose, we propose a new object named as GOP-Manager, which is responsible for generating independently decodable byte-streams. Given the inter-view dependencies, GOP-Manager tries to create two or more streams that has all of the required frames.

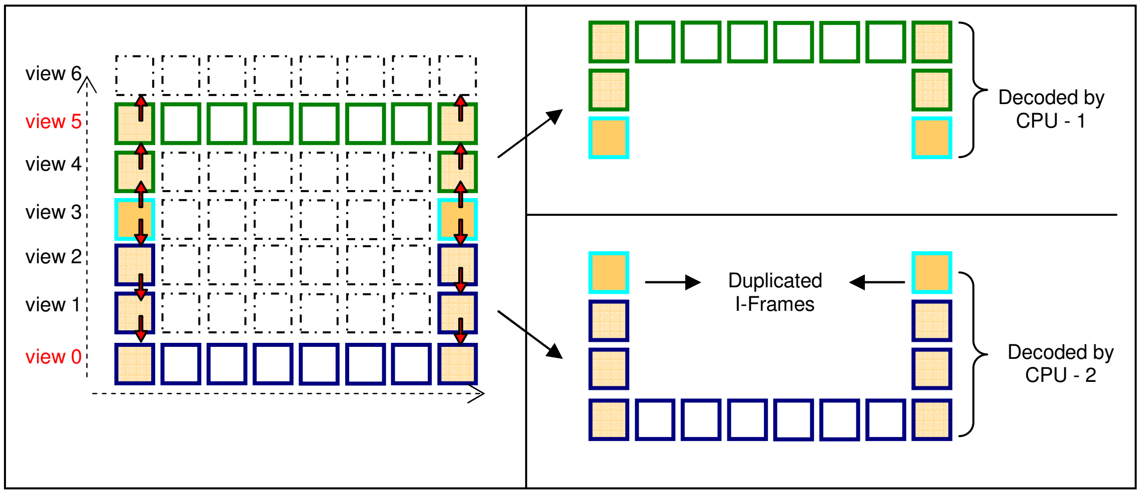


Figure 2.12: Generation of independently decodable streams for the stereo case

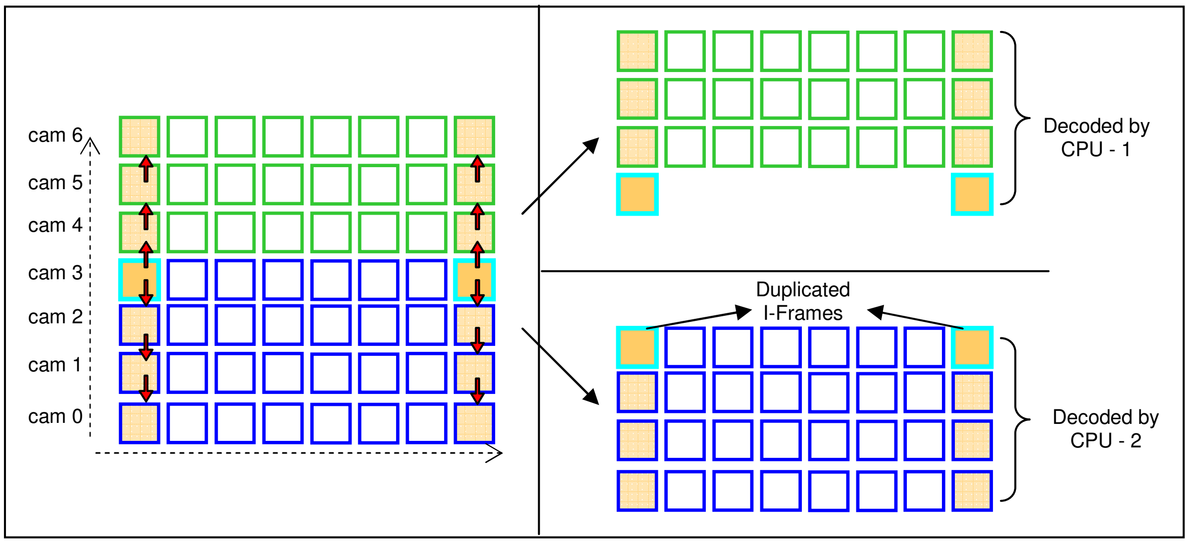


Figure 2.13: Generation of independently decodable streams for the multi-view case

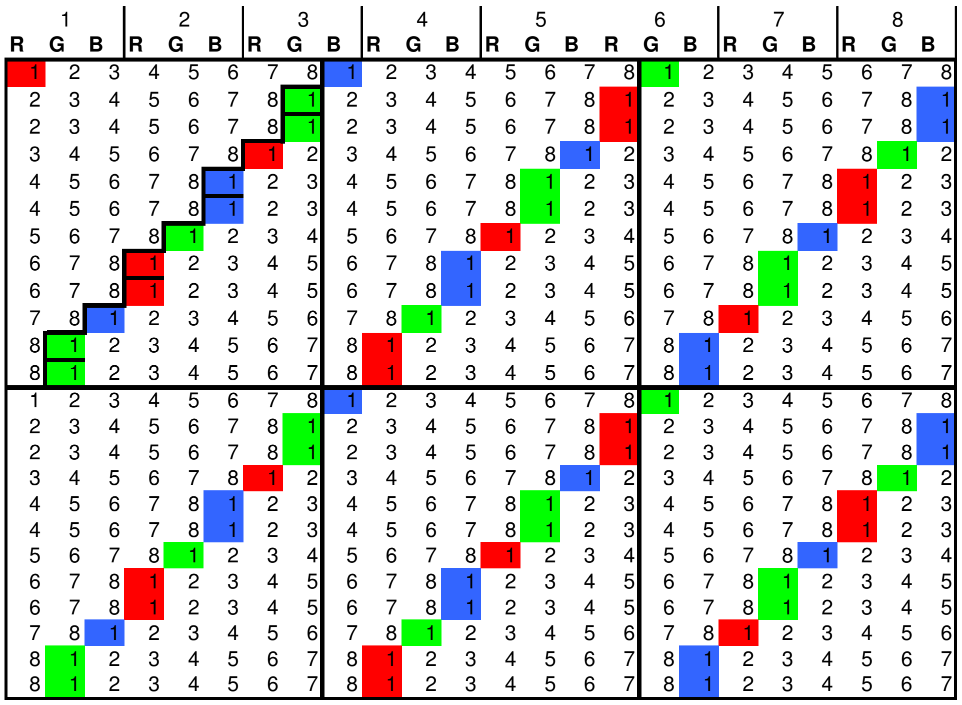
We start by defining the term of independently decodable byte-stream in a GOP. A stream is independently decodable if it requires no other frame to decode all the frames it includes. In Figure 2.10, the frames that are highlighted with dark red border forms an independently decodable stream. However, with its current order it cannot be divided into two separate streams. In order to achieve this we need to duplicate the I-Frames as shown in Figure 2.11. The same idea is valid for the multi-view displays as shown in Figure 2.12

2.6.3 Interdigitizing

Both the number of required views and the way these views are displayed on the screen depends on the type of the display. Views from the same time instant needs to be merged before presentation for lenticular-sheet or parallax-barrier displays. This process is called interdigitizing, where N-views from WxH size images are combined to generate a single image of same size. As a result, the actual spatial resolution of the video is decremented.

In RGB representation each pixel at x and y coordinate has three values that represents the strength of red, blue, and green color. Commonly each pixel is represented by 3 bytes for RGB values. In such configuration, the maximum value for a single color is 255.

If we numerate these RGB colors as 1, 2 and 3 respectively then we can identify each sub-pixel value in an image with three variables x, y and c. Let x and y be the coordinate of the interdigitized image and c as the sub-pixel number, then we can define F(x; y; c) which maps a sub-pixel at the interzigged image to a sub-pixel in original views. For instance, if F(50; 70; 1) = 5, then the red component of the pixel in 50th column and 70th row of the interdigitized image mapped to the 5th camera's red component at the corresponding pixel location. The Figure 2.13 is a visual representation of the function we define and name it as the Interzigging pattern of the display. Once the interdigitizing operation is over, the generated image is ready to display.



1			2			3			4			5			6			7			8		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1
2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1
3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2
4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3
4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3
5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4
6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5
6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5
7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6
8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1
2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1
3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2
4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3
4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3
5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4
6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5
6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5
7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6
8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7

Figure 2.14: Interzigging pattern for an 8 view display

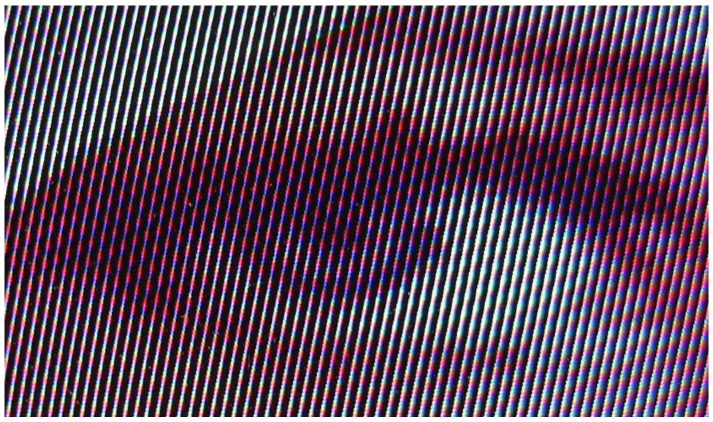


Figure 2.15: An interdigitized image

Chapter 3

EXPERIMENTAL RESULT

Efficient method for streaming video over the Internet depends on multiple parameters. Parameters such as bandwidth capacity of the link or rate of packet loss are not adjustable neither by the streaming system. However, the system can adapt to those input parameters by adjusting rate of channel coding. In this chapter, we will start with introducing the experimental results for estimating these parameters. Later, we will describe our settings for the actual video streaming over the Internet.

Video encoding performance, channel coding performance, and network analysis are three main subjects of our experiments. In the video encoding experiments, we investigate the compression efficiency of MVC and compare it with simulcast encoding. Another aim is to observe the effect of encoding schemes that allows multi-threaded decoding and calculate the cost of suboptimal encoding which is required for multi-threaded decoding. In the channel coding experiments, we tried to identify the performance of Raptor codes that is the FEC mechanism available in our implementation. Our performance criterion is the ability to recover the lost source packets that appear at varying rate. The aim is to determine the effective rate of channel coding for fighting against packet losses efficiently. Finally in the network analysis, we tried to determine the characteristics of the link between KU and METU as a case study. Network analysis and channel coding experiments provide valuable feedback to determine the rate of protection for real packet loss scenarios. The packetization strategy is also selected based on the results of network analysis.

3.1 Encoding Results

3.1.1 Comparison of MVC coding and Simulcast Coding

Implementing an MVC compatible streaming architecture requires modifications in protocols, mandates efficient buffering algorithms and introduces complexities in decoding process. Therefore, the performance gain of MVC encoding should be justified. Otherwise software developers can prefer simplicity to insignificant performance increase in encoding.

Naturally, the tests reveal that MVC has higher compression efficiency then simulcast coding for both stereo and multi-view cases. However, the gain highly depends on the features of the content. In Figure 3.1 and 3.2, you can see the PSNR/bitrate comparison of MVC and simulcast encoding for eight-view Adile and Iceberg sequences respectively. The difference in the encoding performance of MVC for a syntactic content and an actual recording is noticeable. The x-axis is the required bandwidth to achieve the PSNR value in the y-axis. Figures exhibits that simulcast encoding require higher bandwidth to achieve the same quality when compared with the MVC encoding.

The difference in the encoding performance between MVC encoding and simulcast becomes more obvious as the number of views increase. This fact can be explained as follows; Although MVC encoding cannot compress the independent view better than the simulcast encoder, for each additional view MVC introduce a stream at relatively lower rate due to interview dependencies. Therefore, as the number of views increase, the gap between the simulcast and MVC encoding increase as well.

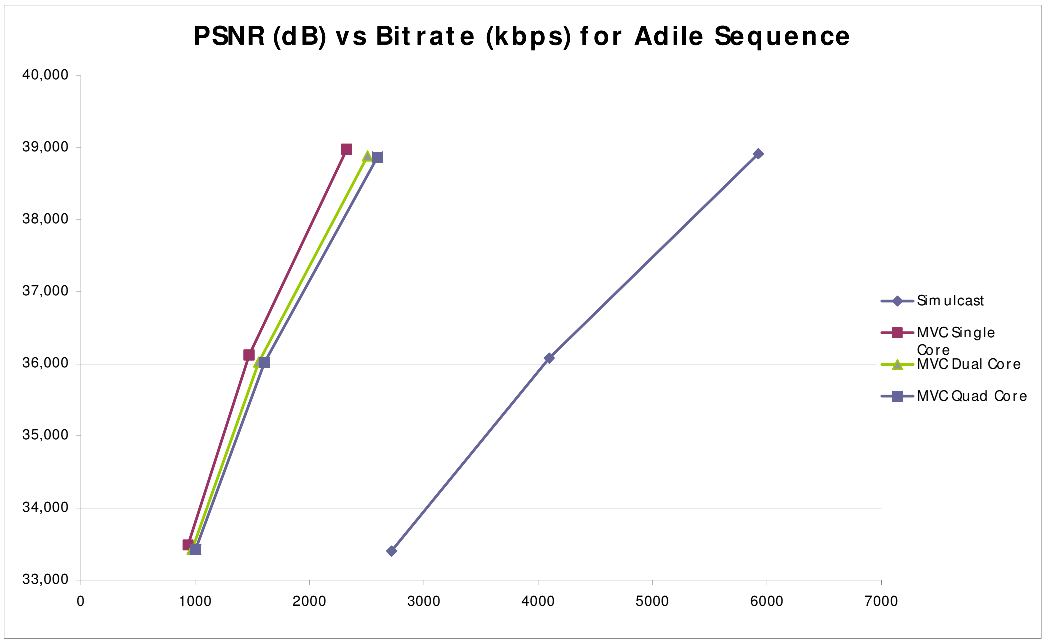


Figure 3.1 PSNR/Bitrate Comparison between MVC and Simulcast Coding

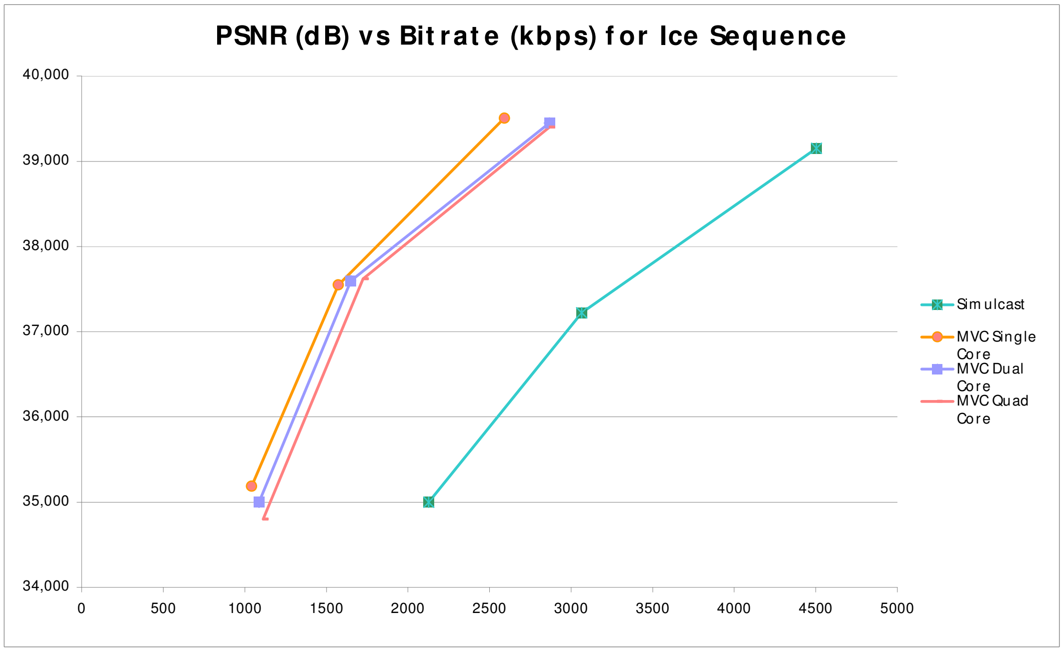


Figure 3.2: PSNR/Bit-Rate comparison between MVC and Simulcast Coding (8-View)

We aim to find a performance criterion that does not depend on the number of views. Such parameter should reflect the on the aim of MVC encoding scheme. If we consider the implemented hierarchical model in Figure 2.4, it is clear that MVC tries to decrease the size of I-frames at the dependent views. Therefore, we can calculate the efficiency of MVC by comparing the total size of I-Frames in independent stream vs. dependent streams. Such comparisons are presented in Table 3.1, 3.2 and 3.3.

Table 3.1 lists the size of streams generated by encoding Adile sequence with MVC. Stream 4 is the independent view so its total size overmatches the other streams. The rate allocation for B-Frames is roughly one fifth of the I-Frames for stream 4. MVC coding reduces the total size of I-Frames in the dependent views such that it becomes less than total size of B-Frames, indicating successful compression. The rate of reduction is given under the efficiency column and you can compare it with Table 3.2, which displays the results for the same content only for lower quality. As the quality decreases, compression becomes an easier task, allowing further reduction in the size of I-Frames.

The rate of compression decreases for an actual recording when compared with a computer-generated content. The difference in the settings of cameras such as white balance, calibration and noise decrease the efficiency of MVC encoding. Table 3.3 presents rate allocation between I-Frames and B-Frames for the Rena sequence that is captured by 1D camera array. Both the rate of I-Frames and also the efficiency of interview compression decreased drastically. Therefore MVC did not provide very significant results for Rena sequence when compared with the computer generated Adile sequence. Although the compression efficiency decreased only by half, the reduction in the size of I-Frames further reduced the effect of compression.

Adile 8 View MVC (PSNR 36,7)					
	Total	B-Frame	I-Frame	B/I Ratio	Compression
Stream 0	1710,70	1101,17	609,53	1,81	0,12
Stream 1	1679,47	1091,17	588,30	1,85	0,11
Stream 2	1648,85	1081,90	566,95	1,91	0,11
Stream 3	1762,32	1066,85	695,47	1,53	0,13
Stream 4	6218,14	1039,35	**5178,79**	0,20	-
Stream 5	1770,94	1067,26	703,68	1,52	0,14
Stream 6	1662,25	1079,23	583,02	1,85	0,11
Stream 7	1666,42	1083,74	582,68	1,86	0,11

Table 3.1: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Adile, PSNR 36.7)

Adile 8 View MVC (PSNR 28,25)					
	Total	B-Frame	I-Frame	B/I Ratio	Compression
Stream 0	390,66	291,97	98,69	2,96	0,07
Stream 1	388,20	288,59	99,60	2,90	0,07
Stream 2	375,60	280,85	94,75	2,96	0,07
Stream 3	406,20	275,84	130,36	2,12	0,10
Stream 4	1599,83	265,89	**1333,93**	0,20	-
Stream 5	409,88	278,84	131,05	2,13	0,10
Stream 6	378,86	283,39	95,48	2,97	0,07
Stream 7	382,45	284,01	98,44	2,89	0,07

Table 3.2: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Adile, PSNR 28.25)

Rena 8 View MVC (PSNR: 28,7)					
	Total	B-Frame	I-Frame	B/I Ratio	Compression
Stream 0	760,09	671,99	88,10	7,63	0,23
Stream 1	752,53	666,48	86,05	7,75	0,22
Stream 2	846,92	772,03	74,89	10,31	0,19
Stream 3	800,37	717,67	82,70	8,68	0,21
Stream 4	892,05	502,04	**390,01**	1,29	-
Stream 5	800,37	717,67	82,70	8,68	0,21
Stream 6	846,92	772,03	74,89	10,31	0,19
Stream 7	752,53	666,48	86,05	7,75	0,22

Table 3.3: Size of MVC Streams and byte allocation between B-Frames and I-Frames (Rena, PSNR 28.7)

3.1.2 Effect of Slice Mode

Slice mode is an encoding parameter that has an effect over the bitrate of the output stream. When slice mode is disabled, single NAL Unit encapsulates each frame. If the size of that NAL Unit is larger than path MTU, then the packet is fragmented at the network layer and payload is distributed to multiple fragments. Loss of any of those fragments makes the rest of them useless. This is a clear violation of application layer framing.

It is possible to perform fragmentation at the application layer by enabling slice mode. ALF is achieved by dividing the frame into slices and appending header information that is required for decoding the slice. This approah increases the chance of benefiting from each received fragment.

However, appending headers to slices increase the bitrate. In Table 3.4 presents the overhead of introducing splitting frames into slices. Notice that as the quality of the video decreases, size of each frame decreases, and the overhead of slice mode decreases as well.

Slice Overhead			
	PSNR	Total Size	Increase %
Adile QP28 w Slice	37,90	9186,43	
Adile QP28 wo Slice	37,98	8795,85	0,04
Adile QP30 w Slice	36,38	7463,81	
Adile QP30 wo Slice	36,45	7156,25	0,04
Adile QP34 w Slice	33,51	4618,25	
Adile QP34 wo Slice	33,59	4450,57	0,04
Adile QP38 w Slice	30,76	2779,18	
Adile QP38 wo Slice	30,80	2698,24	0,03
Adile QP42 w Slice	28,29	1723,99	
Adile QP42 woSlice	28,32	1673,78	0,03
Adile QP46 w Slice	26,07	1168,33	
Adile QP46 wo Slice	26,09	1145,61	0,02

Table 3.4: Overhead of Slice Mode

3.1.3 Effect of Encoding Scheme Set for Multi-Threaded Decoding

In order to perform simultaneous decoding, client needs to generate independently decodable code blocks based on the encoding scheme. The encoding schemes can deviate from each other. In one extreme each view is independently encoded. One the other extreme, complex dependency architectures are introduced to decrease the bitrate. Figure 3.4 presents the encoding schemes for single, dual and quad-core processors.

As an example, in Figure 3.4 (b) I-Frame at view 4 is referenced by four different paths. Following these paths, it is possible to see the required frames for decoding by quad-core processors. Figure 3.3 summarizes the required frames for each configuration and presents the workload of each core.

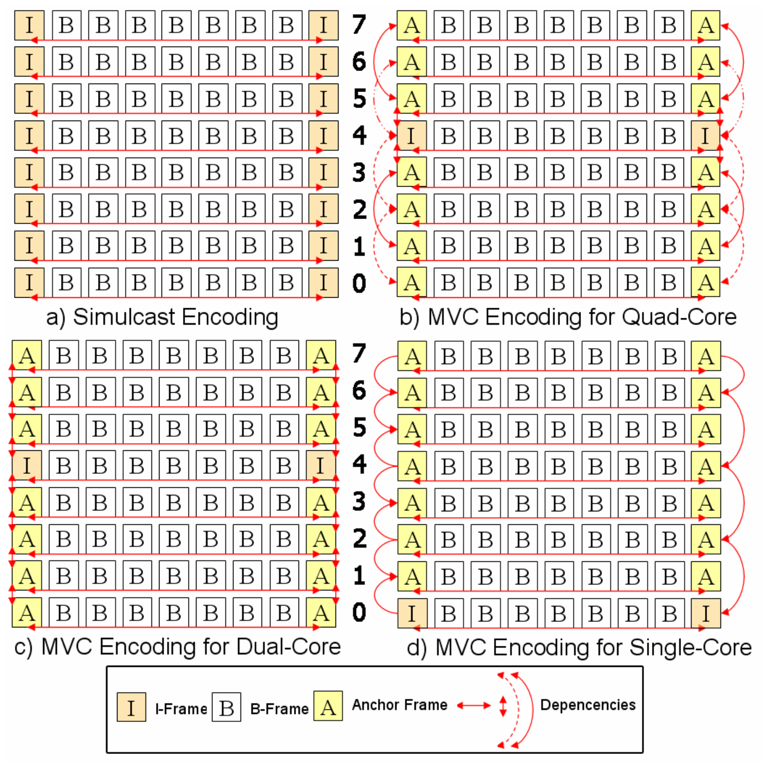

a) Simulcast Encoding
b) MVC Encoding for Quad-Core
c) MVC Encoding for Dual-Core
d) MVC Encoding for Single-Core
7
6
5
4
3
2
1
0
I
B
A
I-Frame
B-Frame
Anchor Frame
Depencencies

Figure 3.3: Multi-Threaded Decoding Schemes

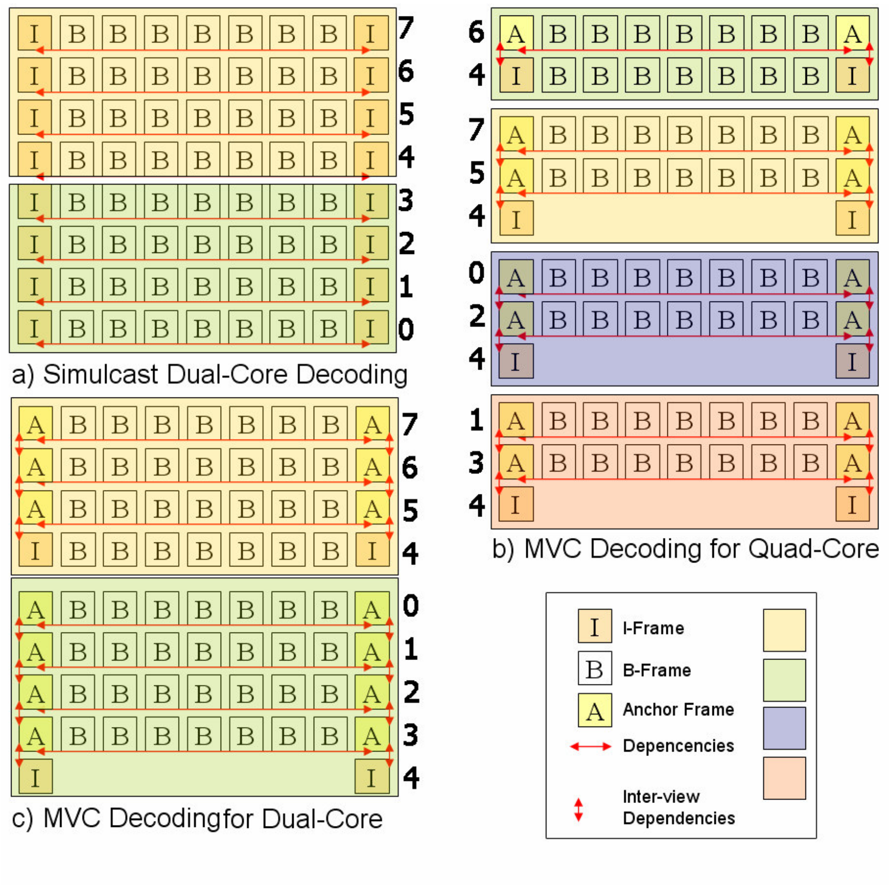

a) Simulcast Dual-Core Decoding
b) MVC Decoding for Quad-Core
c) MVC Decodingfor Dual-Core
I-Frame
B-Frame
Anchor Frame
Depencencies
Inter-view Dependencies

Figure 3.4: Multi-Threaded Encoding Schemes

The encoding efficiency of each scheme is not identical. In dual-core scheme, each view refers to an adjacent view. In the case of quad-core scheme, there are non-adjacent references. As the camera locations deviate from each other, the similarities within the captured frames decrease as well. Therefore, quad-core encoding scheme is not as efficient as the dual-core case. The encoding efficiencies of each configuration are presented in Table 3.5.

3.2 Forward Error Correction Results

3.2.1 Packet Recovery

Channel coding is the major tool for fighting packet losses in our streaming system. We performed series of tests to understand its strengths and weaknesses. The major goal of the tests was to identify the packet loss behavior in which FEC seems useful and identify the cases where FEC does not provide gain. Once we have that knowledge, we can choose the parameters that will give us the maximum throughput or decide not to use forward error correction at all. In the rest of this section, we will describe the test conditions and then present our results.

The test is initiated by setting the parameters such as the block size, channel coding rate and loss percentage. Then the data symbols are fed to FEC encoder to generate the protection symbols. The collection of data and protection symbols is named as a *symbol block*. The packet-loss simulator marks some of the symbols as lost based on the lost percentage that we set initially. Finally, the FEC decoder tries to reconstruct the data symbols without using the marked symbols.

In order to define our performance criteria for channel coding, let us assume that we have the parameters listed in Table 3.5. Once the decoding operation is performed there are three possible outcomes, namely successful decoding, partial decoding, or failed decoding. Successful decoding occurs when we recover all of the lost data symbols. In other words number of recovered symbols (r) is equal to number of lost symbols ($r = l$). If the decoder could recover some of the data symbols, then it is partial decoding ($0 < r < l$). Finally, if we cannot recover any of the symbols, it is failed-decoding ($r = 0$).

m	number of data symbols
n	number of protection symbols
k	block size ($m + n$)
m_R	number of received data symbols at client
n_R	number of received protection symbols at client
l	number of lost packets. ($k - (m_R + n_R)$)
r	number of recovered symbols after FEC decoding ($0 <= r <= l$)
m_D	number of decoded data symbols. ($m_D <= m$)
n_{Req}	number of protection symbols required for successful FEC decoding

Table 3.5 Parameters for Forward Error Correction

We want to achieve successful decoding using fewer protection symbols as possible. In the best case, we get successful decoding when protection symbols are equal to lost symbols, $n = l$. However, a single protection symbol cannot contain more information than data symbols. Therefore, recovering lost packets with the same number of protection packets can only be possible if we repeat those packets. For this reason, partial or even failed decoding can occur when the number of protection symbols is larger than the number of lost symbols.

Our performance criterion for the channel coding is the probability of recovering a lost packet through the FEC decoding operation (r/l). In Table 3.6, you can see the performance of Raptor Codes for varying inputs that are block size, protection rate and channel coding rate. We have two important observations out of this test. Firstly, the cells of same the color display the improvement in packet recovery rate as the block size increases. It is a well-known fact that as the block size increases the efficiency of Fountain Codes increases [45]. The second observation is the critical drop in performance, which is highlighted by the vivid and pale cell pairs. The vivid colored cells are the cases when the loss rate is equal to the protection rate. When compared with the pale colored cell, you can notice the drastic drop in the performance. This observation leads the following conclusion. The required protection rate is higher than the packet loss rate for effective loss recovery.

BLOCK SIZE	Protection Rate	Loss Percentage					
		4	5	9	10	14	15
100	5	0.62	0.37	0.00	0.00	0.00	0.00
	10	0.99	0.99	0.68	0.38	0.01	0.00
	15	1.00	1.00	0.99	0.98	0.64	0.37
200	5	0.79	0.39	0.00	0.00	0.00	0.00
	10	1.00	1.00	0.80	0.39	0.00	0.00
	15	1.00	1.00	1.00	1.00	0.80	0.39
400	5	0.95	0.40	0.00	0.00	0.00	0.00
	10	1.00	1.00	0.95	0.39	0.00	0.00
	15	1.00	1.00	1.00	1.00	0.95	0.40

Table 3.6: Possibility of Recovering a Lost Packet by FEC Decoding

3.2.2 Redundancy Comparison between FEC and MDC

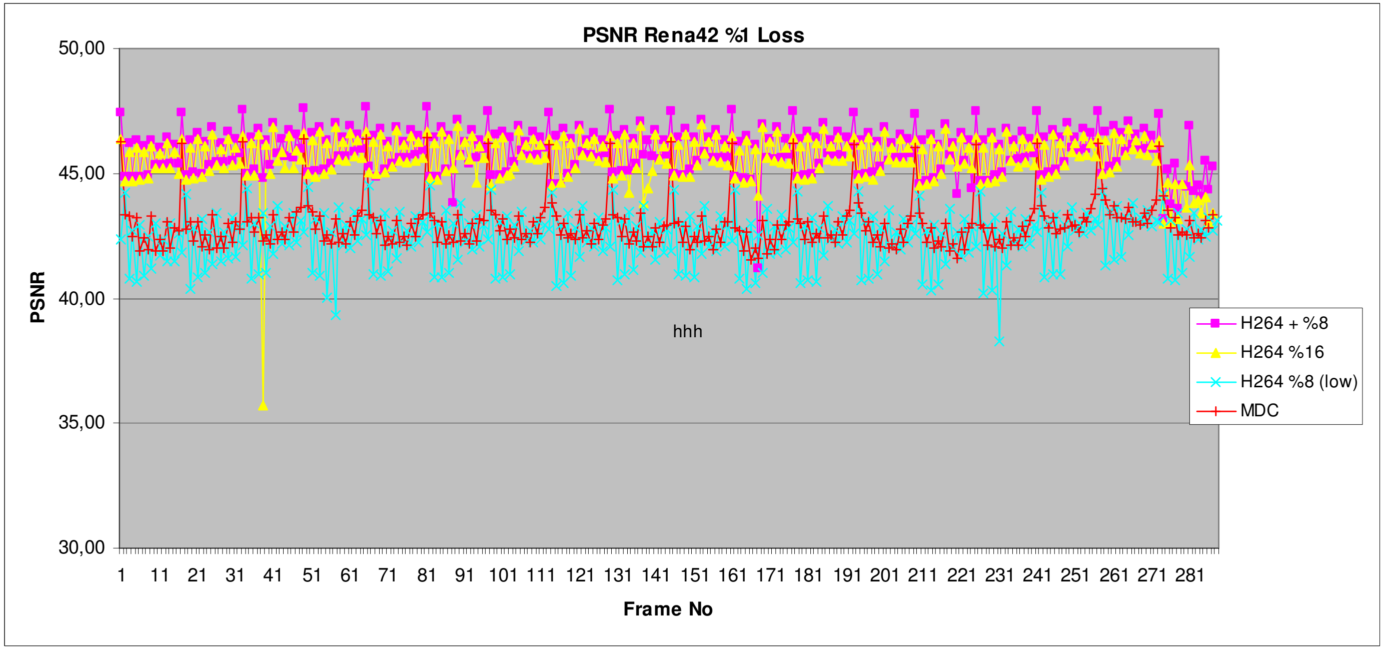
Forward Error Correction mechanism is not the only way of struggling against packet-losses. Automatic Repeat Query (ARQ) and Multiple Description Coding [47], [48] are other alternatives for the same purpose. MDC recently gain the attention in video transmission and presents promising results. All of these methods require extra bandwidth, which would not be needed lossless transmission. In this section, we compare the amount extra data we introduce with FEC and MDC.

In this test, we have three video streams, which are all used the same content as source. The first stream is encoded with H264/AVC and added %8 FEC protection symbols. The second content is again encoded by H264/AVC with %16 protection packets. In order to match their bandwidth, the second video has lower initial PSNR value. The third content is the sum of two MDC pairs. Finally, we introduced low bandwidth H264 video with again %8 coding. In the configuration of the first two content, we tried to match the video rate of the MDC stream. Trying to match a particular bit-rate decreases the efficiency of encoders. The last stream is to show the effect of inefficiency. In Table 3.7, you can see the bit-rate of each content and in Figure 3.3 and 3.4 you can see the simulation results for %1 and %4 packet loss respectively.

No	**Contents**	**Bitrate**	**PSNR**
1	H264 + %8 FEC	1,695	46,05
2	H264 + %16 FEC	1,728	45,74
3	MDC (total)	1,674	43
4	FEC 8 (low)	**0,634**	42,45

Table 3.7: Streams in the Redundancy Test

Results show that, Raptor Codes has higher PSNR/Bitrate performance when compared with the proposed MDC descriptions. The repetition of base layer in the MDC codec introduces the high redundancy. Therefore, for the purpose of loss recovery MDC is costly.



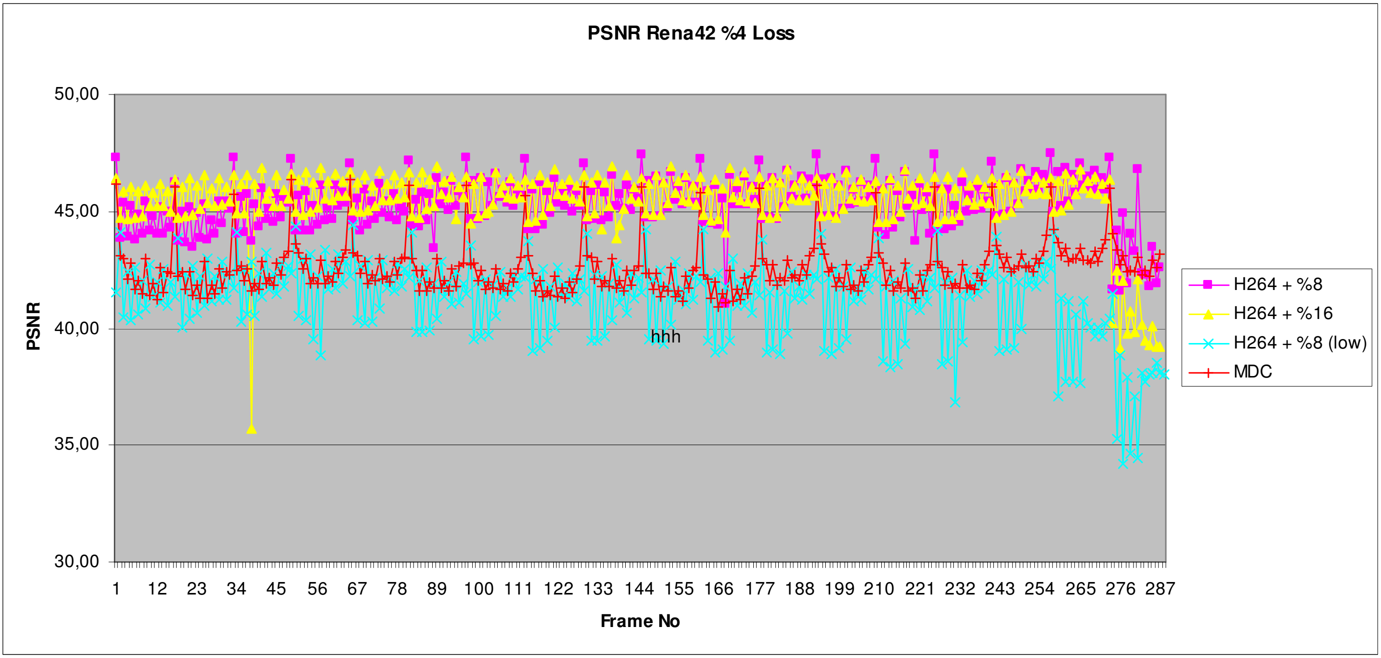


Figure 3.5: Redundancy Comparison between H264+FEC and SVC/MDC Network Analysis

Analyzing the behaviors of a link between two end-points without accessing the parts in the middle is a channeling task. Beside the high number of parameters that seriously affect the overall performance of the system, most of the time these parameters cannot be determined simply. In this section, we will first pinpoint the possible packet loss cases that these parameters have influence. Later we will briefly describe our test platform and present its results. Finally, we will combine the results and explain the conclusions we reach.

3.2.3 Packet Loss Behavior

There are many possible reasons for a packet loss such as unsuitable buffer size at the end nodes, the policies on the routers in between, simple bit-errors and many more. Among this variety of possibilities, the most common one is the failure of a router to forward a packet. Therefore, we will focus on routers and split them into two groups, the edge routers, and intermediate routers. The intermediate routers are in between the first and the last router on the link. They simply forward the packets to the next routers in the path as best as they can. On the other hand, unlike intermediate-routers, the edge routers are also responsible for fair distribution of available bandwidth in the local area network they reside. If we consider Figure 3.8, you can see two separate networks connected to the Internet through a router. The router separates two networks and isolates their internal traffic. Isolation of internal traffic increases the individual throughput of each network. Beside isolation of networks, the router keeps track of connections to the Internet and tries to distribute the available bandwidth as fairly as possible. If one of the connections tries to use more than its share, the router drops packets that belongs to that particular connection. In the case of an institution network, like Koc University, the router sides inside the institution and the network administrator had the privilege to set the bandwidth limit of each node. In the case of home users, the router resides in the ISP and the bandwidth limit is based on the fee paid.

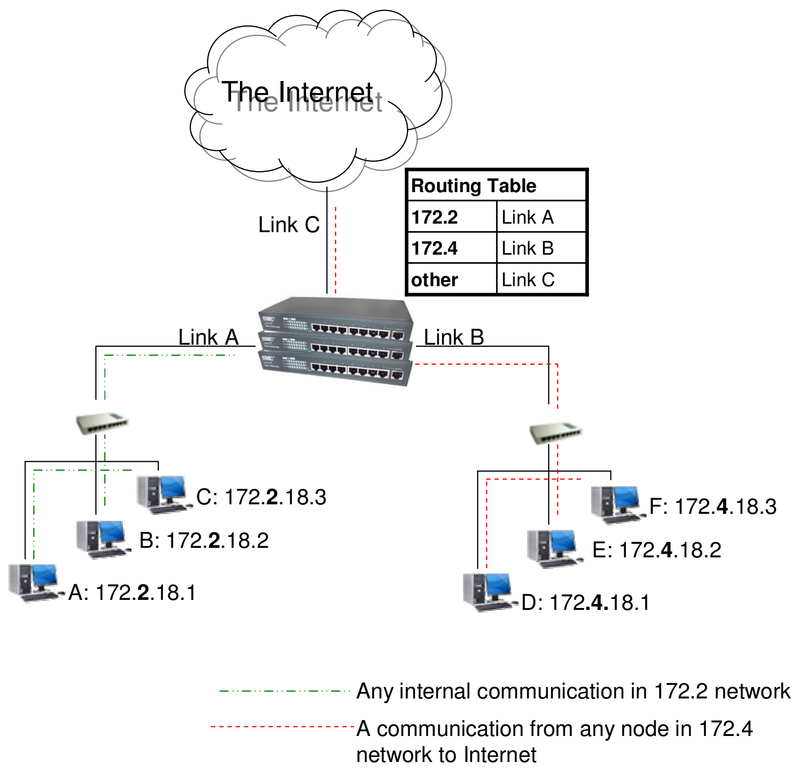


Figure 3.6: A Router in LAN Environment

Packet loss can occur even when you send at a rate below your available bandwidth. This is due to the best effort forwarding mechanism of the Internet. As shown in Figure 3.5 each router has a queue to store the packets, which the router cannot immediately upload to its outgoing link. When the rate of incoming packets is larger than the upload rate, the queue starts to grow. When the queue is full, the router discards any the incoming packets and they are lost. Considering chance of queue overflow in every router between the source and the destination, there is a fair amount of packet loss error when sending below the rate of available bandwidth.

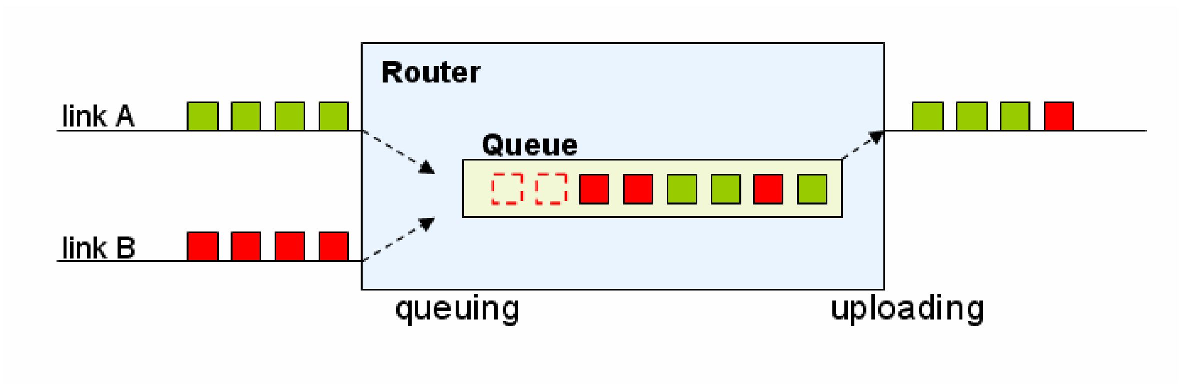


Figure 3.7: Internal structure of a Router

3.2.4 The Packet Loss Experiments

We perform the actual network tests over the link between Koc University and METU. The Figure 3.9 summarizes the underlying packet forwarding mechanism for our test case. In the initialization period, the sender and receiver agrees on test variables over a reliable TCP connection. During a test, sender appends incremental sequence numbers to the UDP packets and forwards them to the receiver, similar to the RTP transmission. The receiver can trace the packet losses and records both their frequency and duration. When the test is over the receiver composes a report, and forwards it to the sender.

The set of test variables consist of three elements, duration, transfer rate (bit-rate), and packet size. The packet size is the amount of data that a single packet carries. The adjustment of packet size allows us to set the packet-rate without modifying the bit-rate of the transfer. In our experiments, we fixed the duration to ten minutes and used packets of 700 or 1400 size in bytes. We had performed multiple tests over a wide range of bit-rate.

Based on our observations on loss reports, we noticed that modifying bit-rate and packet-rate has different consequences for different bit-rates. Therefore, we will present the results for two specific cases. In the first case the bit-rate will be below the available bandwidth, for the second case, they will be comparable to each other.

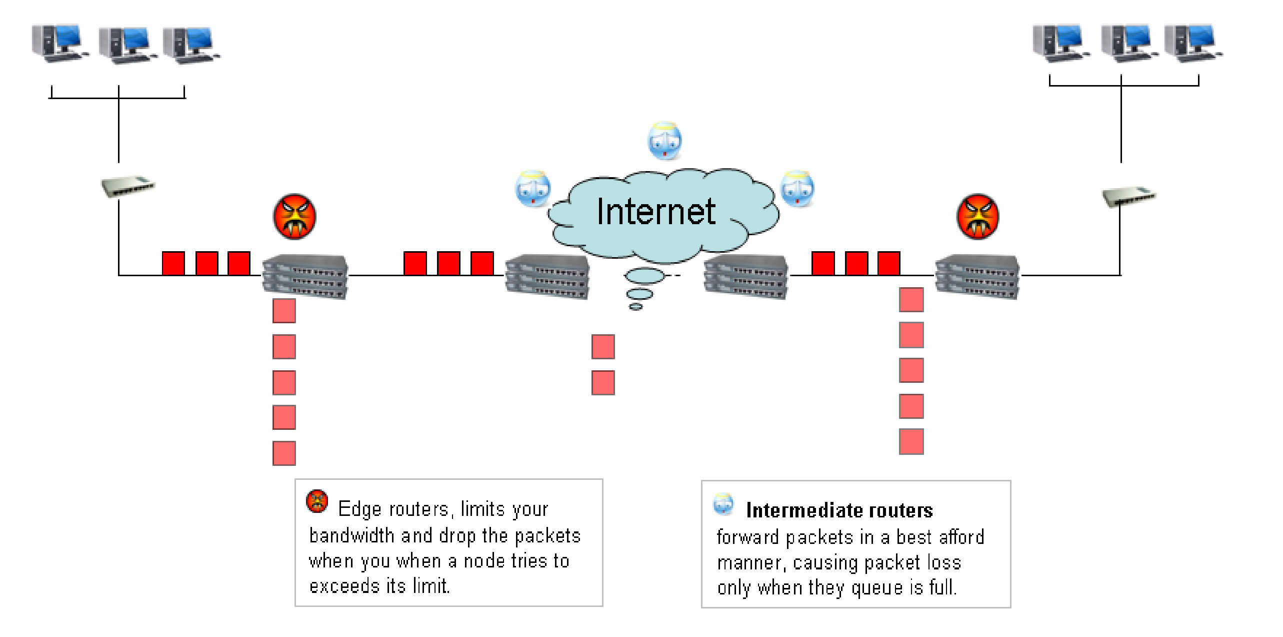

Internet
Edge routers, limits your bandwidth and drop the packets when you when a node tries to exceeds its limit.
Intermediate routers forward packets in a best afford manner, causing packet loss only when they queue is full.

Figure 3.8: Packet Forwarding Model for the Internet

If the transmission rate is below the available bandwidth, then the packet loss rate remains close to the vicinity of a relatively steady rate. Although the increase in packet rate has an incremental influence, packet loss rate does not reach to a disturbing level for the sake of streaming. This can be justified if we consider our model of the packet forwarding in the Internet. As you can see in Figure 3.6, if the bit-rate is below the available bandwidth, then packets are not filtered at the edge routers leaving overflowed queues as the prime reason for packet loss. In such a case, increase in the packet rate, increases the chance of queue overflows. As a result, the connection experiences packet losses with short duration. In table 3.8 you can see the results for low bit-rate transmission.

INPUTS													
Rate (bps)		100	100	150	150	200	200	250	250	300	300	350	350
Rate (pps)		9	18	13	27	18	36	22	45	26,785	54	31	63
Duration (min)		10	10	10	10	10	10	10	10	10	10	10	10
NALU Size		1400	700	1400	700	1400	700	1400	700	1400	700	1400	700
OUTPUTS													
Packets Send		5357	10714	8035	16071	10714	21428	13392	26785	16071	32142	18750	37500
Packets Received		5356	10704	8031	16052	10703	21418	13381	26748	14452	31998	18731	37478
Packets Lost		1	10	7	19	11	38	11	37	30	144	45	120
Loss Percent		0,02	0,09	0,09	0,12	0,10	0,18	0,08	0,14	0,19	0,45	0,24	0,32
Loss Length Distrubution	1-10	100	100	100	100	100	100	100	100	100	100	100	100
	11-20	0	0	0	0	0	0	0	0	0	0	0	0
	21-50	0	0	0	0	0	0	0	0	0	0	0	0
	50-100	0	0	0	0	0	0	0	0	0	0	0	0
	100+	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.8: Packet Loss Behavior During Low Bitrate Transmission

When the rate of the transmission gets closer to the available bandwidth, there is a drastic increase in the packet loss percent. In addition to the increase in packet loss frequency, the duration of the packet losses grows as well. This time the edge-routers are responsible for packet loss as they drop the packets for exceeding the bandwidth limitation. As a result, the ratio of packet loss becomes proportional to the transfer rate. In Table 3.8, we present the results for transmission over a channel, which has a capacity comparable to the desired rate of transmission. One important conclusion is reached if we combine this result with the recovery rate of FEC. We stated in section 3.2 that the ratio of protection symbols should be more than the ratio of packet losses. If we introduce more protection packets when the bandwidth is a bottleneck, we create proportional packet losses. In such a case, the ratio of protection packets cannot be more then the packet loss rate making the channel coding useless.

INPUTS													
Rate (kbps)		1000	1000	1200	1200	1400	1400	1500	1500	1600	1600	2000	2000
Rate (packet/s)		89	179	107	214	125	250	134	268	143	286	179	357
Duration (min)		10	10	10	10	10	10	10	10	10	10	10	10
Packet Size		1400	700	1400	700	1400	700	1400	700	1400	700	1400	700
OUTPUTS													
Packets Send		53571	107142	64285	128571	75000	150000	80357	160714	85714	171428	107142	214285
Packets Received		53569	107126	64173	128435	22754	54471	21826	29682	22248	44791	15668	66886
Packets Lost		49	18	144	294	51784	95696	58260	121353	62902	126631	89437	177545
Loss Percent		0,1	0,0	0,2	0,2	69,0	63,8	72,5	75,5	73,4	73,9	83,5	82,9
Loss Length Distrubution	**1-10**	100	100	100	100	100	2	99	1	100	1	1	3
	11-20	0	0	0	0	0	98	1	98	0	99	99	71
	21-50	0	0	0	0	0	0	1	1	0	0	0	26
	50-100	0	0	0	0	0	0	0	0	0	0	0	0
	100+	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.9: Packet Loss Behavior When Bitrate is Comparable with Available Bandwidth

Chapter 4

CONCLUSION

4.1 Summary

In this study, we have investigated the modules that compose a complete MVV streaming system. We have performed series of tests to understand the characteristics of each module and searched the opportunities for their joint optimization. The results from these tests provide us the answers for several fundamental questions about a MVV streaming system. In the rest of this section, we will present the questions and their answer as the summary of the work done.

The required bandwidth for transmitting multi-view content is a very critical is valuable information. For this purpose, we encoded both computer generated and actual recording sequences. We saw that the required bandwidth to transmit multi-view content is closely related to the number of views requested. In the case of stereo, MVC coding performed poorly as it requires nearly double the bitrate of what it had been for the monocular case. Therefore, one can choose to simulcast encoding for the transmission of stereo video. However, for the multi-view case (8-view), MVC proves to be useful requiring only 2.75 times of the monocular case for a computer generated sequence. It is nearly one third of what it would have been in case of simulcast coding. For an actual recording, it is close to 5 times the monocular case meaning that it is still beneficial to use. Although the encoding is very dependent on the features of the content, these numbers still serve as a guideline.

Secondly, we have investigated the performance of channel coding under varying network conditions. In addition to recoding the rate of channel coding needed for different packet loss, we had identified general characteristic of FEC. We saw that for achieving convenient packet recovery rate, the number of protection packets should be more than the number of lost packet. In order words, the rate of protection should be higher than rate of packet loss. Later we combined this fact, with the loss behavior of the edge-routers. We stated that increasing the channel-coding rate does not produce useful results if the transmission rate is already comparable to the available bandwidth. FEC is a good method for fighting against packet losses, which are due to queuing mechanisms of the intermediate routers. On the other hand, if the packet losses are occurring to the bandwidth limitations in the edge routers, then it is best not to use channel coding at all. We have also compared the redundancy of FEC with MDC. We found that FEC-H264 encoding achiever higher PSNR/bitrate performance when compared with MDC-SVC.

Another issue was the required modifications on the streaming protocols and their backward compatibility. We have proposed modifications to the streaming protocols to accommodate the extra data exchange required for a multi-view streaming session. In the Session Description Protocol, we defined a new parameter for describing the dependency hierarchy of the encoding schemes. We have also proposed a modification for RTSP protocol which allows the client to set the rate of channel-coding. In all the proposals, we have considered the backward compatibility issue.

The packetization strategy and understanding the network characteristics are important not only for multi-view streaming systems, but also for the monocular streamers. Therefore, we have constructed tests to understand the behavior of the channel over the link between Koc University and METU as a case study. We have identified that beside the transmission rate, the rate of packets is another important criteria. In our software, we used slice mode and NALU aggregation to reduce the number of packets introduced to the network.

4.2 Future Work

The methods in video streaming are continuously evolving rapidly, offering new opportunities for the transmission of video over the Internet. Unless extendibility and flexibility issues are considered at the very initial phase of the software development, it is could be very difficult to take advantage of these opportunities. In this section we will first introduce the flexibility of our software and then consider the possible future developments that could take place.

The core of the software is build over an open-source streaming library named as Live555 [49]. It has a very flexible architecture and used by many well-known streaming solutions such as VideoLAN player. Clear abstraction of tasks and well-designed interfaces makes the modification or total replacement of single module very straightforward. In Figure 4.1, you can see a simplified server model build for H263 video delivery. The source object is responsible for extracting a single frame out of the video file stream and forwards it to the RTP Framer. The RTP Framers encapsulates the data without knowing its content. After padding the RTP header, framer forwards to the encapsulated data to the UDP socket where the frame is introduced to network. The only modification for upgrading the system to H264 was the replacement of H263 FileSourceObject. The rest of the stream was unaffected by this modification.

We can introduce a scalable video codec pack to the system in a very similar way. A scalable codec allows the content to be displayed at varying bit-rates. It allows adaptation to the network conditions. SVC adjusts the visual quality of the content depending on the available bandwidth. The combination of scalable video and FEC can be a powerful couple.

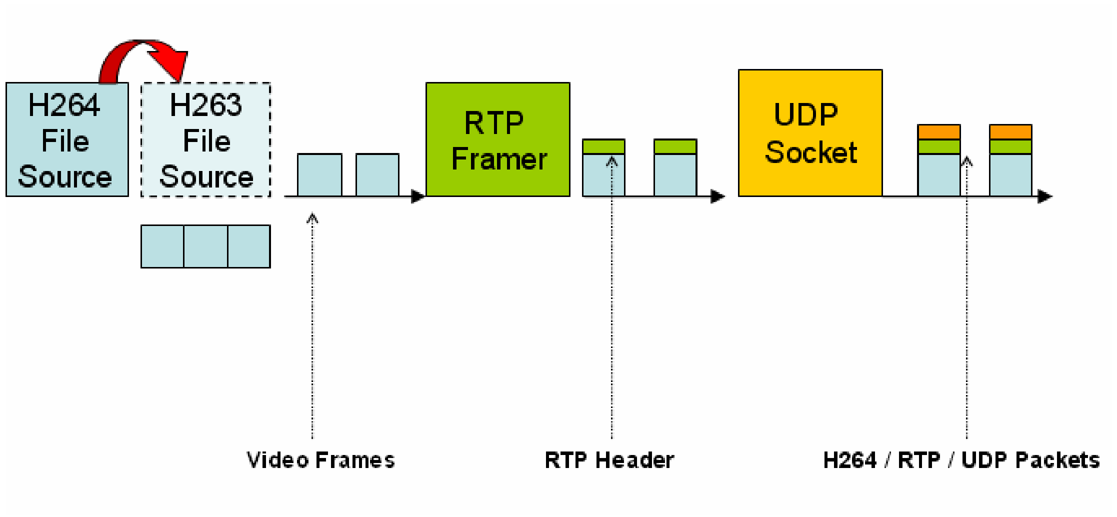


Figure4.1: An Example on the Modularity of Live555 Library

The next possible modification is the replacement of UDP with the newly developed DCCP protocol. DCCP is a transport protocol, which is allows application layer framing, similar to UDP. Additionally, DCCP has major advantages over UDP. DCCP is a connection-oriented protocol and that makes it possible to bypass firewalls if the network administrators approve. It is not possible to bypass firewall automatically when we use UDP as our transmission protocol. In TCP and DCCP there is a phase for establishing and destroying connection. Therefore, the NAT router can automatically trace those connections and perform appropriate forwarding. Since UDP is connectionless, it is not possible to understand when a connection is establish and/or finalized. Secondly, DCCP gives feedback about the state of the network, allowing the scalable video to adjust correspondingly. There is no built-in feedback mechanism in UDP. The combination of scalable codec with DCCP is the candidate for next generation video transmission.

BIBLIOGRAPHY

[1] Maly, W., "Cost of Silicon Viewed from VLSI Design Perspective," *Design Automation, 1994. 31st Conference on* , vol., no., pp. 135-142, 6-10 June 1994

[2] E.Grochowski, and R.D. Halem, "Technological impact of magnetic disk drives on the storage systems" IBM Systems Journal 2003, Vol.42, Num 2.

[3] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *SIGCOMM Symp. Communications Architectures and Protocols*, Philadelphia, PA, Sept. 1990, pp. 200–208.

[4] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[5] H. Schulzrinne, A. Rao and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, April 1998.

[6] M. Handley, V. Jacobson. SDP: Session Description Protocol. RFC 2327, April 1998.

[7] E. Kohler, M.Handley, S.Floyd. DCCP: Datagram Congestion Control Protocol. RFC 4340, March 2006.

[8] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050*, 2003.

[9] Tekalp, A.M.; Kurutepe, E.; Civanlar, M.R., "3DTV over IP," *Signal Processing Magazine, IEEE* , vol.24, no.6, pp.77-87, Nov. 2007

[10] Magnusson, M.; Lenz, R.; Danielsson, P.-E., "Evaluation of methods for shaded surface display of CT-volumes," *Pattern Recognition, 1988., 9th International Conference on* , vol., no., pp.1287-1294 vol.2, 14-17 Nov 1988

[11] Diamond, R., Wynn, A., Thomsen, K., and Turner, J. (1982). Three-dimensional perception for one-eyed guys, in Computational Crystallography, Oxford, Clarendon Press.

[12] Stelmach, L.B.; Tam, W.J.; Meegan, D.V.; Vincent, A.; Corriveau, P., "Human perception of mismatched stereoscopic 3D inputs," *Image Processing, 2000. Proceedings. 2000 International Conference on* , vol.1, no., pp.5-8 vol.1, 2000

[13] Wheatstone C. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision. Philosophical Transactions of the Royal Society of London, 1838.

[14] IJsselsteijn W.A., Seuntiëns P.J.H., and Meesters L.M.J. State-of-the-Art in Human Factors and Quality Issues of Stereoscopic Broadcast Television. Technical Report D1, IST-2001-34396 (ATTEST), 2002.

[15] Tiltman R.F. How "Stereoscopic" Television is Shown. Radio News, 1928.

[16] B. Javidi and F. Okano, eds., Three-dimensional television, video, and display technologies, (Springer, New York, 2002)

[17] YouTube, http://www.youtube.com

[18] GoogleVideo, http://video.google.com

[19] Video AOL, http://video.aol.com/

[20] Helix Projec, http://developer.apple.com/opensource/server/streaming/index.html

[21] VideoLAN Streaming Software, http://www.videolan.org/vlc/

[22] C. Fehn, P. Kauff, M. de Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton, "An Evolutionary and Optimised Approach on 3D-TV," Proc. of IBC, 2002.

[23] L. Onural, T. Sikora, and A. Smolic, "An overview of a new european consortium: Integrated three-dimensional television–capture, transmission and display (3DTV)," Proceedings of European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT), 2004.

[24] M. Tanimoto, “Free viewpoint television-ftv,” Picture Coding Symposium 2004, pp. 15–17.
[25] W. Matusik and H. Pfister, “3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes,” SIGGPRAPH 2004; ACM Trans. on Graphics (TOG) , vol. 23, no. 3, pp. 814-824, August 2004.
[26] A. Vetro, W. Matusik, H. Pfister, and J. Xin, “Coding approaches for end-to-end 3D TV systems,” in *Proc. Picture Coding Symp. (PCS)*, December 2004.
[27] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, Y.A. Reznik, R.N. Inc, and W.A. Seattle, “Video coding for streaming media delivery on the Internet,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 269–281, 2001.
[28] N. Hur, G. Lee, W. You, J. Lee, and C. Ahn,.”An HDTV-compatible 3DTV broadcasting system,” ETRI Journal, vol.. 26, no..2, pp..71-82, Apr. 2004.
[29] G. Petrovic and P. H. N. de With, “Near-future streaming framework for 3D TV applications,” Proc. IEEE Int. Conf. Multimedia and Expo (ICME), pp. 1881-1884, Toronto, Canada, July 2006.
[30] Mobile3DTV Project, http://sp.cs.tut.fi/mobile3dtv
[31] 3D4YOU Project, http://www.3d4you.eu/index.php
[32] Cha Zhang, Tsuhan Chen, "Multi-View Imaging: Capturing and Rendering Interactive Environments," cviie,pp.51-67, Computer Vision for Interactive and Intelligent Environment (CVIIE'05), 2005
[33] David E. DiFranco, Tat-Jen Cham, James M. Rehg, "Reconstruction of 3-D Figure Motion from 2-D Correspondences," *cvpr*, p. 307, 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01) - Volume 1, 2001
[34] P. Kauffa, N. Atzpadina, C. Fehna, M. Müllera, O. Schreer , A. Smolica and R. Tangera. " Depth map creation and image-based rendering for advanced 3DTV services

providing interoperability and scalability", Signal Processing: Image Communication, vol 22, no2, pp. 217–234 , February 2007.

[35] http://www.ptgrey.com/products/bumblebee2/

[36] Knorr, S.; Sikora, T., "An Image-Based Rendering (IBR) Approach for Realistic Stereo View Synthesis of TV Broadcast Based on Structure from Motion," *Image Processing, 2007. ICIP 2007. IEEE International Conference on* , vol.6, no., pp.VI -572-VI -575, Sept. 16 2007-Oct. 19 2007

[37] Chai, B.-B, "A depth map representation for realtime transmission and view-based rendering of a dynamic 3D-scene. " In: First International Symposium on 3D Data Processing Visualization and Transmission, Padova Italy, Sept. 2002.

[38] I. Rec, “H. 264 & ISO/IEC 14496-10 AVC, Advanced video coding for generic audiovisual services,” ITU-T, May, 2003.

[39] K. Mueller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic, T. Oelbaum, and T. Wiegand, “Multi-view video coding based on H.264/AVC using hierarchical B-frames,” in Picture Coding Symposium 2006. PCS, 2006.

[40] Ugur, K.; Hui Liu; Lainema, J.; Gabbouf, M.; Houqiang Li, "Parallel Encoding - Decoding Operation for Multiview Video Coding with High Coding Efficiency," *3DTV Conference, 2007* , vol., no., pp.1-4, 7-9 May 2007

[41] M. Handley, S. Floyd,J. Padhye,J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specificat", RFC 3448, January 2003

[42] J.-P. Wagner, J. Chakareski, and P. Frossard, “Streaming of scalable video from multiple servers using rateless codes,” in *Proceedings of the IEEE International Conference onMultimedia and Expo (ICME ’06)*, pp. 1501–1504, Toronto, Canada, July 2006.

[43] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, “Reliable multimedia download delivery in cellular broadcast networks,” *IEEE Transactions on Broadcasting*, vol. 53, no. 1, part 2, pp. 235–245, 2007.

[44] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W.Xu, “Raptor codes for reliable download delivery in wireless broadcast systems,” in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC ’06)*,

[45] P.Maymounkov, “Online codes,” Tech. Rep. TR2002-833, New York University, New York, NY, USA, November 2002.

[46] M. Luby, “LT codes,” in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS ’02)*, pp. 271–280, Vancouver, Canada, November 2002.

[47] Y. Wang, A. R. Reibman, and S. Lin, “Multiple description coding for video delivery,” *Proc. IEEE*, vol. 93, no. 1, pp. 57-70, Jan. 1995.

[48] Goyal, V.K., "Multiple description coding: compression meets the network," *Signal Processing Magazine, IEEE* , vol.18, no.5, pp.74-93, Sep 2001

[49] Live555 Streaming Protocol Library, http://www.live555.com/

VITA

Cihat Göktığ Gürler was born in İstanbul, Turkey on February 2, 1982. He received his BSC. Degree in Electric and Electronic Engineering from Koç University, Istanbul. He worked as a teaching and research asistanant from September 2006 and August 2008. During his study he worked for the 3DTV Project under the six framework of the European Union.