Adaptive P2P Streaming
with Application to Multi-view Video

by

Cihat Göktuğ Gürler

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

Doctor of Philosophy
in
Electrical Engineering

Koc University
March 2013

Koc University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral'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:

A.Murat Tekalp, Prof.

Reha Civanlar, Prof.

O.Barış Akan, Prof

Öznur Özkasap, Prof.

Sinem Ergen, Prof.

Date: ______________________________

to Sedef

ABSTRACT

Multi-view three-dimensional (3D) video is the next natural step in the evolution of digital media technologies. Recent 3D auto-stereoscopic displays can display multi-view video with up to 200 views. While it is possible to broadcast 3D stereo video (two-views) over digital TV platforms today, streaming over IP provides a more flexible approach for distribution of stereo and free-view 3D media to home and mobile with different connection bandwidth and different 3D displays. Here, flexible transport refers to quality-scalable and view-scalable transport over the Internet. These scalability options are the key to deal with the biggest challenge, which is the scarcity of bandwidth in IP networks, in the delivery of multi-view video. However, even with the scalability options at hand, it is very possible that the bandwidth requirement of the sender side can reach to critical levels and render such a service infeasible. P2P video streaming is a promising approach and has received significant attention recently and can be used to alleviate the problem of bandwidth scarcity in server-client based applications. Unfortunately, P2P also introduces new challenges such as handling unstable peer connections and peers' limited upload capacity. In this thesis, we provide an adaptive P2P video streaming solution that addresses the challenges of multi-view video streaming over P2P networks. We start with reviewing fundamental video transmission concepts and the state of the art P2P video streaming solutions. We then take a look at beyond the state of the art, and introduce the methods for enabling adaptive video streaming for P2P network to distribute legacy monoscopic video. Finally, we move to modifications that are needed to deliver multi-view video in an adaptive manner over the Internet. We provide benchmark test results against the state of the P2P video streaming solutions to prove the superiority of the proposed approach in adaptive video transmission.

ÖZET

3B video teknolojinin bir sonraki adımının çoklu-görüntülü video formatı olması beklenmektedir. Günümüzde stereo 3B yayını dijital platformlardan yapılabilse de, Internet üzerinden yapılacak bir yayın daha esnek bir yapı ile serbest görüş teknolojisinin evlerimize veya mobil platformlara gelmesi mümkündür. Burada esneklikten kastedilen, kalitenin ve/veya görüntü sayısının Internet üzerinden aktarım sırasında ölçeklenebilmesidir. IP ağları üzerinde çoklu görüntü iletilirken yaşanması muhtemel sorunların ölçeklendirme ile çözülebilmesi mümkündür. Dolayısı ile kaliteli bir servis ölçeklendirme için önem arz eder. Ancak söz konusu çok-görüntülü video olunca, ölçeklendirmeye rağmen sunucudaki kapasite gereksinimi kritik seviyelere ulaşabilir. Bu durum fizibilite açısından sorun teşkil eder. P2P ile video aktarımı, sunucu tarafındaki yüksek kapasite gereksinimi sorununu hafifletmek için kullanılabilir. Ancak öncelikle P2P teknolojisi ile birlikte gelen peerların çıkması gibi sorunların çözülmesi gerekmektedir. Bu çalışmaya temel video iletim kavramlarını ile en güncel P2P video iletim tekniklerini inceleyerek başlıyoruz. Sonrasında güncelin daha ötesine geçerek ölçeklendirilebilir P2P görüntü iletimi konusunu sunuyoruz. Son olarak da böylesi bir sistemin çoklu-görüntülü video desteğini nasıl elde edeceğini belirtiyoruz. Tüm bunlara ek olarak, elde edilen sistemin günümüz sistemleri ile karşılaştırmasına da bu çalışmada yer veriyoruz.

ACKNOWLEDGEMENTS

There are numerous people which I’m in depth for their assistance during my study. I would like to start with **Sedef Savaş** who has helped me to become a better man and gave me the strength and the reason to continue my research. There is no way for me to express my gratitude for things she has done for me. **Burak Görkemli** is dear friend and a miraculous person and who makes things better for everyone around himself. It has been my pleasure and rare opportunity for me to work with him, my brother. **Işıl Görkemli**, yet another member of the beautiful Görkemli family, is a perfect friend who I love to share good times and also know that she is by me in times of stress. **Tolga Bağcı** is probably one of the people who I have shared most of my time during my graduate study. It has always been a real pleasure to work with a man who is open-minded and meticulous.

Next, I would like to thank my advisor **Prof. Murat Tekalp**. He is not only a mentor who guides me in my study and warns about pitfalls, he is also a person who brings comfort in times of stress with his unparalleled intelligence and great personality. I would also like to thank **Prof. Reha Civanlar.** His knowledge and character have a huge impact in the way that I conduct my research.

Off course I should express my deepest thanks **to my family** for their never-ending support and love without which I would not have become the man I am now.

TABLE OF CONTENTS

Chapter 1: Introduction 1

1.1 Basic Video Streaming 4

1.1.1 Push vs. Pull Based Systems 4

1.1.2 Layered vs. Cross-Layer Design 7

1.1.3 Adaptation Methods 8

1.2 Basic P2P Architectures 10

1.2.1 Tree Based P2P Solutions 11

1.2.2 Mesh Based P2P Solutions 13

1.3 The Bittorrent: A File Sharing Protocol 13

1.4 Available P2P Video Streaming Solutions 15

1.4.1 PPLive 15

1.4.2 BitTorrent Assisted Video Streaming Service (BASS) 15

1.4.3 Enhancing BitTorrent for Supporting Streaming Applications (Bitos) 15

1.4.4 Tribler 16

1.4.5 P2PNext – The NextShare Platform 16

1.4.6 LayerP2P 17

1.5 3D Video Coding 18

1.5.1 Simulcast Video Coding 19

1.5.2 Dependent Video Coding 20

1.5.3 Multi-view-Plus-Depth Coding 20

1.6 Contributions of This Thesis 21

Chapter 2: Adaptive Video Streaming over P2P Networks 24

2.1 Transmission Unit: Variable Length Chunks with Prioritization 27

2.1.1 Proposed Variable-Length Chunk Format 28

2.1.2 Available Chunk Formats 29

2.1.3 Evaluation of Variable Length Chunks vs. Available Formats 31

2.2 Adaptive Chunk Scheduling 34

2.2.1 Proposed Chunk Scheduling Mechanism 34

2.2.2 Available Methods 40

2.2.3 Evaluation of Proposed Chunk Scheduling Mechanism 41

2.3 Incentive Mechanism: Peer Selection 47

2.3.1 Problem Definition 47

2.3.2 Proposed Unchoking Algorithm 47

2.3.3 Evaluation of the Proposed Incentive Method 49

2.4 Evaluation of Scalable Video Coding vs. Multiple Description Coding 51

2.4.1 Evaluated Video Codecs 51

2.4.2 Implementation of the Evaluated Approaches 52

2.4.3 Simulation Setup and Scenarios 53

2.4.4 Results and Comments 53

2.5 Summary 57

Chapter 3: QoE Aware 3D Video Streaming over P2P Network 58

3.1 State of the Art 3D Video Formats 60

3.2 Quality of Experience Aware Rate Adaptation for Stereoscopic 3D Video 63

3.2.1 Evaluation of Perceived Quality Using Asymmetry in Stereoscopic 3D 63

3.2.2 QoE Aware Scalable Video Coding and Performance Evaluations 68

3.2.3 Effect of Packet Losses on the Perceived Video Quality 69

3.3 Quality of Experience Aware Rate Adaptation for Multi-view Video 72
3.3.1 Scaling Methods for Multi-view Video 72
3.3.2 Subjective Tests 74
3.3.3 Subjective Test Results 76
3.4 Extension of P2P Architecture for QoE Aware 3D Video Streaming 79
3.4.1 Adaptation Decision Module 80
3.4.2 Chunk Scheduler using Multiple Windows 82
3.5 Adaptive Streaming Tests: QoS Perspective 83
3.5.1 Test Setup 83
3.5.2 Test Scenarios 84
3.5.3 Test Results and Interpretations 85
3.6 Adaptive Streaming Tests: QoE Perspective 89
3.6.1 Test Setup 89
3.6.2 Test Scenarios 90
3.6.3 Test Results and Interpretations 90
3.7 Summary 94

Chapter 4: Use-Case: Project DIOMEDES 96
4.1 Overview of the Project 97
4.1.1 Goals and Challenges 97
4.1.2 System Design 99
4.2 Main Seed Server 103
4.2.1 Metadata Creation 104
4.2.2 Chunk Distribution over HTTP 105
4.3 DIOMEDES P2P Engine 105
4.3.1 Adaptation Decision Engine 105

4.3.2 Chunk Validation 106
4.3.3 Selective MDC Approach 107
4.4 Modifications to OpenSVC Decoder 111
4.5 Tracker Protocol 114
4.6 Summary 115

Chapter 5: Conclusions 116
5.1 Summary 117
5.2 Key Findings 119
5.2.1 P2P Video Streaming 119
5.2.2 QoE Aware Rate Scaling for Stereoscopic 3D 120
5.2.3 QoE Aware Rate Scaling for Stereoscopic Multi-view Video 120
5.3 Future Directions 121
5.3.1 P2P Video Streaming 121
5.3.2 3D Video Streaming 122

LIST OF FIGURES

Figure 2.1: Rate distortion performance for Wildlife (1 GOP ~= 0.5 sec) 33

Figure 2.2: Adjusting the size of variable-length window and number of active streams .. 39

Figure 2.3: Average deceived layers 46

Figure 2.4: Number of chunks downloaded 46

Figure 2.5: Average received layers 46

Figure 2.6: Number of chunks downloaded 46

Figure 2.7: Buffering duration of proposed incentive method vs. Tit-For-Tat 50

Figure 2.8: Possible video coding choices for scalability and stability 51

Figure 3.1: 3D video formats and coding options 61

Figure 3.2: Scalable coding options for stereoscopic case. 69

Figure 3.3: Loss intervals in stereoscopic 3D video 70

Figure 3.4: RD performance of test sequences 75

Figure 3.5: Subjective Test Results 79

Figure 3.6: Design for QoE aware P2P video streaming 81

Figure 3.7:Downloading window with two layers and n views 82

Figure 3.8: Total bandwidth capacity of the nodes that provide chunks for tests 85

Figure 3.9: Results for Test 1 86

Figure 3.10: Results for Test 2 87

Figure 3.11: Results for Test 3 87

Figure 3.12: Results for Test 4 88

Figure 3.13: Results for the Test 5 (dynamic) 88

Figure 3.14: 3D video quality during playback of Music video (left view) 91

Figure 3.15: 3D video quality during playback of Music video (right view) 91

Figure 3.16: 3D video quality during playback of Music video 92

Figure 3.17: 3D video quality during playback of Street video 92

Figure 3.18: 3D video quality during playback of Street video.. 93

Figure 3.19: 3D video quality during playback of Street video.. 93

Figure 4.1: Design of the project DIOMEDES... 100

Figure 4.2: Multi-view video and spatial audio display in the University of Surrey 103

Figure 4.3: Push-based model (commonly adopted in tree-based topology) 108

Figure 4.4: Description Scheme in DIOMEDES for GOP size 8...................................... 110

Figure 4.5: Chunk scheduling in for MDC.. 110

Figure 4.6 Spatial dependencies between neighboring macroblocks 112

Figure 4.7: Decoding order of macroblock in parallel order 113

Figure 4.8: Level of speed up using macroblock level parallelism 114

LIST OF TABLES

Table 2.1: Transmission units in mesh-based P2P video streaming solutions 29

Table 2.2: Average results over 50 realizations.. 32

Table 2.3: Summary of the evaluations of the different transmission units 34

Table 2.4: Test content bitrate with respect to average DDR... 43

Table 2.5: Test conditions.. 53

Table 2.6: Test results for the evaluation of SVC, MDC and SMDC approaches 54

Table 3.1: PSNR thresholds for level of asymmetry in asymmetric coding........................ 65

Table 3.2: Test contents for the evaluation of asymmetric coding...................................... 66

Table 3.3: Test contents for the evaluation of asymmetric coding...................................... 66

Table 3.4: Normalized subjective test scores (Above Threshold) 67

Table 3.5: Normalized subjective test scores (Below Threshold) 67

Table 3.6: Test video sequences 71

Table 3.7: Bitrate limitations of tests scenarios for each sequence (kbps).......................... 74

Table 3.8: Adopted methods for each test conditions... 78

Table 9: Test conditions and best case results for adaptive streaming tests........................ 84

Table 4.1: Simple comparison of MDC, state of peers after 10 seconds........................... 107

NOMENCLATURE

ALF	Application Layer Framing
AVC	Advanced Video Coding
BASS	BitTorrent Assisted Video Streaming System
BITOS	Enhancing BitTorrent for supporting streaming applications
CBR	Constant Bitrate Coding
CDN	Content Distribution Network
CGS	Coarse-grained Scalability
DCCP	Datagram Congestion Control Protocol
DVB	Digital Video Broadcasting
GOP	Group of Pictures
HTTP	Hypertext Transfer Protocol
IPTV	IP Television
ITU	International Telecommunication Union
JVT	Joint Video Team
LTE	Long-term Evolution
MDC	Multiple Description Coding
MGS	Medium-grained Scalability
MPEG	Moving Picture Experts Group
MTU	Maximum Transmission Unit
MVV	Multi-view Video
NAL	Network Abstraction Layer
P2P	Peer-to-peer
RTP	Realtime Protocol
RTT	Round Trip Time
SNR	Signal to Noise Ratio
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBR	Variable Bitrate Coding
WebTV	Web Television

Chapter 1

INTRODUCTION

3D technology has already made an impact on the movie industry and drew significant attention from the audience. With wide availability of low cost stereo cameras, 3D displays, and broadband communication options, 3D media is now destined to move from the movie theater to home and mobile platforms. In the near term, popular 3D media will be available at our homes and most likely in the form of stereoscopic and multi-view video with associated spatial audio and users will be able to experience immersive 3D without wearing special glasses.

One of the biggest challenges in the transmission of MVV is dealing with varying bandwidth requirement which is due to varying number of views that should be transmitted according to each display setup of the users [1], [2]. The tradeoff between the quality and the price is the main factor that determines the number of views required. There are display systems that can support up to hundred views and provide holographic-like display of the scene but there are also displays that can support up to five views. The divergence mandate flexible transport mechanisms at a level that is never needed before, making almost all legacy transport solutions inappropriate for the delivery of multi-view video.

The IP platform provides is the one of the most flexible solutions to transmit MVV when compared to digital video broadcasting standards like DVB. With applications such as IPTV and WebTV, the IP network can serve as many views as required by the user display terminal and at a quality level allowed by the capacity of user’s connection. However, while it is theoretically possible to delivery 3D content at variable bit rates, the bandwidth requirement of at the content provider side can reach to critical values where it can be difficult to maintain scalable service against increasing number of recipients. Even if the infrastructure allows such has bandwidth connections, it may also be infeasible due to cost of such a high quality connection. Therefore, IP by itself may not adequately address all the problems present in 3D video delivery.

Peer-to-peer (P2P) solutions can distribute the task of forwarding data over the peers and alleviate the problem of high bandwidth requirement at the sender side, creating a more feasible solution 3D video delivery over IP. Unfortunately, the current infrastructure of the Internet is not in the optimum configuration for a P2P solution which favors symmetric (balanced) download and upload capacity of the peers. Actually, the connections are highly asymmetric, favoring download capacity in most cases and leaving fewer resources for uploading. Therefore, when compared to server-client technology, the P2P solutions are more prone to IP artifacts such as rate fluctuations and varying delays. However, when coupled with the adaptive streaming technology that matches source video rate to the capacity of channel between two end-nodes, P2P can also become a very effective tool for high-quality multimedia delivery.

Achieving scalable solutions is much more feasible with P2P when compared to centralized solutions (even distributed ones such as CDN). However, P2P also introduces new problems and challenges when compared against traditional server-client based systems. Some of those problems can be listed as: i) Bringing all peers that are interested in the same content together, ii) Formatting the content such that it is ready for P2P delivery iii) finding good neighbors among all candidate peers iv) creating robust overlay with neighbors v) handling peer exit events vi) creating incentive mechanism to motive peers to share larger portion of their upload capacity. Some of such issues that been targeted by BitTorrent protocol, which is a successful application for distributing large files over the Internet. However, the Bit-Torrent protocol and its policies almost make it impossible to use for video streaming directly. Therefore, we use the BitTorrent protocol as our starting point and then introduce critical modifications to support video streaming.

In this thesis, we propose a mesh-based P2P overlay that is similar to the BitTorrent approach but optimized for adaptive multi-view video delivery over IP networks. In this chapter, we start with reviewing key video streaming concepts such as cross-layer design, adaptation methods and then we provide a summary of basic P2P architectures and currently available P2P solutions. Then, we move to the 3D video coding and explain fundamental differences in its perception when compared against legacy monoscopic video. In Chapter 2, we explain the necessary modifications to enable monoscopic video streaming over a BitTorrent-like architecture. We start with the design principles of our proposed system and explain why we have chosen a mesh-based topology instead of other alternatives. Then, we start to introduce our system and explain topics such as formation of video chunks and policies regarding chunk exchange and other peer policies. We also provide a detailed analysis of the proposed architecture against well-known solutions such as BitTorrent Assisted Video Streaming System (BASS), Enhancing BitTorrent for supporting streaming applications (BITOS), Tribler, P2PNext, Deftpack and LayerP2P. Following that, we introduce steps to enable 3D support in Chapter 3. We mainly focus on the differences of rate adaptation between monoscopic video versus stereoscopic or multi-view 3D Video. This chapter especially focuses on asymmetric rate allocation in stereoscopic video and view scalability for the multi-view-video case. The evaluation of the adopted adaptation approaches are described as subjective tests. Chapter 4 provides a use case scenario that combines the proposed P2P architecture with the proposed scaling methods for 3D video. The chapter provides detailed description of a system that has hybrid architecture to provide multi-view-video content in a reliable manner. Finally, in Chapter 5, we provide our comments on P2P video streaming and 3D video services as conclusions to this study.

1.1 Basic Video Streaming

In this section, we present fundamental video basic terminologies. Besides the detailed information, we also try to provide well-known implementations if possible to make clear understanding in of the concept. We will begin with discussing two possible approaches for data transmission and explain the difference between a sender oriented (push-based) and receiver oriented (pull-based) streaming solution. And then we will review the possible adaptation methods over IP. Finally, we will introduce two fundamental design approaches which is layered and cross-layer approach.

1.1.1 Push vs. Pull Based Systems

The data transmission approaches can be classified as sender-driven and receiver-driven. In theory, both approaches may seem to be able to provide the same functionalities, but there are significant differences due to the level of information are present at different sites. These differences are especially important when performing adaptive streaming.

The information for performing adaptive streaming has three aspects. First is the structure of the encoded bit-stream. The sender side has full access to the encoded bit-stream. Therefore, sender-driven applications can perform finer rate adaptation when compared to its receiver-driven counterpart because the sender knows about the structure of the bit-stream; it has access to individual segments that are mandatory for decoding and the ones are *discardable* in case of bandwidth scarcity. For instance, when scalable video coding is adopted, the sender-side knows which data must be forwarded and which ones are optional at a slice level. Commonly, such information is not available to a receiver in a receiver-driven application, in which the receiver may have access to a course description of the content and may not choose to discard individual NAL units. The second aspect is the knowledge about the throughput of network. The client side always has full knowledge about the instantaneous throughput, whereas the server side either has no information (such as in the case of UDP with no feedback from the receiver side), or can have an estimate value (in the case of DCCP or TCP) [3]. Therefore, the sender-side commonly needs feedback from the receiver side. In the case of UDP, the feedback must be provided explicitly, whereas in the case of TCP, the average data transmission rate can be considered as an implicit feedback since this value is calculated according to rate of successful transmission of TCP segments. Finally, the portion of the data that is available at the receiver side is (naturally) available to receiver, whereas this information may not be available to the sender unless notified (e.g., explicit acknowledgements in the case of UDP and implicit acknowledgements in the case of TCP). Moreover, it may become even more complicated, if the receiver side has multiple sources as the case of mesh-based P2P networks.

1.1.1.1 Sender Driven, Push Based Video Streaming

In this approach, the server side is responsible for monitoring the quality of data transmission and performing adaptation if there is fluctuation in the available channel capacity. The server side has full access to encoded data stream therefore it can perform fine grain adaptation capabilities by discarding even single NAL Unit if needed [3].

In order to fully utilize the concept of application layer framing and create packets that regard NAL Unit boundaries (which are self-decodable data segments), sender-driven applications commonly prefer packet based communications (e.g., UDP and DCCP). In such a case, it is possible to minimize the effect of packet losses. Commonly, sender-driven solutions adopt RTP protocol that provides information needed to process the payload, such as codec type and timestamp [4], [5].

On the other hand, it is difficult to estimate the bandwidth availability in UDP because the network layer provides no feedback about whether the packet has been successfully transmitted or has been dropped. To overcome this problem, the RTP protocol also provides reports that are generated by the receivers which provide rough-feedback about the state of the network. Using reports, sender-side can estimate the channel capacity and perform rate adaptation to increase the utilization of the channel. If only a subset of video packets is to be delivered due to lack of adequate channel capacity then sender-side can discard NAL units to match the target bitrate. This decision can be based on the priority of NAL units in temporal prediction, favoring I-frames and reference B-frames against non-reference B frames. Moreover, if scalable video coding is adopted, base layer NAL units are prioritized versus enhancement layer NAL units in order to increase the received video quality. In short, the sender-driven applications may perform fine adaptation if they are notified about the rate of available link capacity and the missing data at the receiver site. Two well-known implementations of sender-driven video streaming applications are VideoLAN Client and Darwin Streaming server [6], [7].

1.1.1.2 Receiver Driven, Pull Based Video Streaming

Yet another approach is to allow the receiver to control the streaming process. Since the receiver has the statistical information about the reception quality no reports are needed from the sender side. Moreover, the receiver side has the knowledge about the present data and would request data accordingly. On the other hand, the receiver may not have access to the encoded bit stream and may not address each NAL Unit individually. Instead the receiver may use a metadata file that has rough information about the structure of the multimedia. Commonly, the granularity of the information is at the level of multiple groups of pictures (GOP). Using metadata file, the receiver side change decide on which data segments to request and which ones to discard. Therefore, in case of bandwidth scarcity, the receiver side has to discard a larger segment when compared against sender-driven solutions and may underutilize the link capacity [8].

1.1.2 Layered vs. Cross-Layer Design

It is possible to classify the design approaches for multimedia delivery into two classes based on the utilization of the OSI layers. If the application layer and the transport layer are operating independently then we call it a layered solution, in which the application does not consider the underlying network properties. In the other case, the application layer makes adjustments based on the feedback from the network layer and therefore the second approach is called a cross layer solutions, in which two layers work in harmony to obtain better performance, especially if the available network resources are limited.

In **layered approach**, the application layer does not use the feedback from the network layer and relies on the assumption that the underlying network is good enough to handle any payload with high efficiency. Although, it has inferior performance when compared against cross layer architectures, in has been widely adopted by the industry due to its easy to use implementation. YouTube for instance, does not use the data flow information of the channel. Naturally, in the case of bandwidth scarcity, the viewer experience more interruption in playback.

In **cross-layer approach**, the application layer tries to adapt itself according to the feedback from the transport layer. For instance, if the bandwidth drops below the current rate of video then the application layer should try to decrease the rate of video by performing rate adaptation [9]. Yet in another case for cross layer design is about the size of the NAL units. In RTP/UDP/IP protocol stack, it is best to form RTP packets smaller than path maximum transmission unit (MTU) in order to minimize the effect of packet losses. Otherwise (when the RTP packet is larger than MTU), the payload is split into multiple IP fragments and loss of one fragment nullifies the reception of the other, increasing the impact of individual packet loss over the received bit stream.

1.1.3 Adaptation Methods

Especially with the introduction of technologies such as 3G and LTE, we have a broad spectrum of link capacities and processing power of the nodes in the Internet. Therefore, it is no longer a viable option to use a single version of a content to serve all different type of nodes. Moreover, it may be the case that the link capacity of the nodes changes over time, forcing rate adaptation on the fly. Due to these conditions, the modern streaming solutions has to support two types of adaptation: i) terminal adaptation which corresponds to adjusting the content to the processing/display capacity of the node ii) rate adaptation, which corresponds to adjusting the content to the capacity of the connection with the node in a dynamic manner. In the following, we describe the two different adaptation options.

1.1.3.1 Stream Switching using Non-Scalable Video Coding

Stream switching is an old method that is used mostly in server-client oriented video streaming solutions. In this approach, the video is encoded multiple times at different qualities, (e.g., using different quantization parameters, using different content resolution). Terminal adaptation is inherently enabled, since a client can select a stream at desired resolution. For rate adaptation purposes, a special encoding approach is needed. Switching frames have been inserted periodically. In the case of change in the available link capacity, the node that is in control of the transmission can switching to a stream that is more appropriate for the given network condition. One well-known implementation of stream switching is the Smooth technology, which is developed by Microsoft. Silverlight performs real-time video coding in multiple layers using H.264/AVC standard [8].

1.1.3.2 Layer Switching using Scalable Video Coding

In layered coding, multiple encoding operations are performed at different qualities and the encoded streams exploit and the each quality level depends on the lower ones except the base layer stream, which generates the lowest quality when decoded. The remaining layer(s) can enhance the quality of the video stream if the previous layer has been successfully decoded [10].

Scalable extension of H.264 is a recent implementation and the state of the art in layered video coding. When compared against the H.264/AVC standard, the Scalable video coding (SVC) extension provides rate scalability at the cost of lower encoding efficiency for the enhancement layers. Nevertheless, if all quality layers of SVC are to be obtained by individual encodings using H.264/AVC, the total size of the bit stream is still lower in the SVC case. SVC has a backward compatible syntax that has been standardized by the Joint Video Team (JVT) of the ITU-T VCEG and the ISO/IEC MPEG. For backward compatibility, the base layer is complaint with the H.264/AVC syntax. Only the *discardable* enhancement layers introduce new types of NAL unit, which are to be discarded by decoders that do not support SVC. So, a receiver with H.264/AVC decoder can decode the content but at the lowest (base layer) quality [11].

The SVC provides spatial, temporal and quality scalability. SVC provides temporal scalability through the usage of hierarchical prediction structures, which was also possible in H.264/AVC. The latter two options, spatial and quality scalability, are supported by multilayer coding. Quality scalability is supported in three modes: Coarse-grained scalability (CGS) and medium-grained scalability (MGS) and fine-grained scalability.

However, due to high overhead, the last option has been removed from the standard. In spatial scalability and CGS mode of SNR scalability, if one enhancement NAL unit is discarded than the resultant bit stream is not compliant with the SVC standard because layer ID must be same throughout the bit stream. Therefore, these two options are used for terminal adaptation purposes (e.g., lossless data transmission over TCP or stored medium such as DVD). MGS, on the other hand, does not change the layer ID and therefore any MSG enhancement layer NAL units can be discarded in decreasing “quality_id” order, without violating the SVC syntax. Also, it is possible to fragment an MGS layer into multiple sub-layers by grouping zig-zag scanned transform coefficients and in this way increase the number of rate adaptation points [12]. In other words, MGS scalability is more suitable for rate adaptation purposes, in which some of the enhancement packet can be lost due to lack of available link capacity.

1.2 Basic P2P Architectures

The server-client unicast streaming model is not scalable bandwidth scalable meaning that it is not possible to serve an increasing number of clients without expanding the bandwidth capacity of the server side or creating a larger content distribution networks (CDN). The most important advantage of P2P solutions over traditional server-client architecture is the ability to achieve scalable media distribution solutions at the much feasible costs. The approach is based on reducing the bandwidth requirement of the server-side by utilizing the network capacity of the clients, which are now called peers.

In theory, it is possible to originate only a single copy from a source and duplicate the packets along the path to different peers at the network layer. This could have been the best solution for the scalability problem but unfortunately multicasting at the network layer has not been supported by most of the network routers. Instead, current P2P solutions use overlay networks in which the data is first received by the application at the end node and the re-injected into IP network in order to forward the same data to another peer. Consequently, multiple copies of the data traverse the IP network.

It is evident that relying on peers that may leave the network or stop data transmission anytime has its drawbacks but there are already successful P2P video applications that have managed to alleviate the impact of ungraceful peer exit events over the reliable data transmission on the P2P overlay. It is possible to examine these solutions under two extremes: Tree based (structured) and mesh based (unstructured) solutions.

1.2.1 Tree Based P2P Solutions

Tree-based solutions provide an efficient transport mechanism to deliver content from the server (root) that resides at the top of the tree to peers that are connected to each other in parent-child fashion. Data flow starts immediately once a peer joins a slot in the tree since there is no need for peer search phase. Content is pushed from the root to peers, allowing significantly lower latency in data dissemination when compared to mesh-based approaches [13]-[18]. These features make tree-based solutions suitable for time critical applications such as video broadcasting.

The major problem with tree based solutions is ungraceful peer exit which leads its descendants to starvation. Besides on-the-fly tree reconstruction, there are two possible solutions to this problem in the literature: using multiple parents [13], [14] or using multiple trees [15]-[18]. In the first solution, each peer has backup parent(s) to request content if the current parent leaves the network. The second solution is based on building multiple trees such that whenever a peer leaves the tree, its descendants may continue to receive content from an alternative path. In Split-Stream, trees are formed such that a peer that is an interior node in one of the trees becomes a leaf node for the rest of them [15].

Similarly in Stanford P2P Multicast (SPPM), complementary trees are formed such that path diversity is guaranteed [16]-[18].

Replicating the content for feeding multiple trees leads to redundancy within the network and decreases the overall efficiency of the solution. With multiple description coding (MDC) [19], it is possible to generate self-decodable bit streams (descriptions) with less redundancy. Using multiple descriptions increase the efficiency of video transmission [15], [20]-[22] and provide resilience against packet losses because as long as peers receive one of the descriptions, they can seamlessly decode the content. Moreover, each additional description enhances the video quality.

We note that widely used open or commercial P2P applications/services that rely on purely tree based solutions do not exist. The major reason for this is the lack of sufficient upload capacity of peers due to asymmetric Internet connections. Each peer is expected to feed multiple peers in a tree based solution, which is difficult to realize when peer upload capacity does not match the video rate. Therefore, peers cannot branch into multiple peers; on the contrary multiple peers need jointly serve another peer as depicted in Figure 1.1.

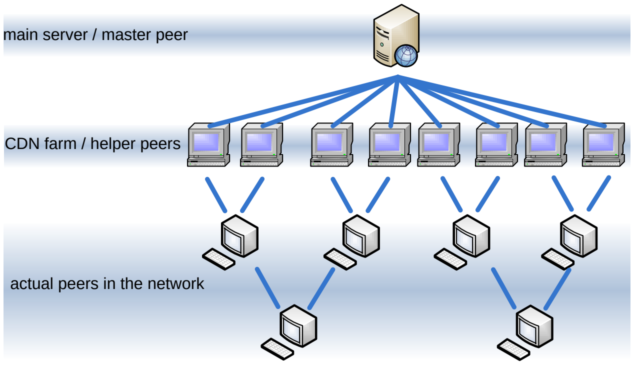


Figure 1.1: Tree formation when upload capacity of peers is half of the media bit rate.

1.2.2 Mesh Based P2P Solutions

In mesh-based solutions, data is distributed over an unstructured network in which each peer can connect to multiple neighbors with no parent-child relation. However, building multiple connections dynamically requires a certain amount of time, which we call the initiation interval. A peer may not be able to fully utilize its resources until the neighbor search and the initiation period are over. Therefore, mesh-based solutions are more suitable for applications that may tolerate some initiation delay.

The good side of the mesh-based solutions is the robustness against peer exit events. When data reception is interrupted for any reason, the receiver may simply switch to another available neighbor and only effected by a limited amount of time. The duration of the wasted time is commonly related to the timeout of the connection between peers, which is mostly governed by the round-trip-time among peers. Therefore, RTT has significant impact over the performance of mesh-based solutions.

1.3 The Bittorrent: A File Sharing Protocol

BitTorrent is a popular protocol for P2P distribution of large files over the Internet. The main idea is to divide content into **equally sized** pieces called chunks [23], [24]. A metadata file (.torrent) is generated that contains an IP address for a tracker server along with the SHA1 hash values for each chunk. The protocol is initiated once a peer (newcomer) connects to the tracker server and receives a sub-list of all peers (swarm) in the same session. There are two types of peers: A seeder has all the chunks and only contributes chunk, a leecher has some chunks missing that it tries to receive. Upon connecting to a tracker server and receiving a sub-list of peers, the newcomer starts to exchange chunks with its neighbors. Once a chunk is received, its hash value is compared to the one in the metadata file. Only then, the neighboring peers are notified about the availability of the chunk.

In BitTorrent, chunk exchange is managed with three fundamental policies. Using the **rarest first policy**, a peer aims to download a chunk that is least distributed in the swarm. Doing so, a peer increases the likelihood of obtaining unique data to exchange with its neighbors and also increases robustness of data delivery against peer churns by increasing the availability of rare chunks. Using the **tit-for-tat policy**, a peer ranks chunk upload requests and favors requests from a neighbor that has provided more data. A peer may choose to reject requests from other neighbors if the contribution from that neighbor is low to punish free-riders. The only exception is the **optimistic un-choking policy**, which forces a peer to upload a chunk to a *lucky* neighbor even if the neighbor has not provided any data yet to the peer in order to allow a new peer to start the session. Moreover, BitTorrent adopts **pipelining** and **end-game-mode** methods to increase the efficiency of data transmission. In pipelining, fixed sized chunks are considered to be further divided to sub-pieces that serve as the data transmission unit. The pipelining refers to requesting multiple sub-pieces at the same time to saturate the downloading capacity, and increase link utilization and throughput but sub-pieces of only one particular chunk can be requested in order to receive a whole chunk as soon as possible The end-game policy is to eliminate the prolonging delays in finishing the last download event due to a peer with very low upload capacity. Normally, a sub-piece is requested only once from a single peer. If a sub-piece is requested from a weak peer (that has limited upload capacity) it does not deteriorate the downloading process since the peer can always download other sub-pieces from other peers and saturate its link capacity. However, when the session is about to end, a weak peer may avoid finalizing the session. When end-game mode is enabled, the remaining sub-pieces are requested from multiple peers to avoid delays due to a weak peer.

1.4 Available P2P Video Streaming Solutions

1.4.1 PPLive

PPLive is an IPTV (Internet Protocol Television) service that operates in China. It is one of the most widely used IPTV service globally. Although, it has been highly used, technical information regarding the service is not well documented. It is known that, PPLive is a mesh-pull based streaming approach. The system supports multiple channels that have bitrate in the range of 0.25Mbps to 1Mbps. Unfortunately, the technical details of the service is not publicly available [25], [26].

1.4.2 BitTorrent Assisted Video Streaming Service (BASS)

The protocol *BASS* propose a system which operates as a hybrid protocol in which clients receive content from a centralized server. In the meantime, the clients that are receiving the same content are also in a BitTorrent session, which they exchange video chunks [27]. The BASS protocol does not modify the chunk picking policy of *BitTorrent* (rarest-first) except a very simple modification that restricts the client from picking chunks that has already been watched. The main purpose of the system is to decrease the bandwidth requirement of the server side by utilizing the BitTorrent protocol however it does not intend to reach to a point where most of the chunks are delivered over the P2P network. Therefore, the bandwidth requirement is very unlikely to saturate against increasing number of peers.

1.4.3 Enhancing BitTorrent for Supporting Streaming Applications (Bitos)

The authors propose a few more modifications on top of BASS approach. The BiTos protocol divides chunks into two groups: i) high priority chunks that are close to the deadline and ii) remaining chunks. The high priority chunks are downloaded in sequential order whereas the remaining chunks are requested using rarest-first policy. When compared their algorithm against sequential downloading, they have managed to achieve significant improvement, however the protocol still miss about 3% of the video chunks in the best case. Therefore, when the system is not backed with a server side, there would be significant interruptions in video playback. Correspondingly, the authors claim that BitTorrent cannot support video streaming without making fundamental changes [28].

1.4.4 Tribler

Tribler is a P2P video sharing solution that is based on the BitTorrent protocol. It was launched as a European project focused on P2P video streaming as well as introducing social aspects with concepts like friendship (budies) and suggestion mechanism to share. Tribler engine adopts a windowing mechanism, which divides chunks into three categories. In the first group, chunks are downloaded in sequential order similar to BiTos. The duration of this window is about 10 seconds. Next, group of chunks are downloaded in a rarest-first-policy however the policy is restricted to select from chunks that are window 40 seconds of the first group. Finally, the last group of chunks includes rest of the content. If the client downloads both first and second group of chunks, then the system switches to rarest-first-mode with no restrictions. According to its specification, Tribler does not employ the ALF concept since a chunk is filled using chop-and-ship approach [29]-[31].

1.4.5 P2PNext – The NextShare Platform

P2PNext is a European project that targets scalable video distribution over P2P networks to improve received video quality by performing rate adaptation. Their NextShare video platform is based on the Tribler core. The *NextShare* is the name of the P2P engine that the P2PNext project has created. It is one of the first P2P platforms that can perform adaptive streaming using scalable video coding and layered chunks. It employs ALF by padding to align the group of pictures (GOP) boundaries with the fixed sized chunks [32]. They have adopted constant bitrate coding (CBR) technique in order to create bit streams at a given rate such that the padding does not cause much overhead. The original chunk scheduling algorithm of NextShare models the chunk picking as a knapsack problem in which the scheduler has to pick the chunk that provides high highest utility in terms of contribution to rate distortion performance [33]. The major problem with that approach is that the distortion gain for each individual chunk is not always available or easy to compute for a P2P engine, making the system difficult to implement. Later, the approach was updated as *Deftpack* by researchers including authors of knapsack approach [34]. In the new version, the P2P client defines five windows (high-priority, mid-priority, low-priority, lowest-priority and past) and the fundamental idea is as follows. In the first window the chunks are scheduled in certain order (first base layer chunk then using a utility function for enhancement layer chunks). Then, the maximum layer that could have been received in the first window is used as a boundary for the remaining two windows that operate rarest-first fashion. So the rarest-first chunk algorithm is applied to the chunks that the peer is more likely to use. Further references to the project can be found here [35].

1.4.6 LayerP2P

LayerP2P [36] is one of the systems that can exploit scalable coding and perform adaptive streaming. The approach successfully explains how variable length transmission units can be explained in the P2P overlays. However, some of the proposed methods are very difficult to implement. We provide two examples. In *LayerP2P*, peers send cumulative chunk request messages to its neighbors periodically. Within this message, the peer requests all the chunks that is available in its neighbor but not in the peer. Such an approach would lead to huge overhead if a new peer sends a request message to a seeder node as it would lead to requesting all the chunks in the session in a single message. Considering that *LayerP2P* adopts a packet based approach that increases the number of segments that a peer can request, the number of bytes that should be sent in each request message can easily reach critical values. We believe that this fact has been overlooked in the simulation environment. Second, a peer uploads a single chunk at a time, serving a single neighbor. While such an approach would be very efficient for a simulation environment (a peer dedicates all its capacity to upload a single chunk, leading to very fast chunk dissemination), a stop-and-wait type of scheduling over a single connection rarely saturates the link capacity in practice. This is why *BitTorrent* allows multiple simultaneous transmissions.

1.5 3D Video Coding

The method of choice for 3D video encoding depends on the transport mechanism (e.g., backward compatible DVB, IP) and raw video format (e.g., stereoscopic 3D, multi-view video). The Table 1.1 presents the available approaches for based on the classification.

When the stereoscopic content is transmitted over fix-bitrate channels such as DVB, there is no need for rate scalability. Therefore, the non-scalable video codecs can be utilized in order to achieve highest rate-distortion performance. Naturally, the best performance is achieved when views are coded dependently (multi-view-video coding, MVC) in which one view is encoded in H.264/AVC syntax whereas the remaining views are predicted from the first view. At this time of writing, there is no multi-view video transmission over fixed bitrate channels.

Table 1.1: 3D Video coding options based on content format and transmission network

Content Format	**Type of Channel**	
	Fix-bitrate (DVB)	**Variable Bitrate (IP, Blu-ray)**
Stereoscopic 3D	• Frame compatible format using H.264/AVC	• Simulcast (Non-scalable) • Simulcast (Scalable) • MVC
Multi-view video	-	• Simulcast (Non-scalable) • Simulcast (Scalable) • MVC • Video-plus-depth

When the 3D content is to be transmitted over a variable length channel such as IP, the video coding options may include scalable codecs to enable adaptive streaming in which part of the encoded bit streams can be discarded to match the content rate to available link capacity. Moreover, it is also possible to use video-plus-depth coding in the case of multi-view video in which color views are transmitted with a corresponding depth-view so that at the client side some more views can be interpolated using depth-image-based rendering techniques. We now present the details of the above mentioned approaches. We will review some common encoding options for adaptive streaming of 3D video in more detail Chapter 3.

1.5.1 Simulcast Video Coding

Simulcast coding refers to encoding each view independently. Since there is no relation among different views, the content can be independently transmitted to the clients. This approach makes it easier to establish a simple 3D setup but at the cost of losing encoding efficiency. Moreover, it is possible to encode views using the SVC standard and obtain scalable 3D video. Here, two approaches can be followed for scalability: Either all views can be coded scalable, or some views can be coded scalable using SVC and others can be coded non-scalable using H.264/AVC. The latter approach has been employed in asymmetric encoding, as described in Chapter 3.

1.5.2 Dependent Video Coding

Multi-view-video coding (MVC) aims to offer high compression efficiency for MVV by exploiting interview redundancies [37]. It is based on the High Profile of H.264/AVC, and features hierarchical B-pictures and flexible prediction structures [38], [39]. In one extreme, each frame can be predicted only from frames of the same view, which is simply adopting simulcast coding. In another extreme, frame prediction spans all views, which is called full prediction, in which views are predicted from their neighboring views in certain order. This approach increases the rate distortion performance but introduces complexities. A simplified prediction scheme is proposed that restricts inter-view prediction to only anchor pictures, and still achieve similar rate-distortion (RD) performances [40]. In MVC, it is important to perform proper illumination compensation either by pre-processing or by weighted inter-view prediction within the coding loop.

Although there has been some work on scalable MVC, they either utilize a subset of scalability options or MVC prediction schemes [41]-[43]. The current implementation of the reference MVC software does not offer any scalability options other than discarding bit stream in certain order which corresponds to temporal and view scalability. It has been noted that large disparity, different camera calibration among views and lightning conditions may adversely affect the performance of MVC.

1.5.3 Multi-view-Plus-Depth Coding

The main idea of video-plus-depth coding is to render some of the views at the client side instead of transmitting additional views to reduce the overall bitrate requirement [44]-

[46]. In multi-view-plus-depth encoding, selected views and associated depth maps can be can either simulcast or dependently encoded using non-scalable or scalable codecs. It is also possible to exploit correlations between the texture video and associated depth maps. For example, in [47] SVC is employed to compress texture videos and associated depth maps jointly, where up to 0.97 dB gain is achieved for the coded depth maps, compared with the simulcast scheme. Joint coding approaches most commonly target sharing some entities between the color (view) component and the depth component, such as the motion vectors. Nevertheless, there are some handicaps in making use of shared motion vector information between the two components. One of them is that the motion vectors computed during rate-distortion optimization process are selected to minimize the energy of the texture residual, which does not show 100% correlation for two components.

1.6 Contributions of This Thesis

The key contributions of this thesis can be listed as follows:

- **Design of a P2P architecture for adaptive streaming** with the following key features:
  - **Variable length chunks** that minimizes the effect of chunk loss events
  - **Variable-size scheduling** that maximizes the randomness in chunk scheduling policy and increases the chance of having distinct chunks between neighbors to increase the rate of chunk exchange
  - **Novel incentive mechanism** in which the peers can use the sharing history of their neighbors in the optimistic unchoking policy to allow a peer with good history to join a new session much faster
  - **QoE aware rate adaptation for stereoscopic 3D and multi-view video**

- **Implementation of the proposed architecture** with following additional features:
  - **Pipelining** to schedule multiple chunks to saturate link capacity
  - **Deadline sensitive chunk scheduling** to avoid scheduling chunks from a peer if it is likely to be late
  - **Web seeding** support
  - **Hybrid data delivery** support and ability to synchronize with DVB channel

- Streaming tests in controlled LAN environment to evaluate the rate adaptation capability
- Streaming tests in PlanetLab environment
  - Evaluation of the proposed architecture in the Internet environment
  - Comparison of the proposed architecture against available solutions.
- Comparison of scalable video coding vs. multiple description coding for P2P video streaming to determine which approach performs better under what conditions. (shared work with K.Tolga Bagci)
- Evaluation of adaptation strategies for streaming stereoscopic 3D video. The evaluation is performed with consideration of different display setups. (Shared work with Gorkem Saygili)
- Evaluation of adaptation strategies for streaming multi-view video. Subjective tests have been performed using multiple streams scaled using various adaptation methods. (Shared work with S. Saadet Savas)

- Modifications to OpenSVC decoder for the DIOMEDES project.
  - Development of a multi-threaded scalable video decoder for low power consumption on mobile devices with multi-core processors. (Shared work with Damla Kilicarslan).
  - Implementation of networking features to receive elementary stream and forward raw images
  - Modification to decoded buffer structure to be able to stitch timestamp information to video frames (slices).

Chapter 2

ADAPTIVE VIDEO STREAMING OVER P2P NETWORKS

Traffic generated by streaming video services remains as the biggest contributor to Internet traffic [48]. With no foreseeable reduction in customer demand and high return value of services due to personalized advertisements, the video over IP industry is expanding and is expected to generate 48 Exabytes of IP traffic per month by 2015, making scalability a key factor for companies to be successful in future video applications [49].

P2P streaming is a potential solution for decreasing traffic at the server side (also known as *seed server*) by leveraging the upload capability of clients. P2P systems, such as Napster, Gnutella, Kazaa, and BitTorrent, have proven successful to share large files. Several propositions for extending P2P systems to video streaming exist, but whether P2P can repeat the same success for streaming time sensitive media is still an open question. The diversity of proposals and difficulty of comparing them are some of the challenges in finding the most successful proposal. In the following, we provide four questions to classify the P2P solutions.

The first question is “What is the topology of the P2P overlay?” or in other words “How peers are connected to each other?” P2P approaches can be classified as tree-based, mesh-based or hybrid. In the tree-based solutions, content flows from a single node (root) to peers (leaves) that are connected to each other in parent-child fashion. Data flow starts immediately once a peer joins the tree, allowing low latency data dissemination. The main problem with the tree-based solutions is the difficulty of maintaining the optimum tree structure in case of peer exits and bandwidth fluctuations. We note that commercial P2P deployments that rely purely on tree-based solutions do not exist [50]. On the other hand, in mesh-based solutions, peer connections have no parent-child relationship and a single peer can have multiple connections but may require some time in establishing peer connections. A comprehensive comparison of the P2P solutions for video streaming shows that mesh-based methods consistently exhibit superior performance over tree-based methods [51].

The second question is “Is the transmission sender-driven or receiver-driven?” The traditional video streaming is sender-driven, where the server side pushes multimedia packets to the receiver. However, the state of the art streaming systems such as dynamic adaptive streaming over HTTP [52] are receiver-driven, where the video is encoded at different qualities in multiple streams and resulting streams are split into pieces (segments or chunks). Then, the receiver side performs rate adaptation by requesting (scheduling) the next chunk at a quality that matches the current rate of data reception. This approach frees the sender side from keeping record for the link with each recipient at the expense of chunk request messages.

The third question is “What is the format of the transmission unit?” The transmission unit is the smallest entity that can be exchanged between two peers and notified to neighbors when successfully received. There are two options: First option is a “network packet” that is bounded by the size of the maximum transmission unit (MTU). With 1500 bytes as a typical size, a packet can contain a small frame or a part of it. The second option is a “chunk” that contains video segment with duration in the order of seconds. Actually, the answer to the previous question has implications on the choice of transmission unit. When the system is sender driven, small packets enable fine adaptation capability. However, small packets cause significant overhead due to scheduling and notify messages in receiver-driven systems, making chunk a more favorable approach for receiver-driven systems.

The fourth question is “What is the main motivation for peers to devote their upload capacity for dissemination of the data” or in other words, “What is the incentive mechanism?” The incentive mechanism should provide benefits to good participants (peers) for their contributions and motivate them to continue to share more of their network resources. Here, a good peer refers to one that uploads (shares) at a level close to or more than their download rate. There are a variety of proposals for providing incentive mechanism. In most of them, peers keep statistics about their neighbors’ contribution rate, (which can be in terms of number of chunks or achieved bitrate) and reject chunk requests of those neighbors with low contribution.

Based on the above classification and evaluation of currently available P2P solutions, we have decided to build our proposed solution on top of the BitTorrent protocol [24], which is known for its superior efficiency when compared to other approaches [53]. It is a receiver-driven, mesh-based topology that creates robustness against peer exit events. The protocol uses equally sized chunks (256Kbyte, 512Kbyte are typical values) as transmission unit and perform chunking in a totally content agnostic manner, which is natural for a file delivery service. As incentive mechanism it uses the tit-for-tat policy; a peer sorts its neighbors based on their contribution in bitrate and reject chunk requests from those below a certain rank. The only exception is the optimistic un-choking policy in which a peer selects a lucky neighbor and accepts its requests even if the neighbor has not provided any data [54]. This causes a new peer to slowly become active in the session.

BitTorrent has been designed to share files and not for video streaming. In this work, we extend BitTorrent by introducing: i) variable length chunks as the transmission unit ii) a buffer driven chunk scheduling policy iii) a score based centralized incentive mechanism to develop a new adaptive P2P video streaming system, called AdaptP2P. Variable length chunk format minimizes the effect of data loss by aligning the GOP boundaries with the chunks for proper application layer framing. The buffer driven scheduler ensures rate adaptive video delivery and minimizes the bandwidth consumption at the seed server that is treated differently from other seeder peers. The scheduling policy is also combined with a centralized incentive mechanism to restrict the video service to a quality that is proportionate to peers' contribution in content distribution. The incentive mechanism is centralized, utilizes the previous sharing performances of the peers to increase the rate that a peer receives chunks from its neighbors when it has just joined a session which is designed to avoid long delays due to tit-for-tat mechanism. We compare these features with Bass [27], Bitos [28], Tribler [29]-[31], NextShare [32], [33], Deftpack [34][35] and LayerP2P [36] and provide some technical analysis when possible.

The remainder of this chapter introduces the three main features of AdaptP2P. Section 2.1 explains the variable length chunk format and compares it against fixed-length chunks. In Section 2.2, we provide the details of buffer driven adaptive chunk scheduling policy and provide results on its rate adaptation capability and resulting bandwidth consumption at the seed server. In Section 2.3, we present the proposed incentive mechanism explaining the methods to achieve reliable bookkeeping. We provide results to show the relation between received video quality and the rate of data a peer provides. Finally, we summarize the Chapter in the last section.

2.1 Transmission Unit: Variable Length Chunks with Prioritization

In P2P video streaming, chunk generation refers to mapping the payload, media stream, into chunks. Because P2P clients cannot use partially received chunks, the format of the chunk has a significant effect over the performance of the system. In the BitTorrent protocol, the size of chunks is fixed, which has also been adopted in BitTorrent compatible streaming derivatives such as BASS, Bitos, Tribler and NextShare. On the other hand, we employ variable size chunks to properly implement the application-layer framing principle and provide better decoding performance in case of chunk losses. Different transmission unit formats are summarized in Table 2.1. We present their performance in Table 2.2 and Table 2.3 below.

2.1.1 Proposed Variable-Length Chunk Format

We employ variable length chunks as the transmission unit mainly for two reasons: First, to minimize error propagation in case of chunk losses. The state of the art video codecs use predictive encoding and entropy coding to achieve higher compression efficiency. If chunking is performed in a content agnostic manner (e.g., using *chop-and-ship* with fixed sized chunks), a lost chunk is likely to nullify the content of some other received chunks because half received network abstraction layer (NAL) units cannot be decoded. More importantly, any received data that refers to a lost chunk is also useless due to predictive coding. Using variable length chunks, the content can be split into self-decodable blocks in accordance to the application layer framing principle. For example, each chunk may consist of an integral number of group-of-pictures (GoP) [55] to avoid error propagation to the neighboring chunks.

The second advantage of using variable length chunks is to be able to fix the duration of video in each chunk. Then, the duration of video per chunk can be determined without parsing the bit stream, which allows the P2P engine to accurately estimate the duration of video in its ready-to-play buffer to improve adaptation capabilities, which are discussed in more detail in Section III (e.g., if the ready-to-play buffer duration is high, a scheduler may download more enhancement layers).

On the other hand, using variable sized chunks slightly complicates the protocol; either the size of each chunk should be recorded in the metadata file (just like their hash values) or peers should exchange chunk size information prior to their transmission. In our implementation, peers send size information in the chunk header (4 extra bytes).

Table 2.1: Transmission units in mesh-based P2P video streaming solutions

Protocol	**Transmission Units**	**Abbreviation**
Bass	Fixed Length Chunks (Chop-and-ship)	FLC (CnS)
Bitos	Fixed Length Chunks (Chop-and-ship)	FLC (CnS)
Tribler	Fixed Length Chunks (Chop-and-ship)	FLC (CnS)
NextShare	Fixed Length Chunks (Padding and CBR Coding)	FLC (Pad&CBR)
LayerP2P	RTP/UDP/IP Packets	RTP Packets
AdaptP2P	Variable Length Chunks	VLC

2.1.2 Available Chunk Formats

2.1.2.1 Fixed Length Chunks

A simple method is to perform packetization with fixed length chunks. BitTorrent and its video streaming derivatives (e.g., Bass, Bitos, Tribler and NextShare) use this approach with chunk length as power of two. However, fixed length packetization is not the best choice since the media GoPs do not properly align with chunk boundaries. In such a case, there are two possible options: i) In the *chop-and-ship* method, video is split into fixed chunks without respecting GOP boundaries, leading to chunks with partial GOPs. ii) In *padding* method, the GOPS are kept intact and not distributed over multiple chunks using padding. The details are provided in the following subsections.

Chop and Ship Method: *Chop-and-ship* packetization refers to mapping video bit stream into fixed size chunks without considering GOP boundaries. There are two drawbacks. First, when a chunk is lost or skipped due to the play-out deadline, it is very likely that the content of neighboring chunks will also be affected even if they have been received correctly because partial NAL units cannot be decoded. Second, the P2P engine cannot determine the duration of the chunks in its buffer. As a result, it becomes very difficult to make critical decisions to provide uninterrupted video play-out at the highest possible quality. The document for Tribler Protocol Specification states that buffer under-run may occur immediately after a re-buffering period [56] because when chunk durations are not fixed, the P2P engine cannot determine the number of chunks it needs to buffer before resuming play-out. Nevertheless, chop and ship method is one of the most commonly adopted policies, probably due to its simple implementation. BASS, BITOS and Tribler use chop-and-ship method for mapping video stream to BitTorrent chunks.

CBR Coding and Padding: Another approach creates chunks that contain fixed number of GOPs by using padding to fill leftover bits, respecting the GOP boundaries [33]. In order to keep amount of padding predictable and low, NextShare uses constant bitrate (CBR) encoding, which is commonly used for video transmission over fixed bitrate channels (e.g., DVB broadcasting). With these features, NextShare can avoid error propagation between chunks and obtain chunks with equal durations and perform accurate adaptation.

However, CBR encoding, which allocates equal number of bits to each GOP, causes loss of coding efficiency. Consider encoding a video where some scenes have high motion/high complexity and other scenes have low motion/ low complexity. Then, at constant bitrate, either high complexity parts will have low quality or low motion parts will have unnecessarily high quality although they could have been encoded with almost the same perceived quality at a lower bitrate. Alternatively, if variable length coding is used instead of CBR coding, the amount of padding may reach critical values, especially for videos that have non-uniform complexity. Moreover, using fixed sized chunks without CBR coding leads to significant overhead as investigated in [55].

2.1.2.2 Packet Based Transmission Units

LayerP2P [36] and some other P2P streaming proposals [57], [58] use RTP packets as the transmission unit. However, there are side effects of using small transmission units in pull-based systems. One of the key functionalities of the metadata file is to introduce content to the receiving peer, so that receiver can *pull* (request) data from other peers. Small transmission units increase the size of the metadata file and more importantly the rate at which peers exchange request and notify messages. A valid proof of this claim can be found in studies over dynamic adaptive http streaming arguments, which is a pull-based system over http [52] and also from studies that focus on chunk sizing in BitTorrent. We also note that almost all RTP based receiver driven systems are evaluated using simulations where such inefficiencies can be overlooked.

2.1.3 Evaluation of Variable Length Chunks vs. Available Formats

2.1.3.1 Comparison of Variable Length Chunks vs. Chop and Ship Method

We have evaluated the decoding performance when a content is chunked using both chop and ship method and variable lengths over GOP boundaries and then streamed over a bandwidth limited channel. The variable length chunks are composed of two GOPs whereas the fixed length chunks have size of 128K bytes. We have used the Network Emulator (NetEM) library available in Linux Kernel and limit channel capacity of a link between two nodes. The receiver actively tries to schedule as many chunks as possible but still miss some due to lack of available bitrate. We have used frame repetition to fill missing frames and achieve reliable PSNR measurement. For this purpose, we have modified the ffmpeg library and recorded decoded picture-order-count (POC) in addition to the decoded pictures. Table 2.2 provides the average statistics over 50 emulation tests and reveals the difference in achieved PSNR quality between two approaches, favoring the variable length chunks.

Table 2.2: Average results over 50 realizations

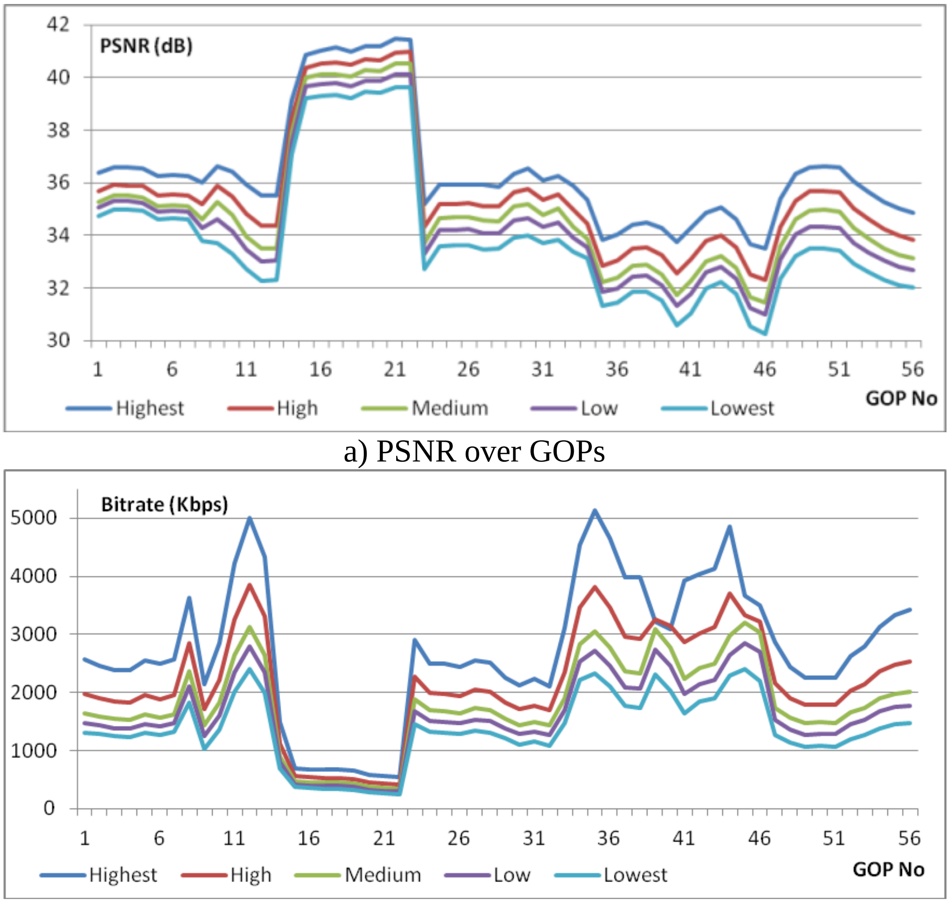
	Fix Length Chunks	**Variable Length Chunks**	**Lossless (benchmark)**
# decoded frames	720.36	769.45	897
PSNR	37.55	38.73	39.99

2.1.3.2 Comparison of Variable Length Chunks vs. CBR and Padding Method

In this section, we demonstrate the adverse effect of CBR coding in a variable bitrate channel such as IP network. We have encoded the sample movie `wildlife.wmv` (freely available in Windows 7, resolution*1280x720*) using the reference H.264/AVC encoder. We have performed variable bitrate coding (VBR) with 5 different quantization parameters. Figure 2.1-a depicts the change in quality (PSNR) over GOPs and Figure 2.1-b presents the bitrates of the corresponding streams for the same intervals.

If CBR mode is utilized instead of VBR and minimum 32dB PSNR value is set for quality assurance, then red line or any stream above red line can be selected as they would satisfy the minimum quality requirement. Assuming that the stream that corresponds to the high quality is chosen, the coding rate must be about 4 Mbps which is the peak of the red line in Figure 2.1-b. However, the average bitrate of the VBR encoded bitstream is about 2.2 Mbps almost half of the rate for CBR. As a secondary argument, we present the bitrate that is provided in the study that introduces the chunk scheduling policy of NextShare [32]. The authors perform their video streaming tests using a sequence that has resolution 640x480 (low resolution with only 1/3 of 720p content) is encoded at 3Mbps. This bitrate is considerably high as a direct consequence of using constant bitrate coding.

We present the summary of our evaluations in Table 2.3. We believe that variable length chunking has negligible messaging overhead when compared to the overhead of CBR and padding. We also believe that variable length chunking delivers improved performance over *chop-and-ship* method in terms of error recovery. Variable length chunks also maintain feasible message exchange rates when compared against packet based solutions. For instance, using a chunk composed of two GOPs of size 16 frames requires almost a single request message (chunk has duration of ~1 second at 30fps). In a RTP packet based approach, the number of requests quickly ramps to 30 times one request per each frame (RTP packet).



a) PSNR over GOPs

b) Bitrate over GOPs

Figure 2.1: Rate distortion performance for Wildlife (1 GOP ~= 0.5 sec)

Table 2.3: Summary of the evaluations of the different transmission units

	Cons	Pros
Fixed Length Chunks (Chop and Ship)	Error propagation in chunk loss	Simple
Fixed Length Chunks (Padding &CBR)	Lower encoding efficiency. Packetization overhead	Chunks have equal duration. Good for adaptation purposes.
RTP Packets	High number of message exchange among peer.	Minimized effect of chunk loss
VLC	Chunk sizes must be negotiated	Decreased effect of chunk loss Chunks have equal duration that P2P engine has access

2.2 Adaptive Chunk Scheduling

The chunk scheduling policy is perhaps the most important aspect of mesh-based P2P video streaming systems. It is the prime reason that makes BitTorrent unsuitable for multimedia delivery and therefore addressed by almost all P2P video streaming proposals. We start by introducing the proposed chunk scheduling mechanism for AdaptP2P. We then provide an overview of currently available solutions and explain the key differences between the proposed and available solutions. Finally, we present streaming test results that are performed over the PlanetLab environment to evaluate AdaptP2P against two currently available solutions: Tribler and Deftpack.

2.2.1 Proposed Chunk Scheduling Mechanism

The BitTorrent protocol is designed to share large binary files that can be used once the download is complete. A peer schedules chunks that are least distributed in the swarm with the intention of obtaining some chunks that can be traded for others. This approach is not suitable for media because presentation deadlines of chunks are not considered in the scheduling process.

AdaptP2P has two fundamental modifications to the BitTorrent protocol regarding chunk scheduling policy. First, AdaptP2P distinguishes the content originator (seed server) from other peers in the swarm including the peers that become a seeder over time. This distinction allows AdaptP2P to intelligently schedule the chunks in a way that minimizes the workload on the seed server. Second, AdaptP2P uses a windowing mechanism to achieve timeliness in chunk scheduling procedure. In the following, we provide the details of the windowing mechanism in AdaptP2P.

There are two scheduling windows in AdaptP2P. The first window is called the *variable-length window*. It is responsible for both rate adaptation and introducing randomness in chunk picking process as much as possible. The *variable-length window* starts from next chunk to be delivered to media player. The window has a variable length that is determined based on the available buffer duration. The size of the window can be as low as 1 chunk (implicitly forcing sequential downloading) and can grow up to a parameter called *maxWindowSize* (w_{MAX}) that is set to a value such that the window duration is about twenty seconds. All the chunks within the *variable length window* are downloaded with unbiased randomization unlike most other protocol in which the first windows enforce scheduling in certain order. The prioritization among chunks is handled by adjusting the size of the windows.

If all the chunks in the variable-length-window is downloaded (or being downloaded), then the scheduler starts to pick chunks from the next window (*rarest-first-window*) that contains all the remaining chunks within the session. The scheduler picks the chunk that is least distributed in the swarm to increase the possibility of chunk exchange among peers in the *rarest-first-window*. Moreover, the pace of the chunk reception from the seed server is decreased to allow the seed server to respond more urgent chunk requests. In the following subsections, we describe how the size of the *variable-length window* is determined and how rate adaptation is performed to match the content bitrate to the available network rate if scalable video coding is adopted.

Buffer driven window sizing: Besides maintaining timely content delivery, the scheduling window is also responsible for introducing randomness in chunk scheduling and increasing the possibility of having distinct chunks in different peers so that they can engage in chunk exchange (referred as the P2P activity). Large window is good for increasing the P2P activity because peers are more likely to obtain different chunks but it can also jeopardize the chance of satisfying the real-time requirements since the scheduler can be late to pick some of the chunks. On the other hand, short window duration limits the chance of having distinct chunks among peers and decreases P2P activity. The best case is to have the window size as large as possible but should respect the timeliness requirements.

The duration of ready-to-play buffer is the parameter that we use to adjust the window size. It refers to the duration of downloaded chunks that can be presented to the user continuously starting from the next chunk after the last displayed chunk for the active layers (as explained in the next subsection). When buffer duration is high, peers have more time to receive the chunks therefore a longer window can be tolerated. In the opposite case, it is best to have a short window to force the scheduler to pick chunks that are about to be consumed by the media player. In the extreme case, if the buffer duration is very short, the window size is set to 1, forcing the scheduling to operating in total sequential downloading. The key idea in our approach is that, as soon as the buffer duration increases, peers increase the window size and introduce some randomness in chunk scheduling policy.

The following equation can be used to determine the window size (w_s in number of chunks). The size of the window should be equal (or less) than the expected number of chunks that can be downloaded within the duration of the buffer (b_d in seconds). The number of bytes in a window is equal to average chunk size (c_s) times the window size. The download rate ($d_r$in byte per sec.) can be computed and it is possible to achieve the equality in (1) and then compute the size of the window correspondingly.

$$b_d = \frac{window\ size\ in\ bytes}{download\ rate} = \left(\frac{c_s \times\ w_s}{d_r}\right) \tag{2.1}$$

Variable window length is a strong method that strikes a balance between timely data reception and introducing randomness in an adaptive manner. However, it has its own limits in practical use especially if some of the peers have joined the session earlier and in far ahead in time for a newcomer. In such cases, the new peer may find it very difficult to extend the scheduling window to a size, which it can schedule a chunk that is not received by an “elder” peer yet. In other words, variable length windowing is good for introducing randomness among peers that have joined the session within a close interval but may fail to do the same for peers that are separated with significant time durations. For this purpose, we limit maximum size of the *variable-length window*. If all chunks up to this time are downloaded then the peer schedules chunks in rarest-first-fashion.

Buffer driven rate adaptation within variable-length window: In server-client architecture, servers commonly possess high upload capacities while the clients dedicate more bandwidth for downloading (e.g., asymmetric DSL connection), maximizing the average rate of data transmission. This approach improves quality of the connection between the server and client but makes it even more challenging to establish successful symmetric connections in P2P solutions. With limited upload capacity, peers are more prone to rate fluctuations and therefore, performing adaptive streaming is even important for a P2P-based systems.

Adaptive streaming is best performed when encoded bit stream can be divided into sub-streams according to their priority in the decoding process. Then, the least important sub-stream can be discarded for rate adaptation purposes at the cost of degradation in perceived video quality. In AdaptP2P, we do not enforce a method to split an encoded bit stream into prioritized streams of chunks but we provide an example for scalable encoded video sequence. It is also possible to extend for 3D as proposed in [59] to perform adaptive 3D video streaming over P2P.

The bit stream of SVC encoding starts with non-VCL (video coding layer) NAL units that provide information such as frame resolution. These NAL units are repeated at the beginning of each GOP and thus included in every chunk. After the non-VCL NAL units, there exists one base layer NAL unit and possibly multiple enhancement layer NAL units for each frame (if slice mode is off). Then, two types of chunks can be generated: i) base layer chunks are obtained by merging base layer NAL units for a GOP and prioritized ii) enhancement layer chunks are obtained by merging particular enhancement layer NAL units and can be discarded in case of bandwidth scarcity.

Once the chunks of different streams are generated the rate adaptation using buffer duration works as follows: When a streaming session is first initialized, all layers are active and peers buffer the content at highest quality. Once chunks of all the layers are downloaded until the desired buffering location, the player starts to consume the chunks. From now on, the number of active layers is updated each time a chunk is to be scheduled. If the buffer duration is below a certain threshold (*BufferThresholdLOW*), then the layer with the least priority is disabled. In the opposite case, if the duration of the buffer is above a certain threshold for all active streams (*BufferThresholdHI*) then the layer that has the highest priority among the disabled ones is again enabled. There is one important detail in this process. When measuring the buffer to disable a layer, the least prioritized stream that is possibly going to be disabled is not included in the calculation. This approach avoids rapid fluctuations in the number of layers. Otherwise, when a new layer is enabled the buffer may shrink to a level below *BufferThresholdLOW* value.

In order clarify how buffer duration affects both window sizing and the number of active layers, we introduce the following scenario. Consider a peer that has low buffer duration. The window size is 1 and there is only 1 active window, i.e. delivering the base layer chunks. If the data reception rate increases, the buffer duration reaches to *BufferThresholdHI and* window size grows to a certain size (w_{th}). Then, another layer is activated again at same size. Following that period, the buffer duration increases again if the reception rate is high enough and consequently more layers are enabled. If the buffer duration increases when all the layers are enabled and, the size of the variable-length-window are expended until it reaches *maxWindowSize*. Then, if the all chunks in variable-length-window are being downloaded, then more chunks are requested using rarest-first policy. Figure 2.2 summarizes the state of the *variable-length-window* for increasing bandwidth availability.

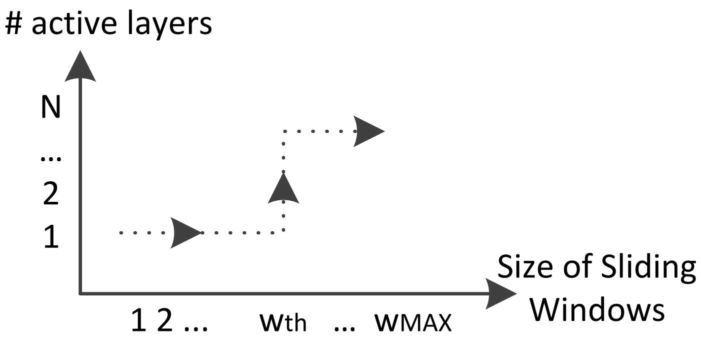


Figure 2.2: Adjusting the size of variable-length window and number of active streams

2.2.2 Available Methods

Non-Adaptive P2P Streaming Systems: The protocol *BASS* [27], does not modify the chunk picking policy of *BitTorrent* (rarest-first) except for scheduling chunks only if the presentation time has already surpassed yet. It may not be fair to compare this algorithm with the adopted procedure in *AdaptP2P* because the service that *BASS* considers P2P only as an augmenting mechanism to decreases the bandwidth requirement. In *Bitos* [28], the authors claim that BitTorrent cannot support video streaming without making fundamental changes and we agree with this claim. The authors propose a simple modification that divides chunks into two groups: i) high priority chunks that are close to the deadline and ii) remaining chunks. When compared against sequential downloading, they have managed to achieve significant improvement, however still lose around 3% of the data in the best case of their simulation results. *Tribler* is a European Union project for streaming P2P video with live streaming support. It outperforms both *BASS* and *Bitos* by using three cascaded scheduling windows [31]. The first (*critical*) window has a short duration in which the chunks are downloaded in sequential order. If this window is complete, then rarest-first scheduling is applied in a time-limited secondary window (size is four times the critical window). Finally, rarest-first scheduling is applied without any boundaries.

Adaptive P2P Streaming Systems: *LayerP2P* is one of the systems that can exploit scalable coding and perform adaptive streaming. It is also the only approach, which we have omitted to implement due to some of its less implementable (in practice) features. We provide two examples. In *LayerP2P*, peers send cumulative chunk request messages to its neighbors periodically. Within this message, the peer requests all the chunks that is available in its neighbor but not in the peer. Such an approach would lead to huge overhead if a new peer sends a request message to a seeder node as it would lead to requesting all the chunks in the session in a single message. Considering that *LayerP2P* adopts a packet based approach that increases the number of segments that a peer can request, the number of bytes that should be sent in each request message can easily reach critical values. We believe that this fact has been overlooked in the simulation environment. Second, a peer uploads a single chunk at a time, serving a single neighbor. While such an approach would be very efficient for a simulation environment (a peer dedicates all its capacity to upload a single chunk, leading to very fast chunk dissemination), a stop-and-wait type of scheduling over a single connection rarely saturates the link capacity in practice. This is why *BitTorrent* allows multiple simultaneous transmissions.

The *NextShare* is one of the first P2P platforms that can perform adaptive streaming using scalable video coding and layered chunks [33]. It models the chunk picking as a knapsack problem in which the scheduler has to pick the chunk that provides high highest utility in terms of contribution to rate distortion performance. The major problem with that approach is that the distortion gain for each individual chunk is not always available or easy to compute for a P2P engine, making the system less implementable. Later, the approach was updated as *Deftpack* [34] by researchers including authors of knapsack approach. In the new version, the P2P client defines five windows (high-priority, mid-priority, low-priority, lowest-priority and past) and the fundamental idea is as follows. In the first window the chunks are scheduled in certain order (first base layer chunk then using a utility function for enhancement layer chunks). Then, the maximum layer that could have been received in the first window is used as a boundary for the remaining two windows that operate rarest-first fashion. So the rarest-first chunk algorithm is applied to the chunks that the peer is more likely to use.

2.2.3 Evaluation of Proposed Chunk Scheduling Mechanism

We have conducted streaming tests over the PlanetLab platform, which consists of globally distributed nodes that are connected with each other without firewall restrictions, in order to compare the proposed scheduling algorithm in AdaptP2P versus i) that of

Tribler (as the best non-adaptive P2P solution [31]) and ii) ***Deftpack*** (as the state of the art adaptive chunk scheduling algorithm [34]). The evaluation criteria are the average delivered quality and the rate of data transmission from the seed server, which is measured as the number of received chunks.

Test Methodology: Although the PlanetLab environment is a very useful platform to conduct realistic tests, it also poses serious challenges that are commonly not present in a simulation environment. One of those challenges is the variations of test conditions that affect the test results in time. There are two main sources of variations: The first one is due to the disparity in the link capacities of the nodes, which may become unavailable sometimes. Secondly, even if the same nodes are used, there can still be significant variations in available link capacity because the nodes can be used by different PlanetLab users simultaneously. In order to minimize the effect of these variations, we have adopted the following policies.

First, we define a term, *test step* which corresponds to performing streaming tests over the same nodes using both evaluated protocols one after the other in random order. Then, we evaluate two protocols by repeating test steps multiple times. In this way, tested protocols are frequently switched and a possible variation is likely to affect both of them. In addition to that we adjust the bitrate of test content according to the average data reception capacity of the peers. To this end, at the beginning of each *test step*, we sort all available nodes according to their download capacity from the seed server and select ten nodes with the highest values for the next test step. At the same time, measure the average data reception rate of selected nodes from the seed server and generate test content accordingly (details are explained in the following subsections). We have observed that choosing content bitrate relative to average node capacity significantly improves the stability of the results and makes them repeatable at different times.

Test Scenarios: We have two fundamental test scenarios. First, we evaluate the performance of AdaptP2P vs. Tribler using single-layer content and then vs. Deftpack using multi-layer content. We consider three different cases. Case I refers to low bitrate content, Case II refers to medium bitrate content and Case III refers to high bitrate content with details provided in Table 2.4. The data reception rate (DRR) refers to the average rate of transmission of selected peers from the seed server. We note that DRR does not represent the maximum link capacity for a peer because link capacity with the other peers can be higher than the connection with the seed server. Moreover, we do not consider Case II for single layer content because it does not introduce significant change in the bitrate to affect the results.

Table 2.4: Test content bitrate with respect to average DDR

CASE	Single-Layer Content Scenario	Three-Layer Content Scenario
I	Content bitrate = 50% of the DRR	Sum of all layers = 50% DRR
II	-	Sum of all layers = 100% DRR
III	Content Bitrate = 90% of the DRR	Base Layer Bitrate = 90% DRR

The content duration is 5 minutes, kept intentionally short to minimize the delay between different protocols within each test step. The video playback begins after a peer buffers content of 5 seconds. If a base-layer chunk is missing, we do not employ re-buffering; instead scheduling windows are slid according to the data reception rate and the bitrate of the base layer. There are two chunks per layer per second, making 600 chunks for the single-layer content and 1800 chunks for the three-layer content. The bitrate of the layers are equally distributed.

In the BitTorrent protocol, a peer may have received the address of a high number of peers from the tracker server however the actual number of active connections are very limited (Section 6.3 Number of Active Connections in [60]). Accordingly, we have created a swarm of 10 peers each having 9 neighbors and 5 active connections. Peers replace the lowest 2 contributors periodically from the 4 non-active connections. We employ the same incentive policy to eliminate the effect of peer selection for uploading and focus on the performance of chunk scheduling policies. Using tit-for-tat policy, a peer assigns 4 upload slots to the best contributors and one more slot to a *lucky* neighbor to perform optimistic unchoking. Peers accept chunk requests only from neighbors with assigned slots.

Implementation of the Evaluated Approaches: ***Tribler*** has three scheduling windows: the sizes of the first two windows are set to 10 and 40 seconds, respectively, while the last window contains the entire remaining chunk [31]. In the first window, chunks are scheduled in sequential order and the other two windows rarest-first policy is adopted. ***Deftpack*** has four scheduling windows: the size of the first two windows is again set to 10 and 40 seconds. In the first window, the base layer chunks are requested in sequential order while enhancement layer chunks are requested using the utility function according to *Equation 1* in [34] but parameter dp_i is omitted because its computation is not provided in the text. In all remaining windows, rarest-first policy is adopted. The vertical limits of windows are set according to the descriptions in [34]. The past window is disabled because the other protocols do not download late chunks once the playback is over.

Results and Interpretations: Figure 2.3 and Figure 2.4 presents the comparison of the average of received layer between AdaptP2P vs. Tribler and AdaptP2P vs. Deftpack, respectively. Similarly Figure 2.5 and Figure 2.6 provide the number of chunks that are scheduled: a) from the seed server and b) from the neighboring peers for the two scenarios. Based on these results, we have the following observations:

- When the available bitrate is significantly higher than the bitrate of the content (CASE I), all solutions provide the maximum quality but AdaptP2P uses less server resources.
- When the available bitrate is not enough to receive the maximum quality from the seed server, the number of chunks that are received from the seed server is close to each other but AdaptP2P still has the lowest score, while providing the best quality (thanks to more chunks that are received from the neighbors).
- In all cases, rate of chunk exchange among peers is higher with AdaptP2P. The difference is more significant when the bitrate of the content is close to the data reception rate of the peers from the seed server (CASE III).

The superior results provided by AdaptP2P are due to the following reasons:

1. AdaptP2P differentiates the seed server (which can be considered as the content originator) from the other peers in the network, allowing it to make intelligent decisions that can decrease the chunk request. For instance, when the buffer duration is high, the AdaptP2P client decreases the frequency of chunk request from the seed server and schedules chunks that are least distributed in the swarm.
2. Both Tribler and Deftpack use a window in which the chunks are downloaded in certain order. In Tribler, sequential downloading is adopted. In the Deftpack, it is sequential for base layer chunks and using a utility function for enhancement layer chunks but since the outcome of the utility function is the same for all peers still there exists a certain ordering. When that is the case, the peers are more likely to schedule the same chunks. Moreover, if the content bitrate is high enough to avoid the protocol to switch to other windows, the whole session is downloaded in a certain order. On the other hand, AdaptP2P does not explicitly adopt sequential downloading at any stage. Instead it uses

a variable length windowing based on its buffer duration. When the buffer duration is very short, the window shrinks to contain only a single chunk, which corresponds to sequential downloading. However, as soon as the window size increases, randomization is introduced, making peers more likely to choose different chunks.

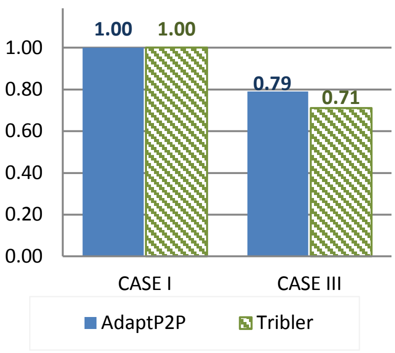
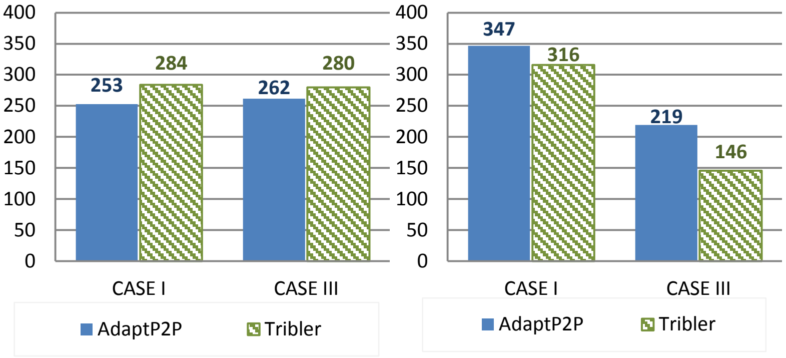


Figure 2.3: Average deceived layers



a) from seed server b) from neighboring peers

Figure 2.4: Number of chunks downloaded

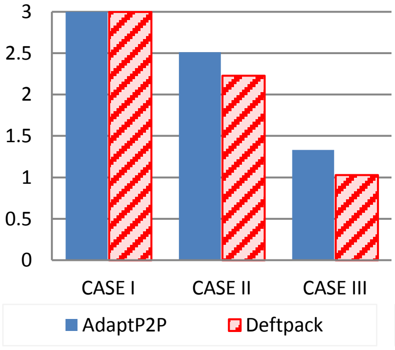
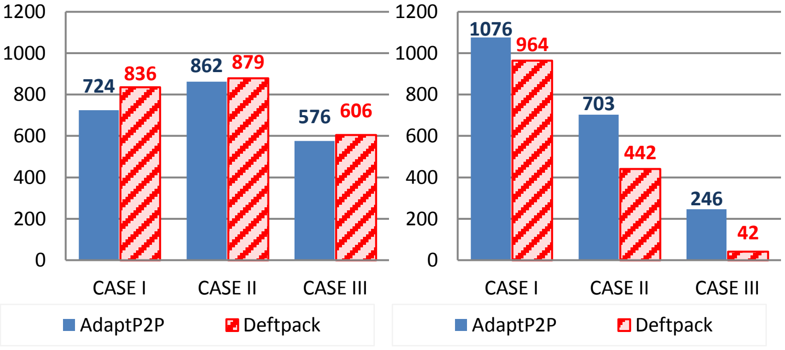


Figure 2.5: Average received layers



a) from seed server b) from neighboring Peers

Figure 2.6: Number of chunks downloaded

2.3 Incentive Mechanism: Peer Selection

2.3.1 Problem Definition

When peers engage in rational behavior and try to maximize their own net profit by minimizing their contribution, they can become a free-rider and pose a serious threat to content dissemination. To discourage from such activity, P2P systems employ incentive mechanisms and provide higher quality of service to those peers which contribute more than the others. The relative increase in the QoS is achieved by restricting peers from forwarding data to their neighbors with bad statistics until they utilize more of their upload capacity. This is known as choking. In the end, a peer that shares more of its resources with its neighbors is less likely to be choked and achieves higher data reception rates.

The problem with this approach is that, when the statistics of a peer is unknown (e.g., a peer joins to a new session) peers have to experience a slow start period regardless of their past behavior because neighbors are skeptic about a peer with no background information to avoid clean sheet attacks. In such a case, a new peer has to wait until optimistically unchoked by one of its neighbors and receives a chunk that it can exchange to build trust. This period (from the time that a peer joins the session until it gains the trust of its neighbors) is highly inefficient and network resources are underutilized. Consequently, users experience longer buffering delays each time they switch from one content (session) to another.

2.3.2 Proposed Unchoking Algorithm

We propose that tracker servers should keep record about chunk exchange statistics of the peers (with permanent IDs when supported) and provide these statistics to neighbors whenever requested. Then, peers can use this feedback to decide which neighbors to choke or unchoke when statistical information is not available. Naturally, this approach introduces overhead for the tracking server but it is negligible when compared with the overhead of streaming content over the server-client approach. The overhead can be justified if it increases efficiency and usability of P2P video streaming solutions.

In AdaptP2P, a peer uses the tracker feedback as a weighting coefficient when choosing a neighbor unchoke optimistically. In our implementation, the inactive peers are sorted according to contribution feedback history and inverse of their ranking (e.g., the highest one among N peers receive N) is used as a weighting. Using this method, a peer with good history in integrated to the system much faster when compared to unbiased optimistic unchoking. This would allow a good peer to switch from one content to another must faster, creating yet another incentive mechanism to share his/her resources. In addition to these, AdaptP2P incentive mechanism has two regulations regarding the seed server. First, the seed server does not provide enhancement layer chunks to peers with low scores. Next, whenever a peer downloads a chunk from the seed server, peers' contribution score is decreased to force peer to minimize its chunk requests from the seed server.

We provide two approaches for reliable data collection. A P2P streaming solution is can choose either of the methods based on where their preference in the reliability/overhead trade-off. In both methods, the tracker server assigns random neighbors to an incoming peer and accepts statistics only from those peer that is has assigned. This is the fundamentally to avoid fake id attacks, in which a single client launches two applications, one of which reports false chunk exchange events to improve its statistics of the other.

The first method is the simple case, in which peers periodically report the total number of chunks that they have received from each neighbor. The period of the report can be in the order of minutes since the statistics are expected to be populated over a very long duration. Once a report is received, the tracker server first checks if the report comes from a node that it has assigned and if so increases the sharing statistics of the source neighbor. In this approach, there exists a risk of not reporting the results but this approach would not benefit the user and in the end hurts the whole community since a good peer is discourages.

However, even if this is the case, since the important criteria is the ranking (not the absolute value), as long as some of the peers report correctly, peers would still have a score that reflects their contribution to the network.

The second method addresses the risk of not reporting a downloaded chunk. When a peer joins the session for the first time, tracker assigns a seed number. Based on this seed number, chunk ID and layer ID information the peer generates a key and encrypts chunks before forwarding them. Using the same information, the tracker server can compute the key that is needed to decrypt the content. Then, once a peer receives a chunk from one of its neighbors, (or is about to receive the chunk to avoid delays), the peer asks for the key to decrypt the chunk. Then, the tracker responds with the key and the increases the upload statistics of source neighbor if the request comes from a peer that the tracker server has assigned. By this way, a peer cannot use a downloaded chunk unless it connects to the tracker server. This message exchange among the receiver peer and the tracker server introduces extra overhead. However, the overhead is still at acceptable when compared against the server-client modal.

2.3.3 Evaluation of the Proposed Incentive Method

The main feature of the AdaptP2P incentive mechanism is to make peers' past statistics to affect the duration of buffering delay as peers join to a new session. As an evaluation of this effect, we have assigned distinct scores to multiple peers and then introduced these peers into an ongoing session and forced them compete for buffering the content. Then, we have measured the buffering durations and compared them versus the case of using tit-for-tat in which the optimistic unchoking is performed totally randomly.

Test Scenario: We use three-layer scalable content of duration 10 minutes. The overall content bitrate is set to 90% of the average download rate of the peers from the seed server. We initiate the session initially with five peers and then wait for five minutes. After that, we introduce five more peers that have distinct scores to represent the rate of their past contribution. Each peer in the session has four upload slots assigned to the neighbors with best download rates (these slots are filled by peers are that are started in the first phase) and one more slot that is assigned to optimistic unchoking (set to one of the newcomer). The seed server provides only the base layer for these peers. Finally, we measure the duration for peer to buffer 5 seconds of content.

Results: Figure 2.7 presents the average time duration of the buffering period for the five peers, which have different history feedback score at the tracker server site as stated in the x-axis. The figure also presents the average buffering duration when tit-for-tat based incentive mechanism is adopted. The results shows that, if proposed unchoking algorithm is adopted then a good peer that has a good record has shorter buffering delay when joining a new session compared to Tit-for-tat algorithm.

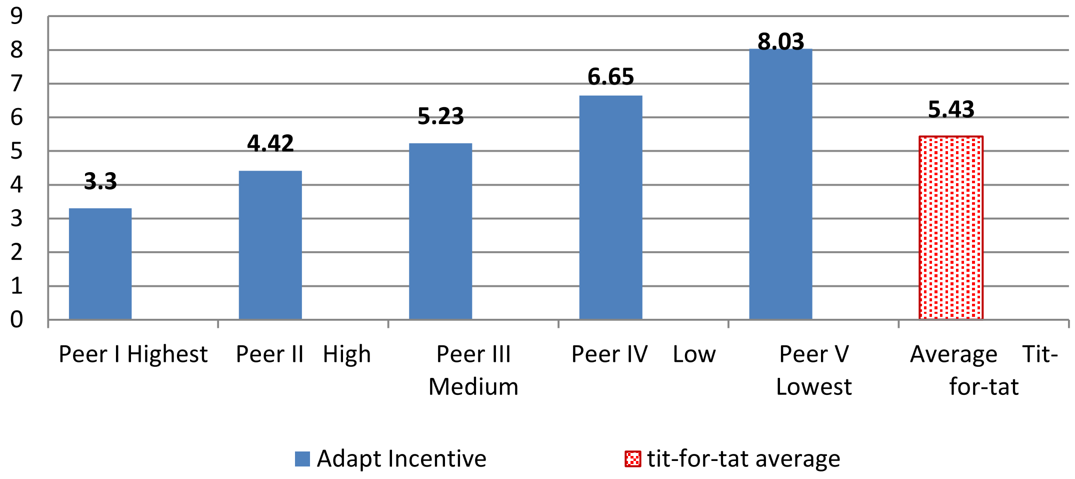


Figure 2.7: Buffering duration of proposed incentive method vs. Tit-For-Tat

2.4 Evaluation of Scalable Video Coding vs. Multiple Description Coding

One of the challenges with the P2P video streaming is developing efficient adaptation strategies to alleviate peer bandwidth fluctuations and peer exit events. There are two aspects of adaptation strategies that indeed interact with each other: i) P2P network-level adaptation schemes and ii) choice of video codec. In the previous sections, we have focused on the first aspect. In this section, we evaluate the performances of SVC, MDC and scalable multiple description coding (SMDC) using PeerSim simulation environment [61].

2.4.1 Evaluated Video Codecs

In a P2P network, fundamentally there are two types of complications. This first one is naturally the peer exit events, in which clients disconnect from the network without any notification. The second problem is the variations in the effective link capacity among peers.

Figure 2.8 presents likely preferred video coding choices under combination of these two events. However, combination of choice of video encoding and proper network adaptation can significantly affect end-to-end streaming performance over P2P networks. To this end, we evaluate the performances of SVC, MDC and SMDC video coding methods over a simulation environment that has similar features with the AdaptP2P approach.

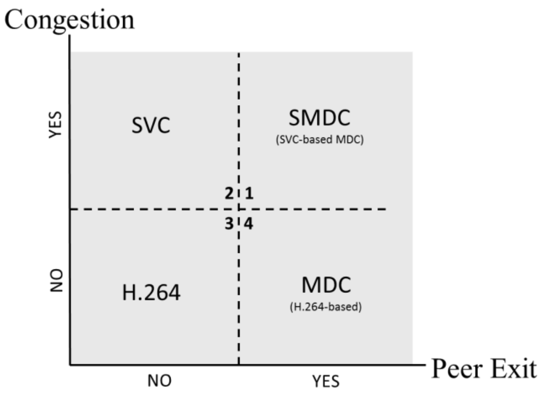


Figure 2.8: Possible video coding choices for scalability and stability

2.4.2 Implementation of the Evaluated Approaches

Scalable Video Coding: When the SVC approach is adopted, there are two types of chunks: i) base layer chunks, which are mandatory for video decoding and prioritized, ii) enhancement layer chunks, which are optional and increases the received video quality when available. When the simulation is run using SVC codec, the buffer duration is used as the key parameter for rate adaptation. Similar to AdaptP2P case, whenever the buffer duration is below a certain value, enhancement layer chunks are discarded. The bitrate of the SVC content is set to 2.2 Mbps (1.1 base + 1.1 enhc.) to simulate 10% overhead when the base content is at 2 Mbps and encoded using H.264/AVC codec [62].

Multiple Description Coding: In the case of using MDC, two independently decodable chunks are generated. When a peer is scheduling MDC chunks, it tries to receive different descriptions from different sources (if possible). By this way, if chunk retrieval fails due to a peer exit event, the receiver has more chance to remain unaffected from this event. Similar to SVC, when the buffer duration is low, peer discards one of the streams. However, unlike the SVC, the decision of choosing the description to discard is not predefined (since both descriptions have equal prioritization). Instead, when the buffer duration is low, each peer evaluated the number of chunks it has in its buffer for each description, and then discards the one with lower value. We have evaluated two different overhead for MDC case (since there are no standard description generation method) one at 10% and another at 50%, making 2.2 Mbps (1.1 + 1.1) and 3 Mbps (1.5 + 1.5) respectively.

Scalable Multi Description Coding: The last option is the combination of the first two methods. The chunk scheduling policy is as follows. When the buffer duration gets low, first the enhancement layers are discarded. If the available link capacity is still not adequate to receive the content at a steady pace, then one of the descriptions are discarded in a similar way with the MDC method. The bitrate of the SMDC is set to 2.2Mbps and 3Mbps. In both cases, the stream is split into four substreams, 2 for layering and 2 for multiple descriptions. One of the best feature of SMDC approach is that, the minimum require bitrate to support a single substream is lower when compared to MDC and SVC case. This gives leverage to SDMC in operating at low bitrates.

2.4.3 Simulation Setup and Scenarios

The P2P streaming experiments are performed over a partially connected mesh overlay. The overlay initially consists of 100 leecher peers, 4 seeder peers and 1 seed server. A peer can connect to up to 15 neighbors and if they have less than 5 connections, they actively seek new peers to connect. When a new peer joins a session, initial buffering delay is set to 8 chunks.

We evaluate P2P video streaming performance under different network conditions as summarized in Table 2.5. We consider three parameters: average upload rate with cross-traffic on the links, end-to-end propagation delay (Round-Trip-Time, *RTT*) and ungraceful peer exits.

Table 2.5: Test conditions

Test Parameter		**Values**
U_{AVG}	Average Upload (Mbps)	1.9, 3.0
RTT	Round-trip-time (ms)	50, 200
p_E	Peer Exit Ratio (%)	1, 50

2.4.4 Results and Comments

In Table 2.6, a result cell for SVC contains two numbers. The number at the bottom shows the percentage of received base layer chunks and the number at the top represents the percentage for cases in which both base and enhancements chunks are received. Cells for test results of MDC case also contain two numbers separated with "/" sign. The number on the left indicates the percentage when both descriptions are received, while the number on the right stands for the cases when only one description is received. Consequently, the cells regarding SMDC contain four numbers: horizontal split for layer information and vertical split for description information in the same manner with above cases.

Table 2.6: Test results for the evaluation of SVC, MDC and SMDC approaches

Avg. Upload Rate (U_{AVG})				**1.9 Mbps**		**3.0 Mbps**	
RTT				**50 ms**	**200 ms**	**50 ms**	**200 ms**
Peer Exit Ratio (p_E)				**1%**			
SVC	**Redundancy/Overhead**	**10%**	**B+E**	**25.1**	**5.6**	98.9	11.9
			B	**100**	**47.2**	100	98.9
MDC		**10%**	**-**	**14.4 / 85.2**	6.1 / 33.4	100 / 0	8.6 / 90.9
		50%	**-**	6.4 / 41.6	5.4 / 23.6	42 / 58	6.8 / 41.4
SMDC		**10%**	**B+E**	**20.1 / 40.2**	**4.6 / 2.6**	90.8 / 6.9	5.4 / 2.9
			B	**100 / 0**	**6.2 / 89.9**	100 / 0	93 / 6.8
		50%	**B+E**	6.2 / 4.8	4.3 / 0.7	41.6 / 42.2	5.1 / 2.6
			B	97.3 / 2.7	6 / 42.5	100 / 0	11.3 / 87.3
Peer Exit Ratio (p_E)				**50%**			
SVC	**Redundancy/Overhead**	**10%**	**B+E**	35.2	6.6	**99.9**	**21.8**
			B	100	66.2	**100**	**99.5**
MDC		**10%**	**-**	17.1 / 82.8	6.1 / 45.6	**97.5 / 2.5**	**15.4 / 84.2**
		50%	**-**	6.6 / 57.6	5.2 / 31.1	41.8 / 58.2	6.9 / 57.4
SMDC		**10%**	**B+E**	34.4 / 42.5	4.7 / 13.2	**98.1 / 1.4**	**7.1 / 15**
			B	99.9 / 0.1	6.2 / 92.7	**100 / 0**	**96.6 / 3.3**
		50%	**B+E**	6.2 / 18.6	4.4 / 2.1	49.6 / 38.6	5.1 / 2.8
			B	99.6 / 0.4	6.1 / 62.7	95.3 / 4.7	25.3 / 74.3

In the following, we present two interpretations of the results:

Interpretation 1: SVC provides longer uninterrupted video playback duration and better quality when compared to 10% redundant MDC (at equal bitrate) where U_{AVG} is 1.9 Mbps, which is not adequate to stream video at the maximum quality. With MDC, peers are able to decode 99.6% of the video (14.4% two descriptions, 85.2% single description) when p_E is 1% and *RTT* is 50ms. With SVC, peers are able to decode the whole video sequence (25.1% of the time it is full quality, 74.9% of time at base layer quality) at higher quality. This is due to difference in the availability of chunks in two options. With MDC, some peers may drop description 1 while others may drop description 2 while performing rate adaptation. Consequently, availability of chunks from a particular description decreases. In the SVC, all peers discard the same stream so the availability of the base layer chunks is not affected. The performance gap with SVC widens with higher redundancy in MDC.

When U_{AVG} is increased to 3.0 Mbps, while keeping the other parameters fixed, MDC yields a slightly higher chunk delivery ratio and covers longer video duration when compared to SVC. This minor difference is due to the scheduling policy used with SVC, which favors base layer chunks over enhancement layer chunks, leading to a more pessimistic scheduling. In the MDC, the descriptions are equally important and without favoring a particular bitstream, MDC can achieve slightly higher delivery rates. However, even if the delivery rate of MDC is higher, SVC still outperforms MDC when we consider the PSNR values with up to 0.4dB difference when the received streams are decoded. The observation stays the same for different values of *RTT* and p_E and SVC delivers higher PSNR value against 10% redundant MDC.

When we compare SVC codec against 10% redundant SMDC under the same initial conditions (U_{AVG} is 1.9 Mbps, *RTT* is 50 ms and p_E is 1%), we observe that SDMC scheme is able to deliver a larger portion of the chunks. In SMDC, all base layer chunks of both descriptions are received. In addition, approximately 60% of the video is decoded beyond base layer quality (20.1% two description, 40.2% single description). In SVC, all base layer chunks and only 25.1% of the enhancement layer chunks are delivered. In SMDC, each description is further divided into two streams for scalability, resulting in lower bitrate for a single stream. This allows easier dissemination and increasing the quality-of-service (QoS) performance. However, SVC approach again outperforms the SMDC case with respect to achieved PSNR values: 34.19dB with SVC and 33.54dB with 10% redundant SMDC. For other cases, the gap regarding the achieved average PSNR value increases.

Conclusion 1: SVC outperforms both MDC and SMDC in the achieved PSNR value as long as the available bitrate is adequate for transmitting the base layer chunks even if the chunk delivery ratio of SVC is less than the MDC based approaches. Higher encoding efficiency compensates for the missing chunks.

Interpretation 2: Our initial expectation was that MDC/SMDC based approaches would outperform SVC under high peer exit ratios; however, results indicate that the stability of the system is not affected significantly based on the encoding approach. We observe that when the system is mesh-based and receiver-driven (pull-based), where a single peer has multiple source candidates, peers are able to recover peer exit events. This is especially in the case if peers have sufficiently long buffer duration that allows peers to detect if source peer exits and to switch to a new source for a chunk whose transmission is interrupted due to an ungraceful peer exit event.

Conclusion 2: Even if SVC is not specifically designed for providing robustness against path/peer failures, a receiver-driven P2P solution can still utilize SVC and achieve robust video transmission.

2.5 Summary

In this chapter, we propose three fundamental modifications on the BitTorrent protocol to achieve higher performance in P2P video streaming. AdaptP2P supports variable length chunks that have the boundaries aligned with the multimedia data segments. The test results show that variable length chunks outperforms fixed sized chunk in the case of data losses. Moreover, variable length chunks have fixed duration, allowing the P2P engine to calculate the buffer duration precisely without parsing the bit stream.

We propose a novel chunk scheduling policy that has three important features: i) AdaptP2P distinguishes seed server (content originator) from the other peers including those that become seeder over time. This provides peers the opportunity to avoid requesting chunks from the seed server whenever possible. ii) AdaptP2P has a *variable length scheduling window* to govern chunk scheduling policy. The window aims to maximize randomness in chunk picking to increase the possibility having different chunks in different peers. iii) The scheduling window enables rate adaptive video streaming and introduce graceful quality degradation in case of bandwidth scarcity. The results of P2P video streaming tests in PlanetLab environment shows that AdaptP2P introduces less bandwidth load on seed servers while improving the average video quality.

We have introduced a novel incentive mechanism, which is based on recording the rate that a peer utilizes its upload capacity in a persistent manner. By this way, when a peer with good records joins to a new session, it experiences shorter buffering delays because its neighbors are more likely to pick such a `friendly` peer for optimistically unchoking.

Finally, we have evaluated SVC against MDC and SMDC methods. The results suggest that in mesh-pull based P2P solutions, using multi-layer SVC provides better RD performance when compared against multiple description coding and achieves better scores under the same network conditions.

Chapter 3

QUALITY OF EXPERIENCE AWARE 3D VIDEO STREAMING OVER P2P NETWORKS

3D movies have already made a big impact over the industry. At the 3D theaters, consumers enjoy an immersive experience and while producers enjoy pirate-protection of 3D displays, which hinders people from recording the content with a simple camera. Motivated by this fact, the producers was economically encouraged to shoot a movie in 3D and soon after there has been a rapid increase in their number. With the increasing interest in 3D and the increase in the availability of 3D-ready displays, 3D entertainment is very likely to be ported to our homes and mobile platforms. Until now, this has been the way that display technologies spread in the market; first they appear in movie theaters and then they migrate to homes. This is the path that both high-definition (HD) TV and color TV have followed in the past. However, there are still challenges that should be addressed especially for the delivery of multi-view video content.

The stereoscopic 3D video broadcast is available for home users. It operates over legacy infrastructure in frame compatible format, which requires only 3D-ready display. Unfortunately, the perceived visual quality (although 3D) can be inferior when compared to high-definition 2D video because subsampled stereo views has decreased resolution. Considering that switching to full resolution 3D requires infrastructural modifications, it is a very big challenge to further increase the quality in stereoscopic 3D broadcast.

Similar problem exists for the multi-view video broadcast over fix bitrate channels. The main problem is the varying number of views for each user depending on their MVV display features. In such cases, it is very difficult to establish a homogeneous broadcast format that will satisfy each and every customer without providing the content at the highest quality, which would require huge amount of bandwidth. Therefore, a broadcast mechanism over legacy fix bitrate channels do not exists.

3D content is to be provided to home users however currently available options either provide the content at low quality (as in the case of frame compatible format with stereoscopic 3D) or does not support at all (as in the case of multi-view video). As a result, we consider the IP network as one of the best (if not the best) alternative to currently available channels for the delivery of 3D video.

We provide three reasons that make IP a good candidate for video delivery. First, by its packet switching nature, IP provides variable link capacity that can be increased based on users preferences. Therefore IP can accommodate the multi-view-video delivery systems that would need variable capacity based on users' requirements. Second, the access to IP is now easier than ever. Especially after the introduction of 3G technology, now people can have quite high connection capacity over mobile devices. Moreover, these mobile devices can even distribute their own connection to other devices in their neighborhood, making IP accessible virtually from everywhere. Moreover, with the next generation the capacity of such links reaches very high values that can support high bitrate requirement of multi-view video. Finally, two-way communication in IP network allows interactive services. In such an environment, a user can recommend some contents to his/her friends in a social network, making a huge impact on content dissemination. Moreover, content providers may collect some information then enable personalized advertisements, which are an added value to the service. Therefore, when coupled with two-way communication, the service is open to innovations and may provide increased benefits for the providers.

In this chapter, we consider the challenges of 3D video streaming over IP networks. We believe that at the core of the problem there are two key challenges of regarding the delivery of 3D video over the IP network. The first challenge is the lack of quality-of-experience aware adaptation strategies for both stereoscopic 3D and multi-view video. As we have stated in Chapter II, rate adaptation is a critical feature for providing reliable video service over best-effort IP network. Second, the high bitrate requirement, especially in the case of multi-view video poses a serious challenge since it may become difficult to serve increasing number of clients if the required rate of data transmission reaches critical values. In the rest of this chapter, we address these two issues. In Section 3.1, we provide the state of the art in 3D video formats and 3D video coding. Section 3.2 and Section 3.3 addresses QoE Aware rate adaptation mechanism for stereoscopic 3D and multi-view video respectfully. These sections provide a systematic approach to device an adaptation mechanism, which we utilize in the proposed 3D video streaming solution. Following these discussions, we provide a method to utilize these adaptation methods in the proposed P2P video streaming approach in Section 3.4. Finally, we combine these approaches and provide a summary of the proposed 3D video streaming solution in Section 3.5.

3.1 State of the Art 3D Video Formats

Stereoscopic and multi-view videos are available as current 3D video formats as depicted in Figure 3.1. Common stereo video formats are frame-compatible and full-resolution (sequential) formats. There are also depth-based representations, which are often preferred for efficient transmission of multi-view video as the number of views increases.

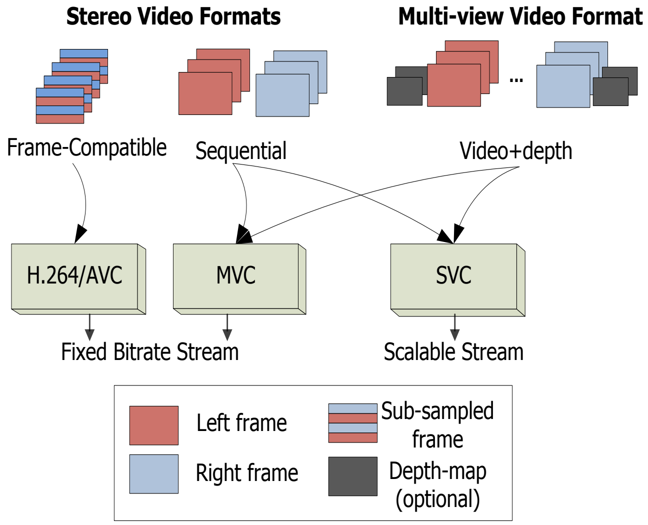


Figure 3.1: 3D video formats and coding options

Frame compatible stereo video formats have been developed to provide 3DTV services over the existing digital TV broadcast infrastructures [63]. They employ pixel subsampling in order to keep the frame size and rate the same with that of 2D video. Common subsampling patterns include side-by-side, top-and-bottom, line interleaved, and checkerboard. Side-by-side format applies horizontal subsampling to the left and right views, reducing horizontal resolution by 50%. The subsampled frames are then put together side-by-side. Likewise, top-and-bottom format vertically subsamples the left and right views, and stitch them over-under. In the line interleaved format, the left and right views are again subsampled vertically, but put together in an interleaved fashion. Checkerboard format subsamples left and right views in an offset grid pattern and multiplexes them into a single frame in a checkerboard layout. Among these formats, side-by-side and top-and-bottom are selected as mandatory for broadcast by the latest HDMI specification 1.4a [64].

Frame packing, which is the mandatory format for movie and game content in the HDMI specification version 1.4a, stores frames of left and right views **sequentially**, without any change in resolution. This format, which supports full HD stereo video, requires, in the worst case, twice as much bandwidth of monocular video. The extra bandwidth requirement may be kept around 50% if we use the Multi-view Video Coding (MVC) standard, which is selected by the Blu-ray Disc Association as the coding format for 3D video.

An alternative stereo video format is **view-plus-depth**, where a single view and its associated depth map are transmitted to render a stereo pair at the decoder side. It was proposed by the European project Advanced Three-Dimensional Television System Technologies (ATTEST) [44] to develop a backwards compatible 3DTV service using a layered bit stream with MPEG-2 coded monocular view in the base layer and encoded depth information in a supplementary layer. MPEG has specified a container format for view-plus-depth data in “ISO/IEC 23002-3 Representation of Auxiliary Video and Supplemental Information” also called MPEG-C Part3 [45], [46]. It has later been proposed to extend this format to multi-view-video-plus-depth (MVD), where *N* views and *N* depth maps are used to generate *M* views at the decoder, with $N \leq M$ [65].

Each frame of the depth map conveys the distance of corresponding video pixels from the camera. The depth values are scaled and represented with 8 bits, where higher values represent points that are closer to the camera. Therefore, the depth map can be regarded as a gray scale video, which can be compressed very efficiently using state of the art codecs, due to its smooth and less structured nature. Typically, a single depth map requires 15%-20% of the bit rate necessary to encode the original video [66]. Furthermore, an MVC codec can be utilized to exploit the inter-view correlations between depth maps for the MVD representation [67].

The depth map information needs to be accurately captured/computed, encoded, and transmitted in order to render intermediate views accurately using the received reference view and depth map. A difficulty with the view-plus-depth format is generation of accurate depth maps. Although there are cameras that can generate disparity maps, they typically offer limited performance. Algorithms for depth and disparity estimation have been studied extensively in the computer vision literature, which either utilize disparity estimation from rectified images or perform color-based image segmentation [67]. Another difficulty with the view-plus-depth format is the appearance of some regions in the rendered views, which are occluded in the available views. These *disocclusion* regions may be concealed by smoothing the original depth map data to avoid appearance of holes, as in the ATTEST project [44]. Also, it is possible to use multiple view-plus-depth data to prevent *disocclusions* [68].

An extension of the view-plus-depth, which allows better modeling of occlusions, is the layered depth video (LDV). LDV provides multiple depth values for each pixel in a video frame. The number of depth values depends on the number of surfaces in the line of sight for that particular pixel to allow for better rendering of intermediate (synthetic) views [69].

3.2 Quality of Experience Aware Rate Adaptation for Stereoscopic 3D Video

3.2.1 Evaluation of Perceived Quality Using Asymmetry in Stereoscopic 3D

It has been observed that the human visual system (HVS) can perceive high frequencies in 3D stereoscopic video, even if one of the views contains only the low frequency components. This finding suggests that the best perceived quality may be achieved by asymmetric coding where the right and left views are coded at unequal quality levels (e.g., different PSNR values). However, what is the best level of asymmetry and more importantly how the asymmetry should be achieved has been open questions. In this section, we provide the subjective tests that are conducted to evaluate different methods of rate scaling over using different display systems. The results indicate that if the one of the views is encoded at sufficiently high quality and the second view is encoded at a lower quality but above a certain PSNR threshold then the perceived degradation in video quality is almost unnoticeable. Subjective tests also indicate that below this just-noticeable asymmetry threshold, where subtle artifacts start to appear, symmetric coding performs better than asymmetric coding in terms of perceived 3D video quality. Therefore, we show that the choice between asymmetric vs. symmetric coding depends on PSNR; hence, the available total bitrate.

3.2.1.1 State of the art Asymmetric Stereo Video Coding

Perkins et al. used low-pass filtering, which effectively removes high frequency information to create asymmetry among views. The report shows that perceived quality is not significantly affected [70]. Tam et al. [71] and Stelmach et al. [72] are pioneers in asymmetric coding and they argue that depth perception is not affected when asymmetric coding is applied. This finding is supported by Seuntiëns et al. [73], [74] Meesters and Meegan argue that *blockiness* artifact that appears at low bitrates is more disturbing than blurring for the overall stereoscopic perception [75], [76]. Aksay et al. compare temporal versus spatial scaling and conclude that the latter one is superior.

Implementing asymmetry by PSNR reduction is straightforward, and does not require any modification to the current MVC codec. Since the encoding quality of a view depends on the quantization parameter, using different quantization parameters for the left and right views results in PSNR asymmetry. Fehn et al. [77] implement asymmetric coding by spatial scaling within the MVC codec but do not compare it with quality scaling. Stelmach has used MPEG-2 instead ofH.264 for coding however according to Kalva et al. [78] MPEG-2 may cause different artifacts than H.264 on coded video, creating different effects on the perceived quality. Besides, none of these articles consider the influence of 3D display technology on perceived quality of stereoscopic video.

3.2.1.2 Determining the Level of Asymmetry

There should be limitation of the amount of asymmetry can be applied such that the quality degradation in one view is not noticeable. Moreover, this threshold value can be found in terms of PSNR the higher quality view and visual artifacts are perceptually unnoticeable. In order to test this proposition and find the threshold in terms of a PSNR threshold, we have conducted interactive subjective tests using four stereo videos and two different display systems: i) full- resolution polarized projector which is similar to that used in 3D theaters ii) auto-stereoscopic display equipped with parallax barrier, which effectively halves spatial resolution During the test, assessors reduce the quality (PSNR) of one of the views from 40dB down to 25dB. The assessors always compare the current quality level against the 40dB content. The subjective test is finalized when the assessors detects that the quality is inferior when compared to 40dB content. At this point, the PSNR value of the previous content (one higher in quality) is accepted as the threshold value for the particular content and the display. Table 3.1 shows the average threshold values (over all subjects) for two different displays. The value represents the last point that the assessors have not notices quality degradation.

Table 3.1: PSNR thresholds for level of asymmetry in asymmetric coding

Content			**Threshold PSNR (dB)**	
Name	**Resolution**	**Type**	**Polarized Projector**	**Parallax Barrier**
Adile	640x480	Rendered 3D	33	32
Train	704x576	Fixed Camera	33	32
Flower	704x448	Moving Camera	33	31
Iceberg	640x384	2D-3D Conversion	33	32

3.2.1.3 Comparison of Asymmetric Coding vs. Symmetric Coding

Until a certain value (~32dB), the reduction of quality in one view is not noticeable, which states that there exists a threshold. Based on this argument, we have suggested that the rate reduction in one view using asymmetric coding can be used to enhance the quality of the other view, to deliver higher perceived quality when compared to a content, which both views have the same quality. In order to test this proposition, we have conducted the following subjective tests using the DSCQS method [79]. We have used three different video with the RD performances presented in Table 3.2. Finally, in order to validate the first conclusion, we have repeated the test for two cases: i) When both streams are above the threshold value ii) when one stream is below the threshold value.

Table 3.2: Test contents for the evaluation of asymmetric coding
(Above Threshold)

Stream	PSNR (dB)			Bitrate (Kbps)		
	Adile	Flower	Train	Adile	Flower	Train
Asymmetric Left	37.5	37.64	37.24	719	1880	2259
Asymmetric Right	29.5	29.53	29.42	204	280	461
Symmetric Left	34.76	35.54	35.16	496	1147	1426
Symmetric Right	34.79	35.79	35.15	480	1100	1370

Table 3.3: Test contents for the evaluation of asymmetric coding
(Below Threshold)

Stream	PSNR (dB)			Bitrate (Kbps)		
	Adile	Flower	Train	Adile	Flower	Train
Asymmetric Left	38.258	33.861	32.276	794	801	794
Asymmetric Right	29.609	28.485	27.251	205	221	262
Symmetric Left	34.755	31.596	29.973	482	490	499
Symmetric Right	34.786	32.426	30.079	484	524	481

3.2.1.4 Results of the Subjective Tests and Interpretations

The subjective test results are presented in Table 3.4 and Table 3.5. The values represent the test score results in DSCQS method. In this methodology, a lower score represents better quality. Based on these values, we can make the following interpretations.

- At high bitrates (above a high PSNR threshold of about 32 dB for the auxiliary view), asymmetric coding with SNR scaling achieves the best perceived quality.
- At very low bitrates (coding the auxiliary view below a low threshold value of about 28dB), asymmetric coding with spatial scaling achieves the best perceived quality.
- Between these two threshold values, symmetric coding is preferred over asymmetric coding.
- The high and low threshold values show some variation depending on the display technology; e.g., the high threshold value is about 31 dB for a parallax barrier display and 33 dB for a full resolution projection display.

Table 3.4: Normalized subjective test scores (Above Threshold)

	Parallax			**Projector**		
	Adile	**Flower**	**Train**	**Adile**	**Flower**	**Train**
Asymmetric	15 ± 4	2.6 ± 1	6.6 ± 6	4.4 ± 3	7.1 ± 5	5.4 ± 3
Symmetric	25 ± 14	7.8 ± 3	47 ± 3	21 ± 10	12.6 ± 7	5.4 ± 6

Table 3.5: Normalized subjective test scores (Below Threshold)

	Parallax			**Projector**		
	Adile	**Flower**	**Train**	**Adile**	**Flower**	**Train**
Asymmetric	34.4 ± 3	26.4 ± 3	41.4 ± 8	38.4 ± 3	33.1 ± 3	58.1 ± 4
Symmetric	8 ± 3	18.0 ± 7	32 ± 6	21.6 ± 6	25.3 ± 4	43.6 ± 7

3.2.2 QoE Aware Scalable Video Coding and Performance Evaluations

One of the benefits of simulcast coding is to obtain independently decodable streams, without the complexities in the MVC case, simplifying the adaptation method as described in [80], [81]. In cases where inter-view redundancy is not desired (it complicates chunk scheduling procedure) difficult to exploit due to factors such as camera orientation, calibration problems, and lighting differences, simulcast coding may achieve compression efficiency that is comparable with MVC encoding.

Using SVC, There are two encoding options for achieving scalable asymmetric stereoscopic video bit streams when simulcast coding is adopted. In first case, both views are encoded using SVC. In the second case, one with is encoded using scalable SVC codec whereas the other view is encoded high efficient H.264/AVC.

Figure 3.2-a presents a possible layer configuration in terms of quality when both views are encoded using SVC. In this scheme, higher adaptation capability (range between the minimum bandwidth and the maximum bandwidth is higher) is achieved but at the cost of compression efficiency. Hence, it is possible reduce the bitrate lower values, but the maximum quality is also reduced due to scalability overhead. Although the figure shows a symmetric quality among views, asymmetry is obtained when rate adaptation is performed. Hence, one of the views is extracted at the highest possible rate, while the other is kept at the base layer quality or higher, if the bit budget allows. The loss in compression efficiency is due to scalable coding overhead of SVC, which is now applied to both views.

Figure 3.2-b depicts the configuration of the bit streams when one of the views is encoded using H.264/AVC. Comparable adaptation capability is still possible if the gain in bitrate for using H.264/AVC, which offers the same quality at a lower bitrate, is used to augment the enhancement layer of the scalable bit stream. In this scheme, HVS is exploited in either case where the video is transmitted at the maximum or the minimum quality. When the available bandwidth is more than the maximum rate of the video, the quality of the scalable bit stream dominates the perceived quality. On the other hand, when the bandwidth is scarce, the non-scalable bit steam becomes the high quality pair of the asymmetry. In [82], we have tested the adaptation capability of this scheme.

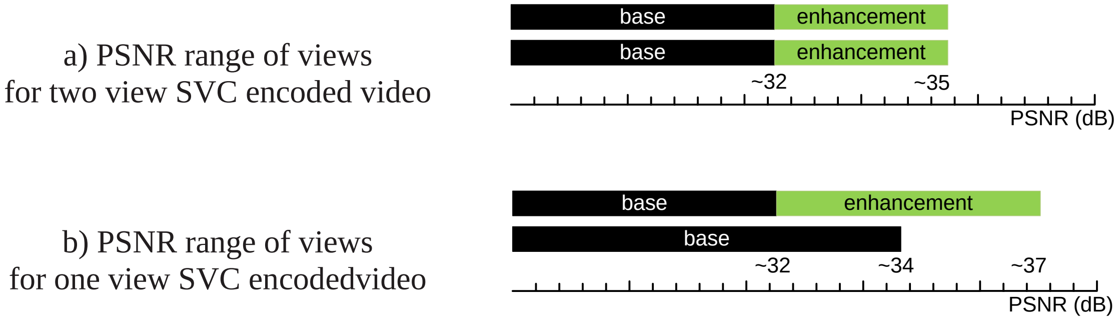


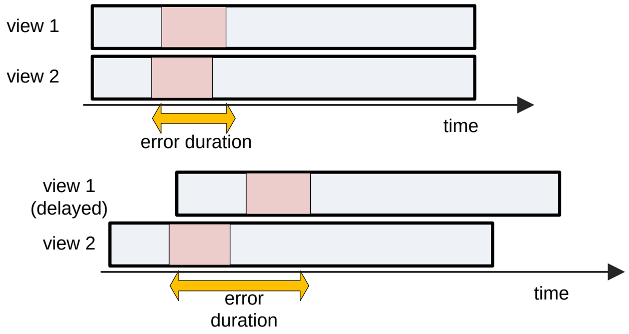
Figure 3.2: Scalable coding options for stereoscopic case.

3.2.3 Effect of Packet Losses on the Perceived Video Quality

Packet losses are inevitable in IP networks. Since IP operates in the best effort sense, UDP packets can simply be dropped. In the case of TCP, data can still be considered as lost if it is delayed until a time that is after the presentation time and became completely useless. So far, researches have mostly focused on perception of 3D video without packet losses, investigating different encoding options such as asymmetric rate allocation. Most of the studies reveal that human visual system (HVS) tends to neglect loss of high frequency components in one of the views [72], [83], [84]. However, the effect of packet losses/delays over visual perception of 3D video and whether HVS can compensate the artifacts generated by concealment methods has not been studied. Since lost data significantly affects the perception error concealment at the receiver side is an important issue for transmitting MVV over IP.

Evaluated Approaches: In this study, we have employed two different error concealment methods. In the first method slice level error concealment algorithm based on slice repetition is employed. When a slice of a frame is lost the corresponding region's residue is replaced from another frame that is closest in picture order count metric. This approach performs well for slices that has limited or no motion, but introduces distortions when there is a significant motion within the lost slice's region. The alternative method is frame repetition, in which where ever some of the slices are lost, a frame that is decoded without any loss is replaces the current frame. This method avoids any structural mismatches between lost slice and remaining residue.

Also, the error duration is a crucial factor that affects the perceived quality. Normally, due to bursty nature of packet losses over the IP [85], the errors that occur for a time interval, affects both views. But it is possible to delay one of views or interleave its data in a way that different sections of the data are transmitted at a time instant. In this case longer interval of the video becomes erroneous but the errors are more likely to be separated over different intervals. We have investigated if HVS favors one of these cases. These cases have been depicted in Figure 3.3.


a) Overlapping error interval

b) Separated error interval

Figure 3.3: Loss intervals in stereoscopic 3D video

Table 3.6: Test video sequences

Name	Resolution	Information
Adile	640x480	Computer generated
Flower	704x448	Moving camera
Train	704x576	Fixed camera

Test Content Preparation: We have encoded the contents enlisted in Table 3.6 using H.264/AVC with GOP size 16 frames. Then, the encoded bit streams are simulated to be sent over channels with %3 and %5 loss rates generated by trace files provided in [86]. In the first transmission methods, both streams are forwarded as is, whereas in the second method, the GOPs are interleaved to decrease the chance of having error in the same time interval for both views. Then received streams are decoded using either frame or slice based error concealment. In the end, we have total of 12 test sequences (3 contents, 2 types of transmission, 2 types of error concealment methods.) and 3 more sequences without any decoding errors.

Subjective Tests and Results: Subjective tests are performed to evaluate the perception of above mentioned conditions in 3D. The testing methodology is based on the Double-Stimulus Continuous Quality-Scale (DSCQS) method. We have tested with 7 male and 3 female assessors and 7 of those were experts in 3D video coding area.

The results reveal that even though the error interval has been extended, people significantly prefer the case in which the loss interval is longer however the effect is distributed to two views. This is probably due to the features of HVS, as it may conceal the effect of packet loss up to a certain degree. Moreover, people have preferred slice based error concealment method even in sequences with high motion (such sequences generate more disturbing mismatches in slice based error concealment methods compared to frame repetition.) More results can be found in [87].

3.3 Quality of Experience Aware Rate Adaptation for Multi-view Video

Stereoscopic 3D video using special 3D glasses has already made an impact on the multimedia industry given the recent surge in availability of 3D movies and 3DTV broadcasts over DVB. We are now progressing towards the next phase in 3D media services in multi-view-video (MVV) format that enables 3D viewing from multiple angles; allowing viewers to see different views by tilting their head without using special glasses.

One of the biggest challenges for MVV content delivery is to support wide range of 3D displays that require different number of views (and possibly different bit rates); making it difficult to transmit over fixed bit-rate channels, such as DVB. Perhaps, transmission of MVV over IP is the most flexible solution for 3D media delivery because IP can provide different transmission capacities to different users. It is also possible to use IP as a supplement to DVB broadcast [1]. In such a scenario, the minimum number of views that is needed for 3D support (stereo views) are transmitted over DVB whereas the remaining views are transmitted over IP depending on the number of views that the client requires.

One of the major challenges must be addressed in order to establish successful multi-view video streaming solution over IP is to develop a rate adaptation mechanism. The Internet operates in the best effort sense; hence, it is possible to experience varying amount of delays and link capacity between two nodes. With a good adaptation mechanism, it may be possible to minimize the effect of rate scaling. In this section, we investigate the possible methods to device the adaptation method for multi-view video.

3.3.1 Scaling Methods for Multi-view Video

The goal of each scaling method is to decrease overall bitrate of transmitted MVV to match the available network rate. This can be done either by decreasing quality (quality scaling) in one or more scalability dimensions (SNR, spatial or temporal) or by discarding view(s) at the cost of transmitting depth-maps (view scaling).

Quality scaling can be applied equally on each view (*symmetric scaling*) or unequally among different views (*asymmetric scaling*). In *symmetric scaling*, all views are affected thus the decrease in perceived video quality is inevitable unless the video is coded at high quality unnecessarily. *Asymmetric scaling* is a more advanced approach as we have discussed in the previous section.

According to the studies on stereoscopic 3D video, asymmetric scaling may yield higher perceived quality compared to symmetric coding at the same bitrate [83]. In the case of multi-view video, asymmetric scaling corresponds to scaling adjacent views at different quality levels. Since, even in a multi-view display, viewers see only two views at a time, it is possible to exploit the discussions about stereoscopic 3D video while devising a method for rate adaptation for multi-view video.

View scaling is an alternative method in which a subgroup of views is transmitted with associated depth maps. Whenever view scaling is applied, some portion of the bit stream that corresponds to particular view(s) is discarded from the transmission procedure. Then, the missing view(s) are interpolated at the client side using depth-image-based-rendering (DIBR) [87]. Depth maps are single channel images with less high frequency components and can be coded with higher efficiency compared to color images. Therefore, inserting multiple depth-maps and then discarding a single color map is still favorable in terms of decreasing the overall bitrate. However, the quality of the associated colored image and the depth map must be high; otherwise artifacts may occur in the interpolated views.

3.3.2 Subjective Tests Methodology

Subjective tests are to evaluate the performance of scaling methods in terms of delivered QoE under different network conditions. In each test, a different target bitrate is set to simulate a certain network condition and test video sequences are encoded to match the target bitrate using one of the scaling methods listed in the next subsection. Then assessors have evaluated each scaling method by comparing the scaled sequences against original sequence based on the Double-Stimulus Continuous Quality-Scale (DSCQS) standard [79]. 12 male and 4 female assessors have attended the tests and 7 of them were experts in 3D video coding area. The tests are performed using 5-view 3D display at 1920x1200 screen resolution that is equipped with lenticular sheet technology.

Preparation of Reference Test Streams: In each subjective test, test sequences (Dog1280x960, Lovebird1280x960, and Pantomime1024x768) are encoded using scaling methods explained below to match the target bitrate. In some tests, some methods cannot be applied if they fail to match target bitrate at an acceptable quality level or fail the asymmetric coding criterion. (In asymmetric coding approach, it is stated that the low quality pair should be above ~32dB. [89]) Table 3.7presents the bitrate limitations and Figure 3.4 depicts their RD performances.

Table 3.7: Bitrate limitations of tests scenarios for each sequence (kbps)

Test No	**Pantomime**	**Dog**	**Lovebird**
1	~ 4600	~ 3800	~ 2500
2	~ 3500	~ 2500	~ 1900
3	~ 2900	~ 2100	~ 1500
4	~ 2300	~ 1700	~ 1300

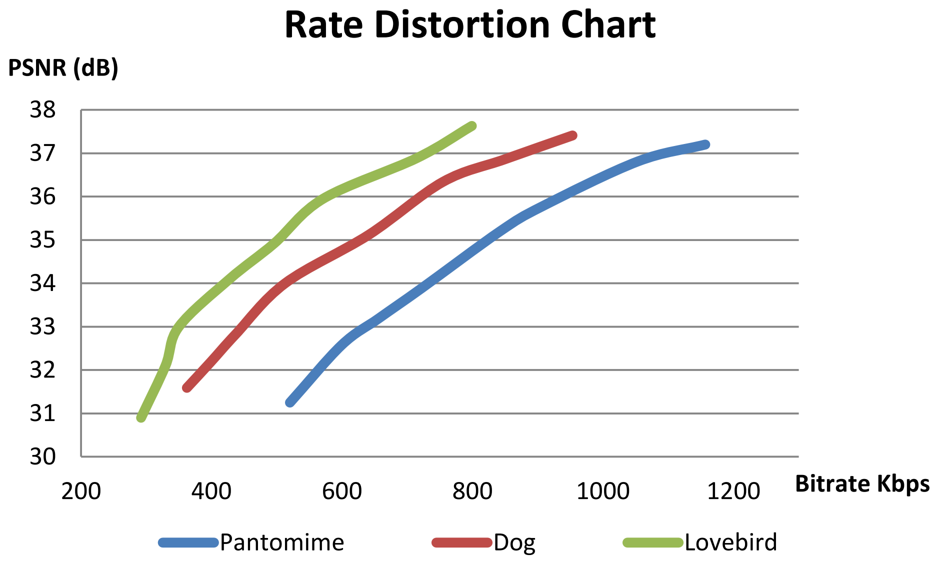


Figure 3.4: RD performance of test sequences

Evaluated Approaches: In the following, we provide the methods that are used to scale the overall bitrate of the content to match the test limitations.

Method 1: In symmetric SNR scaling, all views are encoded at a quantization parameter value higher than the original stream. Increasing the quantization parameter causes courser quantization in the DCT stage and decreases both the bitrate and the quality.

*Method 2:*Symmetric spatial scaling is performed by decimating each view to one fourth of its original resolution and then coding at target bitrate. This approach is especially effective when the bitrate is originally at high definition. Otherwise (when the original content has low resolution), decimation causes more significant artifacts.

*Method 3:*In asymmetric SNR scaling, the intermediate views (view 2 and 4)are encoded at a low quality threshold whereas remaining are first coded at high quality such that the overall bitrate is matched to the test limitations. The drawback of this method is that, it yields marginal bitrate reduction thus it could be applied only in Test 1.

*Method 4:*In asymmetric spatial scaling, view 1, 3 and 5 are first downscaled and then encoded using a QP that matches the overall bitrate to the test limitations.

Method 5: In this method, some of the views are not transmitted at all and estimated by using (DIBR) techniques [87]. In method 5, only the intermediate two views (2-4) are discarded. Then, view 2 is artificially generated view 1 and view 3. Similarly, view 4 is interpolated at the client side using view 3 and view 5.

Method 6: In this method, we preserve only the edge views (1 and 5) and discarded the other views in between. The method 4 uses three reference views to interpolate two missing views. Whereas in this method, we use two reference views to interpolate three missing views. Naturally, the quality of the reference views is higher in method 6, because the bit-budget is split among only two views, not three.

Table 3.8 provides the list of adopted methods for each test case. Moreover, the table provides the PSNR values of the 5 streams (if not discarded) that is obtained using the adopted methods.

3.3.3 Subjective Test Results

Figure 3.5presents the subjective test results for all test sequences with corresponding adaptation method number. Our observations for each test are as follows:

In Test 1, the bitrate budget is high enough to apply all scaling methods. However, since the bitrate is very high, almost all methods scores close to the reference view. Only symmetric SNR scaling introduces barely noticeable artifacts (observed by trained assessors). Interestingly, Method 3 has lower PSNR average than Method 1 but HVS can compensate the difference in quality in stereoscopic 3D [71], [84]. So we have experience a similar result for the MVV case.

InTest2, Method 1 has a lower score; indicating that the artifacts become more visible. The reason clearly can be seen in Table 3.7 for Pantomime sequence. All views are coded at around 33dB and this amount is at a visually perceivable low quality.

For the remaining last two tests, transmitting color images (whether they are coded symmetric or asymmetric) for each views yield perceivably lower quality. Again, this finding is coherent with the previous studies in which authors state that beyond a certain PSNR threshold value, asymmetry performs poorly [83]. When we compare frame based methods (asymmetric or symmetric) against DIBR based methods, we observe that DIBR methods are more favorable at low bit rates if implemented correctly. The results indicate that using less number of reference frames at higher quality is better than using more reference frames at lower quality for DIBR techniques. Then, DIBR based methods either provide the best visual quality or at least they match the score of frame based methods. This observation is evident from the test scores of Method 5 and 6.

Yet another observation is that, in spite of the high PSNR value, blurriness has been observed in interpolated low-resolution videos in the subjective tests. So it is preferable to pick view plus depth solution if the budget is very limited.

Table 3.8: Adopted methods for each test conditions

a) Test parameters for Pantomime

	View Number				
	1	**2**	**3**	**4**	**5**
Reference: ~5700kbps					
	37.6	37.6	37.6	37.6	37.6
Method	**Test 1:** ~4600kbps				
1	35.7	35.7	35.7	35.7	35.7
3	37.6	32.6	37.6	32.6	37.6
4	37.6	35.6	37.6	35.6	37.6
5	37.6		37.6		37.6
Method	**Test 2:** ~3500kbps				
1	33.2	33.2	33.2	33.2	33.2
2	35.6	35.6	35.6	35.6	35.6
4	35.6	34.9	35.6	34.9	35.6
5	35.7		35.7		35.7
Method	**Test 3:** ~2900kbps				
1	32.6	32.6	32.6	32.6	32.6
2	35.6	35.6	35.6	35.6	35.6
4	35.6	32.6	35.6	32.6	35.6
5	33.9		33.9		33.9
6	37.6				37.6
Method	**Test 4:** ~2300kbps				
2	34.2	34.2	34.2	34.2	34.2
5	31.2		31.2		31.2
6	34.8				34.8

b) Test Methods

Test Methods

No	Description
1	Symmetric SNR
2	Symmetric Spatial
3	Asymmetric SNR
4	Asymmetric Spatial
5	DIBR using 3 ref. view
6	DIBR using 2 ref. view

SNR scaled view
░

Spatial scaled view
▓

View depth
▒

Discarded view
□

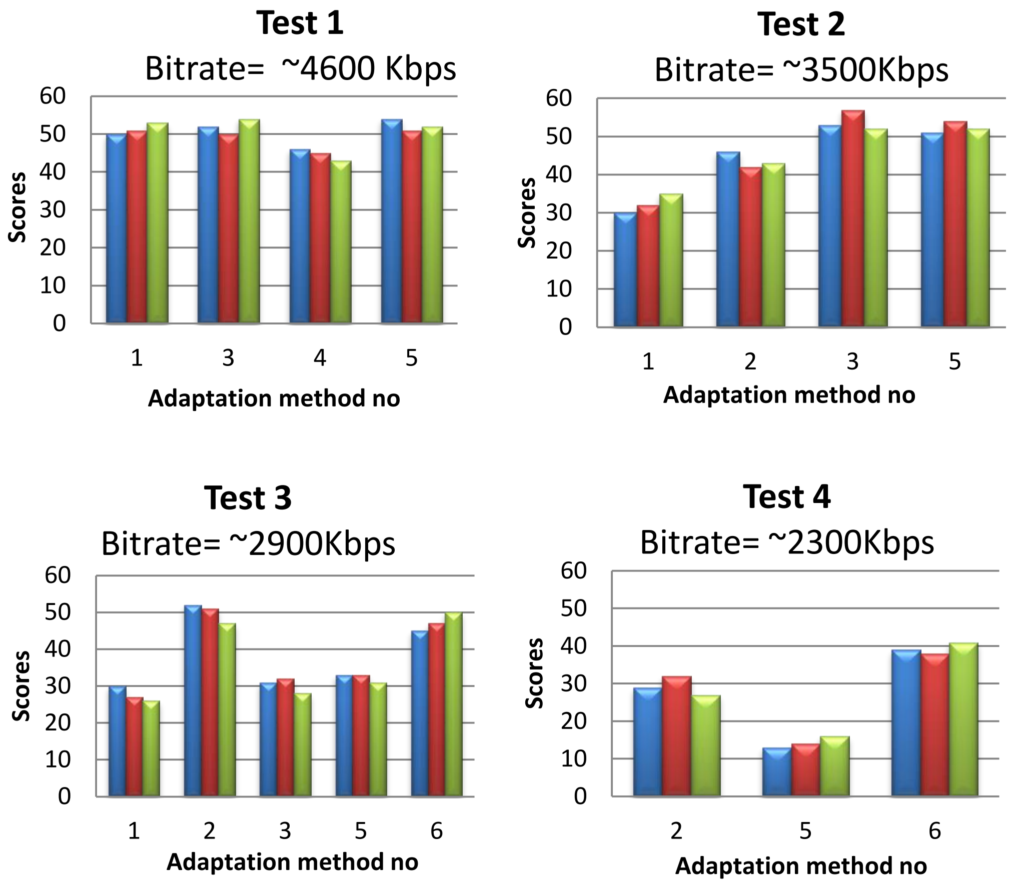


Figure 3.5: Subjective test results

3.4 Extension of P2P Architecture for QoE Aware 3D Video Streaming

Among all the results of the subjective tests that we have conducted regarding both stereoscopic 3D video and multi-view video, one of the key finding in that should be underlined is that the rate adaptation for 3D video can be very complex. In order to support complex decisions that should be taken, one of the best methods is to actually distinguish the decision making process from its implementation. For this purpose, we propose a modified P2P chunk scheduling system that is composed of two modules. The first module is for the adaptation decision module (ADM), which is responsible for making rate adaptation decisions. The second module is the chunk scheduling module (CSM), which is responsible for executing the decisions and making rate adaptation in a way that ADM prefers. In the following sections, we describe how these two modules operate and how modularity is preserved so establish a flexible architecture.

3.4.1 Adaptation Decision Module

One of the key reasons for distinguishing adaptation decisions from their execution is that these decisions are actually affected by too many entities and some of those are not directly related to P2P engine. However, if adaptation decision module can make the evaluations and the instruct the P2P engine about the result, that would simplify the design of the P2P architecture and the job of researchers that work on purely network related issues in P2P solutions.

Figure 3.6 depicts an advanced adaptation system for a P2P architecture in which the ADM isolates factors that affect the rate adaptation decision and instead accesses such information and then forwards only stream prioritization information.

In the following, we provide a few examples to the workflow of the proposed design:

1. For instance, the ADM can receive feedback from a user tracking module that would provide the information about the views that are visible to the user. In such a case, the ADM can increase the priority of the visible views. It may also prioritize the neighboring views that are close to the visible ones, since it is possible that the user can change his/her field of view by a small amount. One key decision is that, when this approach is combined with scalable video coding, there may be a case such that the enhancement layer of a particular view in the FOV can be more important than the base layer of another view. Such a decision can only be made in

the case of MVV and probably not even logical in monoscopic or stereoscopic 3D video.

2. Another example case is when ADM receives feedback about the user preferences. For instance, a user may be watching a live concert and is more interested in object based high definition audio than the high quality video. In such a case, the object based audio can have a higher ranking than the enhancement layers of the video and may become the most important stream if the user desires to do so.
3. Yet another example is when the ADM parses a description file and estimates the prioritization among views based on the available information. For instance, the camera calibration parameters of the color and depth view can be used determine, which views are better estimated and which views are could not be estimated without introducing significant artifacts. Again, such information can be used to prioritize the streams.
4. The combination of all can be used in the decision process with even further feedback from other devices/modules that we do not list.

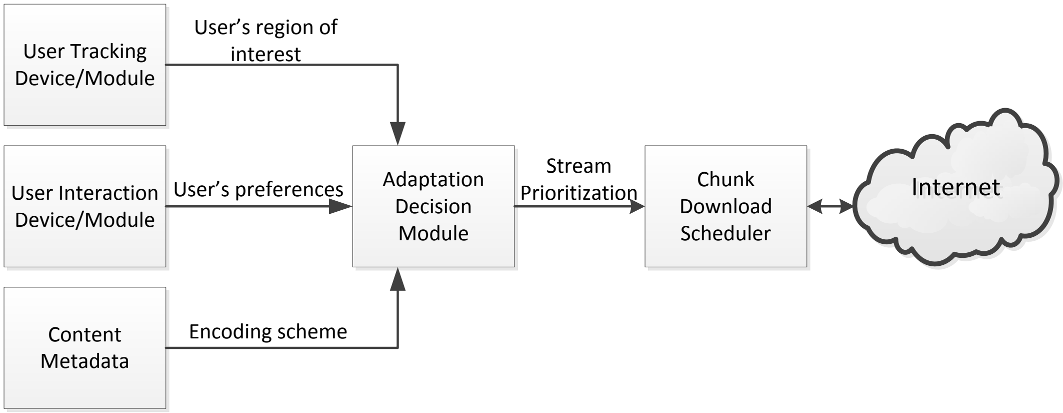

User Tracking Device/Module
User's region of interest
User Interaction Device/Module
User's preferences
Adaptation Decision Module
Stream Prioritization
Chunk Download Scheduler
Internet
Content Metadata
Encoding scheme

Figure 3.6: Design for QoE aware P2P video streaming

3.4.2 Chunk Scheduler using Multiple Windows

One of the simplest methods to implement a modification to support multiple views is to introduce multiple scheduling windows one per each view as depicted in Figure 3.7. In such a case, the available buffer duration is calculated as the minimum buffer duration of individual windows. Then, it is possible to perform rate adaptation using the method in the standard monoscopic video streaming.

When buffer duration decreases to a level that forces the P2Pengine to drop one of the views the scheduler decide on which view to drop using the ranking list forwarded by the ADM. The scheduler does not need to know how to generate such list based on when the user is currently seeing. Such implementation would complicate the P2P engine and would make it a very “static implementation to a particular problem.” In this architecture, the P2P engine remains as “adaptation decision method agnostic” but still be QoE aware.

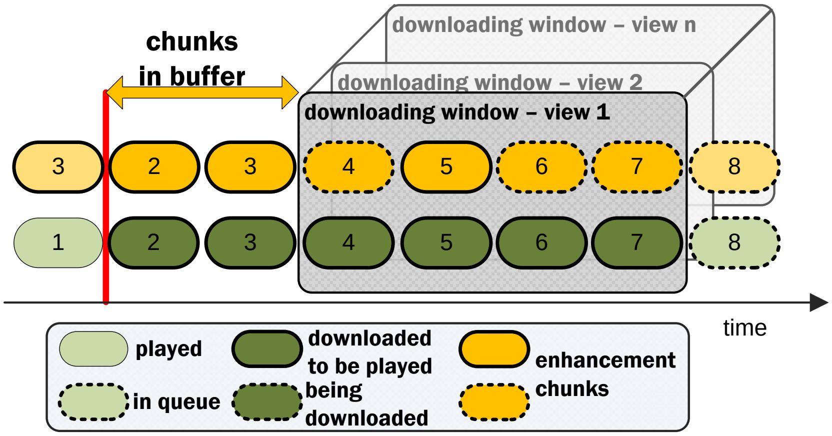


Figure 3.7:Downloading window with two layers and n views
(The red line indicates current location of the player.)

3.5 Adaptive Streaming Tests: QoS Perspective

In this section, we evaluate the adaptation methods that are described in Section 2.2. Our goals are to lose as less number of chunks as possible and if a chunk loss is inevitable due to lack of network resources, the system should perform adaptation in a way to lose chunks of the least important stream first.

We have performed two types of tests. In the first four tests, we use fixed bitrate for the total available network resources and evaluated the performance of the system to deliver to best quality under a steady condition. In the last test, the state of the network changes dynamically to evaluate the performance of the system while adapting different network conditions.

3.5.1 Test Setup

In adaptive streaming tests, we record the downloading results of a peer that is connected to 5 neighboring peers and it is also connected to a main seed server. Each neighbor has 0.5 Mbps upload capacity whereas the main seed server has been limited to 1 Mbps. This is actually a realistic scenario because even in BitTorrent application a peer may have many peers in its list (peer list) but the number of active connections is commonly much less then this number.

As for the content, we have used three dummy streams each at 1Mbps. There are two chunks per second per stream. The duration of the streams is 330 seconds long. The gap between the P2P channel and the DVB channel is set to two seconds, which serve as very limited buffering duration.

3.5.2 Test Scenarios

We have performed of five different tests, each for a different goal. Table 3.9 summarizes the test conditions and the list of target results. Figure 3.8 depicts the total available network resources for the traced peer. In each test scenario, the main seed server is always active but the number of peers that are active can change over time.

Table 3.9: Test conditions and best case results for adaptive streaming tests

Test Conditions			**Best Case Results**		
No	**Network State**	**Total Capacity**	**Stream 1**	**Stream 2**	**Stream 3**
1	Fixed	1.1 MB	Lossless	Total Loss	Total Loss
2	Fixed	1.5 MB	Lossless	Partial	Total Loss
3	Fixed	2.1 MB	Lossless	Lossless	Total Loss
4	Fixed	2.5 MB	Lossless	Lossless	Partial
5	Dynamic	Varying	Lossless	Partial	Partial

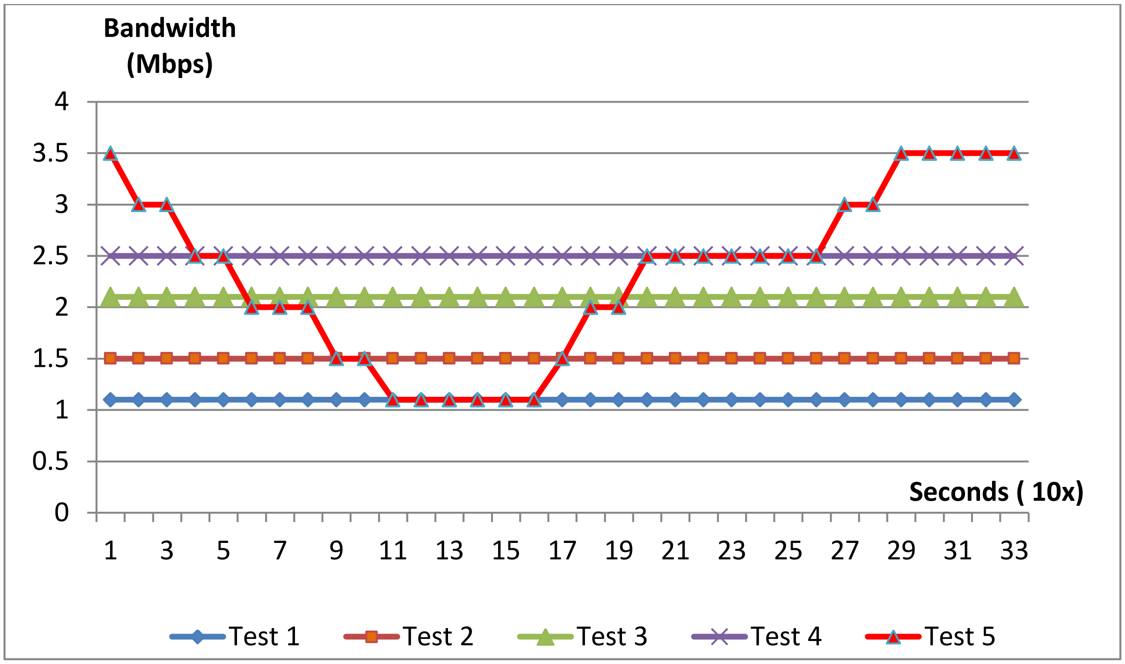


Figure 3.8: Total bandwidth capacity of the nodes that provide chunks for tests

3.5.3 Test Results and Interpretations

The results of the adaptive streaming tests suggest that the system can adapt to different network conditions within a couple of seconds. Figure 3.9 to Figure 3.13 presents the number of lost chunks from each stream for within durations of ten seconds intervals. The losses from the first stream (most important) are colored as red, whereas the blue and gray colors indicates the chunk losses from stream 2 and stream 3 (least important) respectively. There are 20 chunks within that duration for each stream.

In most of the cases, the system achieves the target goals but there are still some points that should be underlined.

When the system first starts, it tries to receive all the streams. If the bitrate is not high enough within this initial stage, then chunk losses can occur from any stream. Thus the system starts in an optimistic manner but adapts to the network conditions within the first five seconds of the streaming session. (Figure 3.9, Figure 3.10, and Figure 3.11)

The system can partially receive a stream if the bitrate is not enough to receive the whole stream but it is high enough to receive some bits. If fluctuations due to such cases cause disturbance for the users' QoE, the player can always discard such streams and display only fully received streams. (Figure 3.10 and Figure 3.12)

The system can respond to dynamic changes in the network significantly well if it has a buffer with two-three seconds of duration. The only condition that the adaptation fails is the case when the system has no buffer like in the case of initial start. (Figure 3.13).

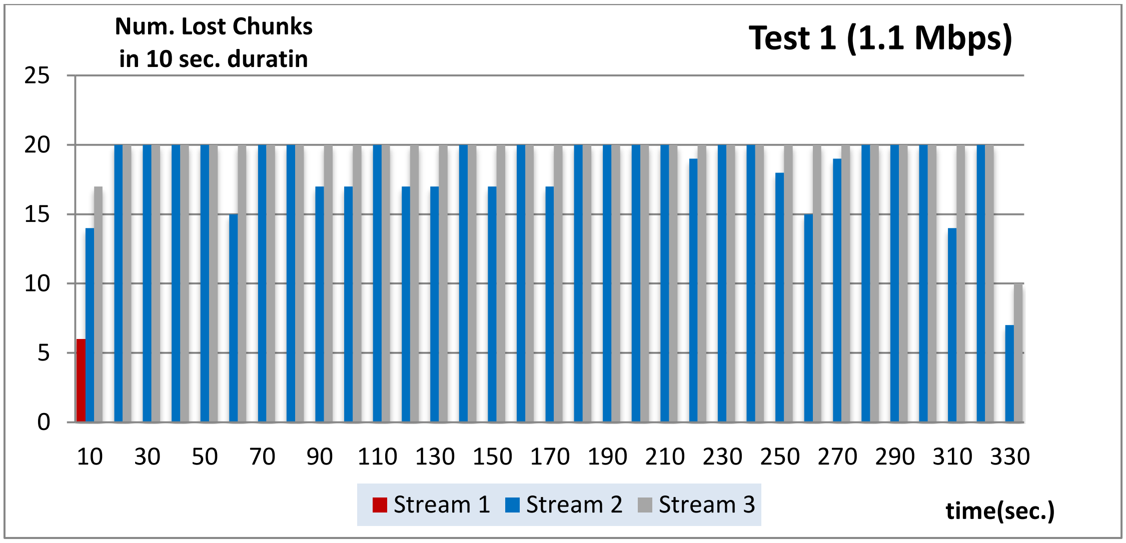


Figure 3.9: Results for Test 1

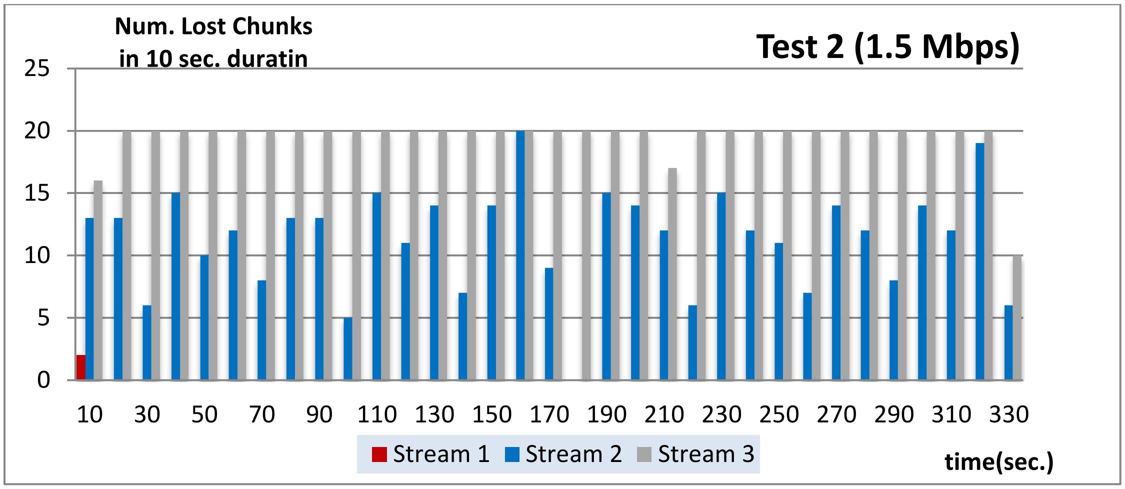


Figure 3.10: Results for Test 2

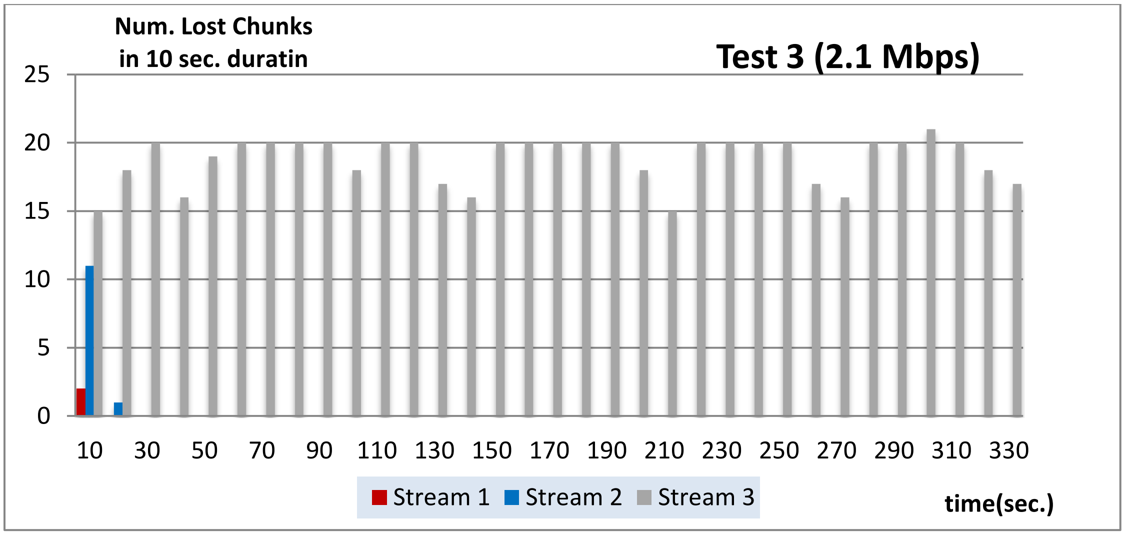


Figure 3.11: Results for Test 3

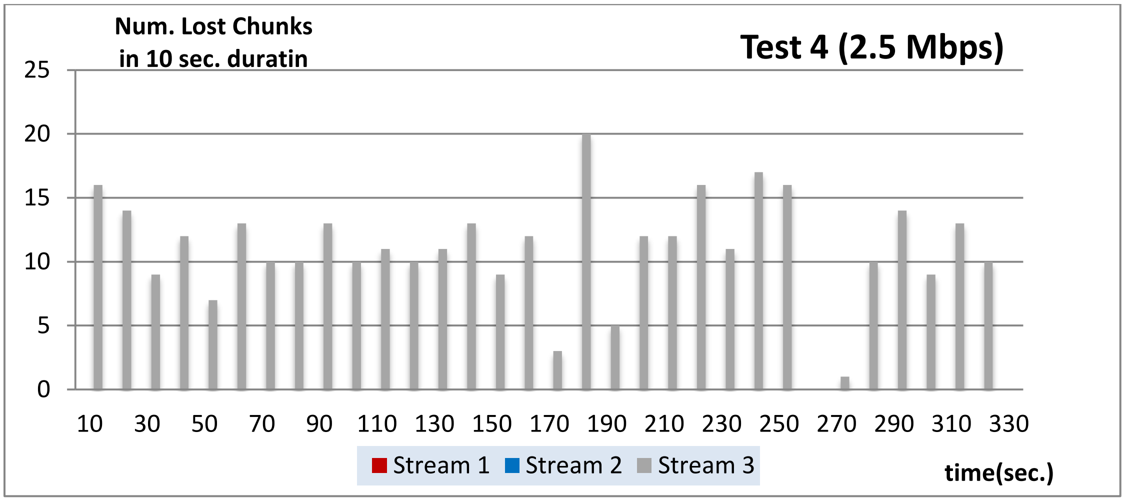


Figure 3.12: Results for Test 4

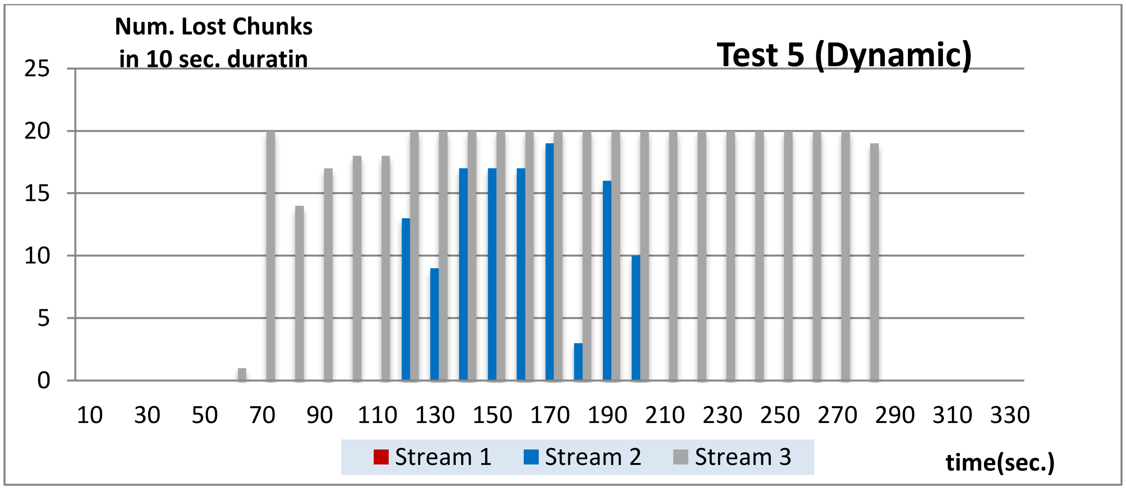


Figure 3.13: Results for the Test 5 (dynamic)

3.6 Adaptive Streaming Tests: QoE Perspective[1]

Accordingly, the P2P streaming client performs adaptation by means of adding or removing streams. Adaptation decision module is in charge of assigning the priorities of each stream depending on the particular contribution of the selected streams in the overall 3D QoE considering user's particular viewing preferences.

The purpose of the series of tests performed in this subsection is to depict the performance of rate adaptation under different server upload rate conditions by measuring the QoE of the user throughout the media playback. The adopted 3D QoE models are reported in [90]. In particular, the QoE models developed and reported for video+depth and stereoscopic video (L-R) are used in outlining the results. It was previously reported that the 3D-QoE models are proved to attain over ~90% correlation with 3D subjective tests scores. Hence, they can be regarded as a reliable measure of users' QoE.

3.6.1 Test Setup

Tests are performed using two video sequences called *Street* (2 views + 2 depth maps 1280x720, 25fps) and *Music* (4 views + 4 depth maps, 1280x720, 25fps). For each of the test video sequence, 8 different streaming tests are performed. First four tests (Tests 1, 2, 3 and 4) contain one seed server and one peer. The QoE measurements are done on that peer. In the second set of tests (Tests 5, 6, 7 and 8), another seeder is added to the existing seed server and the peer. The total duration of the streamed multi-view content is selected as ~35 seconds.

[1] The QoE tests are conducted in collaboration with Dr. Erhan Ekmekcioglu from University of Surrey

3.6.2 Test Scenarios

For Music sequence tests, the total upload capacity of the seed server is set as 8.9 Mbps, 7.4 Mbps, 6.0 Mbps and 4.9 Mbps for Test 1, Test 2, Test 3 and Test 4, respectively. In tests 5, 6, 7 and 8, the upload capacities of both seed servers are held identical. They are set as 4.5 Mbps, 4.0 Mbps, 3.0 Mbps and 2.5 Mbps (each), for Test 5, Test 6, Test 7 and Test 8, respectively. The bit-rate of the VBR encoded multi-view video is around 8.2 Mbps in average, where the average base layer and enhancement layer rate are ~900 kbps each and the depth maps are encoded using single layer at ~450 kbps.

For Street sequence tests, the total upload capacity of the seed server is set as 4.5 Mbps, 3.9 Mbps, 3.4 Mbps and 2.0 Mbps for Test 1, Test 2, Test 3 and Test 4, respectively. In tests 5, 6, 7 and 8, the upload capacities of both seed servers are held identical. They are set as 2.5 Mbps, 2.0 Mbps, 1.7 Mbps and 1.0 Mbps (each), for Test 5, Test 6, Test 7 and Test 8, respectively. The bit-rate of the VBR encoded multi-view video is around 5.2 Mbps in average, where the average base layer and enhancement layer rate are ~1.2 Mbps each and the depth maps are encoded using single layer at ~500 kbps.

In both tests, user is interested in the central stereoscopic video pair. However, based on the adaptation decision engine, the Left video+depth pair is given a higher priority over the Right pair. As per the adaptation decision making algorithm, the enhancement layers of all views are given the least priority. The resulting 3D video quality index is scaled in between 0 and 1, where higher scores (close to 1) mean better subjective performance.

3.6.3 Test Results and Interpretations

Figure 3.14 and Figure 3.15 depict user's 3D QoE results over the duration of the video service for the Music test sequence, considering the QoE model for video + depth format. In other words, they show the user's 3D-QoE, if the stereoscopic video was rendered from either of the video+depth couples. Figure 3.16 depicts the user's 3D-QoE considering the QoE model for stereoscopic (i.e., Left-Right views) format. This format does not comprise the effects of the received depth maps. Similarly, Figure 3.17, Figure 3.18 and Figure 3.19 outline the QoE results for the Street test sequence.

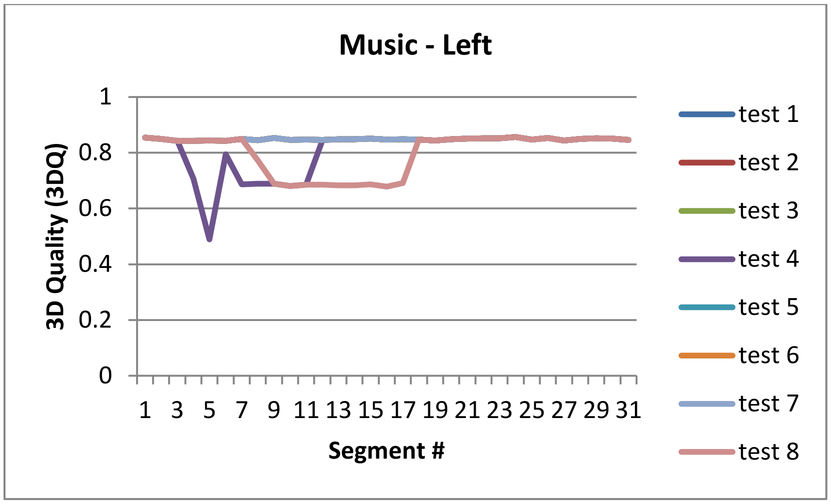


Figure 3.14: 3D video quality during playback of Music video (left view)

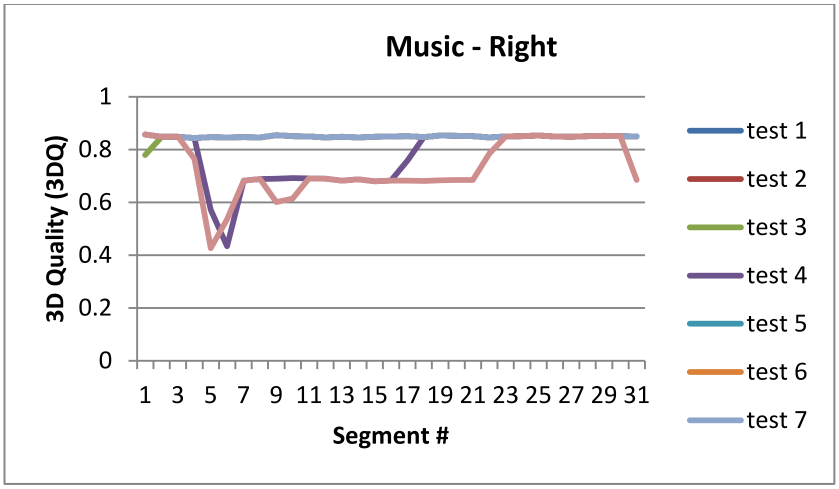


Figure 3.15: 3D video quality during playback of Music video (right view)

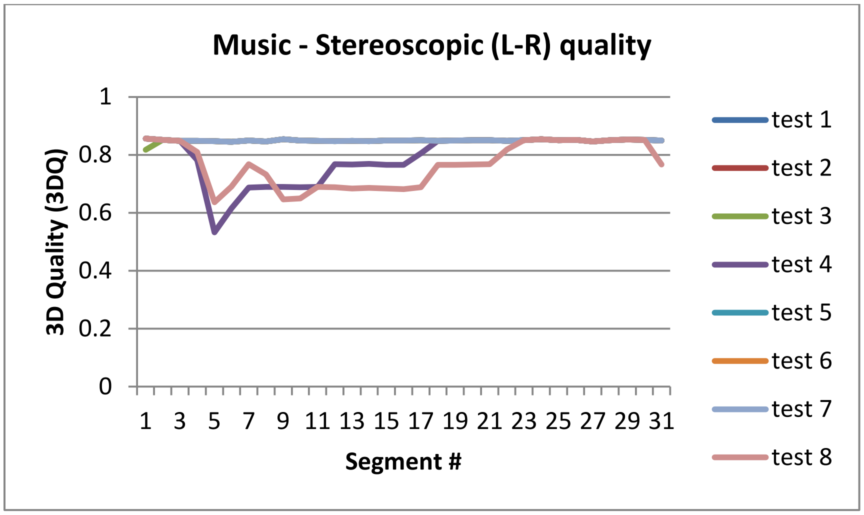


Figure 3.16: 3D video quality during playback of Music video (only color videos)

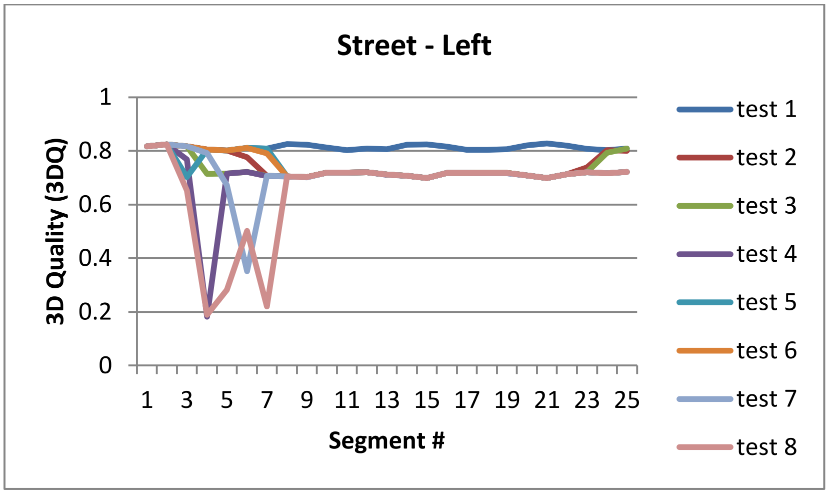


Figure 3.17: 3D video quality during playback of Street video (left view)

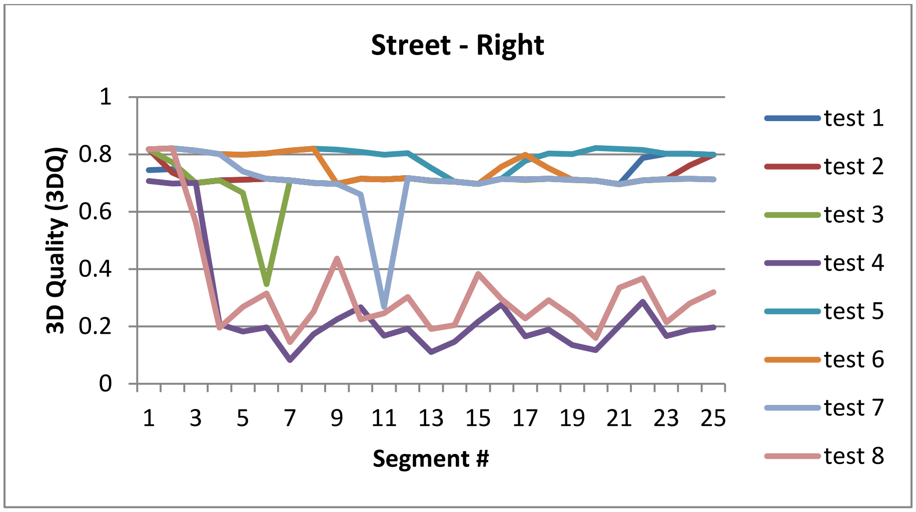


Figure 3.18: 3D video quality during playback of Street video (right view)

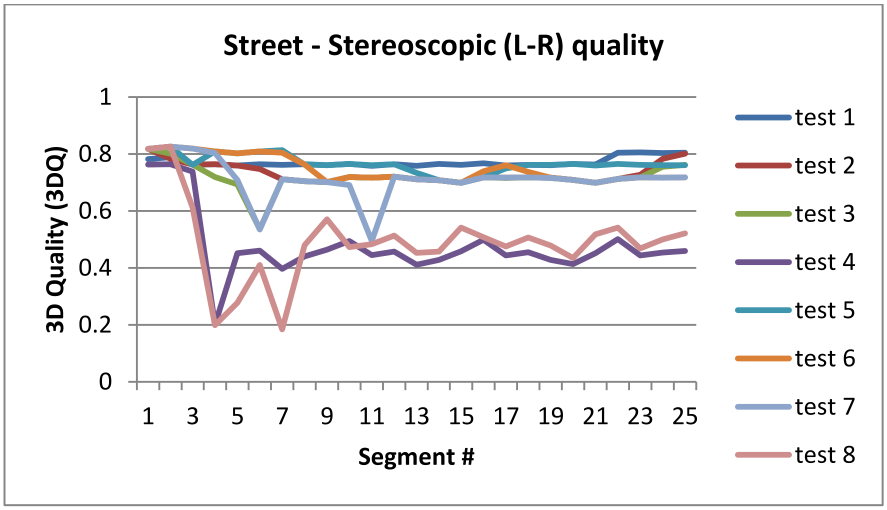


Figure 3.19: 3D video quality during playback of Street video (considering only color videos)

Each segment in the horizontal axes of above figures represents a duration of 32 frames (i.e., ~ 1.3 seconds). According to the presented results, it is clearly seen that in both test videos, and taking into consideration all network tests, the 3D video quality perceived using Left video+depth pair is consistently better than the 3D video quality perceived using Right video+depth pair. This shows us the efficiency of the deployed adaptive streaming technique, which favors the more important streams (as per the directives of the adaptation decision engine) throughout the streaming and even under bandwidth scarcity. Remember that Left view was given a higher priority than the Right view. It is also seen that according to the results of Tests 4 and 8, where the total useful bandwidth drops significantly, as compared to the required source coding rate (less than a half of it), the 3D video quality drops significantly for both pairs, also considering the stereoscopic (L-R) video quality. Nevertheless, the amount of quality degradation in Left pair is remarkably less than the degradation in the Right pair in Tests 4 and 8. Since the Music video originally contains 4 views, the outermost views are given the least importance and they are dropped for most of the time, especially in tests 3, 4, 7 and 8. Nevertheless, since they do not have an effect of the measured QoE in the centre view position, the effects of bandwidth scarcity in Tests 3, 4, 7 and 8 are not as devastating as in Street video, where all involved views are directly affecting the perceived 3D video quality. This is the reason why in average the worst results obtained Music scene are around 0.6, whereas it is around 0.2 in Street, when there are severe chunk losses or stream drops.

3.7 Summary

Broadcast of stereoscopic 3D media over digital TV platforms has already started. However, these platforms provide the 3D content at a limited quality and more importantly they are not flexible enough to support multi-view video due to their fixed bitrate transport mechanism. Hence, we foresee that, in the medium term, multi-view video services will be developed over the IP platform using various architectures, including server-client and peer-to-peer.

When the transmission is performed over the IP channel, the content should be scalable in order to match the source rate to available link capacity. In this chapter, we have evaluated various adaptation techniques for both stereoscopic 3D video and glasses free multi-view video. Moreover, we have introduced the mechanism that can take advantage of the rate adaptation methods to minimize the impact of the rate scaling on the perceived quality. When combined with the state of the art adaptation methods, P2P technology can provide high quality 3D media to home users.

Chapter 4

USE-CASE: PROJECT DIOMEDES

Developing a scalable solution (both in terms of transmission rate, number of views, as well as number of users served) for the distribution of multi-view video is critical for the successful deployment of free-view TV and video services over the Internet. The classical server-client architecture does not seem feasible, since the transmission bandwidth increases almost linearly with the number of transmitted views even with advanced coding algorithms. Moreover, the server-client architecture does not scale well with increasing number of users. On the other hand, P2P architectures may not address all the problems of multi-view video streaming due to huge amount of traffic that exceeds the upload capacity of the peers. One possible method to overcome this problem is to develop a flexible architecture that can combine both approaches in a hybrid manner.

In this chapter, we introduce the project DIOMEDES, as a use-case scenario for the proposed multi-view-video streaming architecture over P2P networks. This chapter should demonstrate the flexibility of the proposal as it can be adapted to project requirement very easily. We start the chapter with a brief overview of the project and explain its goals, the challenges and the design. Then, starting with Section 4.2, we introduce our own contributions to the project. Finally, we provide a conclusion regarding the success of the project in Section 4.6.

4.1 Overview of the Project

4.1.1 Goals and Challenges

The European project DIOMEDES [91],[92] proposes a scalable architecture that utilizes the upload capacity of peers to assist distribution of up to 200 views and associated 3D audio (see Figure 4.2). The DIOMEDES architecture aims to benefit from the current DTV infrastructure as well as the IP network. The project intends to deliver immersive experience by providing glass-free, high quality holographic 3D that is accompanied by object based audio.

There are series of challenges that should be addressed in order to provide a successful reach the project goals. These challenges are provided in the list below.

C1. Efficient Video Coding Scheme and Encapsulation of the Elementary Stream: One of the key requirements of the project is to device an efficient video coding algorithm and an encapsulation formation for the elementary stream. For the IP channel, the content should have variable length feature whereas for the DVB channel, the content must have a fix bitrate over time.

C.2 Hybrid delivery over IP using both server-client modal and P2P: According to the description of work (DoW), the project DIOMEDES uses P2P only as a supplementary source that assists a centralized 3D media server, which we name as main seed server. The challenge here is that the client application at the user terminal should be aware of two types of IP sources: i) main seed server ii) other peers. Then the client application can perform intelligent chunk scheduling to use both sources effectively.

C.3 Synchronization between IP and DVB Channels: One of the major challenges in the project DIOMEDES is to provide an efficient mechanism that will allow the variable bitrate channel (IP) to work synchronous to a fix bitrate channel (DVB). This challenge becomes very difficult to solve especially when the available bitrate of the IP channels drops below a certain value.

C.4 Supporting Wide Range of MVV Displays: Project DIOMEDES aims to create a content distribution mechanism, which can support wide range of MVV display while transmitting only required number of views.

C.5 Synchronization among audio and video clusters: The format of DIOMEDES content is far more complex than any other standard multimedia formats. At the video aspect, we use scalable encoded content. At the audio side, we use object based audio. Both of these approaches require novel decoders that are residing at different work-stations in order to meet the real-time requirements of the project. Therefore, it is very critical that the audio cluster and video clusters remain in synch in order to avoid drifts during playback.

C.6 Real-time processing of multimedia: The amount of data that the DIOMEDES user terminal has to process is significantly higher than any other similar project. Multiple views that have HD resolution are accompanied by object-based audio. Processing all the data in real-time requires significant amount of optimization in processing tools.

C.7 Data transmission between video decoders and video renderer: The video decoders generate only raw images. However, virtual view point synthesis requires huge amount of processing that is performed by another cluster. Then it becomes a problem to transfer huge amount of raw image data from video decoders to video renderers that will process the raw images / depth-maps and generate a synthetic view based on user preferences.

4.1.2 System Design

In order to address all above mentioned challenges, we have designed a solutions that is composed of multiple terminal PCs and high-end software components. The terminals has high processing power, either specialized on CPU or GPU programing whereas the software components are a composition of multiple modules that are loosely coupled with each other in order to maximize the flexibility and to support future extensions. Figure 4.1 depicts the overall system design, which we explain the following.

S.1 MPEG-TS Transport stream for Scalable Encoded Video: DIOMEDES uses SVC extension of H.264/AVC codec and uses SNR scalability option for streams that are to be transmitted over IP channel. This allows the receiver side to perform rate adaptation. On the other hand, standard H.264/AVC codec is adopted for the content over the DVB channel to maximize the encoding efficiency.

Both streams are encapsulated using MPEG-TS format. The format allows multiple sub-streams in a single content under different PIDs. Using this feature, the base-layer streams, enhancement layer streams and the depth-maps are encapsulated as different sub-steams of the same content. Encoding and encapsulation is handled by the **Content Server**, which forwards video chunks to the **Security Module,** which encrypts the chunks and transmits them to the main seed server.

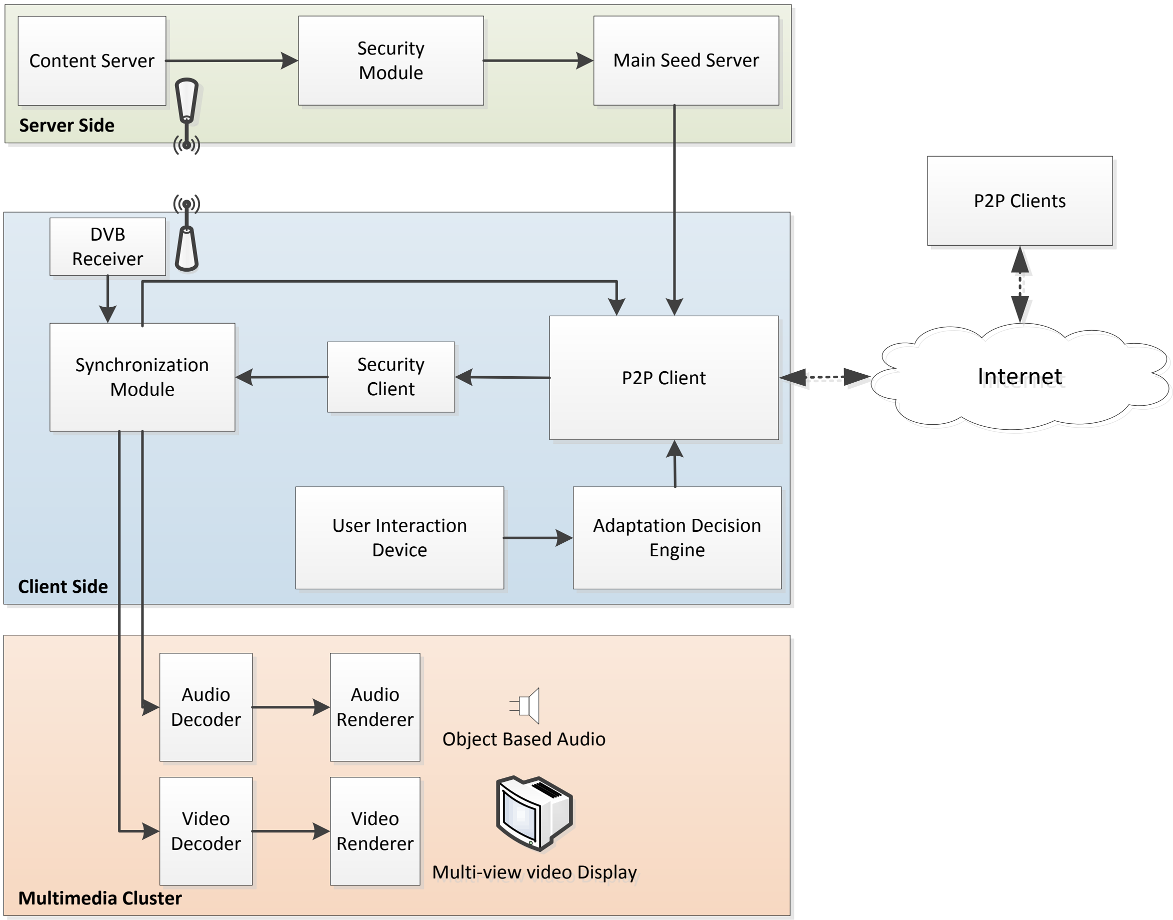


Figure 4.1: Design of the project DIOMEDES

S.2 Intelligent P2P Client: The P2P client is an implementation of the proposed mechanism that we provide in Chapter 2 and Chapter 3. The proposed mechanism can easily distinguish a seed server from other peers including the ones that become seeder over time. Therefore, the P2P client could make intelligent chunk scheduling decisions (e.g., may schedule more chunks from the seed server if the neighbor are unreliable or may avoid scheduling from the seed server in the opposite case.)

S.3 The synchronization module: The system includes a synchronization module that is responsible for synchronizing the content from the DVB and IP channels. In order to do this, the module uses PTS timing information that is present in the DVB signal. Since the rate of DVB is stable, the clock of the DVB is used. The PTS value is then forwarded to the P2P client, which uses this information in the chunk scheduling policy. Consequently, the P2P engine tries to retrieve the chunks that are only useful for the synchronization module.

S.4 Selecting the streams to be transmitted over different channels: This architecture is flexible to support a wide range of displays. For this purpose, two baseline streams are transmitted over the DVB channel. We assume that any 3D display, regardless of the number of views it utilizes, needs two baseline frames. The remaining auxiliary views are transmitted over the IP channels. For instance, a user with an *N*-view display can either receive to *N-2* video streaming over the P2P while receiving the remaining two over the DVB. As an alternative, the user may try to receive all views from the P2P channel if the DVB channel is not available.

S.5 Synchronization between Audio and Video Channels: The synchronization module is also responsible for the synchronization of the audio and video channels. In order to do this, the module first transmits DVB clock feedback to both clusters over a dedicated line with no cross traffic. Moreover, the module can also increase or decrease the difference between PTS and PCR clock of the individual streams. Therefore, it is possible to force one of the clusters to run ahead of the other.

S.6 Dedicated Workstations for Processing Multimedia Content: In order to meet the real-time requirements of the DIOMEDES, we have used multiple cluster nodes to process the content. The first cluster is video cluster which is composed of two video decoder PCs (8 streams (4+4) total) and one video renderer node that has a high-end GPU processor that is utilized in view synthesis. A second cluster is used for decoding the object-based audio content and then rendering sound according to user preferences.

Besides the hardware, we have also implemented a special video decoder that uses multi-threaded video decoding. The video codec has superior decoding performance and also provides additional features, which we will explain throughout this chapter.

S.7 Raw Image Transmission over Infiniband Communication: The two nodes that are dedicated for decoding 8 video streams have to forward raw images to the video renderer. However, the amount of bitrate required to transmit 8 HD video at 25fps is higher than the capacity of gigabit Ethernet. Therefore, we have used Infiniband connection that allows significantly higher link capacities.

In short, the streams arrive from P2P and DVB channels. They chunks from the P2P channel are first forwarded to decryption module and then buffer and the synchronization module. The synchronization module used PCR clock of the DVB channel and then forward streams to corresponding multimedia clusters. The clusters first decode the media and then render the content according to the instructions from the user interaction module. These instructions include the viewing direction and position. The synthetic scene and the corresponding audio are rendered. A sample view is depicted in Figure 4.2.

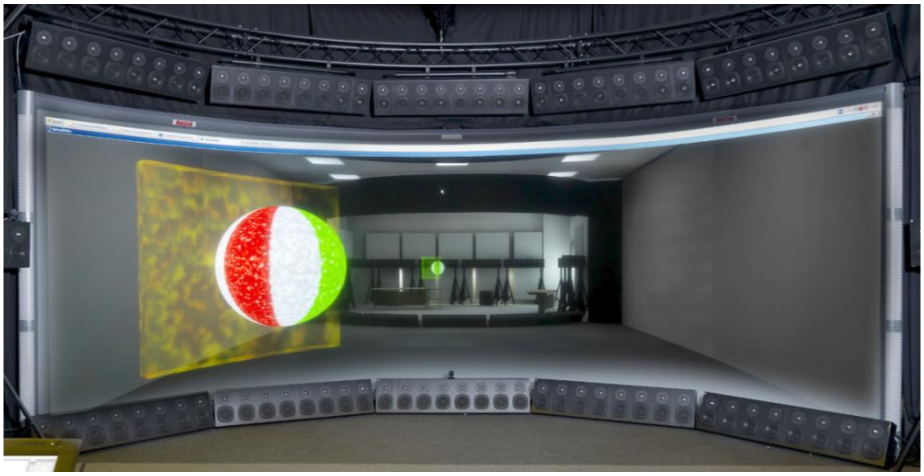


Figure 4.2: Multi-view video and spatial audio display in the University of Surrey (Courtesy of Dr. Stewart Worrall)

4.2 Main Seed Server

The main seed server is the final destination of the chunks at the server side. The server initializes a new session folder upon the reception of the metadata chunk, which provides information such as camera calibration parameters and other content related information such as number of views. The chunks and the KEY information that is used to decrypt the chunks are stored in that directory. When all the content chunks are forwarded by the security server, a termination chunk with zero payload length is used to indicate that the End Of File (EOF) of the streams. Upon reception of the EOF chunk, the main seed server performs the following sanity checks to confirm that the data is ready for delivery:

- For each PID in the metadata, there should be at least one chunk received.
- The PCR value of the first chunks of each stream should be close to each other.

- PCR values should not roll back (i.e. they should be continuously growing).
- The number of streams in the metadata chunk should be equal to the number of total PIDs.
- The GUID string of each content should be unique.

If any of these checks fails, the content is removed from the web server. Otherwise, it is registered to the currently available contents using the given GUID in the metadata.

4.2.1 Metadata Creation

A content metadata file is generated during audio and video coding and encapsulation process. The content metadata file is used in the subsequent processing stages of the content server and the security server. At the same time, the metadata involved within this file is inserted into a special chunk which is exchanged with downloading media consuming peers to let them be able to properly decode and display the content. The contents are registered in the content server with the title and the globally unique identifier (GUID) and tagged with scene type and sub-type, which are stored in the content metadata file. In order to let the content metadata be widely accessible by the involved processing modules, the format is decided to be in extensible mark-up language (XML). Apart from the descriptive title, unique identifier and tags, multi-view capture specific technical metadata is involved in the content metadata file. This metadata is used in the video and audio player units of the client terminals to correctly render the 3D scene. The number of available camera views, extrinsic and intrinsic parameters of each camera, cameras’ IDs (to successfully identify each camera in the corresponding multi-camera set), scene setup (cameras’ positions with respect to each other and with respect to the center of the scene), involved streams’ PIDs and types (e.g. base, enhancement, depth, audio) and intended delivery paths (e.g. DVB or P2P) are other elements of the content metadata file.

4.2.2 Chunk Distribution over HTTP

Once the content is available, peers can request chunks using HTTP protocol. We have used HTTP protocol in order to keep the system compatible with other server-client based adaptive streaming solutions. In HTTP streaming, it is possible to create pull based streaming mechanism that is controlled by the receiver.

4.3 DIOMEDES P2P Engine

4.3.1 Adaptation Decision Engine

Adaptation decision engine's operation and the overall video adaptation approach were described previously in Section 3.4.1. Adaptation decision engine is in charge of deciding on the periodical importance levels of particular streams (video quality layers and depth maps, and audio) and ranking them according to their relative perceptual importance. Actual adaptation with respect to changing network conditions is executed in the P2P streaming client. The input parameters the adaptation decision engine takes are the KPI values for each successive GOP (delivered through the video decoder), user's viewing preferences (i.e., selected free-viewpoint position), list of all available streams associated with the downloaded 3D content (e.g., knowledge on the total number of camera views), camera parameters and the knowledge on the existence of a DVB 3D video reception.

Readers are recommended to refer to D3.6 of the DIOMEDES project to have a deeper insight on the operation of adaptation decision engine. As long as the user's preferred viewing position stays constant and the KPI parameters do not change significantly over time, the output PID ranking list delivered to the P2P streaming client also stays constant. The ranking order tends to change only if the user slides its viewing position using the user interaction module, if there is significant change in the KPI values in the delivered camera views, or the state of multimedia delivery/ context of consumption changes (e.g., DVB reception becomes active, or user switches to a multi-view screen from stereoscopic screen). Audio streams are given always the highest priority over any other video stream, because of two inherent assumptions. First, the disruptions in the spatial audio playback have more detrimental effects on users' QoE. Therefore they are always given the highest chance to be downloaded even under severe network conditions. Second, since the chunk sizes and the content bit-rate for the audio streams are remarkably smaller than that of the video streams, they are not sacrificed at times of bandwidth adaptation through stream truncation. Because, the adaptation range they would offer would be far less significant than the adaptation range the video streams would offer.

4.3.2 Chunk Validation

The chunks are forwarded to the security server to be encrypted. The server generates a chunk header that is not encrypted and provides information such as *chunkID*, *contentID*, *PID*, *PCR* (clock), *chunkNo*, *viewID* and payload size. When a peer receives a chunk, it checks these fields to validate the chunk. If the peer manipulates the unencrypted header, the security client module can detect such actions and inform the P2P module about the integrity failure. In such cases, the client disconnects from that particular peer and updates the blacklist table, which holds list of blocked peers' IDs.

The security server creates distinct key pairs that are used to encrypt and decrypt the chunks for different content. Without those keys decryption of the content is not possible. The key is forwarded to the main seed server as a special chunk with the KEY flag set. Similarly, the security server forwards the metadata information that is described in Section 4.2 to the main seed server in a special metadata chunk.

4.3.3 Selective MDC Approach

Multiple Description Coding (MDC) increases the robustness of data delivery by transmitting distinct descriptions (self-decodable bitstreams) over different paths (path diversity). In theory, if a peer churn (ungraceful exit of a peer) occurs, a receiver can still receive another description and proceed with decoding but at a lower quality output. Two issues should be considered about MDC in mesh-based P2P video streaming solutions.

First, one needs to be careful about the implementation of MDC and keep an eye on the redundancy it causes. A very simple example can illustrate the effect of redundancy. Consider two cases. In first case MDC is used and it has 20% overhead. In the second case, MDC is not used so all the available channel capacity is dedicated to source coding (video in our case.) Table 4.1 presents the state of the peers' video buffer after 10 seconds if no peer exit event has occurred in that duration. Considering that the video rate is 800Kbps, the *Peer1* has 10 seconds buffer whereas the second peer has 12.5 seconds of data in buffer. So, the second peer has 2.5 more seconds to handle a peer exit event. If it can find a new peer in a shorter interval, then not using MDC is a better option and vice versa. This simple derivation is just to show that if not used properly, MDC can be harmful as well, due to the redundancy causes.

Table 4.1: Simple comparison of MDC, state of peers after 10 seconds

	Bandwidth	**Uses MDC**	**Redundancy**	**Actual Receive Rate**	**Data in buffer after 10 seconds**
Peer1	1000Kbps	Yes	20%	800Kbps	8000Kb
Peer2	1000Kbps	No	0%	1000Kbps	10000Kb

Secondly, one of the best cases to apply MDC is adopting it in data distribution with multiple-trees in which the content delivered in push manner. Commonly, in push based streams, the sender does not wait for notification from the receiver side and forwards data to its child node as it receives the data from above. Therefore, the sender commonly is unaware of the buffer state of the receiver (especially the data received from over other trees) as depicted in Figure 4.3. The sender just blindly forwards the content in a way that it maximizes the probability of receiving useful data. However, in mesh-based topologies in which data requests are receiver driver, the receiver does have the knowledge of the data in its buffer. This has two consequences: i) The sender does not have to request the redundant part if it already received it from another peer. ii) What the receiver has to estimate is the probability that a request from another peer will be successful.

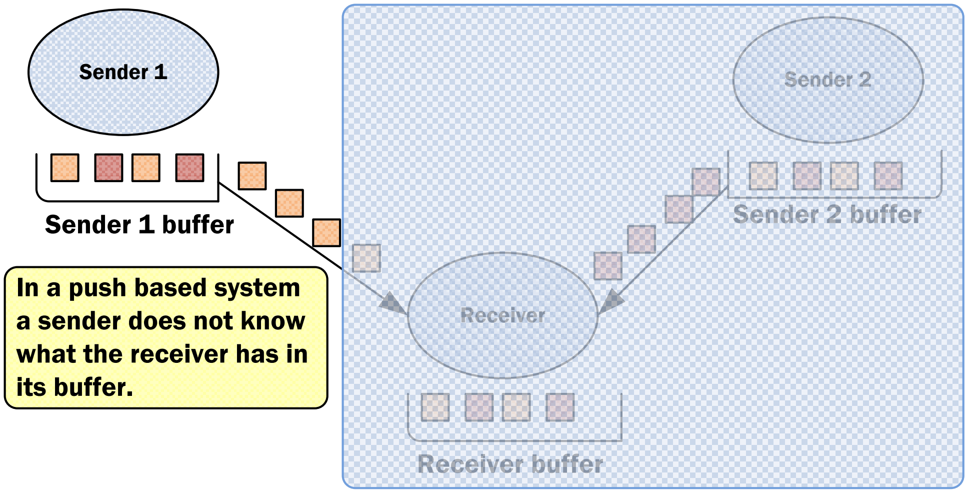

Sender 1
Sender 1 buffer
In a push based system a sender does not know what the receiver has in its buffer.
Sender 2
Sender 2 buffer
Receiver
Receiver buffer

Figure 4.3: Push-based model (commonly adopted in tree-based topology)

DIOMEDES P2P engine is based on the proposed P2P mechanism (AdaptP2P). It has a mesh based topology and a peer has the advantage of deciding when to request chunk, from whom to request and which layer to request. With these options at hand, we adopt an MDC system in which the multiple descriptions are requested only when necessary in order to keep the bitrate as low as possible. For instance, a peer is to request a chunk that is available in peers which have low upload capacity (at least the current connection quality is poor). If the peer has long buffer duration then it does not have to use MDC because it can retry to download it from another peer if the current transmission fails. On the other hand, if the deadline is close then it is safer to request data from multiple peers and try to receive at least one copy of the chunk.

In order to be able to switch between single description coding and multiple description coding the redundant and the unique data of the descriptions should be individually accessible. If the descriptions are packed in a single chunk for instance, then it is not possible to avoid the redundant part of the descriptions. However, if the descriptions are split in two as redundant and unique data then it is possible to perform MDC streaming selectively.

In DIOMEDES P2P, the descriptions are generated using the base and enhancement layer of the content as depicted in Figure 4.4. In our case, the redundant data is the base layer and the unique data is the enhancement layer (that is distributed equally among descriptions.) Since each of these data structures can be addressed separately (they are in different chunks) than it is possible to perform selective MDC streaming. Moreover, a description can also be scaled if the enhancement layer is discarded.

The implementation of MDC in DIOMEDES P2P is as follows. A peer always ranks their neighbors according to their upload rate. When a chunk is scheduled for download, candidate peers are sorted based on their upload rate. If peer with high upload capacity is available, then the chunk is requested only from that particular peer. However, if no peers with high upload capacity is available then the state of the buffer determines the next step. If the duration of buffer is long, then we take the risk of downloading a single chunk from a single peer with low upload capacity. However, if the buffer is low and the chunk is a base layer chunk then we make two identical requests from two different peers. The corresponding state diagram is provided in Figure 4.5.

Description 1

Frame No:	1	2	3	4	5	6	7	8
GOP 1	B	B	B	B	B	B	B	B
Frame No:	9	10	11	12	13	14	15	16
GOP 2	BE	BE	BE	BE	BE	BE	BE	BE

Description 2

Frame No:	1	2	3	4	5	6	7	8
GOP 1	BE	BE	BE	BE	BE	BE	BE	BE
Frame No:	9	10	11	12	13	14	15	16
GOP 2	B	B	B	B	B	B	B	B

Figure 4.4: Multiple description scheme in DIOMEDES for GOP size 8 (B Base layer, E Enhancement layer)

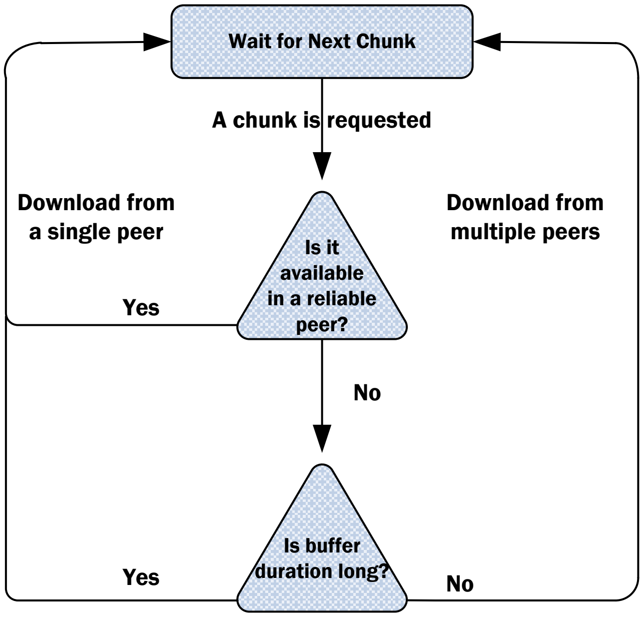

Wait for Next Chunk
A chunk is requested
Download from a single peer
Download from multiple peers
Is it available in a reliable peer?
Yes
No
Is buffer duration long?
Yes
No

Figure 4.5: Chunk scheduling in for MDC

4.4 Modifications to OpenSVC Decoder

In order to meet the real-time decoding requirements of the project, we have started with the OpenSVC Decoder , a modified version of ffmpeg decoder that runs in real-time. However, unlike ffmpeg, OpenSVC Decoder can be natively compiled in Windows platform using Visual studio, which significantly enhances the process of debugging. There were three modifications that we have performed on top of the original project.

The first one was using macroblock level parallelism that further increases the decoding performance. This was significantly important when multiple streams were to be decoded simultaneously otherwise the degradation in the performance of the decoder may force it to operate slower than real-time (especially when four simultaneous decoders are run in a single PC).

Following that, we have added network functionalities to the video decoder. The decoder is part of a large system and operates in a separate PC therefore it needs to communicate with other applications. To this end, we have added socket communication for the decoder to enable elementary steam reception and raw image forwarding.

Finally, time-stamping information was required so that the renderer module can see which frame belongs to what time. For this purpose, we have received timing information along with the elementary stream from the TCP socket. We have modified the source code such that this information is stitched together with a frame. In order to do this, we had to modify all decoding function call and insert and extra variable. We have also modified the state structure of the frame. Later, we provide the timing information to the renderer once the frame gets decoded.

In the following we provide the details of macroblock level parallelism that we have implemented on top of Open SVC Decoder software:

The VCL design of H.264 video coding standard follows a block-based hybrid video coding approach, where each coded picture is partitioned into fixed-size macroblocks

(MBs) that each cover a rectangular area of 16x16 samples of the luma component and 8x8 samples of the chroma components. All samples of a macroblock are either spatially or temporally predicted, and the resulting prediction residue is encoded using transform coding [93]**.** The coding standard ensures that MBs of each frame are processed in a raster-scan order and each of them are coded differentially with respect to its already-decoded neighbors to its left, top-left, top and top-right as shown in Figure 4.6.

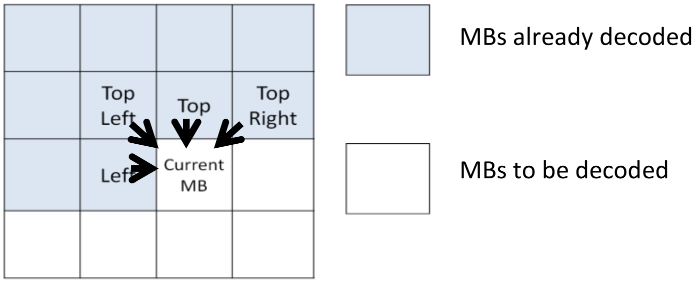
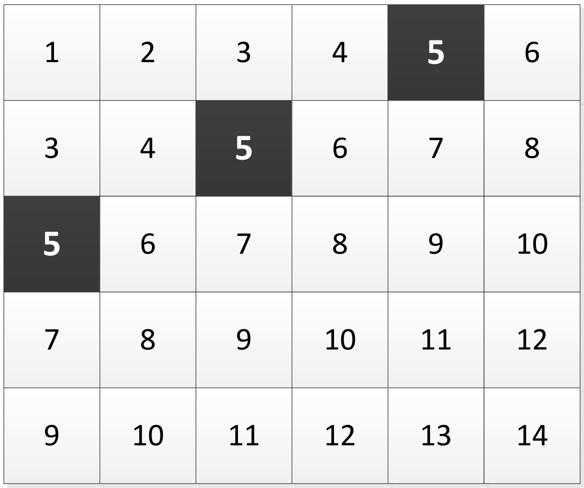


Figure 4.6 Spatial dependencies between neighboring macroblocks

Intel Thread Building Blocks (Intel TBB) library was the choice of environment to implement the parallel algorithm in this section. Due to the dependencies shown in Figure 4.6 left, top-left, top and top-right MBs should be fully decoded before the current MB can be started. The decoding of MBs can be represented as a DAG (Directed Acyclic Graph) with each node of the DAG representing the corresponding decoding of that MB by one processor.



1	2	3	4	**5**	6
3	4	**5**	6	7	8
5	6	7	8	9	10
7	8	9	10	11	12
9	10	11	12	13	14

Figure 4.7: Decoding order of macroblock in parallel order

Figure 4.7 provides a simplified example for macroblock level parallelism. Now, according to dependencies we have explained above, the decoding processes starts with the top-left macroblock (1) and followed by the one of the right (2). At this time two other macroblocks can be decoded simultaneously, that are represented by (3). Following this approach, at the 5th step, three macroblocks can be decoded in parallel.

As the size of the frame increases, the number of macroblocks that can be decoded in parallel increases as well. For instance, for the example above, normally there are 24 macroblocks that should be processed in 24 decoding time (assuming that each MB requires roughly the same amount of time). However, using macroblock level parallelism it is possible to decode these in 14 decoding time.

Unfortunately, the increase in the performance is not a high as the above number suggests. First, the variable length coding (decoding) operation cannot be parallelized, which is a substantial portion of the decoding process. Secondly, there always exists a threading overhead, which is due to thread creation and synchronization. Therefore, the level of achieved improvement was limited in this study.

Measurements And Results: The performance and energy consumption measurement tests were carried out on a laptop running on Intel® CoreTM i7 720-QM quad- core processor at 1.60 GHz with 6M cache. The Enhanced Intel Speedstep Technology (EIST), Turbo Boost Technology and the Intel Hyper-Threading Technology offered with this processor provide the adjustability on the processor performance to observe the changes in the energy consumption for a given task.

Measurements aims to observe the performance change introduced with parallel MB decoding. Only the parts which include the parallelized MB decoding process were taken into account. The speedups were calculated with respect to the original Open SVC function running on sequential MB decoding algorithm is presented in Figure 4.8.

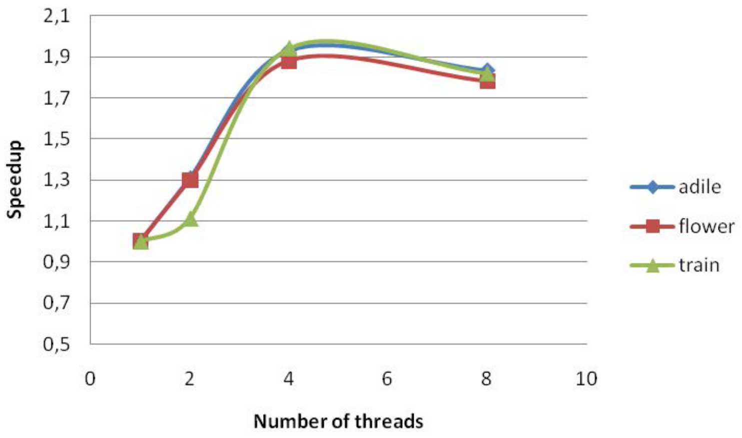


Figure 4.8: Level of speed up using macroblock level parallelism

4.5 Tracker Protocol

The tracker is responsible for tracking peer-list and content servers and informs peers about the current state of the swarm**.** In BitTorrent, tracker server randomly forwards a subgroup of peers. Definitely it is possible to follow an intelligent approach for peer selection. Instead of selecting partners randomly, Hei et al. claim that buffer maps can be utilized to monitor network behavior of a peer and suggested matching peers based on the state of their buffer maps (a data structure that indicates the chunks that are available in a peer) [94]. The authors state that it is important to choose a peer with similar network resources.

In our solution, tracker server clusters peers according to the requested views. When a peer connects for the first time, tracker forwards a random subgroup of peers. However, as the session progresses, the tracker server can make better suggestions to peers. The tracker server simply tries to match peers that are receiving equal number of views (which roughly represents their data reception rate).

4.6 Summary

In this chapter, we have introduced the DIOMEDES project, which has very similar goals with this thesis and therefore stands as a use-case scenario. Besides adapting the proposed P2P video streaming engine to the requirements of DIOMEDES project, we have also contributed to open SVC decoder, which is another key module of the project.

Chapter 5

CONCLUSIONS

As the final words regarding this thesis, we would like to start with a reviewing the features that distinguishes the proposed architecture from the BitTorrent protocol and the currently available solutions (Chapter 2, P2P video streaming). Then, we will discuss how these features can be extended to support 3D video (Chapter 3, extensions to support 3D video). We believe that having a final look the features with reasoning would be an important step in understanding the main arguments of this thesis. We provide this analysis in Section 5.1.

Following that summary of the design, we would like to underline some of our key findings that we find when the proposed solution is compared against currently available approaches regarding P2P video delivery systems. In Section 5.2, we pinpoint these findings and also provide some more regarding streaming of 3D video. Unfortunately, we could not provide a comparison for the latter one, due to lack of mature proposals.

Finally, in Section 5.3, we would like to conclude with future directions in the study of the P2P video delivery. We hope that theses directions would help our colleagues in their future study on P2P video streaming solutions and especially the ones that supports 3D video.

5.1 Summary

We base the AdaptP2P system on BitTorrent approach and adopt many of key features that are present in BitTorrent. These can be listed as:

i) Mesh-based topology

ii) Receiver-driven chunk scheduling

iii) Dividing the content into multiple sub-streams (chunks)

iv) Introducing randomization in the distribution of chunks

v) Providing incentive to peers to utilize their upload capacity

These five fundamental are the key that make the BitTorrent protocol a success. The flat mesh-based topology increases the robustness of the system because in a flat topology participants do not have special duties such as being a super node or group leader. Then, loss of any peer has a limited impact on the remaining ones. In the other approaches where there are super nodes or group leaders (e.g., Napster) loss of important peers introduce more complications that take longer times to recover.

In BitTorrent every node is responsible for using the available information from metadata session descriptor, tracker server and neighbors to maximize its own benefit. This receiver-driven approach creates a more realistic scenario in which peers decide on their. A peer decides from whom to download a chunk and to who upload a chunk. This allows peers to take reasonable actions that allows implementing efficient incentive mechanism, even as simple as tit-for-tat. In a tree-based system however, a peer is dedicated to forward data to some child nodes no matter what they do and how they behave. It becomes increasingly difficult to assess, which node is behaving selfish and which node is contributing enough.

While these five features are key to establishing a successful P2P file sharing application, when it comes to distribute multimedia in real-time they become inefficient.

Starting with the chunk picking policy, many of the fundamental implementations can be tuned for video streaming purposes. In the following, we provide these modifications.

i) Fixed-length chunks are not the optimum packetization format for multimedia data, which is composed on self-decodable units that have variable bit lengths. Therefore, while splitting the content into multiple sub-pieces is a good approach, the format of the chunks can be optimized to increase the efficiency in video streaming.

ii) The simplest and the most obvious modification is in the chunk scheduling policy. The BitTorrent protocol uses rarest first approach, which totally neglects the timeliness concerns. A more suitable approach for choosing the next chunk to be downloaded (scheduling) is a mandatory requirement.

iii) As a binary file sharing platform, the BitTorrent protocol does not have discardable chunks and does not intend to perform rate adaptation. However, the unlike binary file transmission, the multimedia content is loss-tolerant (especially if scalable video coding is adopted). Therefore, instead of trying to receive all chunks, the chunk scheduling policy can be modified to support rate adaptation by discarding less important chunks.

iv) The incentive mechanism that is adopted in BitTorrent forces peers to experience a low efficient period in which they are unchoked by a neighboring peer if they are lucky. While having such a short period of underutilization is acceptable while joining to a new session for file sharing, this situation is not very appropriate for video streaming services, in which users can frequently change their session. Therefore, a more efficient incentive algorithm is needed.

AdaptP2P introduces these modifications to have a better solution for monoscopic video streaming over IP network (details are available in Chapter 2). On top of those, AdaptP2P provide a flexible rate adaptation mechanism, which allows an external module to decide on the way of making adaptation decisions. This functionality is especially important in the case of 3D video streaming because unlike 2D video, the adaptation process is not straightforward. In 2D video, the state of the art can be summarized as adopting scalable video coding and discarding enhancement layer in case of bandwidth scarcity. However, in the 3D case, issues like asymmetric rate scaling and/or view scaling requires more flexible architecture. In order to keep modularity, we define an external module (Adaptation Decision Module) to decide on streams that should be discarded in a way that minimizes its effect of QoE. The P2P engine only executes this decision. The details regarding extensions to support stereoscopic 3D and multi-view video is presented in Chapter 3.

5.2 Key Findings

5.2.1 P2P Video Streaming

In the following list, we provide the key finding when AdaptP2P is compared against currently available approaches.

1. Using variable length chunks increases the system robustness against chunk losses by minimizing the error propagation. Moreover, variable length chunks allows the P2P engine to determine the duration of ready-to-play chunk buffer accurately and can perform rate adaptation with higher precision.
2. Using variable length scheduling window is an effective method to introduce randomness among different peers in chunk scheduling procedure. Doing so, it is more likely that peer would have distinct chunks that they can exchange with their neighbors. This approach increases the rate of chunk exchange among peer and implicitly decreases the number of chunks that are scheduled from the seed server.

3. Using an incentive mechanism that allows the peers to use past statistics of the neighbor in optimistic unchoking policy decreases the initial buffering duration for peers that have good sharing history. The approach creates further incentive to share content with other peers in the session even if the peer can completed watching it.

The above findings are about the streaming monoscopic video over P2P networks. In the following, we provide our key results regarding the 3D video delivery over P2P networks.

5.2.2 QoE Aware Rate Scaling for Stereoscopic 3D

1. Using asymmetric coding increases perceived quality of experience when rate adaptation is performed if the level of asymmetric is kept within a certain bound and the PSNR value of the low-quality pair is about a certain threshold (~32dB)
2. When the above condition is not met, it is best to perform symmetric rate scaling to minimize the effect of adaption on the perceived video quality.

5.2.3 QoE Aware Rate Scaling for Stereoscopic Multi-view Video

1. Depending on the available bitrate, there are various adaption options that minimize the effect of rate scaling on the perceived video quality. For instance, if only a small amount of rate scaling is needed, performing asymmetric scaling between adjacent views provides the best result. However, if the available bitrate is significantly lower than the source rate then dropping intermediate views while keeping the edge views and the edge depth-maps may provide the best QoE. (In this case the missing views are interpolated at the client side.)
2. Rate scaling for multi-view video or stereoscopic 3D can become complex, requiring feedback from multiple sources. In order to keep modularity, it may be the best option to introduce the decision by an external module that can

better access the situation and provide only a ranking list to the P2P engine. Doing so, we isolate the QoE related decision making process from the chunk scheduling policy of the P2P engine.

5.3 Future Directions

5.3.1 P2P Video Streaming

We believe that the key issue in P2P video streaming systems is the backward compatibility with the BitTorrent protocol. We have proven that it is possible to increase the performance for video streaming and provide higher perceived quality if the boundaries of the BitTorrent protocol are extended such as fixing the chunk size and regarding every node equally, without distinguishing the content server from the other nodes, and tunes for multimedia content. Although, there are many studies that are proven to provide superior performance, other parameters such as users desire to continue to use the same protocol can have a bigger impact than the provided performance gain.

Next issue that would affect the future of P2P video streaming is tradeoff between the cost and service quality. Services like YouTube use server-client approach and earn huge profits despite their equally high bandwidth costs. An alternative P2P service, with all its complications, may not provide an efficient solution as well as server-client approach; however the cost would be significantly lower. Then, from the users' point of view, there are two services and both of them free. One of them has superior performance but displays more advertisements, the other (P2P) may have inferior performance does not need to push advertisements as much as the server-client solution which might attract some of the users.

Finally, it may be the case that these two extremes are coupled together as in the case of Project DIOMEDES. The seed server provides the content regardless of the peers' conditions however the P2P overlay has the case of decreasing bandwidth requirement when proper incentives are available.

5.3.2 3D Video Streaming

Broadcast of stereoscopic 3D media over digital TV platforms has already started. However, these platforms cannot provide sufficient bandwidth to broadcast multi-view video due to physical channel limitations. Hence, we foresee that, in the medium term, multi-view video services will be developed using the second method and these services will be deployed over the IP platform using various architectures, including server-client and peer-to-peer. In particular, project DIOMEDES addresses robust and flexible distribution of multi-view TV/video services using a combination of DVB and IP, where stereo DVB broadcast will be complemented by P2P streaming of the remaining views and the corresponding depth-maps to provide free-view TV experience.

Streaming of holographic 3D video over IP is projected in the long term since dynamic holographic display technology and compression methods for such data sources are not mature enough yet.

Bibliography

[1] C.G. Gurler, B. Gorkemli, G. Saygili, and M. Tekalp, "Flexible transport of 3D video over networks" in Proceedings of the IEEE, Vol. 99, No. 4, April 2011

[2] S.S. Savas, C.G. Gurler, A.M. Tekalp, "Adaptive multi-view video streaming over P2P networks considering quality of experience," ACM Multimedia Workshop on Social and Behavioral Networked Media Access, Arizona USA, November, 2011

[3] R.S.Prasad, M.Murray, C.Dovrolis, and K.C.Claffy "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," IEEE Network, 2003.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 3550, Jul. 2003.

[5] J. Lazzaro, "Framing RTP and RTCP packets over connection-oriented transport," RFC 4571, Jul. 2006.

[6] VideoLAN Client (VLC), online available: http://www.videolan.org/vlc/

[7] Darwin Streaming Server, online available: http://dss.macosforge.org/

[8] Microsoft Smooth Streaming, online available: http://www.iis.net/download/SmoothStreaming

[9] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Real-time system for adaptive video streaming based on SVC," *IEEE Trans. on Cirts. and Sys.s for Video Techn.*, vol.17, no.9, pp. 1227-1237, 2007.

[10] B. Gorkemli and A. M. Tekalp, "Adaptation strategies for streaming SVC video," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Hong Kong, Sep. 2010.

[11] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

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

[13] J. H. Jeon, S. C. Son, and J. S. Nam, "Overlay multicast tree recovery scheme using a proactive approach," *Computer Communications*, vol. 31, pp. 3163-3168, 2008.

[14] M. Fesci, E.T. Tunali, and A. M. Tekalp, "Bandwidth-aware multiple multicast tree formation for P2P scalable video streaming using hierarchical clusters," in *Proc. IEEE Int. Conf. on Image Processing (ICIP),* Cairo, Egypt, Nov. 2009.

[15] M. Castro, P. Druschel, A.-M. Kermarrec,A. Nandi, A. Rowstron, and A. Singh, "Split-stream: High-bandwidth content distribution in cooperative environments," in *Proc. Int. Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.

[16] P. Baccichet, J. Noh, E. Setton, and B. Girod, "Content-aware P2P video streaming with low latency," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, Beijing, China, Jul. 2007

[17] J. Noh, P. Baccichet, F. Hartung, A. Mavlankar, and B. Girod, "Stanford peer-to-peer multicast (SPPM) overview and recent extensions," in *Proc. Picture Coding Symposium (PCS),* invited paper, May 2009.

[18] P. P. Baccichet, T. Schierl, T. Wiegand, and B. Girod, "Low-delay Peer-to-Peer streaming using scalable video coding," in *Proc. the International Packet Video Workshop*, Nov. 2007.

[19] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Magazine*, pages 74–93, Sep. 2001.

[20] E. Setton, P. Baccichet, and B. Girod, "Peer-to-peer live multicast: A video perspective," Proc. IEEE, vol. 96, no. 1, pp. 25–38, Jan. 2008.

[21] V. N. Padmanabhan, H.J. Wang, and P.A Chou, "Resilient peer-to-peer streaming," in *Proc. IEEE Int. Conf. on Network Protocols (ICNP)*, 2003.

[22] D. Jurca, J. Chakareski, J.P. Wagner, and P. Frossard, "Enabling adaptive video streaming in P2P systems," *IEEE Comm. Mag.*, vol. 45, no. 6, pp. 108–114, Jun. 2007

[23] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, "The BitTorrent P2P file-sharing system: Measurements and analysis," in *Proc. of the 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 205-216, 2005.

[24] B. Cohen, "Incentives build robustness in BitTorrent," *Workshop on Economics of P2P systems*. Jun. 2003.

[25] S. Spoto, R. Gaeta, M. Grangetto, and M. Sereno, "Analysis of PPLive through active and passive measurements," in IEEE International Symposium on Parallel & Distributed Processing, 2009, pp. 1–7.

[26] PPLive Client. Online-available: http://pplive.en.softonic.com/

[27] C. Dana, D. Li, D. Harrison, and C. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *Proc. International Workshop on Multimedia Signal Processing (MMsP)*, 2005.

[28] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in Global Internet Workshop in conjunction with IEEE INFOCOM 2006, Apr. 2006

[29] Tribler [Online]. Available: http://www.tribler.org/

[30] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips, "Tribler: A social-based Peer-to-Peer system," in *the 5th Inter. Workshop on Peer-to-Peer Systems*, 2006.

[31] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips, "Give-to-Get: Free-riding-resilient video-on-demand in P2P systems," in *Proc. SPIE, Multimedia Computing and Network Conf. (MMCN),* Jan. 2008.

[32] N. Capovilla, M. Eberhard, S. Mignanti, R. Petrocco, and J. Vehkaperä, "An architecture for distributing scalable content over peer-to-peer networks," in Proc. of Int. Conf. on Advances in Mult. (MMEDIA'10), Athens, Greece, 2010

[33] M. Eberhard, T. Szkaliczki, H. Hellwagner, L. Szobonya, and C. Timmerer. “Knapsack problem-based piece-picking algorithms for layered content in peer-to-peer networks,” in Proc. of the 2010 ACM workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking, AVSTP2P , pg. 71-76, New York, 2010

[34] R. Petrocco, M. Eberhard, J. Pouwelse, and D. Epema, “Deftpack: A robust piece-picking algorithm for scalable video coding in p2p systems” in IEEE International Symposium on Multimedia (ISM), pp. 285-292, 2011

[35] Project P2PNext, NextShare Platform, online-available: http://www.p2p-next.org/

[36] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, “LayerP2P: Using layered video chunks in P2P live streaming,” IEEE Transactions on Multimedia, vol. 11, no. 7, pp. 1340-1352, Aug. 2009.

[37] A. Vetro, T. Wiegand, and G. Sullivan, “Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard,” in Proceedings of the IEEE, Vol. 99, No. 4, April 2011

[38] H. Schwarz, D. Marpe, and T. Wiegand, “Analysis of hierarchical B-Pictures and MCTF,” in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, pp. 1929–1932, Toronto, Ontario, Canada, Jul. 2006.

[39] Y. Chen, Y.-K.Wang, K. Ugur, M. Hannuksela, J. Lainema and M. Gabbouj, “The emerging MVC standard for 3D video services,” *EURASIP Jour. on Advances in Signal Processing*, vol. 2009, 2009.

[40] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 17, no. 11, pp. 1461-1473, Nov. 2007

[41] M. Drose, C. Clemens, and T. Sikora, "Extending single-view scalable video coding to multi-view based on H.264/AVC," in *Proc. IEEE Int. Conf. Image Proc.(ICIP)*, pp. 2977–2980, Atlanta, GA, USA, Oct. 2006.

[42] N. Ozbek and A. M. Tekalp, "Scalable multi-view video coding for interactive 3DTV," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, pp. 213–216, Toronto, Ontario, Canada, Jul. 2006.

[43] J. Garbas, U. Fecker, T. Tröger, and A. Kaup, "4D scalable multi-view video coding using disparity compensated view filtering and motion compensated temporal filtering," *Int. Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2006

[44] ATTEST [Online]. Available: http://www.hitech-projects.com/euprojects/attest/

[45] Text of ISO/IEC FDIS 23002-3 Representation of Auxiliary Video and Supplemental Information, ISO/IEC JTC1/SC29/WG11, Doc. N8768, Marrakesh, Morocco, Jan. 2007.

[46] *Text of ISO/IEC 13818-1:2003/FDAM2 Carriage of Auxiliary Data*, ISO/IEC JTC1/SC29/WG11, Doc. N8799, Marrakech, Morocco, Jan. 2007.

[47] S. Tao, Y. Chen, M. M. Hannuksela, Y.-K. Wang, M. Gabbouj and H. Li, “Joint texture and depth map video coding based on the scalable extension of H.264/AVC,” in *Proc. IEEE Int. Symp. on Cir. and Sys. (ISCAS)*, pp. 2253–2256, 2009.

[48] Cisco Visual Networking Index: Forecast and Methodology, 2009–2014, available online: http://large.stanford.edu/courses/2010/ph240/abdulkafi1/docs/white_paper _c11- 481360 .pdf

[49] Cisco Visual Networking Index, Forecast and Methodology, 2010–2015, available online: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ ns827/ VNI_Hyperconnectivity_WP.html

[50] C.G. Gurler, B. Gorkemli, G. Saygili, and M. Tekalp, "Flexible transport of 3D video over networks" in Proceedings of the IEEE, Vol. 99, No. 4, April 2011.

[51] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in Proc. IEEE INFOCOM'07, Anchorage, AK, May 2007.

[52] T. Stockhammer, 2011, "Dynamic Adaptive Streaming over HTTP Standards and Design Principles", ACM Multimedia Systems, San Jose, California, USA, Feb. 2011, pp. 133-143

[53] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. "Do Incentives Build Robustness in BitTorrent?," in Proc. of Networked System Design and Implementation, April 2007.

[54] M. Feldman and J. Chuang. "Overcoming free-riding behavior in peer-to-peer systems," ACM SIGECOM Exchanges, 5(4): pp. 41-50, 2005.

[55] C. G. Gurler, S. S. Savas and A. M. Tekalp, "Variable chunk size and adaptive scheduling window for P2P streaming of scalable video," in Proc. IEEE Int. Conf. Image Processing (ICIP), Orlando, Florida, Oct. 2012.

[56] A. Bakker et al. "Tribler Protocol Specification" http://svn.tribler.org/bt2-design/proto-spec-unified/trunk/proto-spec-current.pdf, 1 2009.

[57] Y. Xu, C. Zhu, W. Zeng, and X. J. Li, "Multiple description coded video streaming in peer-to-peer networks," Signal Processing: Image Communication, vol. 27, no. 5, pp. 412-429, 2012.

[58] Y. Chiang, P. Huang, and H. H. Chen, "SVC or MDC? That's the question," in Proc. IEEE ESTIMedia, pp. 76-82, Oct. 2011.

[59] G. Gurler, K. Bagci and A. M. Tekalp, "Adaptive stereoscopic 3D video streaming", in Proc. IEEE Int. Conf. Image Process.(ICIP), Hong Kong, Sept. 2010.

[60] R. Penno, S. Raghunath, R. Woundy, V. Gurbani, J. Touch "LEDBAT Practices and Recommendations for Managing Multiple Concurrent TCP Connections", IETF draft, Feb. 26,2000

[61] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, The Peersim simulator, http://peersim.sourceforge.net

[62] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp. 1194–1203, Sep. 2007.

[63] Commercial Requirements for DVB 3D-TV, DVB BlueBook A151, July 2010.

[64] HDMI specification 1.4a, HDMI Licensing, LLC., Mar. 2010.

[65] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, “Multi-view video plus depth representation and coding,” in IEEE Int. Conf. on Image Process. (ICIP), San Antonio, TX, Sep. 2007.

[66] A. Smolic, K. Muller, N. Stefanoski, J. Osterman, A. Gotchev, G.B. Akar, G. Triantafyllidis, and A. Koz, “Coding algorithms for 3DTV – A survey,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, pp. 1606-1621, Nov. 2007.

[67] P. Merkle, K. Müller, A. Smolic, and T. Wiegand, “Efficient compression of multiview depth data based on MVC,” in Proc. 3DTV Conference, Kos, Greece, May 2007.

[68] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, “Depth map creation and image based rendering for advanced 3DTV services providing interoperability and scalability,” Signal Processing: Image Comm., Special Issue on 3DTV, Feb. 2007.

[69] J. Shade, S. Gortler, L. Hey, and R. Szeliski, “Layered depth images,” in Proc. of ACM SIGGRAPH, Orlando, Florida, 1998, pp.231–242.

[70] M.G. Perkins, “Data compression of stereo pairs,” IEEE Trans. on Communications, vol. 40, no. 4, pp.684-696, April 1992.

[71] W. J. Tam, L. B. Stelmach, P. Corriveau, “Psychovisual aspects of viewing stereoscopic video sequences,” Electronic Imaging 98,San Jose, CA, pp. 24-30, January 1998.

[72] L. B. Stelamch, W. J. Tam, “Stereoscopic image coding: effect of disparate image-quality in left- and right-eye views,” Signal Processing Image Commun., vol. 14, pp. 111-117, 1998

[73] P. Seuntiëns, L. Meesters, W. Ijsselsteijn, “Perceived quality of compressed stereoscopic images: effects of symmetric and asymmetric JPEG coding and camera separation,” ACM. Trans. Appl. Perception 3 (2), pp. 95-109, 2006.

[74] P. Seuntiëns, L. Meesters, W. Ijsselsteijn, “Perceptual evaluation of JPEG coded stereoscopic images,” Proceedings of the SPIE 5006, pp. 215-226, 2003.

[75] L. M. J. Meesters, W. A. Ijsselsteijn, P. J. H. Seuntiëns, “A survey of perceptual evaluations and requirements of three- dimensional TV,” IEEE Trans. Circuits Syst. Video Techn.. 14, pp. 381-391, 2004.

[76] A. Boev, M. Poikela, A. Gotchev and A. Aksay, “Modeling of the stereoscopic HVS,” Mobile3DTV Project report, D5.1, 2008.

[77] C. Fehn, P. Kauff, S. Cho, H. Kwon, N. Hur, and J. Kim, “Asymmetric coding of stereoscopic video for transmission over T-DMB,” in Proc. 3DTV-CON, Kos, Greece, May 2007.

[78] H. Kalva, L. Christodoulou, L. M. Mayron, O. Marques, and B. Furht, “Design and evaluation of a 3D video system based on H.264 view coding,” Proceedings of NOSSDAV, pp. 68-73,Newport, Rhode Island, May 22-23, 2006.

[79] ITU-R Rec.BT.500-11, “Methodology for subjective assessment of the quality of television pictures,” 2002.

[80] N. Ozbek, A. M. Tekalp and T. Tunali, “Rate allocation between views in scalable stereo video coding using an objective stereo video quality measure,” IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2007.

[81] T. C. Thang, J. Kim, J. W. Kang, J. Yoo, “SVC adaptation: Standard tools and supporting methods,” Signal Processing: Image Communication vol. 24, issue 3, pp. 214-228, March 2009.

[82] G. Gurler, T. Bagci, and A. M. Tekalp, “Adaptive stereoscopic 3D video streaming,” in Proc. IEEE Int. Conf. on Image Processing (ICIP), Hong Kong, Sep. 2010.

[83] G. Saygili, G. Gurler, A. M. Tekalp, “Quality assessment of asymmetric stereo video coding,” in Proc. IEEE Int. Conf. on Image Processing (ICIP), Hong Kong, Sep. 2010.

[84] G. Saygili, G. Gurler, A. M. Tekalp, “3D display-dependent quality evaluation and rate allocation using scalable video coding,” IEEE Int. Con. Image Processing (ICIP), Egypt, pp. 717-720, 2009.

[85] J.Bolot, S. Fosse-Parisis, D. Towsley “Adaptive FEC based Error Control for Interactive Audio in the Internet”, Infocom,San Francisco, CA,1998

[86] S. Wenger, “Error patterns for Internet experiments,” ITU Telecommunications Standardization Sector, Doc. Q15-I-16r1, Oct. 1999

[87] S. Savas, C. Gürler, A.M. Tekalp, E. Ekmekcioglu, S. Worrall and Ahmet Kondoz, "Adaptive streaming of multi-view video over P2P networks," Signal Processing: Image Communication (February 2012), doi:10.1016/j.image.2012.02.013.

[88] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, “Reference Softwares for Depth Estimation and View Synthesis”, ISO/IEC JTC1/SC29/WG11, M15377, April 2008.

[89] G. Saygili, G. Gurler, A. M. Tekalp, “3D display-dependent quality evaluation and rate allocation using scalable video coding,” IEEE Int. Con. Image Processing (ICIP), Egypt, pp. 717-720, 2009.

[90] Project DIOMEDES Deliverable: “Report on the Quality of Experience model and the audio and visual attention models,” online available: http://www.diomedes-project.eu/deliverables/DIOMEDES_D3.4_FINAL.pdf

[91] K. Aydogdu, E. Dogan, H. Gokmen, S. S. Savas, C. G. Gurler, J. Hasselbach, T. Adari, "DIOMEDES: Content Aware and Adaptive Delivery of 3D Media over P2P/IP and DVB-T2," NEM Summit 2011, Implementing future media Internet, Torino Italy, September 2011.

[92] Project DIOMEDES, Distribution of Multi-view Entertainment Using Content-Aware Delivery Systems, online available: http://www.diomedes-project.eu/

[93] Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A.; , "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp.560-576, July 2003

[94] X. Hei, Y. Liu, and K. W. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," JSAC, vol.25, no. 10, Dec. 2007.