# P2P LIVE SCALABLE VIDEO STREAMING WITH ALTO SERVICE

A Thesis

by

İ. Serkan Kırkgül

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Electrical and Electronics Engineering

Özyeğin University
October 2012

# P2P LIVE SCALABLE VIDEO STREAMING WITH ALTO SERVICE

Approved by:

Assoc. Prof. Dr. M. Oğuz Sunay,
Advisor
Department of Electrical and
Electronics Engineering
*Özyeğin University*

Assist. Prof. Dr. İsmail Arı
Department of Computer Science
*Özyeğin University*

Prof. Dr. Tanju Erdem
Department of Electrical and
Electronics Engineering
*Özyeğin University*

Assist. Prof. Dr. Ali Özer Ercan
Department of Electrical and
Electronics Engineering
*Özyeğin University*

Date Approved: 22 October 2012

Dr. A. Serdar Tan

*Türk Telekom Ar-Ge*

*To my mother*

# ABSTRACT

It has been measured that, as high as 80% of the average Internet traffic is now from peer-to-peer (P2P) applications, and upwards of 90% during peak hours. These results tell us that a significant portion of the traffic that ISPs route is P2P traffic. However, since the typical P2P application overlay network model is network-oblivious, due to its nature, it causes inefficient network utilization, and low quality of experience, from the perspective of the network providers and the end users, respectively. This inefficient resource utilization increases inter-ISP traffic, and leads to serious disruptions on ISP economics as well as lower QoE by the users.

A significant portion of the P2P traffic on the Internet is for video applications. Furthermore, real networks are heterogeneous in link rates. Thus a live video streaming service should support streams with different qualities for users using different data rates. A possible solution to this problem is achieved by creating a different video file for each quality level. However, this solution is inefficient due to data duplication.

To improve ISP economics, improve network efficiency and achieve acceptable QoE performance, we implement a P2P Live Scalable Video Streaming Mechanism and Application Layer Traffic Optimization (ALTO) Server, by using different peer ranking algorithms in this thesis. The main objective of this thesis, to show that the ALTO protocol reduces the inter-ISP traffic significantly while maintaining the application performance. Results reveal that, placing an ALTO Server decreases the inter-ISP traffic dramatically while maintaining a satisfactory QoE performance for the streaming video application.

# ÖZETÇE

Yapılan ölçümlere göre, günümüzde ortalama İnternet trafiğinin %80'i gibi büyük bir kısmı eş görevli ağ (P2P) uygulamalarından gelmektedir ve bu değer en yoğun saatlerde %90'ı aşmaktadır. Bu nedenle İnternet servis sağlayıcılarının yönlendirdiği trafiğin önemli bir kısmı P2P trafiğidir. Ancak P2P uygulaması ağ modeli, yapısı gereği fiziksel ağ topolojisinden habersiz olduğu için, ağ kaynaklarının verimsiz kullanılmasına ve kullanıcıların düşük hizmet kalitesi almasına sebep olmaktadır. Bu gereksiz kaynak kullanımı kullanıcıların aldığı hizmetin kalitesini düşürmekle birlikte, servis sağlayıcılar arası trafiği arttırmakta ve servis sağlayıcıların ekonomilerini büyük ölçüde ve olumsuz yönde etkilemektedir.

İnternet'teki P2P trafiğinin önemli bir kısmı video uygulamalarına aittir. Ayrıca, gerçek ağlar hatların hızları bakımından türdeş olmayan bir yapıya sahiptir. Bu nedenle gerçek zamanlı bir video akıtımı servisinin değişik hızlara sahip kullanıcılar için değişik kalitelerde videolar sunması gereklidir. Her bir kalite seviyesi için ayrı bir video dosyası oluşturmak bu soruna bir çözüm olabilir. Ancak, bu çözüm gereksiz veri tekrarı nedeniyle verimsizdir.

Bu tezde servis sağlayıcıların ekonomilerini geliştirmek, ağ verimliliğini arttırmak ve kabul edilebilir hizmet kalitesi sunmak için bir P2P Gerçek Zamanlı Ölçeklenebilir Video Akıtımı Mekanizması ve Uygulama Katmanı Trafik Eniyileme (ALTO) Sunucusunu farklı eş sıralama algoritmaları kullanarak hayata geçirdik. Bu tezin ana hedefi, ALTO protokolünün uygulama başarımını korurken, servis sağlayıcılar arasındaki trafiği önemli ölçüde azalttığını göstermektir. Aldığımız sonuçlara göre, sisteme bir ALTO sunucusu yerleştirmek, video akıtımı uygulaması için tatmin edici hizmet kalitesi sunarken, servis sağlayıcılar arasındaki trafiği önemli ölçüde azaltmaktadır.

# ACKNOWLEDGEMENTS

First, I would like to thank my thesis supervisor Assoc. Prof. M. Oğuz Sunay for his guidance and support during my research. I would not be able to complete this thesis without his guidance.

I also would like to thank my friend and project partner Koray Kökten for his support and friendship. I feel lucky to work with such a good person and colleague. Also I would like to thank my friends in Wireless Information Systems Engineering Research Laboratory Devin Mungan, Gülden Ferazoğlu, Volkan Yazıcı and Erdem Ulusoy for their help.

Finally, I would like to express my deep love for my family. Without them, I would not be the person I am today.

# TABLE OF CONTENTS

**DEDICATION** . . . . . **iii**

**ABSTRACT** . . . . . **iv**

**ÖZETÇE** . . . . . **v**

**ACKNOWLEDGEMENTS** . . . . . **vi**

**LIST OF TABLES** . . . . . **x**

**LIST OF FIGURES** . . . . . **xi**

**I INTRODUCTION** . . . . . **1**

1.1 From Yesterday to Today: Internet . . . . . 1

1.1.1 The Early Days of Internet . . . . . 1

1.1.2 World Wide Web . . . . . 2

1.1.3 Client-Server Model . . . . . 3

1.1.4 CDN Model . . . . . 5

1.1.5 P2P Model . . . . . 6

1.2 P2P Live Video Streaming . . . . . 10

1.2.1 Overlay Construction Techniques . . . . . 11

1.2.2 Challenges . . . . . 14

1.3 P4P: Provider Portal for Applications . . . . . 14

1.4 Contributions of the Thesis . . . . . 17

1.5 Summary of the Thesis . . . . . 17

**II LITERATURE SURVEY** . . . . . **18**

2.1 CoolStreaming/DONet . . . . . 18

2.2 AnySee . . . . . 19

2.3 SplitStream . . . . . 21

2.4 Chainsaw . . . . . 22

2.5 PPLive . . . . . 23

2.6 PeerCast . . . . . 24
2.7 End System Multicast (ESM) . . . . . 25
**III P2P LIVE SCALABLE VIDEO STREAMING PROTOCOL . . 27**
3.1 Overlay Construction . . . . . 27
3.2 Joining and Leaving . . . . . 28
3.3 Partnership Management . . . . . 29
3.4 Video Coding Model . . . . . 32
3.4.1 Buffers . . . . . 33
3.4.2 Play and Pause Mechanisms . . . . . 33
3.4.3 Buffer Map Representation and Exchange . . . . . 34
3.5 Scheduling Algorithm . . . . . 34
**IV APPLICATION LAYER TRAFFIC OPTIMIZATION PROTOCOL 40**
4.1 Protocol Description . . . . . 40
4.1.1 Benefits . . . . . 40
4.1.2 Important Terms . . . . . 41
4.1.3 Protocol Structure . . . . . 41
4.1.4 Map Service . . . . . 42
4.2 Enabling ALTO Service in Our Video Streaming Protocol . . . . . 44
4.2.1 Changes in the Streaming Protocol . . . . . 44
4.2.2 Proposed Cost Calculation Methods . . . . . 45
**V PERFORMANCE EVALUATION . . . . . 48**
5.1 Simulation Setup . . . . . 48
5.1.1 Network Model . . . . . 48
5.1.2 Test Video . . . . . 50
5.2 Simulation Results . . . . . 54
5.2.1 Determination of Parameters . . . . . 54
5.2.2 Effectiveness of ALTO . . . . . 66
**VI CONCLUSION . . . . . 74**

**VII FUTURE WORK** . . . . . **75**

**REFERENCES** . . . . . **76**

**VITA** . . . . . **79**

# LIST OF TABLES


1 Descriptions of Some of the popular P2P Protocols . . . . . 9

2 Definitions of some terms in the scheduling algorithm . . . . . 38

3 Bandwidth Distribution of Peers . . . . . 49

4 Number of Peers with Different Hop Distances . . . . . 49

5 General encoding parameters of the test video. . . . . 51

6 Encoding parameters of the base layer. . . . . 52

7 Encoding parameters of the enhancement layer. . . . . 52

8 Simulation results for the two optimum operation points . . . . . 59

9 PSNR Value Statistics . . . . . 70

10 Time Averages of Inter-ISP Traffic Rates . . . . . 71

# LIST OF FIGURES


1 Logical Overlay of Client-Server Model . . . . . 4

2 Overview of CDN model . . . . . 6

3 Logical Overlay of P2P Model . . . . . 7

4 Measured Internet Traffic rates According to Different Applications (1993 - 2006) [1]. . . . . 8

5 Multiple Tree Overlays . . . . . 12

6 Diffusion and Swarming Phases in Mesh Method . . . . . 13

7 iTracker interfaces and information flow. . . . . 15

8 An example of P2P obtaining network information from iTrackers. . . 16

9 CoolStreaming system diagram . . . . . 18

10 System diagram of an AnySee node . . . . . 20

11 A simple example illustrating the basic approach of SplitStream. . . . 21

12 Basic overlay of PPLive. . . . . 24

13 Architecture of PeerCast system. . . . . 24

14 Architecture of ESM system. . . . . 26

15 Mesh overlay of the protocol. . . . . 28

16 Video structure. . . . . 32

17 Operation regions for the scheduling algorithm . . . . . 36

18 ALTO Service Framework. . . . . 42

19 An example Network Map. . . . . 43

20 Proposed Network Model . . . . . 49

21 A snapshot taken from the test video. . . . . 50

22 Rate distortion curve of the encoded video . . . . . 53

23 Average pause number of peers for different scheduler parameters . . 54

24 Average pause duration of peers for different scheduler parameters . . 55

25 Average starting delay of peers for different scheduler parameters . . 56

26 Average PSNR value of peers for different scheduler parameters . . . 57

27 Illustration of operation points in 3D space of simulation results (z-axis shows average number of pauses) . . . . . 58
28 Illustration of operation points in 3D space of simulation results (z-axis shows average duration of pauses) . . . . . 58
29 Sensitivity of average pause duration of the peers to the buffer length 60
30 Sensitivity of average pre-roll delay of the peers to the buffer length . 61
31 Sensitivity of average PSNR value of the peers to the buffer length . 61
32 Sensitivity of average pause duration of the peers to the scheduling period . . . . . 62
33 Sensitivity of average pre-roll delay of the peers to the scheduling period . . . . . 62
34 Sensitivity of average PSNR value of the peers to the scheduling period 63
35 Average pause number of peers for varying maximum partner number 64
36 Average pause duration of peers for varying maximum partner number 64
37 Average pre-roll delay of peers for varying maximum partner number 65
38 Average PSNR value of peers for varying maximum partner number 65
39 Average Video Download Rates of Peers . . . . . 67
40 Average Video Upload Rates of Peers . . . . . 67
41 CDF of Streaming Starting Times of the Peers . . . . . 68
42 CDF of the Pause Durations of the Peers . . . . . 69
43 CDF of Streaming Completion Times of the Peers . . . . . 70
44 Inter-ISP traffic between ISP 0 and ISP 1 . . . . . 71
45 Inter-ISP traffic between ISP 0 and ISP 2 . . . . . 72
46 Inter-ISP traffic between ISP 1 and ISP 2 . . . . . 72

# CHAPTER I

# INTRODUCTION

In this introductory chapter we briefly summarize the evolution of P2P applications over the Internet. We then provide the contributions of this thesis along with a general summary.

## *1.1 From Yesterday to Today: Internet*

Over the last few decades, Internet has been altering the way people live and communicate with one other. To a large extent, the popularity of the Internet is due to the wide range and ease of content distribution. Since the start of the World Wide Web, collaboration and interacting between users through their networked computers is still one of the main applications of the Internet

### 1.1.1 The Early Days of Internet

Initially, the Internet was not designed for public usage. When research started on the ARPANET project, it was intended for the scientific collaboration and the associated file transferring between two or more computers that have long distances between them. Electronic distribution of resources between users soon gained importance as it became apparent that the traditional academic publication process was too slow for the fast-paced information exchange essential for creating the Internet [2]. As a solution, a number of protocols were defined e.g. FTP. Over the years FTP evolved into the primary means for content retrieval and software distribution over the Internet. However, FTP alone cannot solve content retrieval problem. This is because, if somebody wants to access to a given content on a given server, he has to know the identity of the exact server, on which the content exists. In order to overcome this

problem, some alternate solutions were offered (Archie, Wide Area Information Server (WAIS), Gopher..). However these protocols all have certain disadvantages. For instance, Archie's work mechanism depends on scanning all the servers which contains various types of contents periodically and saving only the names of documents. Thus, anyone who seeks for a document can search the exact server via the Archie platform. The main handicap of Archie is its limitation to pattern matching on file names rather than the actual content of the document [2]. So, in order to solve this problem, WAIS was implemented as a powerful search protocol through the documents in addition to their names and authors or etc. But its user interface was quite hard to use. Therefore few people were used it.

Archie, WAIS and the others emerged around the same period. Soon afterwards, they all were subsumed into another system, the World Wide Web (www).

### 1.1.2 World Wide Web

The World Wide Web is an Internet opportunity that links information accessible via networked computers. This information can be made possible in the form of web pages, which can contain text documents, video or etc., by embedding hyperlinks to all content. A user can reach all the content by following the embedded hyperlinks to each content.

The World Wide Web was born in the Center of European Organization for Nuclear Research (CERN) in 1989. It was implemented by Tim Berners-Lee, to develop the information access between scientists who were several hundreds of people at the time [3]. The increase in the number of users led to the use of hypertext to link accessible content on different computers. Previously the hypertext notion had been designed for making computers respond and require information like humans. Hypertext files contains hyperlinks. A hyperlink can be denoted as underlined text or as icons. Corresponding information can be loaded and displayed simply by clicking

on its hyperlink.

Tim's proposal was approved by the management and the project was launched in the second half of 1990. Tim started developing a hypertext browser/editor and the first version of the program was released at the end of 1990. The program provided a graphical user interface. Simultaneously, CERN student Nicola Pellow wrote a separate line-mode browser. It was followed by the implementation of browsers on different platforms. In 1992, first versions of Erwise, ViolaWWW, and MidasWWW were released for the X/Motif system and in 1993 CERN implemented a browser for Apple Macintosh. At the time, there were about 50 known deployed Web servers and only 0.1% of the Internet traffic was from the WWW. It was promising, but the real breakthrough was experienced with Mosaic which was the first widespread Web browser with a graphical interface. Mosaic was developed at the National Center for Supercomputing Applications (NCSA) by Marc Andreesen and Eric Bina.

By 1994, Andreesen and Bina found Netscape Communications Corporation which was the birthplace of famous Netscape browser family. The Web became more popular and the number of the Web sites increased to 10000 by the beginning of 1995. Netscape's popularity also increased quickly and by 1996 around 75% of the WWW users used Netscape. Realizing Netscape's great success, Microsoft Corporation developed its own browser which was known as Internet Explorer. After the initial success of Netscape, Microsoft overwhelmed all its competitors in the end. According to statistics [4] in July 2004, around 80-90% of the browsers used on the Internet were developed by Microsoft. By 2012 a fair number of alternatives became prominent and Internet Explorer lost its earlier dominance.

### 1.1.3 Client-Server Model

The World Wide Web information offers people huge linked information accessible via the Internet. The information is provided in the Web page form or in general Web

objects. Web objects are available in high capability computers, which are called Web Servers. A web object can be anything, from a simple text document to a live video. The application which sends request to the Web server is known as the Web client. The client sends a request and receives a response which includes the requested web object or an error message back from the server.

This scenario creates the fundamental structure of the Internet, called the client-server model. Every entity has a strictly defined role in this scenario. Servers always answer the requests that they receive from the clients, while clients have only the ability to send requests and get answers from the corresponding servers. The logical topology of this model is shown below in Figure 1.

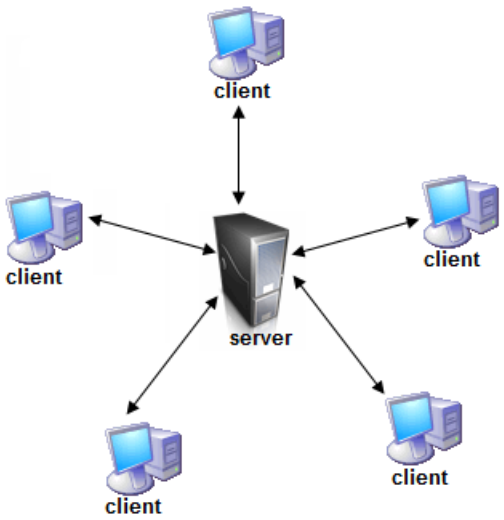


Figure 1: Logical Overlay of Client-Server Model

The Internet is highly decentralized and distributed. However in Figure 1 we have just illustrated a strictly centralized model. This contradiction causes serious scalability problems. For instance in this model all of the requests are handled by a particular web server. If the number requests and the corresponding load to the server are not high, most probably no errors and problems will occur. However, whenever the number of the requests increases, then the server capacity can be easily overwhelmed.

Also there is another potential problem related to the link capacity. Even when the processing capacity is sufficient of the server to handle all the requests, the link capacity which connects the server to the world, may be also easily overwhelmed. These kinds of bottlenecks make the client-server inefficient. In order to solve these bottlenecks and increase the efficiency of this topology, some promising solutions have been developed. Content Delivery Network (CDN) is the most popular and widely used solution [5].

### 1.1.4 CDN Model

A CDN is a large system consisting of servers placed at multiple locations all over the world. CDNs consist of origin servers that are actual source of the content and surrogate servers located close to users that store copies of the original content [6]. In practice these surrogate servers are placed strategically at data centers in order to maintain worldwide distributed copies of the original content.

There are three components in a CDN architecture: the content provider, the CDN provider and clients [7]. A content provider is one who is the owner of content to be distributed. The content is stored in the origin server of the content provider. A CDN provider is a proprietary company or organization which facilitates the required infrastructure to content providers to deliver content reliably and quickly. Clients are entities that retrieve content from the content provider. Figure 2 illustrates an overview of the CDN.

In a CDN, the typical client-server communication is replaced by two flows: one between the client and the surrogate server, and another between the surrogate server and the origin server. This modification reduces congestion over popular servers and increases content availability. Furthermore, CDNs reduce the content provider's need to invest on web site infrastructure and prevents traffic jams on the web since the requested data is closer to the user. CDNs also reduce load on the origin servers.

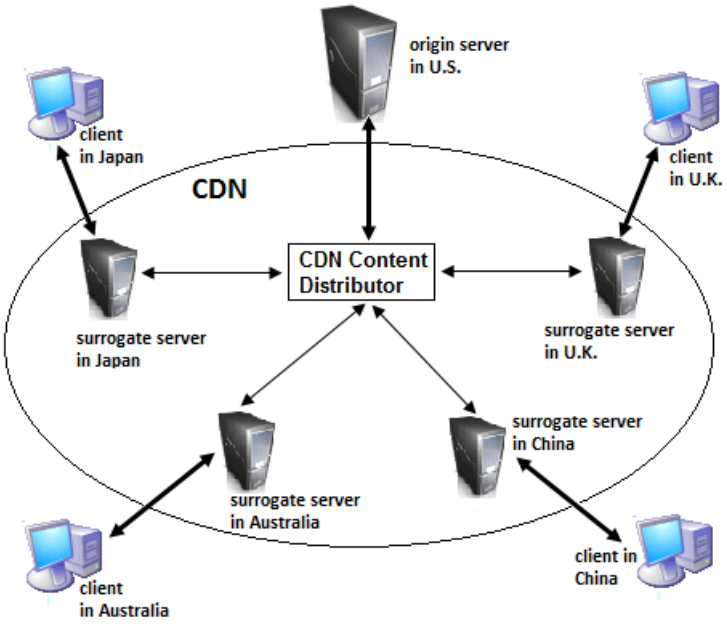


Figure 2: Overview of CDN model

Beside these advantages, it has also challenges like high costs, support, maintenance and verification of the best locations for the surrogate servers.

There are many commercial CDNs like Akamai [8], Mirror Image [9], Limelight Networks [10] etc. as well as academic CDNs such as Coral [11], CoDeeN [12], Globule [13] etc. in today's Internet.

### 1.1.5 P2P Model

In recent years, many important applications have appeared on the Internet that use a decentralized architecture to simultaneously connect millions of clients to one other, and make it possible to communicate and retrieve content. These kind of applications are called peer-to-peer (P2P) since they eliminate the central servers that mediate between the users. Also, their network behavior is defined as an overlay network where applications form a virtual network over the real topology. A brief formal definition of P2P networking is *"a set of technologies that enable the direct exchange of services or data between computers"*[2]

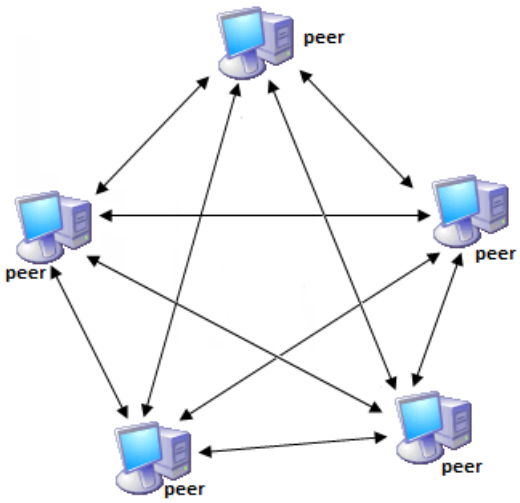


Figure 3: Logical Overlay of P2P Model

The most significant difference between P2P Networking and the Client-Server Networking is, P2P is a decentralized network topology. In a P2P topology a peer can serve as a server to upload content, and it can also request for content from the overlay network it is in simultaneously. In P2P networking architecture, peers do not have to connect a central server to start a communication session to one or more peers, as shown in Figure 3. Moreover, P2P Networks are self-organizing and self-scalable.

#### *1.1.5.1 Strengths and Benefits*

The P2P overlay is a collection of distributed networked peers whose resources are available during the online times of the peers. These resources can be computation, storage capacity, processing capacity, bandwidth etc. While the peers are connected to the overlay, each peer shares the cost of operating the overlay. This sharing makes participating in a P2P overlay easier. So this means, modest hardware and network requirements are quite sufficient, which can be easily satisfied almost by every end user.

As discussed earlier, P2P systems are scalable. However, it was discovered that

not all peers have the same capacity to contribute to the overlay. For example, a firewall can make it difficult to participate in the overlay or the host might have lower CPU speed or memory capacity than other hosts in the overlay.

The distributed nature of P2P networks also increases robustness, enabling peers to find the data without relying on a centralized index server. In other words, there is no single point of failure in the system. Some of the other strengths and open issues will be mentioned in the latter sections, especially related to live video streaming.

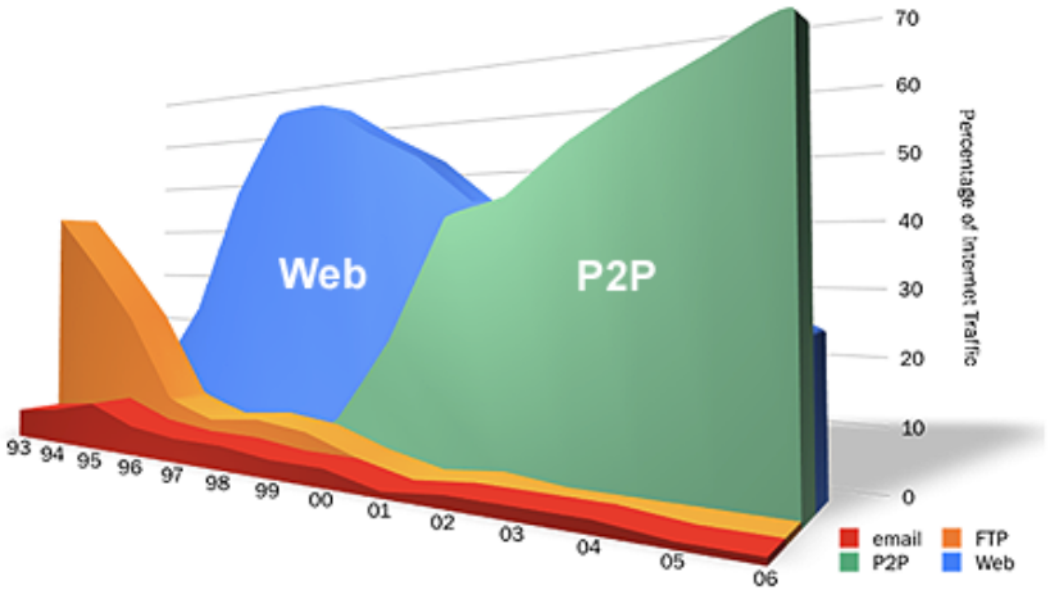


Figure 4: Measured Internet Traffic rates According to Different Applications (1993 - 2006) [1].

The growing popularity of P2P applications especially in recent years, have been changing the total traffic rates of applications on Internet. As can be seen in Figure 4, it was reported that, over the 80% of the average Internet traffic is created by the P2P applications [1].

The total P2P traffic on the Internet is generated by different types of applications. In the following sections, some of these applications are discussed.

#### *1.1.5.2 File Sharing*

Nearly ten years after the World Wide Web, the first P2P-Server based application Napster was developed. This application enabled its users to share their files and popularized the concept of file sharing. However, Napster's central server based service algorithm was a vulnerable point, since it violated the copyright laws. After a short time Napster was found liable and its services were forced down. After Napster shut down, people started to seek alternative protocols for file sharing. Hence, second generation protocols such as Gnutella, FastTrack and BitTorrent were developed. In these protocols there is no need for a central directory to establish an overlay and all the file searches and file transfers are shared among the corresponding peers. The majority of these protocols are based on unstructured overlays. A summary of some of the popular second generation P2P applications is given in Table 1. below [14].

Table 1: Descriptions of Some of the popular P2P Protocols

| **Client app(s)** | **Protocol** | **Description** |
|---|---|---|
| KaZaA | FastTrack | Proprietary unstructured overlay with encrypted protocol, high capacity peers become superpeers; features connection shuffling |
| Limewire | Gnutella | Superpeer unstructured overlay with flooding query propagation |
| eDonkey | Kademlia | Structured overlay based on Kademlia |
| eMule | Kad | Structured overlay based on Kademlia |
| BitTorrent | BitTorrent | An unstructured overlay used for distributing large files in pieces using mutual distribution of the pieces between a set of peers called a swarm. Uses a server to store the torrent and another server called a tracker to identify the swarm members. |

Typically in a P2P file-sharing application, in order to share files the user registers these files using the client application based on their properties such as title, artist, date and format. After the initial registration, other users can search for these files by sending a query with some combinations of the these properties to other online peers

in the network. Peers having files that match the query return information about how to obtain the files. The queries can also be forwarded to other peers. Users can get multiple positive responses and then select the files they want to obtain as well as the peers they wish to connect.

#### *1.1.5.3 Voice over P2P (VoP2P)*

Skype, the first big scale voice-over P2P(VoP2P) application, was developed by the founders of KaZaA. Currently Skype has around 700 million users. Furthermore, several services including P2P voice and video calls, voice calls to PSTN endpoints, presence, and instant messaging are available in Skype. Skype is a proprietary application like KaZaA. When it was first deployed, Skype used a superpeer model, and the superpeers were used for connecting peers behind NATs [15]. In addition, superpeers were also used as media relays. Skype no longer uses P2P overlays for its services.

## *1.2 P2P Live Video Streaming*

The successes of large-scale P2P file sharing and VoP2P applications have motivated the use of P2P for live video streaming (P2PTV). However file sharing and video streaming are very different in their nature. Unlike file-sharing, where systems firstly download the complete file and play it locally, P2PTV systems must provide real-time stream transfer rate to each peer that equal video playback rates.

The live video streaming application over Internet is quite popular in today's Internet. There are several CDN based services available for customers, e.g. Tivibu. However, as stated earlier, CDN based applications are not truly cost effective, so the P2PTV technology seems like a promising approach. Since it is a relatively new research topic, there are several open issues and challenges regarding the design rationales that must be solved. The concept of P2PTV was made popular by Chu et. al [16], who suggested taking advantage of the resources of the users to form

a dynamic delivery network. This idea is reasonable as it does not need any extra infrastructure and is, in theory self-scalable as explained before. Even though, this field is in its early stages, it has become very active research area, and various types of approaches have been proposed. Along with these approaches, many applications have appeared on the Internet, such as PPLive, PPStream, Zattoo, Coolstreaming, etc. [17, 18, 19].

### 1.2.1 Overlay Construction Techniques

One of the key aspects in a P2P live video streaming system is its overlay construction technique. In literature, based on the overlay construction technique employed, the proposed approaches may be divided into two main subclasses; tree-based and mesh-based.

#### *1.2.1.1 Tree-Based P2P Streaming*

In tree-based approaches, participating peers are divided into multiple trees by an overlay construction algorithm [20, 21]. Each peer selects a number of trees to join according to some parameters that are configured by the designer. To minimize the effects of churning[1]and effectively utilizing the network resources, peers are organized into multiple diverse trees as illustrated in Figure 5. At this stage, each peer most generally is chosen to be an internal node in only a diverse tree, and a leaf node, in all the other diverse trees. Then, each part of encoded content - can be a description of a Multiple Description Coding (MDC)[2]encoded content or a layer of a Scalable Video Coding (SVC) encoded content - is sent through a specific tree. As a result, the main focus of these types of designs is the tree construction method.

[1]**Churning** means in P2P Networks, leaving of the peers from the overlay randomly in an undetected fashion

[2]**Multiple Description Coding** is a coding technique to divide a single media stream into k substreams ($k > 1$) and sending them through different routed paths. In order to decode the stream, any substream can be used(unlike scalable video coding [22]). The quality of the stream that a customer experienced increases with the number of receiving substreams.

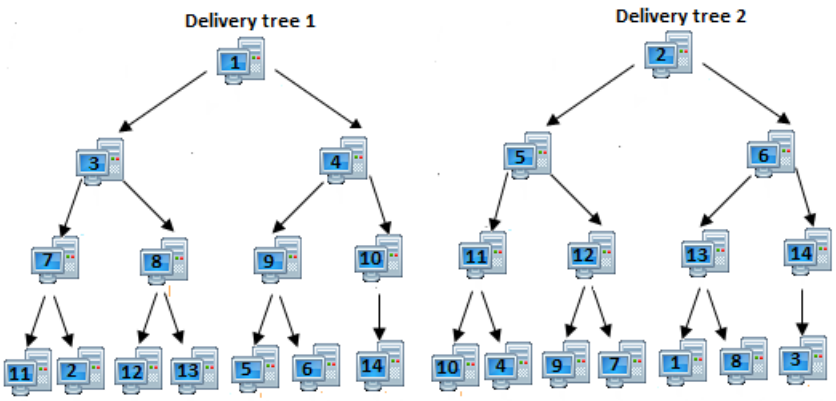


Figure 5: Multiple Tree Overlays

#### *1.2.1.2 Mesh-Based P2P Streaming*

In mesh based approaches, participating peers form a randomly connected overlay, referred to as a mesh [23, 24]. According to this approach, each peer tries to serve a list of members, which are randomly selected. There are some obvious limitations e.g. how many nodes a peer may have. This number is called generally the outgoing degree. Also each peer has a specific number of peers that provide the requested content, commonly referred to as the parents. The number of these peers are called the incoming degree. After connection to the network, any peer contacts a bootstrapping node to receive a set of peers that it can potentially connect and start receiving the streaming data. Then the bootstrapping node recommends any peer a diverse set of children. In the mesh topology there are 2 different phases in order to maximize the utilization of outgoing bandwidth. These phases are illustrated in Figure 6. In this figure, peers are divided into levels based on their shortest distance to the source in terms of hops. The two phases are:

*a)Diffusion Phase:* During this phase, as shown in Figure 6, each packet available at the source, is pulled bye one of the peers which is at level 1, and then by all of the peers that are the children of the level 1 peer [25]. Since each packet is delivered only once from the source, the subset of of peers that receive a packet during its diffusion

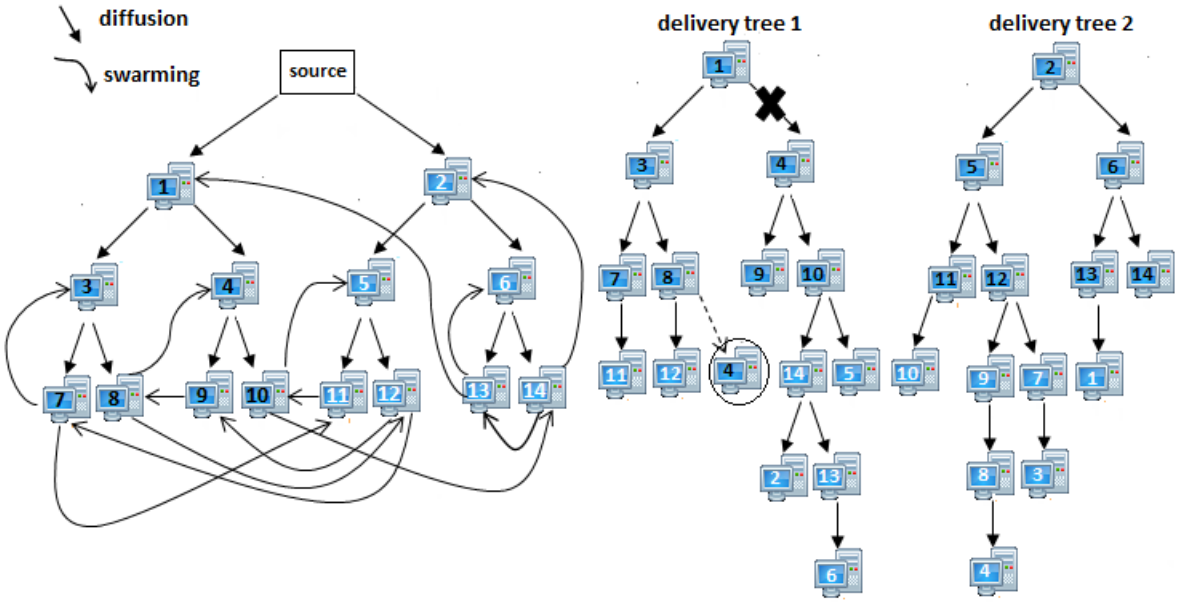


Figure 6: Diffusion and Swarming Phases in Mesh Method

phase, form a subtree, called the diffusion tree.

*b)Swarming Phase:* During this phase, peers, generally in the different diffusion trees, exchange their packets. All the connections from peer $a$ in level $i$ to peer $b$ in level $j$ ($j \leq i$) are used for swarming and thus this phase called as swarming phase.

#### *1.2.1.3 Similarities*

These two defined design architectures have significant similarities as follows;

- First of all, although these different approaches use different overlay construction algorithms, they result in overlays with very similar configurations.
- Second, in both approaches each peer retrieves the content form multiple sources (parents) and serves multiple peers (children).
- Third, both approaches need a sufficient playout time for identifying the paths the streams will go.

### 1.2.2 Challenges

Although the P2P approach has important advantages, it also has noticeable drawbacks. One of the main challenges in building a P2P live video streaming application is how to design an efficient mechanism in terms of using the network provider's resources in an efficient way, and providing a good Quality of Experience (QoE) to the users, in terms of pause duration, play-out time, etc.

First of all, the P2P traffic is naturally symmetric since every node behaves as a client and a server at the same time. However, today's widely available broadband access networks such as Digital Subscriber Loop (DSL) and/or cable modems are asymmetric, with downstream bandwidth values 5 or more times higher than the corresponding upstream bandwidth values [15].

Second, P2P networks find themselves a place at the Application Layer of the Open Systems Interconnection (OSI) Model. This means that P2P networks are network interconnection-oblivious. Therefore, P2P applications specify the pairs of nodes which will communicate and share their resources without any regard to the Internet map and current load. In other words, applications use random peer selection algorithms in order to send a peer list to the requesting peers. This random peer selection mechanism may cause traffic to unnecessarily traverse multiple links inside in an ISP or between ISPs causing unnecessary burden on the network and also results in much more delay for the corresponding video packet which is not desired. In [26] Xie H. et al. mention that, on the average each P2P bit traverses 1000 miles and 5.8 hops. They claim that the average hop count can be reduced to 0.89, without degrading the application's performance.

## *1.3 P4P: Provider Portal for Applications*

As discussed above, P2P applications are network-oblivious. Therefore, many P2P applications may cause inefficient usage of network resources and low application

performance. Both network providers and application servers have proposed various solutions to this problem. For instance, network providers have attempted to use various traffic control techniques. But, none of them are fully satisfactory without the P2P applications' cooperation. P2P applications have also tried to solve this problem by using peering flexibility [27, 28]. Even so, there are fundamental limits on what applications can achieve alone. P2P applications will have to infer various types of network information such as topology, congestion status, cost and policies. Reverse engineering of such information is challenging if not impossible.

As a solution to this problem, Xie et al. have proposed a simple architecture called the P4P (Provider Portal for Applications) to provide an effective traffic control by enabling cooperation between network providers and applications. P4P maintains multiple interfaces for network providers to exchange information with applications about network policies, status and capabilities.

P4P is composed of a data plane which is optional, a control plane and a management plane which monitor the behavior in the control plane. In the control plane, iTrackers are introduced as portals used by network providers. The introduction of iTrackers allows the P4P architecture to enable cooperation between applications and network providers in traffic control and also makes P4P scalable. In the P4P architecture, each network provider has an iTracker for its network.

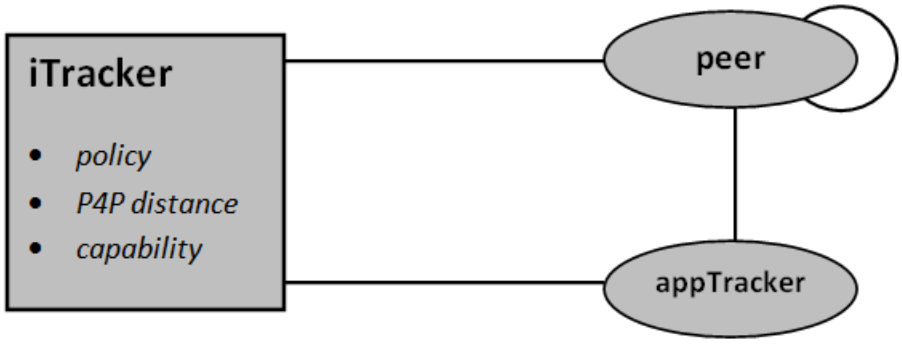


Figure 7: iTracker interfaces and information flow.

In the context of P2P (tracker-based) applications, the potential entities and the information flow in the P4P control plane are shown in Figure 7. The following example interfaces which are also given in Figure 7 are provided by iTracker: The policy interface lets applications acquire information about policies of the network. The p4p-distance interface lets applications learn costs and distances between peers. The capability interface lets content providers or peers ask for network providers' capabilities. A network provider can choose to implement a subset of the interfaces according to its will.

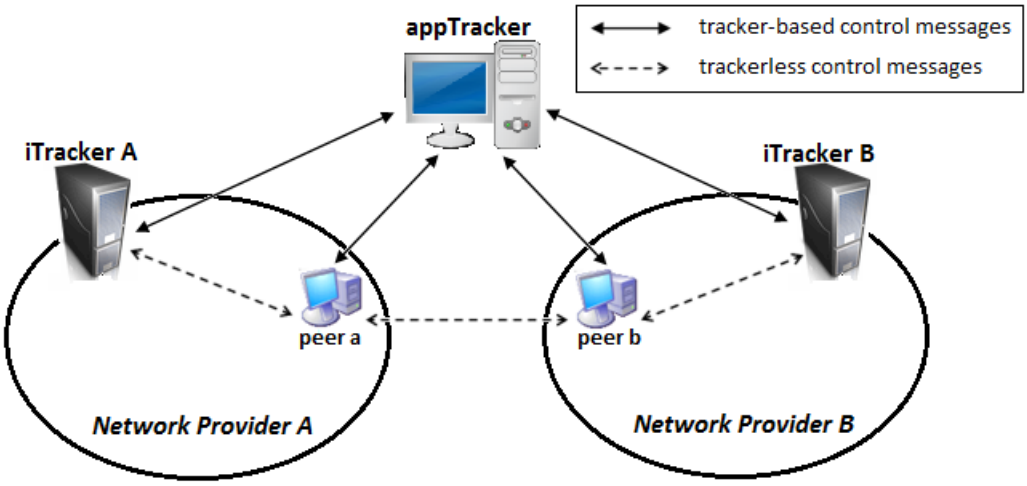


Figure 8: An example of P2P obtaining network information from iTrackers.

An example is given in Figure 8 to illustrate how iTracker's interfaces can be used. In the example, a P2P network covers two network providers named $A$ and $B$. Each network provider has an iTracker. Assume that the P2P application has an application tracker, referred to as the appTracker. A peer first registers with the appTracker. The appTracker obtains network specific information from iTracker $A$ using the interfaces, and perform peer selection for $a$ according to both the application needs and the iTracker information. As a different example, suppose the P2P application is trackerless. Then peer $a$ will retrieve information using the interfaces and choose its peers by itself.

Xie et al. have conducted extensive simulations and real-time experiments on the Internet to demonstrate the effectiveness of their approach. The experiments showed that P4P either improves or maintains the same level of application performance of former P2P applications, while significantly increasing the efficient use of network resources. As a consequence of P4P's success, a working group was founded in IETF so as to standardize the P4P design. The name of the working group is Application Layer Traffic Optimization (ALTO) and detailed information about ALTO is given in Chapter IV.

## *1.4 Contributions of the Thesis*

Building a P2P live streaming application which uses network resource efficiently and also provides a good QoE to its users is the fundamental focus of this thesis. In this thesis, we propose an ALTO service capable of servicing a live scalable video streaming application. We propose an H.264 SVC video encoder based video streaming application suitable for use with the ALTO protocol. We conduct simulations to demonstrate the effectiveness of the proposed streaming video application with ALTO. Simulation results indicate that using the ALTO protocol, scalable video based streaming video P2P applications can obtain good performance while, at the same time, reducing inter-ISP traffic rates.

## *1.5 Summary of the Thesis*

The rest of the thesis is organized as follows. In Chapter II, several P2P live streaming applications proposed in the literature are summarized. Chapter III explains the proposed P2P live scalable video streaming application. Chapter IV describes the ALTO protocol and discusses the changes in the proposed streaming protocol to enable the use of ALTO. Chapter V explains the simulation setup and evaluates the performance of the proposed application. Finally, this thesis is concluded in Chapter VI and possible future research directions are given in Chapter VII.

# CHAPTER II

# LITERATURE SURVEY

In this chapter, we provide an extensive literature survey on P2P live streaming applications that have been proposed and/or that are in use.

## *2.1 CoolStreaming/DONet*

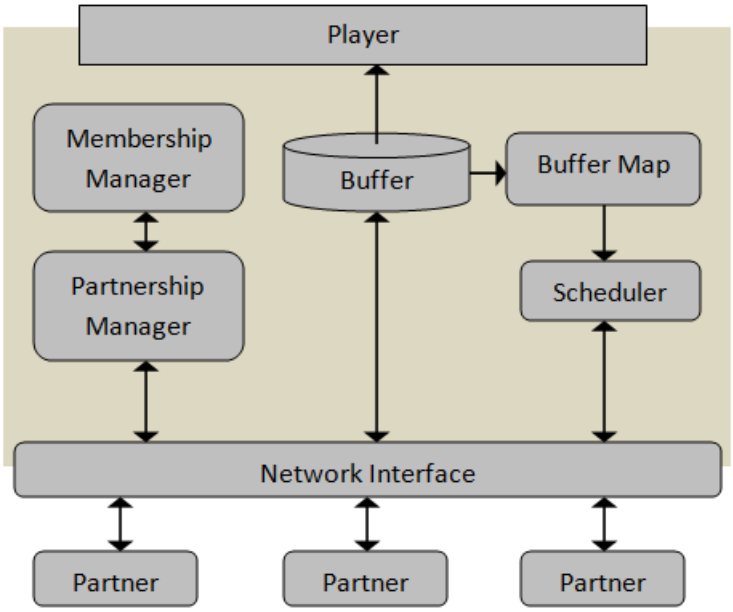


Figure 9: CoolStreaming system diagram

Coolstreaming/DONet is a mesh-based, and practically realized P2P streaming method [29]. Coolstreaming is the name of the application based on the DONet protocol. There are three main modules in Coolstreaming which are Membership Manager, Partnership Manager and Scheduler. Membership Manager provides the partial view of the overlay . Partnership Manager establishes the connections between peers and the bootstrapping server. It also exchanges the availability of stream data

in the Buffer Map (BM). The Scheduler schedules the transmission of video data and decides where and how to get stream data. In this system, every peer node periodically exchanges its BM with a set of partners and also provides data to other peers. Figure 9 shows system diagram of CoolStreaming/DONet.

In DONet, each peer has a unique identifier. Also each peer keeps a membership cache (mCache) that includes a partial list of the identifiers for the other peers in DONet. A newly joined node first contacts the origin node, which is the main source of the stream data. Then the orgin node picks a deputy node from its mCache randomly and redirects the new node to this deputy. The new node can then get a list of nodes from the deputy, and try to establish partnership with these nodes.

In 30 May 2004, a public Internet-based DONet implementation was released with name CoolStreaming v.0.9. This application has attracted over 30000 different users with more than 4000 simultaneously online users at some peak times.

## *2.2 AnySee*

AnySee [30] is a P2P live video streaming method that introduces an inter-overlay optimization based scheme in which nodes can join multiple overlays simultaneously.

As can be seen in Figure 10, AnySee runs accoding to the following workflow. First, an efficient mesh-based overlay is constructed while the location detector based algorithm matches the overlay with the underlying physical topology. Second, the single overlay manager, which uses traditional intra-overlay optimization, looks into joining/leaving processes of peers. Third, the inter-overlay optimization manager searches for suitable paths, constructs backup links, and gets rid of paths with low QoS for each end peer. Fourth, the key node manager distributes resources, and the buffer manager manages the transmission of media data.

In AnySee, every node, with a unique identifier, first contacts the bootstrapping nodes and picks one or several nodes to establish logical links in order to join the

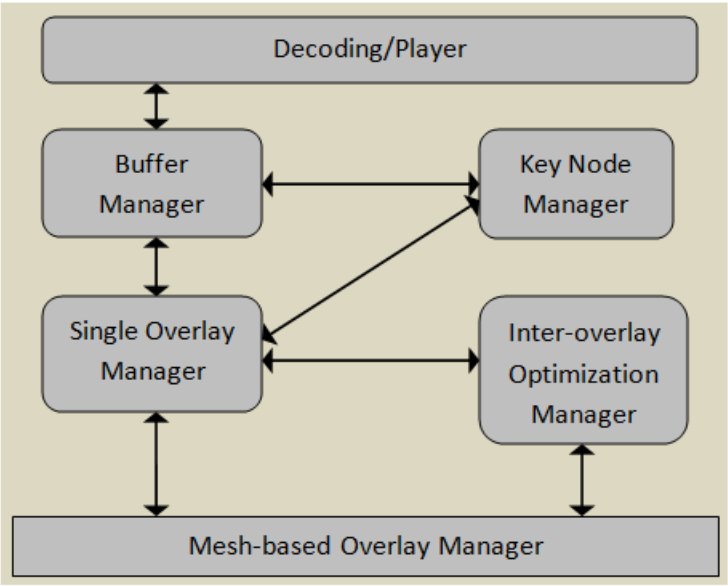


Figure 10: System diagram of an AnySee node

mesh-based overlay. Every node has a bunch of logical neighbours. The important point here is to match the mesh-based overlay with the underlying physical topology. To solve this issue, the mesh-based overlay management uses a location detector based algorithm. There are two main processes in here: flooding-based detection with limited Time To Live (TTL) and updating logical links. In the first process each node periodically sends a message to its neighbours with the following parameters: message ID, TTL, the IP address of the source peer, timestamp when the source forwarded the message, the IP address of the neighbour node and the timestamp when the neighbour node received the message. In the second process, with help of timestamps on peers the path between the node and the neighbours is compared and link is established with the closest neighbour. All peers do the same procedure and the overlay is constructed.

The practical version of AnySee was released in June 2004 in CERNET of China. From June 2004 to February 2005, there were more than 60,000 users that connected to the system.

## 2.3 SplitStream

SplitStream [31] is a high-bandwidth content distribution system based on P2P networking. It can allocate the transmission load between all the contributers and host nodes with different transmission rates.

The key idea is to split the content into k stripes, and forward each stripe in a different multicast tree. Nodes join these trees to receive the stripes that they need. Here the main goal is to build a group of multicast trees such that an inner node in one tree is a leaf node in rest of the trees. Thus, the transmission load can be distributed among all contributing nodes.

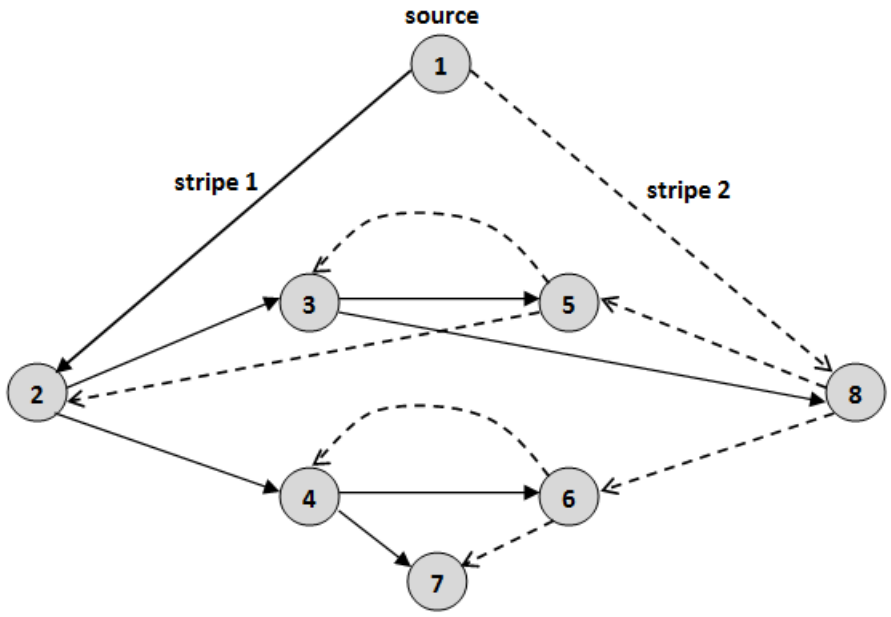


Figure 11: A simple example illustrating the basic approach of SplitStream.

Figure 11 shows how SplitStream distributes the transmission load between the contributing nodes. In this instance, the content is split into two stripes and forwarded in distinct trees. Here for simplicity, it is assumed that the content needs bandwidth of B, and that each stripe needs bandwidth as half of B. Each node other than the source joins both trees to get both stripes, causing an incoming bandwidth need of

B. As can be seen in Figure 11, each node is an interior node in only one tree and delivers the stripe to two children, causing an outgoing bandwidth need of lower than B.

Contributing nodes may join a subset of the trees, doing this they keep their incoming bandwidth need in multiples of $B/k$ where k denotes number of the stripes in the content. Contributing nodes may also keep their outgoing bandwidth need in multiples of $B/k$ by bounding the number of children they have. Therefore, SplitStream can host nodes with various bandwidths, and nodes with asymmetric incoming/outgoing network capacities.

A preliminary performance evaluation of SplitStream is performed by employing 40,000 SplitStream nodes over an emulated network with 5050 core routers based on the Georgia Tech network topology generator. The evaluation results was very promising [31]. The system can allocate the transmission load between the contributing nodes, considering separate node bandwidth limits. Using redundant content encoding technique such as MDC, SplitStream provides flexibility to node failures and ungraceful departures, even while a recovery process is active.

## *2.4 Chainsaw*

Chainsaw is a P2P overlay multicast system that removes trees completely. It is also a pull-based system. Thus, in order to receive a packet peers have to directly request it from one of their neighbors. In this way, data duplication and packet loss can be prevented.

A seed, which is also defined as the source node, produces new packets with monotonically increasing sequence numbers. If it is demanded, there could be multiple seeds in the network. But in [32] it is assumed that they have only one seed in the network.

Each peer connects to a set of neighbors and keeps a list of packets that each

neighbor has. If a peer receives a packet, it inform its neighbors with NOTIFY messages. It is obvious that the seed does not download packets, but it informs peers with NOTIFY messages whenever it produces a new packet.

Every peer has a window of interest, which is the range of packets that the peer wants to obtain at the current time. It also has a window of availability, which is the range of packets that it accepts to send to its neighbors. Remember that, every peer notifies its neighbors about its window of availability. For every neighbor, a peer composes a list of desired packets (a list of packets that the peer wants and the neighbor has). It will then select one or more packets from the list and request them with a REQUEST message.

The seed creates new packets at a constant rate that is called as the stream rate. Nodes have a window of interest with a constant size and slide it forward at the stream rate. If a packet has not been received in time it falls off the trailing edge of the window, and will be considered as lost packet.

## *2.5 PPLive*

PPLive is a very popular P2P Live Streaming application in China. Basically PPLive depends on a mesh-based topology and it has two basic steps [33]. First one is registration and the peer discovery protocol (i.e. Tracker Protocol) and the second one is the P2P Chunk distribution protocol (i.e. Peer Protocol). When a peer connects to the PPLive network firstly it obtains the channel list from the Channel Server. Then the peer selects a channel and requests the peer list of the corresponding channel from the Tracker.

By sending the buffer map to each other, every peer makes a decision about the peers it connects. The buffer map message contains the ID of the first chunk, the length of the buffer map, and the string of zeros and ones to indicate the available and unavailable chunks. PPLive runs over UDP. The basic topology of PPLive is

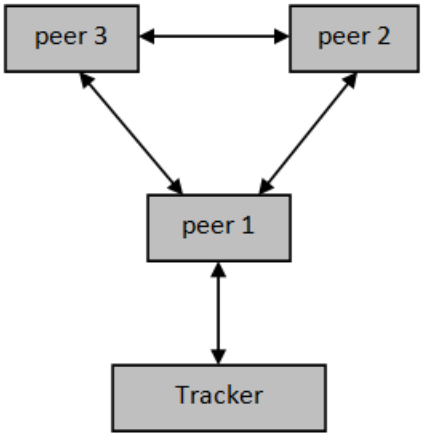


Figure 12: Basic overlay of PPLive.

shown in Figure 12.

PPLive web site [17] reported that in May 2006, PPLive has attracted 400,000 daily users on average with more than 200 channels. PPLive presents video programs with varying rates from 250 Kbps to 400 Kbps. Only a few channels have a high rate of 800 Kbps.

## *2.6 PeerCast*

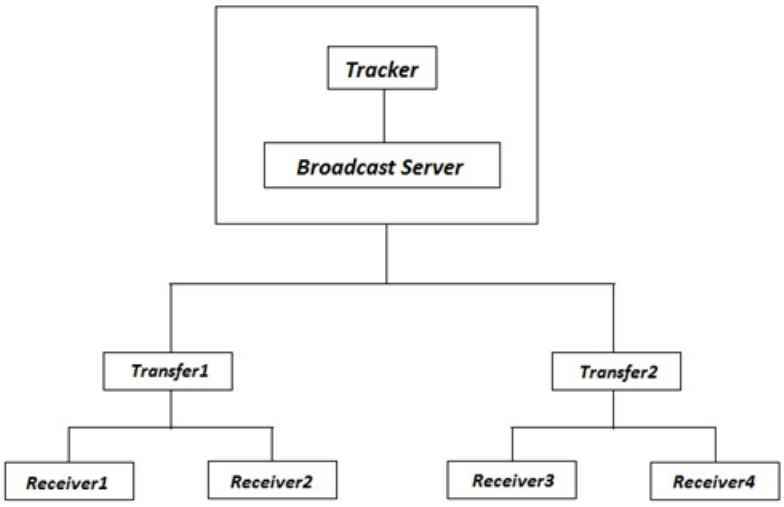


Figure 13: Architecture of PeerCast system.

PeerCast is a tree-based P2P streaming method [33]. Peers in a channel create a Broadcast Tree while they select the Broadcast server as the root of the tree. A Tracker can be implemented separately or it can be merged into the Broadcast server. The function of the Tracker is to select the parent nodes for those newly joined peers. There are two types of nodes in this structure. First, transfer nodes receive and transfer data simultaneously. Second, receiver nodes (leaf nodes) only receive data. A Transfer node forwards the stream to its children. A child can be a transfer node or a receiver. Figure 13 illustrates the architecture of the PeerCast system [33].

After joining a channel and getting the broadcast server address, the peer first sends a request to the server. The server responds with an OK or not considering its available capability. If the answer is OK, it adds the peer to its children-list. If the answer is not OK, the broadcast server selects at most eight nodes of its children and sends a response message, that includes these at most eight nodes, to the peer. After receiving these nodes, the peer selects a peer among them randomly and contacts that peer until it find an available node. Furthermore, each node in the tree informs its parent about its status periodically while the parents update their children-list considering information received from their children.

## *2.7 End System Multicast (ESM)*

ESM is a tree-based approach and has a single tree for its overlay topology [16]. Figure 14 shows the architecture of ESM.

Main functional components of ESM consist of two parts: First, it has a bootstrap protocol, a parent selection algorithm and a light-weight probing protocol. Second, it has a distinct control structure separated from the tree, and a gossip-like algorithm lets each node to know a subset of nodes in the group.

A newly joined node subscribes to a subset of groups via the root (source) node. Then it uses the parent selection algorithm to find a parent. Furthermore, it uses

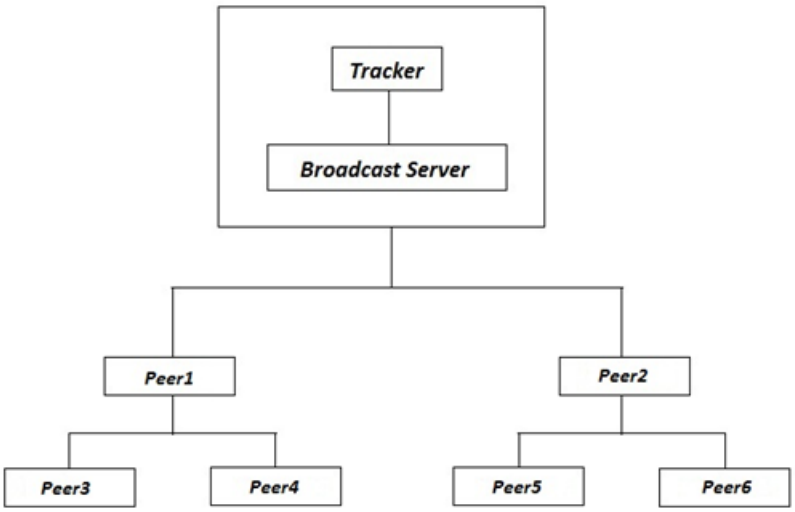


Figure 14: Architecture of ESM system.

light-weight probing protocol on a group of nodes it knows to compare the nodes and select a candidate parent. The parent selection algorithm is also used for dealing with node and network churns.

As an important property, ESM gives support to NATs. In ESM, NATs can be parents or children of public hosts.

# CHAPTER III

# P2P LIVE SCALABLE VIDEO STREAMING PROTOCOL

In this chapter, we introduce the proposed P2P live scalable video streaming protocol. The proposed protocol builds on the CoolStreaming [23] protocol. CoolStreaming was chosen as a starting point due to its ease of implementation and prior success. However there are significant differences between the proposed protocol and CoolStreaming. First of all, CoolStreaming is a trackerless streaming application while we define a tracker-based streaming application. Also it is important to note that our proposed protocol is ALTO capable which means that the tracker of our application can communicate with the ALTO Servers of the ISPs. Last of all, unlike CoolStreaming, the proposed protocol conducts streaming based on the MPEG4-SVC codec.

## *3.1 Overlay Construction*

In our protocol peers on the network construct a mesh-based overlay in order to share the video content amongst them. In order to understand the proposed mesh overlay and its construction mechanism, it is beneficial look at the joining process of a peer to the overlay network. In this process, first, the peer sends a JOIN message to tracker. Upon receiving the JOIN message, the tracker registers the peer to the overlay network and sends a PEER LIST which consists of a subset of peers - their IP addresses - on the overlay network. Having received this peer list, the peer sends a PARTNERSHIP REQUEST message to a subset of the peers from the peer list. If a peer that receives the message accepts it, this peer becomes a parent of the peer that request the partnership. A peer can request video chunks only from their parents and this peer is called a child of those parents. Thus a peer can have both parents and children at the same time and together they are called as partners of that peer.

A peer requests and receives video chunks from its parents and shares its chunks with its children. In conclusion, peers on the network construct a mesh-based overlay establishing these parent-child relationships. The mesh overlay is shown in Figure 15.

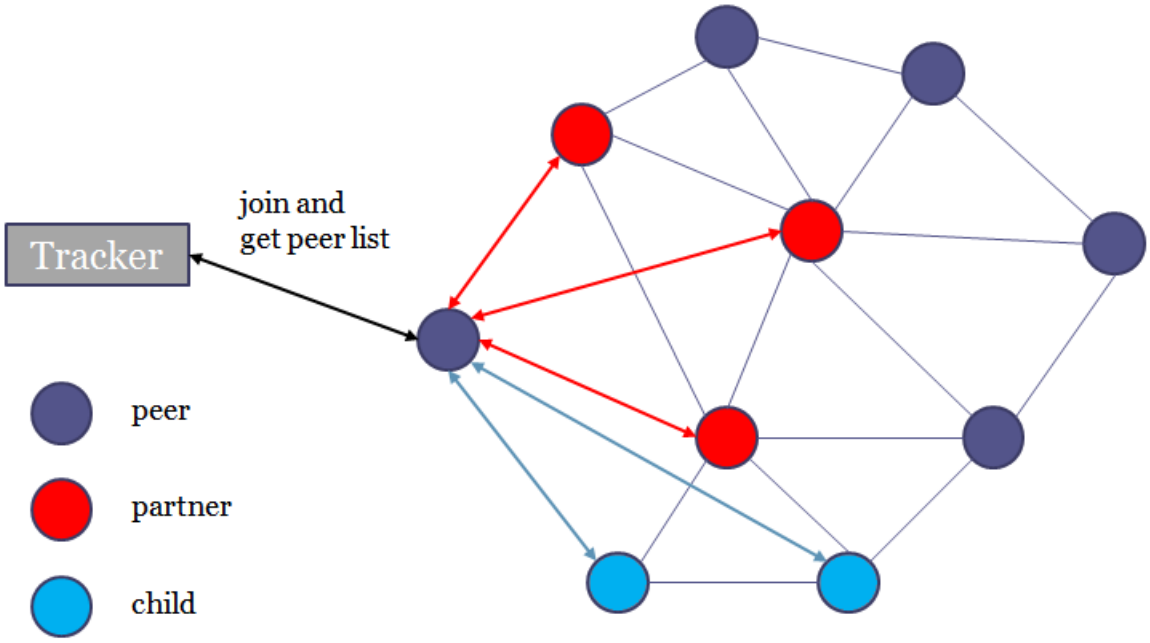


Figure 15: Mesh overlay of the protocol.

## *3.2 Joining and Leaving*

As mentioned before, a peer sends a JOIN message to the tracker in order to join the overlay network. After receiving the JOIN message, the tracker registers the peer - in other words, the tracker adds the IP address of that peer to its peer list - to the overlay network and sends a JOIN REPLY message which acknowledges that the peer is now joined to the overlay network and can request the peer list from the tracker. Having received the JOIN REPLY, the peer sends a PEER LIST REQUEST message to the tracker. After receiving the request message, the tracker randomly selects a subset of its peer list consisting of 20 peers and sends this subset to the peer. Having received the list, the peer randomly selects some of the peers on the list and tries to

establish partnerships. Note that after receiving the first peer list, peers send PEER LIST REQUEST messages to the tracker regularly with a period of 5 seconds in order to refresh their peer list. These processes are summarized in Algorithm 1 (for peers) and 2 (for tracker).

**Algorithm 1** Joining Process for Peers

```
send a JOIN message to tracker;
wait for response;
if response is received then
    send a PEER LIST REQUEST message to tracker;
    wait for peer list;
    if list is received then
        randomly select one of the peers in the list;
        send a PARTNERSHIP REQUEST to that peer;
    end if
end if
```

**Algorithm 2** Joining Process for Tracker

```
wait for messages;
if a JOIN message is received then
    add the peer to the peer list;
    send a JOIN REPLY message to the peer;
else if a PEER LIST REQUEST is received then
    create a list consists of randomly selected 20 peers;
    send this list to the peer;
end if
```

When a peer is leaving the overlay network, it sends a LEAVE NOTICE message to the tracker, the servers, its parents and children. After receiving this message, all nodes remove the corresponding peer from their lists.

## *3.3 Partnership Management*

In order to establish a partnership the following interaction must be realized between two peers. First, peer 1 sends a PARTNERSHIP REQUEST message to peer 2. Having received the request message, peer 2 checks its remaining upload bandwidth.

If there is enough bandwidth, peer 2 decreases its remaining upload bandwidth by an amount that is equal to the size of a video packet, adds peer 1 to its children list and sends an affirmative PARTNERSHIP RESPONSE message to peer 1. With this affirmative response peer 2 also sends its buffer map to peer 1. After receiving this acknowledgement message, peer 1 adds peer 2 to its parent list and with this, partnership establishment is completed. If there is not enough bandwidth, peer 2 sends a negative PARTNERSHIP RESPONSE message to peer 1. After receiving this negative acknowledgement, peer 1 removes peer 2 from its peer list. Actions which are taken by peers when they want to establish a partnership or answer a partnership request are shown in Algorithm 3 (to request) and 4 (to respond), respectively.

**Algorithm 3** Partnership Request

```
randomly select a peer;
send a request message to that peer;
wait for response;
if response is affirmative then
    add the peer to the parent list;
else
    remove the peer from the peer list;
end if
```

**Algorithm 4** Partnership Response

```
get request message;
if remainUpBw ≥ segmentSize then
    // remainUpBw: remaining upload bandwidth
    add the peer to the child list;
    remainUpBw ← remainUpBw − segmentSize;
    send an affirmative response message;
    send all buffer maps;
else
    send a negative response;
end if
```

After receiving a peer list, the peer randomly selects another peer from its peer list and sends a PARTNERSHIP REQUEST message to that peer. After receiving the response, the peer randomly selects another peer and sends a PARTNERSHIP

REQUEST again without any regard to the previous response.

The number of parents that a peer has must be limited since peers with high rates can rapidly obtain many parents and prevent peers with low rates having enough number of parents. On the other hand, a very low limit reduces the performance of the system in terms of high delays and low PSNR values. If this limit is increased, the performance of the system also improves. But at a certain point the improvements become marginal since total load on the network also increases with this limit. Therefore this limit value must be optimally found to maintain a good performance. To find this value, we run simulations in our setup with different limit values and using brute-force optimization select the value that provides the best performance as our limit. This number is equal to 5 in our setup. In our protocol this limit is defined as the maximum parent number and every peer repeats sending PARTNERSHIP REQUEST messages until it reaches the maximum parent number.

Furthermore, for our proposed protocol, we developed a scoring mechanism in order to allow peers update their parent list with new parents of better quality. According to this mechanism, a peer ranks its parents according to a score value which is equal to the number of video packets which it has thus far retrieved from the corresponding parent. When the peer receives a new peer list, it removes the parent with the lowest score from its parent list, resets the score of the remaining parents and tries to get a new parent. Since a peer receives a new peer list every 5 seconds, this scoring mechanism has a period of 5 seconds as well.

If a peer wants to end its partnership with one of its parents, it sends a PARTNERSHIP END message to the parent. After receiving the message, the parent removes the peer from its children list.

## 3.4 Video Coding Model

Real networks are heterogeneous in link rates. Thus a live video streaming service should support streams with different qualities for users with different rates. A possible solution to this problem is achieved by creating a different video file for each quality level. However, this solution is inefficient due to data duplication. To overcome this challenge, the use of Scalable Video Coding (SVC) techniques is more efficient in order to support streams with different qualities by encoding the video only once. A scalable video which is also called layered video consists of a base layer that provides basic quality and enhancement layers that refine quality incrementally. Therefore, by using a layered video stream we can ensure that all users in the network are able to play the stream at least in basic quality.

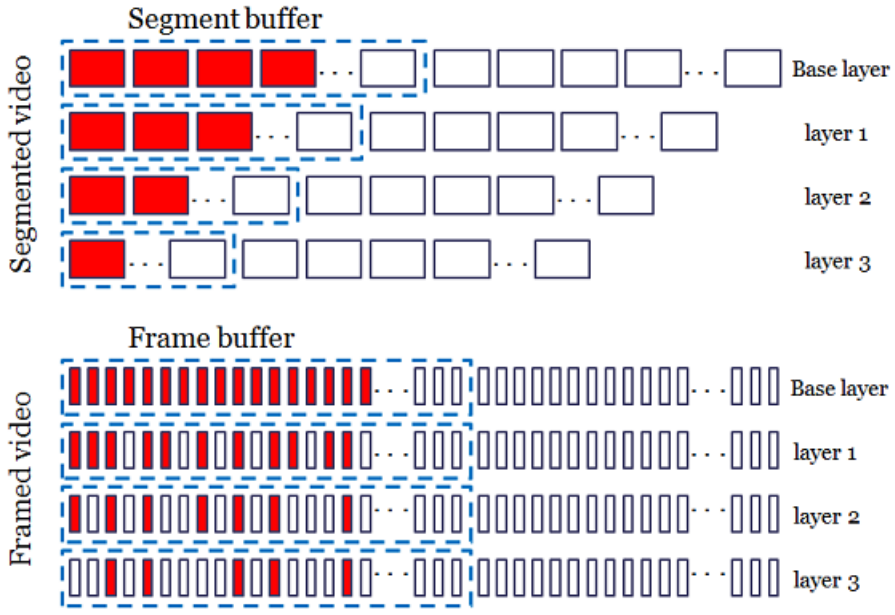


Figure 16: Video structure.

In our protocol, the encoded video has four - one base and three enhancement - layers. Furthermore every layer is divided into equal sized pieces which are called segments. We select the size of a segment so that a frame can be divided into at most two segments. Layers are divided into segments when the frames are in encoding

order to ensure that each segment can be decoded independently. Since the total size of a layer decreases towards the higher layers, the number of the segments in the layers decreases in same way. The structure of our scalable encoded video is shown in Figure 16.

### 3.4.1 Buffers

In our proposed protocol, a peer has two types of buffers for each layer of video. These are segment buffers that consist of downloaded segments and frame buffers that consist of extracted frames from downloaded segments. Frame buffers can be thought of as the play-back buffers of a media player. Since we have four layers, a peer has four segment and four frame buffers. When a segment is received, it is first placed to the appropriate segment buffer. Then, the frames in the segment are extracted and placed into the appropriate frame buffer. It is important to mention here that while segments are placed in encoding order, frames are placed in play-back order. All buffers have a length of 26 seconds. Thus the length of a frame buffer is 624 frames and is same in all layers. Note that we consider a frame rate of 24 fps. On the other hand, segment buffers have different lengths since the number of the segments in the layers decreases towards higher layers.

### 3.4.2 Play and Pause Mechanisms

In order to start the play-back of the video, we require that a peer must fill at least 25% of its base layer frame buffer. But since we want all peers in the network to start the play-back of the video, we modify this rule as follows. A peer that fills up 25% of its base layer frame buffer can start to play the video, but if it doesn't fill up its buffer in 30 seconds, it can also start to play the video. After this buffering stage, the video play-back starts. While a segment buffer slides with a period of *video duration/number of segments in the corresponding layer*, a frame buffer slides with a period of *1/frame rate of the video*. A frame which is found in the corresponding buffer

- at the beginning of the buffer - in its playing time is played and destroyed after that. A segment which gets behind of the corresponding segment buffer is also destroyed. If a base layer frame is not received before its playing time, the corresponding peer enters a pause stage. In this stage the peer waits until it fills up 25% of its base layer frame buffer and then continues to play the video. We implement a similar pause mechanism for base layer segment buffer as well.

### 3.4.3 Buffer Map Representation and Exchange

Availability of the segments in a segment buffer is represented by a Segment Buffer Map (SBM). Since we have four segment buffers, we have four SBMs too. Each peer send its SBMs to its children and receive SBMs from its parents and this SBM exchange is triggered by the shifts in the corresponding segment buffer. Also each peer informs its children when it receives a segment.

## *3.5 Scheduling Algorithm*

Given the SBMs of a peer and that of its parents, a schedule must be generated to request the missing segments from the parents. For this purpose, we modify and use a simple heuristic algorithm which is also used by CoolStreaming. This heuristic algorithm first calculates the number of potential suppliers for each segment. Since a segment with less potential suppliers is more difficult to meet the deadline constraints, the algorithm determines the supplier of each segment and schedules segments starting from those with only one potential supplier, then those with two, and so forth. This heuristic is usually referred to as "Rarest First". Among the multiple potential suppliers, the one with the highest bandwidth and enough available time is selected. The algorithm is executed with a period of 5 seconds to update the schedule. Note that the algorithm is executed first when a peer has its first parent.

Since CoolStreaming doesn't use scalable video, their algorithm isn't readily appropriate to our protocol. So we modified the algorithm for scalable video. In this

modified version of the algorithm, the same process is applied to layers of the scalable video in a tiered fashion starting from base layer. Thus, in our scheduling algorithm base layer segments have priority over enhancement layer segments.

In our scheduling algorithm, there are two important parameters that should be chosen carefully since they can effect the overall performance of the system dramatically. These parameters are the buffer length and the scheduling period. For instance, If the buffer length is very short, the pre-roll delay will be very low since there will be a small number of segments that must be obtained before starting to play-back of the video. On the other hand, number/duration of pauses will be very high since the number of segments that are obtained before their deadline will be very small. When the length of the buffer increases, the pre-roll delay increases and the number/duration of pauses decreases. Video quality experienced by the users is also effected by the buffer length. If the buffer length is very long, the quality of the video will be low since the number of base layer segments that must be obtained before obtaining any enhancement layer segments will be high. When the length of the buffer decreases, the video quality increases. However, buffer length should not be very low, since the video quality will be low as well. This is due to the fact that the number of segments that are obtained on time is very low. Therefore, the buffer length should be short enough to provide high quality, small pre-roll delay experience and long enough to keep number/duration of pauses small.

The scheduling period can also effect performance of the system in terms of pre-roll delay, number/duration of pauses and video quality. If the period is short, the pre-roll delay, number/duration of pauses will be low and video quality will be high, since the unavailable segments will be detected and requested on time. On the other hand, If the period is too short, the traffic on the network will increase unnecessarily and the performance of the system will be effected badly. If the period is too long, performance of the system will be bad again, since unavailable segments can not be

requested on time. Therefore, scheduling period should be short enough to keep the performance of the system high and long enough to keep the unnecessary traffic load small.

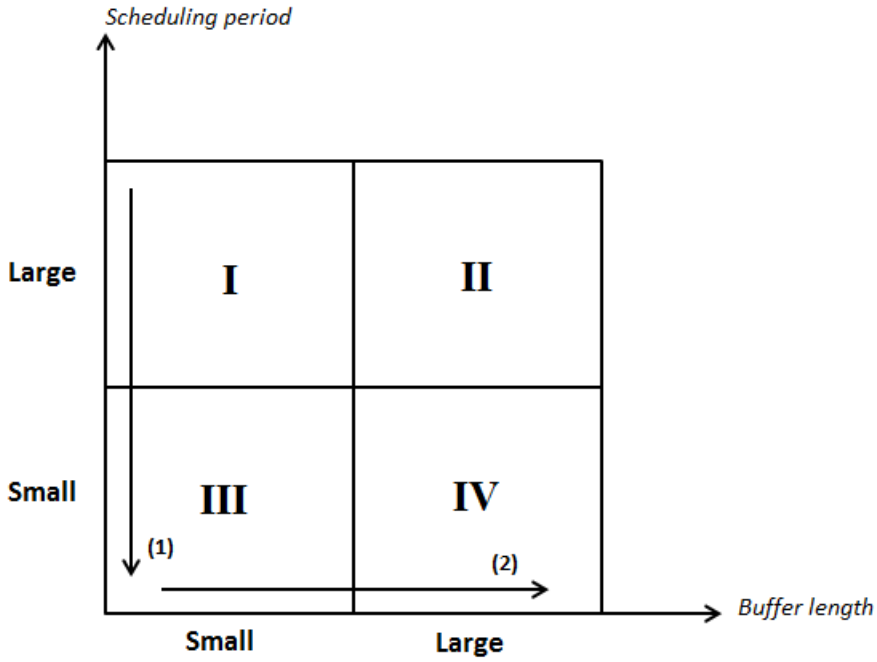


Figure 17: Operation regions for the scheduling algorithm

In Figure 17, the operation range of our application is divided into 4 regions according to scheduling period and buffer length. If the scheduling period decreases while the buffer length is constant and large enough, the performance of the application increases. In Figure 17, this case is represented by the arrow with number 1. If the buffer length increases while the scheduling period is constant and small enough, the performance of the application increases as well. In Figure 17, the arrow with number 2 represents this case. Therefore, the optimum operation point must be in region IV and the worst performance must be seen in region I.

The pseudo code of the scheduling algorithm is given in Algorithms 5 and 6 and definitions of some terms in the algorithm are given in Table 2.

**Algorithm 5** Scheduling Algorithm Part I

```
for layer l to number_of_layers do
    for segment i ∈ seg_buffer(l) do
        if buffer_map(l, i) = 0 then
            for partner j to number_of_partners do
                if video is started to play then
                    aval_time(j, i) ← (seg_num(i)−next_seg(l))×play_period(l);
                else
                    aval_time(j, i) ← 200;
                end if
            end for
            expected_set(l) ← expected_set(l) ∪ {i};
        end if
    end for
    for segment i ∈ expected_set(l) do
        n ← 0
        for partner j to number_of_partners do
            if buffer_map(j, l, i) = 1 then
                n ← n + 1;
            end if
        end for
        if n = 1 then
            k ← arg_r{buffer_map(r, l, i) = 1};
            supplier(i) ← k;
            for segment j ∈ expected_set(l), j > k do
                aval_time(k, j) ← aval_time(k, j) − seg_size/up_band(k);
            end for
        else
            if n ≠ 0 then
                dup_set(n) ← dup_set(n) ∪ {i};
            else if n = 0 then
                if video is not started to play then
                    if sim_time > 5 then
                        select a server randomly;
                        supplier(i) ← server;
                    end if
                else
                    if seg_num(i) − next_seg(l) < buffer_size(l)/2 then
                        select a server randomly;
                        supplier(i) ← server;
                    end if
                end if
            end if
        end if
    end for
```

**Algorithm 6** Scheduling Algorithm Part II

**for** $n = 2$ **to** $number\_of\_partners$ **do**
  **for** each $i \in dup\_set(n)$ **do**
    $k \leftarrow$
      $arg_r\{up\_band(r) > up\_band(r') |$
        $aval\_time(r, i) > seg\_size/up\_band(r)$,
        $aval\_time(r', i) > seg\_size/up\_band(r')$,
        $r, r' \in set\_partners\}$;
    **if** $k \neq null$ **then**
      $supplier(i) \leftarrow k$;
      **for** segment $j \in expected\_set(l), j > k$ **do**
        $aval\_time(k, j) \leftarrow aval\_time(k, j) - seg\_size/up\_band(k)$;
      **end for**
    **end if**
  **end for**
**end for**

**Output:**
$supplier(i)$: **supplier for unavailable segment** $i \in expected\_set$

Table 2: Definitions of some terms in the scheduling algorithm

| **Term** | **Definition** |
|---|---|
| $up\_band(k)$ | upload bandwidth of partner k |
| $buffer\_map(k, l)$ | layer l buffer map of partner k |
| $aval\_time(k, i)$ | available time to obtain segment i from partner k |
| $seg\_size$ | segment size |
| $seg\_num(i)$ | sequence number of segment i |
| $next\_seg(l)$ | sequence number of next segment in layer l |
| $play\_period(l)$ | playing period layer l |
| $seg\_buffer(l)$ | segment buffer of layer l |
| $expected\_set(l)$ | set of segments to be fetched for layer l |
| $dup\_set$ | set of segments that are available in multiple partners |

Servers distribute the segments according to the peer list - consists of 50 peers - which is sent by the tracker with the same period that segment buffers have. Furthermore, a peer can also request a base layer segment which is not available in its parents from the server directly. Thus traffic to the servers increases unexpectedly. In order to prevent this, we formulate the following rules. In the buffering stage, a

peer can request a base layer segment from the server only 5 seconds after joining. In the play-back stage, a peer can request a base segment from the server only if the segment is in the first half of the buffer.

# CHAPTER IV

# APPLICATION LAYER TRAFFIC OPTIMIZATION PROTOCOL

## *4.1 Protocol Description*

The goal of Application-Layer Traffic Optimization (ALTO) is to provide an information service which can help P2P applications to make better decisions with respect to peer selection. ALTO is actually is a working group in IETF and their main goal is standardizing the P4P [26] design under the ALTO name. The initial goal was to improve the performance of P2P applications, but other distributed applications like CDNs and traditional client/server systems also can benefit from ALTO. For example, clients of client-server applications may use information provided by ALTO to select one of several servers or information replicas.

### 4.1.1 Benefits

The ALTO protocol provides many benefits to both end-users and ISPs. It lets ISPs be involved in the peer selection in distributed applications such as P2P in order to keep the traffic local and improve QoE of users. It also supports ISPs to control, the traffic that is in more expensive links like transit links. Applications also can take advantage of ALTO in many ways. For instance, they may not require to infer topology of the network anymore, and some applications may no longer need to measure path metrics indirectly themselves. Furthermore, they can use the knowledge of ISP to avoid bottlenecks and improve performance.

### 4.1.2 Important Terms

The following terms defined in [34, 35] and are used frequently in the following sections. So, it is useful to give the definitions of these terms, before we go any further.

- **ALTO Server**: A logical entity that provides interfaces to the queries to the ALTO service.
- **ALTO Client**: The logical entity that sends ALTO queries. Depending on the architecture of the application, one may embed it in the resource consumer and/or in the resource directory.
- **ALTO Query**: A message sent from an ALTO client to an ALTO server; it requests guidance from the ALTO service.
- **ALTO Response**: A message that contains guiding information from the ALTO service as a reply to an ALTO query.

An ALTO Server has the network information from a network region's point of view. In other words, the ALTO Server provides its "My Internet View" of the network region. Specifically, an ALTO Server identifies network endpoints (and groups that consist of them) and generic costs between them, both from the network region's point of view. Here a network region can be an Autonomous System, an ISP, or perhaps a smaller region or set of ISPs.

### 4.1.3 Protocol Structure

The ALTO Protocol employs a simple extensible framework to carry network information. In general, the ALTO protocol carries information about both Network Locations and the paths between them.

The ALTO Protocol consists of a common transport protocol, messaging structure and encoding, and transaction model. The protocol is divided into information services considering their functionality. The Map Service delivers the main ALTO information to clients. The rest of the information services offers additional functionalities. Here, the Map Service will be explained only due to its importance. The rest of the services are defined in [35]. Figure 18 shows ALTO Service framework.

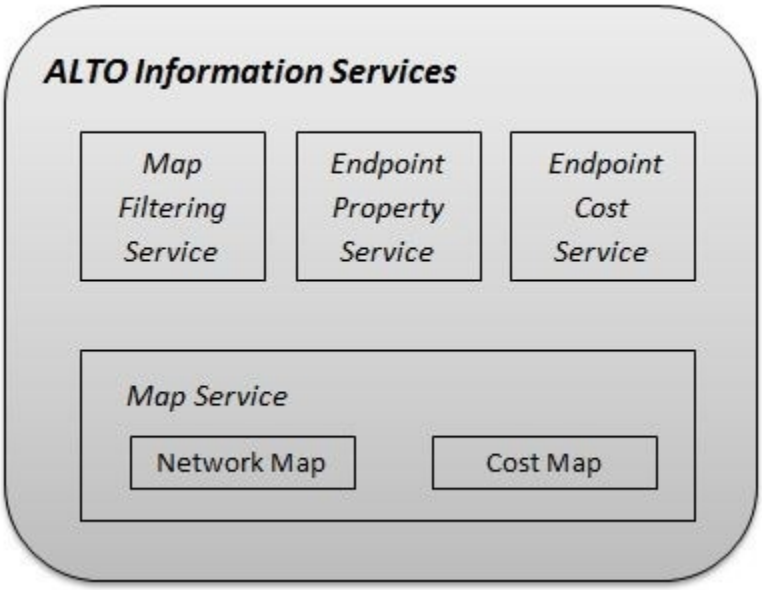


Figure 18: ALTO Service Framework.

### 4.1.4 Map Service

The Map Service delivers information to ALTO Clients in terms of Network Map and Cost Map. While the Network Map gives all endpoint groups defined by the ALTO Server and the endpoints forms each group, the Cost Map gives costs between these groups.

#### *4.1.4.1 Network Map*

An ALTO server can group endpoints together to describe their proximity. ALTO Network Map consists of these resulting groups. The description of proximity changes according to the detail of the ALTO information arranged by the provider. In one instance, a subnet may be selected as a group, while in another instance, endpoints connected to the same PoP may be selected as a group.

Each group has a provider-defined Network Location identifier named a PID. A PID lets to identify a group of endpoints that may be handled similarly, considering network properties like topology, type, etc. An example Network Map is shown in Figure 19.

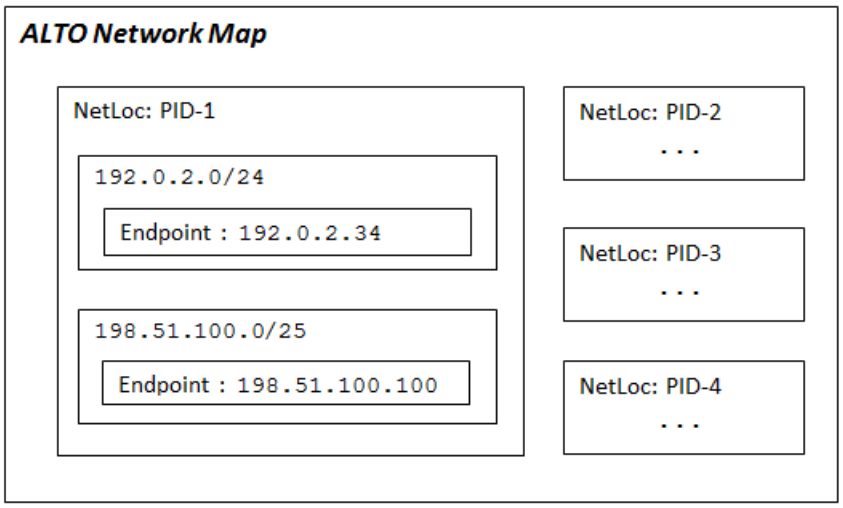


Figure 19: An example Network Map.

#### *4.1.4.2 Cost Map*

An ALTO Server specifies choices between network locations in terms of path costs. A Cost Map provides path cost values pairwise between sets of source and destination locations. Each path cost is calculated as the end-to-end cost from the source to the destination.

Path costs have two attributes: type which indicates what the cost shows, and mode which indicates how the cost should be inferred. Air-miles, hop-counts, or generic routing costs can be examples for type attribute. Due to its mode a cost can be numerical which indicates that performing numerical operations on this cost is safe or ordinal which indicates that the costs values in the Cost Map represents a ranking while lower values means a higher preference.

## *4.2 Enabling ALTO Service in Our Video Streaming Protocol*

In order to make our streaming protocol ALTO service capable, we added ALTO Server to the overlay network as a new entity. In our ALTO Server, Network Map and Cost Map services were implemented but the additional services which are defined in [35] weren't implemented due to simplicity.

### 4.2.1 Changes in the Streaming Protocol

For ALTO capability, we changed the implementation of the tracker so that it will be seen as an ALTO Client. When a peer sends a peer list request, tracker sends an ALTO INFORMATION REQUEST message to the ALTO Server in order to obtain the cost values between all of peers. After receiving ALTO INFORMATION REPLY, the tracker creates a ranked peer list according to the cost values that are received from ALTO Server and sends this list to the peer. After receiving the peer list, the peer selects the peer that is placed at the top of the list and sends a partnership request since the list is already ranked according to cost values. The modified version of the joining process for the tracker is shown in Algorithm 7.

In our simulated network, every ISP has an ALTO Server which calculates the cost information from the ISP's point of view. Thus, when the tracker receives a peer list request, it obtains the cost information from the ALTO Server that is in the same ISP as the peer.

**Algorithm 7** Joining Process for ALTO Capable Tracker

```
wait for messages;
if a JOIN message is received then
    add the peer to the peer list;
    send a JOIN REPLY message to the peer;
else if a PEER LIST REQUEST is received then
    create a list consists of randomly selected 20 peers;
    send a ALTO INFORMATION REQUEST message to the ALTO Server;
    receive the ALTO INFORMATION REPLY;
    rank the list according to the information;
    send this list to the peer;
end if
```

### 4.2.2 Proposed Cost Calculation Methods

There are two cost calculation methods that are used in our ALTO capable streaming protocol.

#### *4.2.2.1 Minimum Delay Based (MDB) Cost Calculation Method*

Delay plays a very important role on Internet, especially for the delay critic applications (*etc.* VoD applications, live video steaming applications). Thus, we propose a minimum delay based cost calculation method in this section. In packet-switched networks, experienced delay of a packet on a simple link and a router is modeled with the following equation [36].

$$d_{nodal} = d_{proc} + d_{queue} + d_{prop} + d_{trans} \tag{1}$$

where $d_{proc}$, $d_{queue}$, $d_{prop}$ and $d_{trans}$ represent the processing, queuing, transmission and propagation delays, respectively. In practice, $d_{proc}$ is often negligible, since the network devices (*e.g.* peers, routers) have reached high processing speeds. It is assumed that the lengths of the links that connect all the devices in the network are distributed uniformly in our simulation. Put another way, $d_{prop_i} = d_{prop_j}$ for every $i, j$, where $i \neq j$. Furthermore, in our simulation it is guaranteed that the traffic intensity [36] of whole network is not bigger than 1. Therefore, considering all of

these assumptions, Eq.(1) can be rewritten as;

$$d_{nodal} \cong d_{prop} + d_{trans} \tag{2}$$

and $d_{prop}$, $d_{trans}$ are defined as

$$d_{prop} = \frac{l}{v} \quad , \quad d_{trans} = \frac{s}{b} \tag{3}$$

where $l$ denotes the length of the link, $v$ is the propagation speed of the link which is close to the speed of light, $s$ is the size of a packet on link and the $b$ is the bandwidth of the link.

Eq.(2) shows how the delay on only a single link can be calculated. Thus, the total delay between $PID_i$ and $PID_j$ can be calculated as;

$$d_{total_{i\to j}} \cong \sum_{\eta=i\to j} d_{nodal_\eta} = \sum_{\eta=i\to j} d_{prop_\eta} + d_{trans_\eta} \tag{4}$$

$$\cong \sum_{\eta=i\to j} \frac{l_\eta}{v_\eta} + \frac{s_\eta}{c_\eta} \tag{5}$$

In this equation $c_\eta$ denotes the maximum bandwidth value of the corresponding link (in other words the bandwidth value of the link while it is fully empty). However, the available bandwidth value of a link at any time $t$ should be considered also. Therefore, the equation should be revised as in Eq.(6).

$$d_{total_{i\to j}} \cong \sum_{\eta=i\to j} \frac{l_\eta}{v_\eta} + \frac{s_\eta}{c_\eta(101 - \%\mu_{\eta(i\to j)})} \tag{6}$$

where, $\%\mu_{\eta(i\to j)}$ represents the percentage utilization on any link from $PID_i$ to $PID_j$. The reason why 101 is used, rather than 100 is to solve the zero denominator problem at the denominator of the equation in case of at least a link is being fully utilized.

#### *4.2.2.2 Distance Based(DB) Peer Ranking Cost Calculation Method*

As an alternative to the proposed method above, a simple cost calculation method is also used in this work. In this method ALTO Server simply calculates the physical

hop number between the requester peer and the candidate peers until reaching it and then assigns ordered cost values to the candidate peers according to the method explained in [37].

# CHAPTER V

# PERFORMANCE EVALUATION

## *5.1 Simulation Setup*

### 5.1.1 Network Model

To evaluate the performance of the overall system, see the effect of the ALTO service and compare the performance of cost calculation methods which we define in the previous chapter, we conduct experiments over a simulated network in OPNET [38] environment. The simulated network consists of 3 ISPs, each with 260 peers (780 peers in total) and all the peers run our proposed live scalable video streaming application. The backbone links are arranged satisfactorily so that the links connecting the peers to the network may only be subject to congestion. A video streaming server is placed at each ISP and the application tracker is placed to ISP0. Furthermore, in each ISP there is an ALTO Server whose duty is to register the network map of the corresponding ISP and create the cost values between PIDs. In our simulations, to avoid quality degradations due to the packet losses reliable communication links that have zero error probability are used. Also our application is implemented over the UDP/IP protocol stack, and any NAT or firewall issues that may limit the connectivity of the peers are not considered. Furthermore, we assume that peers can measure and know their uplink/downlink bandwidths accurately. The bandwidth distribution of peers that is also shown in Table 3 is arranged according to the latest report of Akamai [39].

Figure 20 shows the proposed network model. In our model, there are 260 peers distributed with different hop numbers in each ISP and in total there are 780 peers in three ISPs, as indicated before. Table 4 groups the peers according to the distance

Table 3: Bandwidth Distribution of Peers

| Download(Mbps) | Upload(Mbps) | Percentage(%) |
|---|---|---|
| 0.5 | 0.25 | 5 |
| 1 | 0.5 | 10 |
| 8 | 1 | 20 |
| 20 | 3 | 50 |
| 20 | 5 | 15 |

of the peers to the central router of the corresponding ISP, in terms of hop number. Furthermore, all three ISPs have the same peer distribution. Also for simplicity, all three ISPs have tree based physical topologies. Put another way, there is only one path from any peer to any peer in the network. Thus, multiple routing path problems are ignored in this work.

Table 4: Number of Peers with Different Hop Distances

| Hop Number | Number of Peers |
|---|---|
| 3 | 90 |
| 4 | 150 |
| 5 | 240 |
| 6 | 210 |
| 7 | 90 |

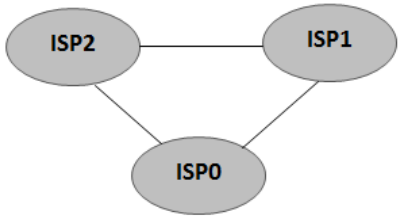


Figure 20: Proposed Network Model

In our simulations, all peers are active during the simulations. In other words, if a peer joins the system, it doesn't leave from the overlay network until the end of the video or the end of the simulation. The performance of the system is evaluated for three different scenarios. In first two scenarios, our ALTO Service enabled P2P

live scalable video streaming protocol is simulated with MDB and DB cost calculation methods, respectively. In the last scenario, ALTO Service is disabled in order to show the benefits of ALTO information.

### 5.1.2 Test Video

Figure 21: A snapshot taken from the test video.

An 854x480 (or 480p) format video which is available at [40] is used in the simulations. The video is the trailer of the *Sintel* which is a short animation movie and a snapshot taken from the video is shown in Figure 21. The uncompressed size of the video is 735 Mb and the video's length is 52 seconds. The video is encoded with SVC [41] in SNR scalability mode with Medium Grain Scalability (MGS) by using the reference software Joint Scalable Video Model (JSVM) of SVC project of Joint Video Team (JVT) [42]. Encoding parameters of general video, base and enhancement layers are shown in Table 5, 6 and 7 respectively.

Table 5: General encoding parameters of the test video.

| **Name** | **Value** | **Description** |
|---|---|---|
| FrameRate | 24 | Maximum frame rate in Hz |
| FramesToBeEncoded | 1253 | Indicates the number of frames of the input video |
| GOPSize | 16 | GOP(group of pictures) size |
| BaseLayerMode | 1 | Indicates whether base layer is coded as 1: AVC compatible without SEI(Supplemental Enhancement Information) messages or 2: AVC compatible with SEI messages |
| IntraPeriod | 32 | Period of intra coded (I) frames |
| CgsSnrRefinement | 1 | Indicates SNR enhancement layers are encoded using whether 1: MGS or 0: CGS |
| EncodeKeyPictures | 1 | 0: no picture of temporal level 0 are encoded as key pictures, 1: pictures refined with MGS are encoded as key pictures, 2: all pictures at temporal level 0 are encoded as key pictures |
| MGSControl | 2 | Indicates which frames are used as references motion estimation (ME) and compensation(MC) 0: frames of current MGS layer are used, 1: frames of highest EL are used for ME and frames of current layer are used for MC, 2: frames of highest EL are used for ME and MC (EL means enhancement layer) |
| SearchMode | 4 | Indicates the motion search algorithm 0: block search, 4: fast search |
| SearchRange | 32 | Indicates maximum search range for motion search |
| NumLayers | 2 | Indicates the number of spatial or SNR scalable layers |

Table 6: Encoding parameters of the base layer.

| **Name** | **Value** | **Description** |
|---|---|---|
| SourceWidth | 854 | The width of the input frames in luma samples |
| SourceHeight | 480 | The height of the input frames in luma samples |
| FrameRateIn | 24 | The frame rate of the input video |
| FrameRateOut | 24 | Indicates the output frame rate of the current layer |
| QP | 55 | Indicates the quantization parameter for the layer |
| MGSVectorMode | 0 | Indicates whether additional quality layers are inserted (1) or not (0) |

Table 7: Encoding parameters of the enhancement layer.

| **Name** | **Value** | **Description** |
|---|---|---|
| SourceWidth | 854 | The width of the input frames in luma samples |
| SourceHeight | 480 | The height of the input frames in luma samples |
| FrameRateIn | 24 | The frame rate of the input video |
| FrameRateOut | 24 | Indicates the output frame rate of the current layer |
| QP | 35 | Indicates the quantization parameter for the layer |
| InterLayerPred | 1 | Inter-layer prediction (0: no, 1: yes, 2: adaptive) |
| MGSVectorMode | 1 | Indicates whether additional quality layers are inserted (1) or not (0) according to MGSVectorX |
| MGSVector0 | 4 | Indicates the number of transform coefficients that are written to the first additional quality layer |
| MGSVector1 | 4 | Indicates the number of transform coefficients that are written to the second additional quality layer |
| MGSVector2 | 8 | Indicates the number of transform coefficients that are written to the third additional quality layer |
| BaseLayerId | 0 | Indicates which layer is used for inter-layer prediction |

Basically the encoded video consists of one base and one enhancement layer. Furthermore, enhancement layer consists of 3 additional layers, since MGS encoded video consists of infinitely many enhancement layers and we selected 3 of them. Therefore, the encoded video has 4 scalable layers in total. Figure 22 shows the rate distortion curve of the encoded video.

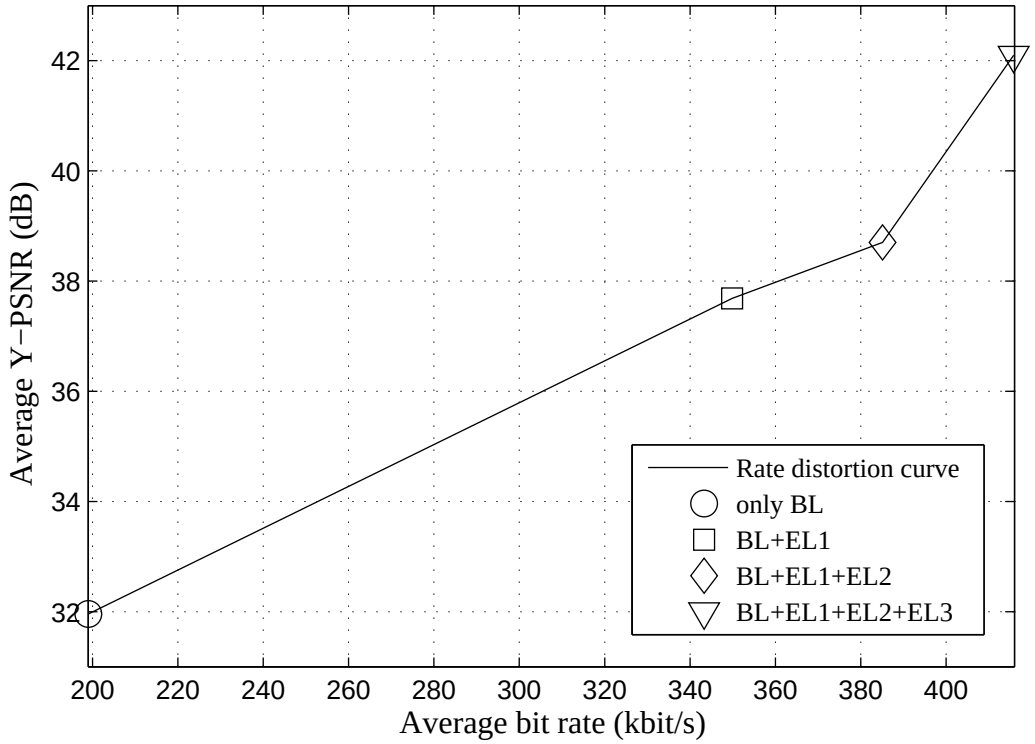


Figure 22: Rate distortion curve of the encoded video

## *5.2 Simulation Results*

### 5.2.1 Determination of Parameters

#### *5.2.1.1 Scheduling Parameters*

In order to find the optimum operation point of our scheduling algorithm in terms of buffer length and scheduling period, we conducted experiments with different parameter values and obtained results in terms of pre-roll delay, number/duration of pauses and PSNR. Experiments were made on 15 operation points while the buffer length takes values of 14, 20, 26, 32, 38 seconds and the scheduling period takes values of 2.5, 5, 7.5 seconds. ALTO service is active in all of these simulations.

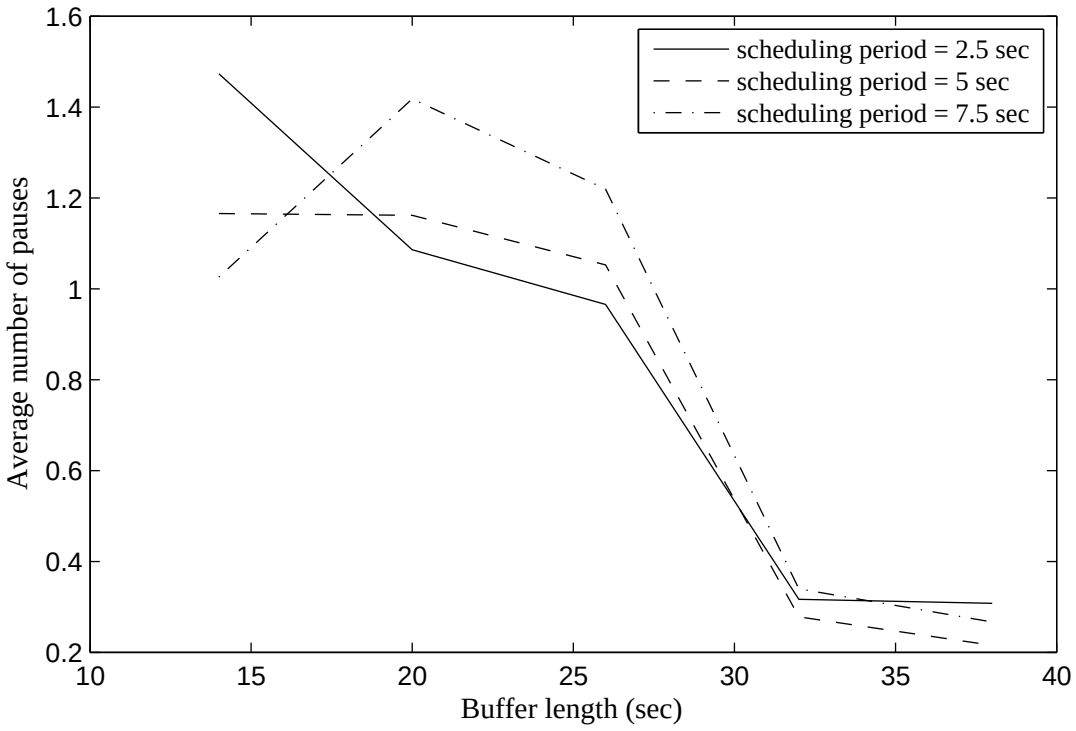


Figure 23: Average pause number of peers for different scheduler parameters

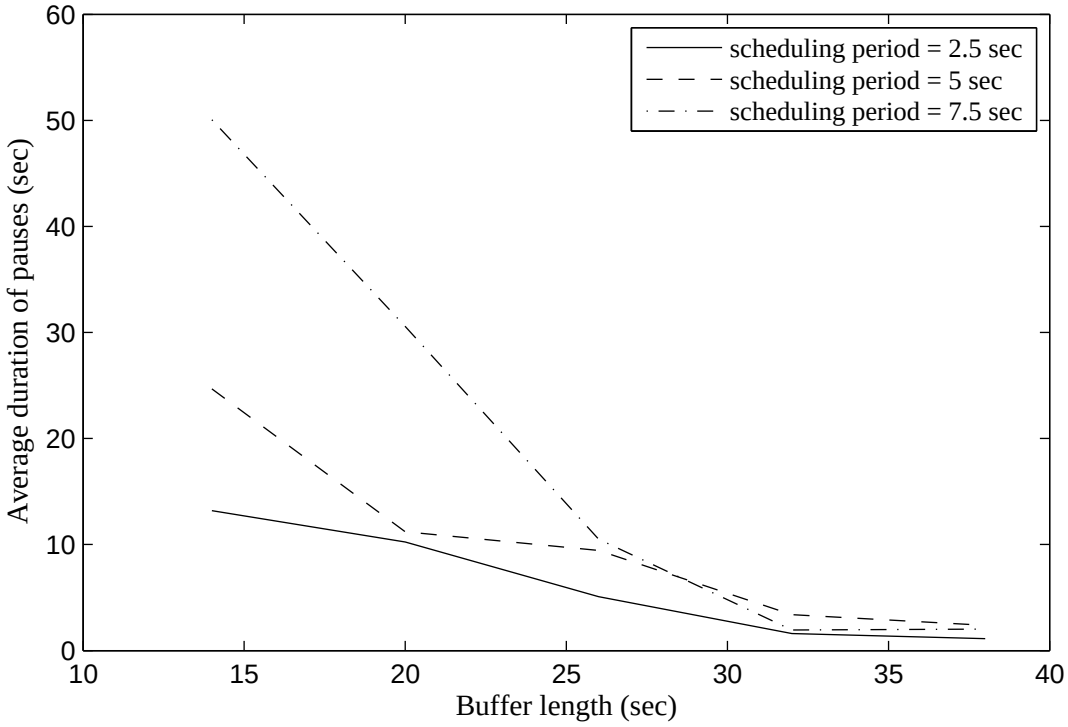


Figure 24: Average pause duration of peers for different scheduler parameters

Change in average pause numbers and durations of peers for different scheduling parameters are shown in Figures 23 and 24, respectively. As expected, the number/duration of pauses decreases with increasing buffer length. But when scheduling period is 7.5 sec. the number of pauses increases when the buffer length is between 14 and 20 seconds. Actually this is also reasonable since, when buffer length is 14 seconds, although the number of pauses is nearly 1, this one pause takes 50 seconds and when buffer length is 20, the number of pauses reaches 1.4 and pauses last for a total of 30 seconds. So in general, pauses decrease with increasing buffer length. Furthermore, as can be seen in the figures, number/duration of pauses decreases with decreasing scheduling period as expected. This positive effect of small scheduling period decreases with increasing buffer length.

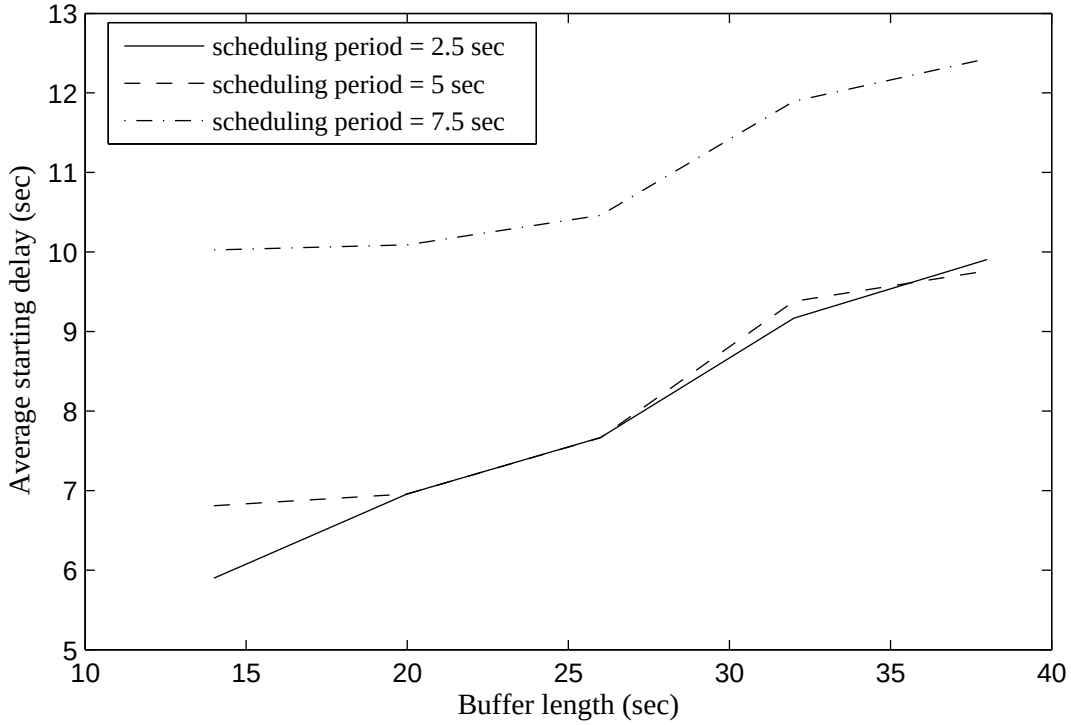


Figure 25: Average starting delay of peers for different scheduler parameters

Figure 25 shows average starting delay of peers for selected operation points. As expected, starting delay increases with increasing buffer length as can be seen in the figure. Similarly, starting delay increases with increasing scheduling period as well. Furthermore, change in pre-roll delays is too low while scheduling period changes from 2.5 to 5. This is reasonable since after a certain value even if scheduling period is decreased, pre-roll delay doesn't change much due increasing unnecessary traffic load.

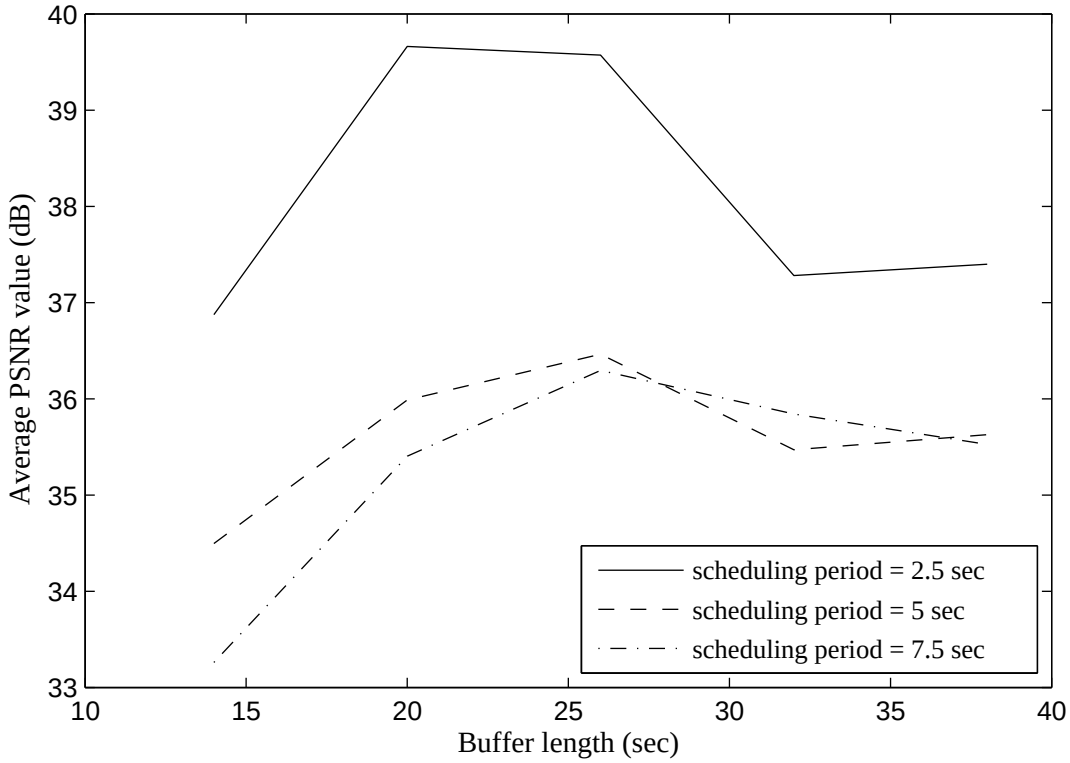


Figure 26: Average PSNR value of peers for different scheduler parameters

Average PSNR value of peers for selected operation points is shown in Figure 26. As can be seen in the figure, PSNR value increases with decreasing scheduler period. When the scheduler period is constant, the PSNR value increases with increasing buffer length until the buffer becomes large enough to provide sufficiently small number of pauses. After this point, the PSNR value decreases with increasing buffer length since the number of base layer segments that must be requested before any enhancement layer segments increases with increasing buffer length as well.

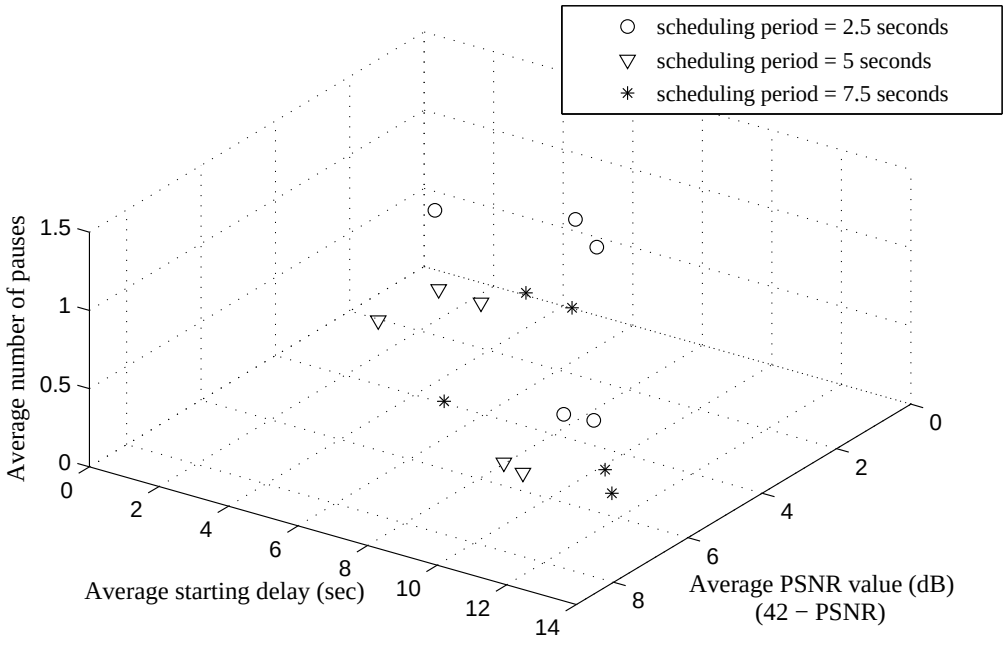


Figure 27: Illustration of operation points in 3D space of simulation results (z-axis shows average number of pauses)

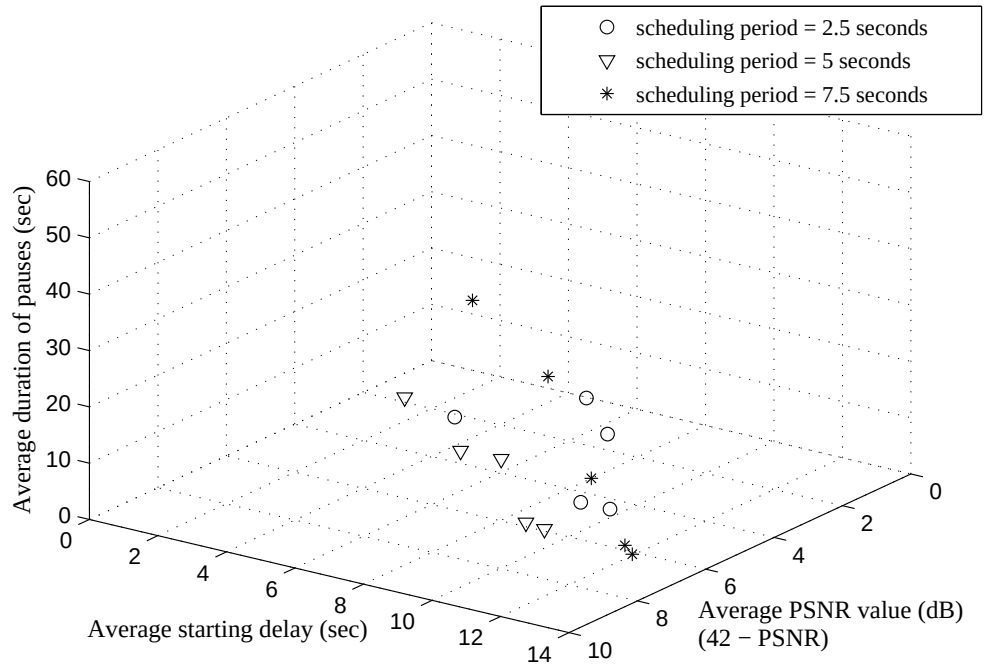


Figure 28: Illustration of operation points in 3D space of simulation results (z-axis shows average duration of pauses)

As can be seen in Figure 27, in order to find optimum operation point we located all operation points in a 3D space whose x-axis shows difference between the best PSNR value for the video and average PSNR value for the operation point, y-axis shows average starting delay for the operation point and z-axis shows average number of pauses for the operation point. As you can see, in this 3D space the origin represents the ideal operation point for the application (Utopia Point in Multi-Objective Optimization). Thus, the closest operation point to the origin is the optimum operation point (best compromise point)for the application. Therefore, we calculated the distance from each operation point to the origin and found that operation point with the 20 sec. buffer length and 2.5 sec. scheduling period optimal. We repeated the same calculations again when the z-axis of the 3D space shows the average duration of pauses instead of average number of pauses. In this case, we found that operation point with the 26 sec. buffer length and 2.5 sec. scheduling period is optimal. Figure 28 shows the 3D space while z-axis shows average duration of pauses.

Table 8: Simulation results for the two optimum operation points

| | Buffer length = 20 sec<br>Scheduling period = 2.5 sec | Buffer length = 26 sec<br>Scheduling period = 2.5 sec |
|---|---|---|
| Number of pauses | 1.1 | 0.9 |
| Duration of pauses (sec) | 10.2 | 5.1 |
| Starting delay (sec) | 7 | 7.7 |
| PSNR (dB) | 39.7 | 39.6 |

Simulation results for the two optimum operation points is shown in Table 8. As can be seen in the table, duration of pauses for the point with 26 sec buffer length is approximately half of the other one and changes in the remaining statistics are very small between the two points. Therefore, we selected the point with 26 sec buffer length and 2.5 sec scheduling period as our optimum operation point. Here it is important note that in this optimization process the pause duration statistic have priority over the remaining statistics since its range is larger than the other ones. To

provide equal priority all statistic must be scaled as they have equal ranges. In our work we didn't scale the statistics since we think that pause duration is one of the most important statistics for a live video streaming application maybe the most important one. Furthermore, we observe that when all statistics have equal importance, in the resulting optimum operation point, the average duration of pauses is doubled while the remaining statistics didn't change much.

If we move from the optimum operation point to the another one due to a reason such as using a different test video to find the optimum point, performance of the application decreases. In order to see how much the performance is effected from this movement, we did a sensitivity analysis on our simulation results. For this purpose, while keeping constant one of the scheduling parameters, we observed how the performance changes in terms of duration of pause, pre-roll delay and PSNR with changing the remaining parameter.

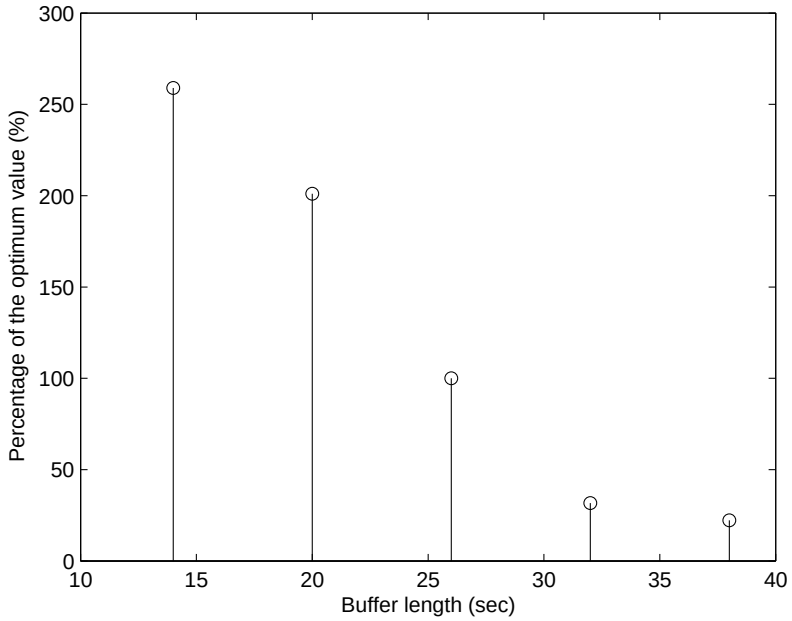


Figure 29: Sensitivity of average pause duration of the peers to the buffer length

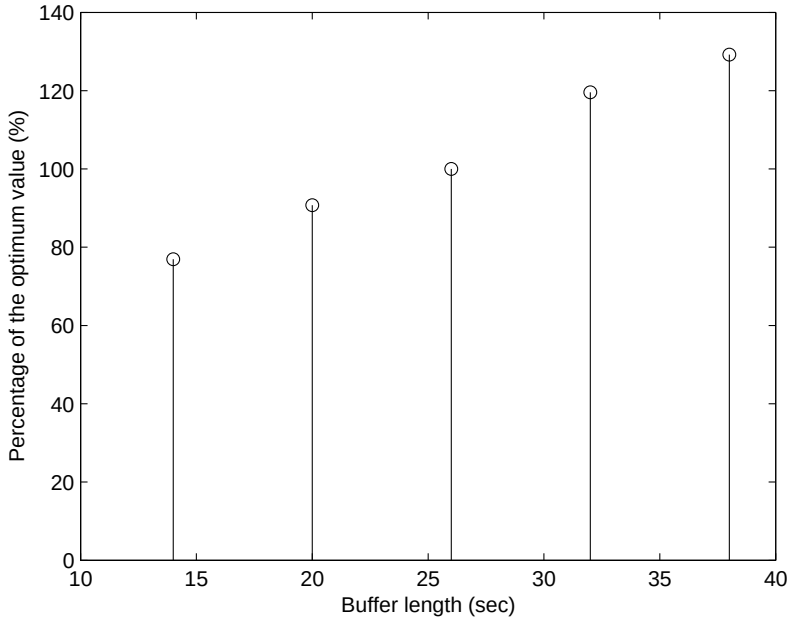


Figure 30: Sensitivity of average pre-roll delay of the peers to the buffer length

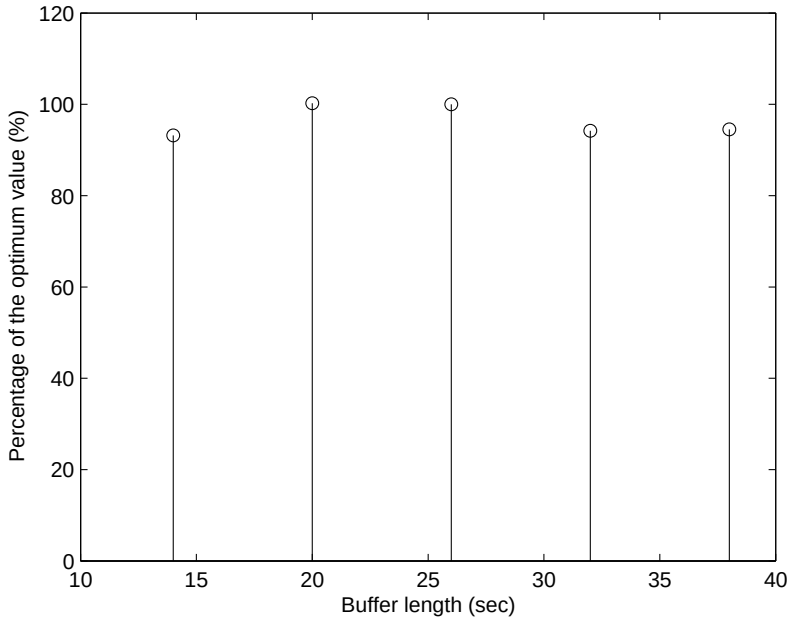


Figure 31: Sensitivity of average PSNR value of the peers to the buffer length

Figures 29, 30, 31 illustrate the sensitivity of the application performance to the buffer length. As can be seen from the figures, average pause duration of the peers

is very sensitive to the buffer length while average pre-roll delay and PSNR value of the peers are not effected much from the buffer length.

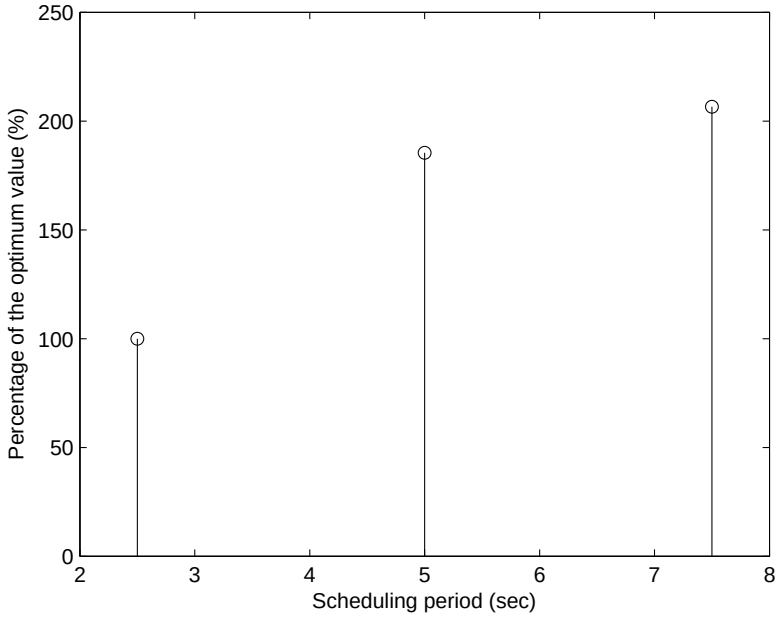


Figure 32: Sensitivity of average pause duration of the peers to the scheduling period

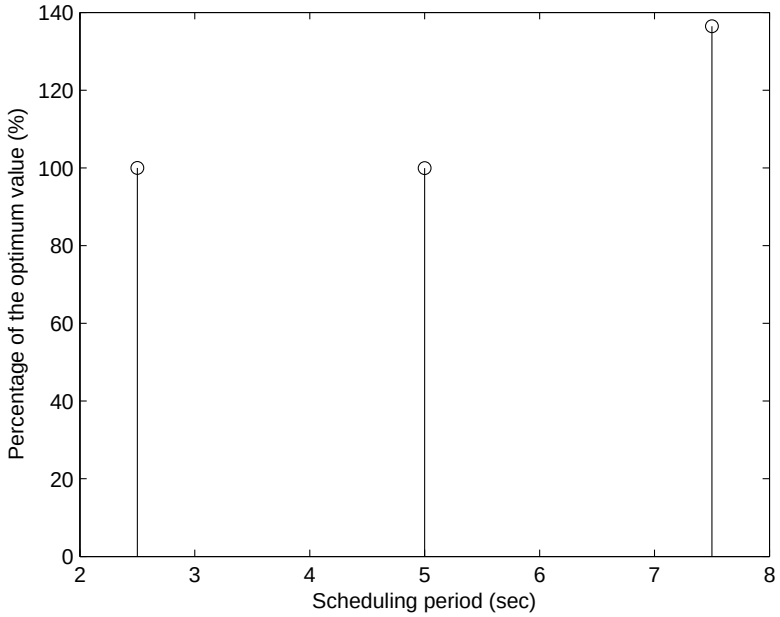


Figure 33: Sensitivity of average pre-roll delay of the peers to the scheduling period

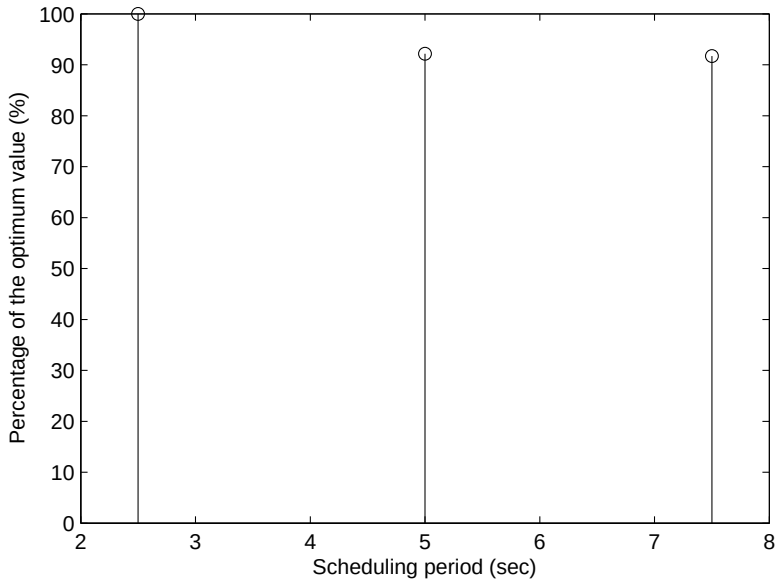


Figure 34: Sensitivity of average PSNR value of the peers to the scheduling period

Similarly, Figures 32, 33, 34 illustrate the sensitivity of the application performance to the scheduling period. According to the figures, average pause duration of the peers is very sensitive to the scheduling period while average pre-roll delay and PSNR value of the peers are not very sensitive to the scheduling period.

#### *5.2.1.2 Maximum Partner Number*

As mentioned in Section 3.3, we made extensive simulations in order to find the optimum value for maximum number of partners that a peer is allowed to have. We conducted simulations when this parameter is varied between 3 and 7 and obtained performance results in terms of pre-roll delay, number/duration of pauses and PSNR. In all of these simulations, ALTO service is active.

Average number of pauses, duration of pauses, starting delay and PSNR value of peers for varying maximum partner numbers are shown is Figures 35, 36, 37 and 38 respectively. As can be seen in the figures, the performance of the application improves significantly in terms of all QoE statistics until the maximum partner number reaches the value of 5. After this value, changes in these statistics become marginal.

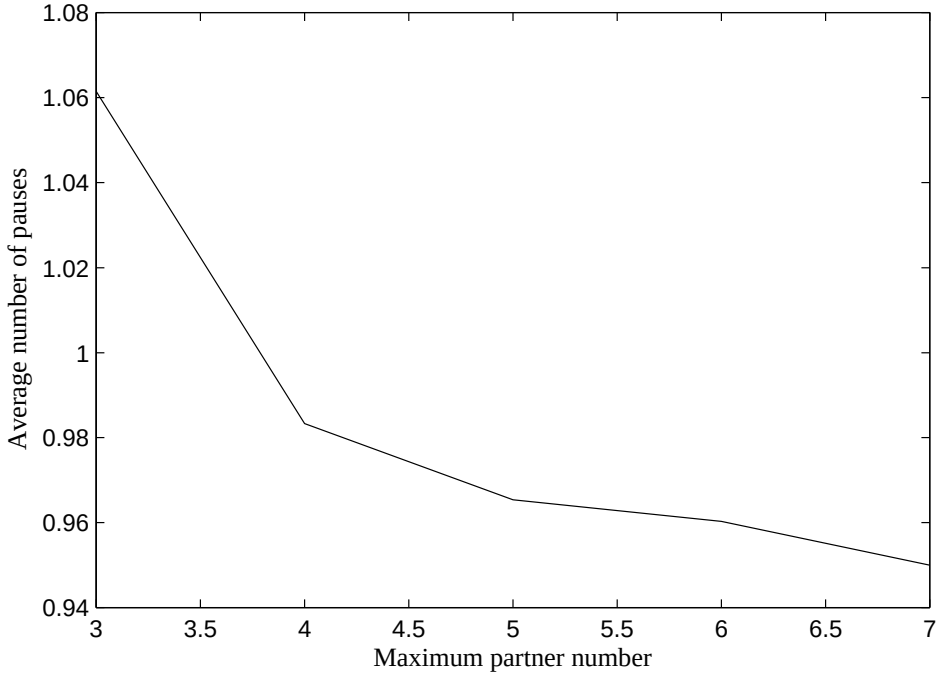


Figure 35: Average pause number of peers for varying maximum partner number

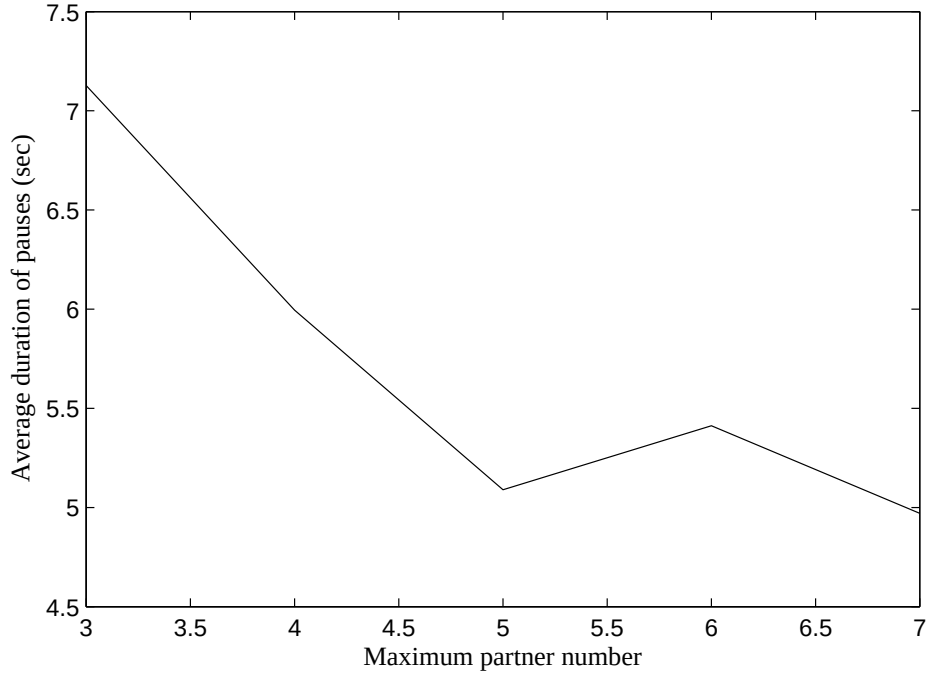


Figure 36: Average pause duration of peers for varying maximum partner number

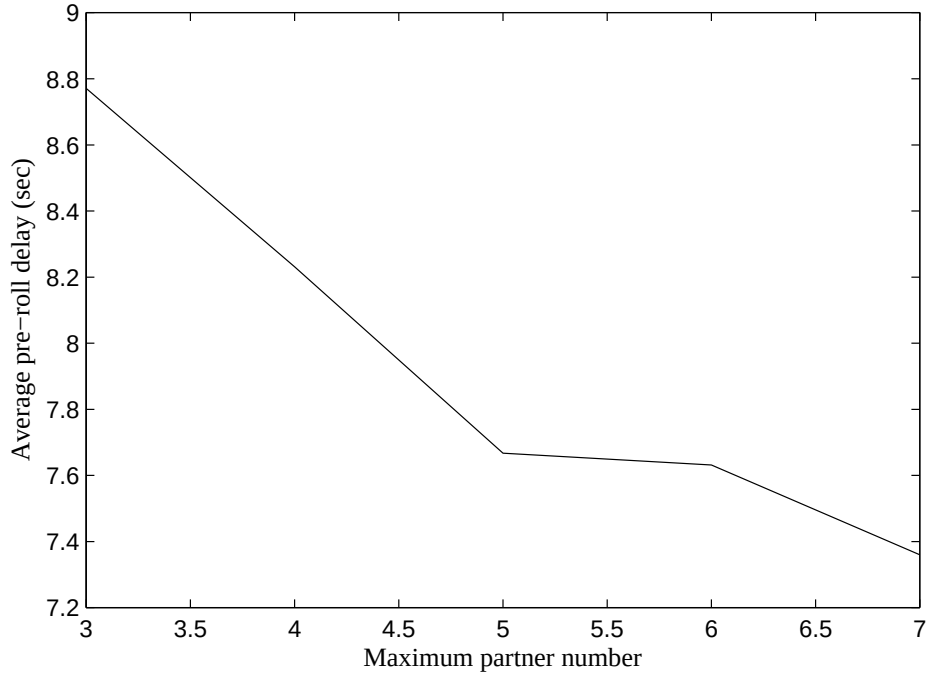


Figure 37: Average pre-roll delay of peers for varying maximum partner number

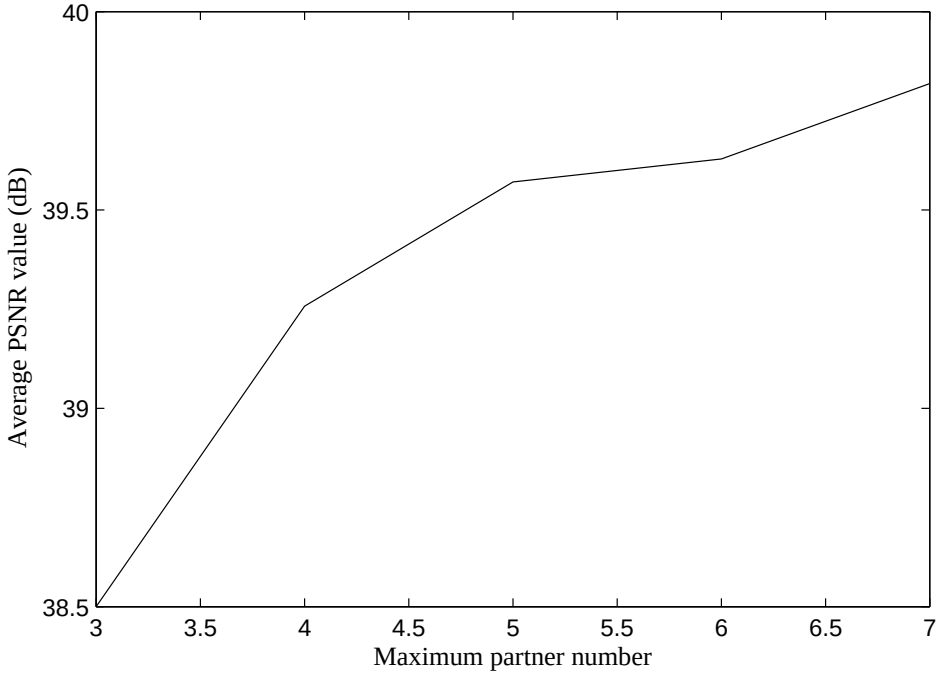


Figure 38: Average PSNR value of peers for varying maximum partner number

Therefore, we selected 5 as the optimum value of maximum partner number.

### 5.2.2 Effectiveness of ALTO

In this section, simulation results for the optimum operation point that is found in previous section are presented and discussed under the titles of the average video download and upload rates, video pause duration of the peers, streamed video quality of the peers in terms of PSNR and inter-ISP traffic rates in order to show effect of ALTO on application performance.

#### *5.2.2.1 Bandwidth Utilization of Peers*

The average download and average upload speeds of all peers are shown in Figures 39 and 40. As can be seen in Figure 39, average download rate of the peers is not effected much from ALTO. Although ALTO service decreases the rate in the first 20 sec especially with hop count ranking algorithm, after 20th second the rate stays at nearly the same level in all three scenarios. With the hop count ranking algorithm, the ALTO server suggests peers that are the closest peers to the requester even if paths to them are congested. This situation explains the lower rate values of hop count ranking scenario in the first 20 seconds. Also as can be seen from the same figure, in all scenarios, for time $t > 58$ sec. average download rate of the peers is very low, since most of the peers finish streaming by then.

Similarly as can be seen in Figure 40, average upload rate is not effected much from ALTO Service. Also, in all scenarios, for time $t > 58$ sec. the average upload rate is very low like the average download rate, since most of the peers have completed streaming by then.

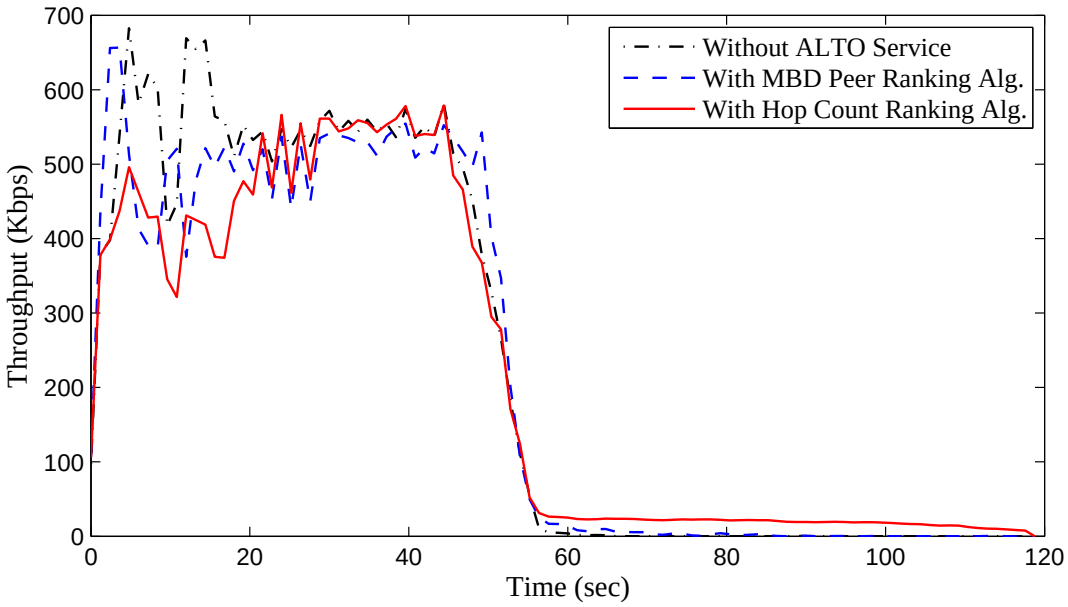


Figure 39: Average Video Download Rates of Peers

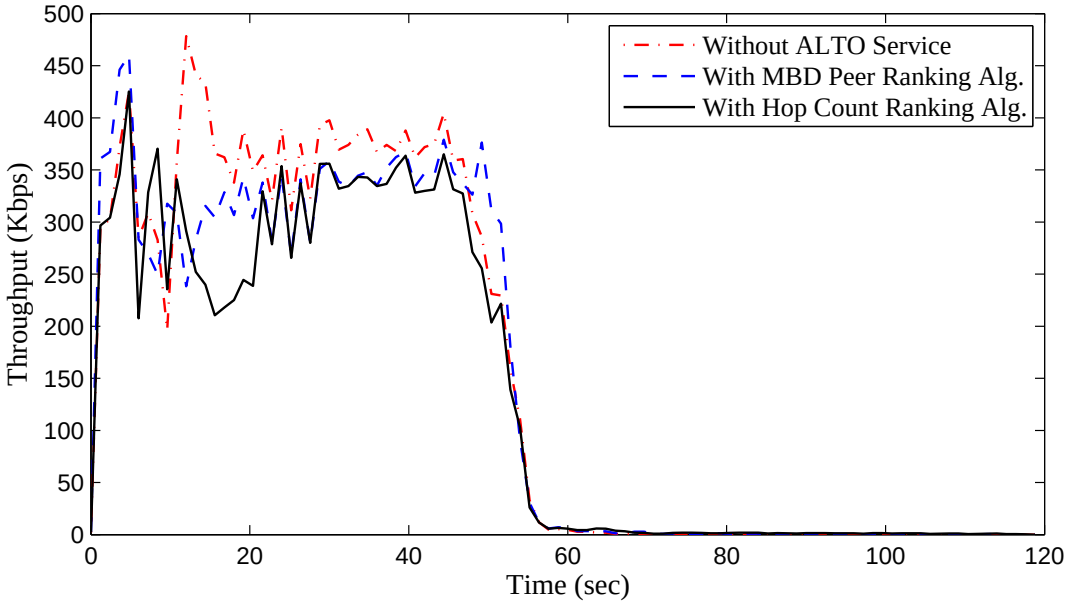


Figure 40: Average Video Upload Rates of Peers

#### 5.2.2.2 *Streaming Starting Time Distribution*

The CDF of streaming starting times are shown in Figure 41. As can be seen in the figure, ALTO Service has a positive effect on streaming starting time. In both ALTO enabled scenarios, approximately 90% of peers start to streaming in 10 seconds while 77% of peers start in ALTO disabled scenario. On the other hand, all of peers have less than 21 seconds buffering delay when ALTO service is not active, while there are peers that have greater than 30 seconds buffering delay when ALTO service is active.

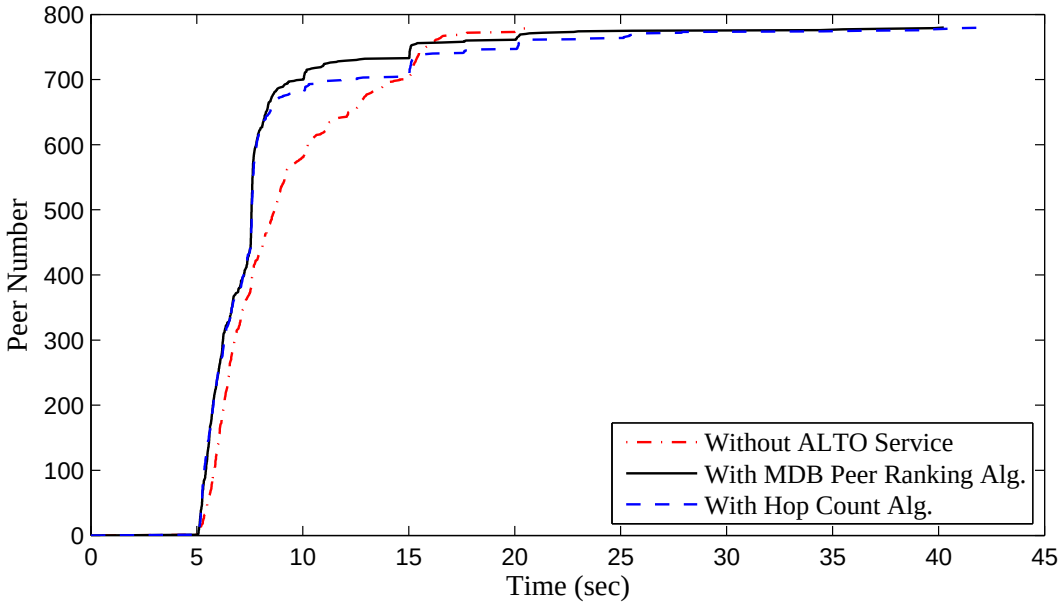


Figure 41: CDF of Streaming Starting Times of the Peers

#### 5.2.2.3 *Pause Duration Distribution*

The CDF of the pause duration of the peers is shown in Figure 42. As can be seen from the figure, the CDFs of all three scenarios are very similar. On the other hand, number of the peers that don't experience any pause during the streaming in ALTO disabled scenario is greater than number of peers with no pauses in ALTO active scenarios. Also according to this figure, approximately 90% of peers experience pauses less than 10 seconds.

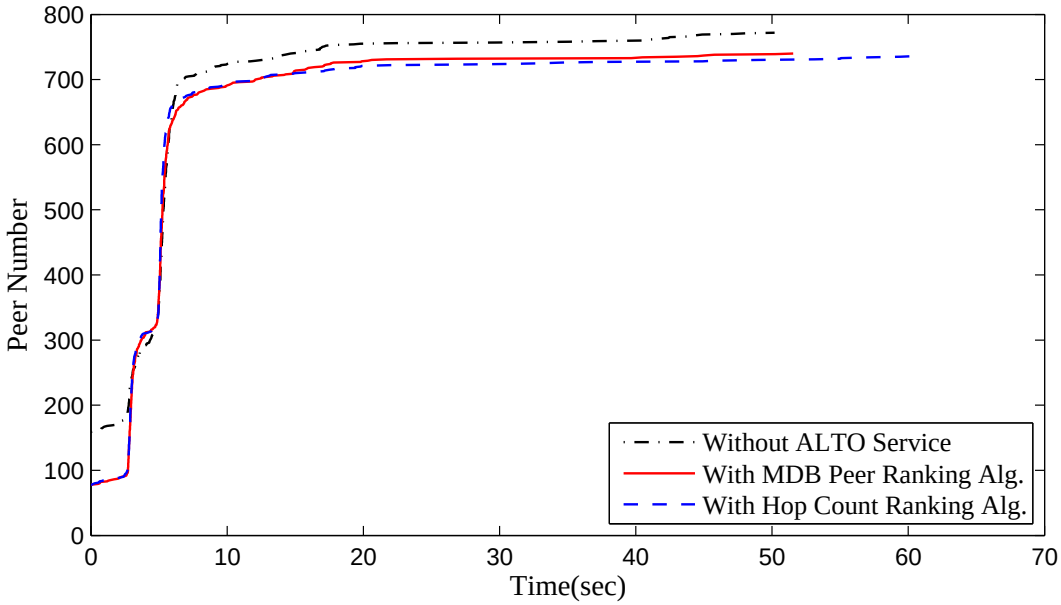


Figure 42: CDF of the Pause Durations of the Peers

#### 5.2.2.4 *Streaming Completion Time Distribution*

Figure 43 shows the CDF of the streaming completion time of the peers. As can be seen from the figure, on the average peers complete their streaming sessions with low delay values (less than 70 sec.) in all three scenarios. On the other hand, with the ALTO service enabled, the number of peers that can watch the whole video decreases slightly.

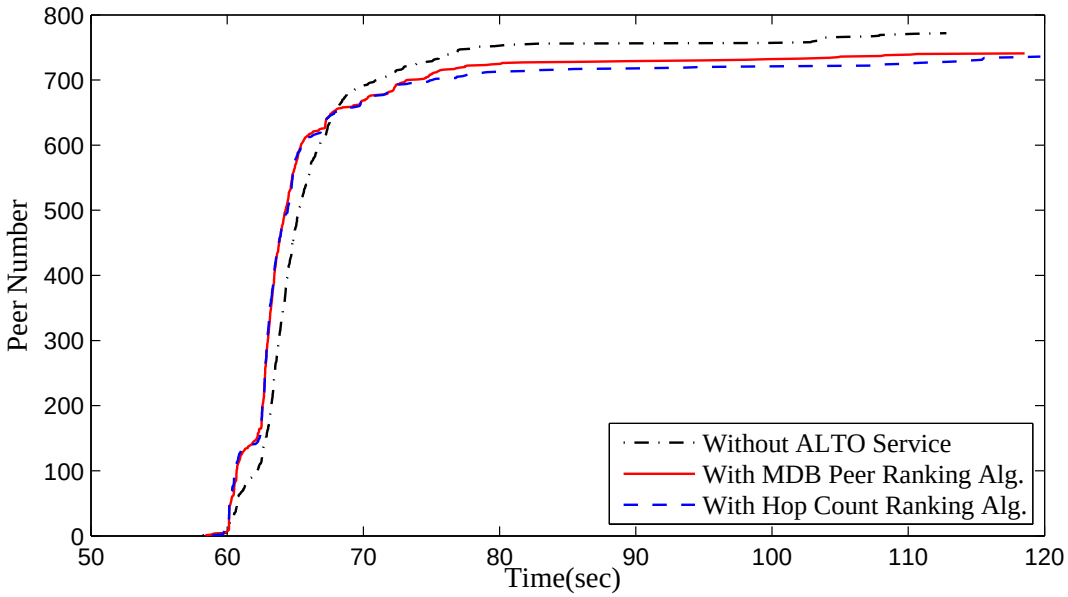


Figure 43: CDF of Streaming Completion Times of the Peers

#### *5.2.2.5 Received Video Quality*

To investigate QoE of the system, the Peak-Signal-to-Noise-Ratio (PSNR) values are calculated for each scenario and each peer that watches the video completely. Table 9 shows maximum, minimum and average PSNR values for each scenario. As can be seen from the table, when the ALTO service is enabled - with MDB or DB method - the average streaming quality of the peers decreases marginally - less than 1 dB - and maximum PSNR values shows that there are peers which can obtain all layers completely in all three scenarios. Here it is important to note that these values are calculated while pauses are not taken into account.

Table 9: PSNR Value Statistics

| dB | Without ALTO | With MDB method | With DB method |
|---|---|---|---|
| *Maximum* | 42.1003 | 42.1003 | 42.1003 |
| *Minimum* | 36.4868 | 32.8844 | 32.9124 |
| *Average* | 40.2258 | 39.5706 | 39.4491 |
| *Variance* | 1.7875 | 2.4438 | 2.6736 |

#### *5.2.2.6 Inter-ISP Traffic*

Table 10 shows time averages of inter-ISP traffic rates between three ISPs for each direction and each scenario. As can be seen from the table, inter-ISP traffic rates are very low in ALTO enabled scenarios, especially with MDB method. According to the table, using the ALTO service with the MDB method, the inter-ISP traffic rates is reduced 96%, approximately.

Table 10: Time Averages of Inter-ISP Traffic Rates

| Direction | Without ALTO | With MDB method | With DB method |
|---|---|---|---|
| $ISP0 \rightarrow ISP1$ | 40.53% | 2.28% | 6.16% |
| $ISP0 \leftarrow ISP1$ | 43.93% | 2.57% | 5.60% |
| $ISP0 \rightarrow ISP2$ | 42.71% | 2.53% | 5.99% |
| $ISP0 \leftarrow ISP2$ | 40.99% | 2.45% | 5.82% |
| $ISP1 \rightarrow ISP2$ | 41.71% | 2.32% | 5.77% |
| $ISP1 \leftarrow ISP2$ | 42.06% | 2.47% | 5.94% |

Figures 44, 45, 46 shows inter-ISP traffic - averaged over both directions of upload and download - in terms of utilization for each scenario between ISP 0 and ISP 1, between ISP 0 and ISP 2, between ISP 1 and ISP 2, respectively.

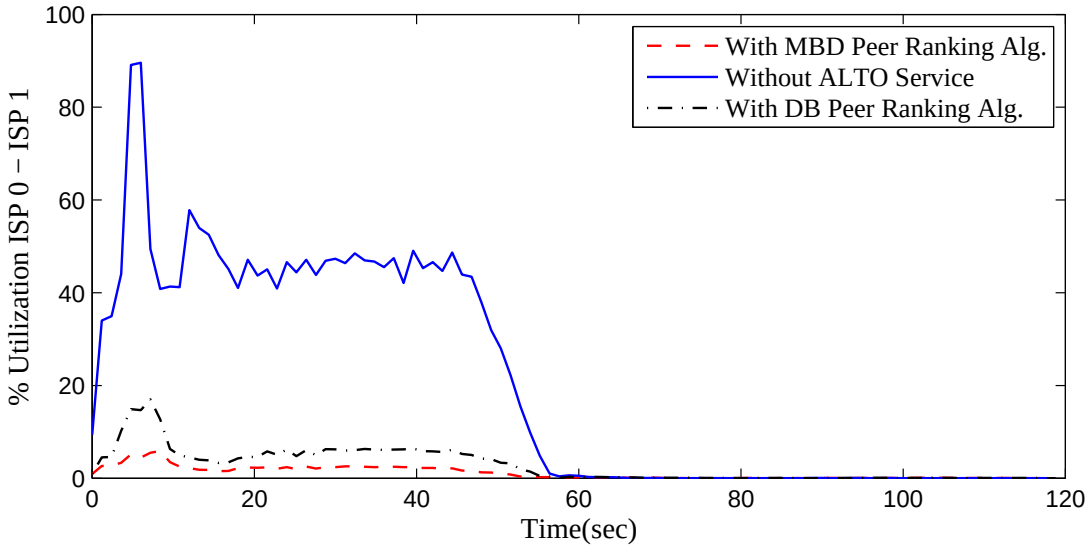


Figure 44: Inter-ISP traffic between ISP 0 and ISP 1

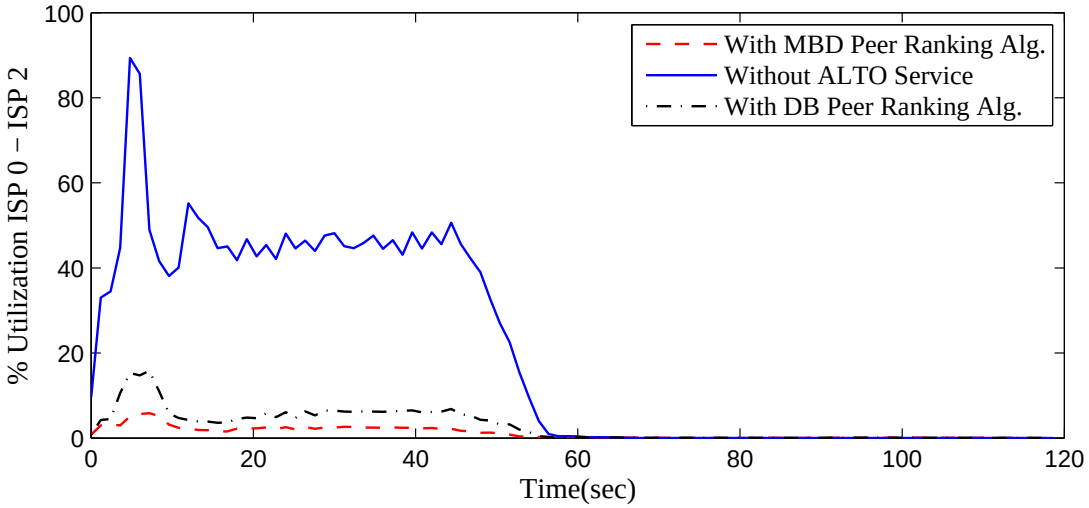


Figure 45: Inter-ISP traffic between ISP 0 and ISP 2

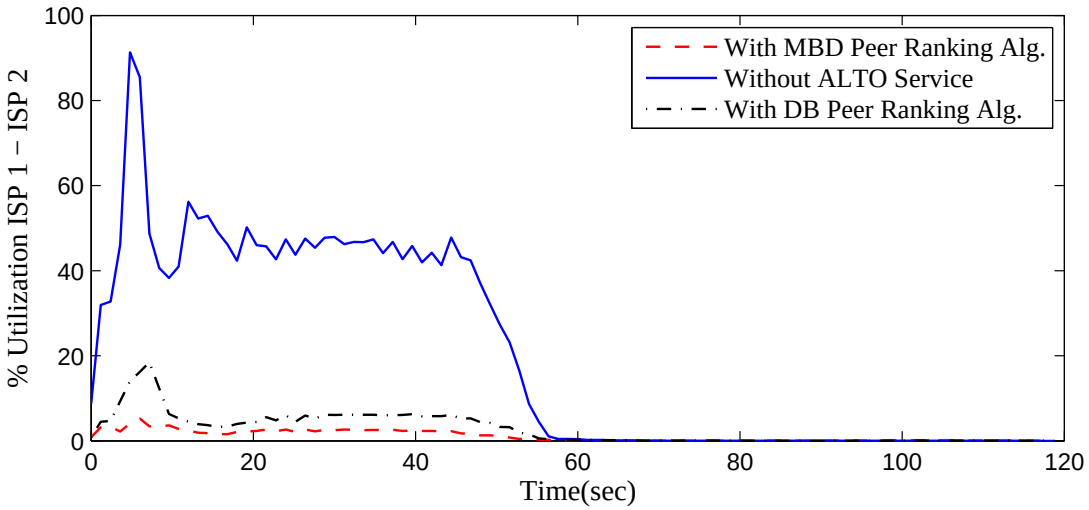


Figure 46: Inter-ISP traffic between ISP 1 and ISP 2

The great reduction in the inter-ISP traffic due to ALTO service can be seen from the figures. As can be seen from the figures inter-ISP utilization takes high values around 5th second especially in ALTO disabled scenario. This situation can be explained as follows: around the 5th second, all peers are in their buffering stage so they all send requests and receive packets in order the fill 25% of their base layer buffers. Furthermore, when the ALTO service is not active, they send these requests to peers from the other ISPs more likely and generate large amounts of inter-ISP

traffic. Moreover, in the buffering stage after the first 5 seconds, a peer can request a base layer segment from a randomly selected server if the segments is not available in all of its partners. Since the server selection is random, these requests also generate large amount of inter-ISP traffic.

# CHAPTER VI

# CONCLUSION

P2P applications are very popular on the Internet and many P2P video streaming services have emerged since P2P networks showed lots of benefits and strengths. However, P2P networks also have noticeable drawbacks and their network-oblivious nature is one of them. This network-oblivious nature effects both network providers and end-users in a negative way. It causes inefficient usage of network providers' resources and low QoE to end-users. Furthermore this problem increases inter-ISP traffic and thus it effects the economics of ISPs badly. Furthermore, video streaming also has important challenges like supporting streams with different qualities for users with different rates.

In this thesis, we introduce a new P2P live scalable video streaming protocol, which provides a good QoE to the users with different rates, and apply ALTO protocol to our streaming protocol in order to prevent negative effects of P2P applications' network-oblivious nature. In the ALTO protocol, we proposed to use the MDB and DB methods for cost calculation. We conducted extensive simulations to evaluate the performance of the system and observe the effect of ALTO protocol on the system.

Simulation results show that ALTO protocol maintains the performance of the system in terms of QoE metrics like average download/upload rates, pause durations, pre-roll delays, while decreasing the inter-ISP traffic rates dramatically. Actually, inter-ISP traffic rates are reduced 96% approximately. Most importantly, ALTO protocol maintains the QoE of the users, while providing all these improvements. Reduction in the average PSNR value of the peers by ALTO protocol is less than 1 dB.

# CHAPTER VII

# FUTURE WORK

Although the results are very promising, new peer selection methods should be developed for improving the performance further. Also, introducing an incentive mechanism to the protocol can be considered as a next goal. Moreover, a further study may investigate the performance of the protocol under a dynamic environment (nodes are joining and leaving dynamically). Besides, in a further study, the performance of the proposed protocol may be evaluated in a more realistic environment like PlanetLab in order to assess the behavior in real time conditions.

# Bibliography


[1] A. Parker, "Addressing the cost and performance challenges of digital media content delivery," in $1^{st}$ *P2P Media Summit*, 2006. `http://www.dcia.info/activities/p2pmsla2006/CacheLogic.ppt`.

[2] M. Hofmann and L. Beaumont, *Content Networking: Architecture, Protocols, and Practice*. The Morgan Kaufmann Series in Networking, Elsevier Science, 2005.

[3] "World Wide Web." `http://en.wikipedia.org/wiki/World_wide_web`.

[4] "W3Schools.com." `http://www.w3schools.com/browsers/browsers_stats.asp`.

[5] G. Held, *A Practical Guide to Content Delivery Networks*. Taylor & Francis, 2005.

[6] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, pp. 101–106, Jan. 2006.

[7] A. K. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," tech. rep., Grid Computing and Distributed Systems (GRIDS) Laboratory, University of Melbourne, Australia, 2006.

[8] "Akamai." `http://www.akamai.com`.

[9] "Mirror Image." `http://www.mirror-image.com`.

[10] "Limelight Networks." `http://www.limelightnetworks.com`.

[11] "Coral." `http://www.coralcdn.org`.

[12] "CoDeeN." `http://www.codeen.cs.princeton.edu`.

[13] "Globule." `http://www.globule.org`.

[14] J. Buford, H. Yu, and E. Lua, *P2P Networking and Applications*. Morgan Kaufmann Series in Networking, Elsevier/Morgan Kaufmann, 2009.

[15] X. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking*. Lecture Notes on Coastal and Estuarine Studies, Springer, 2009.

[16] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, pp. 1456 – 1471, October 2002.

[17] "PPLive." `http://www.pplive.com`.

[18] "PPStream." http://www.ppstream.com.

[19] "Zattoo." http://www.zattoo.com.

[20] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of SIGCOMM '02*, pp. 205–217, 2002.

[21] Y. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '04, pp. 12–12, 2004.

[22] T. Özbilgin and O. Sunay, "Kullanc birlikli Telsiz Grevde Alarda leklenebilir Video letimi," in *IEEE SIU 2008*, April 2008.

[23] X. Zhang, J. Liu, B. Li, and T. shing Peter Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming," in *IEEE Infocom*, 2005.

[24] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A peer-to-peer network for live media streaming using a push-pull approach," in *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pp. 287–290, 2005.

[25] N. Magharei and R. Rejaie, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," in *Proceedings of IEEE INFOCOM*, pp. 1424–1432, 2007.

[26] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pp. 351–362, 2008.

[27] "Joost." http://www.joost.org/whatsjoost.html.

[28] "Kontiki." http://www.kontiki.com.

[29] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "Coolstreaming: Design, Theory, and Practice," *IEEE Transactions on Multimedia.*, vol. 9, pp. 1661–1671, Dec. 2007.

[30] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," in *Proc. of INFOCOM*, Apr. 2006.

[31] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pp. 298–313, 2003.

[32] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: eliminating trees from overlay multicast," in *Proceedings of the 4th international conference on Peer-to-Peer Systems*, IPTPS'05, pp. 127–140, 2005.

[33] Y. Gu, N. Zong, H. Zhang, Y. Zhang, J. Lei, G. Camarillo, Y. Liu, D. Montuno, and L. Xie, "Survey of P2P Streaming Applications." `http://tools.ietf.org/pdf/draft-ietf-ppsp-survey-02.pdf`. IETF Internet Draft, last updated in 2011-07-05.

[34] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement." `http://www.rfc-editor.org/rfc/pdfrfc/rfc5693.txt.pdf`. RFC 5693, published in October 2009.

[35] R. Alimi, R. Penno, and Y. Yang, "ALTO Protocol." `http://tools.ietf.org/pdf/draft-ietf-alto-protocol-12.pdf`. IETF Internet Draft, last updated in 2012-07-11.

[36] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*. Pearson, 5 ed., 2010.

[37] Y. Fukushima, K. Inada, Y. Tao, Y. Fujiwara, and T. Yokohira, "Performance evaluation of AS-friendly peer selection algorithms for P2P live streaming," in *Proceedings of the 15th Asia-Pacific conference on Communications*, APCC'09, pp. 788–792, 2009.

[38] "OPNET." `http://www.opnet.com`.

[39] D. Belson, T. Leighton, and B. Rinklin, "The State of the Internet: $1^{st}$ Quarter, 2012 Report," tech. rep., Akamai, 2010.

[40] "Sintel - Trailer." `http://media.xiph.org/video/derf/y4m/sintel_trailer_2k_720p24.y4m`.

[41] H. Schwarz, D. Marpe, and T. Wieg, "Overview of the scalable video coding extension of the h.264/avc standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1103–1120, 2007.

[42] JVT and ITU-T, *JSVM Software Manual*, 9.19.14 ed., June 2011.

# VITA

İsmail Serkan Kırkgül was born in 1987. He graduated from Antalya Aldemir- Atilla Konuk Anadolu High School in 2006. In 2010, he received his B.Sc. degree from İstanbul University in Department of Electrical and Electronics Engineering. His graduation project was about *Location Estimation in Wireless Communication Networks* and his advisor was Prof. Dr. Hakan Ali Çırpan. He joined Özyeğin University in September 2010 where he is currently an M.Sc. student. He is part of the P4P Project team funded by Türk Telekom. His advisor is Assoc. Prof. Dr. M. Oğuz Sunay.