Ryerson University
Digital Commons @ Ryerson

Theses and dissertations

1-1-2010

A peer-to-peer delivery system for internet short video sharing

Maryam Bashardoust Tajali
Ryerson University

Follow this and additional works at: http://digitalcommons.ryerson.ca/dissertations

Part of the Electrical and Computer Engineering Commons

Recommended Citation
Tajali, Maryam Bashardoust, "A peer-to-peer delivery system for internet short video sharing" (2010). *Theses and dissertations.* Paper 683.

This Thesis is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

A PEER-TO-PEER DELIVERY SYSTEM FOR INTERNET SHORT VIDEO SHARING

by

Maryam Bashardoust Tajali

B.Sc. Computer Science, Ryerson University, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

MASTER OF APPLIED SCIENCE

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

© Maryam Bashardoust Tajali 2010

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Maryam Bashardoust Tajali

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Maryam Bashardoust Tajali

A PEER-TO-PEER DELIVERY SYSTEM FOR INTERNET SHORT VIDEO SHARING

M.A.Sc. Electrical and Computer Engineering, 2010

Maryam Bashardoust Tajali

Department of Electrical and Computer Engineering

Ryerson University

Abstract

In this thesis, we considered the effect of the content delivery network architecture on the popular short video sharing websites such as YouTube. The high number of users demanding videos impacts YouTube scalability which requires a new content delivery structure. Considering the high performance of P2P overlay networks, we propose an efficient peer-to-peer based system for short video sharing in the Internet in which all participant peers are responsible to distribute video replicas they have stored. This system comprises of a BitTorrent like network and a central media streaming server. To proficiently utilize P2P in our system, we propose some important approaches including an efficient and reliable indexing scheme, an efficient downloading strategy, a reliable content distribution mechanism, and a fairness policy. The simulations results demonstrate that the proposed system significantly increases client peers download speed while reduces the server workload and the startup delay for an improved playback quality.

Acknowledgements

I thank my enthusiastic, dedicated and hardworking supervisor Dr. Abdolreza Abhari for his guidance and encouragement on my research. His invaluable supervision and enlightenment are very much appreciated for my success in my Masters degree. This task would not have been accomplished without him. I am very grateful for his support that helped me to achieve my goal.

I would like to extend my gratitude and appreciation to all advisory committee members for their guidance, positive criticisms, and helpful suggestions.

I am deeply grateful to my father, whose guidance lid up my way and filled me up with encouragement in my life. I am very much appreciative and grateful to my mother who helped me on every step in my life, encouraged me every day, and had always been there for me. I am thankful to my sister, Soheila, my best friend, for all her support, guidance and kindness that helped me achieve my successes all my life. I am also thankful to my brothers, Siamak and Masoud, for their care, encouragement, and guidance to my way here. I feel so blessed to have such a loving family and grateful for their support and encouragement along the way with me to where I am today.

This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network and Compute/Calcul Canada.

To my beloved parents, Aghdas and Soheil

for their love and support

to whom all my successes belong to…

In loving memory of

my father, Soheil and

my aunt, Anis

TABLE OF CONTENTS

Chapter 1

1. Motivations and Objectives

1.1. Introduction and Problem Statement.....1
1.2. Scope and Motivation.....2
1.3. Contribution.....3
1.4. Thesis Organization.....4

Chapter 2

2. Background Information and Related Works

2.1. Background Information.....5

2.1.1. Web 2.0 and Social Networking Phenomenon.....5
2.1.2. Video-on-demand (VoD) and User-generated-content (UGC)....6
2.1.3. YouTube.....6
2.1.4. BitTorrent.....7
2.1.5. Architecture of P2P Networks.....8
2.1.6. Network Latency.....9

2.17. Overlay Delay, Playback Delay, and Startup Delay.....11
2.1.8. Discrete-event Simulation and Trace-driven Simulation.....11
2.2. Related Works.....12
2.2.1. Peer-to-Peer Streaming Networks vs Client-Server Streaming Networks.....12
2.2.2. Peer-to-Peer Video on Demand.....12
2.2.3. Peer-to-Peer Fairness.....14

Chapter 3

3. Proposed Peer-to-Peer Model and Implementation Strategy

3.1. P2P-based System for Short Video Sharing.....16
3.1.1. The Proposed Approach in the Designed P2P Structure.....16
3.1.2. Streaming Strategy.....19
3.1.3. Indexing Scheme.....20
3.1.4. Content Distribution Mechanism.....21
3.1.5. Fairness Strategy.....22
3.2. Implementation Strategy.....23
3.2.1. Assumptions.....23
3.2.2. P2P-based Short Video Sharing System Implementation.....24
3.3. Conclusion.....31

Chapter 4

4. Simulation and Results

4.1. Simulation.....32
4.1.1. Simulation Architecture.....32
4.1.2. Simulation Settings.....34
4.1.3. Trace Files Description.....35
4.2. Results.....38
4.2.1. Performance Metrics.....39
4.2.2. Bandwidth Improvement.....41
4.2.3. Network Traffic Improvement.....43
4.2.4. Startup Delay Improvement.....45
4.3. Conclusion.....47

Chapter 5

5. Conclusion and Future Work

5.1. Conclusion.....48
5.2. Future Work.....49

List of Tables

4.1. Users' bandwidth capacity and distribution.....34

4.2. Maximum number of connections for each seeder peer.....35

4.3. Trace files statistic.....37

4.4. The statistics of the videos in trace files.....38

List of Figures

3.1. The principle working of the proposed P2P model for short video sharing.....18

3.2. Illustration of the indexing data structure for all the videos in the network.....26

3.3. Illustration of the indexing data structure for all the active peers in the network.....26

3.4. Download process from multiple seeder peers.....28

4.1. A part of the log file.....36

4.2. Comparison of average download speed per client peer.....42

4.3. Comparison of average upload speed of the server.....43

4.4. Comparison of total data uploaded by the server.....44

4.5. Comparison of average fragments downloaded.....46

List of Flowcharts

4.1. Illustration of processing the simulated request.....33

List of Algorithms

1. Locating source peers…….……...27

2. Downloading the requested video's fragments…….……....29

3. Applying TFT policy……...…….…….30

CHAPTER 1

1. Motivations and Objectives

In this chapter, the motivation and the objectives of this research as well as the problem statement will be explained. The introduction and the problem statement are given in subsection 1.1. The scope and goal of this research are explained in subsection 1.2. The contributions are summarized in subsection 1.3. The thesis organization is given in subsection 1.4.

1.1. Introduction and Problem Statement

In general, there are two types of multimedia streaming available for the users: live streaming and video-on-demand streaming (VoD). In live streaming, the multimedia content is fresh and is being viewed live. In VoD streaming the freshness of the file does not matter, since the content has been created and uploaded on the host server and it can be played at any time. For that reason, VoD streaming delivered by video sharing service providers has become very popular among Internet users. A good example of a universal video sharing service that is highly used by Internet users is YouTube [1].

Peer-to-peer (P2P) topology utilizes various connections among the computer nodes in the network, which results an increase in the total available resources and the capacity of the system, in view of the fact that every node contributes bandwidth, computing power and storage space to the network. These networks are dynamic, because many heterogeneous hosts join and leave the network frequently and freely. P2P technology offers a wide range of facilities for the users, including multimedia streaming, online gaming, etc. P2P is especially highly demanded for its file sharing applications over

Internet such as BitTorrent [2]. In file sharing applications based on P2P topology, files are stored and served over the network among users which requires a balance between uploading and downloading files.

Recently, the rapid growth of sharing and streaming of video-on-demand and user generated multimedia content has emerged on a high number of platforms that provide services for requesting users. According to statistics [3], online video will exceed 91% of consumer traffic by 2014 which includes video-on-demand and the traditional web video (provided by TV providers). Among all, VoD traffic will double every two and a half years until 2014. An example of a highly viewed video sharing service on the Internet is YouTube which is ranked third among all Internet websites based on the three-month traffic ranking in 2010 by Alexa [4]. Moreover, according to 2010 statistics provided by YouTube website [5], over 2 billion videos are being watched every day and hundreds of thousands of videos are uploaded daily. This raises high concerns about YouTube and similar video hosting websites' scalability and high costs of administration and service providing.

1.2. Scope and Motivation

In recent years, P2P file streaming and downloading applications have become very popular Internet applications, since they are able to serve users adequately on large scales. Recent researches and studies show the significant improvement in the performance of multimedia delivery by adopting P2P topology [6, 7, 8]. Considering the characteristics and the benefits of P2P networks, it can be said that it is worth to examine whether or not P2P based video streaming systems have the capability of providing a reliable and scalable platform for Internet users to view any multimedia content.

As it is mentioned in subsection 1.1., the new generation of the media streaming services have severe scalability and Internet bandwidth consumption problems. Considering that

the traditional client-server architecture is not scalable enough with high number of users, applying P2P technology to streaming applications can be useful to decrease the load on the server by utilizing each peer's capabilities in the system. We examine that if utilizing P2P for short video sharing can potentially solve issues that both users and host websites have.

1.3. Contribution

The main contributions of this research are presented as follow:

- YouTube is considered so that the system performance of this popular service provider can be improved by deploying the proposed system.
- The proposed P2P system integrates a model similar to the popular BitTorrent protocol and a central media streaming server. This model has specific design approaches with the purpose of building an efficient and reliable P2P network.

In this research, we propose an efficient and reliable peer-to-peer based network for short video sharing in the Internet in which peers download videos and distribute the replicas among other peers in the network. In this model, peers download and upload during multiple various streaming sessions. A central media streaming server manages all the videos and peers in the network. Every node peer acts as both client and server for sharing short videos in the network. The central server keeps the information about all the videos and all the active peers in the network. Moreover the central server acts as a tracker by storing and managing the detailed information about from which peers each media content can be streamed. Also in the proposed design, some important issues on system reliability and scalability are addressed by proposing an efficient downloading strategy that reduces the startup delay and guarantees a smooth palyback without disruption, a fairness strategy that potentially increases peers' contribution, an efficient indexing scheme by using the central server, and a reliable content distribution mechanism. We design and implement a

P2P based network simulator to efficiently serve short videos on the Internet considering that P2P streaming applications scale very superior even in the large crowd scenarios. Simulated data from current popular multimedia serving website YouTube was used to conduct experiments. The trace-driven simulations and experiments performed during this research provide significant results which show the effectiveness of the proposed system to reduce the server workload, to decrease the startup delay, and to improve the download speed of the client peers.

1.4. Thesis Organization

The rest of the thesis is structured as outlined below:

Chapter 2 presents the background information and related works systematically. In Chapter 3, the proposed P2P based short video sharing model and the implementation strategy are provided. Chapter 4 presents the statistical analysis of the simulations and the experiments performed. The thesis conclusion and potential future work is discussed in Chapter 5.

CHAPTER 2

2. Background Information and Related Works

This chapter presents all the required background information and the significant related works in the area of this research. The background information is given in subsection 2.1. The related works are explained in subsection 2.2.

2.1. Background Information

2.1.1. Web 2.0 and Social Networking Phenomenon:

In recent years, the term Web 2.0 has been associated with web applications that are dynamic and include more user interactions as opposed to Web 1.0 which was static and users could only view the content that was created before. Web 2.0 websites allow users to interact and collaborate with each other and contribute to the website by creating user-generated-content in a virtual community. The examples of Web 2.0 technologies can be social networking sites, web applications, video sharing sites, wikis, and blogs [9]. However the main success occurred within the last few years by creation of popular social networks such as YouTube, MySpace [10], Twitter [11], Flicker [12], and Facebook [13].

2.1.2. Video-on-demand (VoD) and User-generated-content (UGC):

VoD systems allow users to select videos from a large collection and then watch it with user interactivity functions such as start, pause, fast forward, etc. VoD systems either stream the content so that it can be watched in real time, or download it so it can be watched later. This is done through a set-top box which can be a computer, a mobile phone, or any other device that can receive multimedia content. Unlike live streaming systems, in VoD streaming systems the freshness of the file does not matter, since the content is already pre-coded. Therefore users have more control over the content which makes VoD systems increasingly popular among Internet users such as YouTube [14]. As it is mentioned before, online video will exceed 91% of consumer traffic by 2014 which includes video-on-demand and the traditional web video.

UGC refers to any kind of media content such as text, photo, or video that is produced by end-users and is publicly available on the Internet [15]. In recent years, the new generation of the video sharing websites operates as UGC-based sites such as YouTube [16] and provide a user interactive platform for the end-users to upload their own video files, tag them with keywords, rate and comment the videos, and etc. Unlike traditional video sharing sites, YouTube and similar websites have become very successful because of the unique features they offer users. For example in YouTube, videos are connected to related videos and popular videos are noticeable based on the view count of each video.

2.1.3. YouTube:

In recent years, video sharing applications have become very popular among Internet users. The example of the video sharing services can be YouTube, Metacafe [17], Tudou [18] (the largest video sharing website in China) and other similar video hosting websites. Among all these, YouTube is the most popular video sharing service on the Internet and it has been rapidly growing since 2005 when it was created. Alexa Internet [4] ranks YouTube third of the Internet sites in the world based on the three-month traffic ranking in 2010. According to 2010 statistics provided by YouTube website [5], over 2 billion videos are being watched every day and this number is rapidly increasing. Therefore it can be said that at present, YouTube has a significant

impact on Internet traffic. However, according to recent researches, YouTube has severe scalability issues because of its current central architecture that services high number of users daily [6, 19]. Therefore, by knowing that there is an ongoing rapid growth of the highly demanded video sharing services, their costly high bandwidth consumption needs to be considered and their scalability issues need to be addressed.

2.1.4. BitTorrent:

In recent years, BitTorrent has become the most popular file distribution protocol on the Internet [8, 20]. A *torrent* file contains information about the files to be downloaded and the tracker. Clients who are downloading a specific torrent are grouped in a *swarm* and each client can participate in several swarms at a time. Each swarm is managed by a simple server called *tracker* which keeps track of all the clients in the swarm and manages the clients. Peer who has the entire file is called *seeder* and peer who is downloading the file is called *leecher*. Moreover, each media file is divided into a number of small pieces (256 KB) where even these pieces are split into sub-pieces (16 KB) so that a piece is downloaded from several peers at the same time [8]. After client downloads .t*orrent* file, it contacts the tracker to get the list of the other peers who are seeding or downloading the same file and then joins that swarm. To download the first piece, it selects pieces randomly and downloads sub-pieces from multiple peers until the first piece is completed as quickly as possible [20]. Then BitTorrent uses *rarest-first* strategy for download, meaning that the client keeps track of the number of copies of each piece and downloads the pieces with the fewest copies first.

In BitTorrent protocol, tit-for-tat (TFT) mechanism is employed to achieve the fair bandwidth exchange [20, 21, 22]. This policy can be used to reduce the impact of free-riders in the system and to monitor the sharing weight of each user. This ensures a balance between the download and the upload for each user so that the sharing ratio of the user is close to the overall 1 (sharing ratio: total bytes uploaded / total bytes downloaded). Considering that there is no central resource allocation, each peer maximizes its own download rate by downloading from other peers. To upload, a local client uses “optimistic unchoking” algorithm by which it randomly selects a remote client to upload to or when it receives data from it. Thus, the local client increases its

seen performance by uploading data to remote clients. To conclude, BitTorrent handles peers diverse upload bandwidths in two ways: TFT policy rewards peers with higher upload rates and the distribution of sub-pieces speeds up the download process.

2.1.5. Architecture of P2P Networks:

P2P systems are overlay networks used for indexing and peer discovery. These networks are formed dynamically by wide range of diverse nodes that are connected to the network in ad-hoc fashion way. Each node in an ad-hoc network has the freedom of joining or leaving the network without impacting the system notably. Moreover, every node acts as a server and a client at the same time in the network [23]. Generally based on how the nodes in P2P overlay networks are connected to each other, P2P architecture can be classified as two types: structured P2P networks and unstructured P2P networks [24]:

- In structured P2P networks, the connections between the peer nodes are fixed and determined by specific algorithms, providing a structure pattern of overlay links. In these systems, Distributed Hash Table (DHT) is used for indexing in the network. Each data content is identified by a key and DHT uses hash functions to assign keys to peer nodes in the network in order to determine which peer is responsible for which content. This results an efficient routing structure between peers when a search query is fired to locate those specific peers that are holding the desired data content even if the data file is rare. An example of such a system is Chord [25].

- In unstructured P2P, the overlay links are formed randomly without taking into account any algorithms for optimizing the network connections or for organizing the existing network connections. To search for a desired content, flooding mechanism is applied. This way the peer submits the query which goes through the network to find as many peers as possible that have the content and share it. If the data is popular, the query is most likely to be successful in finding high number of provider peers. However, the main disadvantage of such network architecture is that the queries may not always be resolved. For example in the case that the data is not a popular content, which means it is not being shared by many peers in the network then there is not a high chance that the search would be successful. Moreover

flooding causes significant network traffic and hence a poor search efficiency. The examples of popular unstructured P2P networks can be Napster [26], KaZaA [27], Gnutella [28], and BitTorrent. Researches show that in general, for today's mass market data sharing applications, unstructured overlays perform better with more support than structured overlays because peers are extremely transient [29].

In particular, there are three models for this P2P architecture: Pure P2P, hybrid P2P and centralized P2P. Pure P2P networks consist of peer nodes with equal efficacy and ability, meaning that no special node exists in the network for example Gnutella 0.4. In Hybrid P2P, there are special nodes with specific infrastructure functions called *supernode* for example KaZaA. In centralized P2P systems, there is a central authority that manages and handles indexing functions for the entire network and every peer node operates as both client and server for file sharing such as Napster and BitTorrent. The centralized architecture has some similarities with the structured architecture, however in centralized systems the connection between peer nodes are not determined by using any specific algorithm or function. In BitTorrent system the peer nodes share their files independent from the central server which is called *tracker*. This gives such a system an advantage of being fault tolerance meaning that if one of the provider peers disconnects from the network, there are other provider peers that have the same desired content and are able to share it.

2.1.6. Network Latency:

Network latency or delay, is basically the time needed for a packet of data to be transferred from the sender computer to the receiver computer. In some cases, latency is measured based on the round-trip time which is the time it takes to send a packet from the sender to the receiver and to receive back a response from the receiver to the sender [30]. It makes the most sense to measure network latency by the round-trip time. This is because computers are constantly in the process of sending and receiving bytes and are communicating with each other. Once a computer receives some information, it will send some information back to the sender as well. When a source node sends a packet, this packet travels through a path with several routers to get to the destination node. There are several different factors that contribute to network latency. The most

important factors are processing, queuing, transmission, and propagation delays [30]. IP network delays range between a few milliseconds to several hundred milliseconds. There are some methods to improve each delay factor in a way that the user does not experience any noticeable delay. Each delay factor is explained below:

- Processing delay: this is the time taken by routers to process the packet's header and to transmit the packet [30] which is actually the time between receiving the packet by the router and then putting it into the transmission queue. It takes time for each gateway node (router) to examine or change fields in the packet's header. This delay depends on the processing power of network devices [31]. Moreover, computer hardware might be a contributor to network latency. In this case for example by upgrading to a faster processing network devices or hardware, a faster transmission can be obtained.

- Queuing delay: it is the amount of time a packet waits in the routers' queues [30]. Normally a packet traverses through a collection of routers and switches. This delay depends on the intensity of the traffic arriving to the queue [30]. Meaning that the time each packet spends in the queue depends on the number of other packets that are arrived earlier and are waiting to be transmitted, so each packet experiences a different delay depending on the traffic.

- Transmission delay (store-and-forward delay): in network latency, transmission delay refers to the time taken to push all the bits of a packet into the medium link [30]. It should be considered that the medium to transmit data whether wireless, optical fiber line, phone line or others, has delay itself also. Transmission delay is a function of the packet's length and the link's speed (bit-rate) and does not depend on the distance between computer nodes [30]. Since this delay is caused by the data rate of the medium link, this problem can be solved by using faster mediums which reduce the latency and speed up the process**.** However, knowing that in today's networks, having high bit-rate connections is usual; this delay has a minimum impact on the data transmission and therefore is negligible.

- Propagation delay: this is the time taken for the packet to traverse from the source node to the destination node. This delay is a function of the physical distance between sender and receiver routers and the link's bandwidth between the sender and the receiver nodes [30]. In

general, the more distance between participant nodes, the more delay the transmission will have but not a significant delay. It should be considered that this type of delay is harder to control.

2.1.7. Overlay Delay, Playback Delay, and Startup Delay:

As it is mentioned before, P2P systems are overlay networks that are usually implemented as an application network layer on top of the Physical network layer specifically Internet. There is an important type of delay associated with the overlay networks. This delay is referred as overlay delay which is the amount of time required for each computer node to establish connection to the other machines.

During the playback, if the current playback trackbar position has not been downloaded yet, then the playback stops for several seconds while more data is being downloaded and then viewer continues watching the video. Playback delay occurs when the user's download bandwidth is lower than the requested video's bit-rate [32]. Thus, the playback trackbar position should be behind the download trackbar position to assure a continuous playback.

Startup delay is referred to the time that the user needs to wait for the player buffer to receive data from the stream provider. The streaming bit-rate also has an effect on the startup delay.

2.1.8. Discrete-event Simulation and Trace-driven Simulation:

If the events aren't guaranteed to occur at regular time intervals, event-driven simulation is an appropriate choice to use. This approach handles these events in order of increasing time, meaning that the events are prioritized. There are various types of event-driven simulation models; however in this research discrete-event simulation based on trace-driven simulation is the base for the simulated model. Discrete-event simulation uses a discrete-state model of the system. It takes distinct values and models a system that progresses over the time where the state variables of the system may or may not change instantly at specific and separate points of time when an event occurs. A trace is a time-ordered record of events that is used as an input for the simulation [33, 34].

2.2. Related Works

This section contains an overview on the related work:

2.2.1. Peer-to-Peer Streaming Networks vs Client-Server Streaming Networks

There are several research works on traditional streaming networks for examining of how P2P is beneficial in terms of the server workload reduction for multimedia service providers on Internet. Huang et al. [35] research on VoD in the Internet and its costly high bandwidth requirements. Based on the nine-month trace from a client-server VoD deployed in MSN video [36], the authors analyze some user behaviour and video popularity distribution and propose a peer-assisted VoD system so that the workload of the server can be reduced. YouTube and similar service providers are different than the traditional VoD services because of the limited control the service provider has over the creation of user-generated-content [19, 37]. Zink et al. [37] analyze user behaviour, access patterns for videos, and the popularity of videos from the users in a campus network accessing YouTube servers. The results from using the collected data in a trace-driven simulation present that caching reduces network traffic and saves considerable bandwidth. Soraya et al. [19] study YouTube traffic over popular and regular videos, analyze YouTube workload, and suggest that caching improves the network performance and scalability.

Some other significant research studies on P2P systems characteristics for future development of P2P applications. Authors in [38, 39] measure and analyze a large amount of collected data from PPLive and present valuable results on user behaviour and the system performance that can be used to improve P2P VoD systems.

2.2.2. Peer-to-Peer Video on Demand:

In recent years, P2P file streaming and downloading applications have become very popular among Internet users since they provide a scalable and reliable platform for content distribution systems while they decrease the server workload. In P2P live streaming systems, the multimedia content is fresh and is being viewed live such as CoolStreaming [40] and PPLive [41] that

provide live TV streaming (PPLive also provides VoD streaming). However in VoD streaming systems the freshness of the file does not matter, since the content is made before it is viewed [42]. Compare to the P2P live streaming systems, there are fewer P2P VoD streaming systems such as PPLive, Joost [43], PPStream [44], and GridCast [45].

A common approach to obtain a scalable and reliable P2P VoD streaming system is proxy caching or peers segment caching in the network [46, 47, 48]. Guo et al. [48] present PROP: a scalable cache management mechanism based on structured P2P system that significantly improves the quality of media streaming. Another approach proposed by Liang et al. [46] is a distributed cache management model for large-scale P2P VoD which results efficient management and usage of the cache content. In this system, overlay is comprised of some rings where peers whose caching segments provide a full media file are organized into a ring so that latency to locate a video clip is reduced. Jiang et al. [47] propose a chunk-based proxy scheme for unstructured P2P VoD network. This scheme is evaluated by trace-driven simulations using logs collected in GridCast. The results show that for large-scale P2P VoD systems, the proposed approach significantly improves the quality of media streaming and the system scalability.

Most researches on VoD systems focus on constructing application-level multicast overlay in which peers contribute to the system by uploading to the lower layer peers. Yang et al. [49] design an efficient distributed VoD architecture that combines distributed server architecture (DVoD) with multicast P2P system for a more scalable network. Guo et al. [50] propose P2Cast: an architecture that is P2P based to stream video and extends multicast patching scheme. It constructs an application overlay appropriate for streaming and provides a continuous playback if disruption occurs. Gallo et al. [51] consider all types of video delivery (linear TV, VoD, time-shifted TV (tsTV), network Personal Video Recorder (nPVR))[1] and propose a distributed system that uses IP multicast, distributed caching, and P2P content exchange for an efficient media content distribution.

Other researches explore P2P VoD systems by presenting distinct approaches. Cheng et al. [6] propose a P2P video sharing framework that explores the clustering in social networks called NetTube. Based on measurements over YouTube videos, the authors present a bi-layer overlay in which each peer caches the watched videos and pre-fetches predicted video files from the related

[1]LinearTV is the usual live TV program. tsTV is the broadcasted TV program with a time shift i.e. user starts watching the program from the beginning when the program is broadcasted already.

videos of the first watched video. The results show that this system reduces the workload of the server and improves the playback quality. Choe et al. [8] present Toast; a scalable VoD streaming system based on BitTorrent that uses a streaming server to speed up the transfer. Xu et al. [7] integrate an enhanced data-driven overlay network and a multi-way tree and propose SDNet: a distributed storage-assisted overlay network to support P2P VoD services in which videos' segments are pre-fetched and stored in a distributed manner to achieve high search efficiency and low latency.

The differences and similarities of the architecture of the proposed model with the similar works in this area are presented in section 3.1.1.

2.2.3. Peer-to-Peer Fairness:

P2P file sharing networks depend on peers uploading and downloading data in the system. Some peers called free-riders will not upload any data unless there is an incentive or policy to do so, since users gain no personal benefits by uploading files [52]. The existence of free-riders and low bandwidth peers diminishes the overall performance of P2P systems, in view of the fact that these peers consume high download bandwidth while contributing very low upload bandwidth in the network [21, 53, 54].

A common metric of fairness is the sharing ratio of a peer [54]. Sharing ratio is defined as the total number of uploaded bytes divided by the total number of downloaded bytes. There is an enforcement policy in some P2P communities that bans some peers if their sharing ratio is below a specific threshold. This way users need to upload more data even when the target file is downloaded completely. A fair sharing ratio among peers is obtained when each peer contributes to the network as much data as it consumes. However it should be considered that in most P2P networks high number of peers resides behind firewalls or NATs. Mol et al. [54] prove that logically it is impossible to prevent free-riding when more than half of the peers are firewalled. Firewall is a piece of software or hardware that is implemented to block unauthorized accesses to the private networks while connecting to the Internet therefore increasing the overall security of the system. NAT is a router used to translate between address spaces. Any peer behind firewall or NAT, is able to initiate an outgoing connection to other peers and download or upload to

them. However, it cannot accept any incoming connections. As a result firewalled peers are unable to connect to each other. Researchers in this area proposed different techniques that can be used in some P2P networks to make two peers behind firewall able to establish a connection [55, 56]. However it is considerable that when the number of firewalled peers increases, the average sharing ratios of these firewalled peers decrease fast. So for that reason in such situations free-riding cannot be avoided.

In BitTorrent, TFT mechanism is built to achieve the fair bandwidth exchange as it is explained in section 2.1.4. However researches show that the current rate-based TFT cannot prevent unfairness and poor network performance that is mainly caused by free-riders existence [57]. Bharambe et al. [53] propose a block-based TFT policy to reduce the unfairness in BitTorrent. This policy achieves fairness in terms of the volume of the content served in the network. Another research done to prevent free-riders and to improve the efficiency of optimistic unchoking is the novel optimistic unchoking proposed by Ma et al. [22]. This algorithm uses peer's information obtained from the previous streaming and unchokes (uploads to) the peer with the highest expected upload bandwidth.

To provide a better performance for contributing peers, Sherman et al. [21] present FairTorrent; a distributed algorithm that is implemented in a BitTorrent client to reward peers based on their contributions. So if a peer has uploaded the most data, it receives the best performance.

CHAPTER 3

3. Proposed Peer-to-Peer Model and Implementation Strategy

This chapter presents a reliable peer-to-peer based short video sharing system and the implementation strategy. In subsection 3.1., the approach to design the proposed peer-to-peer network for short video sharing in the Internet is discussed. Subsequently, subsection 3.2. elaborates on the implementation strategy for the proposed P2P model. The conclusion of this chapter is provided in subsection 3.3.

3.1. P2P-based System for Short Video Sharing

This section presents the detailed information of the proposed P2P system. First, the main differences of the proposed system with similar researches are discussed and then the details of this model and the significant design approaches are provided:

3.1.1. The Proposed Approach in the Designed P2P Structure

As it is explained in 2.2., there have been numerous researches on P2P systems characteristics and performance. P2P systems can effectively handle the high number of requests in the network since every peer operates as a client and a server. Most of the P2P systems developed for VoD streaming are based on the proxy caching or distributed peers caching in order to reduce the workload of the server [46, 47, 48]. Another technique for VoD delivery is based on overlay multicast technology and some form of P2P network [49, 50, 51]. There are other works that have proposed a unique P2P VoD system and have provided useful results shown by extensive

analysis and simulations [7, 8]. The great majority of these works consider traditional VoD and utilize caching or overlay multicast to solve scalability and server load problems. However as it is mentioned in 1.1., the new generation of video-on-demand and user generated multimedia content has imposed significant increase on Internet traffic in recent years. Unlike all the works done in this area (except [6]), we focus on YouTube and content distribution websites in order to turn them into scalable and reliable service for Internet users. The approach in [6] is similar to this work as the main focus is designing a peer-assisted model that suits YouTube and provides the scalable delivery service for short video sharing. However, our distinct approach is modeling a P2P network based on BitTorrent protocol by utilizing a central streaming server for short video sharing on Internet which enhances the system performance by reducing server workload in a scalable and reliable way. It should be considered that this research is among other few researches performed in the area of short video sharing enhancement.

The proposed P2P-based short video sharing system integrates a P2P system similar to the popular BitTorrent protocol where peers download and upload during multiple streaming sessions. A central streaming server manages all the videos and peers in this network. The system has an unstructured centralized architecture in which peers are self-organized in an unstructured P2P overlay network; meaning that all the peers get seeder peers and videos information from the server in order to form connections to the seeder peers in the network. Furthermore, peers in this P2P network are least dependent on other peers in the network, therefore they join and leave (after applying TFT policy) the system freely. The dedicated server contains all the videos and keeps the information of all the videos and all the active peers in the network. This central streaming server also acts as a tracker by storing and managing the detailed information about from which peers each media content can be streamed. So when a new client peer joins the network, it contacts the server to get information about the seeder peers that have the requested video. If the video has not been downloaded by other peers, server uploads the video to the client. Otherwise, the video is already watched by other users and can be distributed by these peers as seeder peers and also server if required. Thus, the requesting peer connects to the seeder peers and starts downloading the video fragments from multiple seeder peers simultaneously and saves a complete replica of the requested video in its local disk. Figure 3.1. illustrates how this network operates.

Consider the scenario that there is a fixed number of videos available in the network and peers constantly join the network, download videos and keep replicas for future distribution, and never leave the network. After a period of time, majority of videos have been downloaded and have several seeder peers. Therefore, server does not need to upload any video to any client because of high number of available seeder peers who already watched videos. It is clear that this ideal scenario is almost not achievable because not all the videos are going to be requested in the real system and also client peers cannot be connected to the network forever. However, the concept of the proposed system clearly results a system in which peers cooperate in videos' distribution by adding their own upload bandwidth to the network, therefore reducing the workload of the server.

In should be noticed that in this research, "client peer" or "client" is the requesting peer and "seeder peer" or "source peer" is the provider peer who has the complete video file.

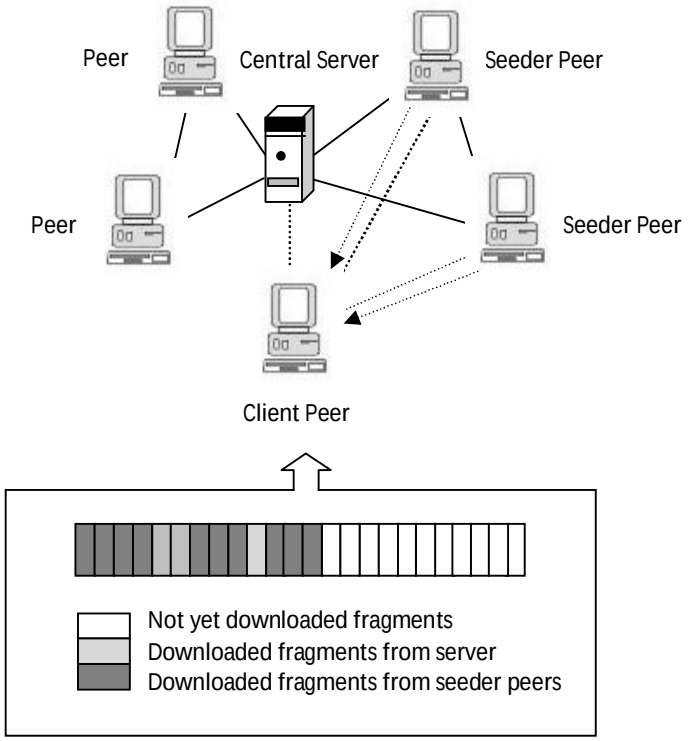


Figure 3.1. The principle working of the proposed P2P model for short video sharing

3.1.2. Streaming Strategy

Researches show that the default BitTorrent download strategy does not work well for VoD streaming because it does not consider when pieces are needed by the user [8] (as the download mechanism is explained in 2.1.4.). To address this issue, the proposed download strategy considers downloading pieces of the media content in a buffer window in order, which are required for a continuous playback by user. This decreases the startup delay and can potentially reduce the chance of pausing during view of the video.

This network has a dynamic structure in which every peer enters the network, sends requests to watch different videos, downloads those videos and saves a copy of every new streamed video, and then leaves the system if it has uploaded as much data as it has downloaded. In this system, when multiple peers connect to a seeder peer to download a multimedia content, its upload bandwidth gets split equally among all the connected peers. In the proposed P2P network, every video is composed of number of fragments; each of 16 KB size similar to sub-pieces size in BitTorrent. For every video, each replica of that video has the same number of fragments as the original video. This is because it is assumed that there is no incomplete download of any video and every seeder peer of a video has all its fragments.

Central Streaming Server:

In the proposed P2P system, when a client peer joins the system, it contacts the server to get the list of available seeder peers for its requested video. If the video is going to be downloaded for the first time, then the server uploads all the video's fragments that will be needed earlier by the viewer. For example peerID 2 (*P2*) requests videoID 4 (*V4*) that has 5 fragments: *f1, f2, f3, f4, f5*. When *P2* starts streaming *V4*, it actually downloads from *f1* to *f5*. For example the downloaded order can be *f1, f2, f3* in 1st second and *f4, f5* in 2nd second or *f1, f2, f3, f4, f5* in 1st second. These different scenarios can occur depending on the available upload bandwidth that server or the seeder peer provides for this download as it is explained in details in section 3.2.2. As soon as the client receives the complete replica of the video, it saves it on its local disk space and becomes a seeder peer for that video. Moreover, during download from multiple seeder peers,

client can receives some fragments from the server relative to the server's available upload bandwidth. This ensures that the client receives all the required fragments as soon as possible for the video playback and to distribute the fragments to other requesting peers as a seeder peer later.

Multiple Seeder Peers:

When a client peer requests a video that has multiple seeder peers other than server, then it connects to those seeder peers and downloads the requested video's fragments concurrently and keeps a complete replica of the requested video. The number of fragments streamed from each seeder peers is based on its available upload bandwidth. During this download session, server may help seeder peers by streaming some fragments to the client so that all the video's fragments are received fast enough. This potentially has a positive effect on reducing the startup delay and improving the playback continuity. It also should be noticed that in general, the client peer downloads all the video's fragments that will be needed earlier for viewing and saves the complete replica on its local disk space and starts distributing the video's fragments subsequently. Hence, the users are able to provide their replicas to each other and reduce the load on the server by contributing their upload bandwidth and storage space in the network.

3.1.3. Indexing Scheme

Data indexing in the system is done by hashing indexed key-words. Hashing-based indexing makes the search system to be entirely controlled. As it is mentioned before, the central server stores all the information of all the videos and all the peers in the network. Server constantly monitors peers' contribution in the network and updates the relevant information after every completed download process. Unlike other approaches, the proposed model performs indexing functions for both peers and videos which makes the entire network very easy to manage. The main indexing functions based on videos id in order to locate the source peers. So when a peer requests for the seeder peers of a specific video, server promptly responds with the list of seeder peers of that video. Moreover, by using the other indexing based on peers id, it is easy to get information about videos a certain peer has stored and manage the peers in the network.

In general, the main operations on the indexing tables are managed by central streaming server. We refer to the indexing hash table of peers as peerMap and the indexing hash table of videos as videoMap. Each line in the peerMap contains information about one peer and each line in the videoMap contains information about one video. The following provides the details of each operation:

Peer joins: When a peer joins the network and requests a video, a line is created in the peerMap for this peer containing its information.

Peer leaves: Peer leaves the network if and only if it has uploaded as much data as it has downloaded. Upon leaving the network, this peer is removed from the list of the seeder peers for every video it shares in the videoMap.

Peer searches for seeder peers: Server provides a list of the seeder peers for the requested video from the videoMap.

Peer downloads a new video: As soon as the client peer's download is completed, server adds the client peer to the list of the seeder peers of the current video in the videoMap.

3.1.4. Content Distribution Mechanism

In this research, YouTube popular daily videos that are requested by the users are considered. The data used to conduct the experiments was synthetically generated similar to YouTube video characteristics. In 2010, YouTube increased the duration limit of the uploaded videos from 10 minutes to 15 minutes which still fits in the short video category. The focus of this research is on short videos that are noticeably smaller in size compared to the long length VoD and a small number of these videos are watched daily by a client. Considering the large storage devices currently available, it is rational to assume that it is possible for the client to save all the requested videos on its hard disk. Therefore, every client contributes its upload bandwidth and its

storage space for the short video distribution in the network. As it is explained in 3.1.2., every client peer in the proposed model keeps a replica of the viewed video on its local disk space when receives the complete video file. This additional storage space builds a dynamic P2P replication system for content replication that meets users' needs.

There are several policies available in order to manage a cache of information stored on the computer and to keep the storage space well-used. Least Recently Used (LRU) and Least Frequently Used (LFU) policies can potentially be applied to manage a limited disk space. In the proposed P2P network, LFU is the best fit to be implemented for discarding replicas that have been requested least often than other replicas, daily or weekly (i.e. 5 videos with the lowest access number are deleted every week).

3.1.5. Fairness Strategy

The major issue of BitTorrent download strategy occurs because of optimistic unchoking mechanism. Researches show that 20% of maximum download rate of a peer can be consumed by a free-rider using this strategy [57]. To address the free-riders problem, the strategy used in the proposed model is a mechanism that enforces each peer to upload as much data as they have downloaded. Thus, if a peer has not uploaded the amount of data equal or more than the amount of data it has downloaded yet; it will remain in the network to distribute videos it has stored. This increases the durability of each peer and hence increases the overall stability and bandwidth of the network.

In the proposed P2P short video sharing system, each peer is required to apply tit-for-tat policy in order to have a fair sharing of network usage. We assume that every peer in the network can accept incoming connections and no peer resides behind NAT gateway or firewall. Therefore, all the peers cooperate with each other to distribute the short videos in the network. To apply this enforcement in the real system, a service provider can set specific rules that users should follow. For instance if some users do not contribute in videos' distribution, they may be banned from future usage of the network resources.

3.2. Implementation Strategy

The implementation of the proposed P2P short video sharing system is based on some assumptions presented in subsection 3.2.1. Subsequently, in subsection 3.2.2., the comprehensive details of the proposed system implementation are described.

3.2.1. Assumptions

In the proposed P2P system, we assume that there is no incomplete streaming for requested videos in order to have complete and equal size replicas of each video around the network. As it is mentioned before, every video is divided into a number of fragments. For every video, all the replicas have the same number of fragments as the original video. This means that if *P12* becomes a client peer and requests *V45*, then it downloads *V45* fragments from available seeder peers and it stays in the network to share the video file after. Hence, *P12* keeps a complete replica of *V45*. As a result, *V45*'s fragments are equally available in the P2P network. Meaning that if *V45* has 3 fragments (*f1*, *f2*, *f3*) and there are 4 replicas of this video available around the network, then there are four *f1*, four *f2*, and four *f3* fragments available for distribution in the network.

Furthermore, we assume that the client peer has enough disk space to store all the videos it has watched. Since, a short video sharing network is simulated in which the videos have small sizes, so storing them does not require an enormous amount of disk space even in scenarios that user is watching high number of videos daily. For that reason, no replacement strategy mentioned in 3.1.4. has been implemented in this research as we assume each client employs sufficient disk space to keep the received replicas.

Usually, video bit-rate of YouTube videos is mostly around 330 kbps [6]. Meaning that most of the videos have been encoded at this speed to be streamed at this speed or higher speeds. This needs to be guaranteed to ensure a continuous playback and a good viewing quality. If the peer's streaming rate is less than the requested video's bit-rate, then the user needs to wait for a few second until more data is transferred to his computer then continues to watch the video. One

reason that this situation occurs is when peer's maximum download rate is less that the requested video's bit-rate, so user faces several pauses during watching the video. In this simulation, we assume that every client peer assures the minimum streaming rate and is able to receive the requested video with required video's bit-rate so it does not encounter playback delay. Also because of our distinct streaming strategy where users receive all the small fragments rapidly, the startup delay is small and the playback can be continuous.

When it comes to multimedia content sharing, the most types of delays that can potentially affect the streaming time are transmission delay and propagation delay. In the proposed short video sharing system, transmission delay is negligible because of streaming small size fragments over considerably high bit-rate connections knowing that nowadays majority of Internet users have high bit-rate connections. In addition, we assume that the seeder peers are very close to the client peers, therefore the propagation delay is not significant. For these reasons, it has not been taken into account any network latency or time required to establish connections to other machines in the simulation.

3.2.2. P2P-based Short Video Sharing System Implementation

The proposed P2P based system for short video sharing is simulated to evaluate this model. The language used for the simulation of this P2P network simulator is Java 6.0 due to its portability and extensibility in Eclipse IDE. The implemented simulator is a discrete-event simulation model in which events occur at various times during the simulation process. Each request for a specific video file is a discrete event, because it happens at an instantaneous time. These events are prioritized and are handled by increasing time. As it is explained in 2.1.9., a discrete-event simulation models a system in which a number of events occur at distinct points of time that may change the state of the system. A discrete-event simulation of the proposed P2P short video serving system has the following state variables: the number of peers in the network and the status of the server. The number of the peers in the network changes when a peer joins the network or leaves the network after it is done downloading and uploading. In the modeled system, server performs management and indexing functions. The status of the server can be idle or busy which changes whenever a peer joins or leaves or requests a video. Moreover, this

simulator is built as a discrete-event simulator that operates as a trace-driven simulator. Events are the time-ordered requests of videos in a log file which is used as an input for the simulation. The more details of the simulation are explained in section 4.1.

The simulator clock starts working when the simulation starts and then it moves forward with the intention that the time in the simulation is synthetically equal to the time that the peer's request takes place in the trace. At each point of simulation time, the clock increases by 1 unit of simulated time. This way, from the beginning to the end of the simulation process, the time is artificially equal to the real time in the network.

In this network, the central streaming server has a constant upload bandwidth. Peers join the network with diverse upload and download bandwidths to collaborate in distributing video files (random bandwidth capacity is generated for each peer). The simulator model uses an input trace and simulates peer activities including joining, leaving, indexing, video downloading, uploading, and TFT policy that are explained next.

Joining and Indexing:

In the proposed model, the main indexing functions based on the videos. We added another indexing in terms of peers in order to achieve an efficient indexing scheme that is managed by server. All the videos in the network are stored in videoMap that maps videoIDs as keys to corresponding associated values which are videos' information. Figure 3.2. demonstrates an overall view of the used data structure and indexing hash table.

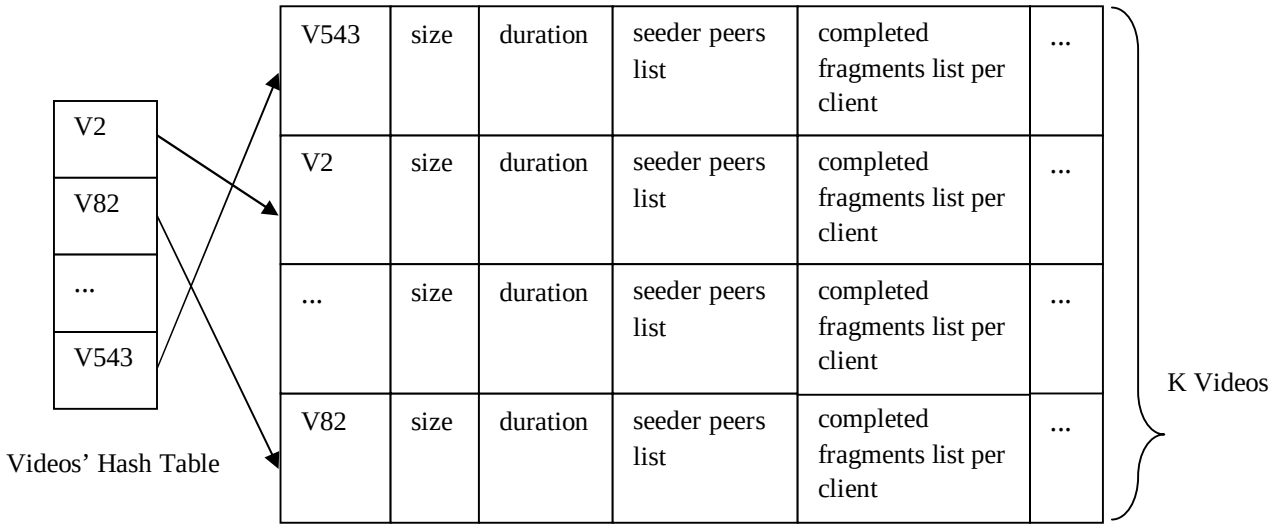


Figure 3.2. Illustration of the indexing data structure for all the videos in the network

In addition, when a client peer joins the network, a line is created in the peerMap with peerID as the key to keep the detailed information of this peer shown in Figure 3.3. The indexing tables build an efficient indexing scheme in the proposed system so that the server easily handles and manages the information of all the videos and active peers in the network. The data structure used for keeping the information of the peers is shown below:

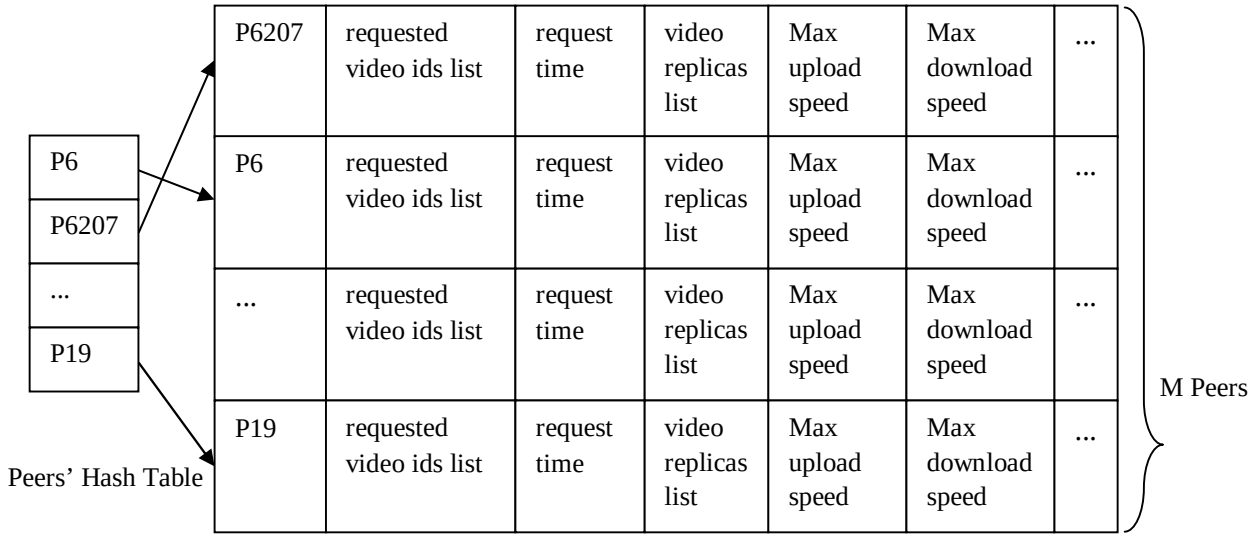


Figure 3.3. Illustration of the indexing data structure for all the active peers in the network

After the client peer enters the system, it contacts the server with requested videoID. Server then responds back with the list of the seeder peers of that specific video. Subsequently, the client peer establishes connections with all the seeder peers and starts downloading video's fragments in a small buffer window in order of the fragment ids. Algorithm 1 shows the indexing process upon contacting the server.

Algorithm 1. Locating source peers

for each video *V* in the video list *videoList* **do**
  if *Vid* is equal to the requested video id **then**
  send the seeder peers list *peers* to client peer *C*
  end if
end for

Downloading and Uploading:

During downloading a video file by a client peer, every seeder peer transfers *N* fragments to the client peer using as much upload bandwidth as it has available that takes one unit of time artificially equal to 1 second. The client peer then receives all the required fragments in sequence and keeps them in a buffer for viewing. Once a client peer downloads a complete video, it starts uploading the requested fragments of that video to other client peers. Algorithm 2 presents a general form of the download process in the simulated P2P network.

Suppose *P16* requests downloading *V5* which is composed of three fragments (each fragment is 16 KB size): *f1*, *f2*, *f3*. There is one seeder peer available for streaming with available upload bandwidth of 96 KBps (768 kbps). Maximum number of fragments can be transferred through this bandwidth is 6 fragments (96/16 = 6). So considering the available upload bandwidth, six fragments can be easily streamed using this bandwidth in 1 second. However, in this case the client peer only needs to download 3 fragments only from *P16* to receive the complete video which takes 1 second without the server's assistant for a fast download.

In another scenario, suppose *P16* requests downloading the next video *V42* which is composed of 10 fragments: *f1, f2, f3, f4, f5, f6, f7, f8, f9, f10*. The source peers to stream from are *P3*, *P25*, and *P83* with available upload bandwidths 96 KBps, 96 KBps, and 48 KBps respectively. Total number of fragments that can be received in 1 second is 15 which is the summation of the maximum number of fragments that can be transferred using each seeder peer's upload bandwidth (6+6+3). Since the total required fragments are 10, the complete video can be buffered in 1 second by downloading from two of the seeder peers if enough upload bandwidth is provided at the same time (without downloading from *P83*) as it is demonstrated in Figure 3.4.

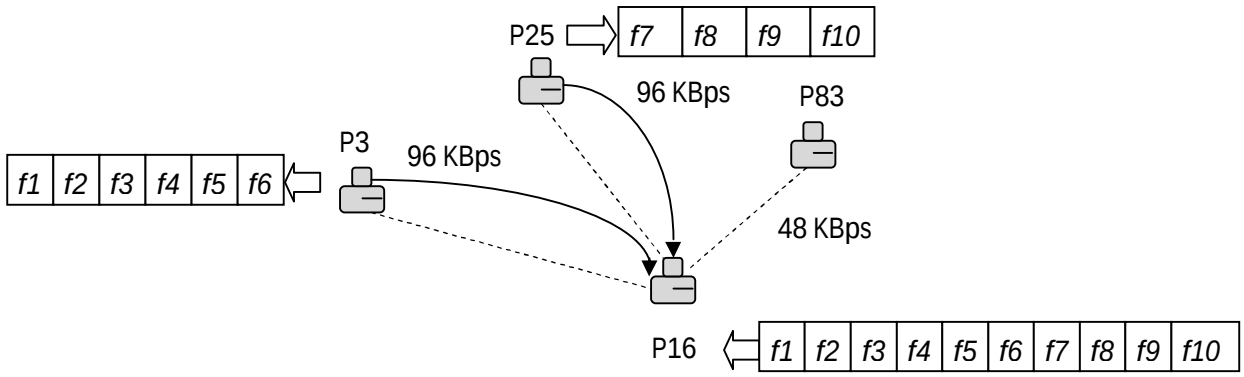


Figure 3.4. Download process from multiple seeder peers

It should be noted that, when multiple peers connect to a seeder peer to download fragments, its upload bandwidth gets split equally among all the connected peers. To download all the fragments of a video, the client peer receives more fragments from higher upload bandwidth seeder peers. This means that each seeder peer contributes to the download process in accordance with its available upload bandwidth. Furthermore, in order to receive the complete replica of the video as fast as possible, the client peer downloads some number of fragments from the server relative to the server's available upload bandwidth by requesting starting and ending fragments id. For example, if the server has 100 KBps upload bandwidth available, a client peer who is going to download a relatively large size video that takes more time to be

downloaded, can download maximum 6 fragments (100/16 = 6) from the server along with other seeder peers and receive as many fragments as the total upload bandwidths provide simultaneously and the download bandwidth of client permits. This mechanism guarantees that the client would receive all the required fragments as soon as possible to have a small startup delay and a continuous playback without disruption. Therefore, in the proposed model, the download rate is higher than the playback rate which results a good viewing quality and a continuous playback. It should be considered that in general, when all the participant nodes can continuously playback the video with a very small start up delay, then it can be said that such a P2P streaming system performs well and satisfies the required Quality of Service (QoS).

Algorithm 2. Downloading the requested video's fragments

```
while the completed fragments list fragmentsCompleted is buffering fragments
    for each seeder peer P in peers do
        max_num_frag = getAvailableUploadSpeed(Pid) / frag_size
        for each fragment f in the missing fragments list fragmentsMissing do
            receive f in fragmentsCompleted
        end for
    end for
end while
add C to peers of V

getAvailableUploadSpeed(peer_ id)
get peer_id maximum upload speed / number of connections to peer_id
```

Tit-for-Tat Policy:

TFT policy is enforced in the model similar to BitTorrent protocol. Meaning that there is a balance between the download and the upload for each peer. Peers in the network are required to upload as much data as they have downloaded so that free-riders are prevented to the possible extent in the network. After tit-for-tat policy is performed by the peer, then the peer leaves the system in order to have a dynamic P2P network. Algorithm 3 demonstrates the tit-for-tat policy applied in the proposed P2P system.

Algorithm 3. Applying TFT policy

```
for each seeder peer P in peers of V do
        if not P.isActive() then
                removeSeederPeer(P)
        end if
end for

isActive()
P uploaded bytes >= downloaded bytes
active ←— false

removeSeederPeer(peer)
for each V in videoList do
        remove peer from peers of V, if it exists
end for
```

3.3. Conclusion

In this chapter, an efficient and reliable peer-to-peer based short video sharing system is proposed which is a hybrid model of a P2P network that combines the BitTorrent system and a central media streaming server. This model is proposed to solve scalability and bandwidth issues that YouTube and similar video hosting websites face. Subsequently, a number of distinct and efficient approaches are proposed for indexing scheme, downloading strategy, content distribution mechanism, and fairness strategy. Afterwards, to evaluate the proposed system, the P2P network simulator is designed and implemented to serve multimedia content on the Internet efficiently. In this chapter, the assumptions and implementation approaches to simulate the system are presented. Next chapter presents how the proposed model has an enhanced performance for short video sharing in the Internet by discussing the simulation's results.

CHAPTER 4

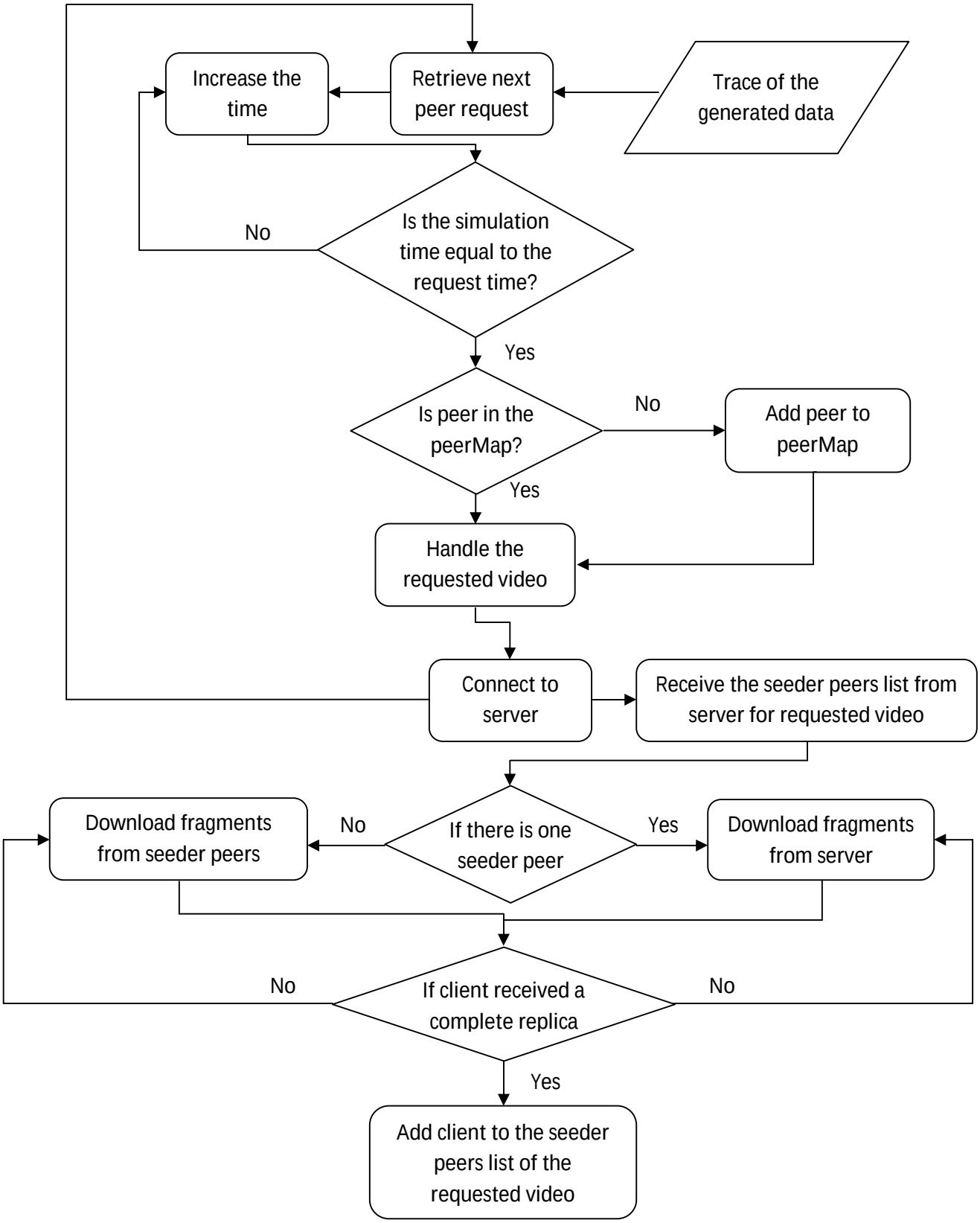
4. Simulation and Results

This chapter presents the performed simulations and experiments followed by the results. In subsection 4.1., the architecture and the settings of the simulation model are explained and the comprehensive details of the input trace files are described. The results of the simulations are systemically presented in subsection 4.2. This chapter is concluded in subsection 4.3.

4.1. Simulation

4.1.1. Simulation Architecture

As the architecture of this simulator is inclusively described in subsection 3.2.2., our simulator is built as a discrete-event simulator that performs a trace-driven simulation. Here, events are the time-ordered requests of videos in a trace file which is the input for the simulator. Each request for a specific video file occurs at an instantaneous time and is handled by the server. At the beginning of the simulation, the simulator clock starts working and increases by 1 unit of time as the simulation advances forward. Therefore, the time in the simulation is synthetically equal to the time that the peer's request takes place in the trace. As a result, from the beginning to the end of the simulation process, the time is artificially equal to the real time in the network. For example a request read from trace is made at time 603 by *P9*; then the request is handled by server when the simulation clock is at time 603. Flowchart 4.1. illustrates the simulation process of handling the generated requests in the trace.



Flowchart 4.1. Illustration of processing the simulated request

4.1.2. Simulation Settings

The proposed P2P network is composed of one central media streaming server, multiple seeder peers, and multiple client peers. The maximum upload speed of the server is set to 500 KBps (4000 Kbps). The limited streaming capacity of the server makes service providing low cost for the video hosting companies which is ideal. Every peer is randomly assigned a specific upload and download bandwidth. Based on the study done on the bandwidth capacity of the Internet users [35], a table is provided that shows the distribution of the bandwidths between the users in 4 input traces used in this system:

Table 4.1. Users' bandwidth capacity and distribution

	Dialup/ISDN	**DSL**	**Cable/Ethernet**
Upload Bandwidth	256 kbps = 32 KBps	384 kbps = 48 KBps	768 kbps = 96 KBps
Download Bandwidth	256 kbps = 32 KBps	1500 kbps = 187 KBps	3000 kbps = 375 KBps
Share in the Network	7%	33%	60%
Number of Peers in Log #1	889	4191	7620
Number of Peers in Log #2	857	4043	7352
Number of Peers in Log #3	867	4088	7435
Number of Peers in Log #4	1050	4950	9000

For each peer including server, a limit is set for the number of parallel connections. Server may upload to 20 client peers at highest simultaneously. Peers in the network have various connections limit in accordance with their upload bandwidth; meaning that they are able to upload to that exact maximum number of client peers concurrently. Table 4.2. shows maximum number of connections for each peer in the network.

Table 4.2. Maximum number of connections for each seeder peer

	Dialup/ISDN	**DSL**	**Cable/Ethernet**
Upload Bandwidth	256 kbps	384 kbps	768 kbps
Maximum Concurrent Connections	2	3	6

4.1.3. Trace Files Description

This section presents the process of generating log files containing user requests which are used to conduct the experiments. The simulated data is generated using a distribution simulator software [58] developed based on the real data distribution in the popular multimedia serving website YouTube. Figure 4.1. shows a part of a trace file that is fed into the simulator as the input file. The number of the peers requesting short videos, the number of requested videos, and the number of requests are distinct in each trace. For each client peer a random value is given between 1 to 8 which represents the total number of video requests at specific times. There are 40000 unique popular short videos on the server side. The mean of the videos' sizes is around 10 MB and the mean of the videos' durations is approximately 5 minutes. In total, four log files are input into the simulator to simulate the user traffic in the proposed P2P model in order to compare the results with the client-server system. Three traces contain users' requests for short videos for the duration of one week and the largest trace represents users' requests for unique short videos for a two weeks period. This trace is the continuation of one of the one week trace (trace #1) that is intentionally used to present the improvement of the system performance as the time progresses.

To generate arrival time for each request in the log files, Poisson distribution is used. In general, Poisson distribution is used to model the number of arrivals over a fixed time interval where the arrivals are independent of one another. In this network, the streaming request arrival pattern follows Poisson distribution where for each peer the next request time is determined after the current access time. This ensures that for every client peer's successive request, the request arrival time occurs after the previously requested video is entirely streamed so that there are no incomplete video download during the simulation process. The abandon time for each peer is actually the time that the peer has performed required data upload and thus it is allowed to leave the network.

Arrival time	**Peer ID**	**Video ID**	**File size (KB)**	**View count**	**Duration (sec)**	**Average rating**	**Rating count**	**Selection probability**	**Related videos Number**	**Child pointer**
44	124	29386	9832	338815	492	1.000000	412	1225336748	6	16558
54	118	13973	7347	47482	469	2.000000	1145	574848974	5	12901
67	261	33304	7315	24216	140	4.000000	5	1388856255	0	0
75	1429	1743	10716	25965	313	4.000000	259	70263256	2	18675
89	2121	1743	10716	25965	313	4.000000	259	70263256	2	18675
110	1314	35156	3443	48297	68	4.000000	22	1465339454	6	5187

Figure 4.1. A part of the log file

Statistics of Log Files:

The general information regarding to every trace used to conduct the experiments are presented in Table 4.3. The example of the last two statistics is when *C1* and *C8* request to watch *V6* for

the first time, they save a replica of *V6*. If *C1* has another request to watch *V6* again, it will not make a replica of it because it already has the file. So, there are two requests to make a replica and one request to watch a video that is already saved on client's machine. Table 4.4. provides statistics of the requested videos in each trace.

Table 4.3. Trace files statistics

	Number of requests	**Number of clients**	**Number of videos**	**Number of videos requested once**	**Number of videos requested more than once**	**Number of video requests to save a replica**	**Number of video requests to watch already saved replicas**
Log #1	36141	12700	16112	10016	6096	33550	2591
Log #2	35010	12252	15799	9905	5894	32876	2134
Log #3	37594	12390	16570	10066	6504	34959	2636
Log #4	67335	15000	24000	10838	13162	61440	5895

Table 4.4. The statistics of the videos in trace files

	Mean of video sizes (MB)	**Mean of video view counts**	**Mean of video durations (min)**
Log #1	10.83	81573.2	4.28
Log #2	10.46	120600.3	4.45
Log #3	10.09	142264.2	4.36
Log #4	10.72	82070.95	4.28

4.2. Results

In this section, the results of the simulations are presented through performance analysis of this model. A specific high performance system was used to perform simulations in this research [59]. The total of 4 log files are input into the simulator to simulate the network traffic. The log files with one week users' accesses are referred as the short traces and the log file with two weeks users' accesses is referred as the long trace. It is shown how the proposed P2P model results an overall improvement in the system performance for the media server and the client

peers in the network. The main intention of this research is reducing server workload the same as other researches in this area. First the bandwidth consumption is presented. Then, the results of the network traffic in terms of the data bytes are provided. And in the end, the startup delay of the video is examined.

4.2.1. Performance Metrics

The designed and implemented P2P network simulator takes each trace file as the input and simulates the user traffic as the output. The results are acquired by analyzing and comparing the simulated network traffic of the proposed P2P short video sharing system and the traditional media streaming networks in terms of performance analysis.

To measure the performance of the proposed system, several performance metrics are applied on the simulated user traffic. The results are analyzed using the following metrics:

- Server average upload speed for each client peer: U_s
- Client peer average download speed for each requested video: D_c
- Total data bytes uploaded by server for each requested video: B_s
- Total number of fragments download in the 1st second by client peer: F_c
- Total number of client peers during simulation process: C
- Total number of client peers downloaded from server: $c \subset C$

Average download speed per client peer:

To obtain the average download speed per client peer, the summation of all requesting peers' average download speed is divided by the total number of requesting peers.

$$\text{Avg. } D_c = \sum_{i=1}^{C} D_{ci} / C$$

Average number of fragments received in the 1st second of streaming by client peer:

The average of received fragments in the 1st second of streaming is determined by the summation of all the fragments downloaded in the 1st second of downloads done by all the client peers divided by the total number of client peers.

$$\text{Avg. } F_c = \sum_{i=1}^{C} F_{ci} / C$$

Average upload speed of the server:

To calculate this average value, total server upload speed is divided by the total number of requesting peers received data from server during simulation time.

$$\text{Avg. } U_s = \sum_{i=1}^{c} U_s / c$$

Total data uploaded by the server:

This is the summation of all the video fragments bytes uploaded by the server in all downloads from server.

$$\text{Total } B_s = \sum_{i=1}^{c} B_s$$

4.2.2. Bandwidth Improvement

The overall bandwidth improvement for both client peers and streaming media server is examined. In general, users who desire to watch a video expect the video to be streamed fast enough so that they can move the playback trackbar to different parts of the video. The architecture of the proposed model provides a reasonably fast download and increases the download speed of each client peer. This is because of the peers' bandwidth contribution to the system. Moreover, the proposed replication mechanism allows users to efficiently collaborate on distributing videos over the network.

The proposed system is reliable because replicas have multiple seeder peers and scalable since TFT policy enforces seeder peers to stay in the network. We study the peers' resilience in the proposed model with regards to effectiveness of the obligatory TFT policy. The analysis of TFT results from the short traces shows that on average peers remain in the network for approximately 2 to 2.5 days (simulated time that is artificially equal to the real time) to upload as much data as they have received and then leave the network. The study on the long trace presents that peers online duration is almost 3 days. This verifies that as the number of requests increases, on average peers would stay somewhat longer in the network. This is because when the time advances forward, more peers are able to upload as much data as they have downloaded and therefore are allowed to leave the network. As a consequence, the average time peers stay in the network will slightly increase. When peers stay longer in the network (online duration), then it results a more stable overlay and a better quality of the streaming service. Therefore, the proposed P2P network is practically stable.

The analysis on the bandwidth speed from the short traces shows that in the proposed system client peers are receiving videos' fragments by using about 48.98%, 53.85%, and 51.84% more download bandwidth than the client-server system for trace #1, trace #2, and trace #3 respectively. As the time extends and the simulation progresses, more video requests take place and more seeder peers are added in the network to distribute more video replicas. The results of the long trace confirm the improvement of the download bandwidth by about 65.14% for client peers. As a result, peers contribute approximately 3 times more bandwidth for downloading videos compared to the client-server model. Figure 4.2. compares the simulations results in terms

of the average download bandwidth received by the client peers in the proposed P2P model and the client-server system.

The most important performance goal is reducing the server load that we intend to achieve. In the proposed architecture, server administrates and contributes in the network. Primarily, the central media streaming server functions as a tracker by which client peers can locate the source peers of the requested videos. Furthermore, the server's upload bandwidth is employed as seeder peers' assistant during each download session because of high bandwidth every media server utilizes.

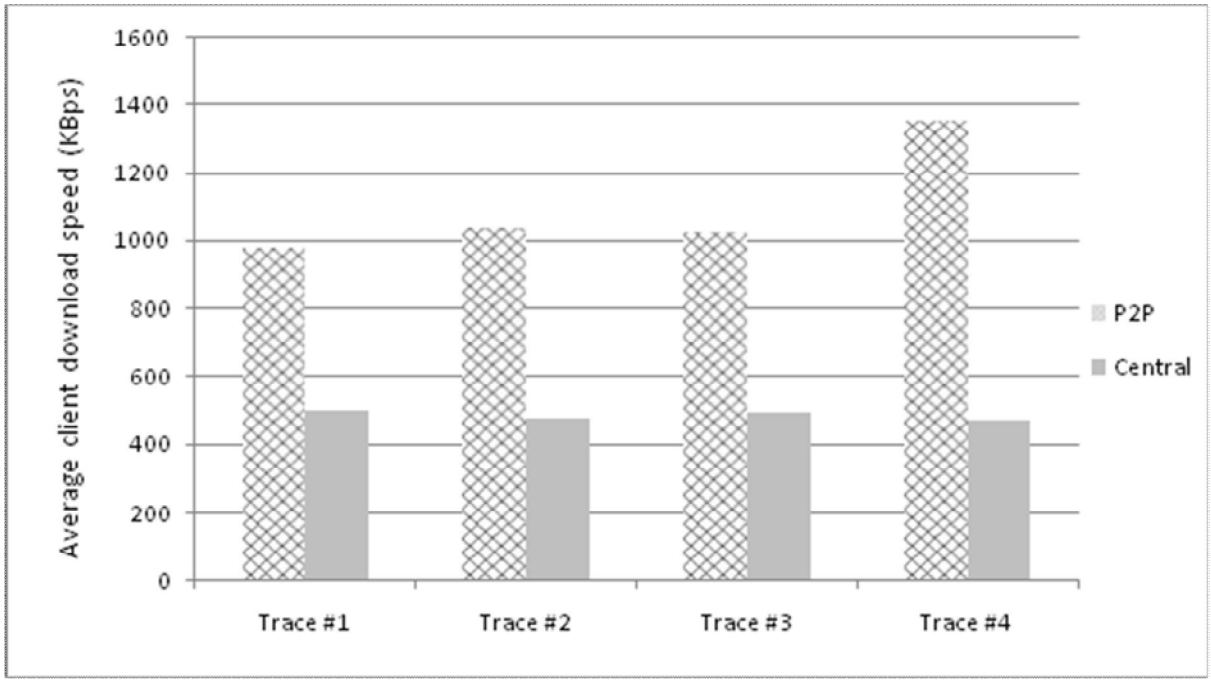


Figure 4.2. Comparison of average download speed per client peer

The results of the simulations demonstrate that this architecture decreases approximately 27.67%, 26.27%, and 28.12% of the server's upload bandwidth consumption for trace #1, trace

#2, and trace #3 correspondingly. As the simulation advances, certainly the usage of the server's upload bandwidth decreases further. This improvement is justifiable by considering the long trace results of around 36.14% less consumption of the server's upload bandwidth compared to the client-server model. This is quite an improvement in reducing the load on the server even when server is contributing in videos distributions in the network. Figure 4.3. compares the consumption of the server upload speed in the proposed P2P architecture and the client-server architecture.

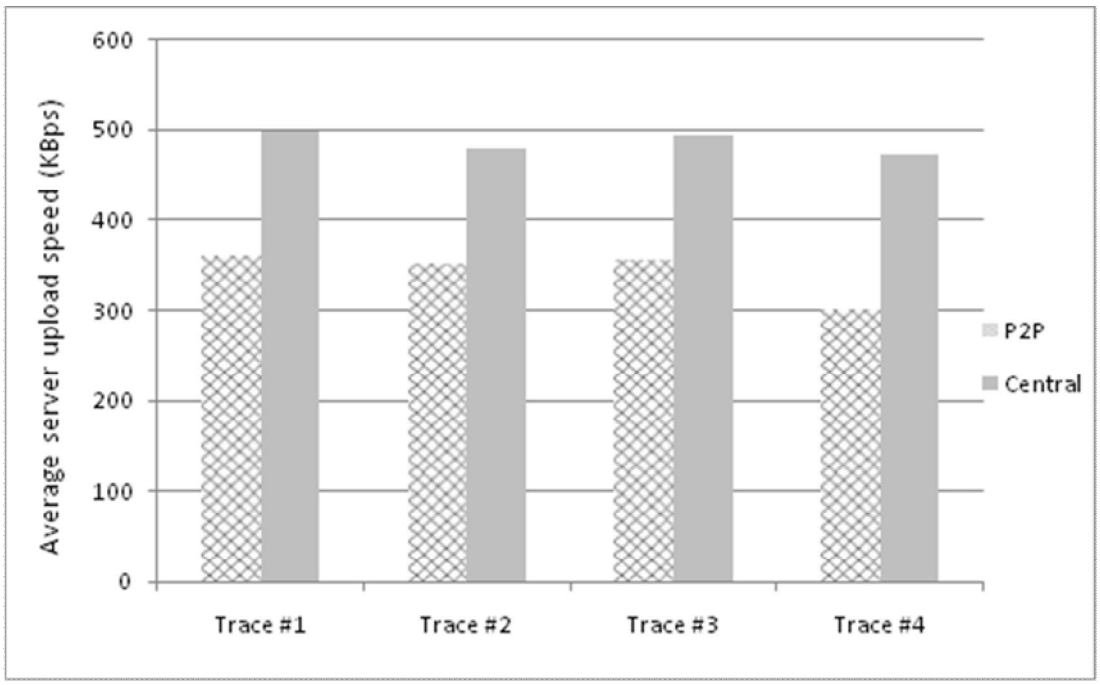


Figure 4.3. Comparison of average upload speed of the server

4.2.3. Network Traffic Improvement

As it has been justified, the consumption of server upload bandwidth is noticeably reduced in the proposed system. Furthermore, the total data that is uploaded by server is also decreased. The

results show that about 22.40%, 23.10%, and 24.31% less bytes are uploaded by server in this model using trace #1, trace #2, and trace #3 respectively as the simulator inputs. In addition, the analysis of the long trace results provides about 33.45% improvement in terms of reducing the amount of data bytes uploaded by server compared to the client-server architecture.

With regards to the server role in the proposed P2P model, server uploads data to client peers even when multiple seeder peers are providing video fragments to the user in order to have a very fast download process. As the time advances forward and more and more requests occur, the number of seeder peers available for a certain video increases. This eventually decreases the use of the server upload bandwidth and the amount of data uploaded by server in a download process in the view of the fact that seeder peers provide enough download bandwidth for a fast reception of the complete replica. This undoubtedly offloads considerable works from server in the network. Consequently, the proposed P2P architecture certainly decreases the network traffic. Figure 4.4. compares the share of the server in the download sessions during simulations in this P2P model and the client-server system.

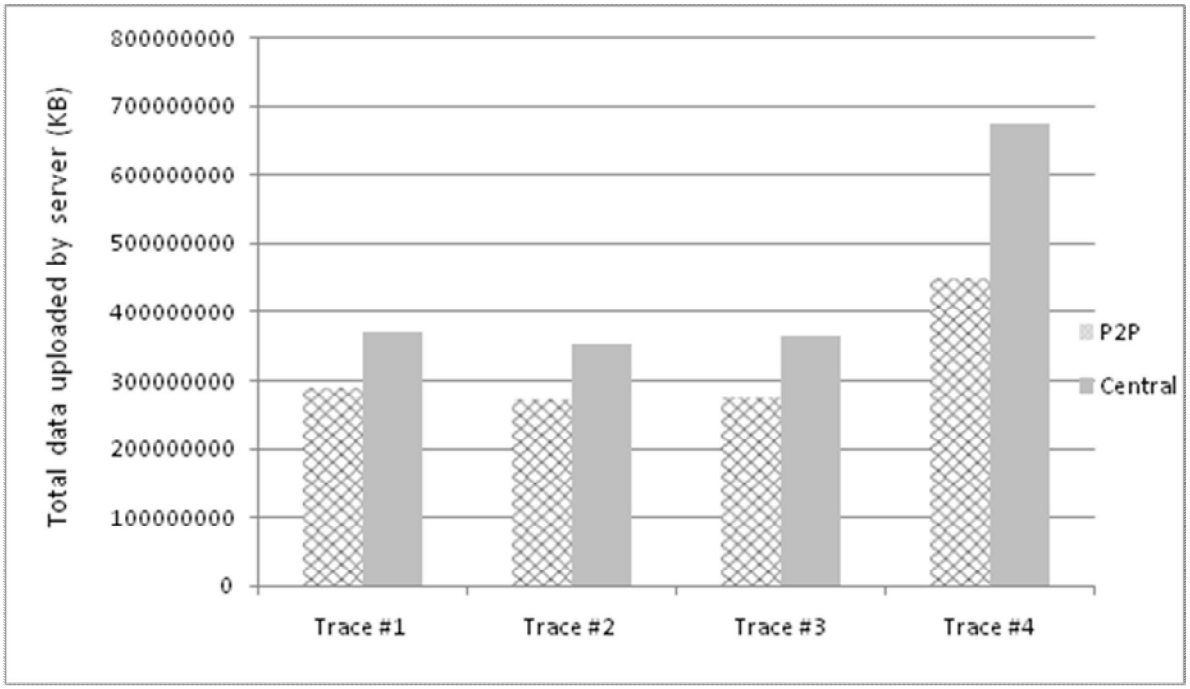


Figure 4.4. Comparison of total data uploaded by the server

4.2.4. Startup Delay Improvement

In the proposed P2P network, client peers download required fragments from multiple seeder peers and server simultaneously. These fragments are received and buffered in order at client's machine. With regards to the proposed download strategy, it is considered to downloading the pieces of the requested short video in order of the required fragments which will be needed for a continuous playback by the user. In addition, the server's upload bandwidth is utilized along with other seeder peers' upload bandwidths for fast download of the videos if necessary. This promises that all the required fragments of the requested video are received as soon as possible. Accordingly, in the proposed P2P network, the download rate is higher than the playback rate to achieve a continuous smooth playback with a small startup delay and without any interruptions. We assume that the minimum downloading rate is guaranteed for all the client peers with the intention of neglecting playback delay.

Upon requesting a video, usually the player buffer is set to hold the 5 seconds of initial data; as a result, the startup delay takes this time as a minimum which typically takes longer for viewers. So, in this simulation because of the fast download, the buffering time of 1 second is considered and therefore the number of fragments that the client peer receives in the 1st second of download process is reported. The results from the short traces demonstrate that the startup delay using trace #1, trace #2, and trace #3 is approximately 55.48%, 63.86%, and 56.85% improved respectively in this model compared to the client-server network. Moreover, the results of the long trace show about 67.83% reduction in the startup delay. The analysis of the long trace as a more stable network shows that approximately 15% of each video is received in the 1st second of download process considering the average received fragments in the 1st second in the proposed P2P model (1453 KB) and the video sizes mean (around 10240 KB). This results of downloading nearly 75% of the video in 5 seconds with regards to the minimum set startup delay if the client continues to receive this average number of fragments in every second of download process. Thus, the complete video file is downloaded in about 7 seconds if ignoring the network delay, which is close to the typical minimum set startup delay with this significant difference that in this time the complete video is received not the initial data. As a result, the distinct proposed model decreases the startup delay and assures a good playback quality. Figure 4.5. compares the

average received fragments in the 1st second of download process in the proposed model and a client-server system.

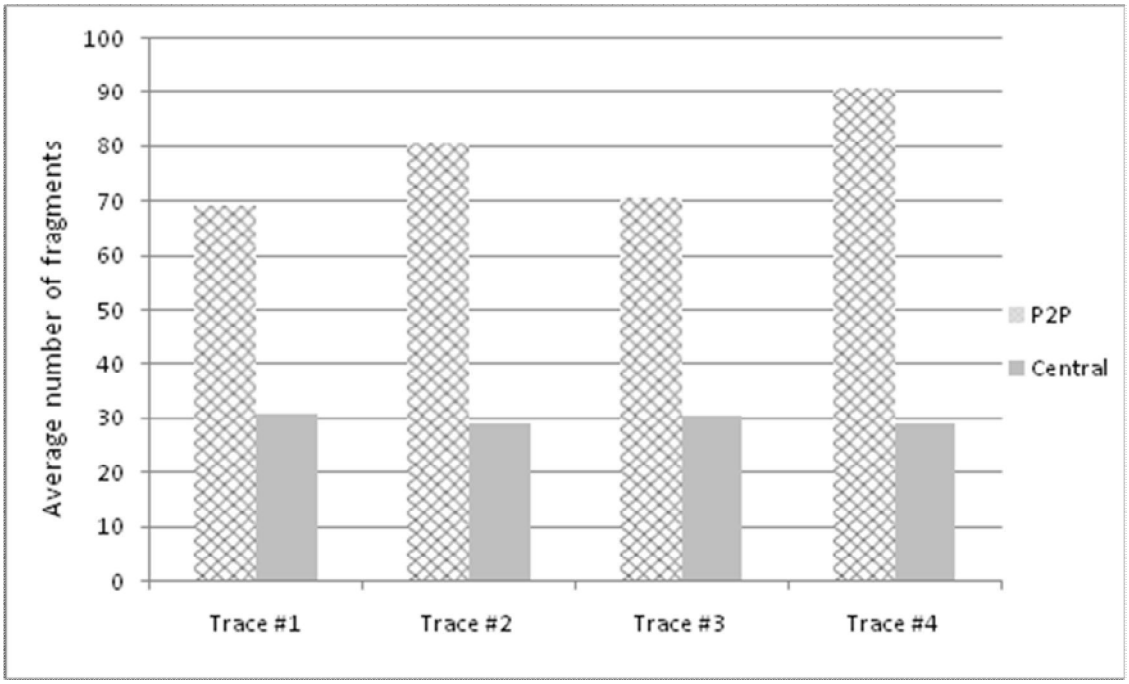


Figure 4.5. Comparison of average fragments downloaded

4.3. Conclusion

The evaluation and analysis of the proposed P2P short video sharing system show a noticeable improvement in the system performance. In this chapter, the extensive simulations and experiments performed provide the results that demonstrate the effectiveness of the proposed architecture for short video distribution in a reliable and scalable fashion. In this model, peers contribute their upload bandwidth in the network. This results a reliable and scalable system with high number of video providers in which the download bandwidth received by the users is significantly increased in a range of approximately 48.98% to 53.85% for one week of access and about 65.14% for two weeks of access. Also, this model reduces the load on the server from about 26.27% to 28.12% for one week and around 36.14% for two weeks. Moreover, the network traffic is decreased in a range of about 22.40% to 24.31% for one week and about 33.45% for two weeks. At last, the startup delay is examined in the proposed model and the results present that the startup delay is lessened approximately 55.48% to 63.86% for one week of access and about 67.83% for two weeks of access, which results providing a good playback quality.

CHAPTER 5

5. Conclusion and Future Work

This chapter presents the conclusion of this thesis by explaining our contributions and approaches in subsection 5.1. Future work is presented in subsection 5.2.

5.1. Conclusion

The rapid growth of the highly demanded video hosting websites has a considerable impact on the Internet traffic. Among all, YouTube holds the highest rank in traffic. In this thesis, an efficient and reliable P2P short video sharing system is proposed that operates similar to famous BitTorrent protocol. Our P2P system embraces a central media streaming server that performs administration tasks, manages videos' requests, and contributes in videos distribution in the network. We proposed important approaches to efficiently utilize P2P in the proposed system which include an efficient indexing scheme in terms of videos in order to locate source peers, an efficient downloading strategy for a good quality playback, a reliable content distribution mechanism, and a fairness policy to increase peers contribution in the network.

The proposed peer-to-peer system then is implemented in order to simulate the user traffic. We have performed extensive simulations and experiments to examine the system performance in this model. The simulations have been conducted by using a trace file containing generated users requests for popular short videos resembling YouTube data distribution. The analysis of the results presents that the proposed model improves the download bandwidth of client peers in a

range of approximately 48.98% to 53.85% for one week of access and about 65.14% for two weeks of access. Also, the load on the server is reduced nearly about 26.27% to 28.12% for one week and around 36.14% for two weeks, since peers are contributing their bandwidths and disk spaces in the system. In addition, the network traffic is considerably reduced in a range of approximately 22.40% to 24.31% for one week and about 33.45% for two weeks. Furthermore, there is a significant reduction in the startup delay in a range of about 55.48% to 63.86% for one week and about 67.83% for two weeks. Hence, a continuous playback without disruptions is achieved for users.

With respect to the experiment's results, it can be stated that the proposed peer-to-peer system has a reliable and scalable architecture for short video sharing and is able to support thousands of simultaneous users. The replication mechanism builds a fault tolerant system with high performance to efficiently serve short videos. This model is potentially beneficial for multimedia service providers and video hosting websites to reform their traditional client-server architecture into a P2P system that improves the system performance while reduces the hosting costs in terms of bandwidth consumption. Our simulation and analytical work provide valuable results that are useful for future development of peer-to-peer multimedia streaming systems.

5.2. Future Work

The concept of research is extending any present work. We expect to study and explore on other aspects of this research in the future work. First, it has not been taken into account any network transmission latencies or time required to establish connections to other machines. The extension of this proposed model will include various scenarios where users are facing with network delay and overlay delay. We will study the impact of any delay on the streaming time and the quality of the video and propose new approaches to minimize these latencies to the point that they are almost not noticeable for the end users.

According to the unique characteristics that YouTube has, the related videos of every requested video will be explored in order to design an efficient approach to eliminate any startup delay associated with the related videos. We consider a distinct pre-fetching strategy based on videos' view count for a smooth transition between videos with a minimum or no delay. Therefore, we expect to design and extend this work in the future by taking related videos into consideration. To address scalability issue further, server can be a super peer in the future work and real system implementation of this model. Moreover, the results of the extensions of this model can be compared with other P2P models.

This work has provided valuable results that can be further analyzed to develop future P2P networks that promise low cost service providing systems which meet users' requirements and satisfies the required Quality of Service.

References

[1] YouTube – Broadcast Yourself
http://www.youtube.com/

[2] BitTorrent
http://www.bittorrent.com/

[3] Cisco Visual Networking Index: Forecast and Methodology, 2009-2014.
http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[4] Alexa – the Web Information Company
http://www.alexa.com/

[5] YouTube Fact Sheet
http://www.youtube.com/t/fact_sheet

[6] Xu Cheng and Jiangchuan Liu. "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing". *INFOCOM 2009, IEEE*, pp. 1152-1160, 2009.

[7] Changqiao Xu, Gabriel-Mario Muntean, Enda Fallon, and Austin Hanley. "Distributed Storage-Assisted Data-Driven Overlay Network for P2P VoD Services". *IEEE Transactions on Broadcasting*, vol. 55, no. 1, pp. 1-10, 2009.

[8] Yung Ryn Choe, Derek L. Schuff, Jagadeesh M. Dyaberi, and Vijay S. Pai. "Improving VoD Server Efficiency with BitTorrent". In *Proceedings of the 15th International Conference on Multimedia (MULTIMEDIA'07)*, pp. 117-126, 2007.

[9] San Murugesan. "Understanding Web 2.0". *IT Professional, IEEE*, vol. 9, no. 4, pp. 34-41, 2007.

[10] MySpace
http://www.myspace.com/

[11] Twitter
http://twitter.com/

[12] Flicker – Photo Sharing
http://www.flickr.com/

[13] Facebook
http://twitter.com/

[14] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang. "Towards Cinematic Internet Video-on-Demand". In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems (Eurosys'08)*, pp. 109-122, 2008.

[15] Ricardo Baeze-Yates. "User Generated Content: How Good Is It?". In *Proceedings of the 3rd Workshop on Information Credibility on the Web (WICOW'09)*, pp. 1-2, 2009.

[16] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, Sue Moon. "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System". In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC'07)*, pp. 1-13, 2007.

[17] Metacafe – Online Video Entertainment
http://www.metacafe.com/

[18] Tudou – Online Video Website
http://www.tudou.com/

[19] Mojgan Soraya, Masood Zamani, and Abdolreza Abhari. "Modeling of Multimedia Files on the Web 2.0"". *21th IEEE Canadian Conference on Electrical and Computer Engineering 2008 (CCECE'08)*, pp. 001387-001392, 2008.

[20] Bram Cohen. "Incentive Build Robustness in BitTorrent". In *Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems*, pp. 1-5, 2003.

[21] Alex Sherman, Jason Nieh, and Clifford Stein. "FairTorrent: Bringing Fairness to Peer-to-Peer Systems". In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'09,)* pp. 133-144, 2009.

[22] Zuhui Ma and Dongyn Qiu. "A Novel Optimistic Unchoking Algorithm for BitTorrent". *6th IEEE Consumer Communications and Networking Conference (CCNC 2009)*, pp. 1-4, 2009.

[23] Rudiger Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications". In *Proceedings of the First International Conference on Peer-to-Peer Computing*, pp. 101-102, 2001.

[24] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems Principles and Paradigms* (2nd ed.). USA: Prentice Hall, 2007, ISBN: 0136135536.

[25] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan**. "**Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications". *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17-32, 2003.

[26] Napster – Digital Music Service
http://www.napster.ca/

[27] Kazaa
http://www.kazaa.com/#

[28] Gnutella
http://rfc-gnutella.sourceforge.net/

[29] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. "Making Gnutella-like P2P Systems Scalable". In *Poceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03)*, 2003.

[30] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet* (3rd ed.). USA: Addison-Wesley, 2004, ISBN: 0321227352.

[31] Mong-Fong Horng and Yau-Hwang Kuo. "Dynamic Slot Allocation to Control Delay in TDMA Wireless Base Station". In *Proceedings of Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, pp. 1126-1131, 2003.

[32] Jean-Paul Wagner and Pascal Frossard. "Playback Delay Optimization in Scalable Video Streaming". *IEEE International Conference on Multimedia and Expo (ICME 2005)*, pp. 860-863, 2005.

[33] Averill Law and W. David Kelton. *Simulation Modeling and Analysis* (3rd ed.). USA: McGraw-Hill, 2000, ISBN: 0070592926.

[34] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York: Wiley-Interscience, 1991, ISBN: 0471503361.

[35] Cheng Huang, Jin Li, and Keith W. Ross. "Can Internet Video-on-Demand be Profitable?". In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'07)*, vol. 37, no. 4, pp. 133-144, 2007.

[36] MSN
http://video.msn.com/video.aspx/

[37] Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose. "Watch Global, Cache Local: YouTube Network Traffic at a Campus Network – Measurements and Implications". In *Proceedings of the 15th SPIE/ACM Multimedia Computing and Networking (MMCN'08)*, vol. 6818, pp. 1-13, 2008.

[38] Yi Zheng, Dan Huang, Wei Zhu, Xiaobing Zhang, Yishuai Chen, and Changjia Chen. "A Measurement Study of P2P VoD System". *International Conference on Research Challenges in Computer Science (ICRCCS'09)*, pp. 174-177, 2009.

[39] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, John C.S. Lui, and Cheng Huang. "Challenges, Design and Analysis of a Large-Scale P2P-VoD System". In *Proceedings of the ACM Conference on Data Communication (SIGCOMM'08).* Vol. 38, no. 4, pp. 375-388, 2008.

[40] CoolStreaming – Broadcast Your TV
http://www.coolstreaming.us/hp.php?lang=en

[41] PPLive
http://www.pplive.com/en/index.html

[42] Kai Wang and Chuang Lin. "Insight into the P2P-VoD System: Performance Modeling and Analysis". In *Proceedings of 18th International Conference on Computer Communications and Networks (ICCCN 2009)*, pp. 1-6, 2009.

[43] Joost
http://www.joost.com/

[44] PPStream
http://www.ppstream.com/

[45] GridCast
http://globalmediaservices.net/products/gridcast-tv/

[46] Weifang Liang, Jihai Huang, and Jianhua Huang. "A Distributed Cache Management Model for P2P VoD System". *2008 International Conference on Computer Science and Software Engineering*, vol. 3, pp. 5-8, 2008.

[47] Wenbin Jiang, Chong Huang, Hai Jin, and Xiaofei Liao. **"**A New Proxy Scheme for Large-Scale P2P VoD System". *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC'08)*, vol. 1, pp. 512-518, 2008.

[48] Lei Guo, Songqing Chen, and Xiaodong Zhang. "Design and Evaluation of a Scalable and Reliable P2P Assited Proxy for On-Demand Streaming Media Delivery". *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, no. 5, pp. 669-682, 2006.

[49] X.Y. Yang, P. Hernandez, F. Cores, L. Souza, A. Ripoll, R. Suppi, and E. Luque. "DVoDP2P: Distributed P2P Assisted Multicast VoD Architecure". *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

[50] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. "P2Cast: Peer-to-Peer Patching Scheme for VoD Service". In *Proceedings of the 12th international conference on World Wide Web*, pp. 301-309, 2003.

[51] Diego Gallo, Charles Miers, Vlad Coroama, Tereza Carvalho, Victor Souza, and Per Karlsson. "A Multimedia Delivery Architecture for IPTV with P2P-based Time-Shift Support". *6th IEEE Consumer Communications and Networking Conference (CCNC 2009)*, pp. 1-2, 2009.

[52] Daniel Hughes, Geoff Coulson, and James Walkerdine. "Free Riding on Gnutella Revisited: The Bell Tolls". *Distributed Systems Online, IEEE*, vol. 6, no. 6, 2005.

[53] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. "Some Observations on BitTorrent Performance". In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'05)*, pp. 398-399, 2005.

[54] J.J.D. Mol, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. "Free-Riding, Fairness, and Firewalls in P2P File-Sharing". *Eighth International Conference on Peer-to-Peer Computing (P2P'08)*, pp. 301-310, 2008.

[55] Saikat Guha and Paul Francis. "Characterization and Measurement of TCP Traversal through NATs and Firewalls". In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*, pp. 199-211, 2005.

[56] Andrew Biggadike, Daniel Ferullo, Geoffrey Wilson, and Adrian Perrig. "NATBLASTER: Establishing TCP Connections between Hosts behind NATs". In *Proceedings of ACM SIGCOMM Asia Workshop*, 2005.

[57] Dongyu Qiu and R. Srikant. "Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks**".** In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04)*, pp. 367-378, 2004.

[58] Abdolreza Abhari and Mojgan Soraya. "Workload Generation for YouTube". *Multimedia Tools and Applications Journal*, vol. 46, pp. 91-118, June 2009.

[59] Shared Hierarchical Academic Research Computing Network (SHARCNET) http://www.sharcnet.ca/