A Simulation Study on Performance of Video-On-Demand Systems

Amirhamzeh Kariminasab

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
December 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işik Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işik Aybay
Supervisor

Examining Committee

1. Prof. Dr. Işik Aybay
2. Assoc. Prof. Dr. Muhammed Salamah
3. Asst. Prof. Dr. Gürcü Öz

ABSTRACT

Within the scope of this study, three implementation techniques in the field of Video-on-Demand systems (LCBBS, Patchpeer and network aware) are chosen. Then, three common parameters (throughput, number of stops and acceptance ratio) for each technique have been selected. Simulation results are used for comparing these three techniques with regards to efficiency. NS software is used for implementing these simulations.

Results show that, the Patchpeer technique exhibits the best performance, and the LCBBS technique has the second best performance.

Keywords: Video-on-Demand, Patchpeer, LCBBS, Network-Aware, Peer to Peer

ÖZ

Bu çalışmanın kapsamı içerisinde, isteğe bağle video izleme sistemleri (VoD) için üç teknik uygulama (LCBBS, Patchpeer ve Network-aware) seçildi. Sonra üç parameter ile (birim zamanda iş miktarı, duraklama sayısı ve kabul oranı) her seçilen teknik için, simülasyonlar yapılarak, üç tekniğin etkinlik bakımından karşılaştırılması yapıldı. Bu simülasyonların uygulamasında NS yazılım kulanıldı.

Sonuç olarak, üç metod içinde Patchpeer tekniği en iyi performansı, LCBBS ikinci en iyi performansı sergiledi.

Anahtar kelimeler: isteğe bağle video izleme sistemleri, Patchpeer, LCBBS, Network-Aware, komşudan-komşuya

I dedicate my dissertation work to my loving parents. A special feeling of gratitude to my adorable father who supports me throughout my entire life.

I also dedicate this dissertation to my many friends who have supported me throughout the process. Reza, Pouria, Behnam, Hemn, Hamed, Masoud and Mehran. I will always appreciate all they have done.

I dedicate this work and give special thanks to my best friend forever, my wife Azadeh and my wonderful daughter Nafiseh and amazing son Mohsen, for being there for me throughout my entire study.

ACKNOWLEDGMENT

I would like to express my special appreciation and thanks to my supervisor Professor Dr. Işik Aybay, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for. I would also like to thank my committee members, for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

A special thanks to my family. Words cannot express how grateful I am to my mother-in law, father-in-law, my mother, and father for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank all of my friends who supported me in writing, and incented me to strive towards my goal. At the end I would like express appreciation to my beloved wife AZADEH who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries. Last but not least to my sweet twins, MOHSEN and NAFISEH who made my student life sweet and lovely.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

A Video on Demand (VoD) system has three components: clients, servers and interconnection networks. It has the aim of delivering multimedia content to one client or a set of clients that may ask for service at any given time. Those clients who are receiving, decoding and displaying the down-streamed multimedia content expect to receive a synchronized and continuous stream. Most of VoD services are currently based on the client/server paradigm [1].

In recent years, VoD has gained interest in industrial and academic environments as a commercial service. One example for VoD service is "YouTube" that became popular by the increased Internet bandwidth. Some VoD systems serve a restricted number of clients. VoD systems with a larger number of clients are more interesting. One can build a large-scale VoD system, which can be distributed in city regions or countries or even across continents by using Internet, but one should always take care of Quality of Service (QoS), which is an important part of VoD service.

In traditional VoD systems, users are simply connected to servers on the Internet, send their requests and download the requested data. However, continuing increase in demand for using the VoD service causes bottlenecks since we need more bandwidth to fulfill all users' demands. As a solution to this problem, researchers suggested to use Peer-to-Peer (P2P) VoD system for real-time video streaming. A P2P VoD system enables to serve large number of peers concurrently. The main idea here is that peers must cache some parts of videos instead of removing video data after watching videos, so that these parts can be used by other peers when they make requests to receive those parts. Therefore, in this system, peers are not only passive receivers, but they can also act as active senders. The P2P VoD system is based on interaction between peers by caching and sharing video segments with other peers through the network in order to reduce the traffic on the server. This improves the performance of video downloading on Internet by the help of participating peers.

In this study, we have compared three different VoD techniques, namely, Patchpeer, LCBBS and Network-aware by using the NS simulator [2] to compare their performance by considering three different evaluation parameters. Graphs and discussions related to simulation are given in Chapter four.

The rest of this thesis is organized as follows.

In Chapter 2, we give an overview of related methods proposed for VoD systems. In Chapter 3, we discuss the three simulated methods (LCBBS, Patchpeer and network-aware) in detail. Chapter 4 includes the results of simulations with graphs and their discussions. The last chapter covers the overall results of the study, conclusions and future work.

Chapter 2

LITERATURE REVIEW

In today's multimedia systems, clients expect streaming systems to be both reliable and fault tolerant. There are three categories of multimedia streaming applications: on-demand streaming, such as video-on-demand; live streaming, such as Internet television; and real-time interactive streaming as in video conferencing and on-line gaming.

This section includes a discussion of Video on Demand systems, their components, and recently proposed systems for improving VoD performance. We start by definitions of basic terms.

2.1 Local Area Network (LAN)

A LAN [3] is composed of an inter-connected network of personal computers and workstations in different locations like office buildings, schools or homes that are able to share data and use common devices such as scanners, data storage devices and printers anywhere on the LAN. Most LANs are built with relatively inexpensive hardware such as network adapters, Ethernet cables, and hubs. The main factors on which we can classify LANs are data transfer and communication rates.

Although LANs are capable of transmitting data at fast rates, the distances are limited, and there is also a limit on the number of computers that can be attached to a single LAN. Figure 2.1 shows the LAN environment

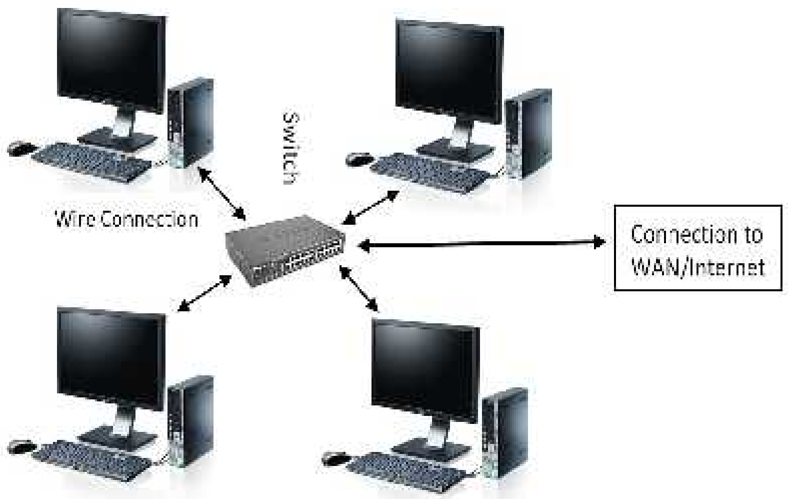


Figure 2.1: Illustration of Local Area Network

2.2 Wide Area Network (WAN)

Most LANs are confined to a single building or group of buildings; One LAN can be connected to other LANs via routers, public communication networks, such as the telephone system, radio waves, leased fiber optic lines or satellites. A system of LANs interconnected in this way is called a wide-area network (WAN) [3]. The largest WAN in existence is the Internet, which is actually a network of WAN's.

Figure 2.2 shows the WAN environment, which is consist of different LAN.

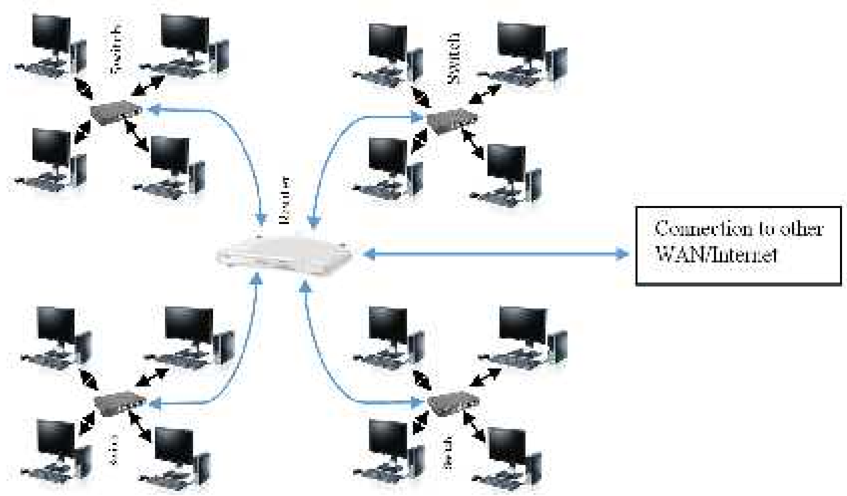


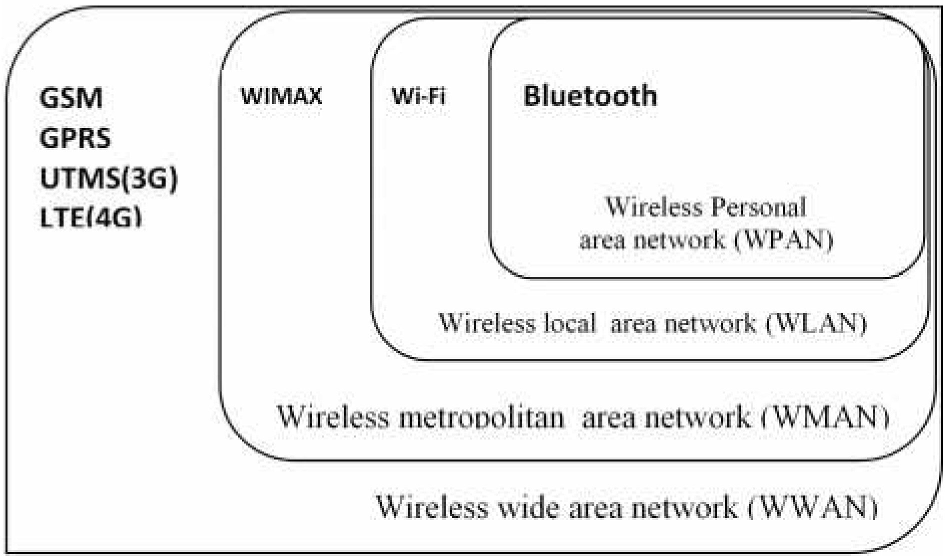
Figure 2.2: Illustration of Wide Area Network

2.3 Wireless LAN (Wi-Fi)

Wi-Fi [3] is a wireless alternative to traditional wired networking that relies on cables to connect networkable devices together. Wireless technologies are widely used in both home and business computer networks. A wireless computer network offers several distinct advantages compared to a wired network but it also has some disadvantages. Advantages of wireless connections include mobility, portability and freedom of movement, and elimination of cables. Disadvantages of wireless connections include additional security concerns, and the potential for radio interference, due to weather, other wireless devices, or obstructions like walls.

There are different types of technologies for transferring data wirelessly, which can be selected based on limitations for designing a network at a place. In Figure 2-3, recent wireless technologies are listed by considering their coverage .These are, wireless personal area network (WPAN), wireless metropolitan area network (WMAN), wireless local area network (WLAN) and wireless wide area network (WWAN).

Figure 2.3 shows the classification for Wireless technologies.


GSM
GPRS
UTMS(3G)
LTE(4G)
WIMAX
Wi-Fi
Bluetooth
Wireless Personal area network (WPAN)
Wireless local area network (WLAN)
Wireless metropolitan area network (WMAN)
Wireless wide area network (WWAN)
Figure 2.3: different technologies for Wireless LAN

Between wireless network technologies, WWAN has witnessed rapid growth recently. One application of WWAN is the cellular network which is a radio network distributed over land through “cells”. Each cell includes a fixed location transceiver known as the “base station”. These cells together provide radio coverage over larger geographical areas. User equipment, such as mobile phones, is therefore able to communicate even if the equipment is moving through cells during transmission.

Cellular network technology supports a hierarchical structure formed by the base transceiver station (BTS) of individual cells, mobile switching center (MSC), location registers and the public switched telephone network (PSTN). The BTS enables cellular devices to make direct communication with mobile phones. This unit also acts as a base station to route calls to the destination base center controller. The base station controller (BSC) coordinates with the MSC to interface with the landline-based PSTN, visitor location register, and home location register to route the calls toward different base center controllers.

Cellular networks maintain information for tracking the location of mobile devices. In response, cellular devices are equipped with the details of appropriate channels for signals from the cellular network systems.

The base station is responsible for monitoring the level of signals when a call is made from a mobile phone. When the user moves away from the geographical coverage area of the base station, the signal level may fall. This can cause a base station to make a request to the MSC to transfer the control to another base station that is receiving the strongest signals without notifying the subscriber; this operation is called “handover” or “handoff”.

2.4 Video on Demand

VoD is a framework, which permits clients to choose and watch/listen to video or sound substance whenever they want, instead of the need to watch at a particular broadcast time. The VoD idea is not new. The first commercial VoD was dispatched in Hong Kong in the early 1990s. In the United States, Oceanic Cable of Hawaii was the first to offer it starting in 2000. Video content remains on the service provider's network and only the program the customer selects is sent to the customer’s home. When a viewer changes the channel, a new stream is transmitted from the provider's server directly to the viewer. IPTV innovation is frequently used to bring video on demand to TVs and PCs. IPTV (Internet Protocol television) can be delivered either live TV or stored video. IPTV basically uses multicasting with Internet Group Management Protocol (IGMP) for live TV shows and Real Time Streaming Protocol for on-demand programs. A TV VOD system can stream content through a set-top box. Alternatively, the service can be delivered over the Internet to home computers, portable computers, high-end cellular telephone sets, video recorders or portable media players (PMP), permitting continuous viewing, or it can be downloaded to a gadget, to watch it later.

The simplest approach to stream video data in a WLAN to users is to create a unicast connection from the server to each requesting user, where the video server is situated behind the base station. Given the high-bandwidth requirement of video data, this approach will quickly consume all available bandwidth of the downlink communication when multiple users request the video service.

There are some functions for VoD systems, which are useful for users like fast rewind, fast forward, slow rewind, fast rewind jump to future/previous frame, etc. For those systems that store and stream the programs from disk drive, these functions ask for additional storage and processing on server side since separate files for fast rewind and forward should be stored on disk or other systems that are memory based. These functions can be run directly from memory, which will put less burden on storage and processing time. Putting video data on LAN is also possible which will gives faster response to clients. Also by using WAN and a video streaming server, it is possible to serve a bigger community.

2.5 Related works

Several researches has studied how to reduce the amount of data transferred from remote servers. In VoD systems, an example is the Peer-to-Peer technique, which is successful in live streaming and down streaming data from other peers.

For VoD, the upcoming video part (segment) is more vital than a later one during file downloading. For decreasing the server's workload, the P2P system tries to utilize the peer's upload bandwidth. By using Peer-to-Peer technique in a LAN consisting of peers, we can decrease the server load. At the beginning, we may not have copies of requested videos and we should download them from remote servers, However as time passes, buffers of peers in the network will be filled with “popular data”, and as a result, the number of references to remote servers will be reduced. Some researchers have proposed increasing the efficiency of VoD systems by using the P2P technology. Some examples of this approach are the client back-end buffering system (LCBBS) [4] [5], data prefetching [6], Patch-peer [7], and Network-Aware systems [8].

There are some proposed techniques, which are similar to P2P VoD systems, but with different procedures in caching, segmentation and sharing data over the network .One of these techniques is the Video Locality Based Buffering Mechanism (LCBBS), that performs well in P2P VoD system. In LCBBS, like other Peer-to-Peer media streaming systems, a part of video will be playing while the remainder of it is downloading; The aim is to help other peers in the network by sharing their cached video segments with the requesting member. This requesting member who is getting data from other peers also saves the downloaded segments for later use by other peers [4] [5].

The Patchpeer method has a similar procedure to increase the QoS, but it is implemented in a mixed environment by integration of WLAN and WWAN networks. It uses a Peer-to-Peer system by making use of the cached data in clients. With the integration of WLAN and WWAN, there can be more clients than LCBBS with the same QoS. Using a carrier mobile network, we can have high availability assurance, while by using mobile Peer-to-Peer network we will have opportunistic use of resources [7].

Patching is a technique that utilizes the multicast service at the network layer to enable on-demand service on the Internet for videos with diverse popularity. By using patching, the user is capable of joining a multicast to download the missed portion of a video over a dedicated patching stream. This technique should be implemented on a media server. Each mobile device has two interfaces, enabling the device to communicate both in the WWAN and WLAN modes. Through WLAN, mobile devices are capable of communicating with Peer-to-Peer network and by WWAN; they can connect to the base station. However, there are two challenges here: the first one is the integration of WWAN and WLAN, and the second one is implementing VoD service on this integrated network.

There are two fundamental characteristics in a mobile wireless hybrid network, which influences the design of Patchpeer method. These are node mobility, and the shared nature of the wireless medium in single-channel networks. Node mobility can break the communication sessions between mobile nodes in the WLAN, and change the network condition, such as the number of on-going communication sessions in a cell. Within a cell, there can only be a certain number of concurrent communication sessions if we want to maintain the QoS of the current streams.

There are two challenges for the Patchpeer design. Firstly, a requesting node has to select the neighboring node from which it will receive the patching stream. If the patching stream is provided by a multi-hop neighbor, the stream is subjected to long delays and frequent link breaks due to node mobility. On the other hand, if the patching stream is to be provided by a 1-hop neighbor, the list of patching stream providers is limited for a requesting client. Secondly, nodes in a WLAN contend with each other for the medium; hence, a neighboring node needs an admission control mechanism when admitting a patch request from a requesting node. Moreover, since nodes in a

WLAN operate in a distributed fashion, the admission control mechanism should also be distributed to avoid overhead.

A requesting peer has three main tasks. Firstly, the peer must find out the start time of the last regular multicast of the requested video. Secondly, the requesting peer must find a patching stream provider among its 1-hop neighbors. Thirdly, the peer downloads and plays the video. The main operation of the server is to decide whether a patching stream or a regular stream is needed for a video request.

Here, the definition of regular and patching streams are as follows:

> Regular stream: download from an existing multicast for remainder of the video.
>
> Patching stream: download from a neighbor node for the missed portion of video.

Each peer in Patch-peer method caches a fixed size part of the video it is watching in a forwarding buffer (fb), so that this video content in their fb can provide the patching stream to late peers in their neighborhood.

A patching stream provider has to satisfy two conditions: 1– Data Condition: the fb of stream provider must hold enough data of the requested video. 2– Network Condition: there must be enough bandwidth in the WLAN to support the new data flow without compromising the QoS of existing data flows.

To select a patching stream provider from the set of candidate peers, the requesting client can use one of the following heuristic selection methods:

1– Random Selection: it randomly selects the patching stream provider in the set of candidate peers.

2– Earliest Response: it selects the neighboring peer who responds to the Request Patch message at the earliest time. The advantage of this selection method is it does not have to wait for the timeout period to expire to start receiving the patching stream.

3– Closest Peer: this is possible only if mobile nodes have knowledge of their locations, i.e. mobile nodes are equipped with positioning devices such as Global Positioning System (GPS). In this selection method, the client (source node) selects the candidate that is closest to destination node. The idea is the close distance means two mobile nodes will be within the transmission range for a longer time.

4– Most Available Bandwidth: it selects the patching stream provider with the most available bandwidth among the set of candidate peers. When other communicating nodes move into the communication range of the patching stream provider, the data rate of the patching stream may be affected.

To evaluate the Patch-peer method under different environment conditions, the authors of [7] carried out a sensitivity analysis study with respect to the following simulation parameters:

1– Buffer Size: There are two types of buffers employed at a peer, a forward buffer and a playback buffer. A larger forward buffer makes an early peer an eligible patch provider to more late peers, and a larger playback buffer to some extent affects the optimal patching window calculated by the server.

2– Request Arrival Rate: Higher request arrival rate means more requests during the same period. A higher request rate could easily overload the server in the original Patching technique, but this could be substantially improved in Patchpeer technique as neighboring peers can provide the patching streams to late peers.

3– Node Mobility: Node mobility does not affect the original Patching technique, but it does play an important role in the Patchpeer method. On the negative side, mobility can break the connection of a patching stream, or it can bring too many active patching streams within the contention area of each other, affecting the delivery rate of each stream. On the positive side, mobility helps bring peers with patching data closer to video requesting peers.

4– Video Length: System resource is held up longer when serving a long video. Consequently, longer videos tend to have negative impact on the acceptance ratio.

5 – Seed Peers: Seed peers represent peers that already have the prefix of the video before the simulation starts. These seed peers model the “steady state” of the system where after the initial starting phase of the system are always some peers that already finish watching the video and remain in the system.

6– Node Density: They keep the number of nodes the same, and vary the size of the geographical area to study the effect of different node densities. A larger area means a more sparse population, while a smaller area means a denser population.

We can conclude that, the Patchpeer method is a video-on-demand delivery technique for wireless environments that overcomes the limitations of the original Patching technique. The Patchpeer technique offers solutions to several challenging issues, most notably the handling of node mobility and admission control in shared single-channel wireless medium networks. The performance study in [7] demonstrates that Patchpeer is better than Patching on acceptance ratio under a wide range of environment settings.

In [8], the authors proposed a Network-aware multicast scheme to supply instantaneous VOD (Video-on- Demand) services on LAN. This system is implemented based on a Peer-to-Peer Grid. By taking advantage of the large storage and the powerful processing capability in client-side devices, the user host serves as both a client and a non-dedicated video server. Peer groups are formed as a collection of peers that share similar interest, and workload may be fairly apportioned among autonomous groups. If two nodes can provide video services to each other, they will have similar interest. Video index information helps nodes to be aware of the interest of their peers. In this VOD system, the adaptive video delivery mainly employs a new dynamic buffering algorithm and an improved video multicast strategy to achieve an optimal utilization of system resources, and the network-aware adaptation makes the system more robust. In cooperating with each other, the relevant servers can supply instantaneous video services for local users.

Other Peer-to-Peer systems are discussed in [9] [10] [11].

In [9], “Y.Zhang and H.Wang” proposed a basic model for a large-scale P2P VoD system that includes a large number of peers with a server. They proposed a model in which a file is divided into a number of segments and encoded for playback at a specific rate. In their system, the server stores all segments of the file with the assumption that the file is able to meet any need from the peers. They also assumed that time is slotted, so during each time slot, a peer can simultaneously provide a limited number of upload connections. Each peer in their system is not only downloading data, but it can also upload data for other peers.

The simulation hypothesis of [9] is configuration parameters for all peers are the same and once a peer finishes the in-order playback it leaves the system. Also, 500Kbps is assumed to be the rate for playback. Based on the results, when the traffic on server is high, peers can handle the system by assisting each other and playback duration does not have much effect on the system.

In another study [10], “Y.He and L.Guan” proposed a system architecture in which, in order to reduce the traffic on server some “helpers” are contributing in the system.

The system proposed in [10] has three types of components: the server, the peers and the helpers. In this system, each peer has a connection with the server without any transitional hub (intermediate node). The helper is a node or a peer that is in an “inactive” status that can help the system for sharing videos.

The simulation hypothesis of [10] is: there are DSL/Cable and Ethernet peers as two types of peers, 3Mbps as server upload capacity 85% of all peers in this system are considered as connected by DSL/cable and 15% by Ethernet with different upstream and downstream rates. Helper peers if not particularly specified is assumed to be 10% of the total number of peers. With all these assumptions, the authors did their simulation under different conditions and their results showed that helpers are enhancing the upload bandwidth of entire system. Also their system improved the streaming capacity in contrast with P2P VoD system without interference of helpers.

In another study, "Z.Lu, S.Zhang and J.Wu" [11] proposed a system architecture for P2P VoD system with the following features: At network layer, there is Unicast IP; by using a utilization method, the system handles asynchronous connections of clients. There is a variable FIFO buffer size per each user. Each user can forward the media file segment to a new user by checking the bandwidth. Based on this implementation, by applying video segmentation with 200Kb size and 100Mbps bandwidth, they concluded that by expanding the quantity of peer nodes in Peer-to-Peer VoD system, the performance would be higher in contrast with system using a central VoD server.

Due to asynchronous interactive behaviors of users and the dynamic nature of peers, achieving efficient random access operation in on-demand video streaming in P2P system is difficult. In [12] authors proposed a network coding equivalent content distribution (NCECD) scheme to efficiently handle interactive and random access VoD operations in P2P systems. In this system, a new client only needs to connect to a sufficient number of parent peers to be able to view the whole video and it rarely needs to find new parents when performing random access operations.

In NCECD, videos are divided into segments that are then further divided into blocks.

These blocks are encoded into independent blocks that are distributed to different peers for local storage. The authors conclude that because of the nature of coding technique, they have failure-tolerant streaming services. A client on the system is connected to multiple parent peers who have stored equivalent media data, and peers are able to stream media data in parallel. Simulation and analysis studies demonstrate that the proposed scheme outperforms other competing schemes such as VMesh, BBTU, and DSL in terms of startup delay, jumping delay, and server stress. Additionally, NCECD can achieve very low block loss probabilities under various system parameters by connecting to an appropriate number of extra parent peers, and can allow for failure-tolerant streaming services in a P2P network [12].

In [13], the authors consider various issues on wireless access networks including problems of providing an efficient and low-cost video streaming service. They indicate that to provide video on demand services over wireless networks, one has to combine the popularity-based video caching with multicast content delivery.

They proposed a content aware architecture, which tackles the speed bottleneck by using a multicast service. It also uses popularity-dependent video caching and patching with application enabled multicast content delivery. The focus in this study is on cellular networks, though the proposed method is applicable for any wireless network that supports multicast. While this method is hard to be implemented in network backbone, its implementation at the access part is easier. The proposed method offers a better quality of service in terms of the expected latency for content delivery.

Due to the limited bandwidth of mobile networks, mobile IPTV has some limitations such as channel zapping time in order to use the channel and bandwidth efficiently.

Also Mobile IPTV cannot transmit all channels at the same time due to lack of network bandwidth. Mobile IPTV services generally use IP multicast technology to reduce duplicate data transmission over the network.

In [14] the authors proposed a scheme to reduce channel-zapping time and increase QoS of mobile IPTV service, and as they are claiming, their scheme is simple but effective. The proposed scheme uses the user channel scheme and network-aware rate control scheme and their experimental results confirms their system is effective.

The authors of [14] proposed an adaptive channel control scheme, which not only controls the pre-buffering channel based on the user behavior, but also adjusts the transmission rate of mobile IPTV based on the mobile network state. Therefore, we can reduce the channel zapping time and improve the user perceived QoS in mobile networks.

In [15], the authors have done an implementation of Peer-to-Peer TV (PPTV) over 3G networks. Their study was based on experience and real measurements obtained from PPTV system on the impact of PPTV characteristics on P2P video streams. The growth in both video content networks and number of users, and developing electronic devices lead to ubiquitous access of users who want to enjoy their favorite videos on their mobile devices (e.g., cellphones, tablets) whenever and wherever they wish.

So far, most studies concentrated on limitations of the video on demand system with fixed scenarios i.e., stationary users. Although, P2P TV became a popular method in fixed Internet, mobile video services are still based on clients-server model. P2P TV is part of a new generation of P2P applications that combine P2P and Internet TV, called P2P TV. Due to these issues, a new method has been introduced by applying P2P technology into mobile video system in 3G mobile networks. Moreover, the challenges of applying P2P paradigm to mobile video distribution over 3G or 4G networks have been explored, and there have been many experiments and measurements on big datasets to provide access for all kinds of diffusion platforms such as HTTP based websites, PC-Client software and mobile platforms. In [15], the mobile platform includes some applications for different devices and the behaviors of the users who are using IPhones and Ipads with IOS and Aphone and Apad with Android over 3G and Wi-Fi networks have been compared.

There are two types of users, standalone PC clients and mobile application users. Due to battery consumption, bandwidth limitation and probably the low quality content, most users are not downloading more than 50% of video and this issue can be considered as a challenge in designing a mobile P2P video System.

These missing parts of downloaded videos lead to a limited number of copy of end parts of video file available in the network. The P2P performance depends on the number of resource copies in the network, and this issue can be an obstacle for usability of a P2P system. Also, in [15] it was observed that the viewing time is longer for Wi-Fi users compared to 3G users, and most Wi-Fi users watch TV series and animations, while 3G users prefer short videos due to their energy and traffic consumption. They usually do not finish watching a full video. Due to the those reasons, Wi-Fi has more stable users that is essential for P2P networks. The viewing time in 3G and Wi-Fi networks on mobile devices is higher on tablet devices because of their bigger screen and processing capabilities. However, there is a big challenge for P2P transfer mode, which mobile devices have, because of their mobility, P2P connections are difficult to maintain. Therefore, P2P for mobile clients hits battery and traffic constraints, which brings strong limitations for developing P2P-Based mobile video distribution systems over 3G networks.

Although, recently some of the aforementioned problems have been solved, the accumulation of the above challenges seems to be serious as any of the above issues alone might block the deployment of P2P based mobile video distribution over 3G network.

After VoD systems became popular, there were some issues that still needed more attention. Due to lack of bandwidth and costly maintenance of these systems, especially for HD video content, the use of Peer-to-Peer (P2P) networking has been proposed. In order to improve the scalability of this system, some problems have been addressed like firewall and free loaders to increase the efficiency of P2P systems. Moreover, the ISPs may want to interfere with P2P traffic since this imposed traffic load on ISPs by P2P networks, and security issues can increase the overload, response time and startup delay.

To address the aforementioned problems, which are the fundamental problems of these systems, a new method has been presented to evaluate the streaming P2P protocol (SPP) architecture. [16] This method leads to a simple and comprehensive solution that addresses all the issues listed above. Additionally, the method has been applied in both simulation and experiments on PlanetLab for evaluation, and the results showed that beside scalability, the system could be implemented efficiently by combining cache nodes and using end user resource found in SPP architecture which leads to low load on servers and ISPs even when the firewalls are on. Low startup delays and a small number of playback errors during the PlanetLab experiments have been observed. PlanetLab is a global research network that supports the development of new network services. Since the beginning of 2003, PlanetLab has been used to develop new technologies for distributed storage, network mapping, Peer-to-Peer systems, distributed hash tables, and query processing by more than 1000 top researchers.

There are four problems, which have been addressed in SPP architecture to increase scalability and efficiency of the network. Firstly, malicious users that try to modify the video content. Secondly, traffic patterns that puts huge cost of distribution data on ISPs. Thirdly, limited connectivity because of firewalls. Finally, free loaders reduce the efficiency of system by not sharing their resources. These four problems can exist in large-scale commercial distribution models and small-scale community networks with limited resources. The contributions of this article are identification of these four issues for P2P based VoD streaming and the evaluation of the SPP architecture. By failing to address all these four problems, issues such as excessive startup delay or frequent playback errors can appear which can make streaming infeasible.

In [17], advantages and disadvantages of using multiple access networks have been explored simultaneously. While these days mobile devices are equipped with multiple network interfaces and they have access to more than one network at the same time, streams are often using just one available Internet connection. By using HTTP's capability of loading requests for specific byte ranges of a file, the implementation of an insignificant on-demand streaming service has been invented in application-layer. The method requires no change to current servers and infrastructure. Moreover, the experiments with multihomed host has been employed and their results show that the potential performance of VoD play out and achieved bandwidth aggregation efficiency is 90%, by downloading from 3 heterogeneous access networks in parallel. Multihomed host means a host, which has several links in order to download fixed-size segments of a file from server simultaneously. In Figure 2.4, an example of multihomed host can be seen.

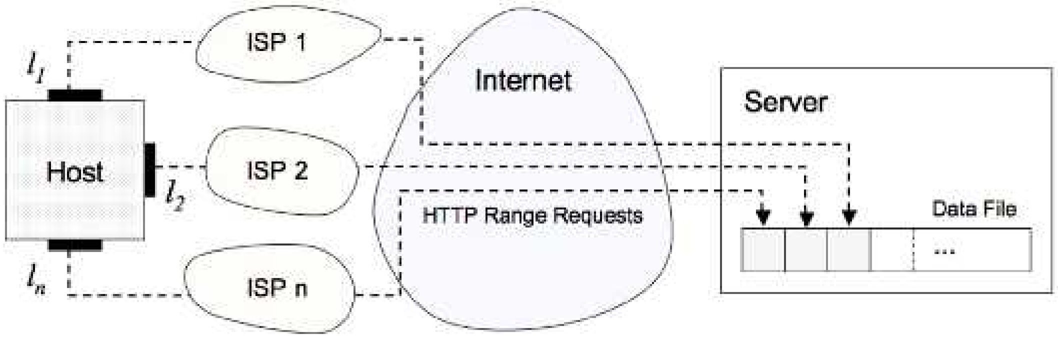


Figure 2.4: General scenario of a multihomed host [17]

In [18], the authors proposed a new technique in a wireless mesh network called dynamic stream merging (DSM). This technique works in the new version of multicast, which is fundamentally different from traditional multicast technique, relying on centralized control on the server side. This kind of multicast is accomplished without the knowledge of the server. In DSM, multicast topology is created through dynamic merging of server's stream at the nodes.

In [18], the authors mostly focused on VoD services over wireless mesh networks (WMN's), in which users, by having access to mesh-like wireless backbone, are capable to have access to VoD services in Internet or LAN through a mesh gateway, which is created by wireless mesh access routers. Their aim is to find a solution to minimize the traffic in mesh network in order to achieve a scalable solution to design a more robust environment, noting the costly dedicated server stream for each video service request and its limited scalability when system faces huge demand for a content by a large number of users. It is mentioned that the multicast technique has to be used in order to allow many users to share a popular server stream unlike traditional multicast, which relies on centralized control logic on server.

DSM is a distributed control technique for making a robust WMN for on-demand wireless access to videos on the Internet; this technique forms the multicast groups incrementally and performs routing in the mesh network adaptively in a distributed manner without involving the VoD server, while the nodes merge two identical streams together to conserve nodes' resources.

Since servers are not attached to the mesh network and it does not have direct access to the underlying multicast routing protocol, DSM strategy has to be used. The simulation results the study in [18] proved that DSM has the potential to provide a robust and cost effective technology to extend the use of VoD for mobile users.

The author in [19] made a review on issues related to topological design and related performance issues over geographically dispersed networks, which involve a network consisting of a set of servers and clients. He proposed a distributed multimedia content delivery system, which has focused on scalable streaming to a large number of distributed clients. The idea is based on the work of Shahabi, who proposed redundant hierarchy (RED-HI) for content management systems of distributed video servers. In this work, the integration of RED-HI and load balancing techniques along with the fault tolerant nature of RED-Hi has been employed. Then ways of, using this technique together with the proposed threshold level mechanism of popularity has been illustrated, which leads to achieve load balancing and efficiency as well as dynamic placement of media content. The content replication, caching and scalable delivery protocols for maintaining and achieving quality in terms of reliability and scalability of multimedia streaming files provides an object placement, location and content delivery architecture that is reliable and fault tolerant.

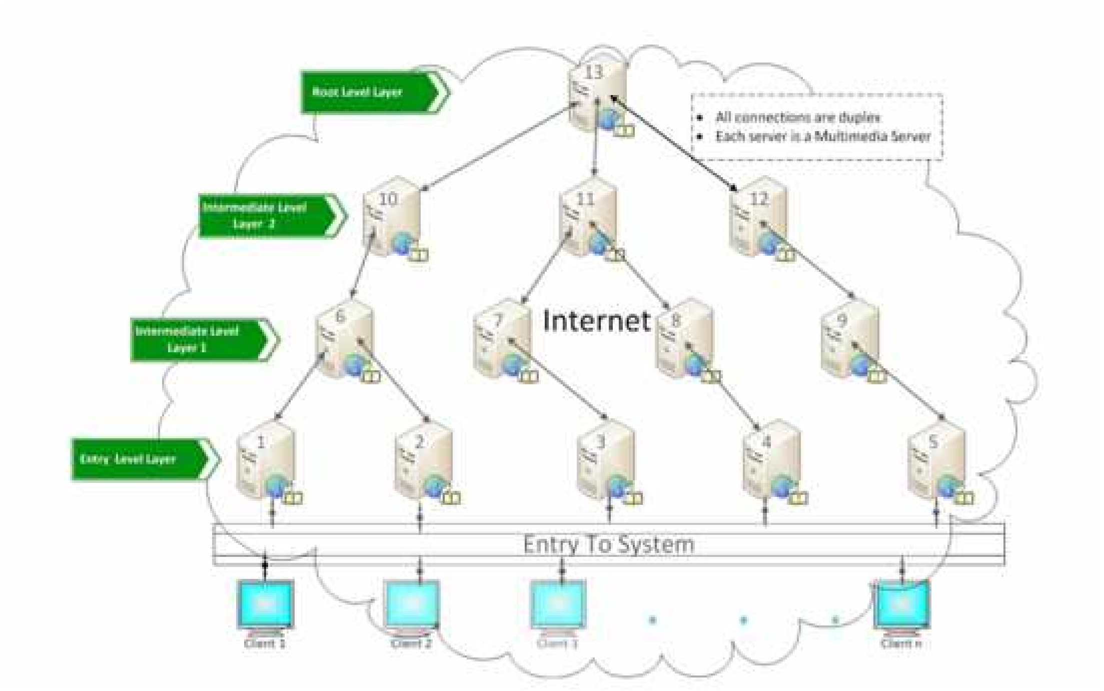

Root Level Layer
13
All connections are duplex
Each server is a Multimedia Server
Intermediate Level Layer 2
10
11
12
Intermediate Level Layer 1
6
7
Internet
8
9
Entry Level Layer
1
2
3
4
5
Entry To System
Client 1
Client 2
Client 3
Client n

Figure 2.5: Network architecture of Non-REDHI [19]

Figure 2.5 and 2.6 are showing a Network architecture with and without using REDHI model.

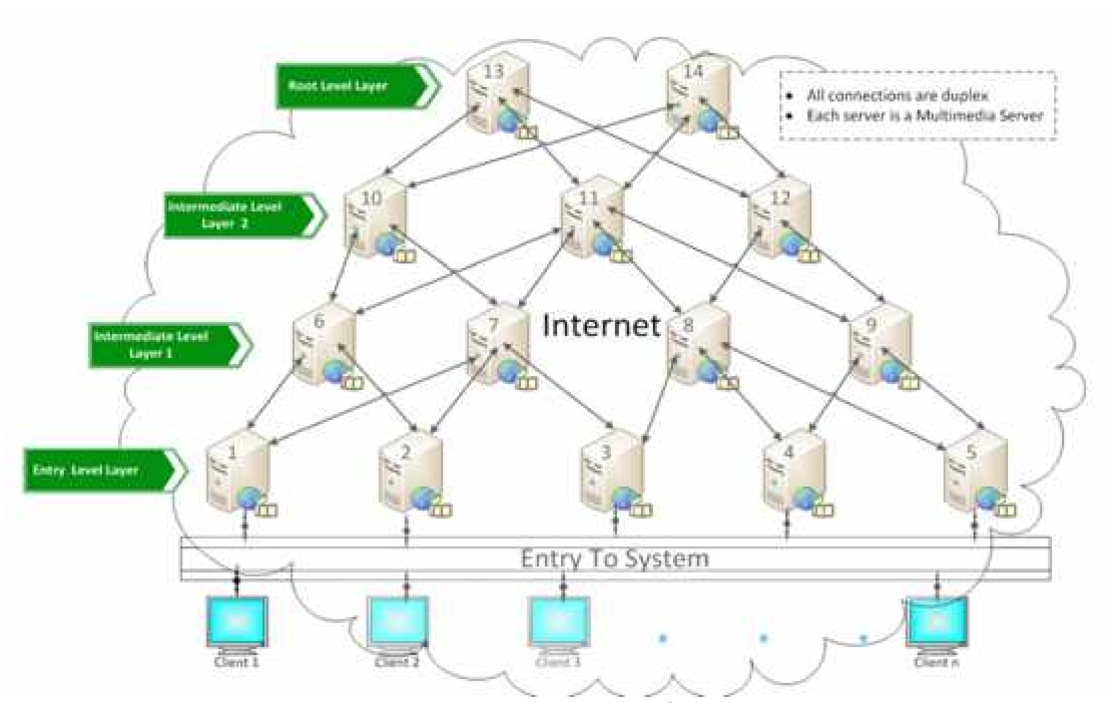

Root Level Layer
Intermediate Level Layer 2
Intermediate Level Layer 1
Entry Level Layer
All connections are duplex
Each server is a Multimedia Server
Internet
13
14
10
11
12
6
7
8
9
1
2
3
4
5
Entry To System
Client 1
Client 2
Client 3
Client n

Figure 2.6: Network architecture of the REDHI model [19]

The main goal for this study is to develop a Distributed Continuous Media Servers (DCMS) system that is able to deliver multimedia data over a network, efficiently. DCMS is designed with hierarchical fashion, so that the clients are distributed geographically dispersed.

In a hierarchical arrangement, each Centralized Continuous Multimedia Server (CCMS) acts as a node of the hierarchy, which resembles a tree. The upper end of the hierarchy is the root and the lower end has the leaves, which are called entry nodes. The entry nodes are responsible of taking the requests from the clients, which are connected normally by broadband connection, will remain for these entry nodes. After that, they will look for the requested media data to see if they have it in their local storage. If not, they will forward the request to their parents in their immediate higher level and finally if no one has the data, they may even reach to the root.

Hence, object placement, object location and delivery are at the center of the resource management component of a DCMS, the object placement task should have the minimum cost from system's perspective. Object location and delivery deal with finding the location where the multimedia object is placed, and then allocating a delivery route together with all resources needed by the system. The efficiency will be achieved by this component of DCMS since locating and delivering tasks are accomplished with minimum system resources being utilized by this component.

Moreover, it will keep all the resources occupied with different streams that are serving different clients to increase the utilization. In order to achieve the highest efficiency, the RED-HI has been introduced to decrease the restriction of having one parent per node for pure hierarchy and allow each node to have two or more parents. By having more than one parent, the possibility of having load balancing in any given path of the topology increases, and it makes the system more fault tolerant and efficient in terms of resource consumption and cost. Although there are more links to each node, since the bandwidth and the load are shared between them, the bandwidth for each node will be the same as pure hierarchy. Therefore, in RED-HI, there is no need to increase the bandwidth, and the possibility for higher connectivity at the negligible cost of an added link increases. Also each CCMS is considered as a single node, where each would have its own storage and bandwidth resources in RED-HI.

General view of REDHI scheme can be seen in figure 2.7.

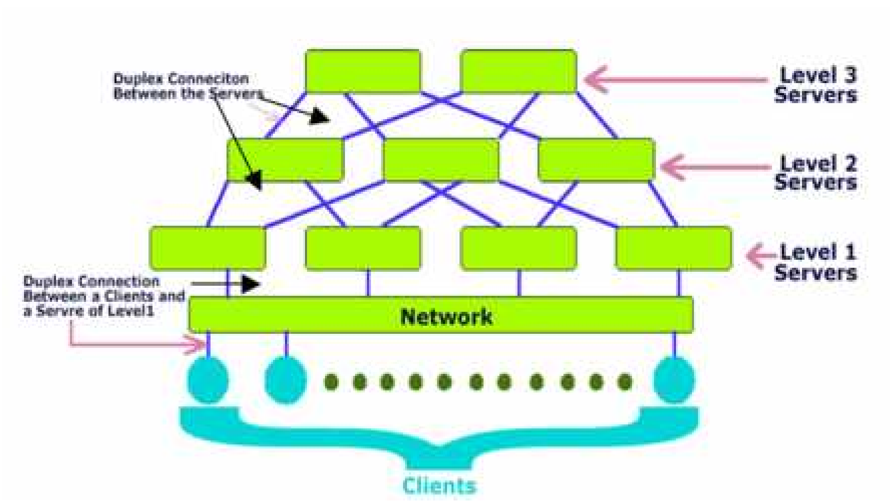


Figure 2.7: General structure of the REDHI system [19]

In RED-HI, whenever a client initiates a request, it is received by a head-end node in the system. The head-end node is a CCMS with agents associated to it and these agents create a query message to be propagated into the DCMS network. Each query message will go through different paths and analyze the costs incurred over the path traversed, incrementally. Eventually, some nodes will be traversed that hold the requested media objects in their local storage. These nodes will then create and transmit positive response messages back to the head-end nodes using the same path that was used by the query message to reach them. The acknowledgement message would entail the resources and their load.

The load and cost information is readily available at the agents of each CCMS. These agents use the information included in the positive response message to determine the best path from their node to the source node. This path would then be allocated and reserved by the head-end node for the streaming of the media object from the source to the client requesting the object from the DCMS system. Once the best node is selected and reserved, the agents would be periodically used to maintain a list of alternate routes to other replicas of the streamed object in the system.

In [19], simulations have illustrated the task of locating an object and retrieving it in a distributed system. It can be performed in a fault tolerant manner using RED-HI by eliminating the possibility of bottlenecks in the system as compared to that for a pure hierarchy. Based on the simulations, the system is highly scalable, resiliently reliable and fault tolerant. By better allocating the resources available, which has been used in DCMS systems, a superior delivery mechanism has been provided.

2.6 Classification of VoD systems

The following classification is developed after studying the literature on Video-on-Demand systems.

Table 2.1: Classification of Aforementioned Approaches

Technique	**Media**	**Transfer protocol**	**Reference**	**Admission control**	**Buffering for later use**	**Dual-purpose clients**	**Buffering**
LCBBS	LAN	P2P	[4] [5]	X	x	X	X
Patchpeer	WWAN+WLAN	P2P	[7]	-	-	X	X
Network-aware	LAN	P2P	[8]	-	-	X	X
Zhang's Model	LAN	P2P	[9]	-	-	X	-
He's Model	LAN	P2P with helpers	[10]	-	-	-	-
Lu's Model	LAN	P2P	[11]	-	-	-	-
NCECD	LAN	P2P	[12]	-	-	-	-
Content-aware architecture	WLAN	-	[13]	-	-	-	-
Adaptive channel control Scheme	WWAN	-	[14]	-	-	-	-
P2PTV	WWAN+WLAN	P2P	[15]	-	X	-	X
SPP architecture	LAN	-	[16]	-	-	-	X
Kasper's model	WWAN	Http	[17]	-	-	-	-
DSM	Wireless (WMN)	-	[18]	-	-		-
REDHI	LAN	-	[19]	-	-	X	X

Some of these methods have been developed for LAN environment [19] [16] [12] [11] [10] [9] [8], while others [18] [13] are in WLAN and [15] [7] are for the combination of WWAN and WLAN. Most of these approaches are using the P2P technique for communication between nodes. Obviously, by combining WWAN and WLAN, a larger number of users in more distributed area can be targeted, and this approach will probably perform better in P2P environment. However as mentioned in this chapter, there are still some open issues in integration of these two types of networks.

Chapter 3

DISCUSSION OF SIMULATED VOD SYSTEMS

In this chapter, the three VoD systems LCBBS, Patchpeer and Network-Aware approaches, which will be compared by simulation, will be discussed in more detail.

3.1 LCBBS

In order to improve the performance of Peer-to-Peer Video on demand system, LCBBS can be installed at the client side. The basic idea of LCBBS is that video data should be divided into segments and stored on peers and servers on the Internet. Whenever a peer wants to request for a data from the servers, it first checks its own buffer to see if there are any desired data segments. If yes, it will not look for servers on the Internet to get the data. If not, it asks from other peers. If peers have the requested video segments, using transmission procedures, data will be received from the local peers. In case these two attempt (own buffer, peers) fail, the request will be referred by LCBBS module to remote servers on the Internet.

The results of LCBBS module simulation [4] [5] have indicated that the performance of the VoD system with LCBBS is enhanced, with particular improvement in startup latency.

A general view of LCBBS model can be seen in figure 3.1.

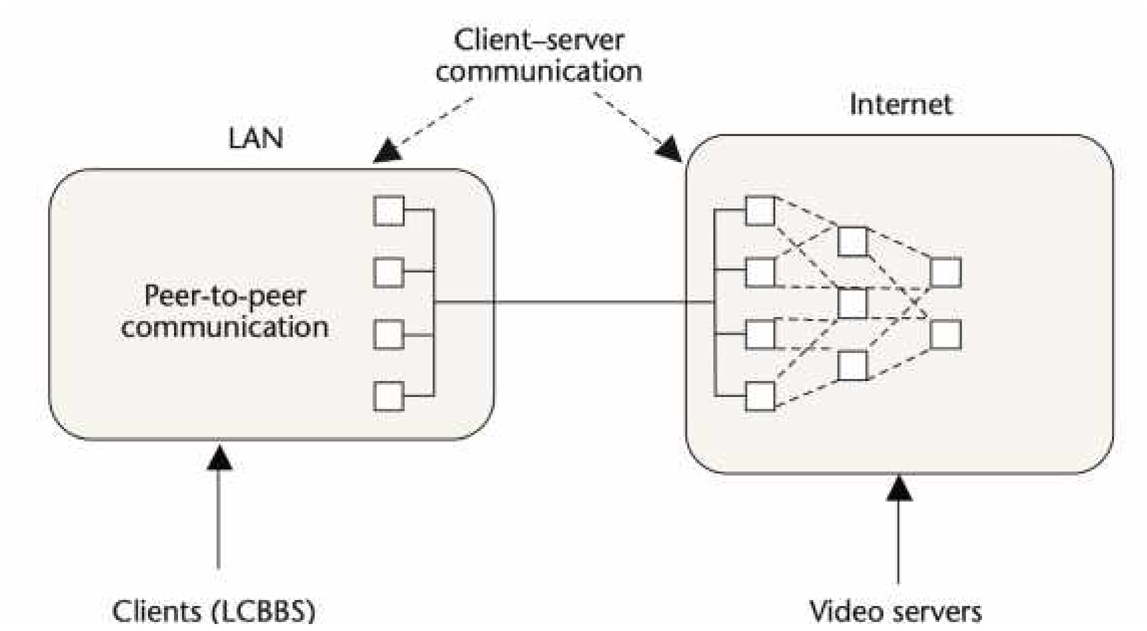


Figure 3.1:General view for LCBBS Paradigm [4]

As demonstrated on Figure 8, LCBBS should be installed as a module on the client side to run the following tasks. Firstly, it should use "SEARCH FUNCTION" to see if the requested data is available on local machine. If not, it will search on LAN on other peers' buffers, and finally in case of not finding the data locally, it will ask data from remote servers by using "TRANSFER FUNCTION". Secondly, it has the responsibility to manage caching the streamed video data, for later use of other peers instead of discarding it.

Illustration of LCBBS LAN client module has been given in figure 3.2.

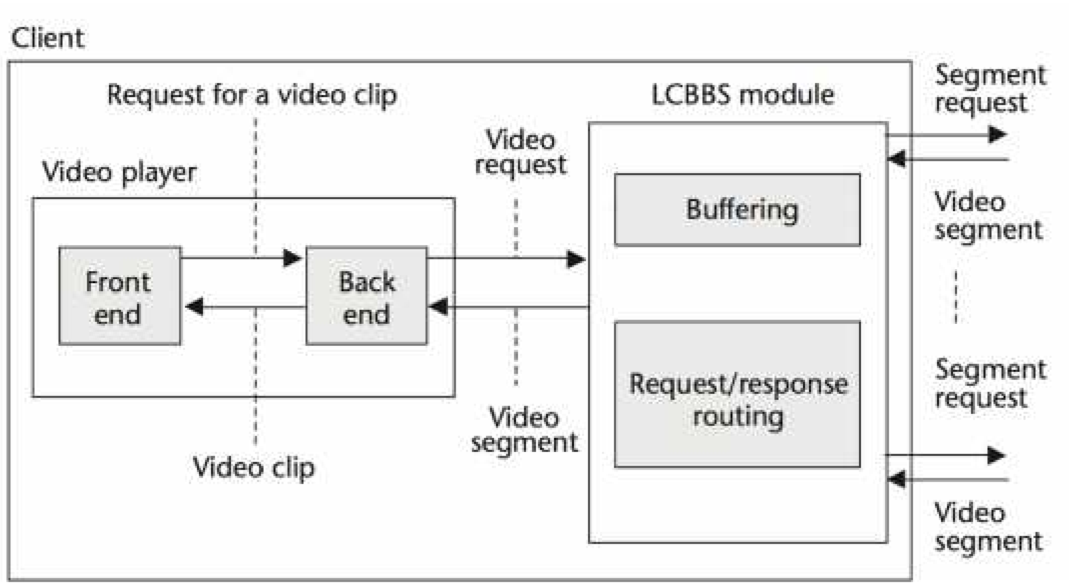


Figure 3.2:LAN Client with LCBBS Module [4]

Most of the time, caching the entire video data needs an extensive volume of auxiliary memory. The LCBBS module divides video data to various segments with a pre-settled size and stores some heading segments of a video in the client buffer.

The LCBBS module works on the back–end of the video player, and it is in charge of directing the video segments.

3.2 Patchpeer

In order to achieve the goal of having access to Video-on-Demand service anywhere and anytime, the features of carrier mobile networks and Peer-to-Peer systems have to be combined. In such a hybrid environment, high availability assurance and opportunistic use of resources will be available for mobile clients. Authors in [7] proposed a technique called Patchpeer to allow the Video-on-demand system scale beyond the bandwidth capacity of the server by using mobile clients not just as passive receivers, but also as active senders of video streams to other mobile clients.

Patching is a technique that utilizes the multicast service at the network layer, to enable true on-demand service on the Internet, for videos with diverse popularity. By patching, users should be capable of joining a multicast session to download the missed portion of a video over a dedicated patching stream. This technique should be implemented on the media server. In the proposed technique, each mobile device has two interfaces, which enable the device to communicate both in the WWAN and WLAN modes. Through the WLAN mode, mobile devices are capable of communicating with the Peer-to-Peer network and by the WWAN mode; they can connect to the base station. However, there are two challenges here in using this hybrid network, the first one is the integration of WWAN and WLAN, and the second one is the implementation of VoD service on this Hybrid network.

The environment for application of Patchpeer can be seen in figure 3.3.

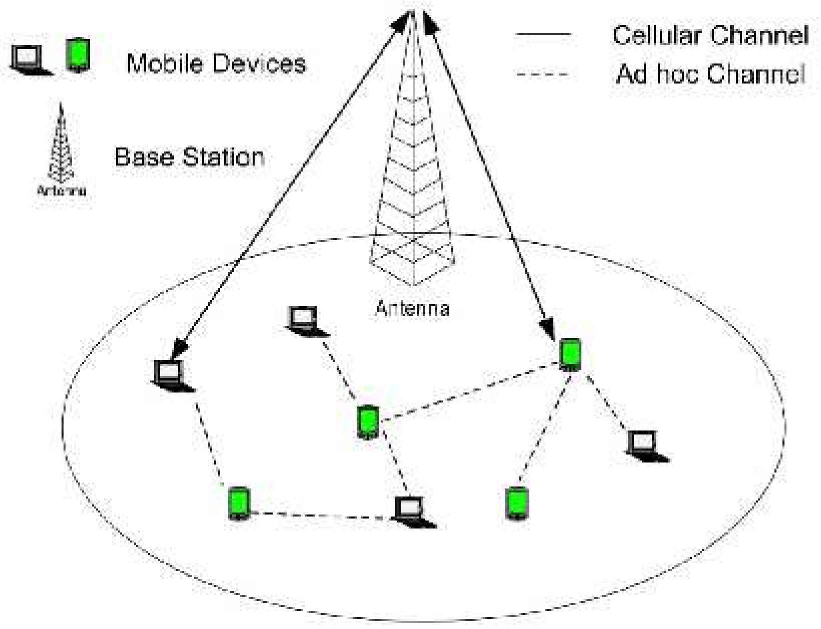


Figure 3.3: The desired environment for application of Patchpeer [7]

Figure 3.4 shows the collaboration diagram of Patchpeer scheme.

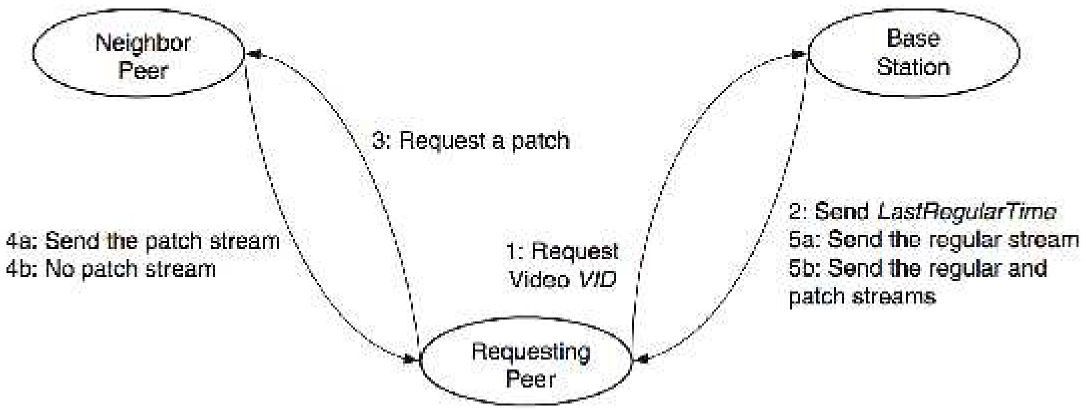

Neighbor Peer
Base Station
3: Request a patch
2: Send *LastRegularTime*
5a: Send the regular stream
5b: Send the regular and patch streams
4a: Send the patch stream
4b: No patch stream
1: Request Video *VID*
Requesting Peer

Figure 3.4 :Collaboration state diagram of Patchpeer method [7]

In this system, requesting peer has three main operations. Firstly, the peer finds out the start time of the last regular multicast of the requested video. Secondly, the requesting peer finds a patching stream provider from its 1-hop neighbors. Thirdly, the peer downloads and plays the video. The main task of the server is to decide whether a patching stream or a regular stream is needed for a video request.

Here the definition of regular and patching streams are as follows:

Regular stream: comes from an existing multicast for remainder of video

Patching stream: comes from neighbor node for downloading the missed portion of video.

Each peer in Patchpeer caches the prefix of the video it is watching in a forwarding buffer (fb). Therefore, early peers can use the video content in their fb to provide the patching stream to late peers in their neighborhood.

A patching stream provider has to satisfy two conditions: firstly, Data Condition: the fb of stream provider must hold enough data of the requested video. Secondly, Network Condition: there must be enough bandwidth in the WLAN to support the new data flow without compromising the QoS of existing data flows.

Figure 3.5 shows the flowchart related to requesting peer and server in Patchpeer method.

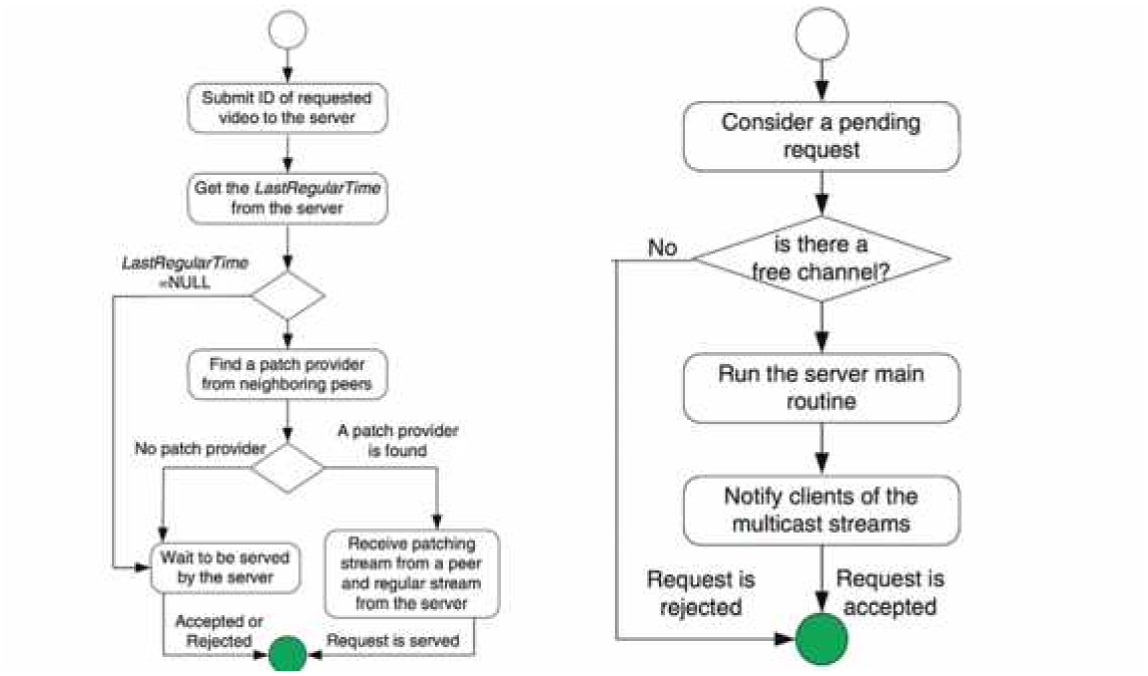

Submit ID of requested video to the server
Get the LastRegularTime from the server
LastRegularTime =NULL
Find a patch provider from neighboring peers
No patch provider
A patch provider is found
Wait to be served by the server
Receive patching stream from a peer and regular stream from the server
Accepted or Rejected
Request is served
Consider a pending request
No
is there a free channel?
Run the server main routine
Notify clients of the multicast streams
Request is rejected
Request is accepted

Figure 3.5: Flowchart diagram for Requesting peer (left) and Server (Right) in Patchpeer method [7]

To select a patching stream provider from the set of candidate peers, the requesting client can use one of the following heuristic selection methods.

a) Random Selection: it randomly selects the patching stream provider in the set of candidate peers.

b) Earliest Response: it selects the neighboring peer who responds to the RequestPatch message at the earliest time. The advantage of this selection method is it does not have to wait for the timeout period to expire to start receiving the patching stream.

c) Closest Peer: assuming mobile nodes have knowledge of their locations, i.e., mobile nodes are equipped with positioning devices such as Global Positioning System (GPS). In this selection method, Client selects the candidate that is closest to the destination node. The idea is the close distance, keeps two mobile nodes within the transmission range longer.

d) Most Available Bandwidth: it selects the patching stream provider with the most available bandwidth among the set of candidate peers. When other communicating nodes move into the communication range of the patching stream provider, the data rate of the patching stream may be affected.

3.3 Network-Aware

In [8], the authors proposed a Network-aware multicast scheme to supply instantaneous VoD services on LAN. This system is the first one proposed among the three P2P VoD approaches we are comparing. It is implemented based on a Peer-to-Peer Grid, which includes some dedicated video servers, many non-dedicated mini-video servers that are installed as software on clients, and some client with low power that can be in the only-viewer node.

By taking advantage of the large storage and the powerful processing capability in client-side devices, the user host serves as both a client and a non-dedicated video server. At the time that the clients are acting as non-dedicated mini video server, they can replicate the entire video file. However, when they are in the only-viewer mode they do not help in increasing the system capability.

If two nodes can provide video services to each other, they will have similar interest. Peer groups are formed as a collection of peers that share similar interest, and workload may be balanced among autonomous groups. Video index information helps nodes to be aware of the interest of their peers. The level of similar interest is assessed as per the video title classes; the quantity of similar titles, and the QoS gained by their peers. Nodes with high level of similar interest are considered as good peers, which forms an autonomous group.

Subsequently, clients may spot their resources with a high probability based on local data. Peers with similar interest are forming a virtual subnet. First, a request for a particular video title is processed in this subnet. In case that this attempt fails, its peers will propagate this request to their own peers and so on. All messages for this request are labeled with timestamps. At the point when there is a timeout, this request will be sent directly to dedicated servers.

The delivery of video media is influenced by network conditions. In a controllable fast network, the QoS can be ensured with a resource-reservation technique. Due to the decentralization process, the resource-reservation strategy for the overall VoD system is complicated, and the bandwidth available to an individual connection can hardly be predicted and controlled determinately. An individual channel is allocated to transmit the whole video in order to fulfill the first demand for this video title. When another new demand for the same video title is admitted after t seconds, whatever remains of the video data can be multicast through the sharing channel.

There are at least two video streams for the new on-demand client. One stream is for the multicasting portion of the video data, and the other stream is for the first t seconds of the video object through an alternate channel allocated at this point. These video streams are merged for playback in client-side storage. In contrast with traditional batching schemes, this multicast policy can supply instantaneous VoD services without postponing the new on-demand requests.

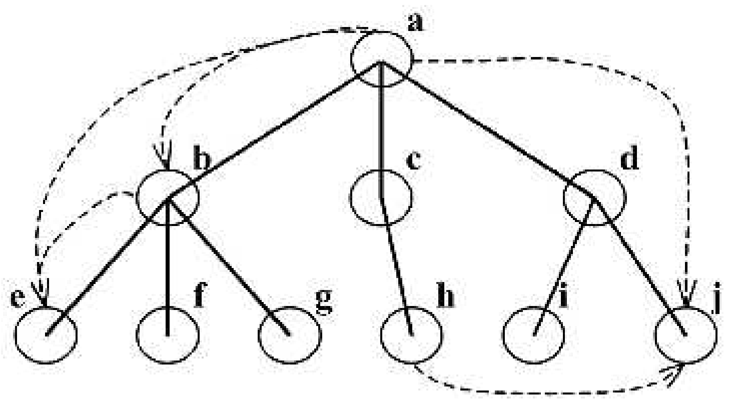


Figure 3.6: Network topology-aware multicast example [8]

Some nodes have the ability to provide service for other nodes when a video stream is cached in a user host during playback time. In Figure 3-6, by using node-link graph representation, network topology-aware multicast has been illustrated. In this case, each node acts as a user host (dedicated server), and each link represents a connection between two nodes.

The non-dedicated video server may satisfy local requirements as soon as possible as the extension of the dedicated video server. When a new on-demand request comes, some peers will be selected to provide VoD services. If this new request is for the same video title processed by these peers, it will join in the multicast delivery for this video title. Otherwise, one video server starts an individual video delivery for its first presentation. This network topology-aware multicast policy is mainly guided by the cooperation among multiple video servers. It is both network-aware and stream-aware.

This policy can also be applied to load balancing and recovery from failure. A substitute video server must be put in use in case of overloading, congestion or component failures.

The GISMO toolset was used for the network-aware approach to generate a workload for driving the simulations. Generator of Internet Streaming Media Objects and workloads (GISMO) enables the specification of a number of streaming media access characteristics, including object popularity , temporal correlation of request, seasonal access patterns, user session durations, user interactivity times and variable bit-rate [20] .

Video data are encoded by the MPEG standard. A Poisson process generates request arrivals. The length of each video title is fixed at 30 minutes. It is assumed that users may cancel their orders if there is a long waiting time.

In this analysis, defection rate and average delay time are chosen as the performance metrics. Defection rate is the percentage of service requests canceled by users. Obviously, minimizing the defection rate maximizes the system throughput, and reducing the average delay time improves the system quality.

For comparison, a FCFS (first come first served) batching VoD scheme was also simulated. A batching scheme delays requests and hopes that more requests for the same video will arrive during the batching interval. In each batch, all the requests for the same video are served in one multicast.

A scenario in which 9 clients run as non-dedicated video servers was firstly considered. The network-aware multicast scheme achieves significantly greater system performance than the FCFS batching scheme. Simulations were again performed for cases where the arrival rate is fixed. Once enough non-dedicated video servers have been added, it is the index and cooperation algorithms that mainly affect the defection rates in this P2P Grid system, but not the system size. The average delay time achieved by this network-aware multicast scheme is less than that achieved by the FCFS batching scheme. When the system scales up to a certain size, the average delay time may slightly increase due to the cost for index and adaptation. In the network-aware system, a Peer-to-Peer grid environment is assumed so the results are valid for only this environment. It is also assumed that the videos are stored as a whole in the client buffers.

Chapter 4

SIMULATION

4.1 Simulation Setup and Performance Metrics

In this chapter we carry out a performance evaluation of the three aforementioned techniques (LCBBS, Patchpeer and Network-aware) using the NS simulator. NS is an object oriented and discrete event driven simulator. The NS software is generally used for simulation of local area networks and wide area networks. NS can be implemented on different operating systems, but in this study, we used the Fedora12 Linux operating system, which is being employed for a number of programming tools with simulation process to run the simulation of NS-all-in-one version 2.35. For simulation and implementation, firstly the network scenario should be designed in OTCL language. This gives us the output as a trace file.

The purpose of the simulation study is to investigate the behavior of the three mentioned techniques (LCBBS, Patchpeer and Network-aware) by using three performance parameters against increasing buffer size.

Table 4.1, 4.2 and 4.3 show the simulation parameters used for evaluating LCBBS, Patchpeer and Network-aware methods.

Table 4.1: Parameters and their values used for LCBBS simulation [4] [5]

Parameters	**Value (s)**
Time to generate a request for a video clip by a client	1200000 ms (20 minutes)
Size of a video segment	128 Kbytes
Size of a video clip	(8200) Segments
Size of a video packet	1500 bytes
Packet transfer time in the internet	100 ms (120 kbps)
Packet transfer time in local area network	(0.5, 2.5 ms)(10 mbps)
Time to prepare a packet by a client or a server s	5 ms
Time to read a video segment from disk by a client or a server	25 ms
Display rate for video clips	350 kbps
LAN bandwidth	10 Mbps
Number of nodes	30

Table 4.2: Parameters and their values used for Patchpeer simulation [7]

Parameter	**Default**	**Variation**
Simulation time (min)	120	N/A
Mean request inter-arrival time (s)	20	10-100
Number of mobile nodes	400	N/A
Transmission range (m)	250	N/A
WLAN bandwidth (kbps)	6000	N/A
Client forwarding / Playback buffer size (min)	12	0-60
WWAN bandwidth (kbps)	2400	N/A
Percentage of seed peers	10	0-20
Number of videos	1	N/A
Normal playback rate (kbps)	300	N/A
Video length (min)	60	10-60
Mean speed (mps)	3	1-10
Speed delta (mps)	0.3	0.1-1
Mean pause time (s)	5	N/A
Pause time delta (s)	1	N/A
Operating area (m×m)	1,000×1,000	$1{,}000^2 - 5{,}000^2$

Table 4.3: Parameters and their values used for Network-Aware simulation [8]

Parameter	**Default value**
Workload for a single simulation run	30000 requests
Number of video titles	300
Length of each video title	30 min
Cache size in user host	5 GB
Max time for wait	3 min
LAN bandwidth	10 Mbps
Number of nodes	30

By comparing these three tables, it can be seen that table 4.2 shows the parameters used for Patchpeer simulation, which is considering a wireless network, while table 4.1 and 4.3 show parameters for wired networks. Considering the difference between these two types of infrastructure, there seems to be no common points for comparison between them. However, in Table 4.1 and 4.3, there is one common parameter, which is the bandwidth that is 10 Mbps. Also for the simulation of all these three techniques, IEEE 802.11 issued as Mac layer protocol and CBR for defining the type of traffic. AODV, which is a routing protocol, where the route between the source and the destination node is considered on an on-demand basis in NS software. DSR is also considered as a routing protocol in LCBBS also DROPTAIL have been used for defining the procedure for removing less used data from the queue additionally MyEvalvid toolset have been used for video transmission and quality evaluation.

The LCBBS algorithm is given as follow:

```
set ns [new Simulator -multicast on]
set group0 [Node allocaddr]
set group1 [Node allocaddr]
set group2 [Node allocaddr]

set f [open out.tr w]
$ns trace-all $f
#$ns use-newtrace
proc finish {} {
     global ns f dst2_f0 dst2_f1 dst2_f2 dst3_f0 dst3_f1 dst4_f0
     $ns flush-trace
     $dst2_f0 closefile
     $dst2_f1 closefile
     $dst2_f2 closefile
     $dst3_f0 closefile
```

```
    $dst3_f1 closefile
    $dst4_f0 closefile
    puts "Simulation completed."
    close $f
    exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n1 10Mb 1ms DropTail
$ns duplex-link $n1 $n2 10Mb 1ms DropTail
$ns duplex-link $n1 $n3 10Mb 1ms DropTail

set max_fragmented_size 1500
#IP header: 20 bytes, UDP header: 8 bytes, Layer-5 header: 12 bytes
set packetSize [expr $max_fragmented_size+40]


set src_udp1 [new Agent/my_UDP]
$src_udp1 set dst_addr_ $group0
$src_udp1 set dst_port_ 100
$src_udp1 set packetSize_ $packetSize
$src_udp1 set rate_ 32Mb
$ns attach-agent $n0 $src_udp1

set dst2_f0 [new Agent/myEvalSVC_Sink]
$dst2_f0 set_filename rd_n2_f0
$dst2_f0 wired
$dst2_f0 set rate_ 32Mb
$n2 attach $dst2_f0 100

set dst3_f0 [new Agent/myEvalSVC_Sink]
$dst3_f0 set_filename rd_n3_f0
$dst3_f0 wired
$dst2_f0 set rate_ 32Mb
$n3 attach $dst3_f0 100

set dst4_f0 [new Agent/myEvalSVC_Sink]
$dst4_f0 set_filename rd_n4_f0
$dst4_f0 wired
$dst2_f0 set rate_ 32Mb
$n4 attach $dst4_f0 100

set original_file_name ns2send_tid0
set trace_file_name video1.dat
set original_file_id [open $original_file_name r]
set trace_file_id [open $trace_file_name w]
set pre_time 0

while {[eof $original_file_id] == 0} {
  gets $original_file_id current_line
  scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
  set time [expr int(($t_ - $pre_time)*8000000.0)]

  if { $tid_ == 0 } {
    set prio_p 1
  }

  if { $tid_ == 1 } {
    set prio_p 1
  }

  if { $tid_ == 2 } {
    set prio_p 1
  }

  puts  $trace_file_id "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
  set pre_time $t_
}

close $original_file_id
close $trace_file_id
```

```
set trace_file [new Tracefile]
$trace_file filename $trace_file_name

set video1 [new Application/Traffic/myEvalSVC]
$video1 attach-agent $src_udp1
$video1 attach-tracefile $trace_file
$video1 wired
$video1 set rate_ 32Mb

set src_udp2 [new Agent/UDP]
$src_udp2 set dst_addr_ $group1
$src_udp2 set dst_port_ 110
$src_udp2 set packetSize_ $packetSize
$src_udp2 set rate_ 32Mb
$ns attach-agent $n0 $src_udp2

set dst2_f1 [new Agent/myEvalSVC_Sink]
$dst2_f1 set_filename rd_n2_f1
$dst2_f1 wired
$dst2_f1 set rate_ 32Mb
$n2 attach $dst2_f0 110

set dst3_f1 [new Agent/myEvalSVC_Sink]
$dst3_f1 set_filename rd_n3_f1
$dst3_f1 wired
$dst3_f1 set rate_ 32Mb
$n3 attach $dst3_f1 110

set original_file_name2 ns2send_tid1
set trace_file_name2 video2.dat
set original_file_id2 [open $original_file_name2 r]
set trace_file_id2 [open $trace_file_name2 w]

set pre_time2 0
while {[eof $original_file_id2] == 0} {
   gets $original_file_id2 current_line
   scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
   set time [expr int(($t_ - $pre_time2)*8000000.0)]

   if { $tid_ == 0 } {
     set prio_p 1
   }

   if { $tid_ == 1 } {
     set prio_p 1
   }

   if { $tid_ == 2 } {
     set prio_p 1
   }

   puts  $trace_file_id2 "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
   set pre_time2 $t_
}

close $original_file_id2
close $trace_file_id2
set trace_file2 [new Tracefile]
$trace_file2 filename $trace_file_name2
set video2 [new Application/Traffic/myEvalSVC]
$video2 attach-agent $src_udp2
$video2 attach-tracefile $trace_file2
$video2 wired
$video2 set rate_ 32Mb

set src_udp3 [new Agent/UDP]
$src_udp3 set dst_addr_ $group2
$src_udp3 set dst_port_ 111
$src_udp3 set packetSize_ $packetSize
$src_udp3 set rate_ 32Mb
$ns attach-agent $n0 $src_udp3

set dst2_f2 [new Agent/myEvalSVC_Sink]
$dst2_f2 set_filename rd_n2_f2
$dst2_f2 wired
```

```
$n2 attach $dst2_f2 111

set original_file_name3 ns2send_tid2
set trace_file_name3 video3.dat
set original_file_id3 [open $original_file_name3 r]
set trace_file_id3 [open $trace_file_name3 w]

set pre_time3 0
while {[eof $original_file_id3] == 0} {
   gets $original_file_id3 current_line
   scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
   set time [expr int(($t_ - $pre_time3)*10000000.0)]

   if { $tid_ == 0 } {
     set prio_p 1
   }

   if { $tid_ == 1 } {
     set prio_p 1
   }

   if { $tid_ == 2 } {
     set prio_p 1
   }

   puts  $trace_file_id3 "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
   set pre_time3 $t_
}

close $original_file_id3
close $trace_file_id3
set trace_file3 [new Tracefile]
$trace_file3 filename $trace_file_name3
set video3 [new Application/Traffic/myEvalSVC]
$video3 attach-agent $src_udp3
$video3 attach-tracefile $trace_file3
$video3 wired

set mproto DM
set mrthandle [$ns mrtproto $mproto]

$ns at 6  "$n2 join-group  $dst2_f0 $group0"
$ns at 6  "$n3 join-group  $dst3_f0 $group0"
$ns at 6  "$n4 join-group  $dst4_f0 $group0"
$ns at 6  "$n2 join-group  $dst2_f1 $group1"
$ns at 6  "$n3 join-group  $dst3_f1 $group1"
$ns at 6  "$n5 join-group  $dst2_f2 $group2"
$ns at 6  "$n6 join-group  $dst3_f0 $group2"
$ns at 6  "$n7 join-group  $dst2_f2 $group2"

$ns at 1.0  "$video1 start"
$ns at 2500.0 "$video1 stop"
$ns at 1.0  "$video2 start"
$ns at 4500.0 "$video2 stop"
$ns at 1.0  "$video3 start"
$ns at 7000.0 "$video3 stop"
$ns at 7200.0 "finish"

$ns run
```

The Patchpeer algorithm is also given as follow:

```
proc getopt {argc argv} {
         global opt
    lappend optlist nn
    for {set i 0} {$i < $argc} {incr i} {
                  set opt($i) [lindex $argv $i]
         }
}
```

```
getopt $argc $argv

#loss_model: 0 for uniform distribution, 1 for GE model
set loss_model  $opt(0)
set pGG $opt(1)
set pBB $opt(2)
set pG $opt(3)
set pB $opt(4)


#===================================
#    Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     400                        ;# number of mobilenodes
set val(rp)     DSR                        ;# routing protocol (DSR, AODV, BATMAN)
set val(x)      1000                       ;# X dimension of topography
set val(y)      1000                       ;# Y dimension of topography
set val(stop)   7200.0                     ;# time of simulation end
Mac/802_11 set dataRate_ 32Mb
Mac/802_11 set basicRate_ 32Mb

#===================================
#       Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile
#$ns use-newtrace
set chan [new $val(chan)];#Create wireless channel

#===================================
#    Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType       $val(ll) \
                -macType      $val(mac) \
                -ifqType      $val(ifq) \
                -ifqLen       $val(ifqlen) \
                -antType      $val(ant) \
                -propType     $val(prop) \
                -phyType      $val(netif) \
                -channel      $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   OFF \
                -macTrace      ON \
                -movementTrace OFF

#===================================
#       Nodes Definition
#===================================
#Create 3 nodes
set n0 [$ns node]
$n0 set X_ 200
$n0 set Y_ 400
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
```

```
set n1 [$ns node]
$n1 set X_ 300
$n1 set Y_ 400
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20

set n2 [$ns node]
$n2 set X_ 200
$n2 set Y_ 400
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20

set n3 [$ns node]
$n3 set X_ 200
$n3 set Y_ 400
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20



set n0if_ [$n0 set netif_(0)]
$n0if_ set-error-level $pGG $pBB $pG $pB $loss_model

set n1if_ [$n1 set netif_(0)]
$n1if_ set-error-level $pGG $pBB $pG $pB $loss_model

#==================================================
set max_fragmented_size 1024
#8 bytes:UDP header, 12 bytes: RTP header
set packetSize [expr $max_fragmented_size+20]

set src_udp1 [new Agent/my_UDP]
$src_udp1 set packetSize_ $packetSize
$src_udp1 set rate_ 2.34Mb
$src_udp1 set_filename wireless_sd
set dst_udp1 [new Agent/myEvalvid_Sink]
$ns attach-agent $n0 $src_udp1
$ns attach-agent $n1 $dst_udp1
$ns connect $src_udp1 $dst_udp1
$dst_udp1 set_filename wireless_rd

set original_file_name Verbose_StarWarsIV.dat
set trace_file_name video1.dat
set original_file_id [open $original_file_name r]
set trace_file_id [open $trace_file_name w]

set frame_count 0

while {[eof $original_file_id] == 0} {

   gets $original_file_id current_line
   scan $current_line "%d%s%d%d" seq_ frametype_ nexttime_ length_

   #1000/25 = 40ms = 40000 us
   set time [expr 1000*40]

   if { $frametype_ == "I" } {
     set type_v 1
   }

   if { $frametype_ == "P" } {
     set type_v 2
   }

   if { $frametype_ == "B" } {
     set type_v 3
   }

   puts $trace_file_id "$time $length_ $type_v $seq_ $max_fragmented_size"
   incr frame_count
}

close $original_file_id
close $trace_file_id
set end_sim_time [expr 1.0 * 40 * ($frame_count)  / 1000]
```

```
puts "$end_sim_time"

set trace_file [new Tracefile]
$trace_file filename $trace_file_name
set video1 [new Application/Traffic/myEvalvid]
$video1 attach-agent $src_udp1
$video1 attach-tracefile $trace_file

$ns at 0.5       "$video1 start"
$ns at $val(stop)  "$video1 stop"

#==================================
#      Termination
#==================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile src_udp1 dst_udp1
   $ns flush-trace
   close $tracefile
   $src_udp1 closefile
   $dst_udp1 closefile
   puts "simulation completed"
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   #$ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

The Network-aware algorithm is given as follow:

```
set ns [new Simulator -multicast on]
set group0 [Node allocaddr]
set group1 [Node allocaddr]
set group2 [Node allocaddr]
set group3 [Node allocaddr]
set group4 [Node allocaddr]
set group5 [Node allocaddr]

set f [open out.tr w]
$ns trace-all $f

proc finish {} {
     global ns f dst2_f0 dst2_f1 dst2_f2 dst3_f0 dst3_f1 dst4_f0
     $ns flush-trace
     $dst2_f0 closefile
     $dst2_f1 closefile
     $dst2_f2 closefile
     $dst3_f0 closefile
     $dst3_f1 closefile
     $dst4_f0 closefile
     puts "Simulation completed."
     close $f
     exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n1 10Mb 180s DropTail
$ns duplex-link $n1 $n2 10Mb 180s DropTail
$ns duplex-link $n1 $n3 10Mb 180s DropTail

set max_fragmented_size 4200
#IP header: 20 bytes, UDP header: 8 bytes, Layer-5 header: 12 bytes
```

```
set packetSize [expr $max_fragmented_size+40]


set src_udp1 [new Agent/UDP]
$src_udp1 set dst_addr_ $group4
$src_udp1 set dst_port_ 100
$src_udp1 set packetSize_ $packetSize
$src_udp1 set rate_ 0.1Mb
$ns attach-agent $n0 $src_udp1

set dst2_f0 [new Agent/myEvalSVC_Sink]
$dst2_f0 set_filename rd_n2_f0
$dst2_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n2 attach $dst2_f0 100

set dst3_f0 [new Agent/myEvalSVC_Sink]
$dst3_f0 set_filename rd_n3_f0
$dst3_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n3 attach $dst3_f0 100

set dst4_f0 [new Agent/myEvalSVC_Sink]
$dst4_f0 set_filename rd_n4_f0
$dst4_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n4 attach $dst4_f0 100

set src_udp2 [new Agent/UDP]
$src_udp2 set dst_addr_ $group5
$src_udp2 set dst_port_ 100
$src_udp2 set packetSize_ $packetSize
$src_udp2 set rate_ 0.1Mb
$ns attach-agent $n5 $src_udp2

set dst2_f0 [new Agent/myEvalSVC_Sink]
$dst2_f0 set_filename rd_n2_f0
$dst2_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n2 attach $dst2_f0 100

set dst3_f0 [new Agent/myEvalSVC_Sink]
$dst3_f0 set_filename rd_n3_f0
$dst3_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n3 attach $dst3_f0 100

set dst4_f0 [new Agent/myEvalSVC_Sink]
$dst4_f0 set_filename rd_n4_f0
$dst4_f0 wired
$dst2_f0 set rate_ 0.1Mb
$n4 attach $dst4_f0 100


set original_file_name ns2send_tid0
set trace_file_name video1.dat
set trace_file_name video4.dat
set trace_file_name video5.dat
set original_file_id [open $original_file_name r]
set trace_file_id [open $trace_file_name w]
set pre_time 0

while {[eof $original_file_id] == 0} {
   gets $original_file_id current_line
   scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
   set time [expr int(($t_ - $pre_time)*8000000.0)]

   if { $tid_ == 0 } {
     set prio_p 1
   }

   if { $tid_ == 1 } {
     set prio_p 1
   }
   if { $tid_ == 2 } {
     set prio_p 1
   }

   puts  $trace_file_id "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
   set pre_time $t_
}

close $original_file_id
close $trace_file_id
set trace_file [new Tracefile]
$trace_file filename $trace_file_name

set video1 [new Application/Traffic/myEvalSVC]
$video1 attach-agent $src_udp1
$video1 attach-tracefile $trace_file
$video1 wired
$video1 set rate_ 0.1Mb

set src_udp2 [new Agent/UDP]
$src_udp2 set dst_addr_ $group1
$src_udp2 set dst_port_ 110
$src_udp2 set packetSize_ $packetSize
$src_udp2 set rate_ 0.1Mb
$ns attach-agent $n0 $src_udp2

set dst2_f1 [new Agent/myEvalSVC_Sink]
$dst2_f1 set_filename rd_n2_f1
$dst2_f1 wired
$dst2_f1 set rate_ 0.1Mb
$n2 attach $dst2_f0 110

set dst3_f1 [new Agent/myEvalSVC_Sink]
$dst3_f1 set_filename rd_n3_f1
$dst3_f1 wired
$dst3_f1 set rate_ 0.1Mb
$n3 attach $dst3_f1 110

set original_file_name2 ns2send_tid1
set trace_file_name2 video2.dat
set trace_file_name video8.dat
set trace_file_name video9.dat
set trace_file_name video10.dat
set trace_file_name video11.dat
set trace_file_name video12.dat
set trace_file_name video13.dat
set original_file_id2 [open $original_file_name2 r]
set trace_file_id2 [open $trace_file_name2 w]

set pre_time2 0
while {[eof $original_file_id2] == 0} {
   gets $original_file_id2 current_line
   scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
   set time [expr int(($t_ - $pre_time2)*8000000.0)]

   if { $tid_ == 0 } {
     set prio_p 1
   }

   if { $tid_ == 1 } {
     set prio_p 1
   }

   if { $tid_ == 2 } {
     set prio_p 1
   }

   puts  $trace_file_id2 "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
   set pre_time2 $t_
}

close $original_file_id2
close $trace_file_id2
set trace_file2 [new Tracefile]
$trace_file2 filename $trace_file_name2
```

```
set video2 [new Application/Traffic/myEvalSVC]
$video2 attach-agent $src_udp2
$video2 attach-tracefile $trace_file2
$video2 wired
$video2 set rate_ 0.1Mb

set src_udp3 [new Agent/UDP]
$src_udp3 set dst_addr_ $group2
$src_udp3 set dst_port_ 111
$src_udp3 set packetSize_ $packetSize
$src_udp3 set rate_ 0.1Mb
$ns attach-agent $n0 $src_udp3

set dst2_f2 [new Agent/myEvalSVC_Sink]
$dst2_f2 set_filename rd_n2_f2
$dst2_f2 wired
$n2 attach $dst2_f2 111

set original_file_name3 ns2send_tid2
set trace_file_name3 video3.dat
set original_file_id3 [open $original_file_name3 r]
set trace_file_id3 [open $trace_file_name3 w]

set pre_time3 0
while {[eof $original_file_id3] == 0} {
   gets $original_file_id3 current_line
   scan $current_line "%f%d%d%d%d%d" t_ size_ lid_ tid_ qid_ fno_
   set time [expr int(($t_ - $pre_time3)*8000000.0)]

   if { $tid_ == 0 } {
     set prio_p 1
   }

   if { $tid_ == 1 } {
     set prio_p 1
   }

   if { $tid_ == 2 } {
     set prio_p 1
   }

   puts  $trace_file_id3 "$time $size_ $lid_ $tid_ $qid_ $fno_ $prio_p $max_fragmented_size"
   set pre_time3 $t_
}

close $original_file_id3
close $trace_file_id3
set trace_file3 [new Tracefile]
$trace_file3 filename $trace_file_name3
set video3 [new Application/Traffic/myEvalSVC]
$video3 attach-agent $src_udp3
$video3 attach-tracefile $trace_file3
$video3 wired

set mproto DM
set mrthandle [$ns mrtproto $mproto]

$ns at 0  "$n2 join-group  $dst2_f0 $group0"
$ns at 0  "$n3 join-group  $dst3_f0 $group0"
$ns at 0  "$n4 join-group  $dst4_f0 $group0"
$ns at 0  "$n2 join-group  $dst2_f1 $group1"
$ns at 0  "$n3 leave-group  $dst3_f0 $group0"

$ns at 0.1  "$video1 start"
$ns at 2500.0 "$video1 stop"
$ns at 0.1  "$video2 start"
$ns at 4500.0 "$video2 stop"
$ns at 0.1  "$video3 start"
$ns at 7000.0 "$video3 stop"
$ns at 7200.0 "finish"

$ns run
```

4.2 Performance Metrics

In this section, we define the performance metrics used for comparison.

4.2.1 Acceptance Ratio

In the ratio of the number of data, packets received by the destination node to the whole number of data packets sent by the source mobile node. It can be evaluated in terms of percentage (%). This parameter is also called success rate of the protocols" as described in Equation (1):

Acceptance Ratio = Number of accepted request / Total number of requests Eq. (1)

Therefore, acceptance ratio varies between 0 and 100.

4.2.2 Throughput

Throughput is the average rate of successful message delivery over a communication channel, which can be denoted by bit or kilobit. This data may be delivered over a physical or logical link, or pass through some network nodes.

$$X = C / T \qquad \text{Eq. (2)}$$

Where X is the throughput, C is the number of requests that are accomplished by the system, and T denotes the total time of system observation.

4.2.3 Average number of stops

Average number of stops that is an important metric, which can increase the users' dissatisfaction when it is on high levels. This parameter is calculated by tracing the output file and counting the number of stops during playback time.

4.3 Simulation Environment

These simulations have been done in different frameworks.

Patchpeer considers a combination of WWAN and WLAN networks.

LCBBS and Network-Aware techniques are implemented over a LAN.

4.4 Results and Discussion

The simulation results are shown in this section in the form of line graphs. The X-axis represents the buffer size. The. Y-axis represents different performance metrics.

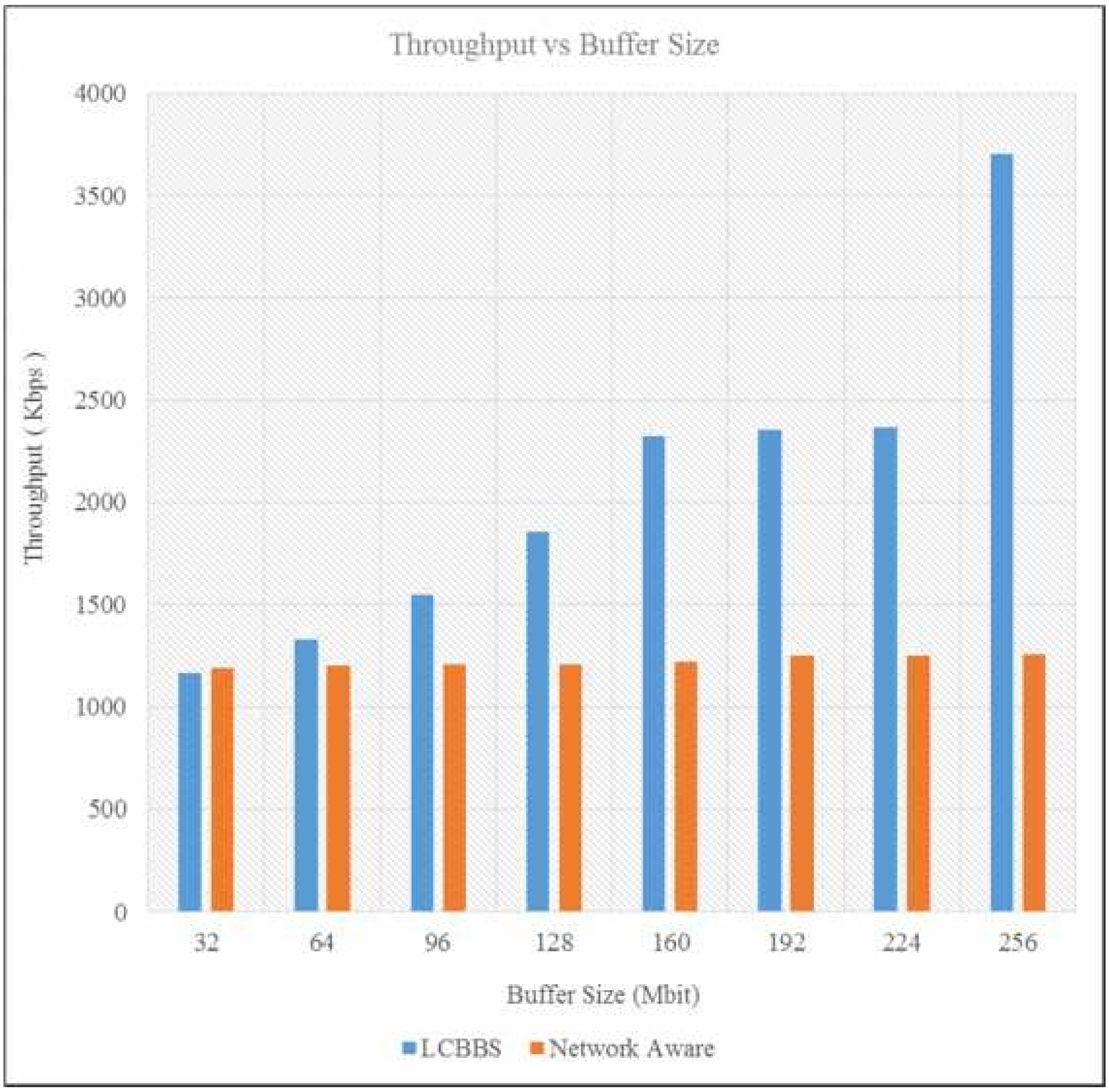


Figure 4.1: LCBBS vs Network-aware, throughput and buffer size

In Figure 4.1, a comparison between Throughput and Buffer Size for both LCBBS and Network-aware methods has been revealed. By increasing buffer size, LCBBS showed better throughput, while Network Aware remained steady in terms of throughput. The rate for LCBBS exceeds 3500 Kbps when the buffer size was set to 256 Mbit. As a result, throughput for LCBBS is better than the Network-aware approach.

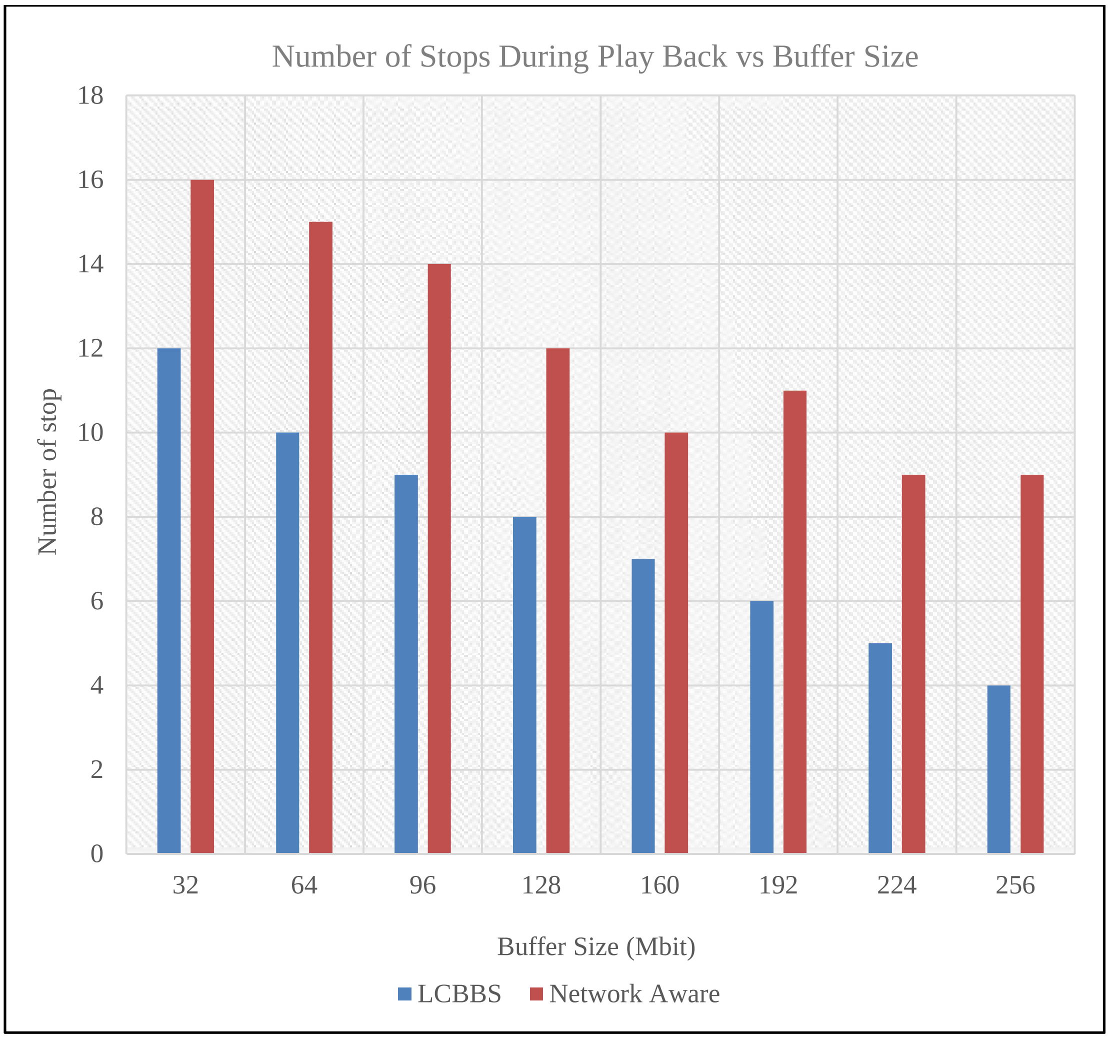


Figure 4.2: LCBBS vs. Network-aware, Number of stops and buffer size

In Figure 4.2, by increasing the buffer size, the number of stops during playback was decreased for both LCBBS and Network Aware; however, the number of stops for LCBBS method was lower than Network Aware; thus, with respect to number of stops, LCBBS also surpasses the Network-aware method.

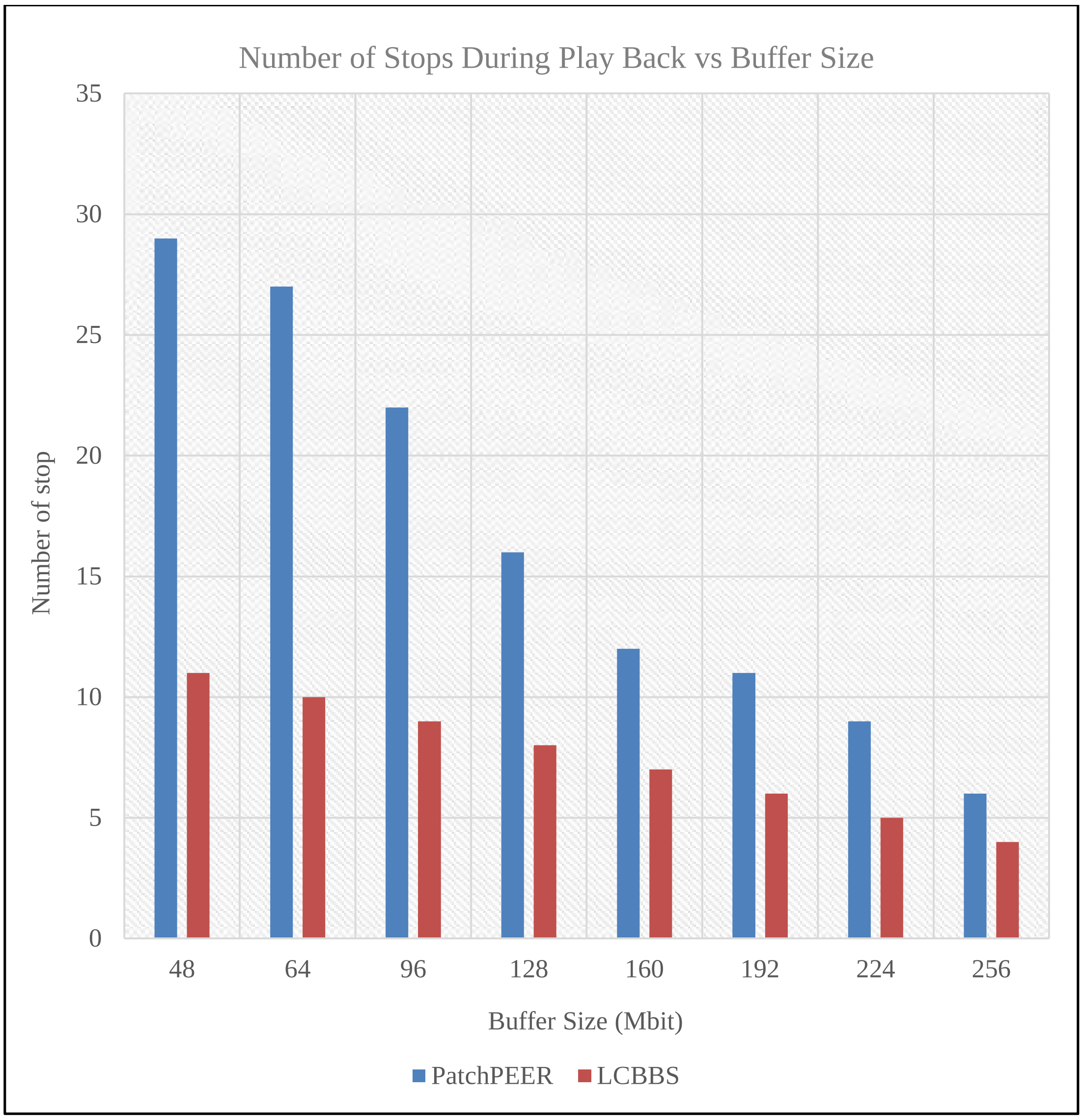


Figure 4.3: Patchpeer vs. LCBBS, Number of stops and buffer size

Figure 4.3 compares the number of stops during play back for Patchpeer and LCBBS. As buffer size is increased, the number of stops decreases for both methods. However, the number of stops during play back for Patchpeer are significantly higher than LCBBS; so LCBBS has better performance in comparison to Patchpeer.

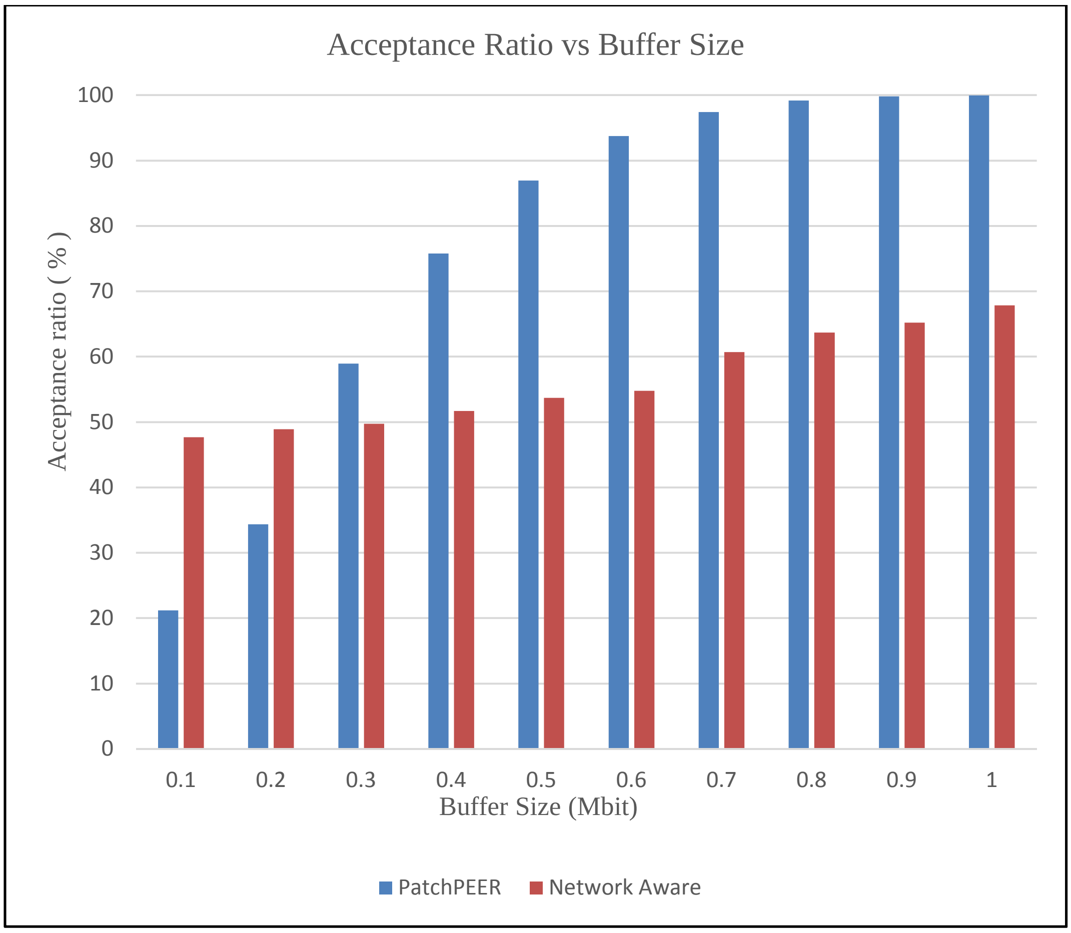


Figure 4.4: Patchpeer vs. Network-aware, Acceptance ratio and buffer size

Figure 4.4 compares the acceptance ratio for Patchpeer and Network-aware method. By increasing the buffer size, Network Aware shows a gentle increment in acceptance ratio especially for large buffer size. Interestingly, Patchpeer method surpasses Network Aware when buffer size exceeds 0.3; and this rate reaches about 100 when buffer size is more than 0.8. The conclusion is, Patchpeer method has higher acceptance ratio for buffer size > 0.4.

The number of buffer size has been chosen regarding the reference for keeping the originality of our simulation .This task is repeated in figures 4.8 and 4.9 as well.

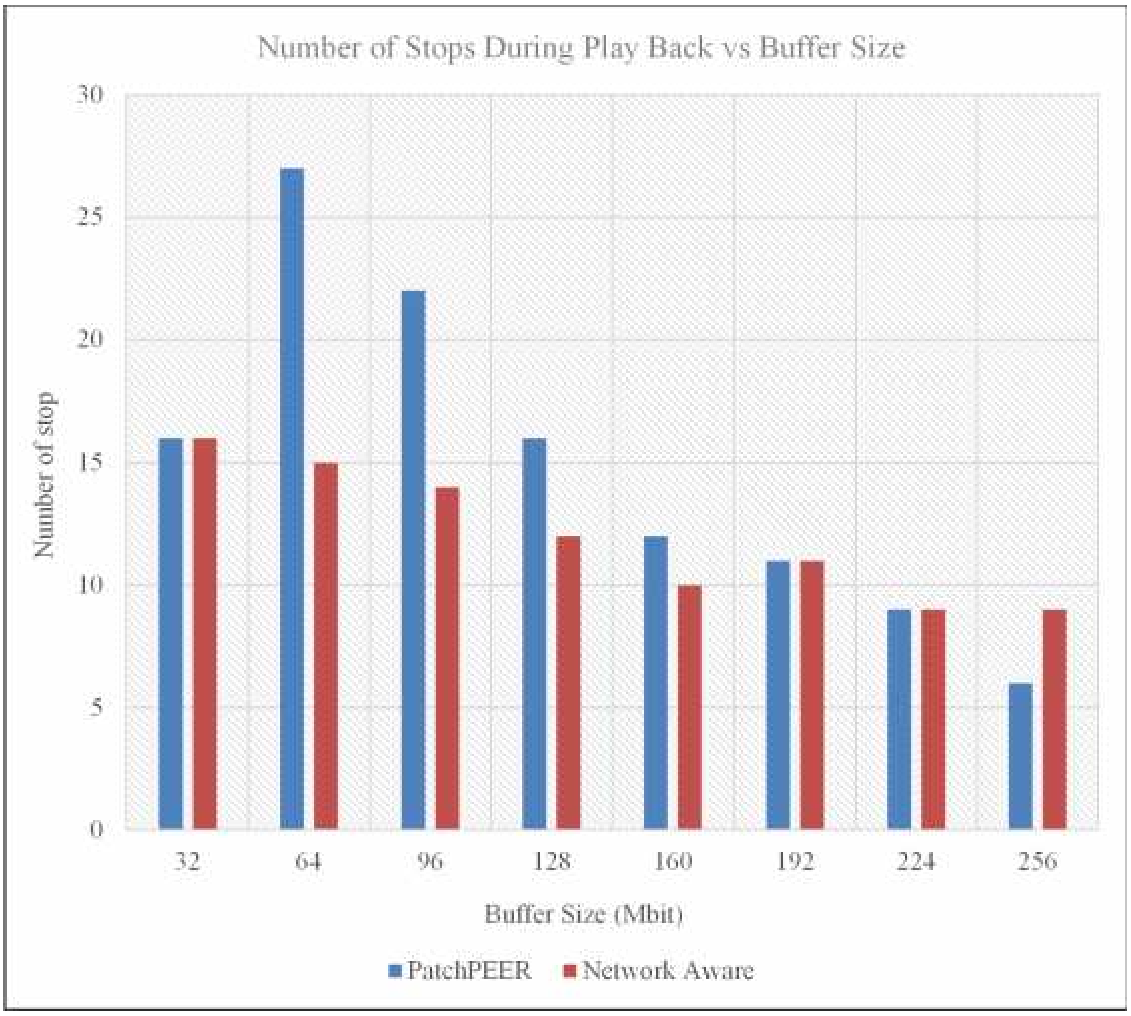


Figure 4.5: Patchpeer vs. Network-aware, Number of stops and buffer size

A comparison between Patchpeer and Network Aware is made in the case of the number of stops during playback in Figure 4.5. Both methods show almost the same result when their buffer size goes up. For Patchpeer, there is a significant jump from first column (32 Mbit) to second (64 Mbit), but ignoring this unforeseen rise, the rest of this column chart shows a normal downfall for Patchpeer method.

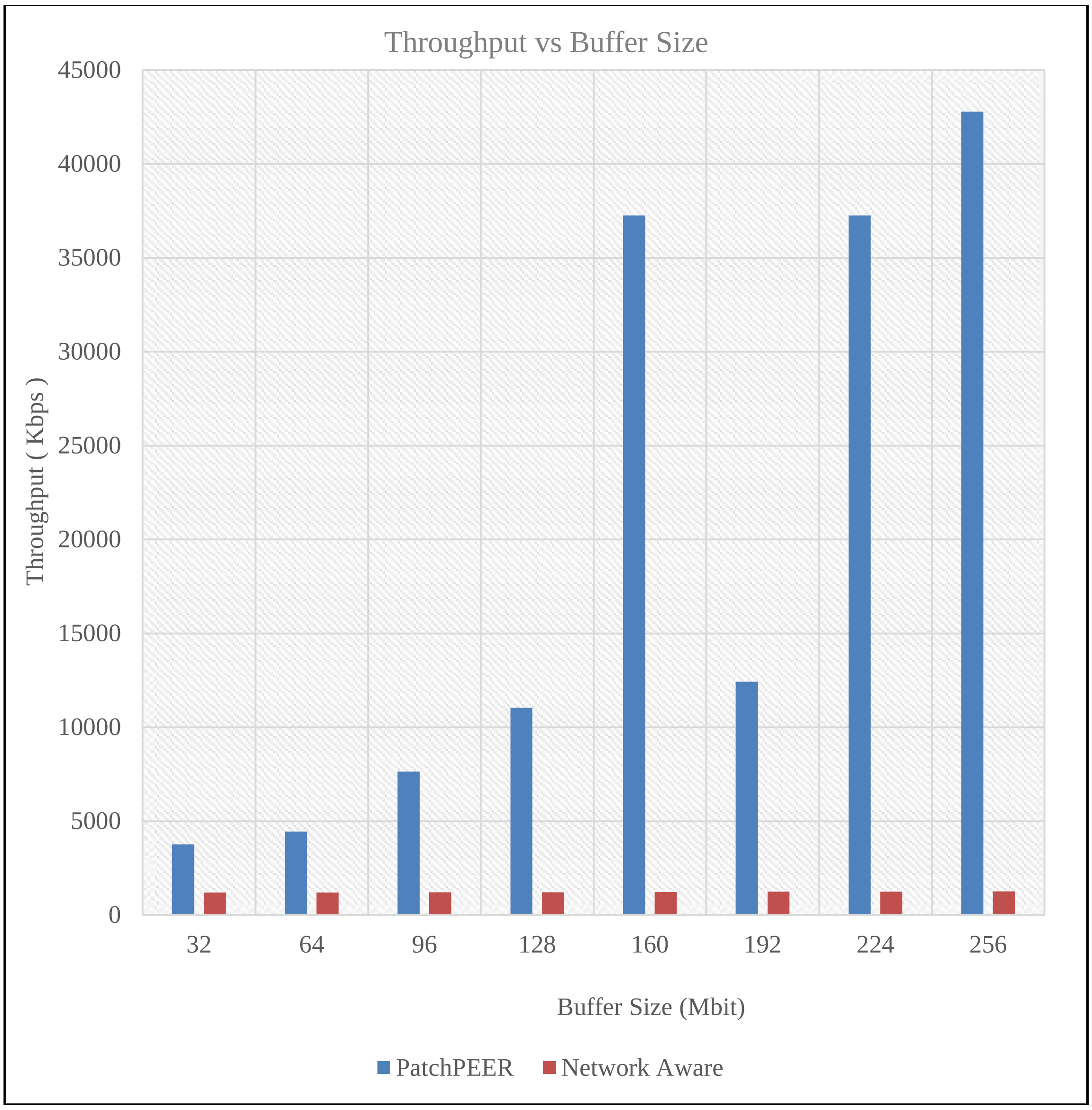


Figure 4.6: Patchpeer vs. Network-aware, Throughput and buffer size

Figure 4.6 shows that, by increasing the buffer size, throughput also goes up in Patchpeer, while this rate remains steady in the case of Network-aware. It can be concluded that the size of buffer has a little effect on Network-aware throughput. At first look, this influence does not show itself but numerical investigation proves these changes. Against this affect, buffer size plays an important role and effects on throughput of Patchpeer approach until 160 Mbit, but after that, we can see a sudden downfall in 192 Mbit, but immediately it starts to rise again.

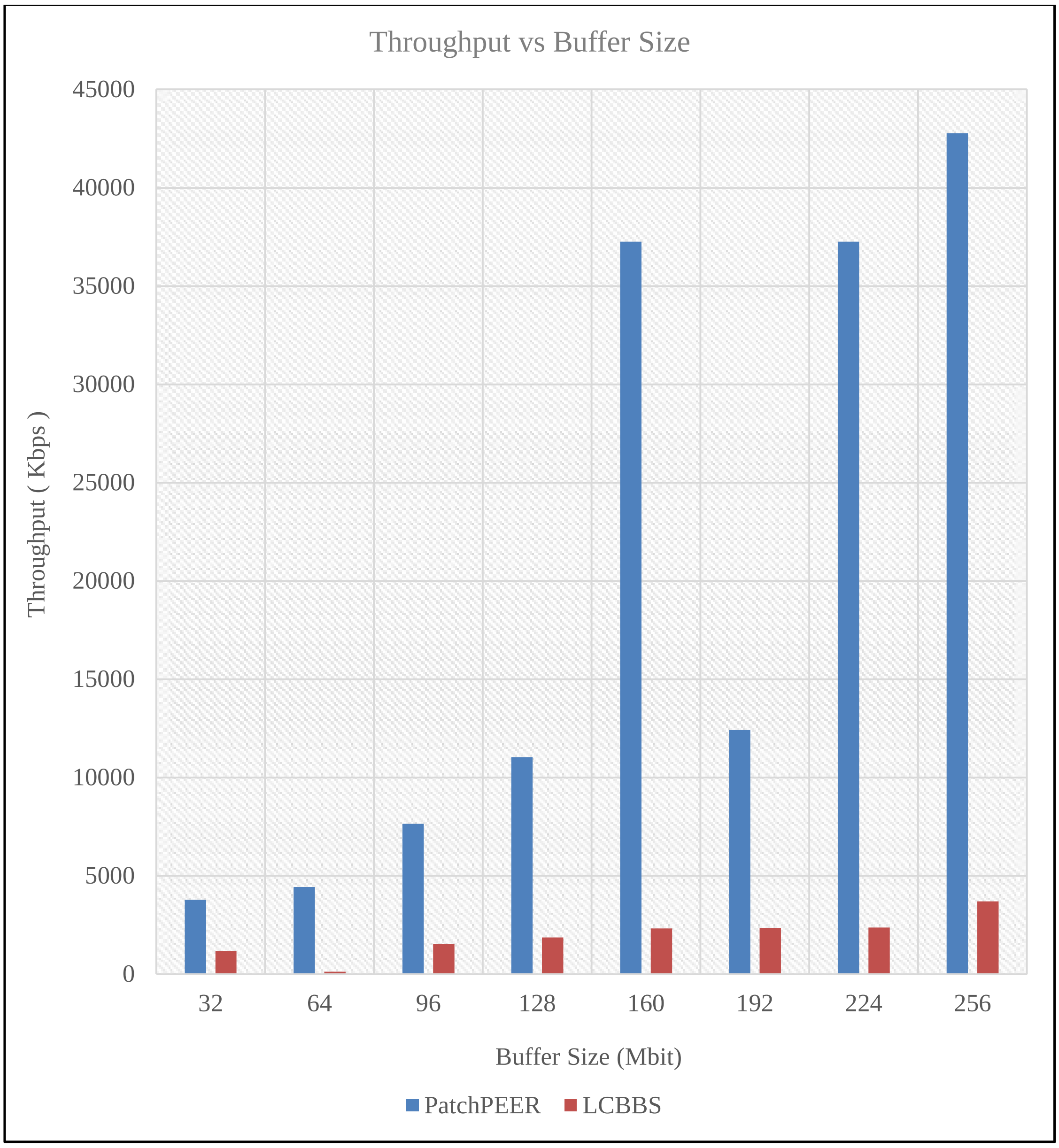


Figure 4.7: Patchpeer vs. LCBBS Throughput and buffer size

According to Figure 4.7, again, the rate of throughput is significantly goes up in Patchpeer in the case of buffer size increment. Conversely, this rate has a gentle change for LCBBS when buffer size grows up. Figure 4-7 shows that, the chart has a sharp downfall from 160 to 192, when it is reaching to 38000 Kbps, but again from 192 it starts to rise and reaches to 42000 Kbps. This rate cannot exceed about 4000 Kbps for LCBBS, while it is over 40000 for Patchpeer when buffer size is set to 256 Mbit.

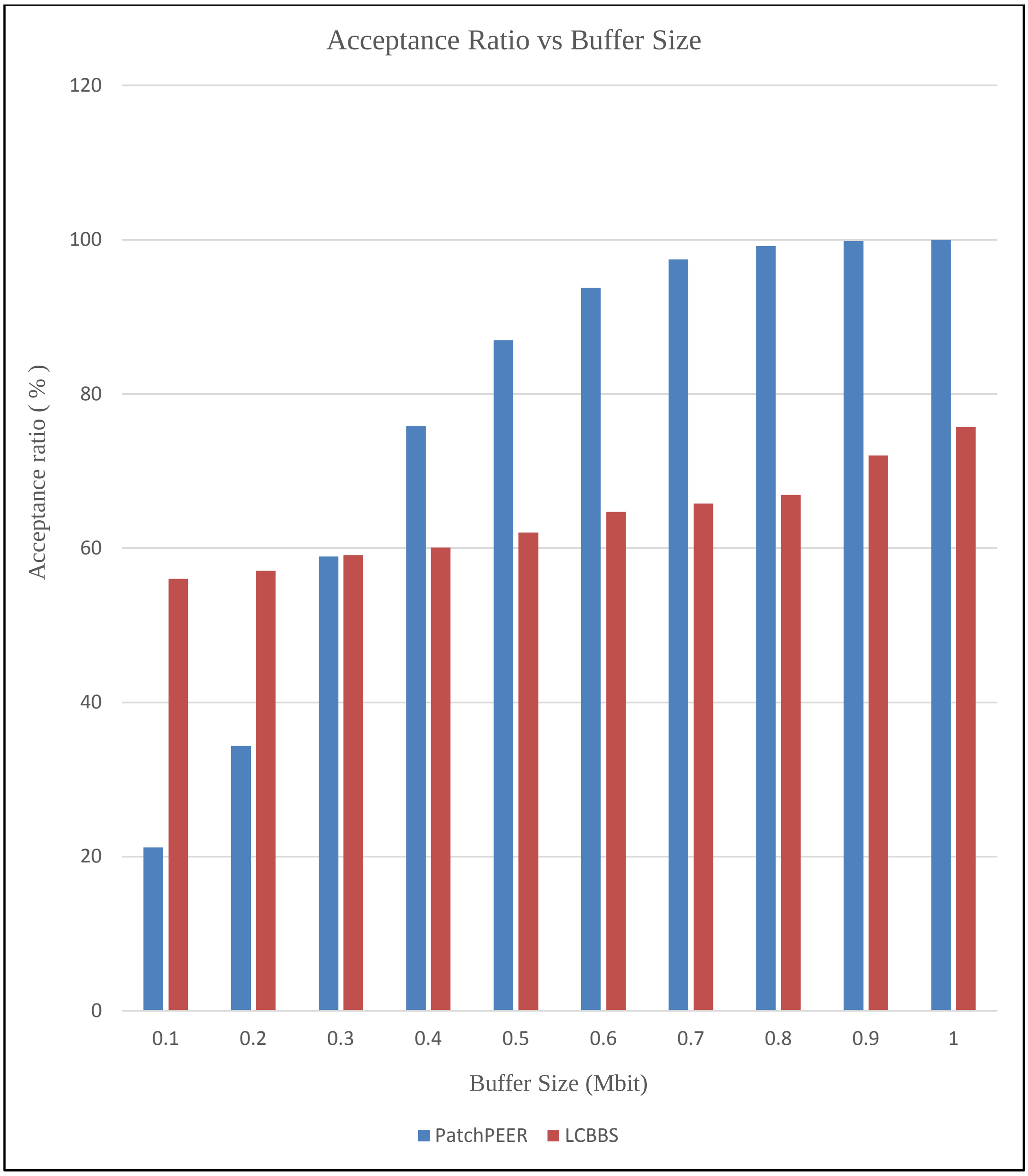


Figure 4.8: Patchpeer vs. LCBBS, Acceptance ratio and buffer size

Figure 4.8 compares acceptance ratio for Patchpeer and LCBBS for different buffer sizes. A significant increment in acceptance ratio is made when buffer size is increased. It shows the role of buffer size on acceptance ratio in Patchpeer method. Nevertheless, buffer size has a small effect on acceptance ratio in LCBBS method.

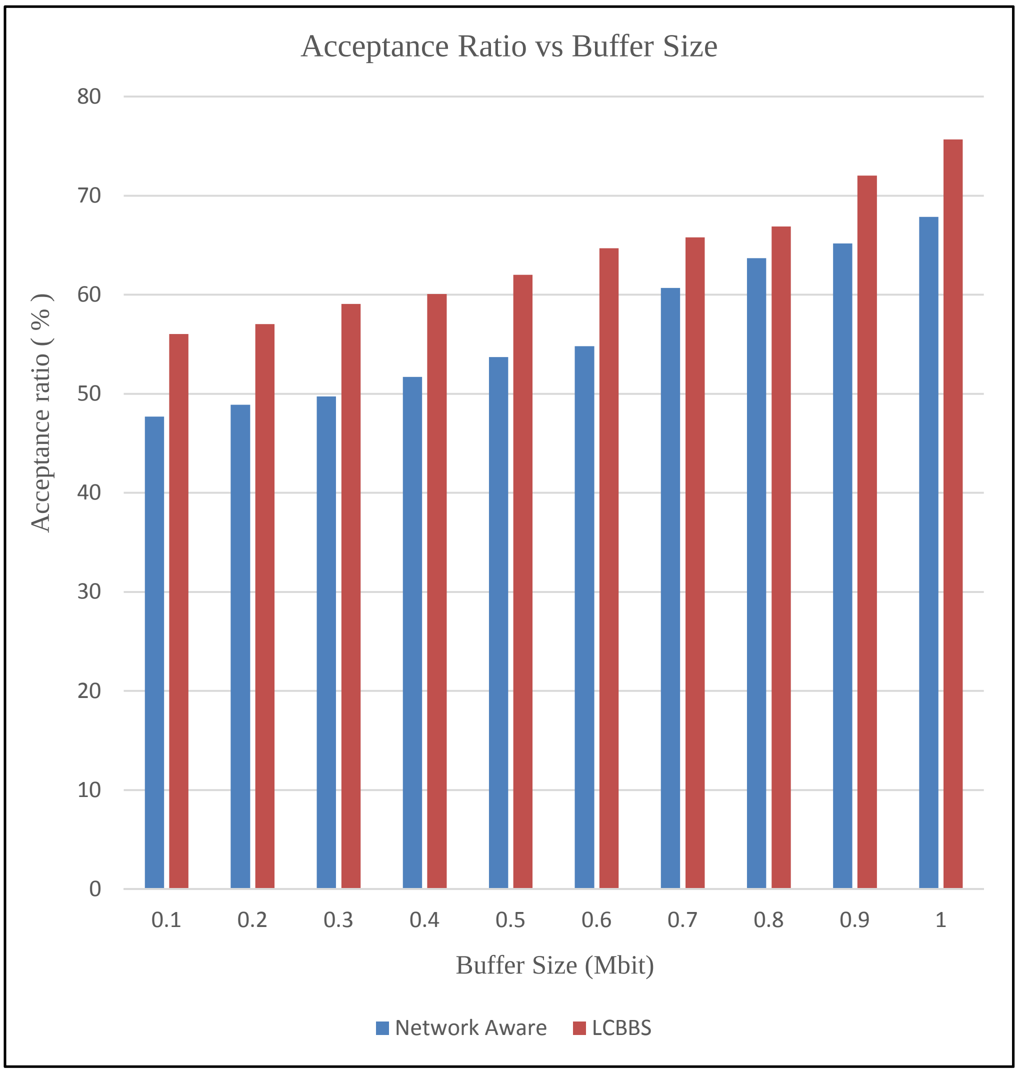


Figure 4.9: LCBBS vs. Network-aware, Acceptance ratio and buffer size

Finally, a comparison between buffer size and acceptance ratio is made for both Network Aware and LCBBS. Figure 4.9 shows that, by increasing buffer size, acceptance ratio is also increased for both methods. However, acceptance ratio for LCBBS is always better than Network-aware.

4.5 Concluding Remarks on Simulation

Based on the results of our simulation and charts, it can be said that simulated methods in this study, have shown the following behavior:

1- In LCBBS method, with increasing buffer size, throughput has increased and the number of stops has decreased, but these changes were less, compared to the Patchpeer method.

2- In Patchpeer method, both throughput and percentage of acceptance ratio have had increased. Throughput in comparison with Network-aware and LCBBS increased sharply. Percentage of acceptance ratio increased a little more than the two other methods. The number of stops decreased with increasing buffer size parallel with Network–aware.

3- Observing the Network-aware technique’s behavior, we see that increasing buffer size has increased throughput slowly but less than LCBBS and much more less than Patchpeer method. It also has an increased percentage of acceptance ratio parallel with LCBBS, but again less than the Patchpeer technique. Finally, the number of stops have decreased, parallel with LCBBS, with increasing buffer size.

Based on the results of our simulation, it can be said that among these three techniques, in most of the cases Patchpeer has shown a better behavior considering all three parameters. Nevertheless, it should be mentioned that LCBBS has also shown a good behavior, but it is not as good as Patchpeer. If LCBBS was implemented on a wireless network, its performance could have been similar or better to the Patchpeer technique, but this obviously requires further work.

Chapter 5

CONCLUSION

The research on Video on Demand (VoD) systems is important, and still faces many challenges and open issues; the main goal in designing such systems must be achieving a feasible large-scale and high-quality service with lower costs and fewer deployment constraints.

The Internet is the most popular environment of connected users; it is a publicly accessible environment of interconnected computer networks accessible worldwide. The Internet is a ‘network of networks’ made up of domestic, academic, business and government entities. Hence, the Internet has become the most important environment to deploy large-scale video service. VoD represents an attractive commercial service to be offered in a public and global scale environment like the Internet.

In the last few years, multimedia streaming technology has shown significant growth on the Internet. There are different applications, such as news, education, training, entertainment or advertising. Currently, video streaming on the Internet basically relies on real-time transmission of live events and streaming of on-demand contents (e.g. YouTube). Nevertheless, multimedia services on the Internet still present strong restrictions of quality and availability; a user commonly deals with long start-up delays, frozen- frames or even the removal of some content for reason of copyright infringement.

As VoD service over the Internet became more popular, research on VoD has intensified. Due to increasing demand for VoD applications in the Internet, research for improving the performance of VoD systems plays an important role. In this thesis, three different methods have been compared in order to show the efficiency of recently proposed methods for VoD systems.

These three methods have been chosen considering their solutions for efficiency and their structure type. Also it should be mentioned that their applicability and their well-defined documents needed to simulate them have been taken into consideration in our choice. In addition, the easiness to find the details of implementation for each technique and having joint points in order to select the simulation parameters had effect on selecting these three technique out of proposed technique in the field of VoD.

LCBBS, Patchpeer and Network-aware are compared in simulated performance evaluations. The results show that by using Patchpeer technique, efficiency will be higher and system will be more reliable also, users will be more satisfied.

The major problem affecting VoD system performance is the difference between download and display rates for requested video clips, which can cause stops during playback time. The results show that among these three methods Patchpeer has less number of stops.

As a future work, it is recommended to investigate the effect of more techniques and other VoD methods with more parameters, in order to be able to compare different methods with respect to different criteria.

Undoubtedly, this purpose requires much more research, survey, new techniques, and their parameters should be taken into consideration.

REFERENCES

[1] Zhang, M., Liu, J., Qin, Z., & Ye, M. (2010, 4 17). Proxy caching for peer-to-peer live streaming . *Computer networks*, *54*(7), 1229-1241.

[2] *The Network Simulator - ns-2*. (n.d.). Retrieved from www.isi.edu/nsnam/ns/

[3] Tanenbaum, A. S., & Wetherall, D. J. (2010). *Computer Networks* (5 ed.). Prentice Hall.

[4] Sarper, H., & Aybay, I. (2007). Improving VoD Performance with LAN Client Back- End Buffering. *IEEE Multimedia*(1), 48-60.

[5] Sarper, H., & Aybay, I. (2007). A video locality based buffering mechanism for VoD systems. *IEEE Transactions on Consumer Electronics, 53*(4), 1521-1528.

[6] Abbasi, U., & Ahmed, T. (2011). COOCHI_G: Cooperative Prefetching Strategy for P2PVideo-on-Demand System. *IEEE Consumer Communication and Networking confrence.*

[7] Tai, T., Kien, A., Ning , J., & Fuyu, l. (2009). PatchPeer: A scalable video-on-demand streaming system in hybrid wireless mobile peer-to-peer networks. *peer to peer networking and applications, 2*(3), 182-201.

[8] He, X., Tang, X., You , J., & Xue, G. (2004). Network-aware multicasting for Video-on-Demand services. *IEEE Transactions on Consumer Electronics, 50*(3), 864-869.

[9] Yuabo , Z., Hui, W., Pei, L., Zhihong , J., & Chao, G. (2010). How can peers assist each other in large-scale P2P-VoD systems. *International conference on Multimedia Information Networking and Security*, (pp. 198-202).

[10] Guan, Yifeng, H., & Ling. (2009). Improving the streaming capacity in P2P VoD systems with helpers. *IEEE International Conference*, (pp. 790-793).

[11] ZhiHui, L., ShiYoung, Z., Jie, W., WeiMing , F., & YiPing, Z. (2007). Design and implementation of a novel P2P- based VOD system using media file segments selecting algorithm. *IEEE international conference* , (pp. 599-604).

[12] Yung-Cheng, K., Chung-Nan, L., Peng-Jung, W., & Hui-Hsiang, K. (2012, June). A Network Coding Equivalent Content Distribution Scheme for Efficient Peer-to-Peer Interactive. *IEEE Transactions on Parrallel and Distributed Systems, 23*(6).

[13] Naor, Z. (2011). Efficient wireless access to video-on-demand services. *Communication Networks and Services Research Conference (CNSR), 2011 Ninth Annual*, (pp. 278-283).

[14] Jahon , K., & Kwangsue, C. (2011). Adaptive channel control scheme to reduce channel zapping time of mobile IPTV service. *Consumer electronics*, *57*(2), 357 - 365.

[15] Sun, Y., Gua, Y., Li, Z., Lin, J., Xie, G., Zhang, X., & Salamatian, K. (2013, March). The Case for P2P Mobile Video System over Wireless Broadband Networks: A Practical Study of Challenges for a Mobile Video Provider. *IEEE Network, 27*(2), 22-27.

[16] Skevik, K.-A., Geobel, V., & Plahemann, T. (2009, March). Evaluation of a comprehensive P2P Video-on-demand streaming system. *COMPUTER NETWORKS* , *53*(4), 434-455.

[17] Kasper, D., Evensen, K., Engelstad, P., Hansen, A., Halvorsen, P., & Griwodz, C. (2010). Enhancing Video-On-Demand Playout over Multiple heterogeneous access network. *7th IEEE Consumer Communications and Networking Conference* (pp. 47-51). IEEE Consumer Communications and Networking Conference.

[18] HUA, K., & Fei, X. (2010). A Dynamic stream merging technique for video-on-demand services over wireless mesh access network. *7Th Annual IEEE Communication Society conference* (pp. 1-9). IEEE.

[19] Shah, M. A. (2014). PHD Thesis "Distributed Continuous Media Streaming – Using Redundant Hierarchy (RED-Hi) Servers" Eastern Med. Univ. *Thesis*.

[20] Jin, S., & Bestavros, A. (2001). *Gismo: generator of streaming media objects and workload* (Vol. 3). ACM sigmetrics perform Eval Rev.

APPENDIX