PROVIDING ROBUST AND COST-EFFECTIVE LARGE-SCALE VIDEO-ON-DEMAND SERVICES IN NETWORKS

SZE KWOK TUNG

Ph.D

The Hong Kong Polytechnic University

2017

The Hong Kong Polytechnic University

Department of Electronic & Information Engineering

Providing Robust and Cost-effective Large-scale Video-on-demand Services in Networks

SZE Kwok Tung

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

June, 2016

Abstract

Video-on-Demand (VoD) service allows users to view any video provided by the service provider at any time and to enjoy the flexible control of video playback. However, the deployment of a large-scale VoD system demands an enormous amount of resources. One of the most challenging design aspects is how to deliver videos to a large community in a cost-effective manner. This thesis presents the results of our works in this area in order to build a robust and cost-effective VoD systems.

A feasibility study on supporting the VoD service over Wireless Mesh Network (WMN) is first carried out. In the proposed system, a video is divided into blocks, and the copies of those blocks are replicated and distributed to the routers in the network evenly. The required copies of each video block decrease with its timing order, i.e., fewer copies are needed for video blocks in higher order. Each router only stores a portion of the video. A video discovery scheme is developed to search the video blocks from the network. Different transmission strategies are considered that introduce three possible enhancements, named Video Block Broadcasting, Peer-to-Peer, and Time-Shifting. These enhancements aim to utilise the bandwidth of the routers so that more users can be served concurrently.

A simulation model is built to study the performance gain of each enhancement regarding blocking probability. Simulation results show that the proposed enhancements can achieve different levels of improvement. The Peer-to-Peer and Time-Shifting enhancements are suitable for all situations while the Video Block Broadcasting can only be used when the VoD system already has a substantial number of free channels. As a result, this motivates us to further study the potential on combining these enhancements.

A PB-P2P VoD system is then proposed. The system is a hybrid of Periodic Broadcasting (PB) and Peer-to-Peer (P2P) mesh pull. The proposed system aims at minimizing the startup delay, maintaining smooth playing, and supporting various interactive commands at best effort. The resources demanded by PB are independent of the number of incoming requests, and the service capability of P2P is increasing with the number of servicing users. Therefore, both techniques have good scalability that is suitable for delivering the popular videos to a large community.

In the system, a video is divided into segments which may be different in length according to the segmentation scheme in use. Each segment is then divided into a number of chunks with equal length. Server uploads the chunks to channels periodically. Clients switch among these channels and query peers in the same network concurrently to download the chunks. Scheduling algorithms are developed for clients to coordinate downloads from both channels and peers simultaneously and seamlessly under various conditions.

The system is implemented in ns-3. Simulations are conducted to examine the impact of different factors to the system performance. Simulation results show PB and P2P are a good complement to each other. P2P is critical to shorten the startup delay, improve the execution of the interactive commands, and recover the corrupted chunk. These characteristics complement the weaknesses of the PB. PB guarantees the maximum of startup delay, ensures smooth playback if no interactive command is involved, and provides an efficient and stable source of video data. These characteristics are difficult to achieve by using P2P alone.

The final work of this thesis is to study the potentials on using the multicast in the P2P network. Although the PB-P2P technique could build a robust and cost-effective VoD system, no every user would contribute his resources such as upload bandwidth to the VoD system in practice due to various reasons. Moreover, the P2P servers upload the chunks through unicast that can only serve a limited number of clients at a time. The multicast could be a potential solution that allows a P2P server to accept and satisfy requests from different clients for the same chunk by single upload stream. Simulation results show that Peer-to-Peer over Multicast (P2P-M) could achieve better performance when the user contribution rate is low and the number of P2P server is limited.

List of Publications

Book Chapter and Journal Paper

1. K. T. Sze, K. M. Ho, and K. T. Lo, *IAENG Transactions on Engineering Technologies: Special Issue of the International MultiConference of Engineers and Computer Scientists 2012*, ch. Efficient Video Streaming Over Wireless Mesh Networks, pp. 353–366. Dordrecht: Springer Netherlands, 2013

2. K. T. Sze, K. T. Lo, and J. Feng, *Encyclopedia of Multimedia*, ch. Multimedia Streaming in Wireless Mesh Networks. Springer, 2016. Accepted to appear

3. K. T. Sze and K. T. Lo, "A Hybrid PB-P2P Video-on-Demand System Minimizing Startup Delay and Supporting Interactive Commands at Best Effort." Under submission

4. K. T. Sze and K. T. Lo, "An Enhancement of a Hybrid PB-P2P Video-on-Demand System by Deploying Multicast in Peer-to-peer Network." In preparation

Conference Paper

1. K. T. Sze, K. M. Ho, and K. T. Lo, "Supporting Video-on-Demand Services over Wireless Mesh Network," in *Proceedings of International Conference on Computers, Communications, Control and Automation*, vol. 1, pp. 357–360, February 2011

2. K. T. Sze, K. M. Ho, and K. T. Lo, "Performance Evaluation of a Video Broadcasting System over Wireless Mesh Network," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, pp. 386–390, March 2012

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to those people who have supported me throughout these years.

First, I would like to express my sincere gratitude to my supervisor, Dr. Kwok Tung Lo, for his endlessly patience, valuable feedback and guidance during the study that makes the works in this thesis possible.

Second, I would also like to express my appreciation to Dr. King Man Ho for his advices and contributions on the works of Wireless Mesh Network (WMN).

Third, I would like to express my deepest gratitude to my family for their understanding and unconditional support throughout the study.

Fourth, I would like to express my big thanks to my best friend, Eunice Lee who is always with me during the study.

Last but not least, I would like to express my thanks to my colleagues, friends, relatives, and staffs of the Department of Electronic and Information Engineering for their help and moral support.

Contents

List of Figures

List of Tables

Abbreviations

ABM	Active Buffer Management
ACC	Acceptance
ADQ	Additional Download Quota
ALM	Application Level Multicast
ANSI	American National Standards Institute
API	Application Programmers Interface
APP	Application
AR	Arrival Rate
Buff	Buffer
CDN	Content Delivery Network
CH	Channel
CPM	Cooperative Peer Assists and Multicast
CPU	Central Processing Unit
CS	Client-Server
DL	Download

FB Fast Backward

FF Fast Forward

FP First Piece

GB Gigabyte

GHz Gigahertz

GNU GPLv2 GNU General Public License, version 2

H2O Home-to-home Online

HDD Hard Disk Drive

ID Identity

IEEE Institute of Electrical and Electronics Engineers

IGMP Internet Group Management Protocol

IMDb Internet Movie Database

INST Installation

IO Interactive Order

IP Internet Protocol

ISM band Industrial, Scientific and Medical band

JB Jump Backward

JF Jump Forward

LP Last Piece

LTS Long Term Support

MAX Maximum

MHz Megahertz

MPPO Middle Play Point Order

OS Operating System

P2P Peer-to-Peer

P2P-U Peer-to-Peer over Unicast

P2P-M Peer-to-Peer over Multicast

Pa Pause

PAE Physical Address Extension

PB Periodic Broadcasting

PB-P2P Hybrid of Periodic Broadcasting and Peer-to-Peer

Pl Playing

PlB Play Backward

PO Playing Order

SB Slow Backward

SEQ Sequence Number

SF Slow Forward

SM Scheduled Multicast

SSD Solid State Drive

TCP Transmission Control Protocol

TTL	Time to Live
UDP	User Datagram Protocol
VoD	Video-on-Demand
WMN	Wireless Mesh Network

Chapter 1

Introduction

1.1 Background and Motivation

Streaming service provides a quick way to access the multimedia content. Instead of downloading the complete copy of the content and then viewing, a streaming user can start the viewing when the beginning portion of the content is ready.

Streaming services can be loosely classified into two types, named live streaming and Video-on-Demand (VoD) [8]. The major differences of these two types include: i) The live streaming service generates the content in real time. The VoD service prepares the content in advance that allows users to view any video provided by service provider at any time. ii) Users in the live streaming service are usually watching the similar portion of the content, i.e., the latest generated content. Users in the VoD service could watch different potions of the content. iii) The VoD service usually supports more interactive commands such as pause and jump.

A basic VoD system consists of video server, delivery network, video clients, and users as shown in Figure 1.1. The video server stores a mass amount of videos and delivers the videos through delivery network. The delivery network

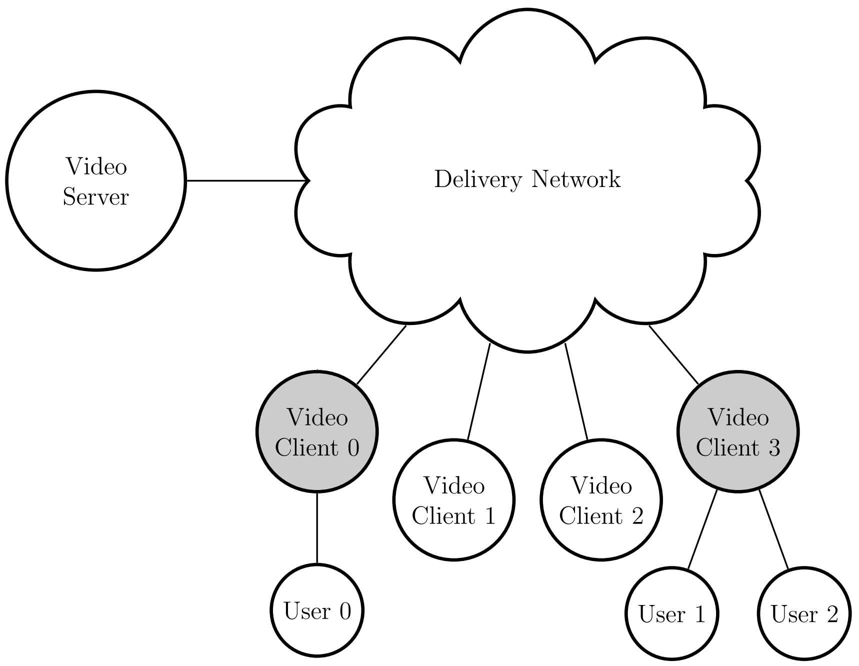


Figure 1.1: Elements of a VoD system

is a high throughput network that can forward video data to multiple clients. The video client is an end user device or application that downloads the video according to the request of the user(s). Normally, the interactive commands such as *pause* and *fast-forward* are also supported.

With the advance in information technologies in the middle of 1990, the implementation of VoD service has been a hot research topic. Many VoD systems were proposed and built. However, the profitable case is extremely rare due to various reasons such as the support of communication protocol, high investment cost in the infrastructure, and the competition with the video rental shops.

After two decades, the improvements and innovations of various technologies such as Peer-to-Peer (P2P) model and Content Delivery Network (CDN) reduce the cost of running a VoD service. These changes make the profitable VoD service possible. As the result, it motivates a numbers of researches and implementations of the VoD services based on these new technologies.

With the deployment of various VoD services such as Hulu [9], Netflix [10], PPS [11], PPTV [12], Tencent [13], Tudou [14], YouKu [15], and Youtube [16], the video traffic over the networks is increased significantly in last decades and shall continue to grow in the future.

Traditionally, Client-Server (CS) model is used to deliver the video data [17]. Server reserves a portion of its resources such as bandwidth for each incoming request. The more the requests, the higher demand on server resources. Unlike the file sharing services, subscribers of VoD services have lower tolerance on delay. A smooth playback experience is critical that the video data are expected to deliver in time. As a result, VoD service providers using CS model need to maintain a large server farm that can satisfy requests coming in the peak hours but which are mostly idle in the other hours.

User behaviour studies such as [7, 18–23] show that requests can be fluctuated considerably even within a day, especially in public holidays. For example, Figure 1.2 shows the requests in the peak hour are at least four times higher than the average number of requests in a day. This figure is based on the data set captured from the TeliaSonera started from 12th May to 13th September 2011 [7]. TeliaSonera is a TV-on-demand service provider with 30000 active users daily. Besides, the video quality could increase considerably within a short period. For example, study [24] shows the bit rate of videos streaming by MSN Video increases 50% more in nine months from 200 kbps to 320 kbps. As a result, these characteristics make CS model hardly be a robust and cost-effective solution for delivering video data.

To deliver video data efficiently, various methods have been proposed, and some of them are adopted by different VoD service providers. The building blocks of these methods include Periodic Broadcasting (PB), Scheduled Multicast (SM), CDN and P2P.

In PB, server uploads video data into channels continuously and periodi-

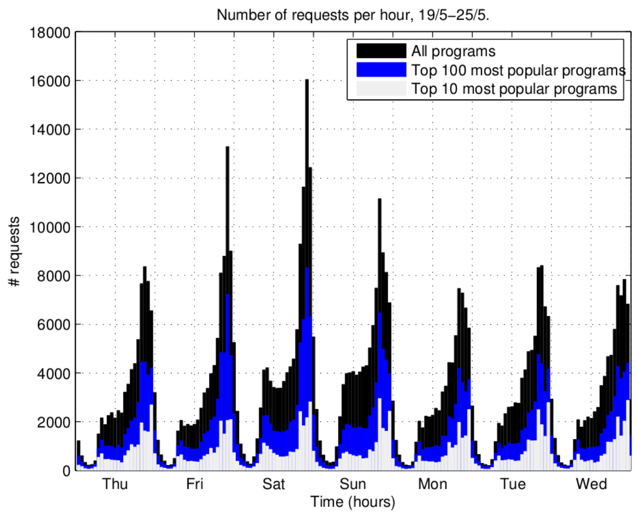


Figure 1.2: Number of requests per hour in a week captured from the Telia-Sonera started from 12th May to 13th September 2011 [7]

cally. Clients fetch the video data by switching among the channel(s). Therefore, resources reserved by the server is independent of the incoming requests that makes PB suitable for delivering popular videos. However, as clients can only fetch the video data currently available in the channel(s), clients who come after the start of each upload cycle suffer a long startup delay and the support on the interactive commands is minimal.

To improve these drawbacks, various approaches have been suggested. For example, staggered broadcasting [25, 26] uploads same video in multiple channels with a fixed shift on start time. The startup delay is inversely proportional to the number of channels. Pyramid broadcasting [27] divides the video into pieces with increasing size and uploads each piece in the dedicated channel with same bandwidth. Harmonic broadcasting [28] divides the video into pieces with equal size and uploads each piece in dedicated channels with different bandwidth. The maximum startup delay of these two approaches is proportional to the size of first piece. Active Buffer Management [29] fetches and caches the playing piece and the pieces surrounded the playing piece in order to provide some supports on the interactive commands such as *slow forward* or *fast* *backward.*

In SM, server allocates a multicast stream dedicated to a group of requests. For example, batching [30–33] serves the requests arrived closely by same multicast stream. First client in each batch suffers longest startup delay. Patching [34] serves the first request immediately by a multicast stream. The following requests are first served by a unicast stream and then merged to the suitable multicast stream. Client experiences minimum startup delay but this approach demands extra resources.

CDN [35–38], also named content distribution network, is formed by a large number of servers deployed in different geographical locations. VoD service provider pushes the video data to the CDN servers. Incoming requests are then redirected and served by the CDN server(s) nearby. CDN provides load balancing, and reduces the traffic of the core network. However, it takes time to synchronize the video data among the CDN servers. A VoD service provider may subscribe to several CDN services at the same time to guarantee the quality of service [39].

In P2P, a client makes requests to server and serves requests from other clients concurrently. P2P can greatly save the cost of the VoD service [24] and has good scalability. However, its stability is sensitive to arrival and departure of peers, and it incurs additional traffic such as messaging among peers. Tree-based push [40] and mesh-based pull [19] are the popular choices on implementing the P2P VoD system [8].

Tree-based push system uses the application layer multicast to deliver the video data. Clients are divided into groups. Clients in the same group are connected to each other to form a tree structure and the root is connected to the server. Each client in the tree receives data from its parent and then pushes to the children. Organization of tree against arrival and departure of peers [41] and utilization of upload bandwidth of peers are critical to this kind of system.

Mesh-based pull system uses the BitTorrent-like protocol [42, 43] that divides video into pieces and exchanges them one by one. Clients pull these pieces from VoD servers and other clients by exchanging information about the availability of pieces periodically.

Each of these building blocks has its own characteristics that motivates us to investigate whether we can take the strengths and complement the weaknesses of these building blocks to build a robust and cost-effective large-scale VoD system. As a result, we first carry out a feasibility study which proposes several techniques for an existing VoD system designed for Wireless Mesh Network (WMN), and examines their performances through a series of computer simulations. The results indicate these techniques can improve the VoD system in different degrees.

We then propose a Hybrid of Periodic Broadcasting and Peer-to-Peer (PB-P2P) system that aims at minimizing the startup delay, maintaining smooth *playing*, and supporting various interactive commands at best effort. The system is implemented in ns-3 and computer simulations are conducted. The results show PB and P2P are a good complement to each other for a robust and cost-effective VoD system.

At last, we investigate the function of multicast in P2P network. Patches are applied to the PB-P2P implementation and computer simulations are conducted. The results indicate the multicast can improve the performance of PB-P2P system when the availability of P2P server is limited.

1.2 Organization of the Thesis

The rest of this thesis is organized as follow: Chapter 2 gives a review on some background knowledge and the building blocks of a large-scale VoD system in more details. Chapter 3 presents the works on improving the VoD service over WMN. Several techniques have been designed and studied. Chapter 4 shows the proposed PB-P2P VoD system. The system combines the strengths and complements the weakness of PB and P2P to provide a robust and cost-effective VoD services. Chapter 5 presents the idea and results on using multicast in the P2P transmission. The system is developed based on the PB-P2P implementation in Chapter 4. Finally, Chapter 6 draws the conclusion of the thesis and gives some research directions.

Chapter 2

Literature Review

This chapter presents the general background knowledge on the Video-on-Demand (VoD) technologies that are essential to the following chapters.

2.1 Unicast, Broadcast, Multicast, and Application Level Multicast

Unicast, Broadcast, Multicast, and Application Level Multicast (ALM) are the popular transmission methods used by various VoD systems to deliver the video data. This section describes the differences among these methods in view of bandwidth consumption.

Figure 2.1 shows an instance of a simple VoD system which will be used as an example in the following sections. As shown in Figure 2.1, the VoD system consists of a video server, a delivery network, and four clients. In the system, suppose only clients C_1 and C_4 are using the VoD services and start watching the same video at the same time. The video is encoded in constant bit-rate and streamed at n Mbps for smooth playback. Moreover, both clients do not issue any interactive command.

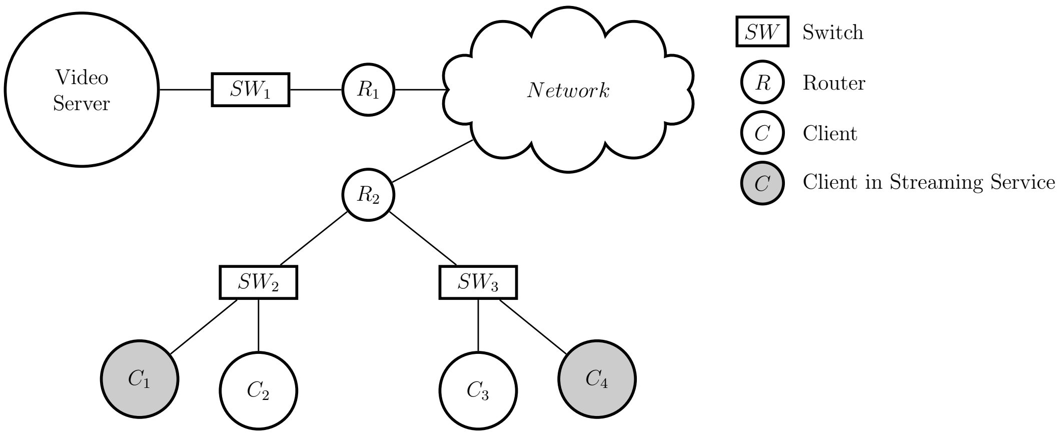


Figure 2.1: Instance of a simple streaming system

2.1.1 Unicast

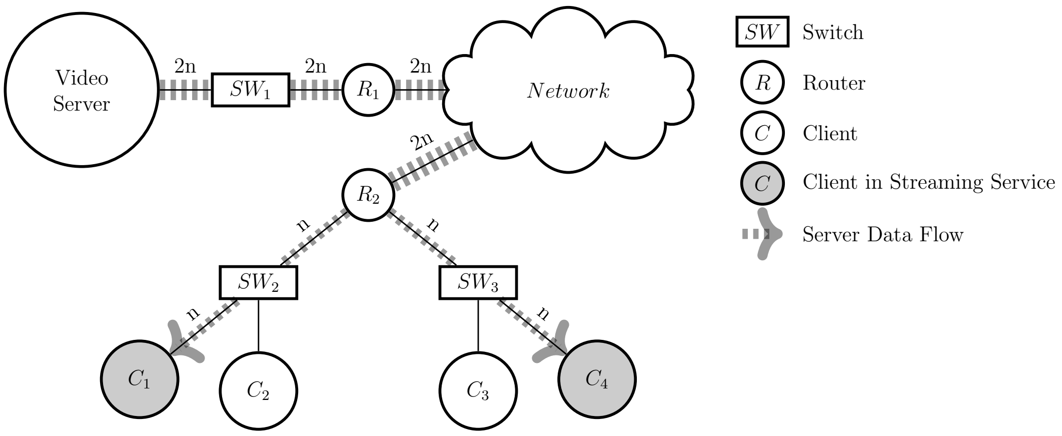


Figure 2.2: Streaming by unicast

Video server allocates a dedicated channel for each incoming request when unicast is used which is an one-to-one transmission method. As shown in Figure 2.2, $2n$ Mbps is required for the video server to satisfy the requests from two clients C_1 and C_4. The upload consumes $2n$ Mbps bandwidth in link $Video\ Server \rightarrow R_2$ and $1n$ Mbps in link $R_2 \rightarrow C_1$ and $R_2 \rightarrow C_4$.

The major advantages of unicast are easy to implement and compatible with the typical network environment. No great changes in network equipments such as allowing broadcast transmission or implementing a new kind of network protocol are needed.

However, unicast has poor scalability. The server bandwidth required is directly proportional to the number of requests. The more the requests arrive, the more bandwidth are demanded. For example, as shown in Figure 2.2, $N \times n$ Mbps is needed if N requests are received by the video server.

2.1.2 Broadcast

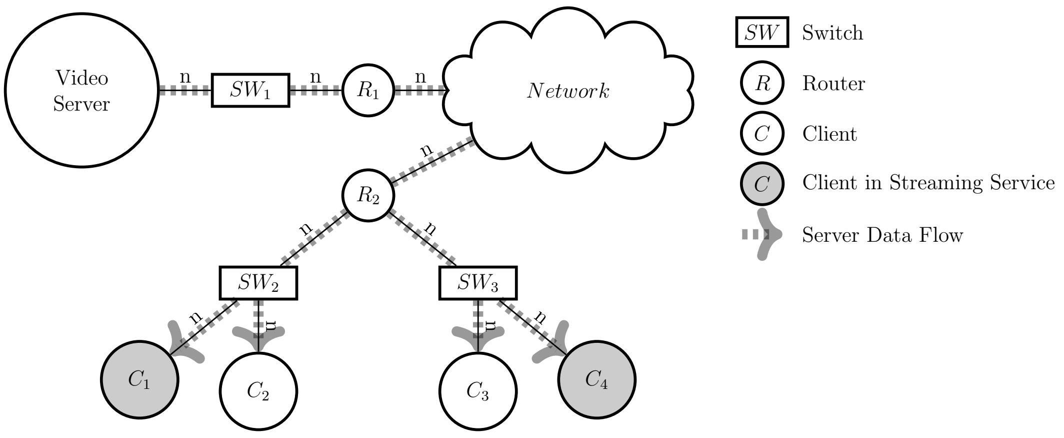


Figure 2.3: Streaming by broadcast

Video server allocates a fixed number of channels for all incoming requests when broadcast is used which is an one-to-many transmission method.

Suppose the video server allocates one channel for each video and all nodes in the delivery network permit the broadcast transmission. Only n Mbps is required for video server to satisfy the requests from two clients C_1 and C_4 as shown in Figure 2.3. The upload consumes n Mbps bandwidth in all links.

The major advantage of broadcast is the required server bandwidth is independent of the number of requests. For example, as shown in Figure 2.3, if two requests for the same video are received by the video server, n Mbps is needed. If N requests for the same video are received by the video server again, n Mbps is needed. The required server bandwidth remains constant at any number of requests that makes broadcast be an efficient method in servicing huge amount of clients. Moreover, broadcast is usually supported in the typical network environment.

However, if only a small portion of clients are requesting the VoD service, broadcast is inefficiency since it consumes the bandwidth and processing power of all nodes within same broadcast domain such as nodes in same Ethernet or same IP subnet. For example, client C_2 and C_3 in Figure 2.3 are forced to receive and examine the video data although they do not request for the VoD service. In a giant network such as Internet, a network of networks, it is rare that a video is requested by a large portion of nodes. Furthermore, the broadcasting data is usually dropped when moving across networks due to various reasons such as security. Therefore, broadcast-based VoD system is rarely found in large-scale network.

2.1.3 Multicast

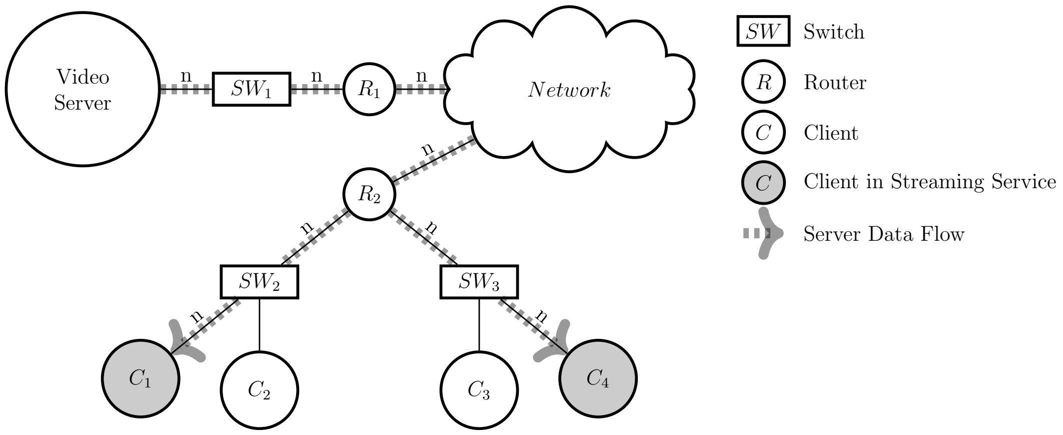


Figure 2.4: Streaming by multicast

Video server allocates a number of channels for the incoming requests when multicast is used which is also an one-to-many transmission method. It is similar to the broadcast except the multicast delivers the video data to the requesting clients only. For example, as shown in Figure 2.4, only n Mbps is required for video server to satisfy the requests from two clients C_1 and C_4. The upload consumes n Mbps bandwidth in link $Video\ Server \rightarrow R_2$,

$R_2 \rightarrow C_1$, and $R_2 \rightarrow C_4$. No video data is delivered to client C_2 and C_3.

The major advantage of multicast is the ability to serve multiple clients in single upload that can greatly save the server bandwidth. For example, as shown in Figure 2.4, if two requests for the same video are received by the video server, n Mbps is needed. If N requests for a same video are received by the video server shortly again, still n Mbps is needed. However, in order to forward the video data to the right destination, more resources are demanded by the nodes in the delivery network. For example, in IPv4 network [44], Internet Group Management Protocol (IGMP) [45–47] is required for the communication between the routers and clients to join or leave a multicast group. The routers are required to maintain an additional multicast table that records the details of each multicast group in use.

2.1.4 Application Level Multicast

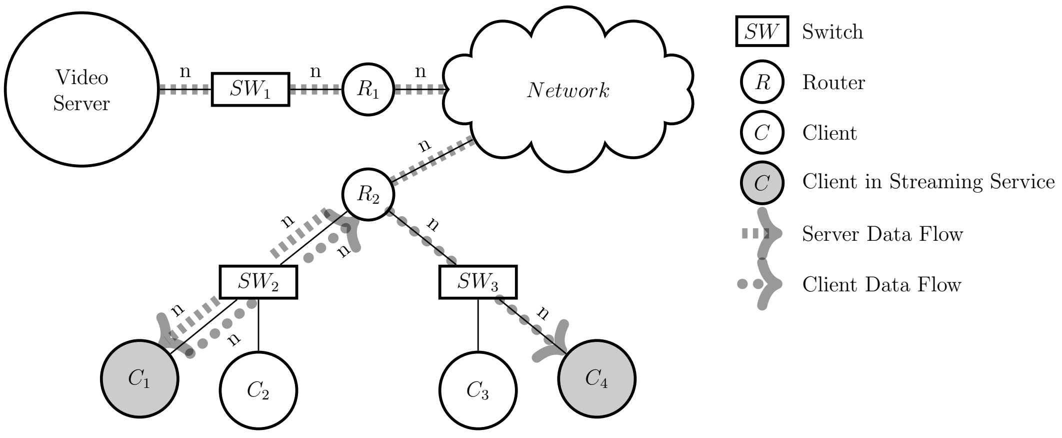


Figure 2.5: Streaming by application level multicast

ALM [48] is similar to the multicast except the multicast function is supported by the end-hosts instead of the nodes in the delivery network. In ALM, end-hosts are grouped and connected to each other to form an overlay network which is a logical network built on top of the existing physical network. The overlay network receives and re-distributes the video from the video server to all end-hosts. For example, as shown in Figure 2.5, only n Mbps is required for video server to satisfy the requests from two clients C_1 and C_4. The video server uploads the video to client C_1 and client C_1 forwards the received video to client C_4. The upload consumes n Mbps bandwidth in link $Video\ Server \rightarrow R_2$, $2n$ Mbps bandwidth in link $R_2 \rightarrow C_1$, and n Mbps bandwidth in link $R_2 \rightarrow C_4$.

The major advantage of ALM is the ability to provide multicast functionality over the unicast network without changing the existing network infrastructure. Moreover, ALM can greatly save the bandwidth of the video server and achieve the efficiency close to the multicast [48]. As shown in Figure 2.5, only n Mbps is required by the video server to satisfy the requests from Client C_1 and C_4. Part of the loading is shifted from the video server to the client C_1.

However, construct a proper overlay network to minimize the duplicated transmission and delay is challenging. For example, suppose client C_3 is now requesting for same video starting from the position that is watching by Client C_1 and C_4. To reduce the duplicated transmission and delay, the client C_3 should download the video from client C_4 instead of client C_1.

2.2 TCP and UDP

Transmission Control Protocol (TCP) [49] and User Datagram Protocol (UDP) [49–51] are the transport protocols commonly used by the VoD systems to deliver the video. TCP maintains an end-to-end connection that provides the reliable and in-sequence delivery. It is an one-to-one transmission method. UDP provides the additional multicast and broadcast functions which are one-to-many transmission methods. Moreover, unlike TCP, UDP would not merge or divide the message. The message boundary defined by the VoD system is preserved. However, UDP does not provide the guarantee on the delivery which would be handled by the VoD system itself.

2.3 Peer-to-Peer Model

Traditionally, the VoD systems are developed based on the Client-Server (CS) model solely. The server is a computer with high processing power and network bandwidth in order to serve a large number of clients concurrently. The clients are the end user devices that download the video.

With the advance in computer and network technologies, the boundary between the client and server becomes blurred that makes Peer-to-Peer (P2P) model possible. In P2P model, all clients act as server and client at the same time. Any client that connects to a P2P-based service will make requests and satisfy requests from the other peers concurrently.

Lets take Napster [52] as an example. Napster is the first famous P2P software for music sharing. When a Napster client is online, it first uploads its own list of shared music to the Napster index server. If a user wants to download a song, the client sends a query to the index server. Then, the index server returns a set of source clients who hold the song. After that, the client initiates the download session from the selected client that has the lowest ping. In other words, all Napster clients share their upload bandwidth and music files with the Napster network when they are online. The capacity of the Napster network increases with the number of servicing clients that makes the P2P model soon becomes a cure for many resource intensive applications such as VoD.

In a P2P-based VoD system, each client contributes part of its resources to the system. Clients are served by the server and other clients. For example, as shown in Figure 2.6, Client 1 may download the video data from the video

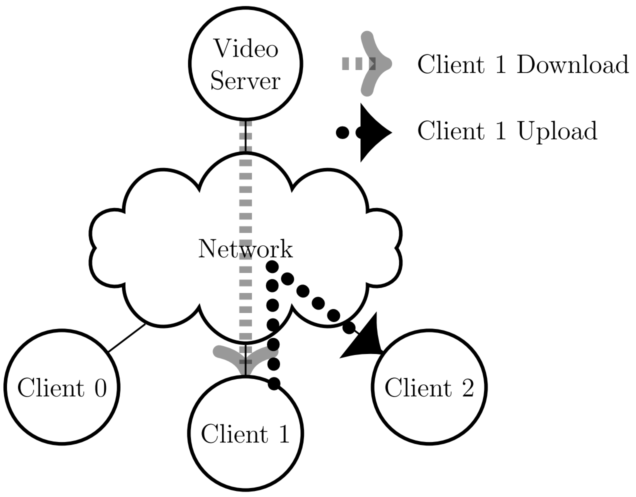


Figure 2.6: Peer-to-peer approach

server and upload the requested video data to Client 2 concurrently. As a result, the service capability is growing with the number of the clients that gives the system better scalability and more robust compared with the CS model. Moreover, since part of the loading is now shifted from the video server to the clients, the resources demanded by the server can be greatly reduced [24]. However, the stability of the P2P-based VoD service is sensitive to the arrival and departure of the clients. Additional traffic is generated such as messaging among clients and server that might not be as network friendly as the CS model [40].

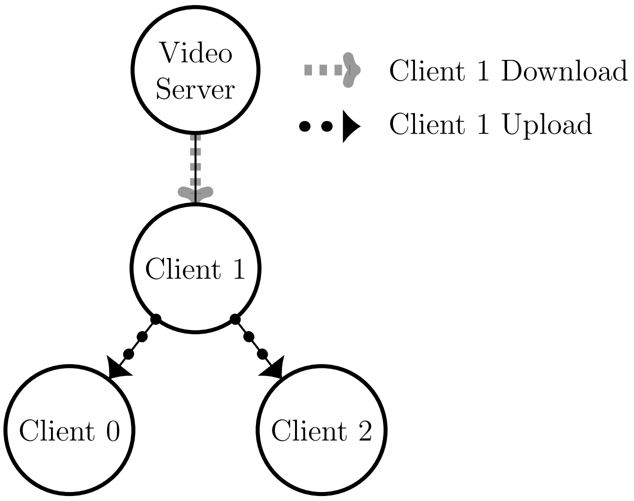


Figure 2.7: Tree overlay

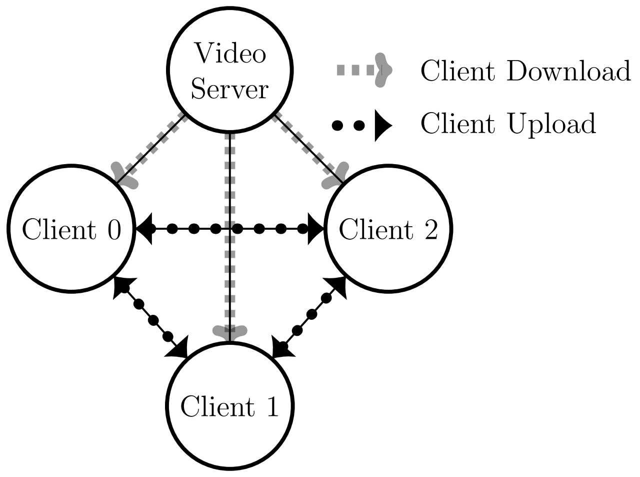


Figure 2.8: Mesh overlay

Tree-based push [40, 53–55] and mesh-based pull [19, 56–58] are two architectures commonly found in the P2P-based systems and studies [8]. The major difference between these two architectures is the overlay network used for streaming, which is the logical network formed on top of the existing network.

For illustration, let's take the topology shown in Figure 2.6 as an example. Clients in tree-based push architectures would be divided into groups based on certain criteria such as arrival time [40]. Then, clients in each group construct the tree overlay that the server pushes the video data to the root of the tree, and then the root forwards the video data to the child in each branch, and so on as shown in Figure 2.7.

The major challenge of the tree-based push architectures is how to maintain the tree structure against the arrival and departure of the clients. Since the video data is delivered from the root to the leaf sequentially, the transmission is blocked if any client leaves except those located at the leaf. An efficient algorithm is required to recover the tree structure. On the other hand, if a client joins, the client must be attached to the tree quickly to reduce the startup delay and utilize the upload bandwidth. Tree-based push architectures usually assume clients in the same tree are viewing the similar portion of the video. Therefore, the interactive commands such as *fast-forward* may not be supported.

Clients in mesh-based pull architectures construct the mesh overlay with other clients. Video is partitioned into pieces and clients pull these pieces from the server and other clients using the BitTorrent-like protocol [42, 43] by periodically exchanging information about the availability of pieces as shown in Figure 2.8.

2.4 Periodic Broadcasting

In Periodic Broadcasting (PB) [25–28, 59–65], the video server uploads the videos to the channels continuously and periodically. Clients download the video by switching between the channel(s). The resources demanded by the VoD system is dependent on the number of channels but not the number of clients in service that makes PB suitable for delivering popular videos.

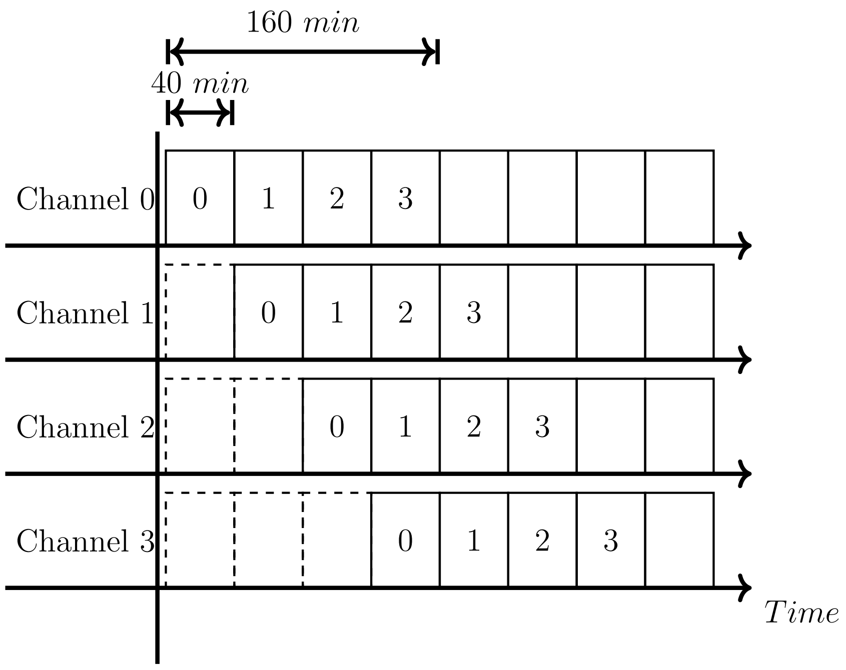


Figure 2.9: An example of stagger broadcasting

Staggered broadcasting [25, 26] is the simplest PB protocol. Suppose the length of a video is L, the video server allocates K channels for the video, and data rate of the channel is equal to the playback rate of the video. The video server first uploads the whole video to each channel with a fixed shift on start time equal to $\frac{L}{K}$. Then, the clients download the video by switching to one of these channels.

The maximum startup delay guaranteed by the protocol is $\frac{L}{K}$ which is inversely proportional to the number of channels. For example, suppose a video has length 160 minutes and 4 channels have been allocated to the video as shown in Figure 2.9. The maximum startup delay of a client is equal to 40 minutes.

As the clients fetch the video from one channel only, the protocol has low requirement on the client resources such as download bandwidth and storage throughput. However, the clients who come after the start of each upload cycle still suffer a long startup delay. As a result, various approaches have been proposed. For example, Harmonic broadcasting [28] divides the video into pieces evenly, and uploads each piece in dedicated channels with different bandwidth. Pyramid broadcasting [27] divides the video into pieces with increasing size, and uploads each piece in the dedicated channel with same bandwidth. Both approaches improve the startup delay. However, the clients are required to fetch video from multiple channels concurrently that demands more resources.

2.5 Wireless Mesh Network

A typical Wireless Mesh Network (WMN), or infrastructure WMN, consists of mesh routers and mesh clients [66, 67] as shown in Figure 2.10. The mesh clients are devices such as smart phone, laptop, and desktop which connect to the mesh routers for network access and have different degrees of mobil-

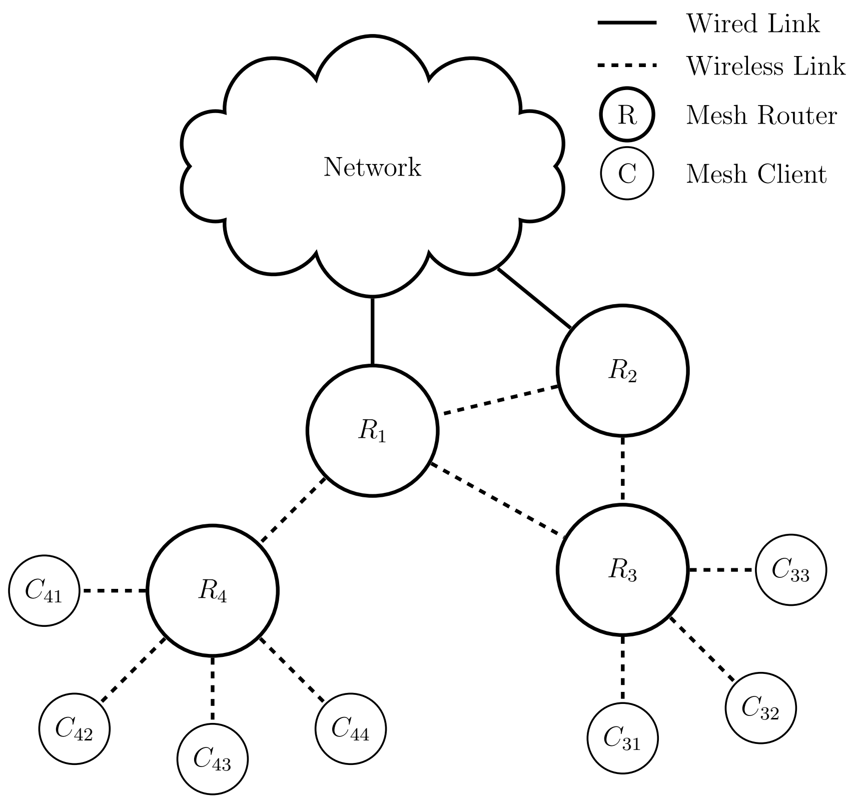


Figure 2.10: Wireless mesh networks

ity. The mesh routers usually have minimum mobility and use multiradio to achieve higher throughput. The mesh routers discover, establish, and maintain links with the neighbouring routers to form the backbone of a multihop wireless network that provides the network access to the mesh clients and routes network traffic to destination. Therefore, mesh clients such as C_{41} and C_{33} in Figure 2.10 can exchange messages via the WMN even no direct wireless link is established. The mesh routers with gateway function such as R_1 and R_2 can connect the WMN to other networks such as an Ethernet network or Internet. This integrates the WMN to other networks that makes devices in different networks can communicate with each other.

Client WMN [66] is another kind of WMN that is formed and maintained by mesh clients solely. The client WMN can combine with the infrastructure WMN to form the hybrid WMN.

WMN does no require wired link, therefore, WMN has lower cost and higher flexibility on deployment compared with other kinds of network such as single hop wireless network or Ethernet network [68, 69]. This makes WMN be an important element for the next generation wireless network. As a result, WMN has been included into various IEEE standards [70] such as 802.16 for wireless metropolitan area networks [71], 802.11 for wireless local area network [72–74], and 802.15 for wireless personal area network [75]. Moreover, various studies and developments have been carried out to explore the potential of WMN. For example, open80211s [76] is an open-source implementation of the IEEE 802.11 wireless mesh standard for the devices which run the Linux kernel. The open80211s aims to improve the interoperability of different 802.11s implementations [69]. One Laptop per Child Foundation [77], a non-profit organization, has implemented the wireless mesh network feature into the XO-1 laptop which is an inexpensive laptop for the children in developing countries where are lack of a communication infrastructure [69]. Cisco Meraki [78] is one of the companies provides commercial WMN solution.

2.5.1 Interference

Unlike the wired network, devices in a WMN share same communication medium with neighbouring devices. The communication medium is the specified frequency spectrum that is regulated by local authority and provides a limited number of channels. For example, IEEE 802.11 divides the 2.4 GHz Industrial, Scientific and Medical band (ISM band) into 14 channels. The availability of these channel is subjected to the regulations of different areas. Channel 1 to 11 are permitted in United States while channel 1 to 13 are permitted in Europe. The center frequency of channel 1 to 13 are separated by 5 MHz started from 2412 MHz to 2472 MHz. Channel 14 has center frequency at 2484 MHz. If only 11 channels are available, Channel 1 (2412 MHz), 6 (2437 MHz), and 11 (2462 MHz), which separate 25 MHz from each other, are the most popular combination of non-overlapping channels for IEEE 802.11b/g/n networks. These non-overlapping channels can be used concurrently without causing interference according to IEEE 802.11 suggestion. In other words, these three non-overlapping channels would be shared by devices operated in the same ISM band such as IEEE 802.11b/g/n devices [79], Bluetooth devices and microwave ovens. As a result, interference could happen easily that deteriorates the link quality.

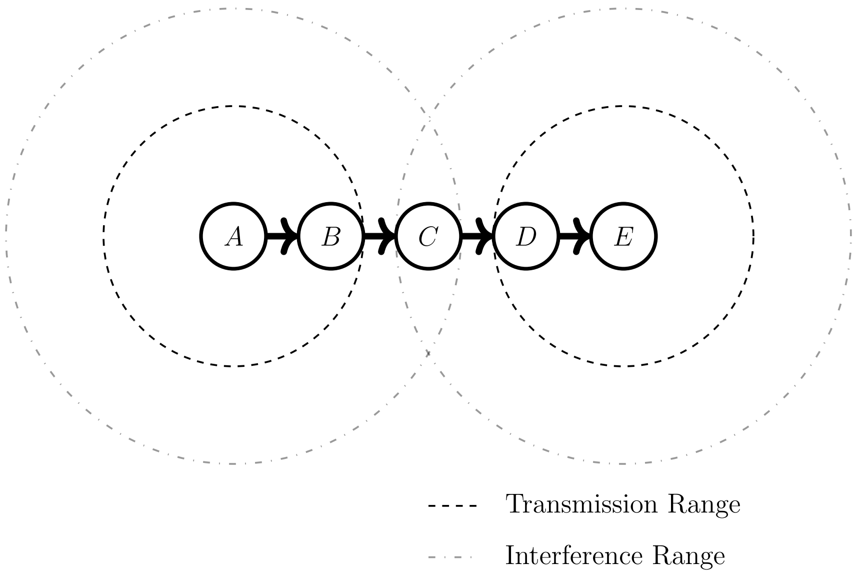


Figure 2.11: Intra-flow interference

Intra-flow and inter-flow interference are two types of interferences found in a multihop wireless network [80]. For ease of illustration, assuming devices shown in Figure 2.11 and Figure 2.12 use same channel for transmission and reception of data, and transmission range and interference range are static. Intra-flow interference happens when data is sent from A to E as shown in Figure 2.11. Devices compete the access of channel to forward the data from same flow. Inter-flow interference happens when data is sent from A to C and E to D concurrently as shown in Figure 2.12. Devices compete the access of

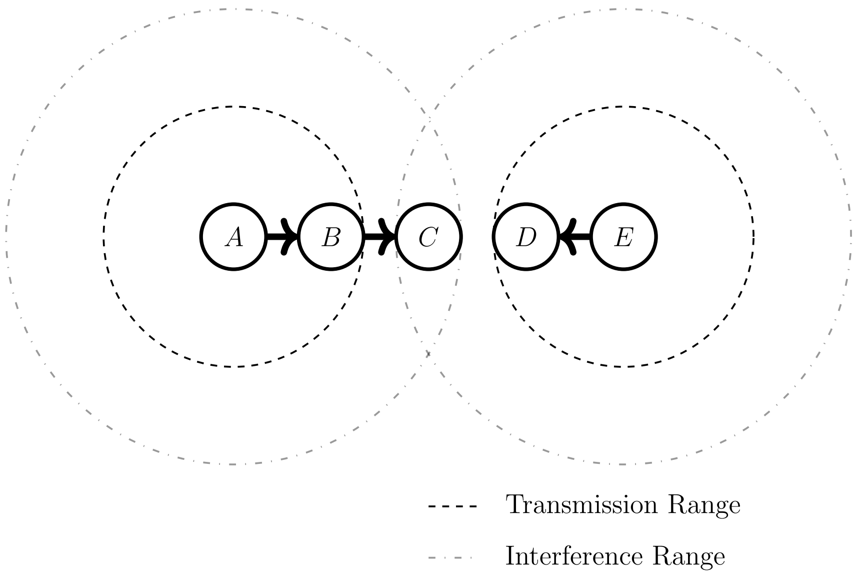


Figure 2.12: Inter-flow interference

channel to forward the data from different flows.

In these two examples, devices share the channel with their neighbouring devices within same interference range. For example, *A* shares the channel with *B* and *C* while *C* shares the channel with all other devices. Therefore, *C* generally suffers biggest interference and has minimum access to the channel that would become the bottleneck during the transmission.

To study the intra-flow and inter-flow interference, Cheng et al. [81] setup a test bed in a 7m x 7m room as shown in Figure 2.13. Various Experiments are conducted. The results shows that the link quality is deteriorated when a flow travels through more number of hops, or additional flow is presented. Moreover, the TCP outperforms the UDP due to congestion control in the TCP.

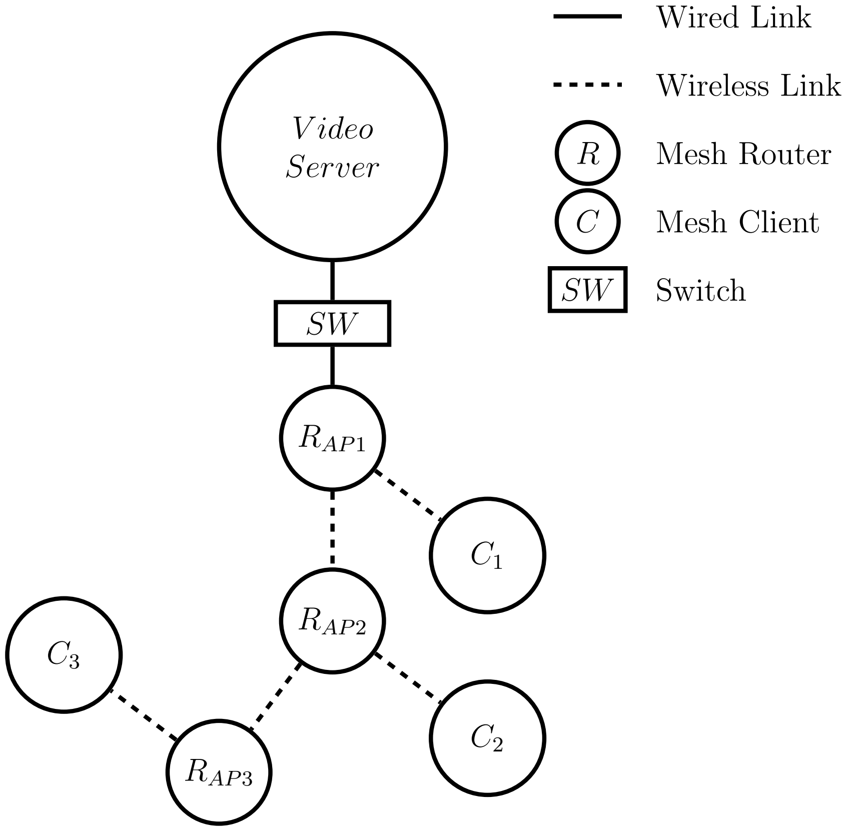


Figure 2.13: Intra-flow and inter-flow interference testbed

2.6 ns-3

ns-3 [82] is a discrete-event simulator aimed for research [83] and education. ns-3 is developed by an open-source project named ns-3 project, licensed under the GNU GPLv2 [84], and supported by various operating systems such as Linux and FreeBSD [85]. The source code of ns-3 can be downloaded from ns-3 homepage [82, 85] or package manager such as ubuntu Software Center and FreeBSD Ports.

ns-3 is regarded as the successor of the ns-2 [86], which is another popular network simulator [87], and is suitable for large-scale network simulations [87].

ns-3 provides a collection of network simulation models implemented in C++ and wrapped through Python. User can design and conduct the simulation by script written in C++ or Python [88]. Moreover, the user can also extend or create the network simulation models when it is needed [89].

2.7 Related Work

To provide quality VoD service, various studies have been conducted over the years, which include supporting the interactive commands and heterogeneous clients over the PB system, collecting and analysing user behaviours from different deployed services, improving the BitTorrent protocol for multimedia applications, and investigating various hybrid VoD system. This section briefly describes some works in these areas.

Fei et al. [29] proposed a technique, named Active Buffer Management (ABM), to support the discontinuous interactive commands over the PB VoD system. This technique keeps the *play point* in the middle of the buffered data by downloading the *player segment* and the segments nearby at best effort. To make *playing* mode running smoothly, a specific video segmentation scheme was proposed and the destination *play point* of the interactive commands is adjusted by the player.

To enhance the ABM, Poon et al. proposed the Greedy Channel Management scheme [90] which combines the ABM and the contingency channel to support the interactive commands. The contingency channel is used when the video data cannot be fetched from broadcast channel(s) in time. Tsai et al. proposed the buffer interpolation algorithm [91] which uses a circular queue to keep the *play point* at the center of the buffer. Wong et al. introduced Static Full Stream Scheduling [92] that takes users' behaviors into account to support the interactive commands in the IP multicast based VoD system. All these enhancements demand extra server resources.

Studies [7,18–22] collected and analysed the user behaviours of various VoD systems that provide different kinds of videos and have different user bases. Early departure behaviour is one of the common findings that a significant portion of requests have relatively short playback duration compared with the length of the requested video. Chen et al. [20] showed that the video browsing is the one of the causes of the early departure behaviours. Users request and sample the videos one by one until the interesting video is found. Carlier et al. [93] suggested that the delay caused by video browsing can be reduced by 1) adding bookmarks to the video, 2) buffering the video data around those bookmarks, and 3) displaying the bookmarks and buffering state at timeline. Besides, Ali-Eldin et al. [21] suggested that inter arrival time distribution fits to a stretched exponential distribution.

Cohen [42] mentioned the aims of the rarest first selection policy in BitTorrent protocol are 1) extract the whole file in shorter period of time from the seed, 2) ensure each piece of the file is always available against the departure of peers, and 3) make sure peers always have pieces that required by their neighbours. Moreover, the choking algorithms were discussed which aim to better utilize the download of peers and against the free riders who fails to maintain an acceptable ratio of download and upload.

Shah and Paris [43] suggested that the rarest-first selection policy and choking algorithms of BitTorrent protocol do not work well in multimedia streaming application. Two modifications were proposed. First, a sliding window mechanism is added to the rarest-first policy. The window contains the chunks which are in urgent need. Peers are restricted to download the rarest chunk within the window. Second, a new randomized tit-for-tat policy is used. It boosts the downloads of new peers at startup. As a result, those new peers are able to acquire more chunks and ready to serve other peers in a shorter period of time.

Chadagorn et al. [94] modified the Bittorrent protocol to support the live streaming in multiple bit rates, named Pilecast which is a P2P streaming system. Pilecast divides peers into different groups according to the bit rate. Streaming starts when the host publishes its torrent file to the tracker. Clients fetch the torrent file, select the suitable group which gives highest bit rate supported by current download bandwidth, and acquires the list of peers in same group from the tracker. The host keeps generating and encoding the content in different bit rates. Each content is then stored in a circular buffer with two slots, which are refreshed regularly. Peers in different groups pull the content from the host and other neighbours. Moreover, peers may switch to different group for smooth playback or higher bit rate according to the current download bandwidth measured at each fixed interval.

Huang et al. [24] showed that the P2P can dramatically reduce the resource of the video server. This study collects and analyzes the user behaviours in a nine-month trace from the MSN Video service, which adopts the client-server model. The analysis results are then used to study the running cost of a peer-assisted VoD system that clients redistribute the video data to each other and the servers provide guarantee on smooth playback by uploading the missing piece of data to the client.

Huang et al. [19] showed the building blocks of the PPLive [12] which is a P2P VoD system with distributed cache and served 150K users concurrently at the time of report. These building blocks includes 1) the size of cache, playback, and transmission unit; 2) the cache policy on client side; 3) the peer and content discovery policy; and 4) the methods to query video data. Moreover, the measurement results of user behaviours in PPLive were also presented.

Guo et al. [40] proposed a P2P patching VoD System named P2Cast. In the system, clients arrive closely are grouped and connected to each other to form a tree structure based on the proposed algorithm. The video data is delivered through application level multicast from server to the root and then each node of the tree. Clients missing the initial part of video data are patched by either the server or other clients. The simulation result showed that P2Cast has lowest requirements on server compared with the VoD systems using unicast or IP multicast patching although highest network loading is observed.

Febiansyah et al. [95] proposed a PB tree push VoD systems, PeerPB, to serve the clients with limited computational power or bandwidth. In PeerPB, server groups the clients, who arrives before the start of current upload cycle of the first segment, into multiple trees. The clients with more computational power or bandwidth are placed at the higher level of the tree. These clients then act as servers that may encode the received video data in lower bit rate and deliver to its child node(s).

Gopalakrishnan et al. [96] proposed a P2P batching VoD system, named Cooperative Peer Assists and Multicast (CPM). In CPM, a client acquires a chunk by first querying the server for outstanding multicast group. If the group is found, the server adds the client to the group and adjusts the start time of the multicast if needed. Otherwise, the client obtains a list of source peers from the server, and queries these peers one by one. If the list of source peers are exhausted, the client queries the server to create a new multicast group which would become an outstanding multicast group for other client requesting same chunk. Simulation results showed that CPM generally gives lowest loading on the server compared with other VoD systems using unicast, multicast, and server assisted P2P.

Chapter 3

Supporting Video-On-Demand Services over Wireless Mesh Network

As mentioned in Section 2.5, the Wireless Mesh Network (WMN) relies on the mesh routers with gateway function to connect to other networks. These routers such as R_1 or R_2 in Figure 2.10 forward all the traffic flowing across the boundary of the WMN. Moreover, these routers share same communication medium with their neighbouring devices that keen competition on the channel access can be expected. As a result, these routers could easily become the bottleneck of the Video-on-Demand (VoD) traffic.

To tackle this problem, a feasibility study is carried out that moves the VoD system inside the WMN. Instead of deploying dedicated servers and attaching to the mesh routers, which could easily become another bottleneck, the video data are distributed to all mesh routers. This approach turns the mesh routers into VoD servers that the VoD clients can be served by different mesh routers.

The proposed system is based on [97]. The system mainly consists of three components: a video block replication scheme, a video block discovery scheme, and three enhancements on video block transmission.

The replication scheme is based on [97] which is developed for the Home-to-home Online (H2O) devices. The scheme divides a video into blocks equally and distributes the copies of those blocks to the routers in the network evenly. The required copies of each video block decrease with its timing order, i.e., fewer copies are needed for video blocks in higher order. Each router only stores a portion of a video.

The discovery scheme is to search the video blocks from the network.

The replication scheme can save system storage. However, the mesh routers holding the sparse video blocks could become one of the bottlenecks of the service. To tackle this issue, different transmission strategies are considered that introduce three possible enhancements, named Video Block Broadcasting, Peer-to-Peer, and Time-Shifting. These enhancements aim to better utilize the bandwidth of the routers so that more clients can be served by the routers concurrently. A simulation model is built to study the performance gain of each enhancement in terms of blocking probability.

The rest of this chapter is organized as follows: Section 3.1 describes the basic elements of the proposed VoD system. Section 3.2 introduces the enhancements. Section 3.3 presents the simulation results that evaluate the performance of system against each enhancement in terms of blocking probability. Section 3.4 is the conclusion.

3.1 Proposed System Architecture

In this section, the network topology is first described. Then, it shows how the video blocks are distributed to the mesh routers, and how a mesh router gathers the missing video blocks from the network.

3.1.1 Network Topology

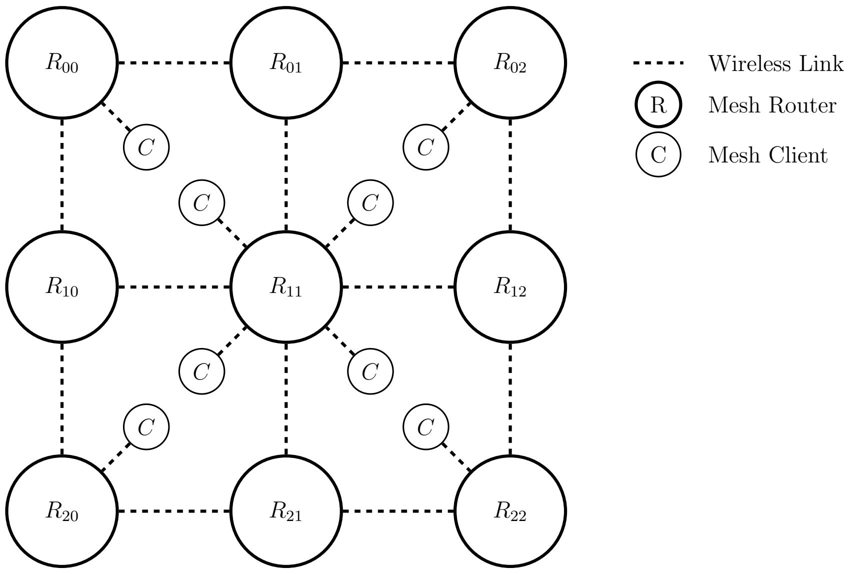


Figure 3.1: Mesh router and client in grid

The grid topology as shown in Figure 3.1 is adopted in our proposed system. It consists of N mesh routers and some mesh clients. The mesh routers are evenly distributed in a square area and the location of each router is fixed during the simulation. Each mesh router can only communicate with the neighboring routers in cardinal directions. It means a mesh router normally has four neighboring routers, i.e., at the top, bottom, left and right, except those routers on the edge of the topology. Moreover, every router holds a portion of video data in its storage. The mesh clients are mobile devices which connect to its nearest router and request for the VoD service. It is assumed that the mesh clients would stay inside the radio range of the connected router when using the VoD service.

3.1.2 Video Block Replication Scheme

Instead of deploying dedicated VoD servers in a mesh network, the video data is distributed to the storage of the mesh routers. Moreover, the replication

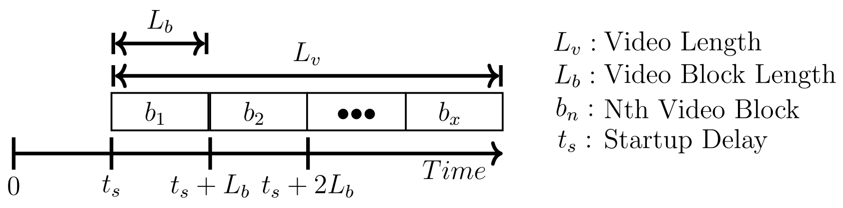


Figure 3.2: Video block model

scheme in [97] is adopted to reduce the storage requirement. Briefly, suppose a video has a length of L_v seconds. The video is first divided into x blocks equally as shown in Figure 3.2. The first video block, b_1, is copied to every router to reduce the startup delay, t_s. The remaining blocks are distributed to the routers which ensure each router can acquire the nth video block before the time $(n - 1) \times L_b + t_s$ if a mesh client requests a VoD service at time 0.

In the simulation model, the nth video block is copied to the routers every $n - 1$ hop(s). For example, the first video block will be copied to every mesh router; the second video block will be copied to routers every one hop and so on. As a result, fewer copies are needed for a video block in higher order.

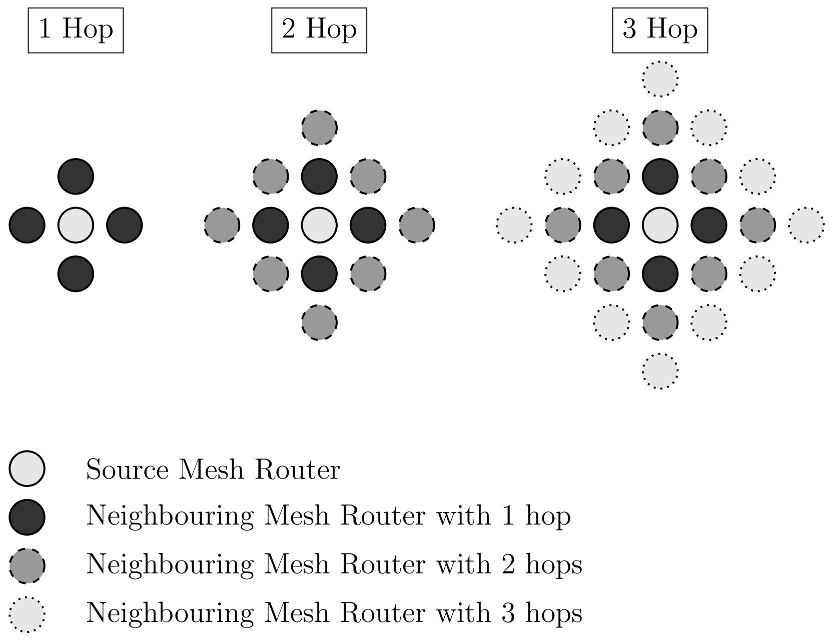

1 Hop
2 Hop
3 Hop
Source Mesh Router
Neighbouring Mesh Router with 1 hop
Neighbouring Mesh Router with 2 hops
Neighbouring Mesh Router with 3 hops

Figure 3.3: Total number of mesh router within N hops

The total number of mesh routers within $n-1$ hop(s) in the grid topology is equal to $2(n-1)^2+2(n-1)+1$ as illustrated in Figure 3.3. For example, there are five mesh routers, which include the source mesh router, in the distance of one hop. Therefore, the minimum number of replicas of nth video block is equal to $r_n = \left\lceil \frac{N}{2(n-1)^2+2(n-1)+1} \right\rceil$ where N is the total number of mesh routers in the grid topology and $n \geq 1$. Then, the minimum storage required by the video is equal to $\sum_{n=1}^{x} r_n$ where x is total number of video blocks of the video.

3.1.3 Video Block Discovery Mechanism

Suppose a video is equally divided into blocks; these blocks are properly scattered over the storage of mesh routers. When a mesh client connects to its nearest mesh router, the client sends a request for the video to the router. Once the router receives the request, the router checks its local storage, generates request messages for those missing video blocks, and sends those request messages to its neighbors. These request messages should at least contain five elements which are Sequence Number (SEQ), Video Block ID, Video Block Arrival Time, Time to Live (TTL), and Timeout. The Sequence Number (SEQ) prevents a request message from looping in the network. The Time to Live (TTL) limits the number of hops that a request message can be forwarded.

When a neighboring router receives a new request message, the router first reserves a channel. If there is not channel available, the router discards the request message. Otherwise, the router searches the requesting video block in its local storage. If the requesting video block is not found, the router forwards the request message to all of its neighboring routers except the sender of the request message. If it is found, a response message will be returned along the requesting path.

After all, when the requesting mesh router receives the response messages, it sends an acknowledgement message to one of the sources mesh routers which has the fewest hop counts. The requesting stage is completed when the source mesh router receives the acknowledgement message. Then, the transmission is started according to the scheduled video block arrival time. Besides, all the corresponding reserved channels will be released if a request message is timeout and does not have any acknowledgement message. In the simulation model, a mesh client terminates the VoD service if any of the requesting video blocks is not found in the requesting stage. As a result, all the reserved channels for this client are released also.

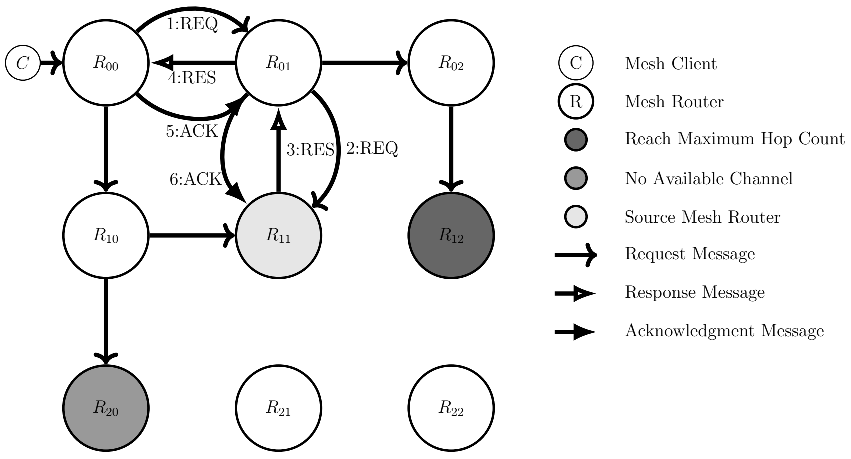


Figure 3.4: Video block discovery scheme for the video block b_4

Figure 3.4 shows an example on searching the video block b_4. At first, a client connects and sends a request for the video to its nearest mesh router R_{00}. Then, R_{00}, the requesting mesh router, searches its local storage and sends the request messages for each the missing video block to neighboring mesh routers, R_{01} and R_{10}.

Suppose one of the missing video blocks is b_4 which is distributed to the routers every three hops. The R_{00} creates the b_4 request message, sets the TTL to three, and sends it to R_{01} and R_{10}. This request message keeps flowing on the mesh network until there is not available channel of current intermediate router such as R_{20}, or the TTL is reached such as R_{12}, or the request message reaches the sources mesh router such as R_{11}. After that, the sources mesh router(s) sends a response message back to R_{00} along the path. Finally, R_{00} sends the acknowledgement to one of the sources mesh routers to confirm the transmission time.

As mentioned in Section 3.1.2, video blocks in higher order have fewer numbers of copies in the network. Therefore, as shown in the example above, it can be expected that the degree of difficulty on accessing a video block shall be generally proportional to its order. As a result, the routers which hold video block in high order could become one of the bottlenecks. To tackle this issue, different transmission strategies are considered to utilize the bandwidth of the mesh routers and improve the accessibility of the video blocks.

3.2 System Enhancements

In order to improve the services performance, three enhancements are proposed and evaluated separately. In the following, Video Block Broadcasting enhancement based on broadcast transmission is first described. Then, Peer-to-Peer and Time-Shifting enhancements based on unicast transmission are presented.

3.2.1 Video Block Broadcasting Enhancement

Video Block Broadcasting enhancement assumes each mesh router reserves channel(s) to forward video block(s) from source mesh router to all other mesh routers actively and continually. For example, suppose the first three video blocks are selected to broadcast. At first, each of these three video blocks is placed to a mesh router separately, and three channels are reserved in all mesh routers. After that, source mesh routers broadcast the video block to their neighbors. Once the neighbors receive the data, they will also broadcast to their neighbors, and so on. It makes these three video blocks available at

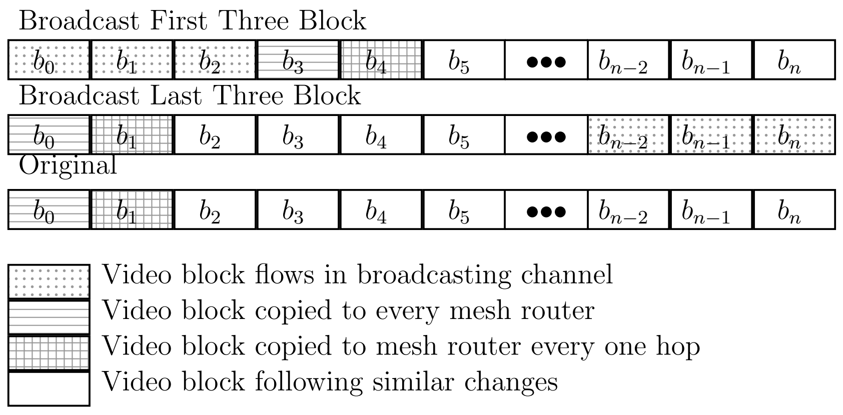

Broadcast First Three Block
b_0 b_1 b_2 b_3 b_4 b_5 ●●● b_{n-2} b_{n-1} b_n
Broadcast Last Three Block
b_0 b_1 b_2 b_3 b_4 b_5 ●●● b_{n-2} b_{n-1} b_n
Original
b_0 b_1 b_2 b_3 b_4 b_5 ●●● b_{n-2} b_{n-1} b_n
Video block flows in broadcasting channel
Video block copied to every mesh router
Video block copied to mesh router every one hop
Video block following similar changes

Figure 3.5: Video block broadcasting enhancement

anytime anywhere by sacrificing three free channels.

Moreover, as illustrated in Figure 3.5, the video block placement is modified accordingly when the Video Block Broadcasting enhancement is used. For example, suppose a video is divided into 15 blocks. When the first three video blocks are selected to broadcast, video block b_4 is copied to every router, video block b_5 is copied to the router every one hop, and so on. The distribution of the rarest video block, b_{15}, is changed from every 14 hops to 11 hops. When a client is downloading the first three video blocks, no additional channel is needed. Channels are only reserved to transmit the video blocks starting from b_4 to b_{15}. When the last three video blocks are selected to broadcast, the rarest video block is changed from b_{15} to b_{12} which is also distributed every 11 hops. However, additional channels, which transmit the beginning video blocks, are needed immediately once a client requests for the VoD service.

In the simulation model, the number of channels reserved in each mesh router is equal to the number of broadcasting video blocks. It is also assumed that the total number of channels in each mesh router is much smaller than total number of video blocks. Therefore, the system can only broadcast a portion of video blocks.

3.2.2 Peer-to-Peer Enhancement

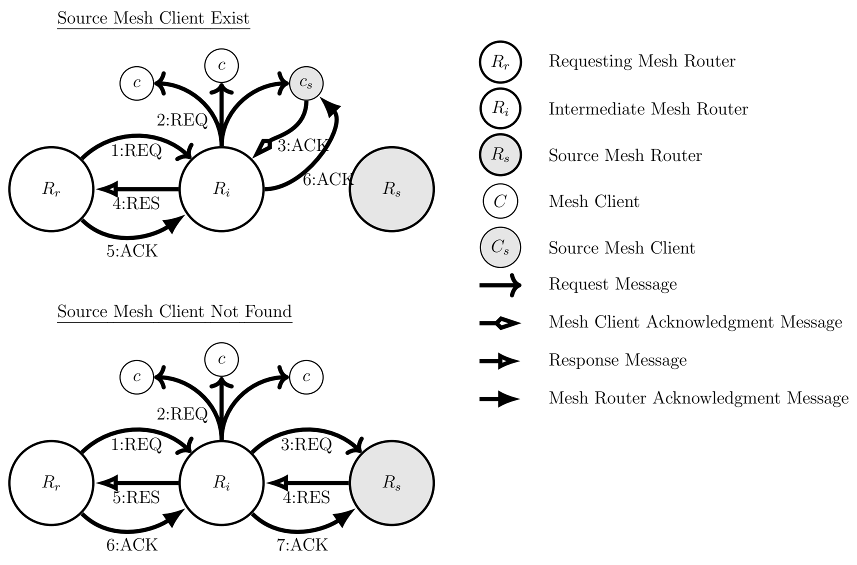


Figure 3.6: Peer-to-peer enhancement

Peer-to-Peer enhancement assumes that each connected mesh client stores a number of latest received video blocks in its buffer and acts as a source mesh client that uploads video block based on request. With this enhancement, the mesh router first searches the buffer of the connected clients before flooding the request message of the missing video block to neighboring mesh routers. For example, as shown in Figure 3.6, when an intermediate mesh router R_i receives a request message from requesting mesh router R_r, R_i checks its storage for the requesting video block as usual. Assume it cannot be found, R_i then broadcasts the request message to all connected mesh clients. If the client(s), C_s, possesses the requesting video block, an acknowledgement message is sent back to the R_i. Then, R_i returns a response message along the requesting path. However, if no client possesses the requesting video block, R_i forwards the request message to neighboring routers R_s, as described in Section 3.1.3. And finally, R_r send an acknowledgment message to R_s or C_s, if response message is received. In the simulation model, it is assumed that the source mesh client will stay in the VoD system until the upload is finished.

3.2.3 Time-Shifting Enhancement

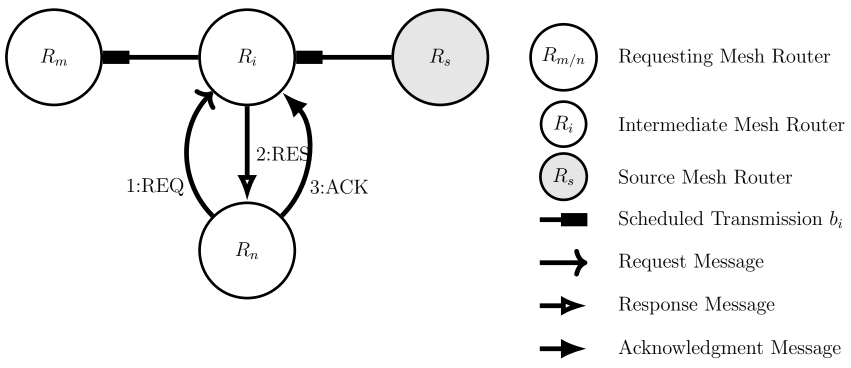


Figure 3.7: Time-shifting enhancement topology

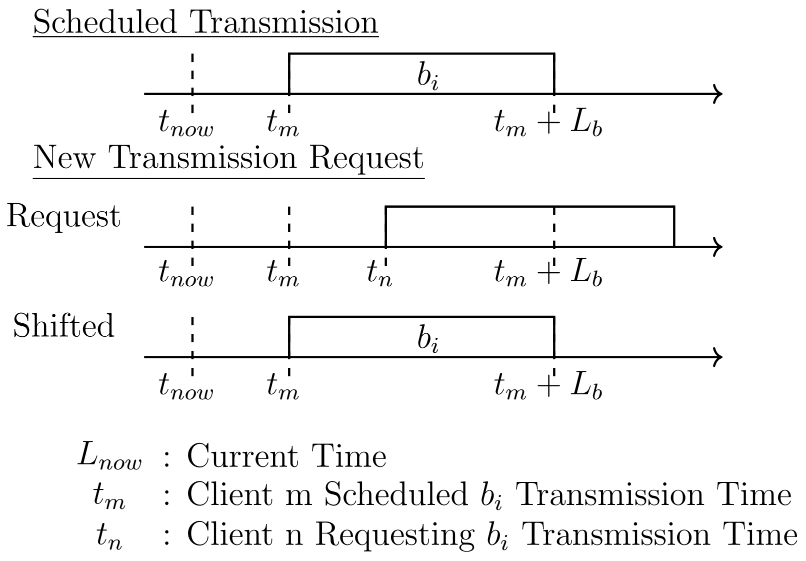


Figure 3.8: Time-shifting enhancement

In Time-Shifting enhancement, requests which arrive closely are grouped together and delivered at the same time. As illustrated in Figure 3.7 and Figure 3.8, suppose there is a scheduled transmission for video block b_i from mesh router R_s to R_m starting from time t_m to $t_m + L_b$. As an intermediate mesh router R_i, it stores this information in a table named shifting table. When R_i accepts another request from router R_n, R_i first searches the shifting table. If the requesting video block is also b_i and the requesting transmission time, t_n, falls between the scheduled transmission time t_m and $t_m + L_b$, R_i returns a response message with updated transmission time which is shifted from t_n to t_m. However, if the requesting video block is not found in the shifting table or the requesting transmission time does not match, R_i acts according to the description in Section 3.1.3. It means R_i then searches its local storage. If the requesting video block is found, a response message is returned along the path. If the video block is not found, R_i forwards the request to its neighboring routers, and so on.

3.3 Simulation Result

In order to evaluate the proposed VoD system and enhancements, a simulation model is built based on C programming language. It assumes that there are N mesh routers formed a grid topology. Each mesh router can only communicate with the neighboring routers in cardinal direction, and has a number of channels which transmission rate is equal to the playback rate of the video as described in Section 3.1.1. The video is divided into blocks equally and distributed to the storage of the mesh routers according to the methods described in Section 3.1.2 and Section 3.2.1. Each client asks for the video after it connects to its nearest mesh router. The request pattern is modeled as the Poisson process. The mechanism of video block discovery follows the description in Section 3.1.3, Section 3.2.2, and Section 3.2.3. Connected clients watch the video if it is confirmed in the requesting stage that all the video blocks are available in the desired transmission time. Otherwise, clients quit the VoD service, which are called blocked.

3.3.1 Video Block Broadcasting Enhancement

In this set of simulations, there are 400 mesh routers in total. Each mesh router equips 10 channels for transmission. The length of video is two hours.

The video is divided into 15 blocks equally and distributed according to the method described in Section 3.1.2 and Section 3.2.1.

It is expected that the system performance will be improved by adopting the Video Block Broadcasting enhancement since it improves the availability of the rare video blocks by scarifying free channels. However, the improvement should be decreased with the increase of arrival rate. It is because the bottleneck shall be shifted from the routers which hold the rare video blocks to the total number of free channels available in the VoD system.

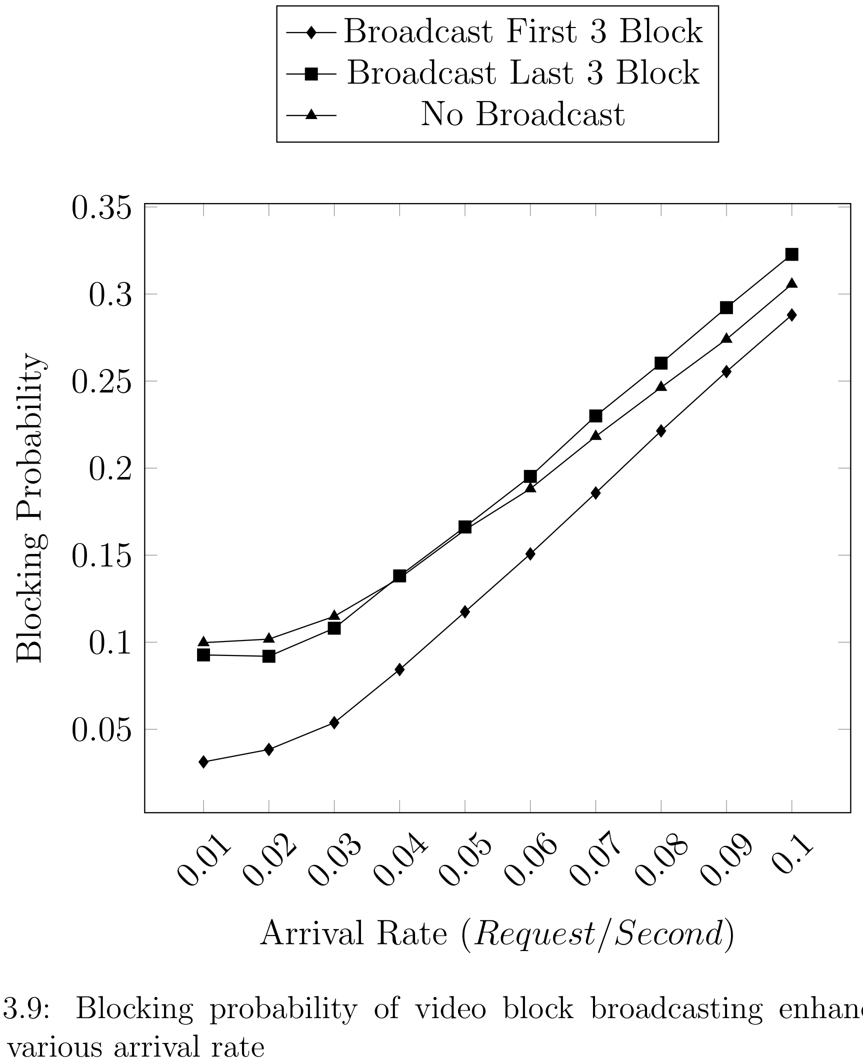


Figure 3.9: Blocking probability of video block broadcasting enhancement against various arrival rate

Figure 3.9 shows the blocking probability against various arrival rates in three different scenarios which are broadcasting the first three video blocks, broadcasting the last three video blocks, and no broadcasting. Result suggests that

1. Broadcasting the first three video blocks has lower blocking probability than broadcasting the last three video blocks.
2. The use of Video Block Broadcasting enhancement does give lower blocking probability but the improvement does decrease with the increase of arrival rate and finally deteriorate the system performance.

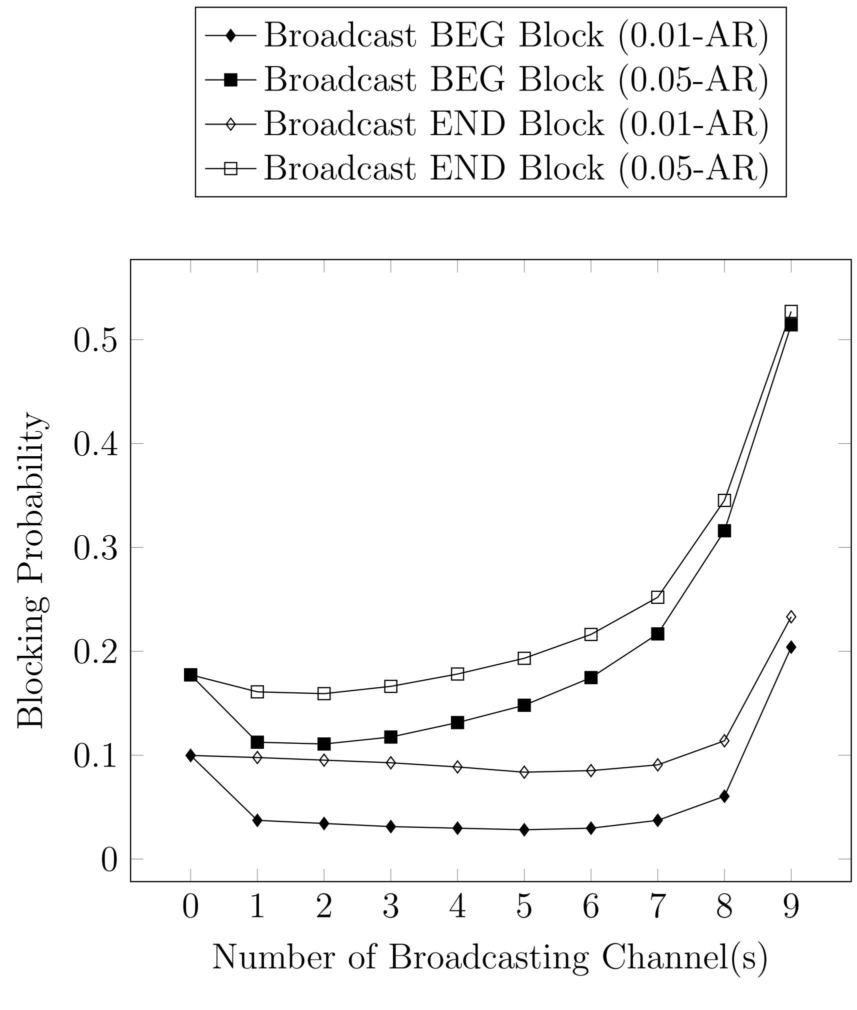


Figure 3.10: Blocking probability of video block broadcasting enhancement against various number of broadcasting channel(s) under 0.01 and 0.05 Arrival Rate (AR)

In order to study the Point 1 further, another simulation is carried out and the result is in Figure 3.10. It shows the blocking probability against various number of broadcasting channels in two different arrival rates. It should be noted that the number of broadcasting video blocks is equal to the number of channels used for broadcasting, and the total number of channels in each mesh router remains 10.

As shown in Figure 3.10, broadcasting beginning blocks always give lower blocking probability than the one broadcasting ending blocks under the same arrival rate and number of broadcasting channel. The result suggests that broadcasting beginning blocks is better than the one broadcasting ending blocks in current simulation model.

Figure 3.11 is the result of another simulation which is to study Point 2. Figure 3.11 shows the blocking probability against various number of channels equipped in each mesh router. In this simulation, half of channels in each mesh router are reserved for broadcasting and the rest are used as usual.

Figure 3.11 shows that the Video Block Broadcasting Enhancement can improve the blocking probability when mesh routers have greater number of channels. For example, an improvement in blocking probability is observed when total number of transmission channels reaches 8 at 0.01 arrival rate and 12 at 0.05 arrival rate. In conclusion, in order to obtain better system performance with Video Block Broadcasting, the following conditions are required:

1. Broadcasting the video blocks in lower order.

2. A substantial number of channels equipped in mesh routers.

3.3.2 Peer-to-Peer and Time-Shifting Enhancement

In this set of simulations, there are 400 mesh routers in total. The length of video is two hours. The video is divided into 30 blocks equally and distributed according to the method described in Section 3.1.2.

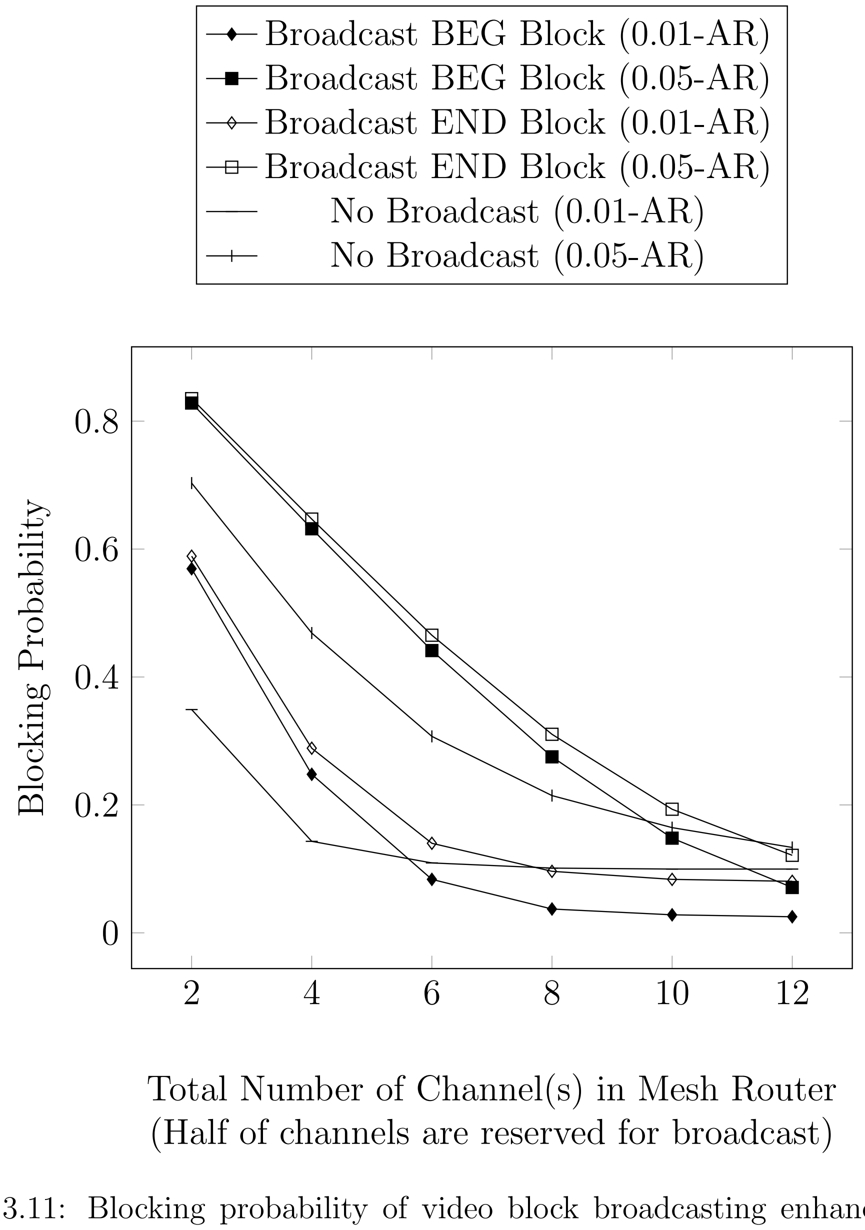


Figure 3.11: Blocking probability of video block broadcasting enhancement against various number of channel(s) equipped in mesh router under 0.01 and 0.05 Arrival Rate (AR) (Half of channels are reserved for broadcasting)

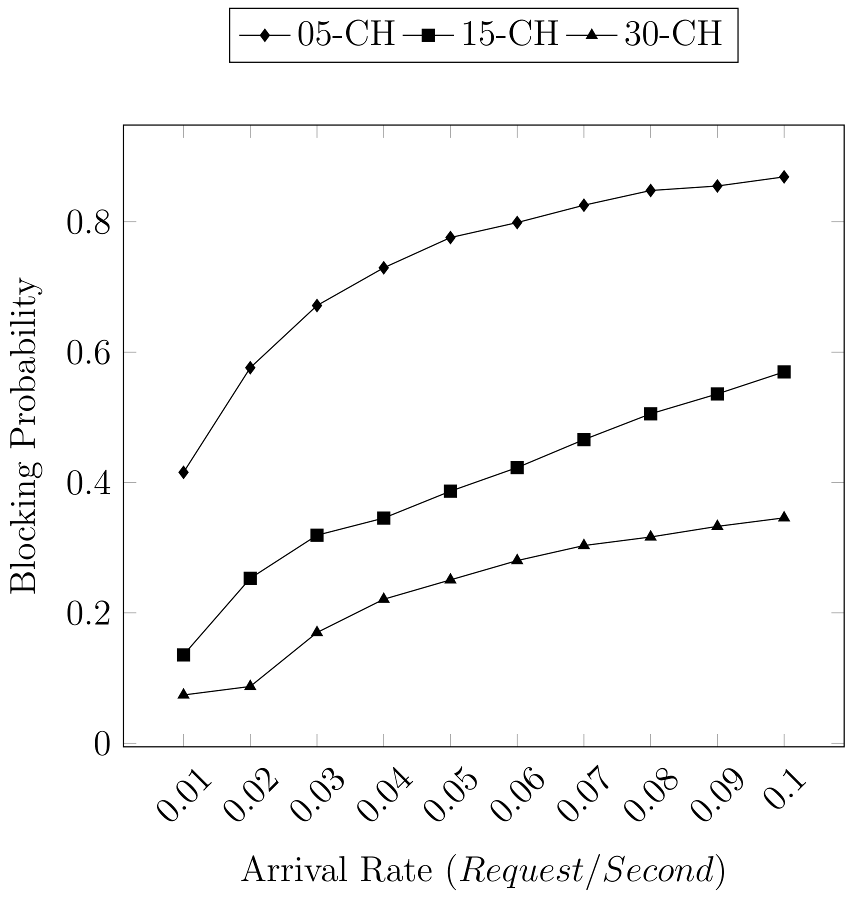


Figure 3.12: Blocking probability (without enhancement) against various arrival rate

Figure 3.12 shows the blocking probability against various arrival rates. In general, the blocking probability increases with the increase of arrival rate but decreases with the increase of number of channels. This result will be used to compare the simulation results of Peer-to-Peer and Time-Shifting enhancement in the following sections.

Peer-to-Peer Enhancement

It is expected that the Peer-to-Peer enhancement could improve the system performance and the extent of improvement shall be proportional to the number of video blocks stored in mesh clients.

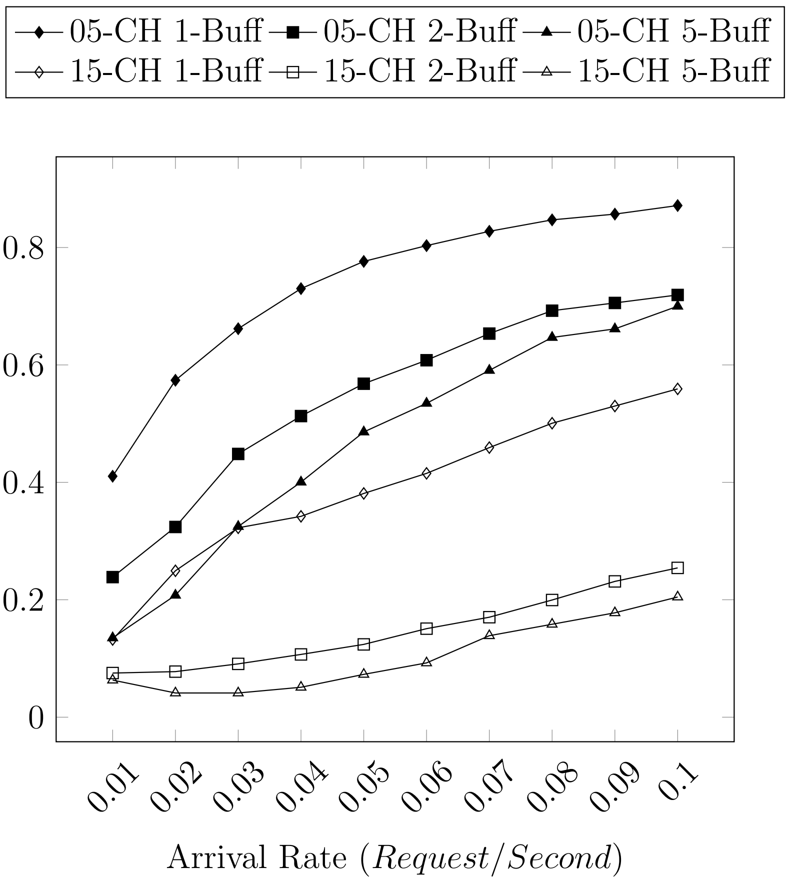


Figure 3.13: Blocking probability of peer-to-peer enhancement against various arrival rates under different number of Channels (CHs) and client Buffer (Buff) size

Figure 3.13 shows the blocking probability against arrival rates under various channels and client buffer sizes. Compared with Figure 3.12,

1. The improvement in blocking probability is obvious when the size of buffer is increased from one to two blocks.

2. The extent of improvement in blocking probability decreases with the increase of client buffer size.

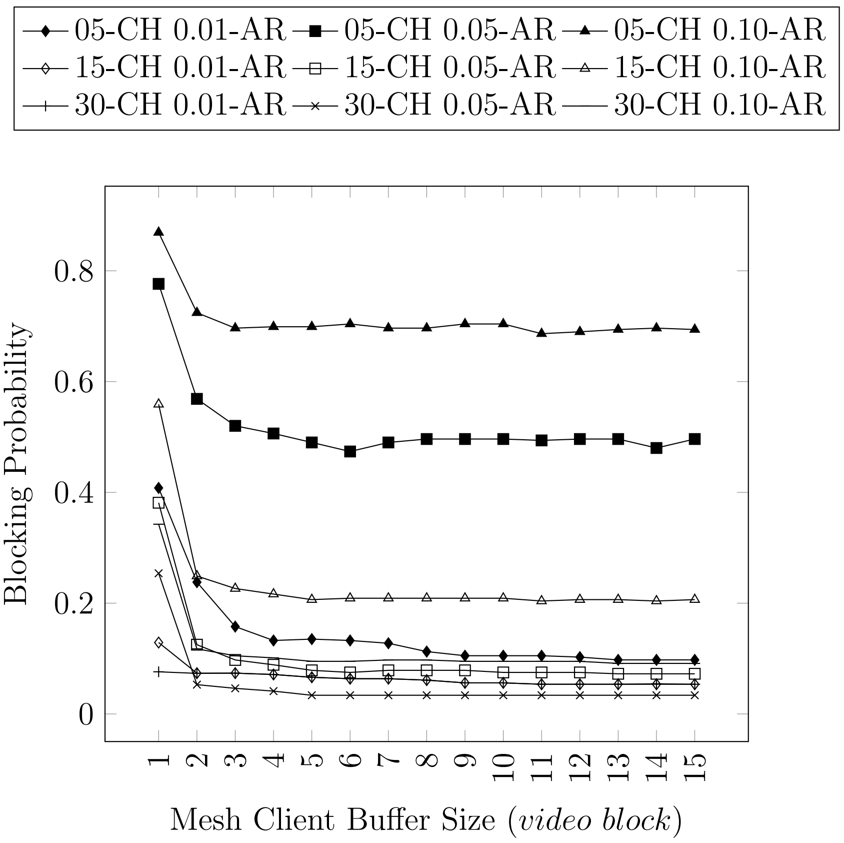


Figure 3.14: Blocking probability of peer-to-peer enhancement against various client Buffer (Buff) size under different number of Channels (CHs) and Arrival Rate (AR)

Figure 3.14 shows the blocking probability against various client buffer sizes under different number of channels and arrival rates. This result further indicates the improvement on blocking probability is negligible when the client buffer size increases beyond certain limit. For example, when the number of channels and arrival rate is equal to 5 and 0.01 respectively, the improvement on blocking probability is negligible if the client buffer size is greater than 4.

Time-Shifting Enhancement

It is expected the Time-Shifting enhancement could improve the system performance as it effectively reduce the loading of upstream router by putting requests of same video block in a group and delivering at the same time.

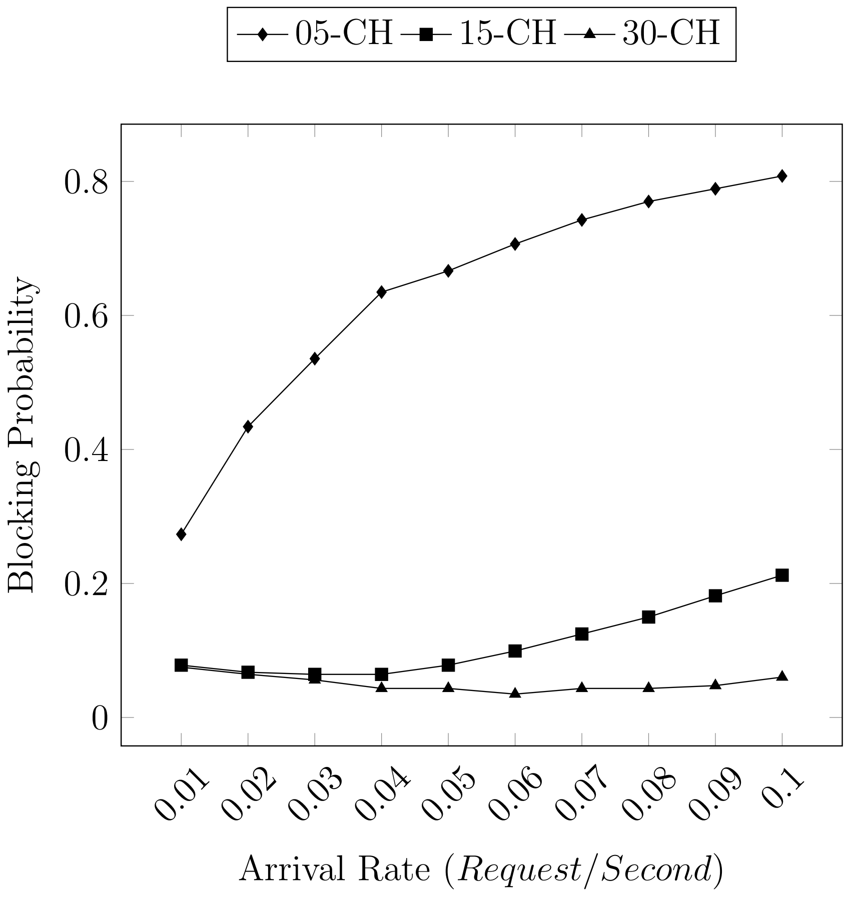


Figure 3.15: Blocking probability of time-shifting enhancement against various arrival rates under different number of Channels (CHs)

Figure 3.15 shows that the blocking probability against arrival rate in different number of channels. Compared with Figure 3.12, an obvious improvement is observed. The improvement increases with the number of channels equipped in the mesh routers.

3.4 Conclusion

In this chapter, a feasibility study on supporting the VoD system over WMN is carried out. The VoD system consists of a video block replication scheme, a video block discovery scheme, and three enhancements.

The video block replication scheme adopts the idea from [97] in order to reduce the storage requirement. Briefly, the video is divided into blocks equally. The nth block is distributed to every router $n-1$ hop(s). The necessary copies of each video block gradually decrease with its timing order, and each mesh router only holds a portion of the video.

The video block discovery scheme is to acquire all the missing video blocks from the mesh network.

To tackle the bottleneck introduced by the replication scheme, different transmission strategies are considered which are named Video Block Broadcasting, Peer-to-Peer, and Time-Shifting. According to the simulation results, Peer-to-Peer and Time-Shifting are suitable for all situations while Video Block Broadcasting can only be applied to the VoD System which already has substantial number of free channels. As a result, this motivates us to further study the potential on combining these enhancements for building a large-scale VoD system.

Chapter 4

A Hybrid PB-P2P Video-on-demand System

The factors such as fluctuation of user requests and increase in video quality make Client-Server (CS) model hardly be a robust and cost-effective solution for delivering video data as mentioned in Section 1.1. As a result, this chapter proposes and studies a Video-on-Demand (VoD) system which is a hybrid of Periodic Broadcasting (PB) and Peer-to-Peer (P2P) mesh pull. The system aims at minimizing the startup delay, maintaining smooth *playing*, and supporting various interactive commands at best effort.

PB and P2P are two building blocks commonly found in the proposals of large-scale VoD system. However, each of these building blocks has its limitations. For example, PB has limited supports on interactive commands while P2P cannot guarantee the maximum of the startup delay and playback delay. Therefore, this motivates us to investigate whether we can take the strengths and complement the weaknesses of each other if we bring the PB and the P2P together.

Various studies [7, 18–22] show that the popular videos account for vast majority of requests, i.e., the 2% of the videos account for 48% of requests found in the study of TeliaSonera [7]. Inheriting the strengths from the PB and the P2P, the proposed system can serve huge number of requests with minimum requirements on the server and has high resistance to the flash crowd. These makes the system suitable for delivering the popular videos.

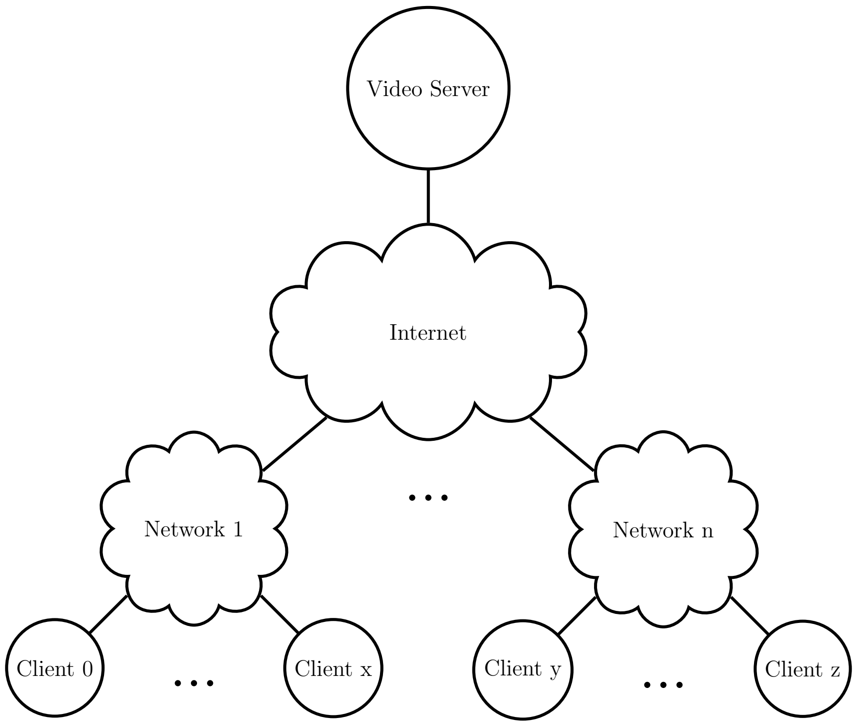


Figure 4.1: Network topology of the proposed system

In the proposed system, as illustrated in Figure 4.1, server uploads the video to channels periodically and clients download video data by switching among these channels. Meanwhile, clients exchange messages and pull video data from each other in the same network. Scheduling algorithms are developed for clients to download the video data from both channels and peers simultaneously and seamlessly under various conditions. The network in Figure 4.1 can be a physical or logical network. It defines the boundary of P2P traffic that ensure the P2P traffic would not flow to the other networks. For example, as shown in Figure 4.1, Client 0 in Network 1 would not receive the P2P message sent by client y and z in Network n.

To study the performance of the proposed system, computer simulations have been conducted in ns-3 [82,85]. Simulation results show PB and P2P are a good complement to each other for a robust and cost-effective VoD services.

The rest of the chapter is organized as follow: Section 4.1 defines the terms used in the rest of sections. Section 4.2 describes the proposed VoD System generally, and its implementation details are in Section 4.3. Section 4.4 discusses the simulation environment, topology, metrics, and parameters. Section 4.5 shows the simulation result. Section 4.6 is the conclusion of this chapter.

4.1 Terminology

To facilitate the discussion, this section introduces the terms used in following sections.

Elapsed Time The time offset from the beginning of video.

Remaining Time The time offset from the end of video.

Chunk A part of video which can be played independently.

Segment A chain of chunks.

Allocated Chunk The chunk that is allocated in the buffer and data may not be injected into the buffer.

Non-allocated Chunk The chunk that is not yet allocated in the buffer.

Injected Chunk The chunk that is allocated and data is injected into the buffer.

Non-injected Chunk The chunk that data is yet injected into the buffer.

Player Chunk / Segment The chunk / segment that is currently required by the player that may not yet be allocated or injected.

Play Point The frame in a chunk required by the player at particular moment and identified by *elapsed (remaining) time.*

First-class Chunk The chunk has higher priority that will not be replaced by other chunks in the buffer at that particular moment.

Non-first-class Chunk The chunk does not belong to first-class at that particular moment.

Chunk / Segment Length The total time for the player to consume this chunk / segment in *playing* mode.

Chunk Play Length The length of the portion of a chunk which will be processed by the player.

4.2 The Proposed PB-P2P VoD System

4.2.1 System Overview

As shown in Figure 4.2, video servers are connected to either the Internet or the network closed to clients. The servers contain all videos and multicasts the video data to corresponding channels periodically. Clients download and play the video data by switching over these channels and from other clients in same network.

The following gives detailed descriptions on each part of the system.

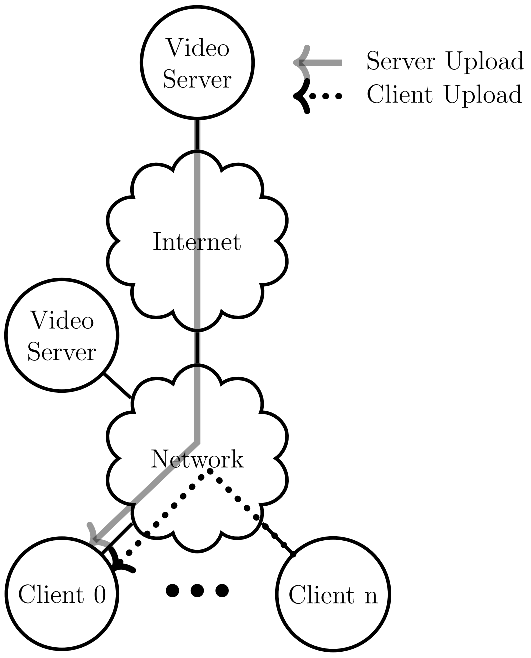


Figure 4.2: Proposed topology

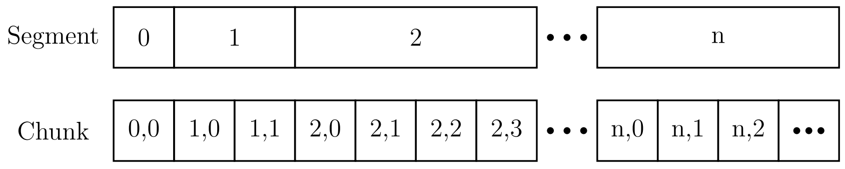


Figure 4.3: Video data model

4.2.2 Video

Each video is divided into segments, as shown in Figure 4.3, which may be different in length according to the segmentation scheme in use. Each segment is then divided into a number of chunks with equal length. Each chunk can be played by the player without the presence of other chunks. The Length of a chunk is the total playing time of that chunk in *playing* mode. Same length does not imply same size in storage. As shown in Figure 4.3, a unique identifier, (a, b), is assigned to each chunk where a is the segment index started from zero and b is the chunk index started from zero.

4.2.3 Video Segmentation Scheme

Theoretically, any video segmentation scheme can be used in the proposed system provided that the player can run the *playing* mode smoothly if no interactive command is issued.

To facilitate the discussion in the following, the segmentation method of the Active Buffer Management (ABM) scheme in [29] is used in the following sections. The maximum relative size of segment is set to four and the relative segment size is:

$$f(sid) = \begin{cases} 1 & \text{if } sid = 0 \\ 2 & \text{if } sid = 1 \\ 4 & \text{if } sid \geq 2 \end{cases}$$

where *sid* is the segment index.

To keep *playing* mode running smoothly and *play point* in the middle of the buffer, in the worst case, ABM scheme requires a client to download the video data from three channels concurrently. In brief, suppose the *player segment* is *k*. If the length of the segment *k* and *k+1* are not equal, named pyramid phase, the client is required to download the segment *k*, *k+1*, and *k+2*. Otherwise, segment *k-1*, *k*, and *k+1* are downloaded, named equal segment phase.

Let's take Figure 4.4 as an example. Suppose a user decides to playback the video from the beginning at time t_1 and the buffer is empty. Chunks filled in grey color would be downloaded first from channel 0, 1, and 2. The playback starts when chunk (0,0) is downloaded. Chunk (1,0) would be ready when chunk (0,0) is finished.

As another example, suppose the buffer size is limited, the player just jumps to somewhere in segment *k*, segment *k* and *k+1* have same length, and segment *k-1*, *k*, *k+1* are not found in the buffer. In this situation, resources would be allocated to download segment *k-1*, *k*, and *k+1*.

4.2.4 Server

Server allocates channels for each video and uploads chunks in each segment to corresponding channel periodically. If a video has n segments, n number of channels are allocated as illustrated in Figure 4.4. The total upload bandwidth of server depends on the number of videos and the segmentation scheme applied to each video but not the number of clients in service.

As shown in Figure 4.5, the chunk upload cycle starts from t_{start} to t_{end}. Upload can only start after t_{start} and must finish before t_{end} that steady bit rate is not required during the upload period. The idle time, started from t_{Ready} to t_{Start}, is the time for clients to switch over channels which is depended on the underlying transmission technology.

4.2.5 Client

Client is a specialized software or hardware that retrieves video data from the PB channel(s) and P2P network according to the user input. Figure 4.6 shows

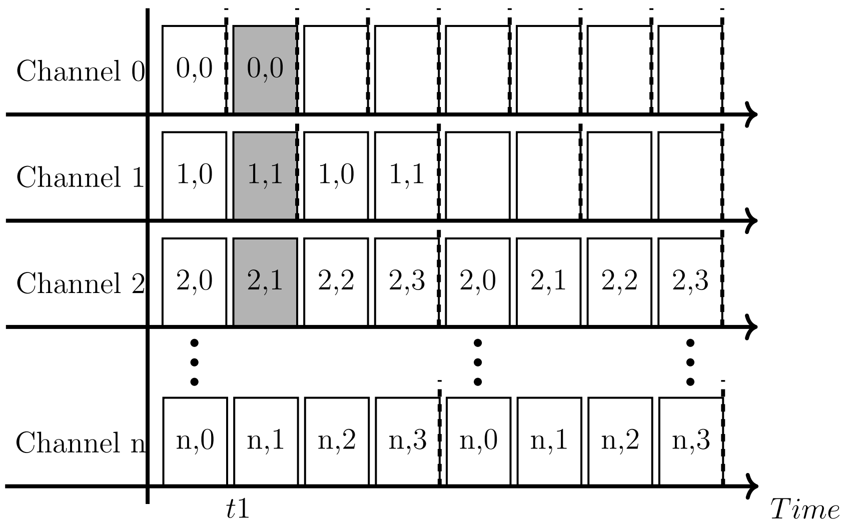


Figure 4.4: Channel model

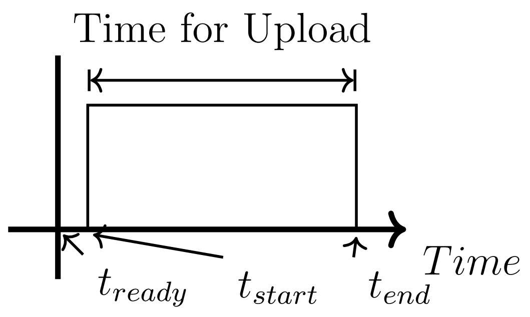


Figure 4.5: Chunk upload cycle

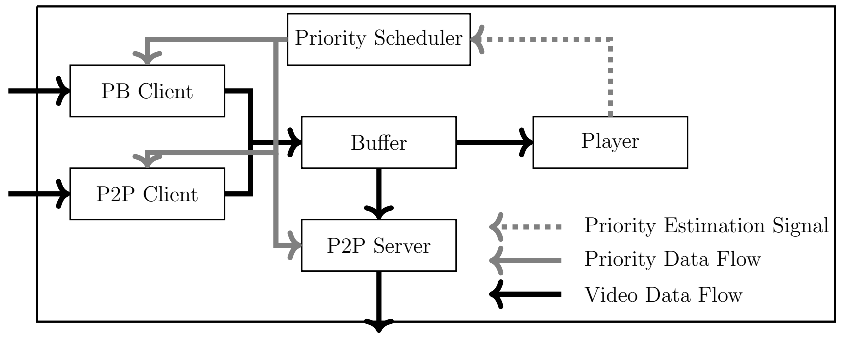


Figure 4.6: Priority and data flow in a client

the major components and their interactions in a client.

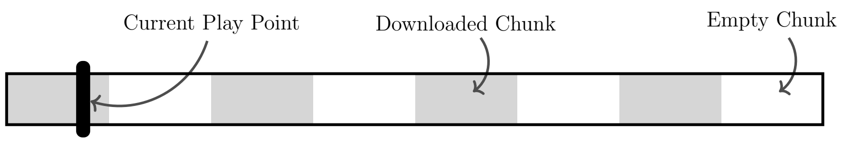


Figure 4.7: Player timeline

Player plays chunks in various modes including *Playing*, *Play Backward*, *Fast Forward*, *Fast Backward*, *Slow Forward*, *Slow Backward*, *Jump Forward*, *Jump Backward*, and *Pause* as shown in Table 4.1 in Section 4.3.3. The player timeline shows the current *play point* and display the state of chunks in the buffer. For example, as shown in Figure 4.7, grey color represents those *injected chunks* and white color represents the others. It could serve an importance reference for users when selecting player modes [93].

Apart from *playing* and *pause*, users are suggested to specify the mode's end point with the aid of thumbnail images of the video. This end point will be an important reference to estimate priority of all chunks for keeping player running smoothly especially when buffer size is limited. If the mode's end point is not provided in these modes, it would be the end (begin) of the video if the *play point* is moving forward (backward).

To determine the download order of chunks, priority scheduler is called irregularly triggered by various conditions to update the priority of all chunks. The result of estimation is then passed to interested parties such as PB and P2P client. The details of the priority scheduler would be described in Section 4.3.4.

PB client downloads chunks from the PB channels. According to the information retrieved from server, PB client can estimate the chunks available in closest upload cycle and fetch chunks based on the priority.

P2P client pulls chunks from other peers. P2P client listens the messages from P2P servers in dedicated channel(s) available in the connected network. It then processes the received messages and uses them to query the P2P servers for chunks according to the priority. The channel(s) is only known by clients in the same network. Therefore, it can contain all P2P traffic within the network.

Unlike PB client that bandwidth is restricted by the subscribed channel, P2P client may negotiate with P2P server for the transmission bandwidth.

PB client and P2P client keep downloading chunks concurrently until buffer filled with *first-class chunks*. PB client stops the download temporarily if chunks with higher priority are not available in closest upload cycle. P2P client stops the download temporarily if P2P servers in other clients offering chunks with higher priority are not available.

Making PB and P2P client work with each other is necessary since both clients download chunks in parallel. There are differences between PB and P2P download. For example,

1. PB client can only download chunks available in channels. P2P client can only download chunks advertised by P2P servers in other clients.

2. PB server keeps uploading chunks to channels in schedule. P2P server uploads chunk based on request.

3. PB server has high availability. P2P server may go offline or terminate the upload anytime.

4. There is not guarantee that a chunk is always available in other peers.

Therefore, the PB client is more reliable than P2P client that makes PB client superior on selecting chunks to download. PB client can request P2P client to terminate the download if PB client is going to download same chunk from channel. As a result, the actual download order of chunks is affected by the estimation result from the priority scheduler, the availability in PB channel(s) and P2P network, and the cooperation between PB and P2P client.

P2P server handles queries from peers and publishes chunk information to dedicated channel(s) that do not require the additional supports such as tracker.

Buffer, or cache, stores the chunks downloaded by PB and P2P client. Chunk with lowest priority is replaced by chunk with higher priority one by one if buffer cannot store the whole video.

The following briefly summarizes the steps of interactions for the scenario that a user requests to playback a video under *playing* mode.

1. Player signals the priority scheduler and try to fetch the chunk from the buffer.

2. Priority scheduler collects necessary information from various components and estimates the priority of all chunks which are then sent to the PB client, P2P client, and P2P server.

3. PB client estimates the chunks available in closest upload cycle based on the information retrieved from the server and downloads the *non-injected chunks* according to the priority.

4. P2P client listens the messages from other P2P servers in dedicated channel(s) and queries the P2P servers about the *non-allocated chunks* one by one according to the priority.

5. P2P server uses the latest priority information when composing the message for other P2P clients.

6. The chunks downloaded by PB client or P2P client are injected into the buffer. The buffer signals the player about the new chunk.

7. Player playbacks the chunks one by one according to the user input.

8. Player signals the priority scheduler again to update the priority when it is needed.

4.3 Implementation Detail

In order to study the proposed Hybrid of Periodic Broadcasting and Peer-to-Peer (PB-P2P) VoD System, the system is implemented as a ns-3 module and executed by ns-3 simulator [82, 85]. The module mainly consists of three applications: *VoD_Server*, *VoD_Network*, and *VoD_Client*. Each application contains several components. The following sections describes the abstractions and implementation details of the major components.

4.3.1 Abstraction on Chunk and Buffer

Chunks with same length can have different size in storage and transmission. The size is affected by various parameters such as video content, resolution, frame rate, compression format, network protocols, and transmission devices.

In the module, chunk does not contain real video data but a set of attributes such as unique id, length, range of *elapsed time*, range of *remaining time*, and offset from the first chunk. As a result, the size of the buffer is also measured in unit of time.

For simplicity, chunk is the smallest unit for the transmission, storage, and playback. The playback of a chunk can only be started when the data of whole chunk is injected into the buffer.

4.3.2 User Control

The user control simulates the user input. As illustrated in Figure 4.8, the user control generates and issues command to the player. The player executes

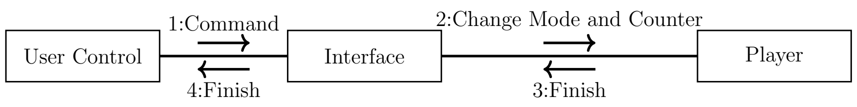


Figure 4.8: User control

the command, switches back to the *playing* mode, and signals the user control. After that the user control waits for a random period of time and issue another command.

The command has two elements: the interactive mode and the reference value. The interactive mode is one of the interactive operations shown in Figure 4.9 except *playing*. The reference value is the absolute moving distance for modes except *pause* in respective direction. If *pause* is selected, the reference value denotes the total pause time. The probability of each mode can be set explicitly. The reference value follows exponential distribution and its value step is one second.

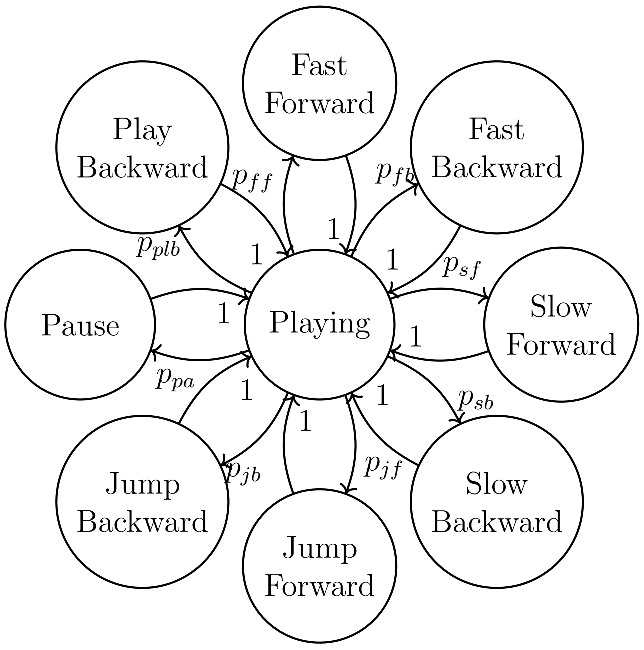


Figure 4.9: Player state diagram

4.3.3 Player

The player plays chunks and handles command from the user control. The player adjusts reference value of the received command that ensures the player will not move beyond the start or the end of the video. Invalid command, such as trying to *play backward* at the start of video, will be discarded and the user control will be notified immediately.

Table 4.1 shows the properties of the player modes. The relative speed is the playback speed relative to the *playing* mode. Positive speed means playing forward while negative speed means playing backward. Modes except *jump forward*, *jump backward*, and *pause* require chunks data to execute.

Table 4.1: Player mode

	Relative Speed	Require Chunk Data
Playing (Pl)	1	Yes
Play Backward (PlB)	-1	Yes
Fast Forward (FF)	+2	Yes
Fast Backward (FB)	-2	Yes
Slow Forward (SF)	+0.5	Yes
Slow Backward (SB)	-0.5	Yes
Jump Forward (JF)	N/A	No
Jump Backward (JB)	N/A	No
Pause (Pa)	N/A	No

The step of playback times is the timestep of ns-3 simulator, which is 1 nanosecond by default in ns-3 simulator.

Player takes one timestep to complete the *jump forward* and *jump backward*. Player takes the time, specified in the command from the user control, to complete the *pause*. For other modes, assume buffer stores all required chunks, the playback time of a chunk, t_c, is:

$$t_c = \left\lceil \frac{l_c}{s_m} \right\rceil \tag{4.1}$$

where l_c is the play length of the chunk, and s_m is the relative speed of mode as shown in Table 4.1.

The playback time of a player mode, t_m, is:

$$t_m = \sum_{t_{c_i} \in A} t_{c_i} \tag{4.2}$$

where t_{c_i} is the playback time of chunk i, and A is the set of chunks involved in this mode.

Ceiling is used to calculate playback time of a chunk as shown in (4.1). An extra timestep is added if the play length of a chunk cannot be evenly divided by the speed. Therefore, player can only load one chunk at one timestep. This keeps the player's code manageable as it is hard to manipulate the execution order of events in same timestep under ns-3. It is also the reason why ceiling is chosen instead of truncate, rounding, or accumulate the remainder.

Freeze policy is used if player reaches a *non-injected chunk*. Player temporarily holds until either the chunk is available or receives command from the user control. Player does not skip chunk which is adopted in some VoD systems such as [29].

4.3.4 Priority Scheduler

The priority scheduler would not take the rareness of chunk into consideration since the PB server ensures all chunks are available in channels periodically. No client will be demanded to download the rarest but less important chunk.

The major concern of the priority scheduler is to keep the playback running smoothly. To achieve this goal, two kinds of chunks should be downloaded first. They are

- chunks will soon be required by the player. The downloads of these chunks must be finished in time for smooth playback.
- chunks surrounds the *player chunk*. These chunks are needed to increase the resilience on interactive commands.

The priority scheduler determines the download order of all chunks so that chunks of these two kinds would be downloaded first.

The download order is computed based on offset to the *player chunk* at interactive modes and *playing* mode, and comprises of three elements, named Interactive Order (IO), Playing Order (PO), and Middle Play Point Order (MPPO). To ensure smooth *playing* and support interactive commands at best effort, IO overrides PO and MPPO. PO overrides MPPO.

IO aims to make the interactive modes running smoothly. IO only applies to the chunks involving in the interactive modes that require chunk data as shown in Table 4.1. The chunks involved and close to the *player chunk* have higher priority.

PO guarantees the smooth playback of the *playing* mode if no interactive command is issued and no chunks can be downloaded through P2P. PO applies to the selected chunks according to the video segmentation scheme in use. Moreover, PO might be adjusted dynamically to override the MPPO for various purposes such as to increase the availability of a chunk

MPPO aims to increase the resilience on interactive commands. MPPO applies to all chunks and assigns higher priority to the chunks close to the *player chunk* under the assumption that the player has slightly higher probability to move forward than backward which should be adjusted according to the user behaviour in practical environment.

Table 4.2: Download order activation condition for ABM scheme

	Pyramid Phase	Equal Segment Phase
Interactive Modes (Require Chunk)	IO, PO, MPPO	IO, MPPO
Interactive Modes (No Require Chunk)	PO, MPPO	MPPO
Playing Mode	PO, MPPO	MPPO

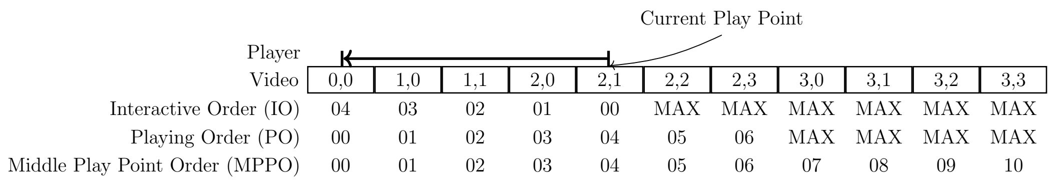
```
1: Reset priority object                                  ▷ Initial value set to MAX
2: $p \leftarrow$ player chunk
3: $l \leftarrow$ last chunk in current mode
4: $e \leftarrow$ last chunk at the end of video
5: $r \leftarrow p$                                        ▷ reference chunk
6: if Player in FF, FB, SF, SB, PlB mode then
7:     Update priority object's IO started from $p$ to $l$
8:     $r \leftarrow l$
9: end if
10: if Pyramid Phase when playing $r$ then
11:     Update priority object's PO started from $r$ to $e$
12: end if
13: Update priority object's MPPO based on $r$
14: Generate chunk priority
15: Send priority result to interested components
```

Figure 4.10: Priority scheduler algorithm for ABM scheme

The following examples illustrate how these orders are estimated. For ease of illustration, suppose the ABM scheme mentioned in Section 4.2.3 is used. The scheme partitions the video into four segments and the *chunk length* is equal to the length of the first segment. Table 4.2 summarizes the activation condition for each download order. Figure 4.10 shows the priority scheduler algorithm.

Figure 4.11 shows the download order of all chunks when the player is in *fast backward* from the center of chunk (2,1) to (0,0). The player resumes to *playing* at chunk(0,0) when *fast backward* is finished.

IO is calculated based on the absolute offset from the chunk (2,1). When

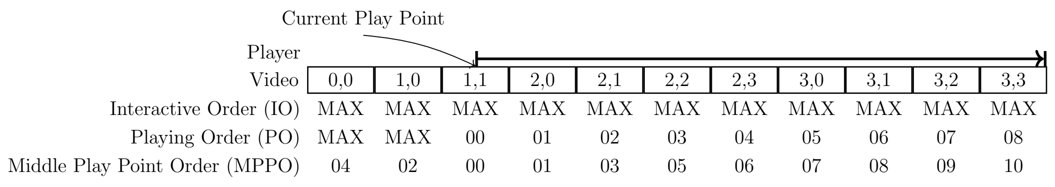


Video	0,0	1,0	1,1	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3
Interactive Order (IO)	04	03	02	01	00	MAX	MAX	MAX	MAX	MAX	MAX
Playing Order (PO)	00	01	02	03	04	05	06	MAX	MAX	MAX	MAX
Middle Play Point Order (MPPO)	00	01	02	03	04	05	06	07	08	09	10

Figure 4.11: Download order in fast backward mode

the *fast backward* is finished, the segment 0 is the *player segment.* As the segment 0 and 1 have different length (pyramid phase), ABM requires the segment 0, 1, and 2 to be downloaded first to make the *playing* mode running smoothly. Therefore, PO is activated and calculated based on the absolute offset from the chunk (0,0). MAX is assigned to the chunks do not involve in the IO and PO. MAX is a value greater or equal to the number of chunks, i.e., 10 in this example. The MPPO is calculated based on the offset from the chunk (0,0).

The download order of a chunk is determined by the value in its column as shown in Figure 4.11. For example, the chunk (2,1) has value 000404 while the chunk (3,3) has value 101010. The smaller the value, the higher the priority. Therefore, chunks should be downloaded in order from (2,1) to (0,0) and then (2,2) to (3,3).

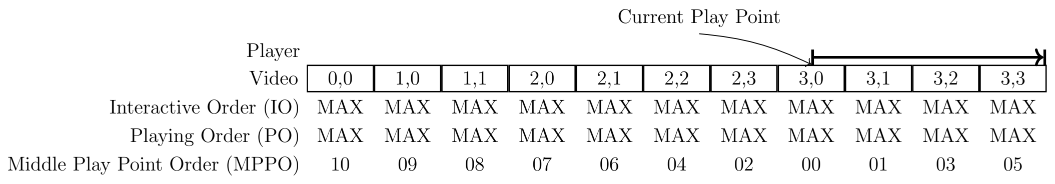


Video	0,0	1,0	1,1	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3
Interactive Order (IO)	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX
Playing Order (PO)	MAX	MAX	00	01	02	03	04	05	06	07	08
Middle Play Point Order (MPPO)	04	02	00	01	03	05	06	07	08	09	10

Figure 4.12: Download order in playing mode 1

Figure 4.12 shows the download order of all chunks when the player is in the *playing* mode from the center of chunk (1,1) to the end of (3,3).

As no interactive command is involved, IO of all chunks are set to MAX value. PO of the chunk (0,0) and (1,0) are set to MAX value since these chunks do not involve in the *playing* mode. MPPO is calculated based on the offset from the chunk (1,1). If two chunks have same absolute offset, the one close to the end of the video has lower value. The result shows that the chunks should be downloaded in order from (1,1) to (3,3), (1,0), and (0,0).



Video	0,0	1,0	1,1	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3
Interactive Order (IO)	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX
Playing Order (PO)	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX
Middle Play Point Order (MPPO)	10	09	08	07	06	04	02	00	01	03	05

Figure 4.13: Download order in playing mode 2

Figure 4.13 shows the download order of all chunks when player is in the *playing* mode from the center of chunk (3,0) to the end of chunk (3,3), which is in the equal segment phase. As mentioned in Section 4.2.3, the requirement of the equal segment phase is to download the *player segment* and segments nearby which is already fulfilled by MPPO. Therefore, PO is not activated.

The result shows that client should download the chunks according to the MPPO. In other words, suppose a client can fetch chunks from three channels and peer concurrently and its buffer is empty. The PB client joins and fetches chunks from Channel 3, 2, and 1 while the P2P client keeps searching chunks from peers following the download order. If same chunk is both available in the PB channel and neighbouring peer, the chunk would be downloaded from the PB channel. For details of cooperation between the PB and P2P client, please refer to Section 4.2.5 and Section 4.3.8.

4.3.5 Priority Update

The priority is updated when the player starts, a mode starts or completes, and the player requests another chunk. The flow on updating priority triggered by the event of mode start is shown in Figure 4.14 and the other events is shown in Figure 4.15.

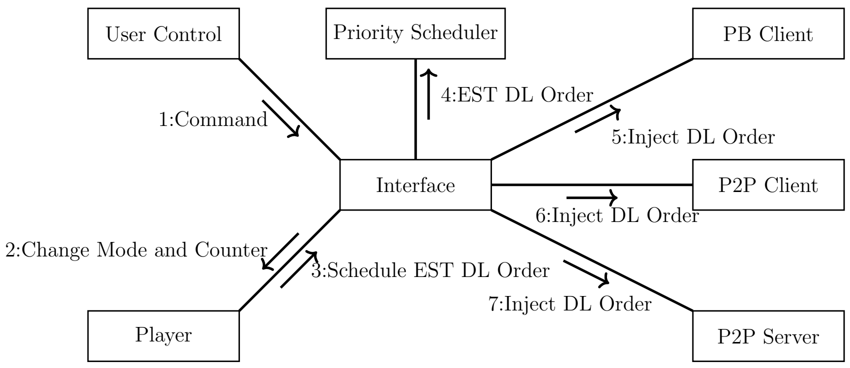


Figure 4.14: Update priority triggered by mode start

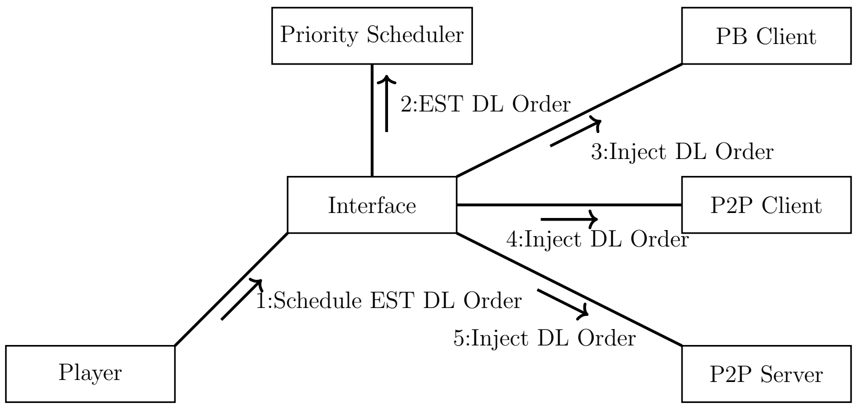


Figure 4.15: Update priority triggered by player start, mode complete, or player requests another chunk

Although the priority may be updated frequently, the update only makes PB and P2P client to start new download. The download(s) already started are not affected for several reasons.

1. It makes PB and P2P client immune from consecutive changes of player mode.

2. There is not guarantee that P2P client can download chunk with higher priority from its peers.

3. If server uploads chunks concurrently, PB client cannot download the whole new chunk by switching to another channel once started.

4. It assumes that the reference value from the user control may be relatively small compared with buffered video length which the downloading chunks are still useful.

5. Each download is short enough which makes response time acceptable.

4.3.6 PB Client

PB client uses information, retrieved from PB server at the very beginning, to estimate the chunks available in closest upload cycle in each channel. PB client creates a number of PB loaders to fetch these chunks. As the channel bandwidth is controlled by PB server, one PB loader can only reserve one channel quota.

Figure 4.16 shows the message flow for PB client to acquire chunks from channel 3. Briefly, PB client creates a PB loader for channel 3. The loader then sends join message to network, receives chunks, and finally sends leave message to network. After all, PB client deletes the loader.

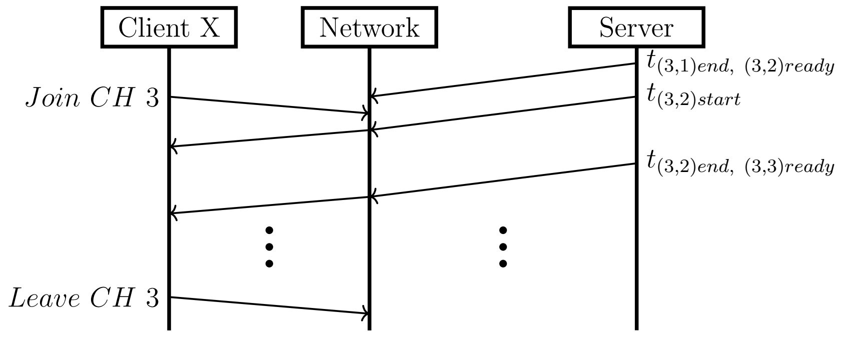


Figure 4.16: PB message diagram

4.3.7 P2P Client

P2P client collects chunk advertisements from peers by joining dedicated multicast group(s) at the very beginning. P2P client creates P2P loader to query and download chunk from P2P server in another peer.

Currently, P2P client creates only one loader at a time. The loader reserves all channel quotas assigned to the P2P client. Intuitively, the shorter the download time, the less chance to be terminated by P2P servers during transmission. Moreover, it also reduces the number of simulation parameters that reduces the complexity of the module while is enough to explorer the potential of the proposed system.

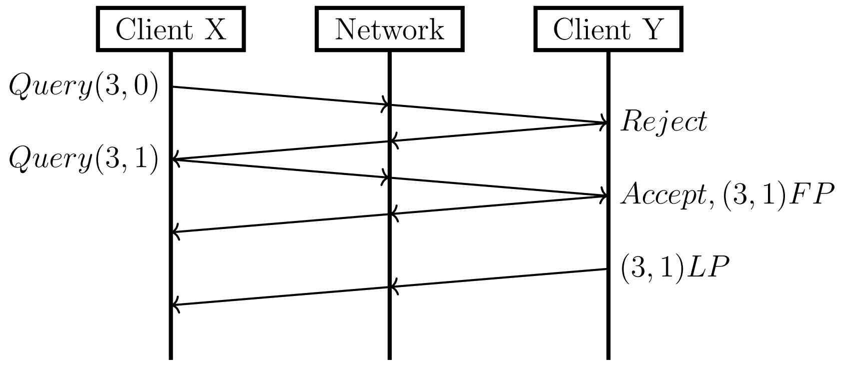


Figure 4.17: P2P message diagram

Figure 4.17 gives an example of P2P client's operation. Suppose X needs chunks (3,0) and (3,1) that are only available in Y based on the received advertisements. X first creates a P2P loader to query Y for chunk (3,0). Y rejects the query due to various reasons such as server queue is full at that moment.

X then deletes the loader and creates a new one to query Y for chunk (3,1). Y accepts the query, allocates resources, and uploads First Piece (FP) of the chunk (3,1). The download ends when X receives the Last Piece (LP) of chunk (3,1). Afterward, X deletes the loader.

The transmission can be terminated anytime by sending message to another side. It happens, for example, when Y deletes chunk (3,1) for a chunk with higher priority. Or, X decides to watch another video.

For each query, X erases the corresponding chunk advertisement. Therefore, X cannot query Y for chunk (3,0) again until X receives a new advertisement from Y. This measure protects Y from waves of query.

If multiple peers for the querying chunk are found, P2P client can select the peer randomly or based on the content of advertisement. The former method aims to give certain level of load balance. The latter method aims to select the most stable peer. It is usual that same chunk is found in multiple peers. However, it is unusual that these peers give same level of importance to this chunk. Therefore, the latter method selects the peer who has less chance to delete the querying chunk. For example, the peer who puts the querying chunk in *first-class*, lower download order, and playback the chunk in lower speed.

P2P client stores a limited number advertisements for each *non-injected chunk*. If the limit is reached, P2P client keeps the stable one. The old advertisement is erased when P2P client receives new advertisement from same peer.

4.3.8 Cooperation between PB and P2P Client

In general, according to the download order, PB client creates PB loaders for chunks which are available in closest upload cycle and have not been assigned to other PB loaders while P2P client creates P2P loaders for chunks which are found in received advertisements and have not been assigned to other PB and P2P loaders. Both PB and P2P client run until no chunks with lower download order is found.

PB client can request P2P client to terminate the download if PB client starts to download

1. same *first-class chunk*, or
2. *non-first class chunks* with higher priority and the buffer is already full.

4.3.9 P2P Server

The P2P server composes and advertises chunk information to the dedicated multicast group(s) once a new download order is received. The chunk information consists of client ID, player speed, chunk properties such as chunk ID, first-class flag, injected flag, and chunk download order.

P2P server resets a timer after each advertisement. If timer reaches zero, P2P server sends out latest advertisement again. This mechanism ensures advertisement can be sent out periodically when download order is not getting update, i.e., in *pause* mode for a long time.

Currently, P2P server can be configured to advertise chunks in one of the following kinds: *injected chunks*, *first-class chunks*, *injected* and *first-class chunks*, or all chunks.

When a client is shutdown or selects another video, P2P server rejects all incoming queries of the old video and sends out an empty advertisement that erases the advertisement stored in peers. It is because chunks of old video can be deleted at anytime.

P2P server handles queries from peers. Accepted queries are inserted into queue and served based on first-come-first-serve policy. P2P servers serves one query at one time. Details of querying process can be found in Section 4.3.7.

4.3.10 PB Server, PB and P2P Network

PB server handles channel information query, and uploads chunks to channels periodically.

The PB and P2P network manage the PB and P2P networks, handle the incoming messages, and forward to suitable nodes.

4.3.11 Channel Control

The channel control manages a limited number of channel quotas. Bandwidth reserved by one channel quota is equal to the bandwidth used by one PB channel. The P2P server, P2P client, PB server, and PB client reserve the quota(s) from the channel control for each transmission of chunk. The reserved quota(s) is returned to the channel control when the transmission is completed. The transmission time of a chunk is inversely proportional to the number of reserved channel quotas.

Currently, the time from t_{ready} to t_{end}, as shown in Figure 4.5, is equal to *chunk length*. As a result, longer idle time, from t_{ready} to t_{start}, requires higher PB server throughput. On the other hand, since P2P server does not have the idle time, time to upload a chunk is equal to $t_{end} - t_{start}$ for one channel quota.

P2P Server Chunk Transmission Time (Time from t_{start} to t_{end})

Suppose s is the chunk size in the storage, l is the chunk play length, t is the idle time from t_{ready} to t_{start}, B_1 is the bandwidth allocated to 1 channel quota, and T_n is the transmission time of a chunk if n channel quotas are allocated:

$$B_1 = \frac{s}{l-t}$$

$$\begin{aligned} T_n &= s \div (n \times B_1) \\ &= s \div (n \times \frac{s}{l-t}) \\ &= s \times \frac{l-t}{ns} \\ &= \frac{l-t}{n} \end{aligned} \tag{4.3}$$

4.3.12 Data Collection and Analysis

Event sinks are written and inserted into nearly all components. These sinks collect the data and send to the MySQL database [98] that makes querying the simulation data in run-time simple. The data includes, for example, the details on the player processing chunk; the details on the P2P client querying chunk; or, the details on the P2P server handling query.

Stored routines are called to compute the performance metrics, as shown in Section 4.4.3, at the end of simulation.

4.3.13 Visualizer

Sometimes it is helpful to visualize the simulation. Therefore, two components are written which can be enabled individually.

The first component is a python plugin written to work with PyViz [99], a live simulation visualizer came with ns-3. PyViz calls the plugin periodically.

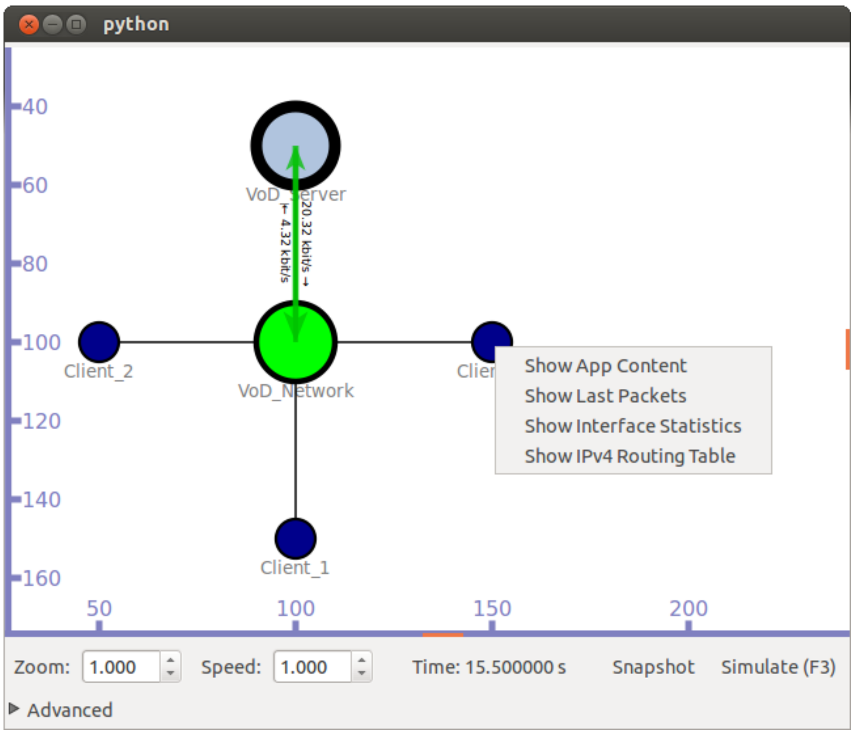


Figure 4.18: PyViz with an additional plugin

The plugin then reads application installed on each node through the Python bindings generated by API scanner [100]. As shown in Figure 4.18, the plugin

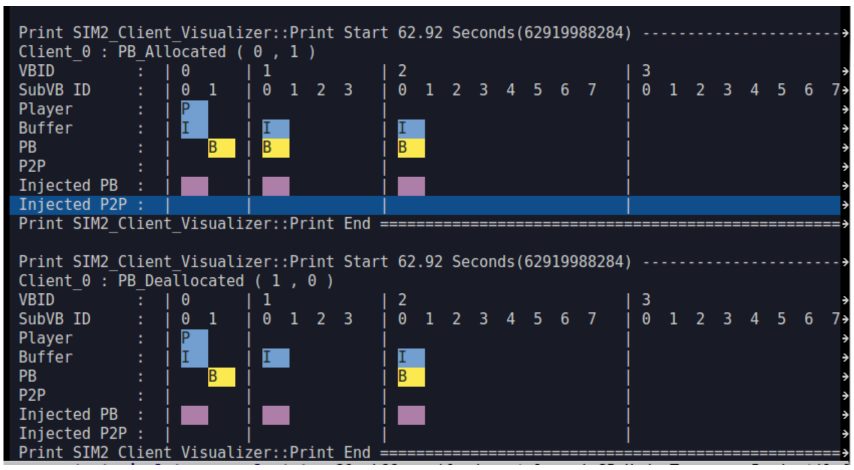
i) prints the node name under each node such as *VoD_Network*;

ii) allows each kind of application has its own color scheme for each state such as standby and working;

iii) prints the node state in terminal by mouse right click and select the "Show APP Content" option; and

iv) changes the client node size according to the total freeze time of the player. The longer the freeze time, the bigger the node size.

Another component is a text visualizer as shown in Figure 4.19. This visualizer prints the state of a specified client in terminal in color by using ANSI escape code in real time when any event sink is triggered.



```
Print SIM2_Client_Visualizer::Print Start 62.92 Seconds(62919988284) ---------------------→
Client_0 : PB_Allocated ( 0 , 1 )
VBID         :  | 0     | 1           | 2                     | 3                     →
SubVB ID     :  | 0  1  | 0  1  2  3  | 0  1  2  3  4  5  6  7  | 0  1  2  3  4  5  6  7→
Player       :  | P     |             |                       |                       →
Buffer       :  | I     | I           | I                     |                       →
PB           :  |    B  | B           | B                     |                       →
P2P          :  |       |             |                       |                       →
Injected PB  :  |       |             |                       |                       →
Injected P2P :  |       |             |                       |                       →
Print SIM2_Client_Visualizer::Print End ===============================================→

Print SIM2_Client_Visualizer::Print Start 62.92 Seconds(62919988284) ---------------------→
Client_0 : PB_Deallocated ( 1 , 0 )
VBID         :  | 0     | 1           | 2                     | 3                     →
SubVB ID     :  | 0  1  | 0  1  2  3  | 0  1  2  3  4  5  6  7  | 0  1  2  3  4  5  6  7→
Player       :  | P     |             |                       |                       →
Buffer       :  | I     | I           | I                     |                       →
PB           :  |    B  |             | B                     |                       →
P2P          :  |       |             |                       |                       →
Injected PB  :  |       |             |                       |                       →
Injected P2P :  |       |             |                       |                       →
Print SIM2_Client_Visualizer::Print End ===============================================→
```

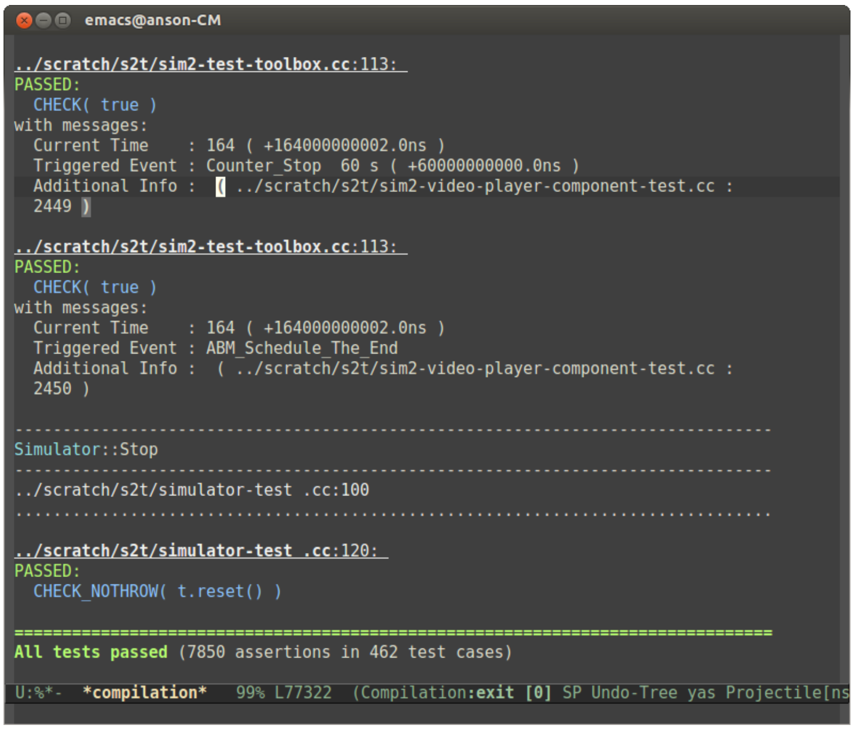
Figure 4.19: Output of text visualizer shown in emacs

4.3.14 Validation

Multiple test suites are written for each component. These test suites validate the behaviours of each component individually in specified scenarios with the aid of mock objects as shown in Figure 4.20. For example, it ensures the player fetches right chunk at right timestep at various modes; the scheduler gives right priority at various situations; the PB and P2P client allocate loader to right chunks following the download order. Test suites, including the official one, are playing a critical role especially in code refactoring, ns-3 upgrade, and changing the simulation environment. It guarantees all components are working properly.

As ns-3 logging module is disabled in optimal build, marcos are written to catch logical errors at best effort. These marcos reports the error and terminates the simulation once triggered. Marcos are deployed in the module, and *ns3::Time* and *ns3::int64x64_t* class that are used to detect the potential overflow and underflow.

Stored routines are written for database to verify the collected data. The simulation is terminated if the fields in a record are not updated in correct order or wrong value is written into a record. For example, the simulation



```
emacs@anson-CM

../scratch/s2t/sim2-test-toolbox.cc:113:
PASSED:
  CHECK( true )
with messages:
  Current Time    : 164 ( +164000000002.0ns )
  Triggered Event : Counter_Stop  60 s ( +60000000000.0ns )
  Additional Info :  ( ../scratch/s2t/sim2-video-player-component-test.cc :
  2449 )

../scratch/s2t/sim2-test-toolbox.cc:113:
PASSED:
  CHECK( true )
with messages:
  Current Time    : 164 ( +164000000002.0ns )
  Triggered Event : ABM_Schedule_The_End
  Additional Info :  ( ../scratch/s2t/sim2-video-player-component-test.cc :
  2450 )

-------------------------------------------------------------------------------
Simulator::Stop
-------------------------------------------------------------------------------
../scratch/s2t/simulator-test .cc:100
...............................................................................

../scratch/s2t/simulator-test .cc:120:
PASSED:
  CHECK_NOTHROW( t.reset() )

===============================================================================
All tests passed (7850 assertions in 462 test cases)

U:%*-  *compilation*   99% L77322  (Compilation:exit [0] SP Undo-Tree yas Projectile[ns
```

Figure 4.20: Validate the behaviours of each component through multiple test suites automatically

is terminated if delete time of a chunk is smaller than its create time in the record.

Finally, simple scenarios are conducted and verified with the aid of visualizers manually.

4.4 Simulation Setting

Computer simulations are conducted to evaluate the performance of the proposed PB-P2P VoD system and a pure PB VoD system. The following describes the details about these simulations.

4.4.1 Simulation Environment

Table 4.3 shows the specification of computer used in the simulation and Table 4.4 shows the version of packages which are not included in ns-3 prerequisites [85].

Table 4.3: Computer specification

Type	Model Name
Motherboard	Gigabyte Z97M-D3H
CPU Model	Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz
Memory Size	16 GB
Main HDD	Samsung SSD 840 250 GB
OS	ubuntu 12.04 LTS 32 bit PAE

Table 4.4: Package version

Package	Version
ns-3	3.24.1
MySQL Connector for C++	1.1.0-3build1
Boost C++ Libraries	1.46.1

4.4.2 Star Topology

Star topology, as shown in Figure 4.18, is used in the simulation. Server node and all client nodes are connected to the network node. The point-to-point channel model is used to simulate the channel between two nodes. The ns-3 Internet stack is installed to all nodes. The receive and send buffers of sockets, queue mode and queue length of point-to-point network device, bandwidth of point-to-point channel are adjusted accordingly in order to work with the channel control. *VoD_Server*, *VoD_Network*, and *VoD_Client* applications are installed to server node, network node, and client node respectively.

4.4.3 Performance Metric

A user session consists mainly of startup phase and playback phase. When the user selects a video, the startup phase starts. When the player consumes the first chunk of the video, the startup phase switches to playback phase. Both phases end when the user stops the video playback.

The performance of the startup phase is evaluated by the startup delay which is the time started the startup phase to the time switched to the playback phase.

The performance of the playback phase is evaluated by the smoothness of the video playback in term of total freeze time and frequency. When the player executes a mode required chunk data and the *player chunk* is not yet injected into the buffer, the *play point* freezes, i.e., *pause*. The total freeze time is the total time that the *play point* is frozen while the frequency is the total number of discrete freezes.

During the simulation, a new record is sent to database when the player changes mode or moves from a chunk to another. This record contains three important fields which are request time, start time, and stop time. Request time is the first moment that the player attempts to run the mode. Start time is the time that the player successfully runs the mode. Stop time is the time that the player finishes the mode or the chunk.

The freeze time is the difference between start time and request time if start time is available. Otherwise, it is the difference between stop and request time. Modes do not require chunks, as shown in Table 4.1, have zero freeze time.

The playback time is the difference between stop and start time if start time is available. Otherwise, the playback time is zero.

The followings show the definitions of the performance metrics.

Average Startup Delay

This delay indicates how long clients should wait for *playing* after the video is selected. The definition is:

$$D_s = \frac{\sum\limits_{i \in C_a} (d_{s_i})}{n} \tag{4.4}$$

where C_a is the set of all clients and n is the number of clients, i.e., size of C_a. d_{s_i} is the startup delay of client i which is the total freeze time accumulated from the first record of client i to the record which contains the smallest start time and runs in a mode requiring chunk. Clients who have never played any chunk is excluded. These clients are rare and typically found nearly the end of simulation.

Average Freeze Frequency per Client

This metric indicates the average number of freezes experienced by clients which is defined by:

$$N_{FClient} = \frac{\sum\limits_{i \in C_a} f_i}{n} \tag{4.5}$$

where C_a is the set of all clients and n is the number of clients, i.e., size of C_a. f_i is the total number of freezes experienced by client i excluding the freezes before startup.

Freeze Time Ratio

This metric shows the smoothness on playback experienced by clients. The definition is:

$$R_{FT} = \frac{\sum\limits_{j \in R_p} t_{f_j}}{\sum\limits_{j \in R_p} t_{p_j}} \tag{4.6}$$

where R_p is the records of the player of all clients excluding the records before startup. t_{f_j} is the freeze time in record j. t_{p_j} is the playback time in record j.

Average Freeze Time per Command

The metric shows the relationship between the total freeze time and the number of interactive commands. The definition is:

$$T_{FCmd} = \frac{\sum\limits_{j \in R_p} t_{f_j}}{c_t} \tag{4.7}$$

where R_p is the records of the player of all clients excluding the records before startup. t_{f_j} is the freeze time in record j. c_t is the number of interactive commands in whole simulation.

Average Freeze Frequency per Command

This metric shows the relationship between the number of freezes and the number of interactive commands. The definition is:

$$N_{FCmd} = \frac{\sum\limits_{i \in C_a} f_i}{c_t} \tag{4.8}$$

where C_a is the set of all clients. f_i is the total number of freezes experienced by client i excluding the freezes before startup. c_t is the number of interactive commands in whole simulation.

PB to P2P Download Ratio

This metric compares workloads of PB and P2P loaders.

$$R_{PB-P2P} = \frac{d_{pb}}{d_{p2p}} \tag{4.9}$$

where d_{pb} and d_{p2p} are the number of chunks downloaded by PB loaders and P2P loaders respectively in whole simulation.

4.4.4 Simulation Parameter

Several studies [7, 18–22] have performed to investigate the user behaviours of VoD systems in different aspects, each of which has different user base and different kinds of video. One of the common findings of these studies is that a large proportion of video session is quite short. The cumulative distribution function of session length reported from each study fits log-normal, exponential, or power law with different parameters. The arrival rate can fluctuate significantly in a day and Poisson process still endorse in most of these studies.

As a result, exponential random variables are used in the following simulations to generate the client arrival pattern and session length. It is assumed that if a client finishes the video before the end of session, this client will switch to another video and its P2P server will not serve the clients watching the old video.

Besides, the run number of the random seed used for each simulation is determined by hashing the simulation start time that ensures the random numbers are unique across the simulations.

Table 4.5 shows the common parameters used in the simulations. The parameters overridden will be stated in the head of each of the following subsections.

The total simulation time is the time that the first client finishes initialization to the time that the last client starts the shutdown procedures. The start sequence of nodes in the simulation is network, server, and then clients. The shutdown sequence is in the reverse order.

The study of TeliaSonera in the year of 2011 [7], a TV-on-demand service provider with 30000 active users daily, shows that the most popular program has around 5000 requests per week (0.008 arrivals per second). Moreover,

Table 4.5: Simulation parameter

Parameter	Value
Total Simulation Time	$Video\ Length \times 1.5$
Client Arrival Rate per Second	0.03
Client Lifetime Mean	$Video\ Length \times 0.7$
Video Length	160.5 *minutes*
Chunk Length	45 *seconds*
Segmentation Scheme	Scheme proposes in [29]. Number of chunks in segment: $f(sid) = \begin{cases} 2 & \text{if } sid = 0 \\ 4 & \text{if } sid = 1 \\ 8 & \text{if } sid \geq 2 \end{cases}$ where *sid* is the segment id.
Total Server Channel (Segment)	28
Client Channel Control Quota for Download	6 (Refer to Table 4.6 for the details of download quota assignment.)
Point-to-point Channel Delay	20 *milliseconds*
User Control Mean Playing Time	10 *minutes*
User Control Mean Interactive Duration	5 *minutes*
Enabled Interactive Mode	*All*
Player Interactive Mode Probability	$1/n$, where n is the total number of activated interactive mode
Player Starting Elapsed Time	0 *seconds*
P2P Server Queue Length	1
P2P Server Advertisement Mode	*Injected Chunks*
P2P Server Advertisement Timer	$Chunk\ Length + 1\ s$
P2P Client Maximum Number of Stored Advertisements per Chunk	5
P2P Client Source Select Mode	*Most Stable Source*
Client Buffer Size	36 *minutes*
PB Channel Idle Time	1 *seconds*

Table 4.6: Download quota assignment in PB scenario and PB-P2P scenario

	PB Client	P2P Client
PB Scenario	6	0
PB-P2P Scenario	3	3

the requests in the peak hour are at least four times higher than the average number of requests in a day i.e., 0.032 arrival rate per second. These data gives a brief picture on the scale of the network simulated in the following experiments when the arrival rate is configured at 0.03.

The video length is configured to 160.5 minutes which is close to the average length of top ten movies in IMDb [101], 161.7 minutes.

The minimum download quotas required by the ABM [29] is three. Therefore, three download quotas are assigned to the PB client component in both PB and PB-P2P scenarios. In order to study the differences between the conventional PB system and the proposed PB-P2P system, three additional download quotas are assigned to PB client component in PB scenario and P2P client component in PB-P2P scenario. Both components would not share the download quotas. The upload quotas of P2P server component are equal to the download quotas assigned to P2P client component. The impact on changing the download quota(s) is examined in Section 4.5.5. Table 4.6 summaries the assignment of the download quotas in PB and PB-P2P scenarios.

The mean playing time of the user control is 10 minutes. The mean interactive duration of the user control is 5 minutes which affects the reference value of the interactive commands.

The size of the buffer is 36 minutes which can store six biggest segments, i.e., $\frac{45\times8\times6}{60} = 36\ minutes$. With this setting, PB Client can download six biggest segments at the same time.

A chunk can only be played when it has been downloaded and injected into the buffer. If a user playbacks the video started from the beginning and the video data can only be downloaded from the PB client component, the maximum startup delay is

$$d_s \approx l_{s_0} + (l_{c_{00}} - t_{idle}) \tag{4.10}$$

where l_{s_0} is the length of segment 0, $l_{c_{00}}$ is the length of chunk (0,0), and t_{idle} is the idle time of the PB channel.

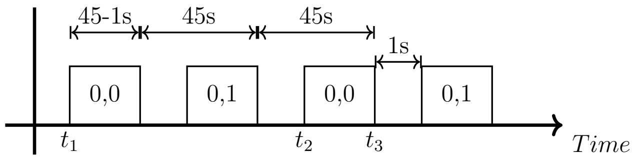


Figure 4.21: Estimate the startup delay

For example, as illustrated in Figure 4.21, suppose a user in the PB scenario playbacks the video started from the beginning using the parameters in Table 4.5, the user would experience the maximum startup delay if the PB client join the channel after t_1. The download of chunk (0,0) can only start at t_2 and finish at t_3. Therefore, the startup delay is close to $(45-1)+(45\times 2) = 134$ *seconds*.

Clients in PB scenario have longer buffering time at startup as shown in Section 4.5. This would generally improve the metrics of PB scenario except the average startup delay since the records created before the playback of the first chunk are excluded.

4.5 Simulation Result

In this section, we first study the system when only *playing* mode is involved. Then, we examine the resources demanded by different interactive commands. Finally, different system parameters such as download quota and buffer size are changed to observe the impacts to the client playback experience.

4.5.1 Playing Mode Only

The user control is disabled in this section to study the situation when only *playing* mode is involved.

Table 4.7: Only playing mode is enabled

	PB	PB-P2P
Average Startup Delay (*Seconds*)	88.96	29.02
Freeze Time Ratio	0	0.00
Average Freeze Frequency per Client	0	0.16
PB to P2P Download Ratio	NA	0.38

Table 4.7 shows the simulation results of the PB and PB-P2P. As no interactive command is involved, the metrics, average freeze time per command and average freeze frequency per command, are not applicable in this section.

As shown in Table 4.7, PB and PB-P2P both (almost) guarantee the smooth playback if no interactive command is involved as zero freeze time ratio is reported.

It is seen that the PB-P2P can further reduce the startup delay from 88.96 to 29.02 seconds. However, it also shortens the buffering time at startup that some clients may experience a little freeze when the first chunk is finished.

According to the PB-P2P simulation data, around 86.45% of freezes occur at the startup, 13.35% of freezes occur at the second chunk, and 0.21% of freezes occur at the third chunk.

The average and maximum freeze time occurred at second and third chunk is 6.29 and 22.78 seconds respectively. These freezes can be solved by adding a small delay on the player at startup or postponing the playback until the second chunk is injected.

Besides, PB to P2P download ratio is equal to 0.38 in PB-P2P that means the P2P client acquires most of the chunks under the current set of parameters.

4.5.2 Interactive Mode - ALL

This section studies the impact of various interactive commands on system performance by changing the mean interactive duration. This change affects the reference value in each interactive command that might put more loading on PB client and P2P client.

Figure 4.22a shows the average startup delay of PB is longer than the one in PB-P2P and remains steady across the simulations which agree to the results shown in Section 4.5.1.

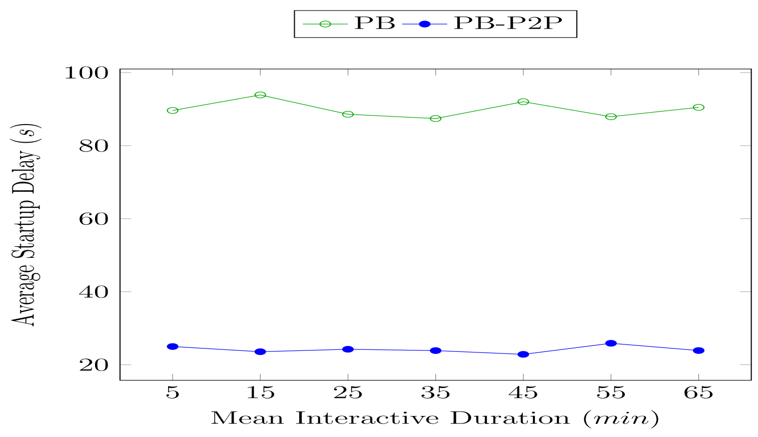
Figure 4.22b shows the increase in the duration deteriorates the performance of both scenarios. However, the PB-P2P suffers less than the PB. For example, when the duration reaches 35 minutes, the average freeze time per command is around 5 seconds in the PB-P2P compared with the 19 seconds in the PB. This result reveals the P2P can greatly enhance the playback experience of various interactive commands.

Figure 4.22c shows the difference in freezes frequency between the PB and PB-P2P is minimal.

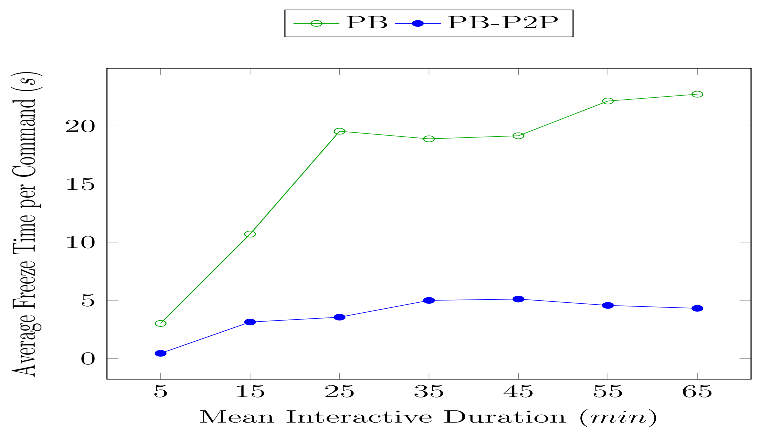
Figure 4.22d shows clients acquire more chunks from the PB channels when the duration increases. It demonstrates the importance of the PB server that provides an efficient and stable source of chunks.

4.5.3 Interactive Mode - Jumps / Fasts / Slows

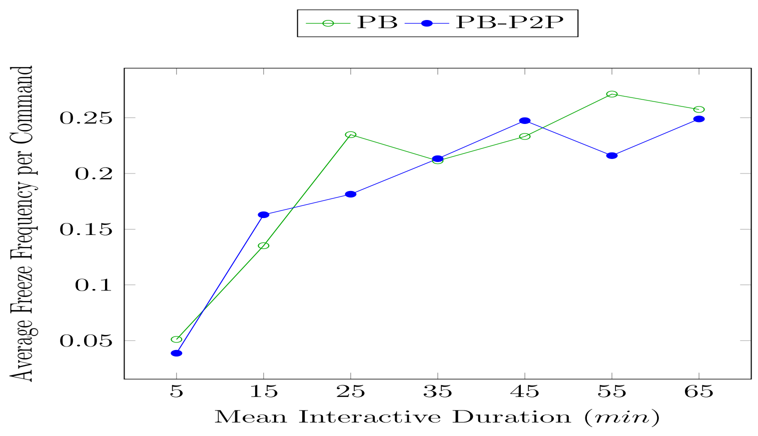
In the following, three groups of interactive modes are further examined which are 1) *jump forward / backward*, 2) *fast forward / backward*, 3) *slow forward / backward*.



(a)

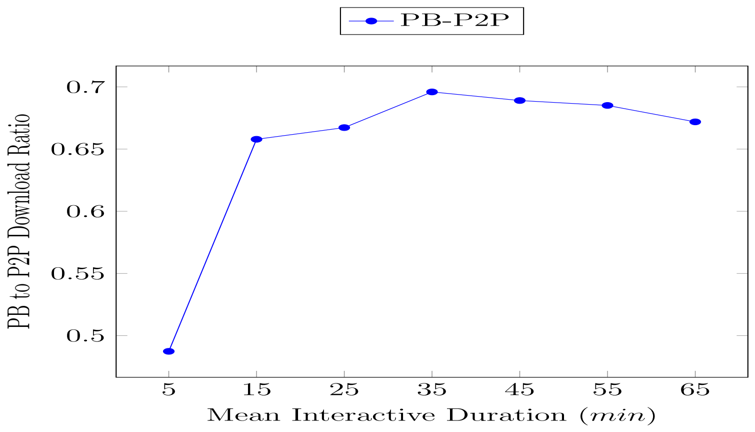


(b)

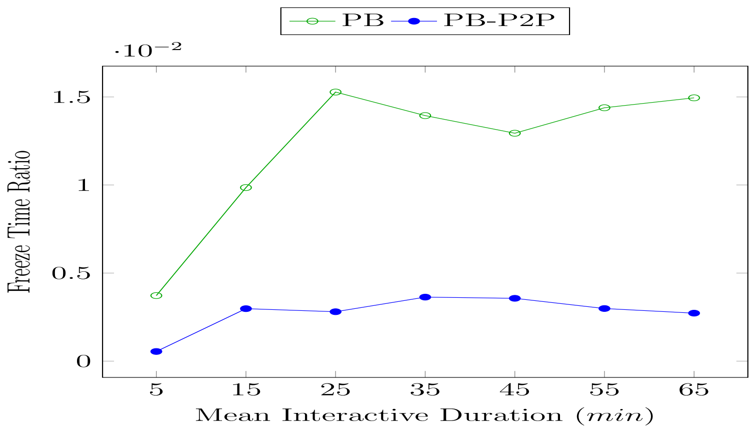


(c)

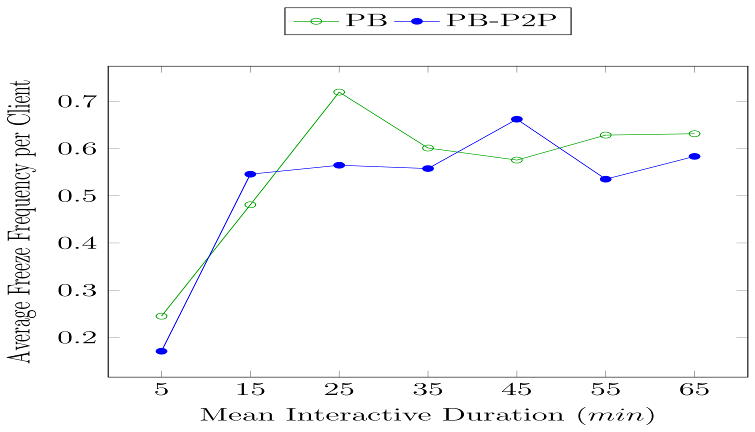
Figure 4.22: Change the mean interactive duration and enable all interactive modes



(d)



(e)



(f)

Figure 4.22: Change the mean interactive duration and enable all interactive modes

Each of these groups represents a unique characteristic of the interactive modes. The *jumps* represent the modes that change the *play point* without playback intermediate chunk(s). The *fasts* (*slows*) represent the modes that playback chunk in speed higher (lower) than the *playing* mode.

In each simulation, only one group of modes is activated. The user control can only select modes specified in the group with equal probability.

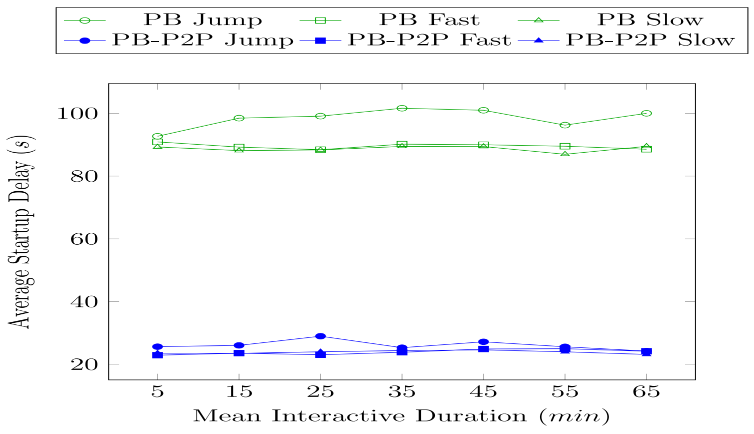
Figure 4.23a shows average startup delay remains steady across the simulations except for the cases of PB Jump in which larger fluctuations and higher delay are observed. This is because clients in PB usually has longer startup delay that more clients jump to a bigger segment before the player consumes any chunk data. The bigger the segment is, the longer the upload cycle, and hence, the longer the startup delay. A similar pattern is also observed in the PB-P2P. The PB-P2P Jump has slightly longer startup delay than the PB-P2P Fast and Slow. However, the increase is smaller than the one observed PB Jump.

According to the simulation data, when the mean interactive duration is 35 minutes, the percentage of the clients who issue the *Jump* command at the beginning of the video is 11.25% in the PB and 2.36% in the PB-P2P. The average jumping distance is 1926.89 seconds (32.11 minutes) in the PB and 2083.64 seconds (34.73 minutes) in the PB-P2P.

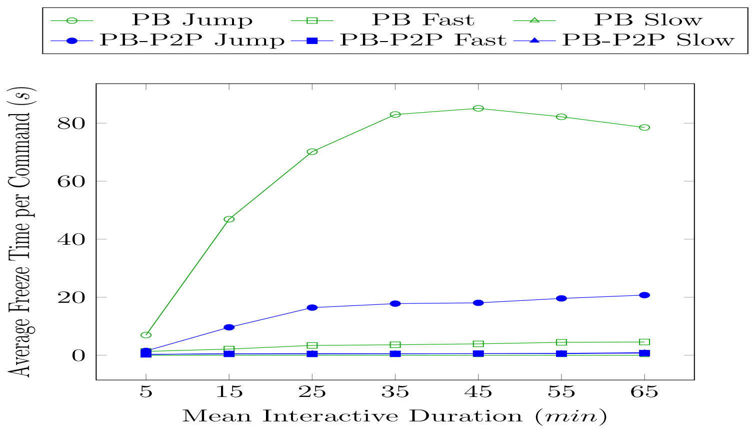
These simulation results show

1. the use of the P2P can reduce the startup delay which is a good complement of the PB, and

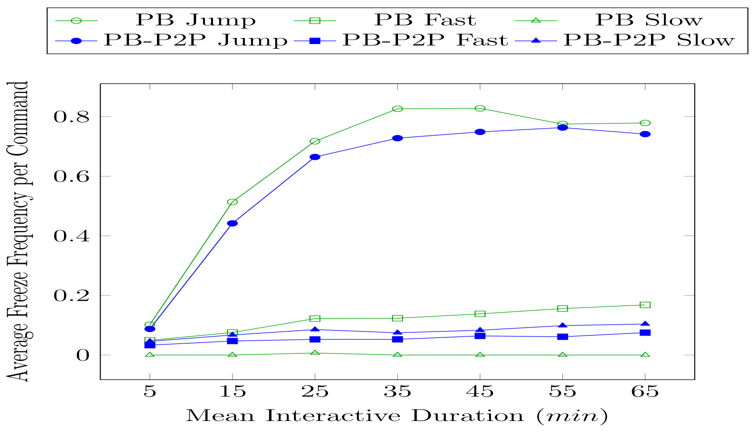
2. the traditional segmentation schemes that reduce the startup delay by minimizing the upload time of the first segment may not work well in VoD system supporting the interactive commands since clients might not play the chunks in sequential order from the beginning to the end.



(a)

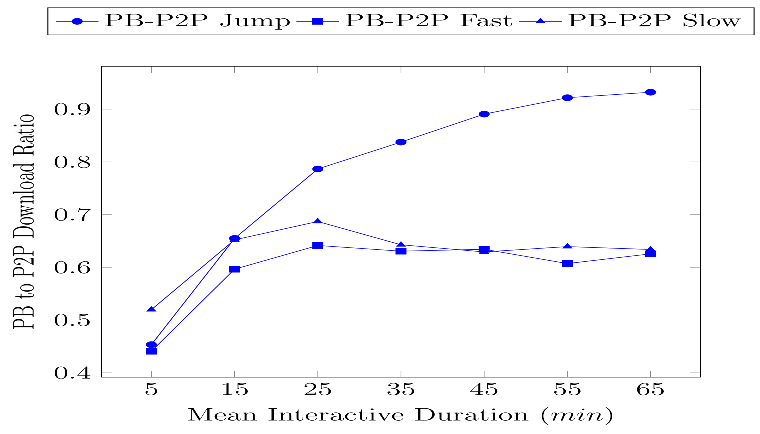


(b)

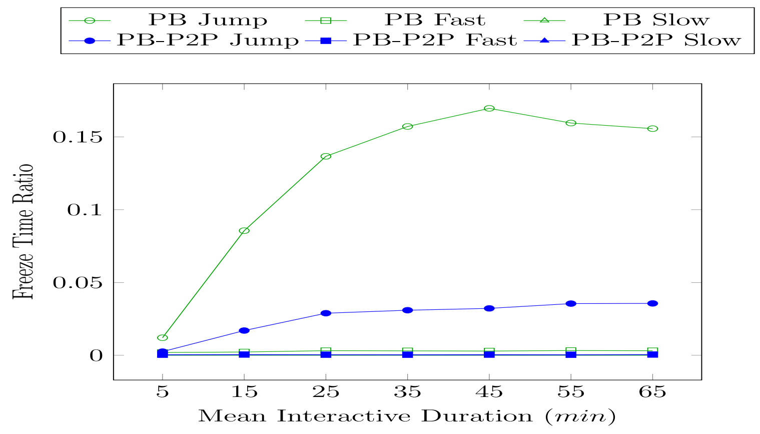


(c)

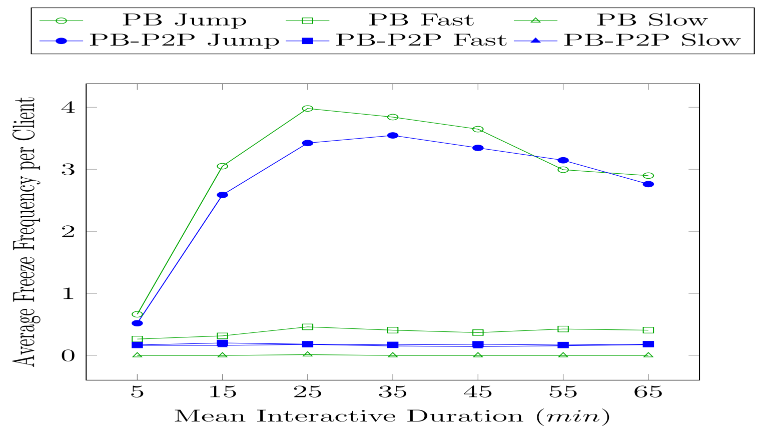
Figure 4.23: Change the mean interactive duration and enable jump / fast / slow



(d)



(e)



(f)

Figure 4.23: Change the mean interactive duration and enable jump / fast / slow

Figure 4.23b and Figure 4.23c show increase in the duration of *jumps* and *fasts* deteriorates the system performance. Clients of the PB-P2P usually have better playback experience which agrees with the results shown in Section 4.5.2. On the other hand, increase in the duration of the *slows* would not (almost) affect the playback in both PB and PB-P2P. Clients in the PB-P2P Slow may experience a little freeze due to low buffering time at startup which is similar to Section 4.5.1. For example, in the case of the PB-P2P Slow with 35 minutes Mean Interactive Duration, 87.21% of freezes happen at startup, and 12.79% of freezes happen at the second chunk. The average and maximum freeze time occurred at the second chunk are 7.91 and 22.54 seconds respectively.

The simulation result shows the resources demanded by each kind of interactive commands are arranged in the following ascending order: *slows*, *fasts*, *jumps*.

Figure 4.23d further reveals the importance of an efficient and stable source of chunks, i.e., the PB Server. For example, the ratio reaches to 0.84 when the duration is 35 minutes in the PB-P2P Jump case.

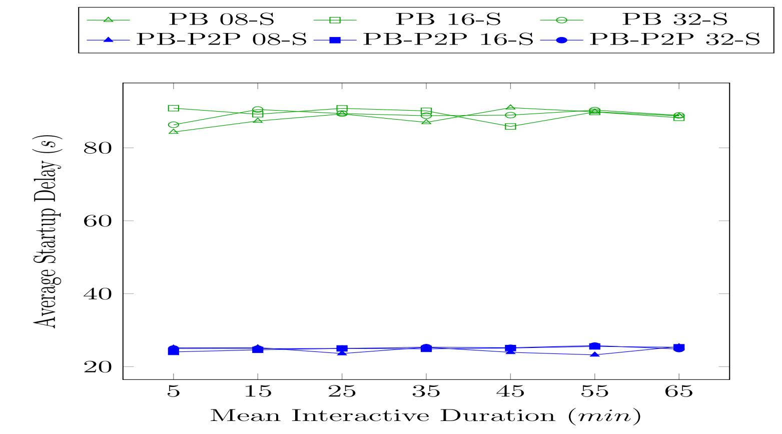
4.5.4 Interactive Mode - Fasts Speed

This section further examines the impact of the *fasts* on the system performance by changing the *fasts* relative speed to ±8, ±16, and ±32.

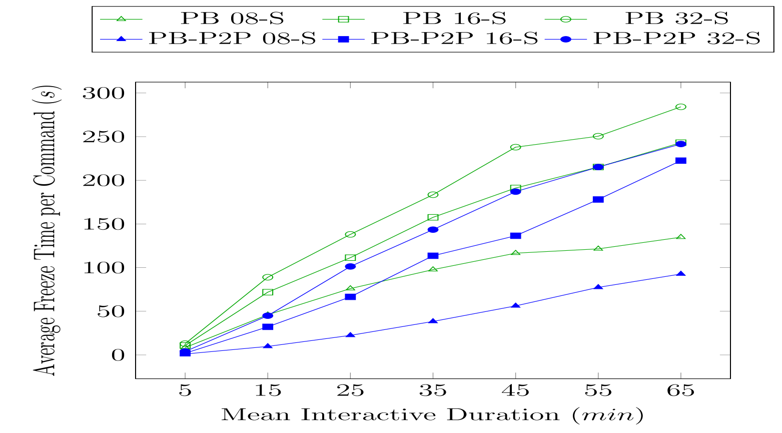
In each simulation, the user control selects either *fast forward* or *fast backward* with equal probability.

Figure 4.24b and Figure 4.24c show increase in the *fasts* relative speed deteriorates the performance of both scenarios that could be worse than the *jumps* as shown in Figure 4.23b and Figure 4.23c in Section 4.5.3.

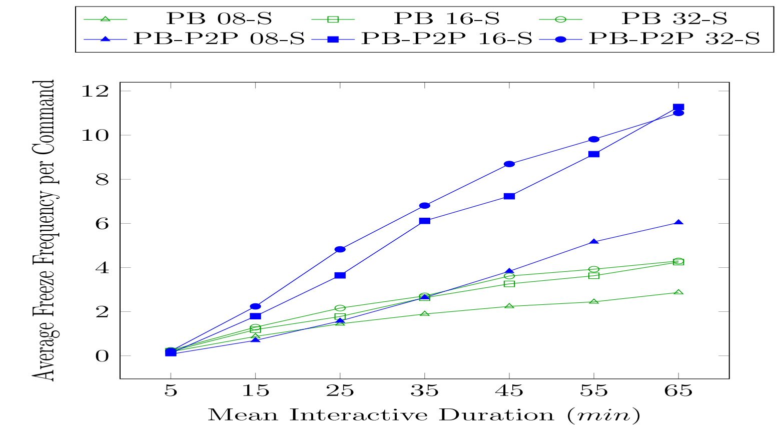
Overall, the PB-P2P has less playback delay but more freezes. It is because each client has only six download quotas. The download rate is far lower



(a)

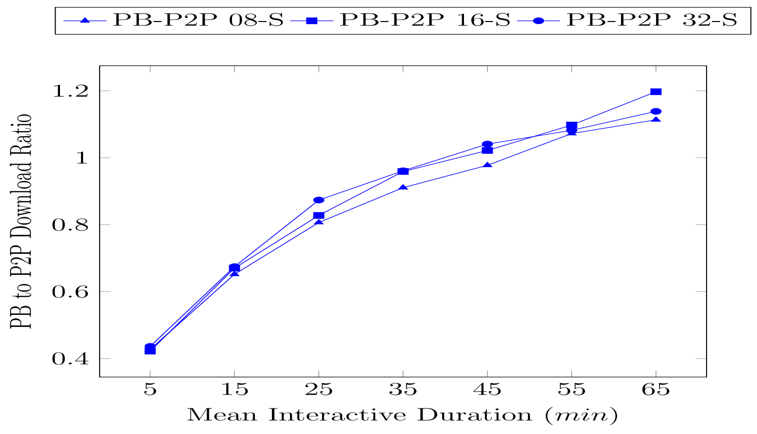


(b)

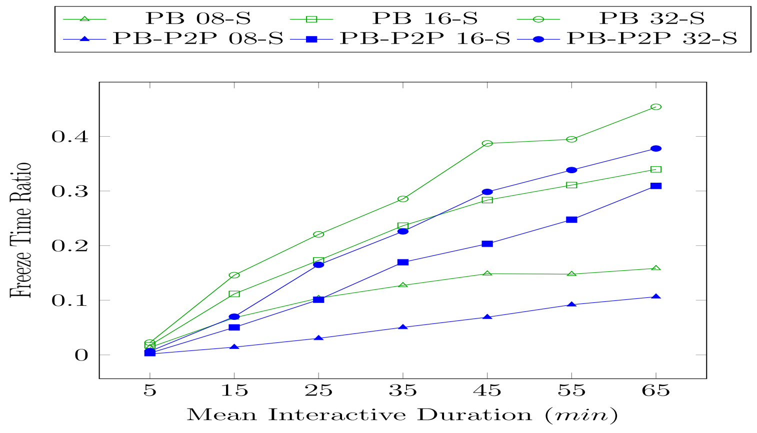


(c)

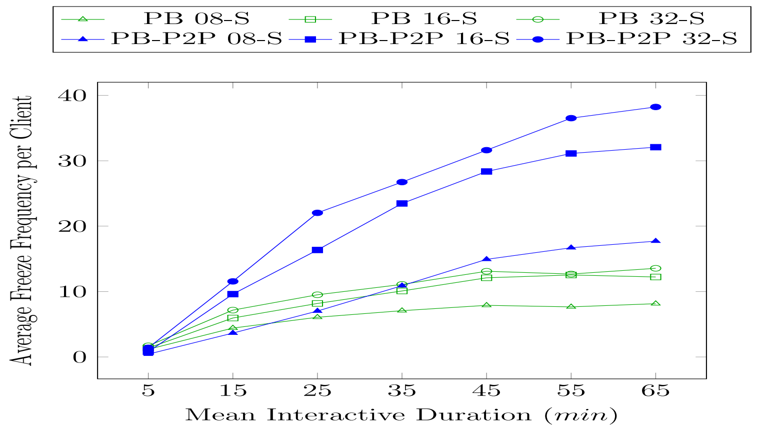
Figure 4.24: Change the speed of fast forward / backward



(d)



(e)



(f)

Figure 4.24: Change the speed of fast forward / backward

than the *fasts* playback rate. Freezes happen when the buffered chunks are exhausted.

This result shows the resource demanded by each kind of the interactive commands should be changed to the following ascending orders: *slows*, *low-speed fasts*, *jumps*, *high-speed fasts*. This order shall be an important reference for enabling the interactive commands selectively when the resource is limited.

Besides, the average startup delay remains steady across the simulations as shown in Figure 4.24a. The PB to P2P download ratio rises with the increase in the duration and the *fasts* relative speed which agrees with the results in Section 4.5.2 and Section 4.5.3 as shown in Figure 4.24d.

4.5.5 Download Quota and Buffer Size

This section examines the system performance by changing the download quota and buffer size. At first, three download quotas are assigned to the PB client component in both PB and PB-P2P scenarios. Then, we increase the download quota of PB client component in the PB scenario and P2P client component in the PB-P2P scenario. The upload quota assigned to the P2P server component is equal to the download quota assigned to the P2P client component. Table 4.8 summarises the assignment where ADQ stands for Additional Download Quota.

Table 4.8: Additional Download Quota (ADQ) assignment

Scenario	PB Client DL Quota	P2P Client DL Quota
PB	$3 + ADQ$	0
PB-P2P	3	ADQ

Clients with three different buffer sizes are tested, named minimum buffer (MIN-B), 14 Buffer (14-B), and Full Buffer (F-B). The size of the buffer component is equal to $(3 + ADQ) \times 6$ minutes in MIN-B, 14×6 minutes in 14-B, and 160.5 minutes in F-B, where 6 minutes is the length of the biggest segment and 160.5 minutes is the length of the video.

The user control only enables the *high-speed fasts* which gives the biggest impact to the system among other interactive commands as shown in Section 4.5.4. The user control selects either *fast forward* or *fast backward* equally. The relative speed of the *fasts* is configured to ± 16. The mean interactive duration is 35 minutes.

Figure 4.25a shows the startup delay remains steady in the PB and decreases rapidly in the PB-P2P when the download quota is increased.

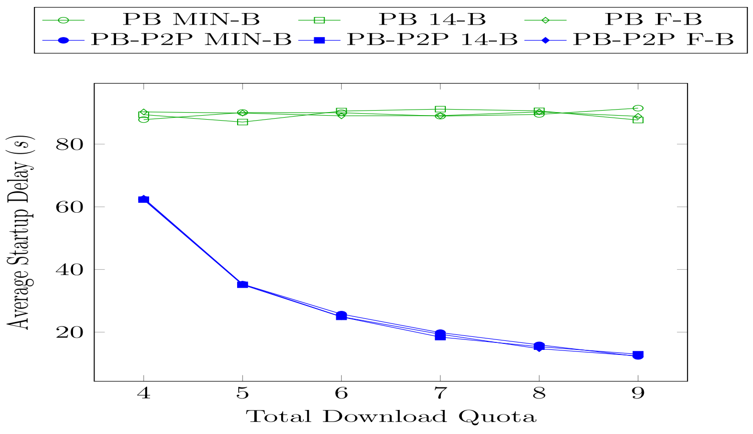
Figure 4.25b and Figure 4.25c show

1) increase in buffer size can significantly improve the performance when the download quotas are limited, and

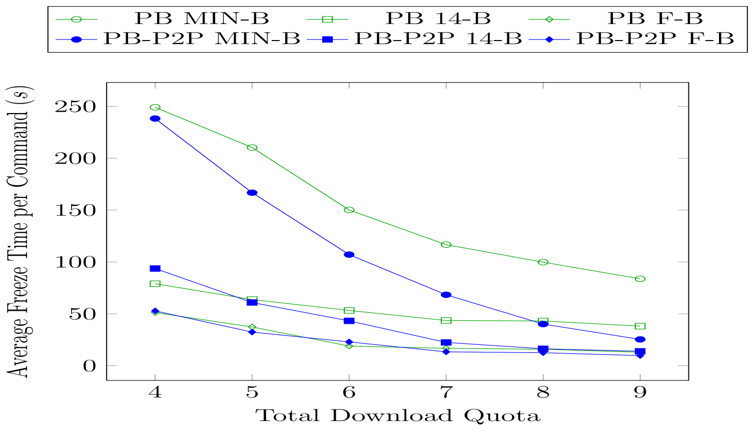
2) when the ADQ is increased, the P2P client component could bring greater improvement than the PB client component that is the key to provide smooth playback experience when interactive command is involved.

Figure 4.25d shows the download ratio declines with the increase in download quota(s). Biggest decline is observed when the download quota is risen from one to two. This decline suggests the P2P client component works better when the download speed is faster than a PB channel under current scheduling algorithms.

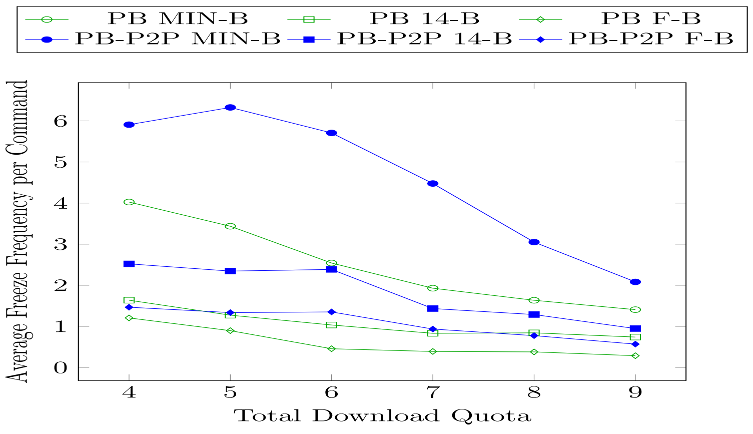
In conclusion, increase in buffer size and download quota improves the playback experience. The presence of the P2P is critical to improve the startup delay and the execution of the interactive commands. However, the use of the P2P also increases the traffic on the local network. As a result, to be network friendly, we may activate the P2P client component only in the situations such as 1) fetching the chunks at startup, or 2) executing the interactive commands,



(a)

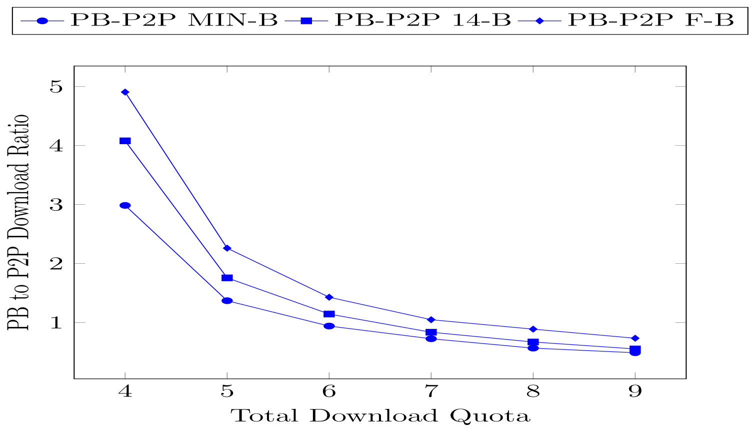


(b)

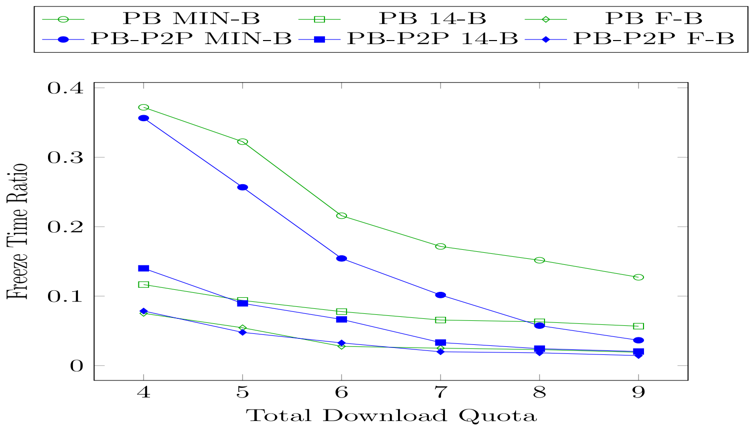


(c)

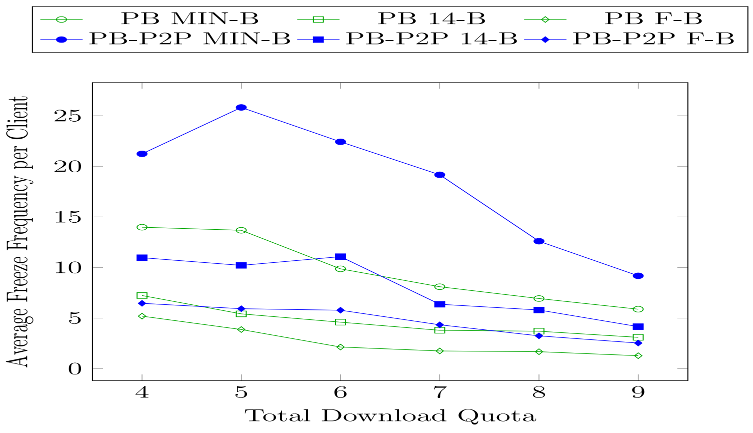
Figure 4.25: Change the download quota and buffer size



(d)



(e)



(f)

Figure 4.25: Change the download quota and buffer size

or 3) the proportion of *first-class chunks* in the buffer drops below the certain limit.

4.5.6 Chunk Length

This section examines the system performance by changing the *chunk length*. Three kinds of *chunk length* are selected which are 15, 18, and 30 seconds. The length of the video and segments remain unchanged.

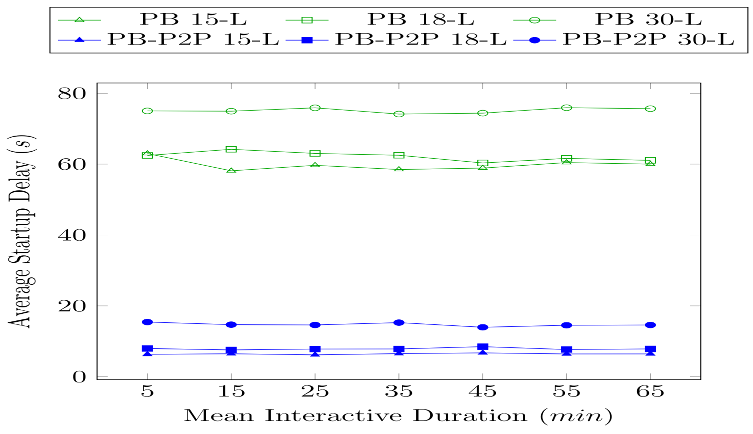
Only the *high-speed fasts* are enabled in the user control that gives the biggest impact to the system among other interactive commands as shown in Section 4.5.4. The user control selects either the *fast forward* or *fast backward* equally, and the relative speed is configured to ± 16.

Figure 4.26a shows shorter *chunk length* gives smaller startup delay that benefits PB most as explained in (4.10).

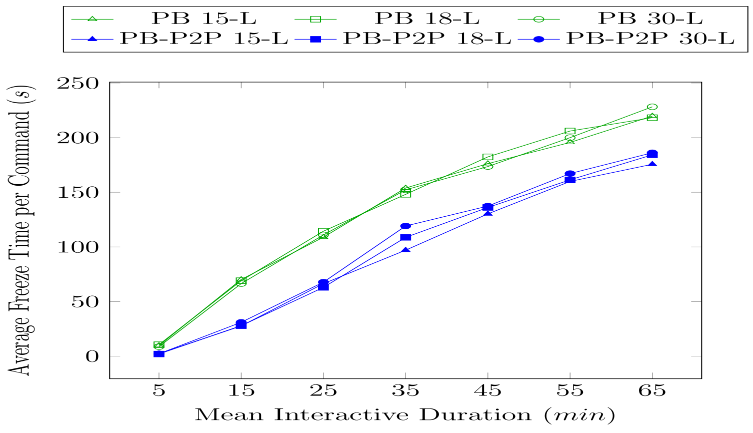
Figure 4.26b shows the decrease in *chunk length* reduces the freeze time of PB-P2P slightly. No improvement is observed in PB. Figure 4.26d shows shorter *chunk length* gives slightly lower download ratio. This means P2P clients can acquire more important chunks, and therefore, reduces the freeze time.

Figure 4.26c shows the freeze frequency rises with the decrease in *chunk length* in both PB and PB-P2P. It is because the download rate is far lower than the playback rate of the *fasts* and the segments are divided into more chunks that cause more freezes when the buffered chunks are exhausted.

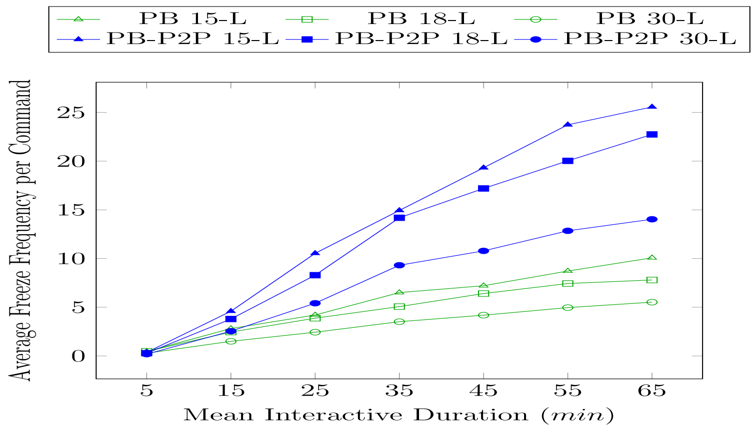
Overall, the decrease in the *chunk length* improves the system performance slightly. However, this might also increase the overheads that vary between the actual implementations and might not be fully shown in the simulations of this section. These overheads include 1) the metadata attached in each chunk, 2) the size and frequency of the chunk advertisements, and 3) the



(a)

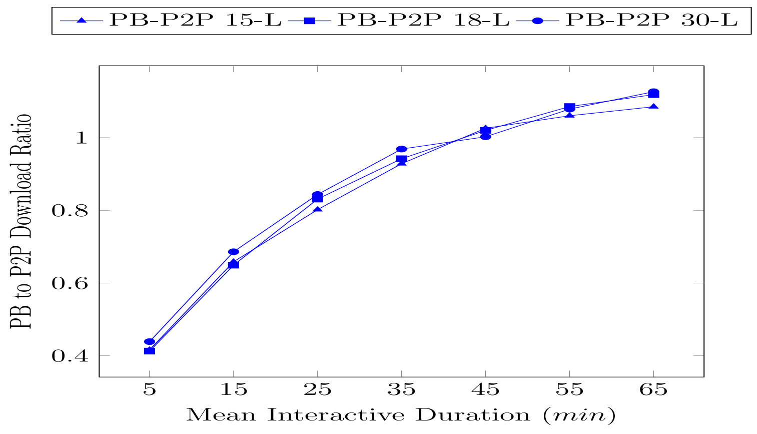


(b)

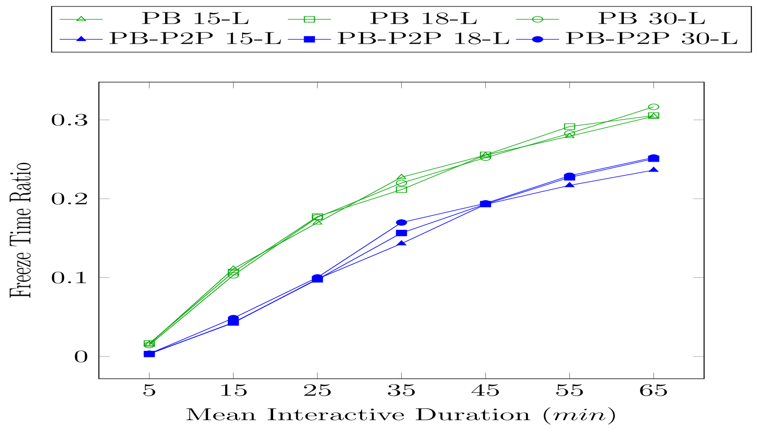


(c)

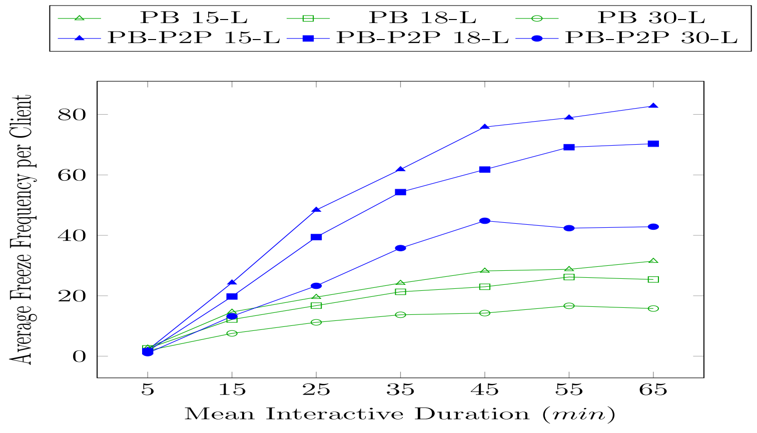
Figure 4.26: Change the chunk length



(d)



(e)



(f)

Figure 4.26: Change the chunk length

frequency that P2P clients query and download the chunk from P2P servers. As a reference, PPLive [19], a commercial P2P VoD system, reported that the average overhead is about 10% in the production environment.

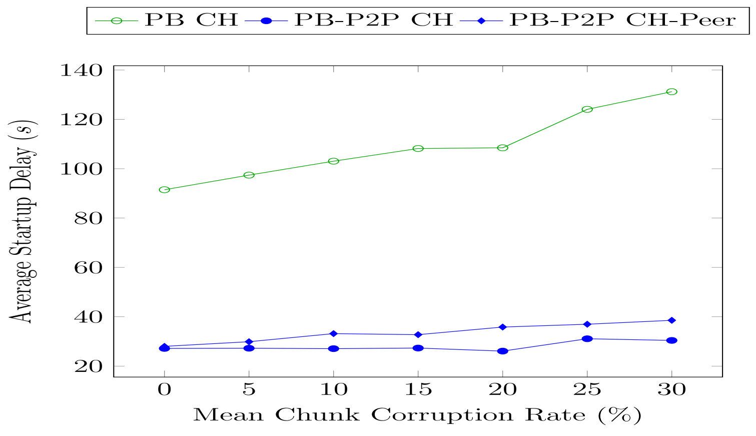
4.5.7 Corrupted Chunk in Playing Mode

This section examines the impact of the chunk corruption that the chunk received is different from the one transmitted. Chunk corruption happens when chunks are transmitted at best-effort such as IP multicast [45].

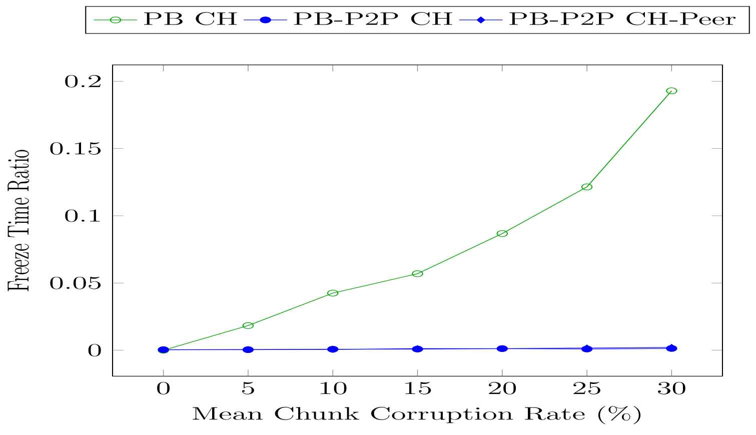
In the following simulations, a percentage of chunks received by the PB client and the P2P client are now filtered (corrupted) which is determined by a uniform random number. The filtered chunks would not be injected into the buffer.

To show the difference between the PB and the PB-P2P system, three cases are designed, named PB CH, PB-P2P CH, PB-P2P CH-Peer. The PB client filter is enabled in all cases while the P2P client filter is only enabled in the PB-P2P CH-Peer. The user control is disabled that the player always runs in the *playing* mode.

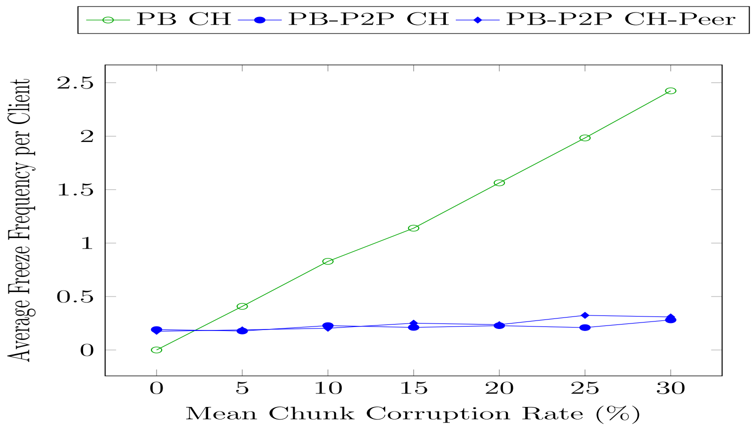
Figure 4.27 shows the increase in chunk corruption rate deteriorates the startup and playback performance of both PB and PB-P2P. However, the PB-P2P cases suffer much less. It is reasonable that if a received chunk is corrupted, the PB client can only download the same chunk in next upload cycle while the P2P client can immediately query the next P2P server if existed. Therefore, the P2P client can recover the corrupted chunk in the shorter period. This simulation result shows the use of the P2P is a good complement of the PB even if only *playing* is involved.



(a)



(b)



(c)

Figure 4.27: Change the chunk corruption rate

4.5.8 Network Loading

To reveal the potential of the proposed system further, this section examines three different scenarios against various client arrival rates and mean interactive durations. These scenarios are the original PB and PB-P2P scenarios, and a new P2P scenario.

In the P2P scenario, all six download quotas are assigned to the P2P client component that downloads a single chunk at a time. The PB server is replaced by five standalone P2P servers. These P2P servers are ready before the arrival of the first client, and shutdown after the departure of the last client.

The standalone P2P server is similar to the P2P server component installed on clients except for the chunk advertisement. The standalone P2P server sends out a special chunk advertisement periodically. The P2P Client components receive, process, and store these advertisements that would only be used when a chunk is not found in the peers. In other words, a P2P client component would query these standalone P2P servers one by one if any chunk cannot be downloaded from the peers.

A new performance metric, named total injected data packet, is introduced in this section. This metric shows the total number of the data packets that are injected into the network by the clients and the servers. This metric loosely reflects the loading of the network.

In the following simulations, only the *Fast Forward* and *Fast Backward* are enabled, and the relative speed is set to ± 16. The total upload quotas assigned to the five standalone P2P servers in the P2P scenario is $6 \times 5 = 30$ and is slightly more than the 28 quotas assigned to the PB server in the PB and PB-P2P scenarios.

Figure 4.28a shows the startup delay of each scenario remains steady and follows the following ascending orders: P2P ($\approx 8.88s$), PB-P2P ($\approx 25.48s$),

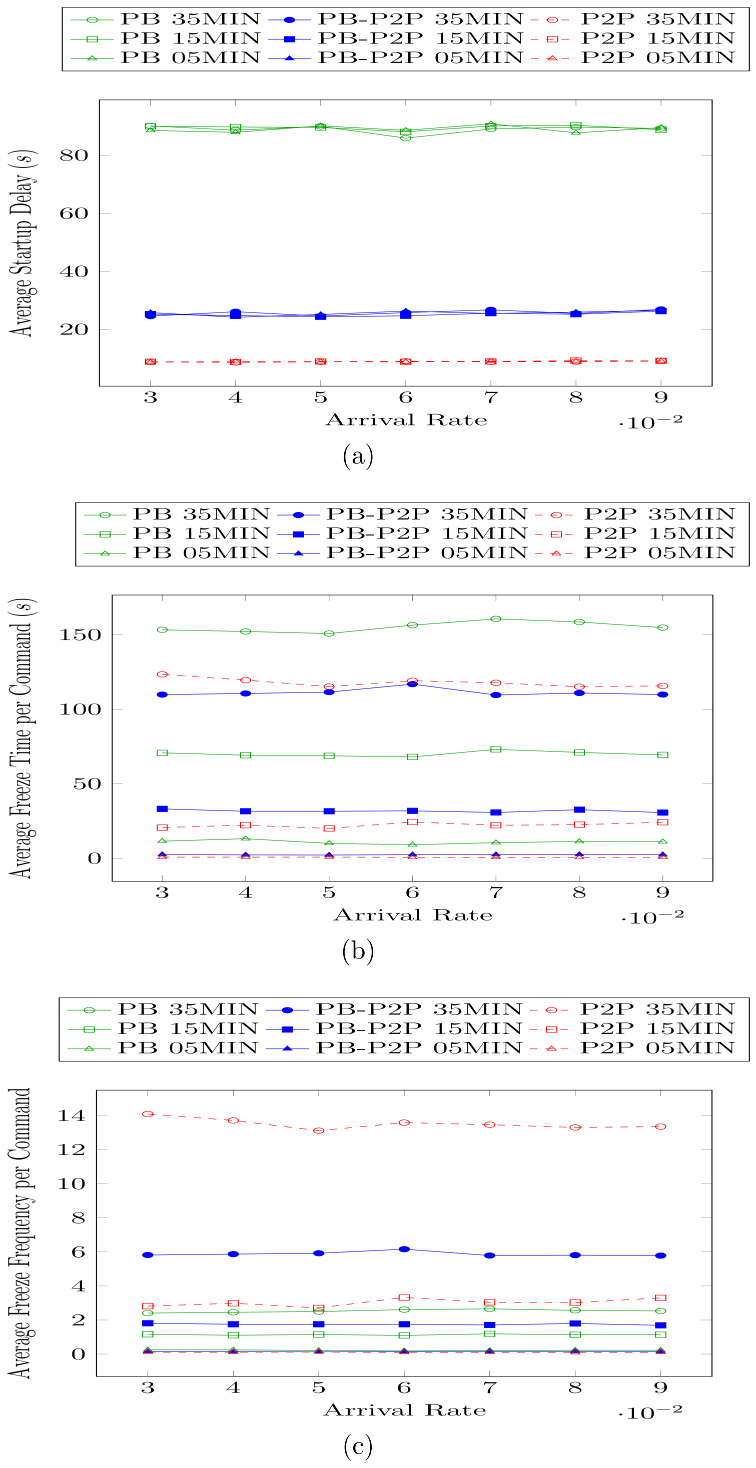
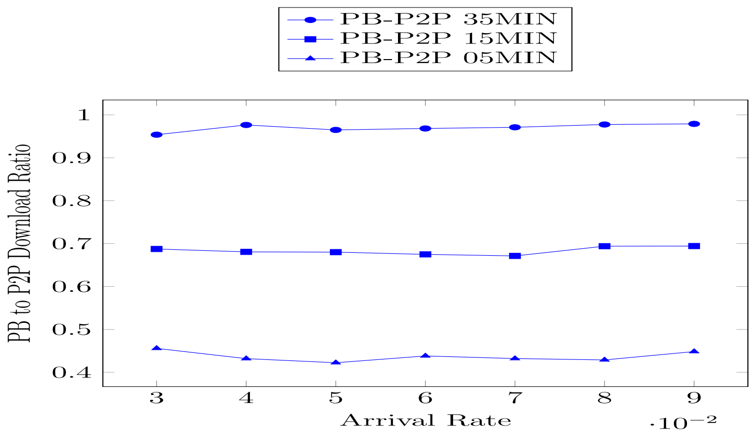
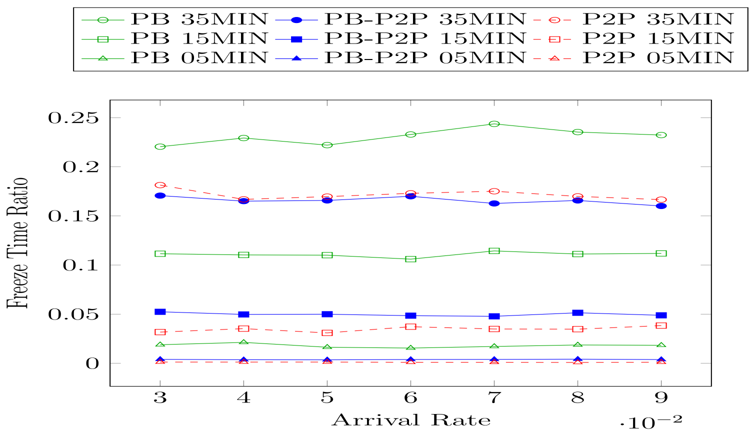


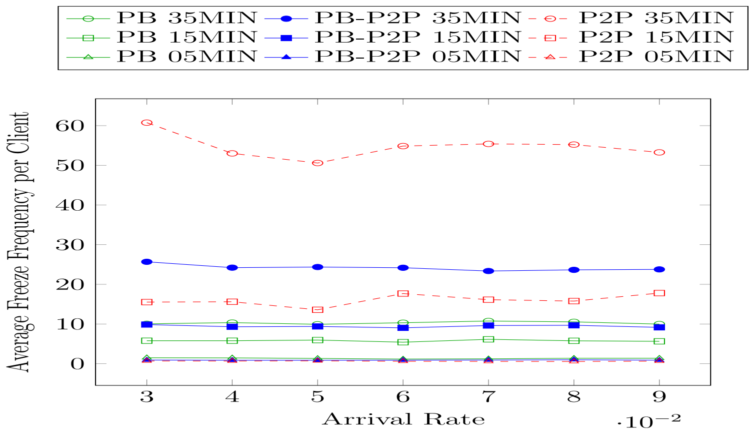
Figure 4.28: Examine the PB, PB-P2P, and P2P scenarios in various client arrival rates



(d)



(e)



(f)

Figure 4.28: Examine the PB, PB-P2P, and P2P scenarios in various client arrival rates

and PB ($\approx 89.17s$).

Figure 4.28b and Figure 4.28c show the PB-P2P has the smallest freeze time when the mean interactive duration is 35 minutes. However, the difference is narrowing when the duration decreases which the P2P could outperform the PB-P2P.

Figure 4.28g shows an enormous number of data packets have been injected into the network by the nodes in the P2P scenario, and the number rises with the increase of the arrival rate and the duration. The nodes in the PB scenario have injected the smallest number of data packets which remains constant across the simulations. The number of data packets injected by the nodes in the PB-P2P scenario is in the between of the P2P and the PB.

Overall, the simulation results show the PB gives the lowest network loading but has the least resistance to the increase in the mean interactive duration. The P2P could deliver the best performance in certain situations but also increase the network loading significantly. The PB-P2P, inheriting the strengths from the PB and the P2P, has the high resistance to the increase in the duration and injects much fewer data packets than the P2P to the network. These characteristics makes the PB-P2P be a good candidate on building a robust and cost-effective large-scale VoD system. Besides, the simulation result also reveals the potential of the PB-P2P when the download quota assigned to the PB and P2P client component could be adjusted freely.

Comparison with Other Closely Related VoD Systems

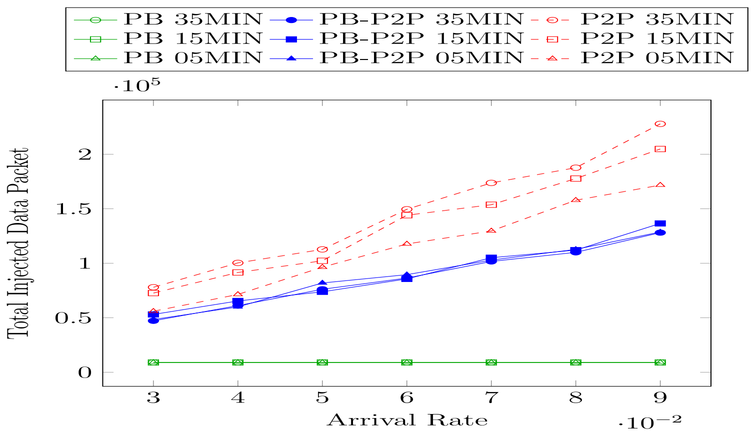
There are some studies on the hybrid of the multicast and P2P VoD systems. However, to the best of our knowledge, we are the first who propose the hybrid of the PB and P2P mesh pull VoD system and study the performance of the interactive commands in detail. The following describes some works closely related to this area.

Gopalakrishnan et al. [96] proposed a VoD system named Cooperative Peer Assists and Multicast (CPM). This system aims to reduce the server side loading by using the combination of pre-fetching (prepopulation), multicast, unicast, and P2P techniques. When a client receives the user request, the client acquires the chunks by following methods in order: 1) client's buffer which stores the beginning portion of the video, 2) existing multicast transmission scheduled on the server, 3) unicast transmission by the neighbouring peer, and 4) new multicast transmission scheduled on the server.

The CPM study shows the combination of the multicast and P2P can reduce the server side loading compared with other systems adopted the unicast, batching, or server assisted P2P. The multicast and P2P could be a good complement to each other. Another related study can be found in [102] that developed a mathematical model to compute the optimal starting points of the multicast sessions. Both studies have not investigated the support and impact of the interactive commands that could be a challenging problem when a significant portion of requests have relatively short playback duration, and the video browsing behaviour [20] is involved as described in Section 2.7.

Febiansyah et al. [95] proposed a hybrid of the PB and P2P systems named PeerPB that aims to support the clients with limited resources such as computational power or bandwidth. The system groups the clients who arrive closely to build the multicast trees. The clients with sufficient resources are placed at the higher level of the tree to serve the clients with limited resources, i.e., encode the received video data in lower bit rate and send to the child node(s). This approach could reduce the resource requirements of the PB VoD system, and might be adopted into our proposed system.

Fei et al. [29] proposed a technique, named Active Buffer Management, to support the discontinuous interactive commands. The idea behind ABM is to keep the *play point* in the middle of the buffered video data by joining the PB channels selectively. The ABM improves the latency on executing the interactive commands. However, the performance deteriorates quickly when the interactive duration is increased. As a result, several enhancements [90–92] have been proposed which demands extra server resources.



(g)

Figure 4.28: Examine the PB, PB-P2P, and P2P scenarios in various client arrival rates

4.6 Conclusion

This chapter proposes a PB-P2P VoD system which is a hybrid of PB and P2P mesh pull. In the system, server uploads the video data to channels periodically while clients switch among these channels and query peers in the same network concurrently to download the video data based on the proposed scheduling algorithms.

To study the behaviours of the proposed system, the system is implemented in ns-3 and various computer simulations are conducted. The simulation results show PB and P2P are a good complement to each other for a robust and cost-effective VoD system. For example,

- PB guarantees the maximum of startup delay, ensures smooth *playing* if no interactive command is involved, and provides an efficient and stable source of video data. These are difficult to achieve by using P2P alone.
- P2P is critical to shorten the startup delay, improve the execution of the interactive commands, and recover the corrupted chunk. These complements the weaknesses of the PB.

Moreover, the simulation results also show

- the proposed system gives flexibility on resource allocation. Users can allocate extra resources such as bandwidth and buffer to improve the playback experience further. On the other hand, users can disable the P2P completely when resources are limited.
- resources required by each kind of interactive commands are different and listed in the following ascending orders:

  1) slow forward / backward,
  2) fast forward / backward in low speed,

3) jump forward / backward, and

4) fast forward / backward in high speed

- the P2P performance can be improved further by dividing the video into smaller chunks.

Studies are carried out to explore the potential of the PB-P2P VoD System which includes the buffering algorithm for startup phase, segmentation scheme for PB-P2P VoD System, improvement of PB to P2P download ratio, and support on clients with limited resource such as bandwidth.

Chapter 5

Deploying Multicast in Peer-to-peer Network

The Hybrid of Periodic Broadcasting and Peer-to-Peer (PB-P2P) technique could be a robust and cost-effective solution for large-scale Video-on-Demand (VoD) system as shown in Chapter 4. However, in practice, some users are not able to run the Peer-to-Peer (P2P) servers due to various reasons such as firewall blockage or limited upload bandwidth. Moreover, the presence of free riders, who are reluctant to share their resources, make the situation even worse. Furthermore, the P2P servers use unicast to upload the chunks that can only serve one client in each upload. As a result, the availability of the free P2P servers can be greatly reduced that P2P clients may face a keen competition to secure an upload from neighbouring P2P servers.

In order to tackle this problem, the potentials of using the multicast in the P2P network are studied in this chapter. Patches are applied to the original P2P client described in Section 4.3.7 and original P2P server described in Section 4.3.9 to support the multicast function that allows P2P server to satisfy multiple clients in each upload.

For ease of illustration, the new kinds of P2P client and server are named

Peer-to-Peer over Multicast (P2P-M) client and P2P-M server while the original P2P client and server are named Peer-to-Peer over Unicast (P2P-U) client and P2P-U server in this chapter. Moreover, the term P2P server refers both P2P-U server and P2P-M server and the term P2P client refers both P2P-U client and P2P-M client in this chapter.

In the proposed P2P-M system, a P2P-M client queries the P2P-M servers as usual following the description in Section 4.3.7. The difference is when the query is accepted, the client joins the multicast group specified in the accept message and listens for the chunk data instead of building the direct connection to the server. P2P-M server accepts and satisfies requests from different clients for the same chunk by single upload stream. Unlike the Periodic Broadcasting (PB) server, the P2P-M server can terminate the upload at any time if the uploading chunk is replaced by another chunk in higher priority.

To study the performance of P2P-M, computer model and simulations have been built and conducted in ns-3 [82,85]. The rest of the chapter is organized as follow: Section 5.1 describes the P2P-M server and client in detail. Section 5.2 discusses the simulation environment, topology, metrics, and parameters. Section 5.3 shows the simulation result. Section 5.4 is the conclusion of this chapter.

5.1 P2P-M Server and Client

This section describes the patches to the P2P-U client and server in Section 4.3.7 and Section 4.3.9 in detail.

5.1.1 P2P-M Client

Join the multicast group specified in the accept message and listen to the chunk data are the major changes to the P2P-U client described in Section 4.3.7.

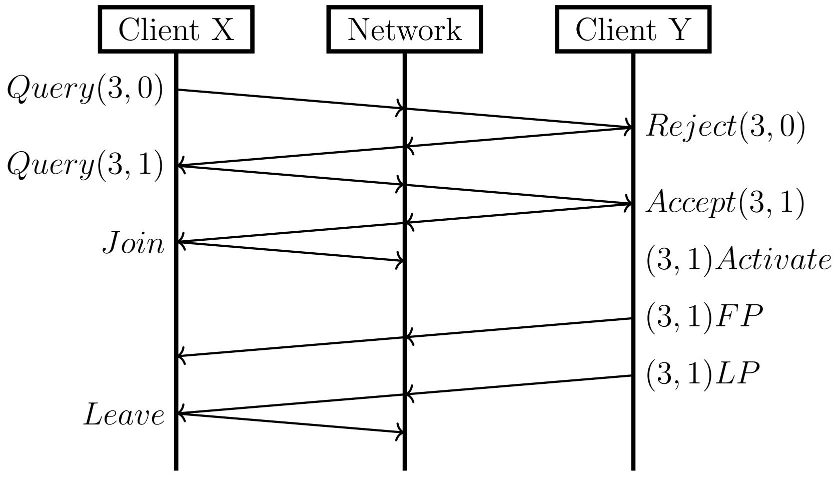


Figure 5.1: P2P-M message diagram

Figure 5.1 illustrates a typical P2P-M client operation. Suppose X needs chunks (3,0) and (3,1) that are only available in Y based on the received advertisements. Y is now uploading chunk (3,0) and the server queue length of Y is one that has already allocated to chunks (3,1).

X first creates a P2P-M loader to query Y for the chunk (3,0). Y rejects the query as the chunk (3,0) is not found in the server queue and the server queue is full. X then deletes the loader and creates a new one to query Y for the chunk (3,1). Since chunk (3,1) is found in the server queue of Y, Y returns the accept message that contains the information of the multicast group used by Y.

When X receives and processes the accept message, X joins the specified multicast group and listens to the chunk (3,1) data. The download ends when X receives the Last Piece (LP) of the chunk (3,1), deletes the loader, and sends the leave message to the network. Currently, P2P-M client discards all chunks except the requesting chunk that are transmitted by the specified multicast group.

Similar to the P2P-U client, the transmission can be terminated at any time by sending a message to another side. Moreover, if the P2P-M client has already joined the related multicast group, the client also sends a leave message to the network.

5.1.2 P2P-M Server

Creating tasks and serving queries from different clients for the same chunk by a single upload stream are the major changes to the P2P-U server described in Section 4.3.9.

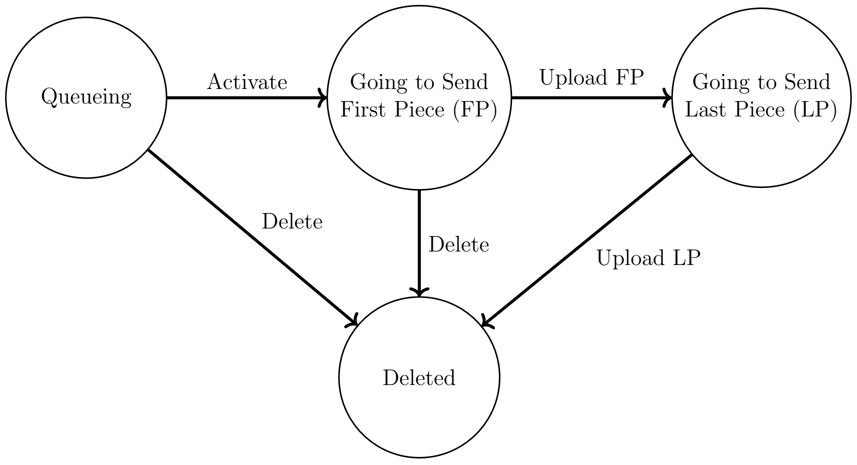


Figure 5.2: P2P-M server task state diagram

Figure 5.2 shows all the possible states of a P2P-M server task. A newly created task has state *Queuing*. When the task is activated, the state changes from *Queuing* to *Going to Send First Piece*. When the First Piece (FP) of the chunk is uploaded, the state changes from *Going to Send First Piece* to *Going to Send Last Piece*. Finally, the state changes from *Going to Send Last Piece* to *Deleted* when the LP of the chunk is uploaded. Moreover, since the P2P-M server and client can delete the task or cancel the query at any time, all the other states are also linked to *Deleted*. For example, P2P-M server would delete a task if the chunk requested by P2P-M clients is replaced by another chunk in higher priority or all P2P-M clients inside the task cancel the request. A *Deleted* task will be removed from the server queue immediately.

When a P2P-M server receives a query from a P2P-M client, the server first searches the task(s) in *Queuing* state. If a match is found, the server accepts the query. Otherwise, the server tries to create a new task and append the task to the server queue. If it is done, the server accepts the query. Otherwise, the server rejects the query. In other words, the queries requesting the chunk found in the server queue are grouped and served by signal task, which is kind of batching technique [30].

A dedicated multicast group is assigned to each P2P-M server. P2P-M server activates one task and uploads the requesting chunk to the multicast group at a time based on the first-come-first-serve policy until the server queue is empty. The time that a task stays in *Queueing* state is determined by the total serving time of the preceding tasks. If the P2P-M server queue is empty, any newly created task is activated immediately in the current design.

The patch of P2P-M server introduces a new parameter, named P2P-M server multicast idle time. This parameter specifies the total time for a task must stay in the *Going to Send First Piece* state after the activation. It is the buffering time for P2P-M clients to join the related multicast group and is similar to PB channel idle time described in Section 4.2.4.

5.2 Simulation Setting

Same simulation environments, network topology, and performance metrics described in Section 4.4.1, Section 4.4.2, and Section 4.4.3 are used. The following sections describe the definition of the additional performance metrics.

5.2.1 Additional Performance Metric

To show the differences between the P2P-U and P2P-M. Two additional metrics are introduced, named P2P query acceptance ratio and average P2P group size.

P2P Query Acceptance Ratio

This metric reflects the cost on securing an accept from the P2P servers. The definition is:

$$R_{acc} = \frac{n_{acc}}{n_{rej} + n_{acc}} \tag{5.1}$$

where n_{rej} and n_{acc} are the number of rejected and accepted queries in whole simulation.

Average P2P-M Group Size

This metric shows the average number of P2P-M clients served in each task that includes the clients who leave in the middle of the transmission. The definition is:

$$N_{GSize} = \frac{\sum\limits_{k \in R_{P2P-M}} n_k}{n_R} \tag{5.2}$$

where R_{P2P-M} is the set of all P2P-M transmissions records. n_k is the total number of clients served in the kth P2P-M transmission. n_R is the total number of P2P-M transmission, i.e., the size of R_{P2P-M}.

5.2.2 Simulation Parameter

Parameters in Section 4.4.4 are almost adopted in the following simulations except those related to the segmentation scheme.

The video is evenly divided into 37 segments with length 264 seconds (4.4 minutes). Each segment is evenly divided into six chunks with length 44 seconds. PB Server allocates 37 channels and uploads each segment in a dedicated channel continuously and periodically.

One channel quota is assigned to PB Client in order to maintain the smooth

Table 5.1: Simulation parameter

Parameter	Value
Total Simulation Time	$Video\ Length \times 1.5$
Client Arrival Rate per Second	0.03
Client Lifetime Mean	$Video\ Length \times 0.7$
Video Length	162.8 *minutes*
Chunk Length	44 *seconds*
Segment Length	264 *seconds*
Total Server Channel (Segment)	37
Client Channel Control Quota for Download	PB Client: 1, P2P Client: 2
Point-to-point Channel Delay	20 *milliseconds*
User Control Mean Playing Time	10 *minutes*
User Control Mean Interactive Duration	5 *minutes*
Enabled Interactive Mode	*All*
Player Interactive Mode Probability	$1/n$, where n is the total number of activated interactive mode
Player Starting Elapsed Time	0 *seconds*
P2P Server Queue Length	1
P2P Server Advertisement Mode	*Injected Chunks*
P2P Server Advertisement Timer	$Chunk\ Length + 1\ s$
P2P Client Maximum Number of Stored Advertisements per Chunk	5
P2P Client Source Select Mode	*Most Stable Source*
P2P Server Installation Rate	100%
Client Buffer Size	13.2 *minutes*
PB Channel Idle Time	1 *seconds*
P2P-M Server Multicast Idle Time	1 *seconds*

playback if no interactive command is involved. Two channel quotas are assigned to P2P Client because the simulation result in Figure 4.25d of Section 4.5.5 shows that P2P Client works better when the download speed is faster than a PB channel under current scheduling algorithms. The buffer size is 13.2 minutes which can store three segments (18 chunks) at the same time.

P2P server installation rate and P2P-M server multicast idle time are two new parameters introduced in this chapter. P2P server installation rate controls the availability of the P2P Server. For example, 30% installation rate means only 30% of incoming clients, which is determined by a uniform random number, will install and activate the P2P server.

P2P-M server multicast idle time is set to one second which is as same as PB channel idle time and is only applicable to the P2P-M servers. For example, as shown in Table 5.1, the chunk length is 44 seconds, each P2P server has 2 channel quotas, and P2P-M server multicast idle time is 1 second. According to (4.3), a P2P-U server takes $T_2 = \frac{l-t}{n} = \frac{44-1}{2} = 21.5$ *seconds* to upload a chunk since the P2P-M server multicast idle time is not applicable. On the other hand, a P2P-M server takes $1 + T_2 = 1 + 21.5 = 22.5$ *seconds* to upload a chunk. 1 second is spent on the *Going to Send First Piece* state. The other 21.5 seconds are used to upload the chunk.

5.3 Simulation Result

In this section, we examine the performance of P2P-U and P2P-M under two different situations. Firstly, we disable the user control component that only *playing* mode is involved. Secondly, we enable the user control component and activate *high-speed fasts*.

5.3.1 P2P Server Installation Rates – Playing

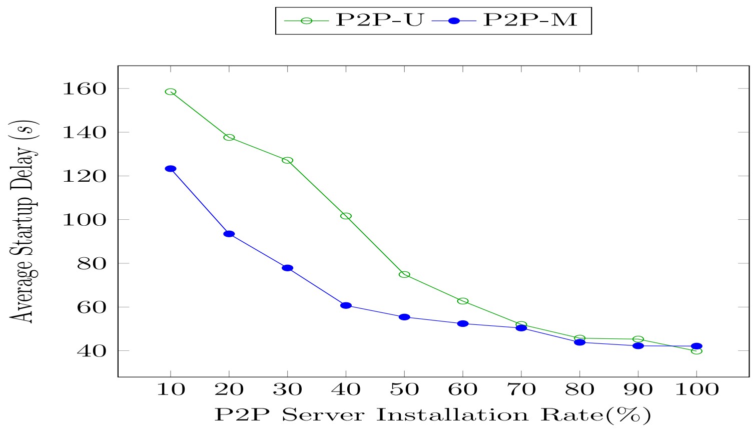
The user control is disabled in this section in order to study the differences between the P2P-U and P2P-M against various P2P server installation rates under *playing* mode.

The metrics average freeze time per command and average freeze frequency per command are not applicable in this section as no interactive command is involved.

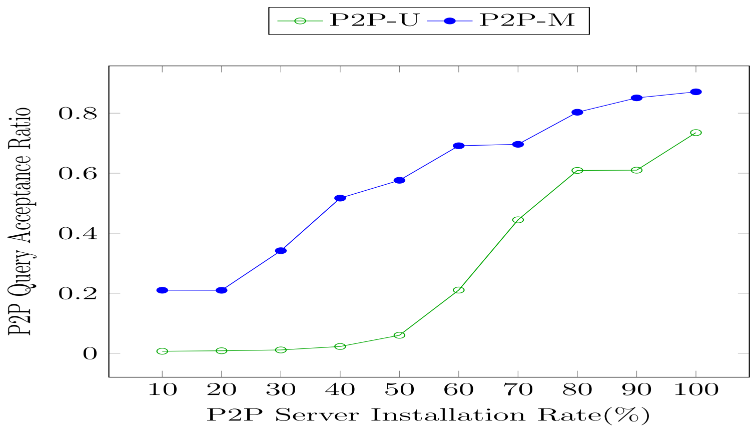
Figure 5.3a shows that P2P-M has lower startup delay than P2P-U when the installation rate is 60% or below. The difference is narrowing and becomes minor when the installation rate reaches 70% or above.

Figure 5.3b and Figure 5.3c illustrate the causes. Figure 5.3c shows P2P-M has much lower download ratio than P2P-U when the installation rate is 40% or below. This difference shows P2P-M clients can acquire much more chunks from neighbouring peers than P2P-U clients, and therefore, lower startup delay is resulted.

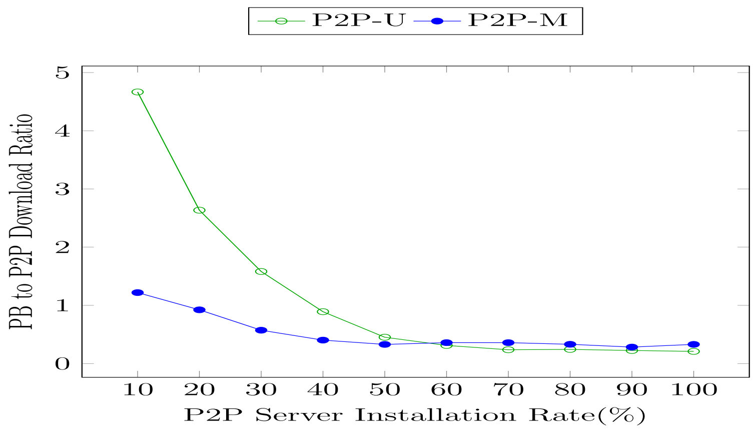
Figure 5.3b shows that P2P-U has lower query acceptance ratio than P2P-M in all simulations. P2P-U clients are facing keener competition on securing an upload from the neighbouring peers than the P2P-M clients. As described in Section 4.3.7, a P2P Client queries the P2P servers found in the stored advertisements one by one following the download order. The number of stored P2P advertisements for each chunk is limited. When a query is sent, the related advertisement is erased to protect the P2P servers from waves of query. Therefore, if the query acceptance ratio is low, P2P clients generally cannot download the most urgent chunk from neighbouring peers that deteriorates the startup delay. As the result, P2P-M still has lower startup delay when the installation rate reaches 50% and 60% even if the difference in download ratio becomes negligible.



(a)

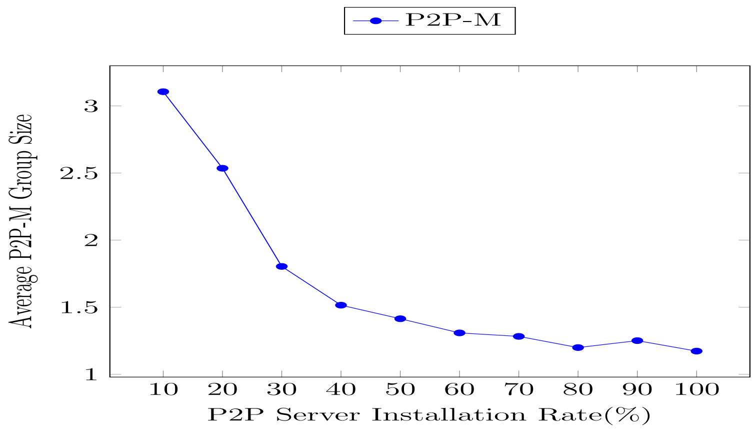


(b)

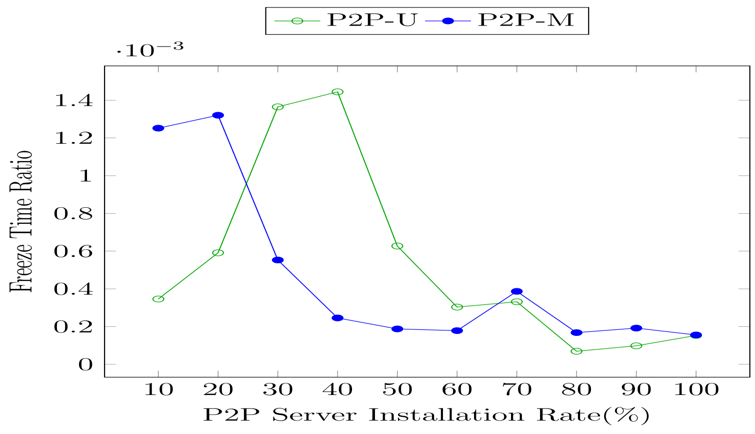


(c)

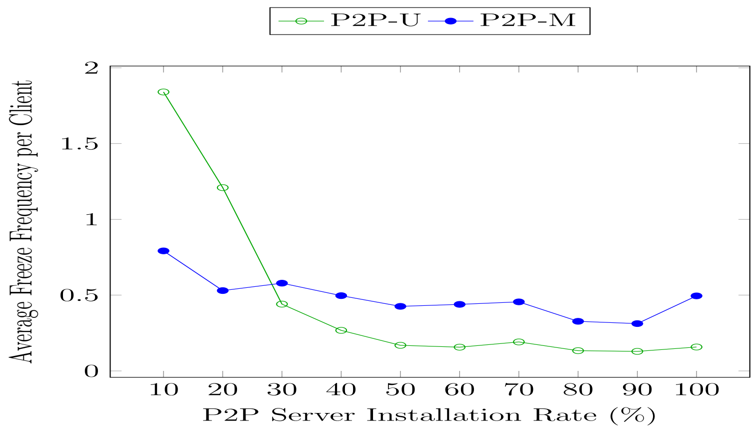
Figure 5.3: Changing the installation rate of the P2P server component



(d)



(e)



(f)

Figure 5.3: Changing the installation rate of the P2P server component

Figure 5.3d shows the average number of P2P-M clients served in each P2P-M transmission. The decrease in the number of served client further explains why the performance difference between P2P-M and P2P-U is narrowing.

Figure 5.3e and Figure 5.3f show that the differences in freeze time and frequency are minimal once the playback is started. Both P2P-M and P2P-U offer a quality playback experience. The unit of y-axis of Figure 5.3e is 10^{-3}. Suppose a client lifetime is $Video\ Length \times 0.7 = 162.8 \times 60 \times 0.7 = 6837.6\ seconds$. When freeze time ratio is equal to 1×10^{-3}, it actually represents a freeze time of $1 \times 10^{-3} \times 6837.6 = 6.8376\ seconds$.

Table 5.2: Three different scenarios found in the P2P-U simulations classified by P2P Query Acceptance (ACC) Ratio and PB to P2P Download (DL) Ratio

	P2P Query ACC Ratio	PB to P2P DL Ratio	P2P-U Server INST Rate
1	Low	High	10% – 40%
2	Low	Low	50% – 60%
3	High	Low	70% – 100%

In general, three different scenarios can be observed across the simulations. Let's take the P2P-U as an example which is summarized in Table 5.2. In scenario one, P2P client component experiences a keen competition on securing an upload from neighbouring peers and a large portion of chunks are downloaded from the PB channels. In scenario two, P2P client component still experience a keen competition but the component can keep downloading the chunks from neighbouring peers. In scenario three, P2P client component experiences low competition and the component can keep downloading the chunks in lower download order from neighbouring peers. The simulation result shows that the P2P-M achieves better performance in scenarios one and two under current scheduling algorithm.

5.3.2 P2P Client Maximum Number of Stored Advertisements per Chunk – Playing

The maximum number of stored advertisements per chunk is set to five to reduce the memory consumption. However, this might deteriorate the performance of the P2P client when P2P query acceptance ratio is low. Therefore, in this section, we increase the limit of stored advertisements that allows P2P client to query more P2P servers one by one. This change aims to ensure the performance difference between P2P-U and P2P-M shown in Section 5.3.1 is caused by the shortage of P2P server mainly but not the number of the stored advertisements.

In the following simulations, the user control is disabled. Three different scenarios are tested where P2P server installation rate is set to 25%, 55%, and 85% respectively. In all scenarios, the maximum number of the stored advertisements per chunk is increased from 5 to 35.

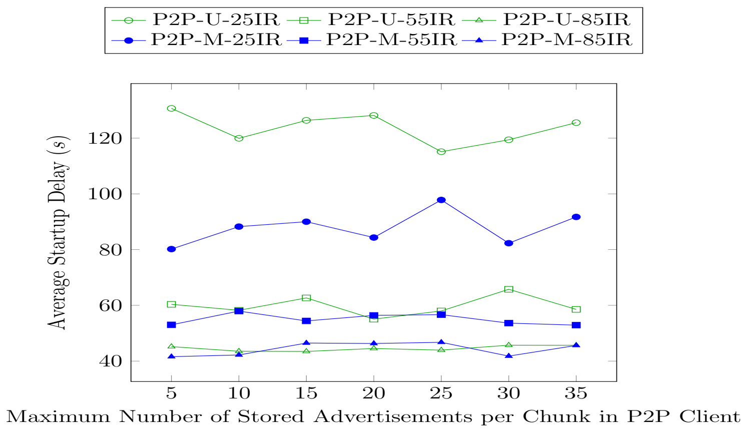
Figure 5.4a shows the startup delay remains steady across the simulations. P2P-M has lower startup delay than P2P-U, and the difference is narrowing when the installation rate is increased from 25% to 85%.

Figure 5.4b shows the query acceptance ratio remains steady. Moreover, P2P-M has significantly higher query acceptance ratio than P2P-U.

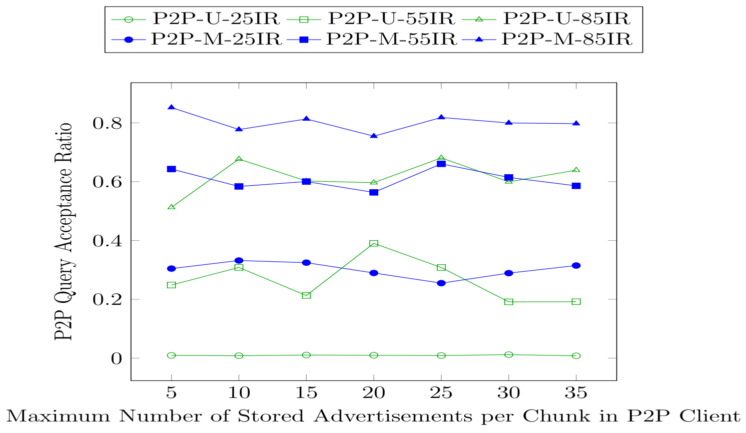
Figure 5.4c shows the download ratio remains steady across the simulations. For example, when the installation rate is 25%, P2P-U clients still rely on the PB channels to acquire most of the chunks.

Figure 5.4d shows no significant change is observed when the number of stored advertisements is increased from 5 to 35.

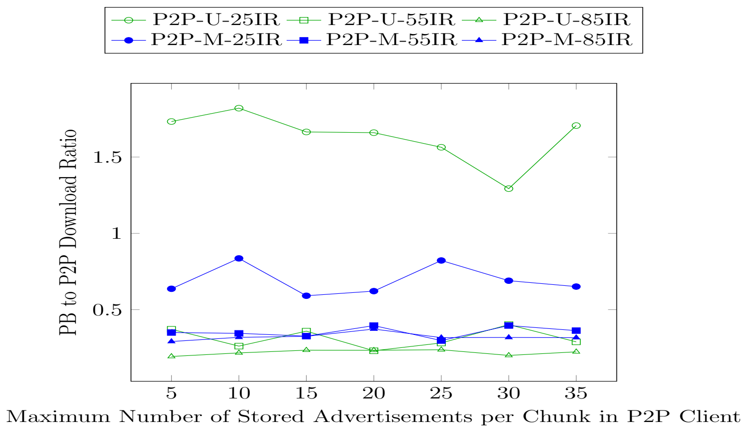
Figure 5.4e and Figure 5.4f show the differences in freeze time and frequency are minimal once the playback is started which agrees with the simulations results shown in Section 5.3.1.



(a)

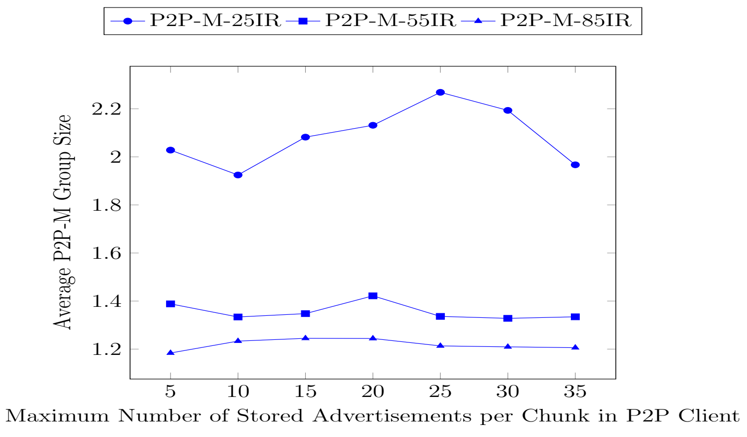


(b)

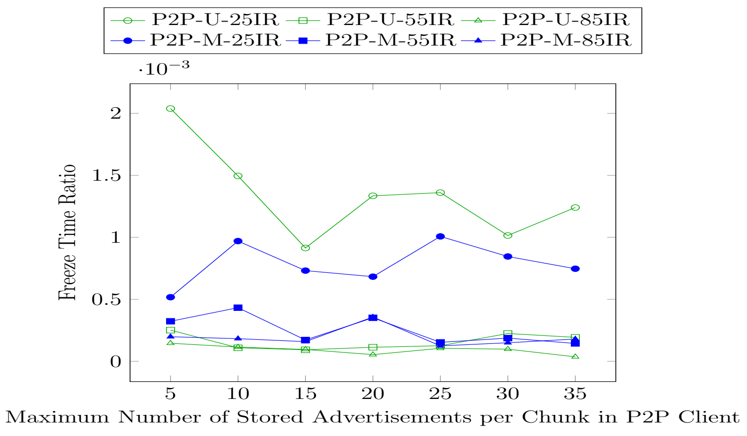


(c)

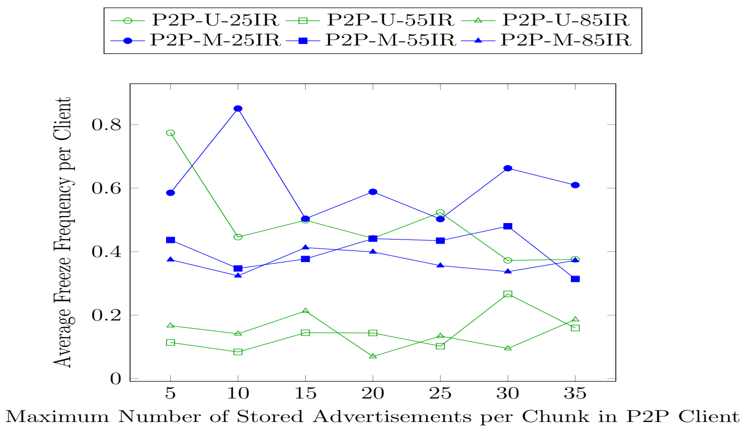
Figure 5.4: Change the maximum number of stored advertisements per chunk in P2P client



(d)



(e)



(f)

Figure 5.4: Change the maximum number of stored advertisements per chunk in P2P client

In conclusion, the simulation results show the performance difference between P2P-U and P2P-M shown in Section 5.3.1 is not dominated by the default number of the stored advertisements.

5.3.3 P2P Server Advertisement Timer Reset Value – Playing

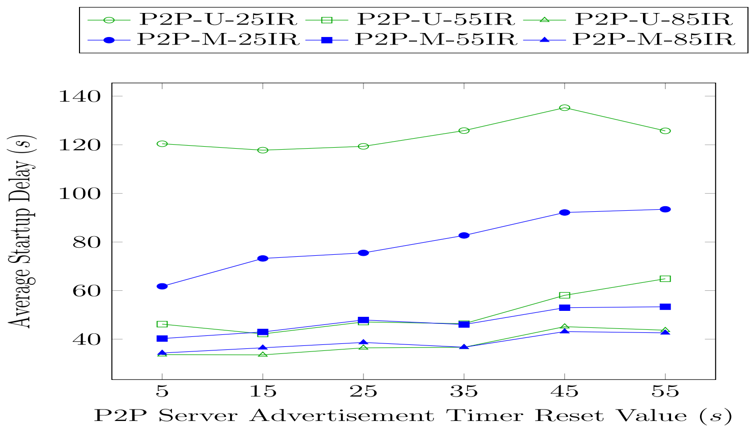
This section investigates the impact on changing the reset value of P2P server advertisement timer. Similar to Section 5.3.2, this section aims to ensure the performance difference between P2P-U and P2P-M shown in Section 5.3.1 is caused by the shortage of free P2P server mainly.

As described in Section 4.3.9, each P2P server sends the latest advertisement when either a new download order is received or the P2P server advertisement timer reaches zero. Thus, the timer reset value is the maximum time between two consecutive advertisements transmitted by the same P2P server. For example, as shown in Table 5.1, the timer reset value is $Chunk\ Length + 1 = 45s$ which guarantees the P2P servers would send out an advertisement every 45 seconds. In other words, decrease in the timer reset value allows a P2P client to query the same P2P server for the same chunk again in shorter periods of time.

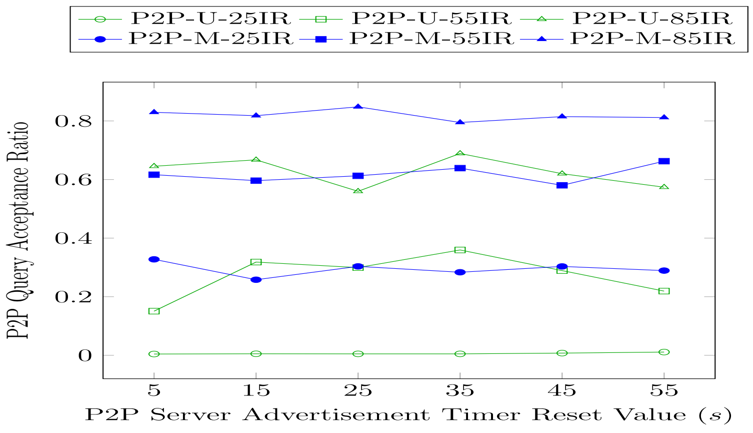
In the following simulations, the user control is disabled. Three different scenarios are tested where P2P server installation rate is set to 25%, 55%, and 85% respectively, and the timer reset value is increased from 5 to 55 seconds.

Figure 5.5a shows decrease in the timer reset value improves the startup delay slightly. However, P2P-M still has the lower startup delay than P2P-U, and the difference is narrowing when the installation rate is increased from 25% to 85%.

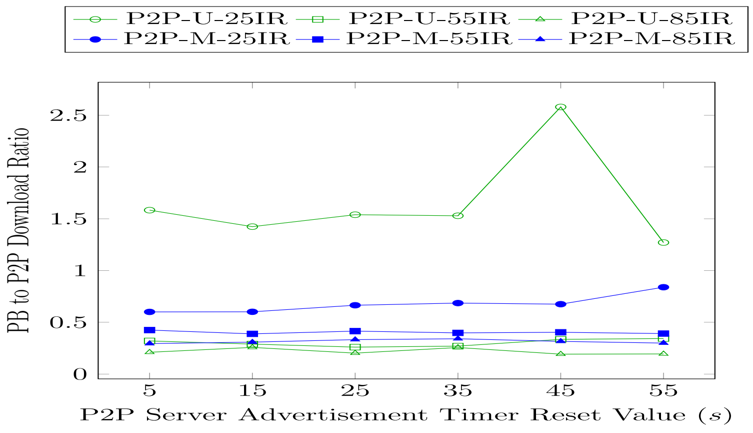
Figure 5.5b shows the query acceptance ratio remains steady, and P2P-M



(a)

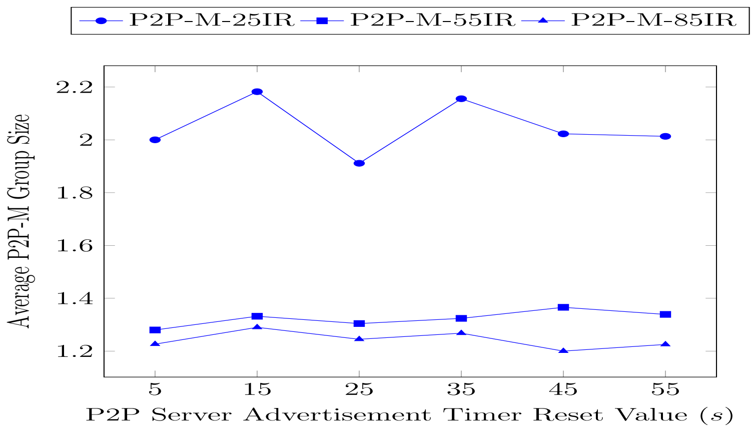


(b)

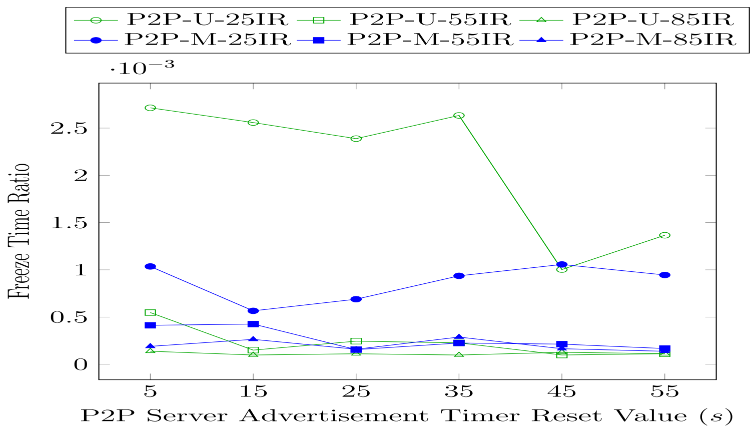


(c)

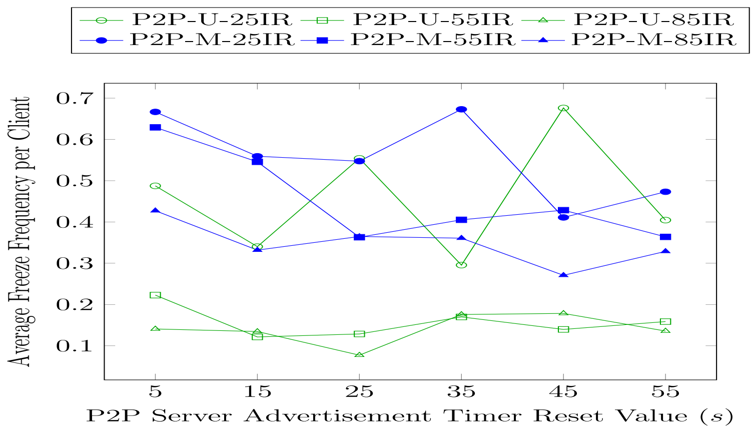
Figure 5.5: Change the advertisement timer reset value in P2P server



(d)



(e)



(f)

Figure 5.5: Change the advertisement timer reset value in P2P server

has significantly higher query acceptance ratio than P2P-U.

Figure 5.5c shows P2P-U clients still download most of the chunks from the PB channel(s) when installation rate is 25% as high download ratio is observed.

Figure 5.5d shows the average number of P2P-M clients served in each P2P-M transmission remains steady across the simulations.

Figure 5.5e and Figure 5.5f show the differences in freeze time and frequency are minimal once the playback is started which agrees with the simulations results shown in Section 5.3.1 and Section 5.3.2.

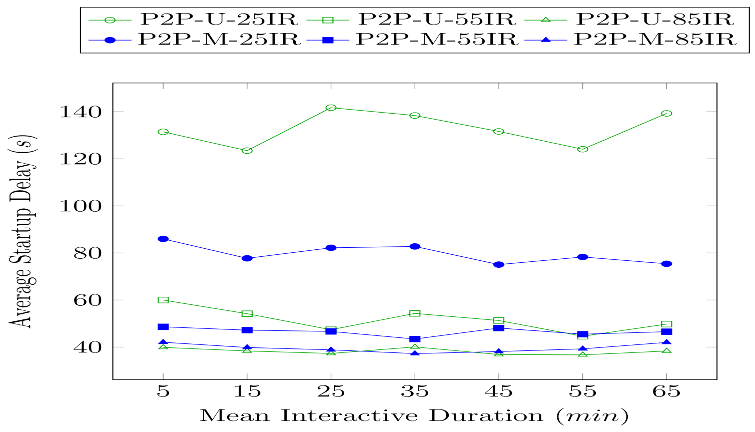
In conclusion, the simulation results clearly show changing the timer reset value does not boost the performance of P2P-U. The performance difference between P2P-U and P2P-M shown in Section 5.3.1 is caused by the shortage of P2P server, and the use of multicast do improve the availability of the P2P server.

5.3.4 Interactive Mode – High-speed Fasts

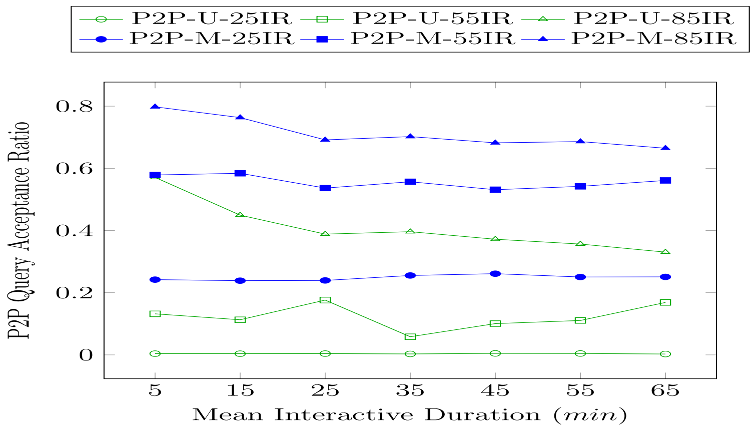
The *high-speed fasts* are the interactive command that gives the biggest impact to the performance as shown in Section 4.5.4. Therefore, to further study the difference between the P2P-U and P2P-M, this section examines the impact of the *fasts* in three P2P server installation rates, i.e., 25%, 55%, and 85%, and various mean interactive durations. The user control is enabled and selects either *fast forward* or *fast backward* equally which relative speed is configured to ±16.

Figure 5.6a shows the average startup delay of the P2P-M is shorter than the P2P-U when the installation rate is 25%. However, when the installation rate reaches 55% and 85%, no obvious difference can be observed.

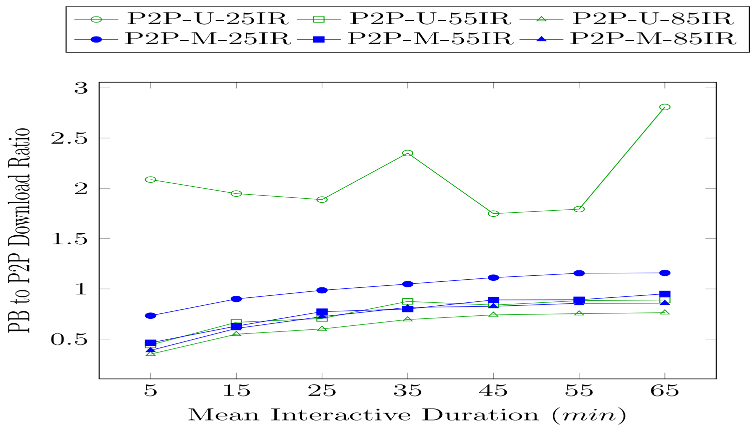
Figure 5.6b shows the P2P-M has significantly higher query acceptance



(a)

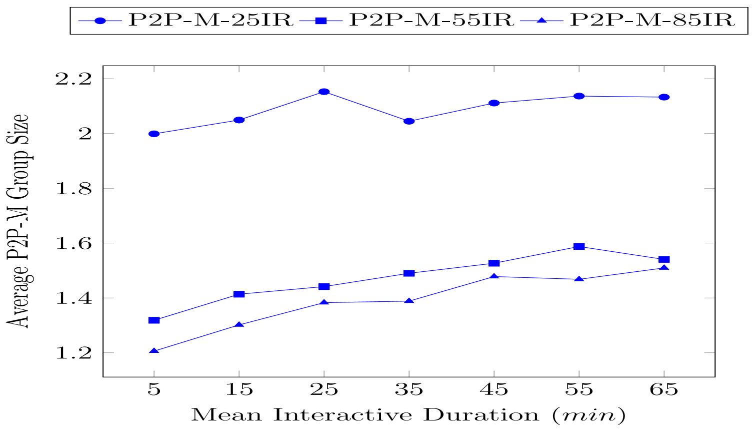


(b)

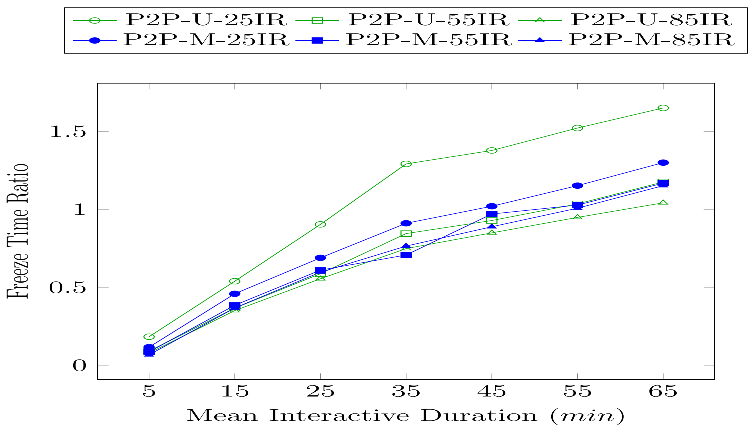


(c)

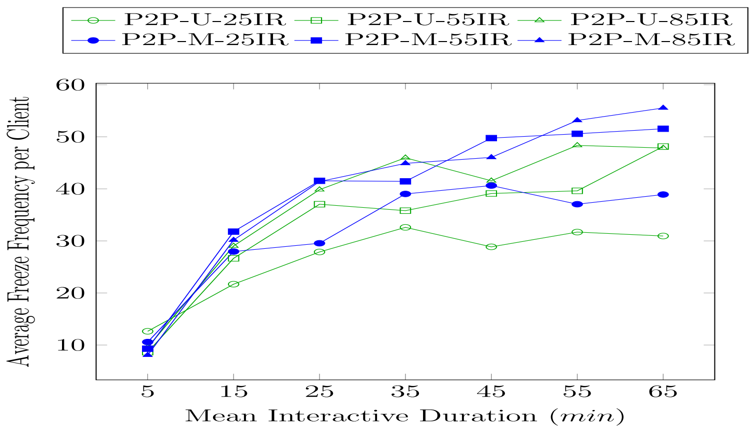
Figure 5.6: Change the mean interactive duration and enable high-speed fasts



(d)

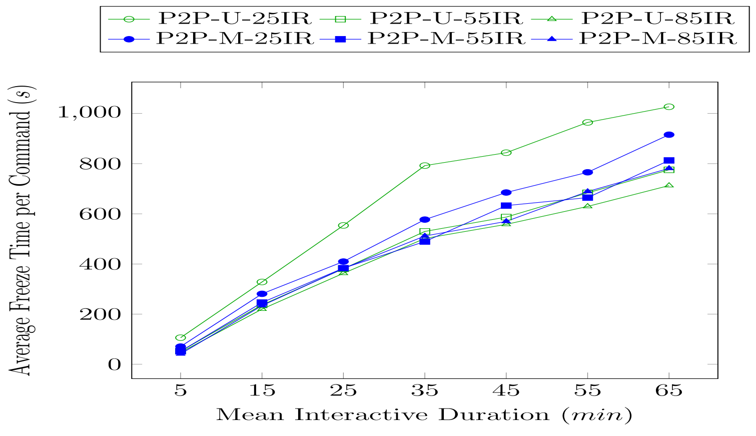


(e)

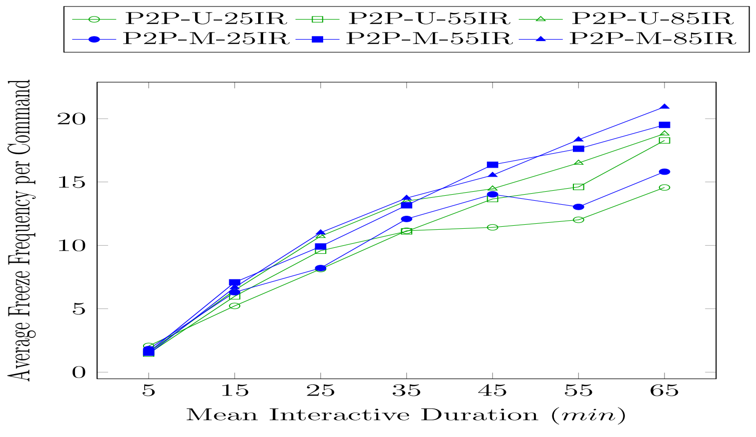


(f)

Figure 5.6: Change the mean interactive duration and enable high-speed fasts



(g)



(h)

Figure 5.6: Change the mean interactive duration and enable high-speed fasts

ratio than the P2P-U across all simulations under same installation rate.

Figure 5.6c shows the download ratio has increased gradually when the mean interactive duration is increased. Besides, the P2P-U still acquire most of the chunks from the PB channels when installation rate is 25%.

Figure 5.6g and Figure 5.6h show the freeze time and frequency are risen with the increase in the mean interactive duration. Moreover, the P2P-M achieves better performance when the installation rate is 25%. However, when the installation rate reaches 55% and 85%, no obvious difference can be observed. Similar patterns are also found in Figure 5.6e and Figure 5.6f.

In conclusion, the P2P-M has higher query acceptance ratio than the P2P-U across all simulations that could deliver better performance in terms of startup delay, freeze time, and freeze frequency when the installation rate is 25%. However, when the installation rate is increased to 55% and 85%, the difference between P2P-M and P2P-U becomes minimal.

5.4 Conclusion

In this chapter, the potentials of using the multicast in the P2P network is examined. The proposed system is based on the PB-P2P implementation developed in Chapter 4. The system allows the P2P-M server to satisfy multiple requests in single upload.

Computer simulations are conducted with a simple segmentation scheme that divides the video into equal-length segments and chunks. The result shows the use of the multicast can boost the performance of the PB-P2P system when the number of free P2P server is limited.

Studies are planned to improve the batching and queueing policies that aims to increase the average number of P2P-M clients served in each P2P-M transmission. We believe that, with these well-designed policies, the P2P-M can achieve an excellent performance in all situations that might provide an incentive to those network operators to support the multicast in production network better.

Chapter 6

Conclusion and Future Work

Video-on-Demand (VoD) service makes the access to various kinds of videos easy through the network. A VoD user can watch any videos provided by the service provider at anytime and anywhere with the support of various interactive commands.

Traditionally, the VoD services are built based on the Client-Server (CS) model and deliver the video by unicast. The required resources are directly proportional to the number of clients in service that could greatly increase the cost.

With the advance of the technologies and the improvement of the network infrastructure, alternative solutions have been proposed and studied. The building blocks of these solutions include Periodic Broadcasting (PB), Scheduled Multicast (SM), Content Delivery Network (CDN), and Peer-to-Peer (P2P). Each of these building blocks has its own strengths and weaknesses that motivates us to study the potential on combining these building blocks for a robust and cost-effective large-scale VoD system. As a result, a number of contributions have been made and presented in this thesis.

We first propose and investigate the performance of three possible enhancements, named Video Block Broadcasting, Peer-to-Peer, and Time-Shifting, on a VoD system designed for Wireless Mesh Network (WMN) through a series of computer simulations. The results show that each of these enhancements improves the system performance in different degrees that encourages us to further study the potential on combining these enhancements.

A PB-P2P VoD system is then proposed which is a hybrid of PB and P2P mesh pull. The system is implemented in ns-3. Computer simulations are conducted that examine the system in various situations in details. The simulation results show that the system can achieve a higher performance compared with the pure PB or P2P VoD system in terms of startup delay, freeze time, and freeze frequency.

Since both PB and P2P techniques have good scalability, the PB-P2P VoD system would be a good candidate on building a large-scale VoD system. However, in practice, not every user would contribute his (her) resources due to various reasons. The power of the P2P technique could be greatly reduced. As a result, the final work of this thesis is to investigate the possibility on using the multicast in the P2P network. Patches, along with the simple queuing and batching policies, are applied to the implementation of the PB-P2P VoD system. Computer simulations show that the use of the multicast could greatly increase the availability of the P2P servers, and therefore, improve the quality of the VoD service.

This thesis presents a potential solution on building a robust and cost-effective large-scale VoD system. However, there are still some issues that need to be addressed. For example, the use of the P2P could reduce the startup delay but it also shortens the buffering time that some clients would experience a little freeze when first chunk is finished. Therefore, a buffering algorithm is needed to estimate the suitable time to start the playback once the beginning chunks are downloaded. Moreover, a suitable segmentation scheme for the PB-P2P VoD system that can work well with various interactive commands is also required. Especially, the video browsing behaviour is commonly observed in the user behaviour studies. Furthermore, the algorithms to increase the PB to P2P download ratio are also critical to be network friendly.

Besides, there are some practical problems needed to be tackled. For example, a better solution to collect and analyze the raw data of the experiments is needed. Manual operations should be avoided as much as possible. Optimization on the simulation model is another challenging task which aims to reduce the time and resources demanded by each experiment. Distributed approach may be one of the potential solutions. Finally, a handy visualizer is also needed. Instant feedback from the visualizer could be a useful tool on studying and validating the behaviours of the simulation model.

References

[1] K. T. Sze, K. M. Ho, and K. T. Lo, *IAENG Transactions on Engineering Technologies: Special Issue of the International MultiConference of Engineers and Computer Scientists 2012*, ch. Efficient Video Streaming Over Wireless Mesh Networks, pp. 353–366. Dordrecht: Springer Netherlands, 2013.

[2] K. T. Sze, K. T. Lo, and J. Feng, *Encyclopedia of Multimedia*, ch. Multimedia Streaming in Wireless Mesh Networks. Springer, 2016. Accepted to appear.

[3] K. T. Sze and K. T. Lo, "A Hybrid PB-P2P Video-on-Demand System Minimizing Startup Delay and Supporting Interactive Commands at Best Effort." Under submission.

[4] K. T. Sze and K. T. Lo, "An Enhancement of a Hybrid PB-P2P Video-on-Demand System by Deploying Multicast in Peer-to-peer Network." In preparation.

[5] K. T. Sze, K. M. Ho, and K. T. Lo, "Supporting Video-on-Demand Services over Wireless Mesh Network," in *Proceedings of International Conference on Computers, Communications, Control and Automation*, vol. 1, pp. 357–360, February 2011.

[6] K. T. Sze, K. M. Ho, and K. T. Lo, “Performance Evaluation of a Video Broadcasting System over Wireless Mesh Network,” in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, pp. 386–390, March 2012.

[7] H. Abrahamsson and M. Nordmark, “Program popularity and viewer behaviour in a large tv-on-demand system,” in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC ’12, (New York, NY, USA), pp. 199–210, ACM, 2012.

[8] J. Esch, “Peer-to-Peer media streaming: Insights and new developments,” *Proceedings of the IEEE*, vol. 99, pp. 2087–2088, Dec 2011.

[9] “Hulu homepage.” `https://www.hulu.com/`. Accessed: 2017-02-22.

[10] “Netflix homepage.” `https://www.netflix.com/hk-en/`. Accessed: 2016-03-27.

[11] “PPS homepage.” `http://www.pps.tv/`. Accessed: 2016-04-01.

[12] “PPTV homepage.” `http://www.pptv.com/`. Accessed: 2016-04-01.

[13] “Tencent homepage.” `http://v.qq.com/`. Accessed: 2016-04-01.

[14] “Tudou homepage.” `www.tudou.com`. Accessed: 2016-04-01.

[15] “YouKu homepage.” `http://www.youku.com/`. Accessed: 2016-04-01.

[16] “Youtube homepage.” `https://www.youtube.com/`. Accessed: 2016-04-01.

[17] B. Li, Z. Wang, J. Liu, and W. Zhu, “Two decades of internet video streaming: A retrospective view,” *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, pp. 33:1–33:20, Oct. 2013.

[18] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, “Understanding user behavior in large-scale video-on-demand systems,” *SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 333–344, Apr. 2006.

[19] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, “Challenges, design and analysis of a large-scale P2P-VoD system,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 375–388, Aug. 2008.

[20] L. Chen, Y. Zhou, and D. M. Chiu, “A study of user behavior in online VoD services,” *Computer Communications*, vol. 46, pp. 66 – 75, 2014.

[21] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, “Analysis and characterization of a video-on-demand service workload,” in *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys ’15, (New York, NY, USA), pp. 189–200, ACM, 2015.

[22] I. Ullah, G. Doyen, G. Bonnet, and D. Gaiti, “A survey and synthesis of user behavior measurements in P2P streaming systems,” *Communications Surveys Tutorials, IEEE*, vol. 14, pp. 734–749, Third 2012.

[23] Y. Chen, Y. Liu, B. Zhang, and W. Zhu, “On distribution of user movie watching time in a large-scale video streaming system,” in *2014 IEEE International Conference on Communications (ICC)*, pp. 1825–1830, June 2014.

[24] C. Huang, J. Li, and K. W. Ross, “Can internet video-on-demand be profitable?,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 133–144, Aug. 2007.

[25] Z. Fei, I. Kamel, S. Mukherjee, and M. H. Ammar, “Providing interactive functions for staggered multicast near video-on-demand systems,”

in *Multimedia Computing and Systems, 1999. IEEE International Conference on*, vol. 2, pp. 949–953 vol.2, Jul 1999.

[26] J. B. Kwon and H. Y. Yeom, "Providing VCR functionality in staggered video broadcasting," *IEEE Transactions on Consumer Electronics*, vol. 48, pp. 41–48, Feb 2002.

[27] T. Viswanathan, S.and Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, vol. 4, no. 4, pp. 197–208, 1996.

[28] L.-S. Juhn and L.-M. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting*, vol. 43, pp. 268–271, Sep 1997.

[29] Z. Fei, M. Ammar, I. Kamel, and S. Mukherjee, "An active buffer management technique for providing interactive functions in broadcast video-on-demand systems," *Multimedia, IEEE Transactions on*, vol. 7, pp. 942–950, Oct 2005.

[30] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, no. 3, pp. 112–121, 1996.

[31] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optimal batching policies for video-on-demand storage servers," in *Multimedia Computing and Systems, 1996., Proceedings of the Third IEEE International Conference on*, pp. 253–258, Jun 1996.

[32] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: a generalized batching technique for video-on-demand systems," in *Multimedia Com-*

puting and Systems '97. Proceedings., IEEE International Conference on, pp. 110–117, June 1997.

[33] W. Liao and V. O. K. Li, "The split and merge (sam) protocol for interactive video-on-demand systems," in *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, vol. 3, pp. 1349–1356 vol.3, Apr 1997.

[34] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proceedings of the Sixth ACM International Conference on Multimedia*, MULTIMEDIA '98, (New York, NY, USA), pp. 191–200, ACM, 1998.

[35] G. Zhang, W. Liu, X. Hei, and W. Cheng, "Unreeling xunlei kankan: Understanding hybrid cdn-p2p video-on-demand streaming," *IEEE Transactions on Multimedia*, vol. 17, pp. 229–242, Feb 2015.

[36] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, pp. 68–74, Nov 2003.

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

[38] A. Passarella, "A survey on content-centric technologies for the current internet: CDN and P2P solutions," *Computer Communications*, vol. 35, no. 1, pp. 1 – 32, 2012.

[39] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. L. Zhang, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *INFOCOM, 2012 Proceedings IEEE*, pp. 1620–1628, March 2012.

[40] Y. Guo, K. Suh, J. Kurose, and D. Towsley, “P2Cast: Peer-to-peer patching scheme for VoD service,” in *Proceedings of the 12th International Conference on World Wide Web*, WWW ’03, (New York, NY, USA), pp. 301–309, ACM, 2003.

[41] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, “Distributing streaming media content using cooperative networking,” in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV ’02, (New York, NY, USA), pp. 177–186, ACM, 2002.

[42] B. Cohen, “Incentives build robustness in bittorrent,” 2003.

[43] J. F. Paris and P. Shah, “Peer-to-peer multimedia streaming using bittorrent,” in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pp. 340–347, April 2007.

[44] J. Postel, “Internet protocol.” RFC 791, Sept. 1981.

[45] S. Deering, “Host extensions for IP multicasting.” RFC 1112, Aug. 1989.

[46] W. Fenner, “Internet group management protocol, version 2.” RFC 2236, Nov. 1997.

[47] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, “Internet group management protocol, version 3.” RFC 3376, Oct. 2002.

[48] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1456–1471, Oct 2002.

[49] J. Postel, “Transmission control protocol.” RFC 793, sep 1981.

[50] R. Braden, “Requirements for internet hosts - communication layers.” RFC 1122, Oct. 1989.

[51] T. Socolofsky, “TCP/IP tutorial.” RFC 1180, Jan. 1991.

[52] “Napster homepage.” `http://gb.napster.com/`. Accessed: 2016-06-20.

[53] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable application layer multicast,” in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM ’02, (New York, NY, USA), pp. 205–217, ACM, 2002.

[54] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “Splitstream: High-bandwidth multicast in cooperative environments,” *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 298–313, Oct. 2003.

[55] V. Venkataraman, K. Yoshida, and P. Francis, “Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast,” in *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pp. 2–11, Nov 2006.

[56] X. Zhang, J. Liu, B. Li, and Y. S. P. Yum, “Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming,” in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, pp. 2102–2111 vol. 3, March 2005.

[57] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, “Chainsaw: Eliminating trees from overlay multicast,” in *Proceedings of the 4th International Conference on Peer-to-Peer Systems*, IPTPS’05, (Berlin, Heidelberg), pp. 127–140, Springer-Verlag, 2005.

[58] C. Dana, D. Li, D. Harrison, and C. n. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4, Oct 2005.

[59] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, (New York, NY, USA), pp. 89–100, ACM, 1997.

[60] J.-F. Paris, S. Carter, and D. Long, "A hybrid broadcasting protocol for video on demand," 1999.

[61] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *Multimedia Computing and Systems, 1996., Proceedings of the Third IEEE International Conference on*, pp. 118–126, Jun 1996.

[62] L.-S. Juhn and L.-M. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Broadcasting*, vol. 44, pp. 100–105, Mar 1998.

[63] K. A. Hua, M. A. Tantaoui, and W. Tavanapong, "Video delivery technologies for large-scale deployment of multimedia applications," *Proceedings of the IEEE*, vol. 92, pp. 1439–1451, Sept 2004.

[64] J. Choi, A. Reaz, and B. Mukherjee, "A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes," *Communications Surveys Tutorials, IEEE*, vol. 14, pp. 156–169, First 2012.

[65] A. Hu, "Video-on-demand broadcasting protocols: a comprehensive study," in *INFOCOM 2001. Twentieth Annual Joint Conference of*

the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 1, pp. 508–517 vol.1, 2001.

[66] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, pp. S23–S30, Sept 2005.

[67] N. Nandiraju, D. Nandiraju, L. Santhanam, B. He, J. Wang, and D. P. Agrawal, "Wireless mesh networks: Current challenges and future directions of web-in-the-sky," *IEEE Wireless Communications*, vol. 14, pp. 79–89, August 2007.

[68] S. M. Faccin, C. Wijting, J. Kenckt, and A. Damle, "Mesh WLAN networks: concept and system design," *IEEE Wireless Communications*, vol. 13, pp. 10–17, April 2006.

[69] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "IEEE 802.11s: The WLAN mesh standard," *IEEE Wireless Communications*, vol. 17, pp. 104–111, February 2010.

[70] M. J. Lee, J. Zheng, Y.-B. Ko, and D. M. Shrestha, "Emerging standards for wireless mesh technology," *IEEE Wireless Communications*, vol. 13, pp. 56–63, April 2006.

[71] "IEEE 802.16 homepage." `http://www.ieee802.org/16/`. Accessed: 2016-05-20.

[72] "IEEE 802.11 homepage." `http://www.ieee802.org/11/`. Accessed: 2016-05-20.

[73] L. Verma, M. Fakharzadeh, and S. Choi, "Wifi on steroids: 802.11ac and 802.11ad," *IEEE Wireless Communications*, vol. 20, pp. 30–35, December 2013.

[74] R. C. Carrano, L. C. S. Magalhaes, D. C. M. Saade, and C. V. N. Albuquerque, "IEEE 802.11s multihop MAC: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 13, pp. 52–67, First 2011.

[75] "IEEE 802.15 homepage." `http://www.ieee802.org/15/`. Accessed: 2016-05-20.

[76] "open80211s homepage." `www.o11s.org`. Accessed: 2016-05-20.

[77] "OLPC homepage." `http://one.laptop.org/`. Accessed: 2016-05-20.

[78] "Meraki homepage." `http://meraki.com`. Accessed: 2016-05-20.

[79] O. Bejarano, E. W. Knightly, and M. Park, "IEEE 802.11ac: from channelization to multi-user MIMO," *IEEE Communications Magazine*, vol. 51, pp. 84–90, October 2013.

[80] H. Zhai, J. Wang, and Y. Fang, "Distributed packet scheduling for multihop flows in ad hoc networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 2, pp. 1081–1086 Vol.2, March 2004.

[81] X. Cheng, P. Mohapatra, S.-J. Lee, and S. Banerjee, "Performance evaluation of video streaming in multihop wireless mesh networks," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, (New York, NY, USA), pp. 57–62, ACM, 2008.

[82] "ns-3 homepage." `https://www.nsnam.org/`. Accessed: 2015-12-15.

[83] "ns-3 publications." `https://www.nsnam.org/wiki/Papers`. Accessed: 2015-12-15.

[84] “ns-3 licensing.” `https://www.nsnam.org/developers/contributing-code/licensing/`. Accessed: 2015-12-15.

[85] “ns-3 installation.” `https://www.nsnam.org/wiki/Installation`. Accessed: 2015-12-15.

[86] “ns-2 homepage.” `http://www.isi.edu/nsnam/ns/`. Accessed: 2015-06-15.

[87] E. Weingärtner, H. Vom Lehn, and K. Wehrle, “A performance comparison of recent network simulators,” in *Proceedings of the 2009 IEEE International Conference on Communications*, ICC’09, (Piscataway, NJ, USA), pp. 1287–1291, IEEE Press, 2009.

[88] “Python homepage.” `https://www.python.org/`. Accessed: 2016-06-20.

[89] “Adding a new module to ns-3.” `https://www.nsnam.org/docs/manual/html/new-modules.html`. Accessed: 2016-06-20.

[90] W.-F. Poon, K.-T. Lo, and J. Feng, “Performance study on implementation of VCR functionality in staggered broadcast video-on-demand systems,” *Journal of Systems and Software*, vol. 75, no. 12, pp. 95 – 107, 2005. Software Engineering Education and Training.

[91] H.-M. Tsai, J.-W. Ding, S.-C. Cheng, and Y.-M. Huang, “Providing continuous VCR function with interpolated active buffer management for near VOD system on ubiquitous multimedia environment,” in *Ubi-Media Computing, 2008 First IEEE International Conference on*, pp. 231–236, July 2008.

[92] Y. W. Wong, J. Y. B. Lee, V. O. K. Li, and G. S. H. Chan, “Supporting interactive video-on-demand with adaptive multicast streaming,” *IEEE*

Transactions on Circuits and Systems for Video Technology, vol. 17, pp. 129–142, Feb 2007.

[93] A. Carlier, V. Charvillat, and W. T. Ooi, "A video timeline with bookmarks and prefetch state for faster video browsing," in *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, (New York, NY, USA), pp. 967–970, ACM, 2015.

[94] A. Chadagorn, I. Khalil, C. Cameron, and Z. Tari, "PileCast: Multiple bit rate live video streaming over BitTorrent," *Journal of Network and Computer Applications*, vol. 39, pp. 167 – 178, 2014.

[95] H. Febiansyah, D. Khairani, and J. B. Kwon, "Peer-assisted adaptation in periodic broadcasting of videos for heterogeneous clients," *Peer-to-Peer Networking and Applications*, vol. 6, no. 3, pp. 277–293, 2012.

[96] V. Gopalakrishnan, B. Bhattacharjee, K. K. Ramakrishnan, R. Jana, and D. Srivastava, "CPM: Adaptive video-on-demand with cooperative peer assists and multicast," in *INFOCOM 2009, IEEE*, pp. 91–99, April 2009.

[97] S. Ghandeharizadeh, B. Krishnamachari, and S. Song, "Placement of continuous media in wireless peer-to-peer networks," *IEEE Transactions on Multimedia*, vol. 6, pp. 335–342, April 2004.

[98] "HOWTO use ns-3 with other libraries." `https://www.nsnam.org/wiki/HOWTO_use_ns-3_with_other_libraries`. Accessed: 2016-03-27.

[99] "PyViz wiki." `https://www.nsnam.org/wiki/PyViz`. Accessed: 2016-01-20.

[100] "Python bindings." `https://www.nsnam.org/wiki/Python_bindings`. Accessed: 2016-01-20.

[101] "Top rated movies." `http://www.imdb.com/chart/top`. Accessed: 2016-03-01.

[102] Y. Kim, Y. Kim, H. Yoon, and I. Yeom, "Peer-assisted multimedia delivery using periodic multicast," *Information Sciences*, vol. 298, pp. 425 – 446, 2015.