The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Scalable Transmission Solutions for Media Streaming in Heterogeneous Network Environment

by

HO King Man

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

October 2007

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which as been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

________________________(Signed)

_______HO King Man_______(Name of student)

Abstract

of the thesis entitled

"Scalable Transmission Solutions for Media Streaming in Heterogeneous Network Environment"

submitted by HO King Man

for the degree of Doctor of Philosophy

at The Hong Kong Polytechnic University in Oct 2007

With the advances in high-performance networks and digital video technology, different Video-on-Demand (VoD) systems have come into practice in recent years. With this technology, users can playback the video content without waiting for the entire file to arrive and also enjoy the flexible control of video playback. However, deployment of a large-scale VoD system requires an enormous amount of server and network resources in order to archive hundreds/thousands of videos for customers and handle an enormous number of concurrent video streams. Thus, one of the most challenging design aspects of a VoD system is how to deliver videos to a large community economically. On the other hand, clients can connect to the network with different communication technologies that the downstream rates vary from 56kbps to 10Mbps or above. Heterogeneity of the network environment is another design issue that should be addressed in order to meet different capabilities of the clients' devices in the system. This thesis presents the results of our work on development and analysis of a VoD system in heterogeneous network environment.

In this dissertation, we first investigate a feasible solution for building a unified model for a VoD system in heterogeneous environment that integrates the technologies of proxy caching, video broadcasting, replicated/layering videos. With this model, we study various design options and perform system dimensioning. Then, we extend the framework by developing a complementary approach using both video replication and layering for video streaming such that the system performance can be further improved. On the other hand, we also investigate the benefit of renegotiation about video quality when the system resources cannot satisfy the request from a client.

Although the hierarchical architecture approach can greatly improve the system performance, the video server is still the bottleneck in such client-server architecture. To tackle this problem, we consider using the peer-to-peer (P2P) approach to address the issues of system scalability. In this regard, we first propose a peer-to-peer batching (PPB) policy to exploit the multicast/broadcast capability of the network and P2P paradigm to efficiently deliver video data to a large number of clients. The objective of this policy is to consider the trade-off between the network bandwidth requirement for P2P transmission and multicast delivery such that the overall transmission cost can be minimized. In addition, we also develop a fault tolerance and recovery procedure for this policy to take the peer's departure behaviour into account for better management of the system resources. It is found that this policy can leverage the workload of the central server about 50%. To further improve the scalability of the system, we also propose a distributed scheme to disperse the duty of multicast/broadcast delivery from the central server to a number of high-end machines denoted peer servers, which have a higher storage and bandwidth capacity than normal peers. We also show how this framework can be applied to the existing broadcasting protocols to take benefits from the P2P paradigm. To investigate the system behaviour of such framework, another unified model is also developed. With this model, we can have a better understanding of the system dynamics. This model also provides guidelines for efficient management of system resources and realization of VoD services.

List of Publications

Book Chapters and International Journal Papers

[1] K.M. Ho, K.T. Lo and J. Feng, "Multimedia Streaming on the Internet," *Encyclopedia of Multimedia*, edited by B.Furth, ISBN: 0-387-24395-X, pp.595-603, February 2006, Springer, U.S.A.

[2] K.M. Ho, W.F. Poon and K.T. Lo, 'Performance study of large-scale video streaming services in highly heterogeneous environment,' *IEEE Transactions on Broadcasting*, vol.53, no.4, pp.763-773, Dec.2007.

[3] K.M. Ho, K.T. Lo and J. Feng, "Transmission Strategies for Media Streaming in Peer-to-Peer (P2P) Networks," *Encyclopedia of Wireless and Mobile Communications*, edited by B.Furht, ISBN: 9781420043266, 2008, CRC Press (Chapter accepted, to be published).

[4] K.M. Ho and K.T. Lo, "Large Scale Multimedia Streaming in Heterogeneous Network Environments," *Encyclopedia of Multimedia*, 2nd Edition, edited by B.Furth, Springer, U.S.A. (Chapter accepted, to be published).

[5] K.M. Ho, W.F. Poon and K.T. Lo, "Video-on-Demand Systems with Cooperative Clients in Multicast Environment," revised version submitted, *IEEE Trans. System*

and Video Technology, 2007.

[6] K.M. Ho, and K.T. Lo, "Video Broadcasting Scheme for Streaming Applications with Cooperative Clients," submitted for publication, *IEEE Trans. Broadcasting*, 2007.

[7] K.M. Ho and K.T. Lo, "Peer-to-Peer Video-on-Demand System in Broadcast Environment," submitted for publication, *IEEE Trans. Parallel and Distributed System*, 2007.

[8] K.M. Ho and K.T. Lo, "Large-Scale Video Streaming Services for Heterogeneous Environment," submitted for publication, *Journal of Visual Communications and Image Representation*, 2007.

International Conference Papers

[9] K.M. Ho, W.F. Poon and K.T. Lo, "Peer-to-Peer Batching Policy for Video-on-Demand System," *Proceedings IEEE First International Conference on Communications and Networking in China 2006 (ChinaCom '06)*, pp. 1-6, Oct. 2006, Beijing, China.

[10] K.M. Ho, W.F. Poon and K.T. Lo, "Peer-to-Peer Broadcasting Scheme for Video-on-Demand System," *Proceedings IEEE International Symposium on Communications and Information Technologies 2006 (ISCIT '06)*, pp. 434-439, Oct. 2006, Bangkok, Thailand.

[11] K.M. Ho, W.F. Poon and K.T. Lo, "Enhanced Peer-to-Peer Batching Policy for

Video-on-Demand System," *Proceedings IEEE International Symposium on Communications and Information Technologies 2006 (ISCIT '06)*, pp. 148-151, Oct. 2006, Bangkok, Thailand.

[12] K.M. Ho, K.T. Lo and J. Feng, "Design of a Peer-to-Peer Video-on-Demand System with the Consideration of Fault Exception," *Proceedings IEEE International Conference on Multimedia and Expo 2007 (ICME 2007)*, pp. 1071-1074, Jul. 2007, Beijing, China.

[13] K.M. Ho and K.T. Lo, "Design of Decentralized Video-on-Demand System with Cooperative Clients in Multicast Environment," *Proceedings of Pacific-Rim Conference on Multimedia (PCM 2007)*, pp. 401-404, Dec. 2007, Hong Kong, China.

[14] K.M. Ho, W.F. Poon and K.T. Lo, "Investigating the Performance for Hierarchical Video-on-Demand System in Heterogeneous Environment," *Proceedings IEEE The International Conference on Information Networking 2008 (ICOIN 2008)*, Jan 2008, Busan, Korea. (to be published)..

[15] K.M. Ho, K.T. Lo and J. Feng, "Cooperative Transmission strategy for Video-on-demand System," *Proceedings IEEE The International Conference on Information Networking 2008 (ICOIN 2008)*, Jan 2008, Busan, Korea. (to be published).

[16] K.M. Ho and K.T. Lo, "A Simple Model for Peer-to-Peer Video-on-Demand System in Broadcast Environment," *Proceedings IEEE The International Conference on*

Information Networking 2008 (ICOIN 2008), Jan 2008, Busan, Korea. (to be published).

[17] K.M. Ho and K.T. Lo, "Provision of Fault-Tolerant Mechanism for Peer-to-Peer Video-on-Demand System in Broadcast Environment," *Proceedings IEEE International Conference on Circuits and Systems for Communications (ICCSC 2008)*, May 2008, Shanghai, China. (to be published).

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to those people who have provided me valuable opinions and encouragement on my research work during these research years.

Special thanks and appreciation are given to my supervisor, Dr. K.T. Lo for his feasible research guidance and directions on my research topic which has made this thesis possible. His continuous support and inspiration made my research studies enjoyable. I would also like to express my thanks to Dr. W.F. Poon for his invaluable advice. His rich knowledge in video-on-demand system contributed a lot to this work.

I would also express my gratitude to all the members and staff in the Department of Electronic and Information Engineering and the Centre of Multimedia Signal Processing for providing me a comfortable and pleasant working environment for my research studies, and the Hong Kong Polytechnic University for their financial support to carry out my research work.

Heartily, I am grateful to my family and my friends, for their love, trust and support throughout all these years.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ASP	: Active Serving Peer
BM	: Buffer Map
BP	: Bitplane
CBR	: Constant Bit Rate
CDN	: Content Distribution Network
DPCS	: Distributed Probabilistic Clock Synchronization
ECC:	: Erasure Correcting Codes
HB	: Harmonic Broadcasting
DCT	: Discrete Cosine Transform
DHT	: Distributed Hash Table
EWMA	: Exponentially Weighted Moving Average
FCFS	: First-Come-First-Serve
FEC	: Forward Error Code
FGS	: Fine Granularity Scalability
GL	: Group Leader
GOP	: Group-of-Picture

I/O	: Input/ Output
LAN	: Local Area Network
MB	: Marcoblock
MPEG	: Motion Picture Experts Group
MSB	: Most Significant Bits
MQLF	: Maximum-Queue-Length-First
MFQLF	: Maximum-Factored-Queue-Length-First
MTTF	: Mean Time to Failure
MTTR	: Mean Time to Repair
N-VoD	: Near video-on-demand
NC	: Network coding
NIC	: Network interface card
No-VoD	: Broadcast
PA	: Packet Assignment
PALS	: Peer-to-Peer Adaptive Layered Streaming
PB	: Pyramid Broadcasting
PC	: Peer Client
PBPB	: Permutation-Based Pyramid Broadcasting
PPA	: Packet Partition Algorithm

PPV : Pay-per-view

PS : Peer Server

PSG : Peer Server Group

P2P : Peer-to-Peer

PPC : Peer-to-Peer Channel

QA : Quality Adaptation

QoS : Quality-of-Service

Q-VoD : Quasi Video-on-Demand

RAA : Rate Allocation Algorithm

RTSP : Real-Time Streaming Protocol

SB : Stagger Broadcasting

SKB : Skyscraper Broadcasting

SIP : Session Initiation Protocol

SPC : Serving Peer Client

T-VoD : True video-on-demand

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

VBR : Variable Bit Rate

VCD : Video Compact Disc

VoD : Video-on-Demand

WAN : Wide Area Network

Chapter 1

Introduction

1.1 Introduction

The Internet has seen miraculous growth since its appearance. Web browsing and file transfer are the dominant services provided through the Internet. However, these kinds of service providing information about text, pictures and document exchange are no longer satisfied the demand of clients. With the recent advances in digital technologies, such as high-speed networking, media compression technologies and fast computer processing power, more and more multimedia applications involving digital audio and video are come into practice on the Internet.

Video-on-demand (VoD) is an enabling technology [SINCO91] allowing geographically distributed users to interactively access video files from remote VoD servers. With this technology, users can playback the video content without waiting for the entire file to arrive and also enjoy the flexible control of video playback with VCR-like functions. Compared with conventional data communications, delivery of video data has more stringent requirements on network bandwidth, delay and loss [LU00]. However, the

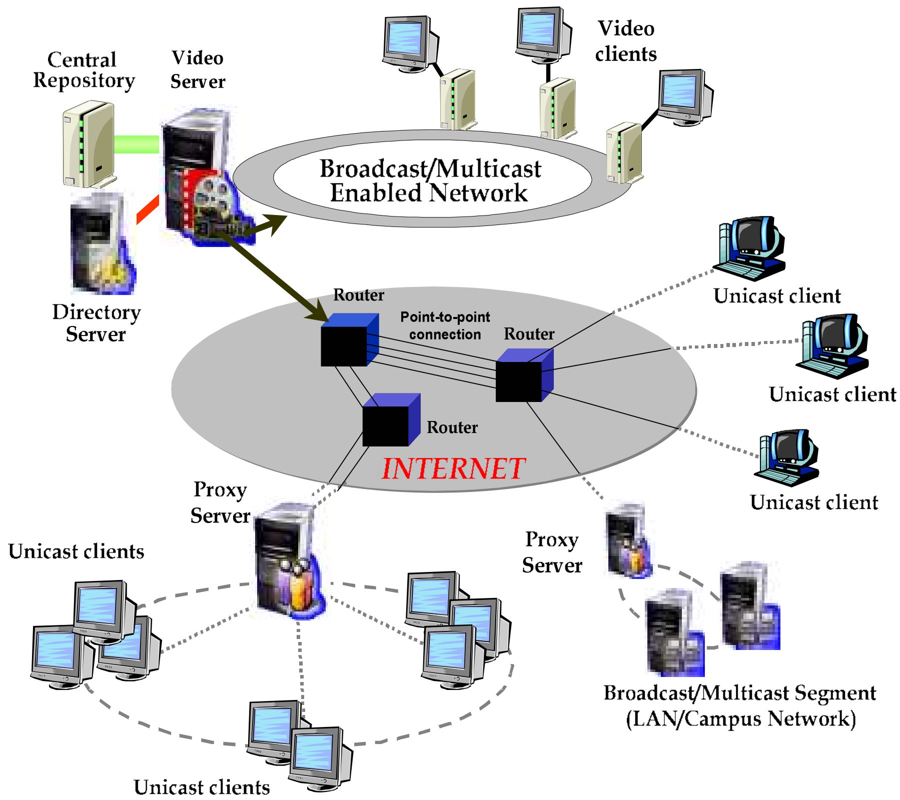


Figure 1. Architecture of a Typical Large-Scale VoD System

current Internet is inherently a packet-switched network that was not designed to handle isochronous (continuous time-based) traffic such as audio and video. The Internet can only provide best-effort services and has no guarantee on the quality of service (QoS) for audio and video transmission. In addition, a large-scale VoD system must have a large storage capacity to archive hundreds or even thousands of videos and a high performance I/O bandwidth to handle an enormous number of concurrent video streams. As a result, there are still many open issues in designing protocols and transmission strategies for VoD systems.

Figure 1 shows a typical large-scale VoD system architecture that basically consists of four main components: video server, directory server, proxy server and clients. While a number of clients connected via a local area network (LAN), wide area network (WAN) provides a mean to interconnect video servers and geographically-dispersed clients together. In order to effectively utilize the system resources, video objects should first be compressed and then stored in central repository. The video server is responsible for managing and allocating resources for requests from clients. It determines whether there are sufficient resources such as disk bandwidth and available network bandwidth to provide continuous transfer of data before accepting a request from the client. The server is also able to support both unicast and broadcast connections depending on the infrastructure of the underlying network. The client is only responsible for generating a request and receiving the acknowledgment to and from the video server. The interaction between the server and clients may follow proprietary mechanisms or well-defined open standards, such as real-time streaming protocol (RTSP) [SCHUL98]. When the server accepts the request, a dedicated channel is allocated between the server and client. Then, video data is forwarded along this channel to client which decodes the compressed video data and then renders video contents onto display. If the server does not have enough resource to support the service, the request will be blocked. Obviously, it is annoyed to clients when they are blocked by the system frequently. Nevertheless, due to the limited resources of the system, blocking may not be avoided.

In order to provide cost-effective and scalable solutions for a VoD system such that more clients can admit to the system, various designs have been proposed in terms of system architecture [THOUI07], bandwidth allocation [DAN95] and transmission schemes [HUA04]. Among these, the broadcasting and data sharing techniques exploit the broadcast capability of a network such that video contents are distributed along a number of video channels shared among clients. In addition, hierarchical architectures have also been explored to provide a cost effective implementation of a VoD system. In such hierarchical frameworks, video data are temporarily stored in a proxy server so that the workload of the central server can be significantly reduced. Recently, peer-to-peer (P2P) communications have become a popular alternative solution to support large-scale VoD services. Under this transmission strategy, each end-point called peer is operated as client and server simultaneously such that the bottleneck of the system is no longer at the server side. Nevertheless, most of the previous works mainly focused on reducing the bandwidth required for the VoD system and providing the VoD services in a homogeneous environment. In practical situations, clients can connect to the network, says Internet, with different communication technologies such as modem, ASDL and wireless link and their downstream rates vary from 56kbps to 10Mbps or above. In this thesis, we investigate a feasible solution for building an efficient hierarchical VoD system using proxy caching coupled with broadcasting and appropriate coding schemes in heterogeneous network environments. We also explore the feasibility of using the broadcast/multicast capability of the network coupled with P2P transmission strategy for video transmission. In addition, we develop generalized mathematical models to study the performance of our proposed frameworks.

1.2 Objective of the Thesis

Deployment of a large-scale VoD system requires an enormous amount of server and network resources. Thus, one of the most challenging aspects of a VoD design is how to deliver videos to a large community economically. Hierarchical architecture is one of the possible solutions for building a cost-effective VoD system. However, most of the previous researches mainly focused on homogeneous environment. Actually, clients can connect to the network with different communication technologies that the downstream rates vary from 56kbps to 10Mbps or above in practical situations. In this dissertation, we focus on the design issues of the scalability of a VoD system in heterogeneous environments. One of our research objectives is to investigate a feasible solution for building a unified model for VoD in heterogeneous environments by integrating the technologies of proxy caching, video broadcasting, replicated/layering videos. With this model, we can study various design options and perform system dimensioning.

Peer-to-Peer (P2P) and multicast/broadcast approaches are other common transmission strategies to provide scalable and cost efficient solutions for VoD services.

The former approach requires small server resources and provides a negligible delay to users, but the bandwidth requirement inside the network will be rapidly increased when more customers join the video session. On the other hand, if the system simply uses a multicast/broadcast scheme to deliver a video, customers will experience a noticeable delay before watching the video whereas the overall bandwidth requirement will not be significantly increased. Based on their natures of video transmission, we consider the trade-off between the network bandwidth requirement for P2P transmission and multicast/broadcast delivery. Hence, another focus of the thesis is to exploit the multicast/broadcast capability of the network and P2P paradigm to efficiently deliver video data to the clients.

1.3 Organization of the Dissertation

Following the introduction of the thesis, a brief review on different technological issues for a VoD system is given in Chapter 2. In this chapter, we first present various existing video compression methods especially the scalable/layered coding techniques for video streaming applications. Then, we describe different existing VoD architectures as well as various transmission strategies for VoD system. We also highlight the strength and weakness of these approaches.

We first investigate possible solutions for building a large-scale VoD system in a heterogeneous network environment in Chapter 3. In this chapter, we explore the impact of the broadcasting schemes coupled with proxy caching for video transmission. To meet different clients' bandwidth constraints, videos are encoded into a number of quality levels with replication or layered encoding. We investigate the efficiency of the proposed framework by developing a mathematical model to effectively compare the performance of video replication and that of video layering for video streaming under different scenarios and parameter settings. The model can also be applied to different system configurations such as centralized/ distributed, unicast/broadcast as well as replication/layering, which can assist the system designer to study various design options and to perform system dimensioning.

Based on the analysis in Chapter 3, we have known that the layering approach is suitable for proxy caching and video broadcasting while replication is favorable to end-to-end transmission. The results introduce an interesting question whether the system performance can be further improved if both coding schemes are deployed in the system concurrently based on their natures. Therefore, in Chapter 4, we extend the framework in Chapter 3 by developing a complementary approach using both video replication and layering for video streaming such that the system performance can be further improved. On the other hand, the client will be blocked when the system resources cannot satisfy client's request in the previous proposed architecture. However, this request may still be accepted if the system resources can support a lower quality level of the requested video. Therefore, we also investigate the benefit of renegotiation about the video quality when the system resources are limited in this chapter.

After analytically investigating the performance of a hierarchical VoD system in heterogeneous network environments by our mathematics model, it can be concluded that the system performance can be significantly improved when video broadcasting technique and an appropriate coding scheme are used. Nevertheless, one problem to implement such framework is that the bottleneck of the system is still at the server side and thus such client-server architecture does still not scale very well. Thus, we turn our focus to another approach to address the issues of system scalability – peer-to-peer (P2P). We explore the feasibility of using the multicast delivery coupled with P2P transmission strategy for video streaming in Chapter 5. In this chapter, a new batching policy called peer-to-peer batching (PPB) is proposed. To provide an insensitive delay services with PPB, clients first construct a P2P network and use a cache-and-relay manner (i.e. chaining [SHEU97]) to deliver the leading portion of the video for the late-coming clients during the beginning of the video session. When the size of the P2P network is large enough, they will be grouped together and served by a multicast channel so that the network bandwidth will not be exhausted rapidly. Due to the dynamic nature of the P2P application, we also consider the issues of fault exception in our system framework. In addition, we develop a mathematical model to evaluate the performance analytically and determine the optimal state of the system such that the overall transmission cost of the system is minimized.

PPB can share the workload of the central server roughly by half. However, the central server still reserves a lot of resources for video multicasting. To further improve the scalability of the system, in Chapter 6, we extend the framework of PPB to develop a generic VoD system model that uses a P2P paradigm coupled with video broadcasting and CDN-like approach for video delivery. We also propose a distributed scheme to disperse the duty of multicast delivery from the central server to a number of high-end machines denoted peer servers. In the proposed framework, a video is first divided into two parts. The first part of the video is transmitted among peers by P2P such that customers can obtain the service in a short time. The second part of the video is periodically broadcasted by a number of peer servers having a higher storage space and bandwidth capacity than normal peers. This content delivery strategy allows the workload of the system disperse over the network and also takes a benefit from video broadcasting. We also consider the heterogeneity of outbound bandwidth of peers in this framework. To avoid the disruption of the service due to the dynamic of P2P applications, a central server is still deployed to provide a certain level of reliability. The focus of this chapter is to study the features of the proposed VoD architecture by analytical model. We examine how the partition of the video impacts on the system performance in terms of bandwidth requirement. Then, we investigate how various system parameters affect the proliferation of the system capacity as well as the usage of the central server resources. We also determine the minimum number of peer servers required for the system. This study allows us better understanding of system dynamics and provides guidelines for the management of design resources and realization of VoD services based on this architecture.

Finally, we draw a conclusion of the thesis and give directions for future research in Chapter 7.

Chapter 2

Literature Reviews

2.1 Introduction

In this chapter, an overview and literature survey for Video-on-Demand (VoD) systems are presented. Different technical issues related to the design of a VoD system are examined. In Section 2.2, we first briefly describe various existing video compression methods especially the scalable/layered coding techniques for video streaming applications. An overview of the typical architectures for VoD systems is presented in Section 2.3. We also describe different types of video services and various transmission strategies in this section. In Section 2.4, the P2P-based frameworks for media streaming will be discussed. Finally, we present several novel hybrid architectures for VoD services in Section 2.5. The problems encountered in the existing technologies will be highlighted.

2.2 Video Compression Techniques

As the large volume of raw video data imposes a stringent bandwidth and disk space requirement, compression is widely employed to achieve transmission and storage efficiency. MPEG-1 [ISO93] released in 1993 was the first video coding standard proposed by the ISO MPEG committee. It is targeted for the compression of CIF or SIF full-motion video at a rate of 1.5 Mbps for storage applications such as Video CD (VCD). It uses block-based discrete cosine transform (DCT) and motion compensation to reduce redundancies in spatial and temporal domain respectively [HASKE97]. Basically, MPEG-1 is a non-scalable compression scheme and thus it requires the clients with bandwidth which is identical to the compressed data rate. However, the system may need to serve different clients with different capabilities simultaneously. Therefore, multiple versions with identical content but at different compressed data rates should be supported by the system. This strategy is known as replication.

To support the quality adaptation in a video application according to the available bandwidth between the server and clients, the layered coding technique is an alternative solution to flexibly provide different qualities of video. With layered encoding, streams of different rates can be extracted from a single stream. The basic principle of layered coding is that the video is encoded in a base layer and one or more enhancement layers. The base layer can be decoded independently while the enhancement layers should be decoded cumulatively. That is, layer k should be decoded along with layer 1 to layer $k-1$. The base layer only provides a low or coarse quality but the enhancement layers contribute to the improvement of the video quality. Depends on the available bandwidth of the clients, they can request different layers of the encoded video to obtain different qualities of the video.

In general, more enhancement layers received higher video quality can be obtained. MPEG-2 [ISO96a, ISO96b] is the first video compression standard that can support scalable coding in MPEG series. MPEG-2 defines three types of scalability modes: signal-to-noise ratio (SNR), spatial and temporal. These allow a different set of tradeoffs in bandwidth, video resolution and overall implementation complexity. In the SNR scalability, different visual qualities of video can be extracted from different layers in a single stream. While the spatial scalability is targeted for the enhancement of the video resolution, the temporal scalability is focused on frame rate scalability. MPEG-2 allows a combination of the scalability modes that leads to the hybrid scalability.

As the encoder may not know the network condition, the scalable coding approach defined in MPEG-2 providing only a step-like quality enhancement may not be able to fully utilize the available bit-rate of the channel. On the other hand, the decoder may not be able to decode all the received data fast enough for reconstruction. Therefore, the objective of video coding for multimedia streaming is to optimize the video quality over a given bit-rate range instead of a given bit-rate. Also the bitstream should be partially decodable at any bit-rate within the bit-rate range to reconstruct with optimized quality [LI01]. To meet these demands, a new scalable coding mechanism, called fine granularity scalability (FGS) [ISO00] was proposed in MPEG-4. An FGS encoder compresses raw video data into two streams, base layer bitstream and enhancement layer bitstream. Similar to the traditional video encoder, it relies on two basic methods for compression: intra-frame DCT coding for reduction of spatial redundancy and inter-frame motion compensation for reduction of temporal redundancy. Different from MPEG-2 scalable encoder, the FGS encoder produces the enhancement stream using bitplane coding, which is achieved by coding the difference between the DCT coefficients on the reconstructed frame and the original frame and then extracting each bit from 64 DCT coefficients with same significant to form a bitplane (BP). Therefore, all the most significant bits (MSB) from the 64 DCT coefficients form BP-1 and all the second MSB form BP-2, and so on. With this coding technique, the encoder can truncate the bitstream of the enhancement layer anywhere to achieve continuous rate control.

As mentioned, video can be coded by either replication or layered coding to adapt clients' requirement as clients can connect to the network with different communication technologies such as modem, ASDL and wireless links and their downstream rates vary from 56kbps to 10Mbps. There is a common belief that layered coding is performed better than the replication approach. The main argument is that the replication method can cause large increase in the amount of storage and waste bandwidth by essentially duplicating the video content. However, the layering approach also has a flaw that it generates additional bandwidth overhead [KIMUR99] such that it actually requires more transmission bandwidth than the replication approach for the same quality level. Therefore, in order to clarify their efficiencies for video transmission in heterogeneous environment, we will develop an analytical model to compare these two coding approaches for a VoD system in Chapter 3.

2.3 Video-on-Demand (VoD) System

With the explosive growth of the Internet, the demand for various multimedia applications is rapidly increasing in recent years. Among different multimedia applications, Video-on-Demand (VoD) is playing a very important role. With VoD, customers can choose their desired video at arbitrary time they wish via public communication networks. Such systems are required to store several hundreds of videos as well as serve thousands of customers simultaneously. When a user wants to watch a video, he or she makes a request from a list of available programs and the video is ready to playback within a short time. During enjoying the video, the user may also generate a sequence of VCR commands to control the flow of playback. These simple interactions involve many complicated mechanisms and policies operating behind the user. In addition, transmission of video content requires a significant amount of disk storage and network bandwidth. In order to support a cost-effective and scalable solution for a large-scale VoD system, various designs have been proposed in terms of system architecture, transmission schemes and bandwidth allocation based on the nature of different video services.

2.3.1 Categories of Video Services

Video services can be classified into the following categories based on the amount of interactivity allowed and the scheduling policies of data delivery deployed [LITTL96, MA02]:

i. *Broadcast (No-VoD)*: This service is similar to the traditional television broadcasting system where a user is a passive participant and has no control over the video session. A user only selects a specific programme provided by the system to view.

ii. *Pay-per-view (PPV)*: This service is similar to the cable TV system where a user signs up and pays for specific programmes scheduled at predetermined times. In this case, a user still does not have any control right for a particular video session.

iii. *Near video-on-demand (N-VoD)*: Users arriving within a short time interval and requesting for the same video are served using one video stream. Users can select any movie at any time but the server still dominates the service in determining when to start the video. Since it is possible that multiple channels with the same programme skew in time, users can switch among these channels that simple VCR functions such as forward and reverse can be achieved.

iv. *Quasi video-on-demand (Q-VoD)*: It is a threshold-based N-VoD system in which users are grouped based on a threshold of interest. Users can perform rudimentary temporal control activities by switching to a different group.

v. *True video-on-demand (T-VoD)*: In this case, the user has complete control over the video session. Users can request and view any video at any time with full VCR capabilities including forward, reverse, freeze and random positioning. Allocating a dedicated stream to every user is the simple way to provide T-VoD service. However, it is not a cost-effective and scalable approach.

Obviously, No-VoD is the cheapest solution to provide video services. However, the user is constrained to watch the undesired programmes probably. In contrast, T-VoD is the most expensive solution but it provides the most ideal service that fulfills most of users' requirement on service customization. Thus, there is a trade-off between the system cost and user's expectation or requirement.

2.3.2 Examples of Current Video Streaming Applications

NOW TV (http://www.now-tv.com) and HKBN (http://www.hkbn.net) are two broadband TV service providers in Hong Kong, which provide PPV and N-VoD services for customers. Basically, the system architecture is based on a client-server approach with broadcast capability. In the system, each programme owns a dedicated broadcast channel to deliver the video contents over the network. Customers can enjoy their favor programmes by subscribing one of these broadcast channels through the remote controller. YouTube (http://www.youtube.com) is a video sharing website where users can upload, view and share video clips. Video playback technology of YouTube is based on Macromedia's Flash

Player 9 which supports H.264 video and HE-AAC audio. Video contents can be delivered by "Streamed via RTMP (Real-Time Message Protocol)" or "Progressive download via HTTP (Hyper Text Transmission Protocol) [BERME96]". While a special streaming server, the Flash Media Server, is required to support "Streamed via RTMP", the existing generic HTTP servers can still be used to support "Progressive download via HTTP". YouTube relies on progressive download to provide video services for users. Although it cannot offer a real-time broadcasting but it can provide the lowest cost solution for VoD with interactive function that allows users to seek to any part of the video before buffering is complete. PPLive (http://www.pplive.com) is a P2P streaming application created in Huazhong University of Science and Technology, People's Republic of China. It enables users to share video content with each other over the Internet by P2P manner. The technology behind it is similar to that of Bittorrent (http://www.bittorrent.com), where the users upload their cached video content and currently download their desired programs.

2.3.3 Architecture of a Video-on-Demand System

The VoD systems proposed in the last decade typically can be classified into four main architectures [THOUI07]: centralized, proxy-based, content distribution network (CDN) and peer-to-peer (P2P) architecture. The diagrams for these architectures of a VoD system are shown in Figure 2.1.

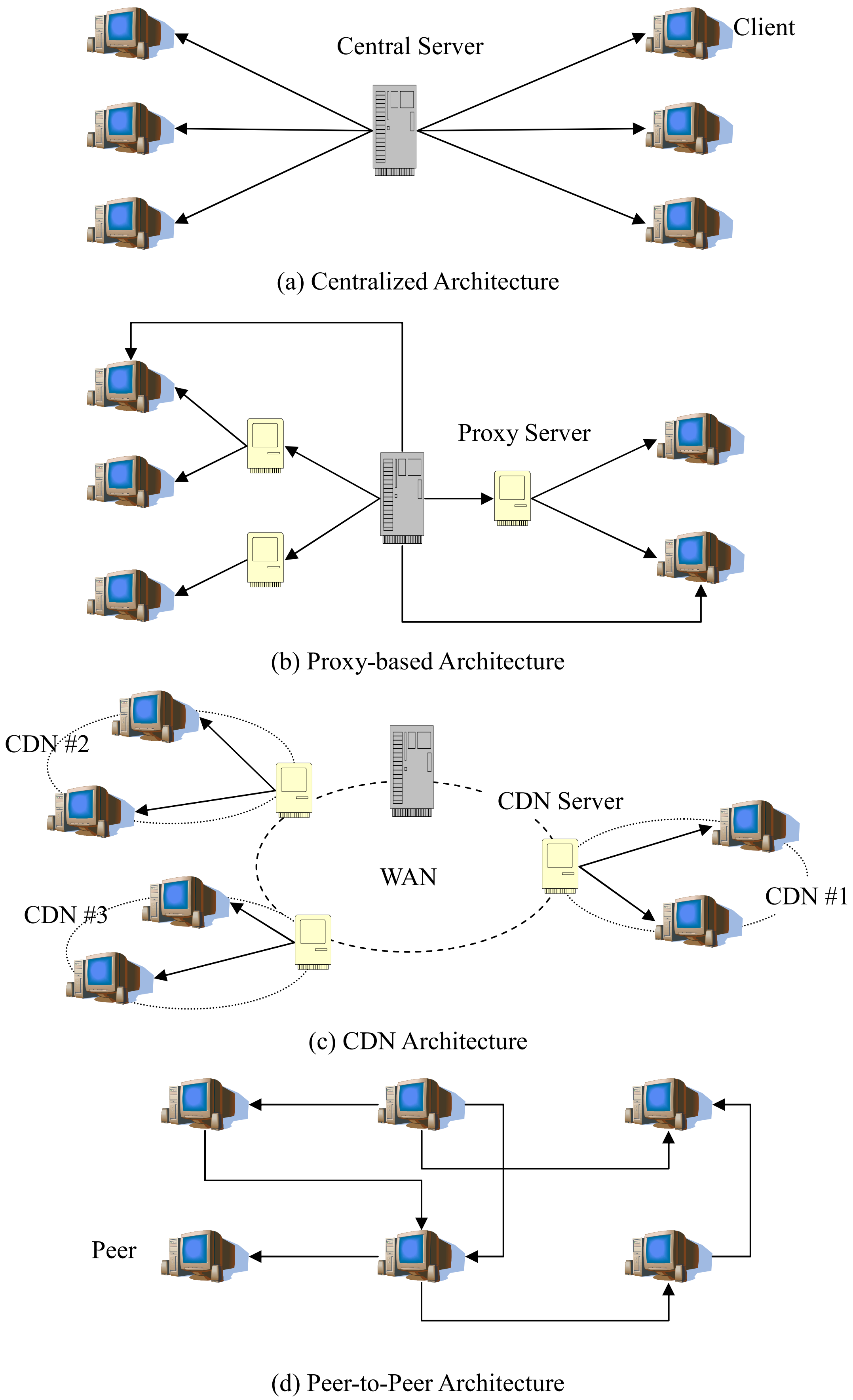


Figure 2.1. Architectures of a Video-on-Demand System

In a centralized architecture, the system consists of two main components: the central server and client (Figure 2.1a). The central server has a large storage space to store all the available videos for clients connected via a wide area network (WAN) or local area network (LAN). In such framework, all the requests from clients are handled at the central server. The request process starts with generating a request message from clients to the central server. In response to the client's request, the central server serves each request individually with a dedicated channel[1]. This operation is simple to implement. However, this architecture is excessively expensive and non-scalable because the bandwidth bottleneck of the central server limits the number of clients it can serve. Furthermore, the introduction of long service latencies is another critical factor affecting the system performance, which is especially significant when the video is transmitted over the WAN.

To leverage the workload of the central server and reduce the service latencies, an intermediate device called proxy is sit between the central server and clients (Figure 2.1b). In the proxy-based architecture, a portion of video is cached in the proxy. The request generated by a client is served by the proxy if it has a cached portion of the requested video. Meanwhile, the central server also delivers the uncached portion of the video to the client directly. Existing caching mechanisms can mainly be classified into four categories [LIU04]: sliding-interval caching, prefix caching, segment caching and rate-split caching. Sliding-interval [TEWAR98] caches the playback interval between two requests. Prefix

[1] The unit of server capacity required to sustain the playback of one video stream is referred to as a channel [HUA04].

caching [SEN99] divides the video into two parts named prefix and suffix. Prefix is the leading portion of the video which is cached in the proxy while suffix is the rest of the video which is stored in the central server. Upon receiving a client's request, the proxy delivers the prefix to the client, meanwhile, it also downloads the suffix from the central server and then relays to the client. Segment caching [CHEN04] generalizes the prefix caching by partitioning a video object into a number of segments. The proxy caches one or several segments based on the caching decision algorithm. In rate-split caching [ZHANG00], the central server stores the video frame with the data rate which is less than a threshold called cutoff rate. If the data rate of the video frame is higher than the cutoff rate, it is partitioned into two parts where the cutoff is the boundary such that the transmission rate of the central server can keep constant.

Content distribution network (CDN) is an extension of the proxy caching. In such architecture (Figure 2.1c), a number of CDN servers are deployed at the edge of the network core. Unlike proxy which only stores a portion of the video, a full copy of the video is replicated in each CDN server. Then, clients request the video from their closest CDN servers directly. This architecture significantly reduces the workload of the central server and provides a better quality of service (QoS) to clients.

In a P2P architecture, a number of clients denoted peers self-organize into an overlay network via unicast connections (Figure 2.1d). Each peer operates as client and server simultaneously that it retrieves what it requests from the overlay network and forwards/relays what it has to the overlay network as well. The peer requesting the service is denoted as requesting peer while the peer providing the service is denoted as supplying peer. To subscribe the service, a peer establishes one or several dedicated channel(s) to other peers in the system and caches the incoming video contents for subsequent peers. Since each peer contributes its own resources to the system, the system capacity is vastly proliferated compared to the previous architectures.

2.3.4 Transmission Strategies for Video-on-Demand System

Unicast is the simplest way to deliver video contents over a VoD system. Each client generates a request for the desired video to the server. Once this request has been accepted, the server will allocate a dedicated channel to this client who can acquire a full control right for this channel. Therefore, the unicast approach can provide T-VoD services. However, such approach incurs very high costs of the system and lack of scalability. In fact, if the server does not have sufficient resources to support the request, the client may either wait in the system or is blocked from the system. The average waiting time or blocking probability can be modeled as an M/G/c queue. To conserve the server resources and improve the system performance, the exploitation of the broadcast capabilities of the network has been investigated.

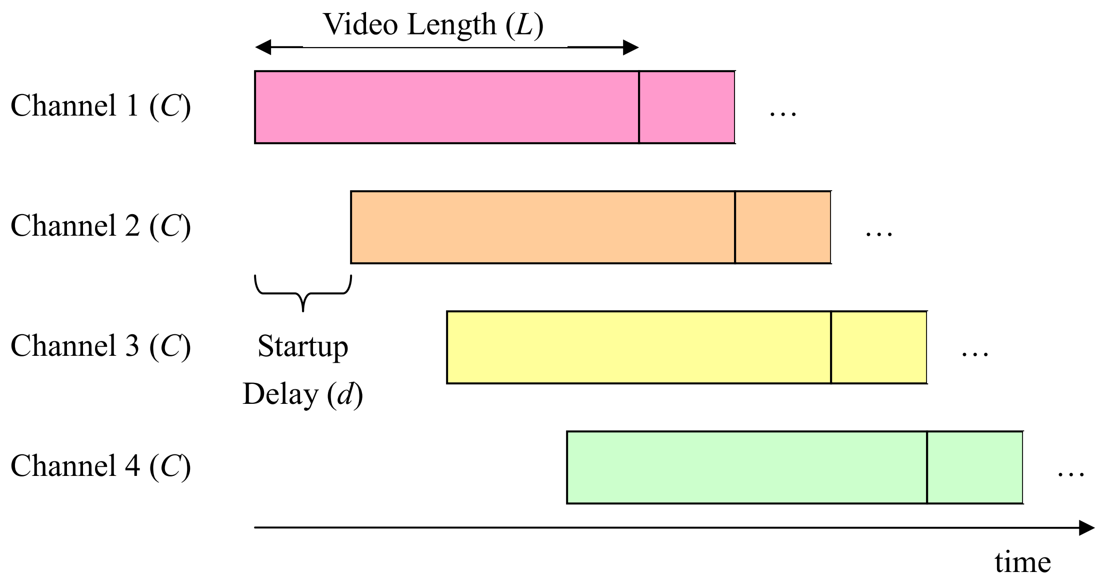


Figure 2.2. Staggered Broadcasting Protocol

A number of periodic broadcast protocols have been proposed in recent years to exploit the broadcast facility of modern communication networks to provide an efficient means of one-to-many data transmission [DAN94, VISWA96, HUA97, JUHN97]. In this scheme, a video is first partitioned into a number of segments, each of which is transmitted on the designated video channel periodically. Clients do not need to make any requests explicitly. They only listen and fetch the desired segments from the appropriate channels according to the download policy. Therefore, the server bandwidth requirement is independent of the number of clients that the system can support. The conventional approach to implement the broadcast scheme is to open a new video channel at a fixed interval. Staggered Broadcasting (SB) [DAN94] is the earliest implementation based on this mechanism. Figure 2.2 illustrates the idea of SB. Assume that the video length is L seconds with the data rate of C Mbit/s. If the phase delay between two video channels is of

d seconds, the system requires to allocate $\frac{L}{d}$ number of video channels. d is also the maximum access time denoted startup delay that the longest time any client needs to wait. Therefore, for example, 4 video channels are allocated to each video if the length of the video is 120 minutes and the startup delay of the system is 30 minutes. In SB, the requirement of clients is very simple. Each client only needs to have a bandwidth capacity at the playback rate of the video and it does not need any storage requirement to cache the data. However, each client has to suffer from very long startup delay of $\frac{d}{2}$ seconds on average. To reduce the startup latency, the system should linearly increase the number of video channels.

To improve the startup latency, Pyramid Broadcasting (PB) was proposed in [VISWA96] with the cost of a large client buffer. In this scheme, the server bandwidth (B) is divided into k video channels of equal bandwidth. Suppose that there are M videos provided by the system. The video is partitioned into k segments of geometrically increasing size and the segments size of the i^{th} video segment (S_i^{PB}) can be determined by eqn.(2.1).

$$S_i^{PB} = \frac{\alpha^{i-1}(\alpha - 1)}{\alpha^k - 1} \cdot L \qquad (2.1)$$

where $\alpha = \frac{B}{CMk}$ [VISWA96]. Then, the server broadcasts one of these segments of the video in a separate video channel periodically. In addition, different videos are mingles together in each logical channel. Client can fetch segment at its earliest occurrence from at most two consecutive channels simultaneously. The startup delay of the scheme (D^{PB}) is given by eqn.(2.2).

$$D^{PB} = \frac{CLMk(\alpha - 1)}{B(\alpha^k - 1)} \tag{2.2}$$

This approach requires each segment to be transmitted in a very high rate in order to provide on time delivery of the videos such that this approach is very strict with the requirement of client machine. Its variant, Permutation-Based Pyramid Broadcasting (PBPB) [AGGAR96b], was proposed to tackle the problem of high client machine requirement. The idea behind this variation is to divide each logical channel into *n* sub-channels and in turn broadcast each replica of segment with a uniform phase delay. Therefore, the transmission rate of each segment is reduced.

To further enhance the performance of PBPB so that the client requirement can be reduced, Skyscraper Broadcasting (SKB) was developed in [HUA97]. In this scheme, as shown in Figure 2.3, the server bandwidth is divided into a number of logical channels of bandwidth equal to the playback rate of the video (i.e. *C* Mbits/s). A video is partitioned into a number of segments and the size of each segment is determined by the segment size progression, such as {1,2,2,5,5,12,12,25,25,…}, but the size of the last segment is limited by the client buffer size (*Buf*). In general, the segments size of the i^{th} video segment (S_i^{SKB}) can be computed by eqn.(2.3).

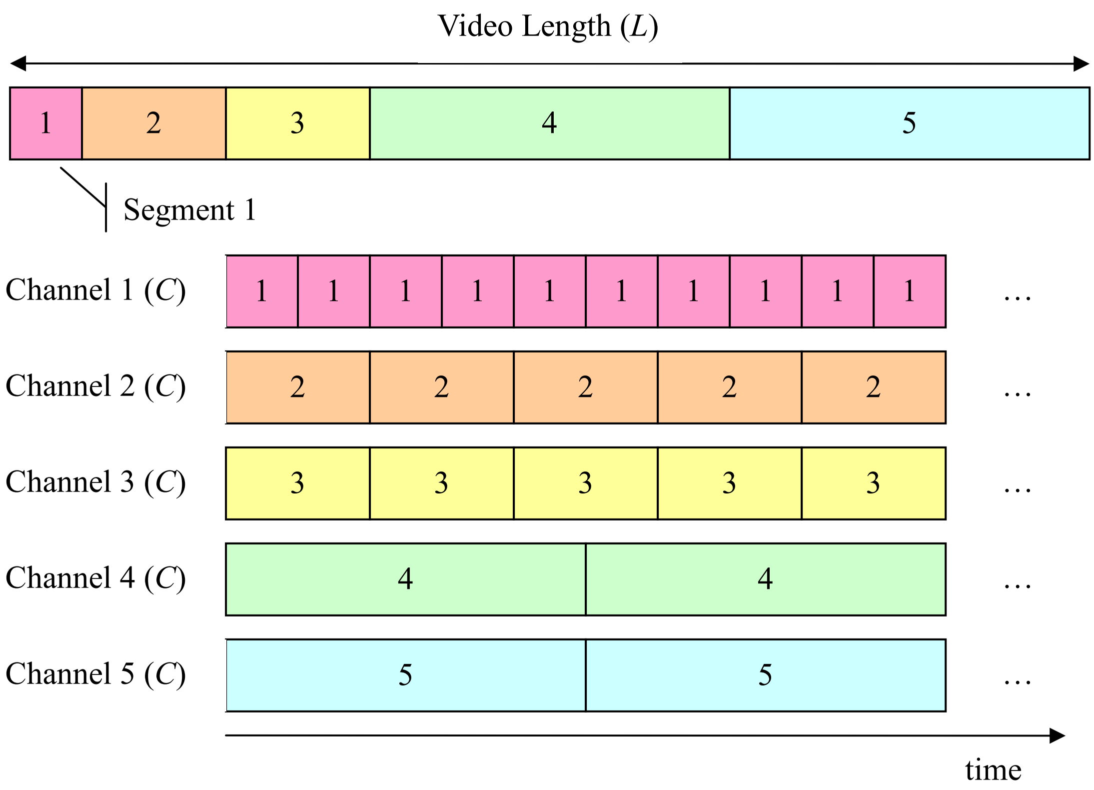


Figure 2.3. Skyscraper Broadcasting Protocol

$$S_i^{SKB} = \begin{cases} 1 & ,i=1 \\ 2 & ,i=2,3 \\ (2+2\left\lfloor \frac{i}{2} \right\rfloor - i)S_{i-1}^{SKB} + (1+2\left\lfloor \frac{i}{2} \right\rfloor - i)(1+2\left\lfloor \frac{i-4\lfloor 1/4 \rfloor}{2} \right\rfloor), otherwise \end{cases} \quad (2.3)$$

Clients are required to download from at most two channels (segments) at any time. The startup delay of this scheme (D^{SKB}) is given by eqn.(2.4).

$$D^{SKB} = \frac{L}{\sum_{j=1}^{n} \min(S_j^{SKB}, Buf)} \quad (2.4)$$

Originally, SKB was only designed for supporting an N-VoD service to clients. Some later works [EAGER01, POON03] had successfully modified SKB to provide a true-VoD service. In [EAGER01], a partitioned dynamic skyscraper (PDS) was proposed. Instead of

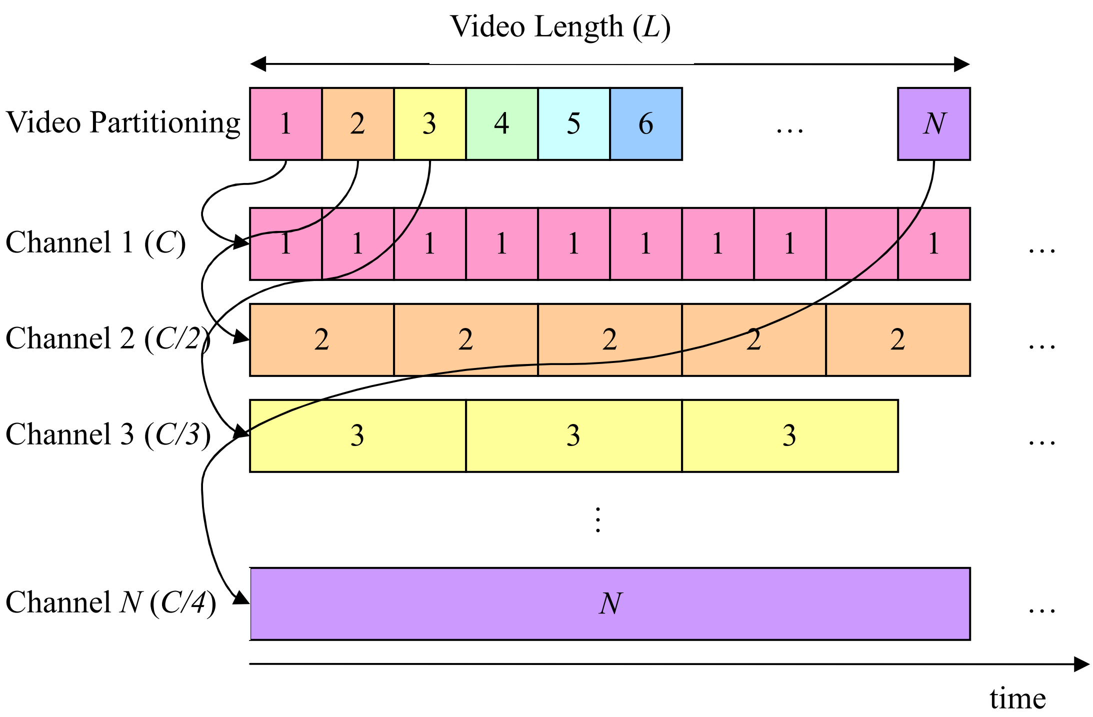


Figure 2.4. Harmonic Broadcasting Protocol

retrieving the first segment from the broadcast channel, a unicast connection is established between each newly admitted client and the central server such that the first segment can be served directly without a sensitive delay experienced. In [POON03], by using the concept of patching and reorganizing the first segment into a number of small sub-segments, a new client can receive the missing sub-segment from the patching channels to guarantee a continue playback.

Harmonic Broadcasting (HB) [JUHN97] provided an alternative solution to support video broadcasting. In HB, as shown in Figure 2.4, a video is divided into equal size of segments which are broadcasted in logical channels of decreasing bandwidth. The bandwidth allocation of channel i (X_i) in this approach follows the harmonic series which can be computed by $X_i = \frac{C}{i}$. Each channel is only required to handle one segment of video that segment i (S_i) is periodically being broadcasted on the rate of X_i. The playback duration of a segment is defined as a slot. According to the designated bandwidth of each logical channel, each segment may occupy several slots for transmission. Clients start fetching data from each channel right after it can start downloading the first segment. When the client is ready to playback S_i, it has already received $i - 1$ slots of data from that segment and the last slot of that segment can be received during playback time of the segment. This approach requires much less server bandwidth than the pyramid-based broadcasting protocols. The startup delay of the video is determined by the size of the first segment. The main flaw in HB is that it cannot always deliver all the segments on time. Its variants, such as Cautions-Harmonic [PARIS98a], Quasi-Harmonic and Poly-Harmonic [PARIS98b], were proposed to address this problem.

From the above policy, we can know that the unicast scheme is unable to cope with the overload situation while the system resources will be under-utilized with the use of broadcast scheme during low loads. In order to address some of the drawbacks of the previous two policies, a multicast scheme was thus proposed. In the multicast approach, customers arriving within a short time interval for the request of same video are batched together. This approach is also referred to as *batching*. Customers arrived to the system are first placed on a queue and the system will select a number of customers from the queue(s) to serve based on different service policies when the system resources become available.

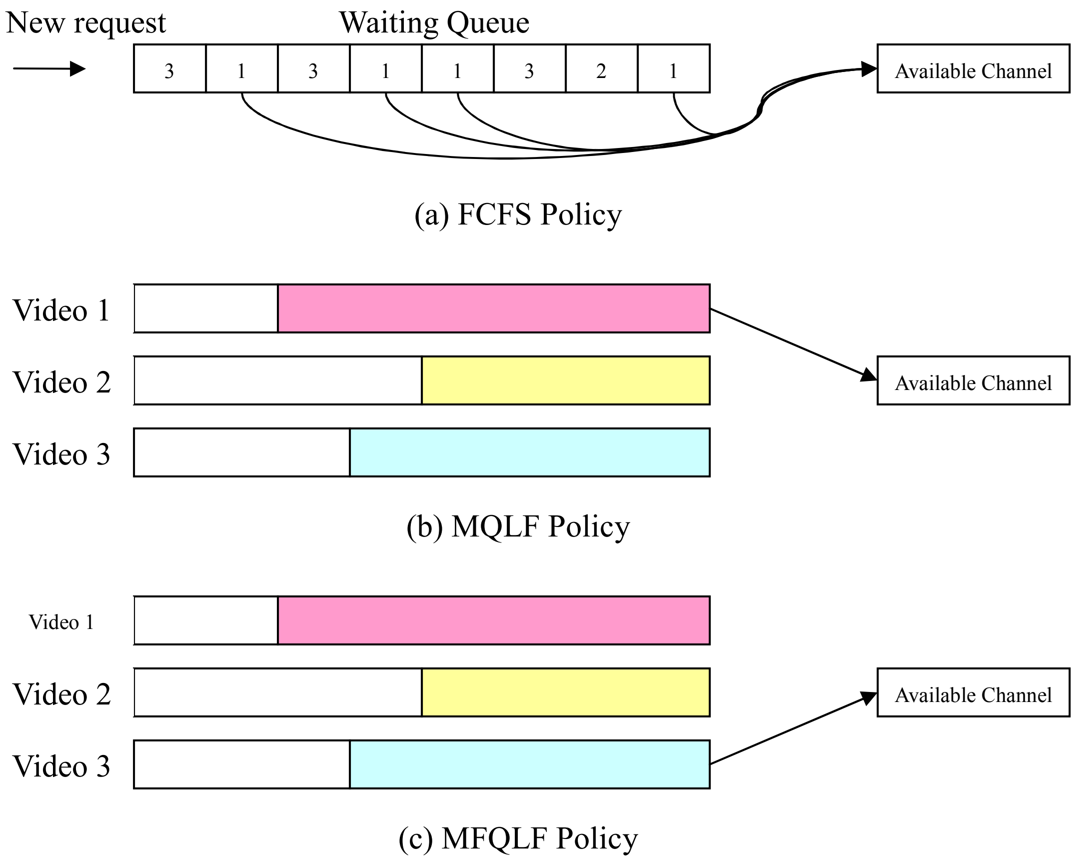


Figure 2.5. Various Batching Policy

Figure 2.5 illustrates three static multicast policies. In the first-come-first serve (FCFS) policy [DAN96], all customers are first queued on the same queue. Then, when the system resource is got ready, the oldest request with the longest waiting time will be batched and served next. In Figure 2.5a, since the oldest request is the request of video 1, the system will fetch all customers requesting the video 1 from the queue under this policy. This policy treats each customer equally regardless of the popularity of the video and thus provides a good fairness of services. However, as the serving batch is based on the arriving time of customers, this policy may introduce a drawback of low system throughput in case the next selected batch only has fewer requests compared to the other batches. Therefore, the maximum-queue-length-first (MQLF) policy [DAN96] was proposed to address this problem. Under this policy, each video has its own waiting queue for customers and the system serves the queue with the longest queue length (as shown in Figure 2.5b). This approach can maximize the utilization of system resource. Nevertheless, requesting of less popular video may not be served by the system that induces a problem on fairness as the next batch is selected only according to the length of the queue. In order to provide reasonable fairness and also high system throughput, maximum-factored-queued-length-first (MFQLF) [AGGAR96a] revised the batching mechanism of MFQL. MFQLF uses the same queuing strategy as MFQL but there is a slight difference on selection of queue to be served next. As shown in Figure 2.5c, when the system resource is free, each queue is first weighted by a factor of $p_i^{-\frac{1}{2}}$, where p_i is the popularity of video *i*, the system then selects the queue with the highest weighted value to be served. This factor avoids the server from always fetching the longest queue. In this batching-based multicast approach, clients have to wait in a queue and thus they only obtain N-VoD services.

To provide T-VoD services for customers in a multicast environment, the system should allow late-arriving requests for the same video to join the existing ongoing multicast channel. Transmission rate adjustment and extra channel allocation are two general ideas to achieve this purpose. Golubchik *et al*. [GOLUB96] proposed a *piggybacking* approach which merges different customers together by adjusting the transmission rate of the video stream. When a new customer arrives, the server first increases the transmission rate of the video stream to a new customer and slows down the transmission rate of the current video stream to the previous customers for the same video. When they reach the same playback point in the video, the server merges the two video streams and serves them with a single multicast channel. However, this adjustment should be controlled within 5% to preserve the display quality of the video that limits the number of channels it can merge to save resources. Furthermore, this method has to store a replica of videos with different playout rates in the server or adjust the playout rate in real time which increases the complexity of the system. Patching [HUA98] is another scheme providing T-VoD services over multicast, which is achieved by allocating an extra channel to customer from the server so that the customer can enjoy the service without waiting. When the late-arriving customer requests a video, it first joins and caches the contents from the newest ongoing multicast channel for the same video. Since it is only retrieving the later portion of video from the multicast channel, the initial portion of video is missed. Therefore, the server allocates an extra channel called patching stream that delivers the missing portion of the video to customer. So, the customer is downloading the video from two streams (patching stream and multicast stream) simultaneously in this scheme. When finishing playing back the video content in the patching stream, it switches to playback the video content in its local buffer. The buffer size equipped depends on the time interval between consecutive multicast channels for the same video.

2.4 Video Transmission in Peer-to-Peer Network

The P2P architecture for video transmission is emerging in recent years which can eliminate the need for costly dedicated video servers as in the traditional client-sever approach. The beauty of this architecture is that the system is inherently scalable, i.e. each admitted user contributes its processing power, data storage and bandwidth to increase the capacity of the system. Research [SRIPA04] has shown that the P2P approach is a feasible way to support large-scale VoD services, but this approach still faces some design challenges. In a P2P system, peers do not always stay in the system that they can leave and enter the system in arbitrary time. Such dynamic nature of P2P framework requires the system to have a quick searching and a graceful recovery procedure to localize supplying peers and handle service failures. Unlike a powerful dedicated video server, peers can only contribute a limited bandwidth and storage capacities to the system. In addition, the available bandwidth of the supplying peers might fluctuate unexpectedly. Therefore, the system should provide an efficient way for supplying peers assignment and scheduling to keep the streaming quality unaffected. In order to keep the service without disruption, peers are required to exchange information periodically. To prevent such information overloading the network, overhead for exchanging information among peers has to be kept small. To address these issues, a number of peer locating mechanisms and transmission strategies have been proposed.

2.4.1 Peer Locating Mechanisms

To achieve a good quality of service, each requesting peer has to find sufficient supplying peers with sufficient bandwidth and low latency. Thus, locating supplying peer is an essential process in a P2P VoD application.

Keeping a centralized directory of all peers in a directory server is the simplest way and commonly used approach for locating peers. In this approach, all peer information such as its available bandwidth and network address are stored in a special list denoted peer directory in the directory server. When a new requesting peer issues a request, it is first redirected to the directory server. Then, the directory server selects the most suitable supplying peers from the peer directory for the requesting peer. Meanwhile, the directory server also updates its peer directory with the requesting peer's information. If a peer wants to leave the system, it should generate a LEAVE message to the directory server so that its entry can be eliminated from the peer directory. The main advantage of this approach is simple deployment and easy implementation. However, since all peers information should be maintained, the directory server requires to keep track $O(N)$ states given N peers in the system. Therefore, it might overload the system if N is large.

To disperse the workload of locating peer over the system, peers construct a hierarchical overlay structure that allows a new requesting peer adaptively locates the supplying peer over it. In this approach, a new requesting peer first communicates with the overlay's rendezvous point which is the data source in the hierarchy. In response to the request, the rendezvous point returns a list of connected peers down one level in the hierarchy. Then, the requesting peer probes each peer in the list and determines the most suitable peer *P*. The requesting peer contacts *P* which replies with another list of its connected peers down one level in the hierarchy. The requesting peer finds out the best peer from the new list. This process repeats again until the requesting peer reaches a position in the overlay where it can obtain the best QoS.

Distributed Hash Table (DHT) is another distributed approach for locating supplying peers in P2P application. In DHT, each file is associated with a key *k* which is generated by a SHA1 hash function. The peer holding this file sends a message PUT(*k, data*) to any other peers participating in the DHT. This message is forwarded from peer to peer through the overlay network until it reaches the single peer responsible for key *k*. There is a routing table maintained at each peer in the DHT. When a requesting peer wants to retrieve the contents of the file, it first hashes the requested file to produce key *k* and queries any peer in the DHT to find the data associated with key *k* with a message GET(*k*). This message will be routed through the overlay again to the node responsible for key *k*, which will reply with the requested data. Research works [ROWST01, STOIC03] have proven that the query message is routed through $O(\log N)$ peers only for each request and each peer only

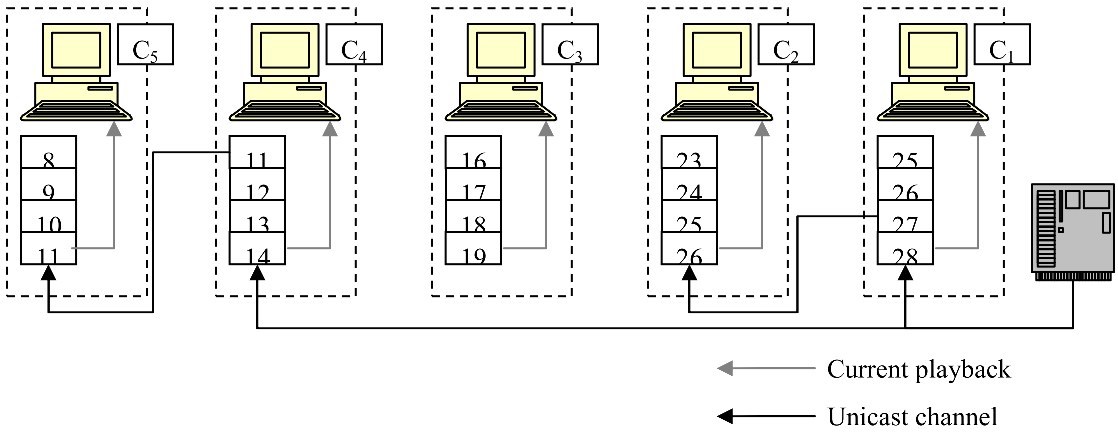


Figure 2.6. Operation of Chaining Scheme

requires to hold *O(log N)* states in its routing table.

2.4.2 Transmission Strategies

Chaining [SHEU97] forms the foundation of various P2P approaches for video delivery. We first briefly introduce the basic concept of the chaining-based scheme as follows. When the central server receives a new video request from a client, it first determines whether it is the first request for the new video session. If so, the server will open a unicast connection and deliver the requested video to the client immediately. Otherwise, the server redirects this request to the latest arrived client in the current video session, who will be responsible to serve this client. Thus, a late-coming client will first setup a connection to the latest arrived client and then retrieve the video content from that client. This interconnection mechanism between clients to pipeline video data forms a video chain. To accomplish this mechanism, each client should cache a sliding window of the video into local buffer during playback. If the video content in the latest client's buffer in the current video chain does not meet the playback time of next client, a new video session (i.e. a new video chain) is initialized. In chaining-based schemes, video is partitioned into fixed-sized N segments which are numbered from 1 to N and transmitted in sequence along the video chain. Each client has a local buffer with the size of B segments (normally, $B \leqq N$). To illustrate the flow of data, it is first assumed that transmission of one segment spends one time unit. Figure 2.8 illustrates how these various schemes operate. There are five clients, C_1, C_2, C_3, C_4 and C_5, each of which has a buffer of four segments (i.e. $B=4$). The arrival time between two successive arrived clients, C_j and C_k, denotes playback gap ($T_{j,k}$). Let the arrival time of C_i be t_i, $T_{j,k}$ can be computed by $T_{j,k} = t_k - t_j$. It is shown from the figure that $T_{1,2} = 2$, $T_{3,4}= 7$, $T_{3,4}= 7$ and $_{4,5}= 3$. In basic chaining, if C_j is the tail of the current video chain, C_k can join this chain only when $T_{j,k} \leqq B$. Otherwise, a new video chain is initialized and C_k becomes the head of the new chain. Figure 2.6 shows an example operation of a basic chaining scheme at a snapshot, $t_1 + 29$. At first, C_1 issues a request for video i to the central server at t_1. The server examines that it is the first request for this video and thus allocates a new video channel to serve this client. C_1 is receiving the video data from this channel, meanwhile, storing in its local buffer. At t_2, C_2 arrives and generates a request for the same video. Since the playback gap of C_1 and C_2 is 2 that mean C_1 already has the first two segments of video i in its buffer, they can be chained together. Hence, the server redirects this request to C_1 which then setups a connection to serve C_2. Just a moment has elapsed, C_3 arrives. Obviously, it cannot join the existing video chain because the playback gap of C_2 and C_3 is larger than 4. Therefore, the central server generates a new stream for C_3. Due to the same reason, another new stream should be established to serve C_{k4}. Finally, C_5 can chain to C_4 because their playback gap fulfill the predefined condition. In this example, five video channels are allocated in the system (three of them generated from central server, two of them setup from clients). Under this scheme, except for the current playback segment, other segments in the buffer have already been played out and used for chaining the following client. This type of buffering technique is so called backward buffer. DirectStream [GUO03] improves “Chaining” by taking peer’s outbound bandwidth into account. Each peer is permitted to serve more than one peer to fully utilize its outbound bandwidth and thus a higher degree application layer multicast tree is constructed. It also considers the smooth playback by delaying the playback time to prevent from buffer starvation caused by an early departure of the peer’s parent.

While chaining-based transmission scheme uses different buffering techniques to reduce the server bandwidth requirement, P2Cast [NICOLO03] uses an alternative approach, patching, to accomplish the same purpose. In P2Cast, clients arriving close within a predefined threshold T are grouped to form a *session*. Clients in the same session construct an application layer multicast tree denoted base tree and the whole copy of video data transmits along this tree from the central server. The video data flows on base tree is

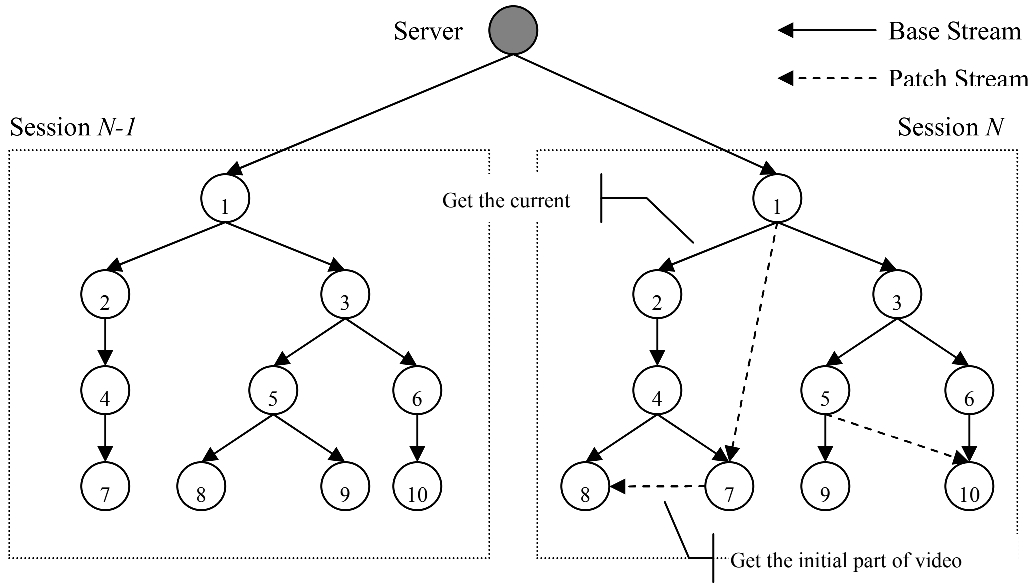
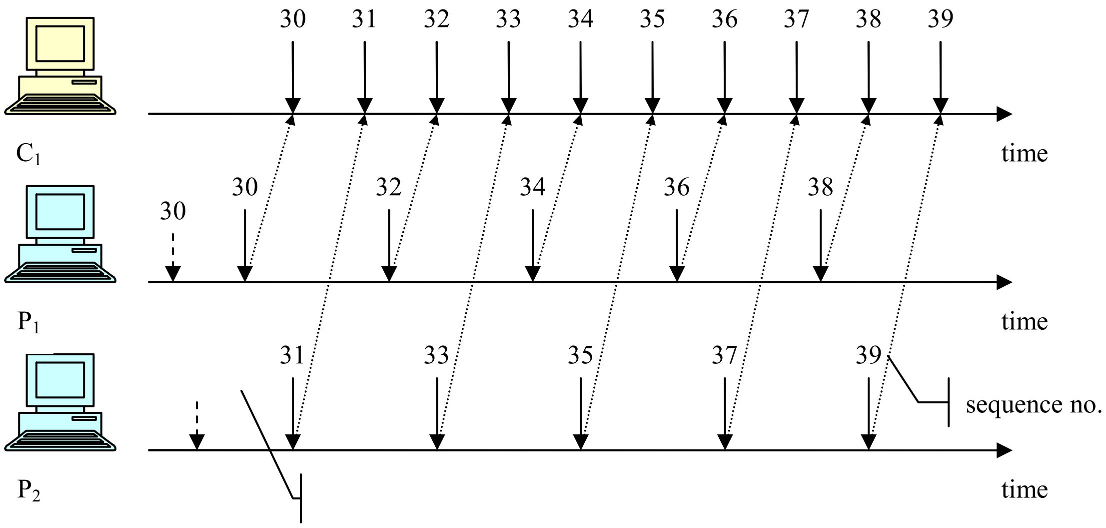


Figure 2.7. The Ideal of the P2PCast

called base stream. Each client has to join the base tree in a specific session and retrieve the video data from the base stream immediately upon the request has been accepted. Obviously, in the same session, except for the first arrival client, other late-coming clients also miss the video content from when this session starts to when they arrives. Thus, they setup another channel denoted patching channel to other peer (called patch server) to obtain the initial part of the video simultaneously. Therefore, under this scheme, the base tree can grow continuously within *T* without breakdown. Figure 2.7 is an example to show the idea of this scheme. The number inside the circle identifies the arrival sequence of the peer (i.e. late-coming clients has higher number). All clients in session *N-1* have completed the patching process and thus they only receive the rest part of the video data from the base stream. In session *N*, three clients are still receiving the initial part of the video from patch servers and also downloading the current video data from the base stream simultaneously.

D(1)	D(2)	D(3)	D(4)	D(5)	S(1)	S(2)	S(3)	S(4)	S(5)	Sync

(a) Format of the Control Packet



(b) Transmission Scenario

Figure 2.8. The Idea of PPA

Apparently, the central server bandwidth requirement is governed by *T*. Large *T* results in lower server resources but it increases the duration of patching of clients that both clients' buffer and network bandwidth requirement are also increased. Therefore, *T* is a critical factor affecting the performance of the whole system.

Nguyen *et al.* [NGUYE04] proposed a distributed video streaming framework using a receiver-driven protocol for simultaneous video streaming from multiple senders to a single receiver. This protocol consists of a rate allocation algorithm (RAA) and a packet partition algorithm (PPA). Once the desired video object has been found from *M* peers in the P2P network, the receiving peer first uses RAA to determine how to split the total rate of the video between *M* sending peers in order to minimize the probability of irrecoverable loss for a given amount of FEC as well as ensure the scheduled transmsission rate that does not excess the available bandwidth of the sender. During video streaming, each sending peer should estimate and send its round-trip time (RTT) to the receiving peer periodically. The receiving peer uses this information and the estimated loss rate of each sending peer as parameters for RAA. The outputs of RAA embedded in control packet are then transmitted to the corresponding sending peers. Then, based on this feedback information in the control packet, each sending peer uses PPA to determine which packets should be sent in order to prevent packet duplication and minimize the startup delay. We describe the details of PPA as follows. Figure 2.8(a) depicts the format of control packet, which includes three types of information, for 5 sending peers. *D(i)* specifies the estimated delay from sending peer i (P_i) to the receiving peer and *S(i)* shows the allowable sending rate of P_i. The *Sync.* field holds the starting sequence number that is used for determining the next packet to send by each sending peer. The time at which the control packet with starting sequence number *k'* is sent by the receiving peer is denoted as $T_{k'}$. Therefore, each sending peer has a global knowledge about all participants throught the control packet. If *n(j, k, k')* denotes the number of packets already sent by P_j since packet *k'* and up to *k*, the estimated arrival time of the k^{th} packet sent by P_j can be computed by $n(j,k,k')S(j)^{-1}+2D(j)$. Since the playback time of the k^{th} packet wih respect to $T_{k'}$ is $P_{k'}(k)$, the estimated time difference between the arrival and playback time of the k^{th} packet sent by P_j is expressed by

$A_{k'}(j,k) = P_{k'}(k) - \left(n(j,k,k')S(j)^{-1} + 2D(j)\right)$. Based on such information, each sending peer can determine the next packet to send. P_j first computes $A_{k'}(i,k)$ of P_i for packet k, where $i=1,2,...M$. If it finds $A_{k'}(i,k)$ at a maximum when $i=j$, it sends packet k. Otherwise, it simply increases k by one and repeats the procedure again. Therefore, P_j is allowed to send packet k only if P_j gets the highest value on $A_{k'}(i,k)$ among other sending peers such that the probability of packet k being late can be minimized. Figure 2.8(b) illustrates an example operation of PPA. In this example, there are two sending peers, P_1 and P_2, with the same sending rate. The *Sync* sequence number in control packet is 30. Since *D(1)* is lower than *D(2)* (control packet arrived in P_1 first) and thus $A_{30}(1,30)$ is higher than $A_{30}(2,30)$, packet 30 is sent by P_1. For packet 31, because P_1 has transmsitted one packet over the network, obvisouly, $A_{31}(2,31)$ is higher than $A_{31}(1,31)$ such that packet 31 is sent by P_2. Therefore, following packets are interlacing transmitted by P_1 and P_2.

Another *M-to-1* transmission protocol is P2P Adaptive Layered Streaming (PALS) [AGARW06] which involves the consideration to layered encoded video stream. In this scheme, the receiver monitors the exponentially weighted moving average (EWMA) bandwidth from each sender and determines EWMA of aggregate bandwidth from all senders (T_{ewma}). Time is divided into a number of fixed intervals, each of which represents a range of timestamps for packets. The receiver maintains a window of time (Δ) called active buffering window and deploys a sliding window (SW) mechanism to govern the

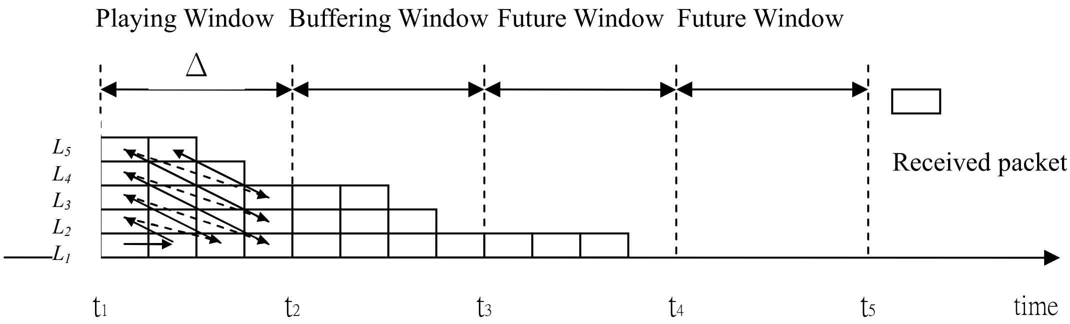


Figure 2.9. Sliding Window and Packet Ordering in PALS

movement of this window along the intervals such that packet with timestamp identical to the interval inside the window must be requested and recevied. With the assumption of the value of T_{ewma} kept unchange for one window, the number of incoming packets can be determine by $\frac{T_{ewma} \cdot \Delta}{S}$, where S is the size of packet. Based on the relative positive, the window can be divided into three groups, namely Playing Window, Buffering Window and Future Window. Figure 2.9 shows an illustration of these windows. Buffering Window [t_2, t_3] is the current position of the active buffering window. Packets belongs to this window must be requested at this point. Playing Window [t_1, t_2] is the previous window where the player fetches packets for playback. In this window, some packets have not been delivered but still have sufficient time for delivery of these packets. Furture Window [t_2, t_3] is the next window of the current window. If the current estimated bandwidth is large enough or most of the packets within the current window have been already arrived, packets belongs to this window can be requested early. After determining the aggregate bandwidth and number of incoming packet, the Quality Adaptation (QA) mechanism is invoked. The QA mechanism is launched once per window to detemine the required packets for each active layer based on variations of T_{ewma}. Denote n be the number of active layers and C be the data rate of each layer. When the aggregate bandwidth is higher than the stream bandwidth ($nC \le T_{ewma}$), QA assigns excess bandwidth to request future packets filled in the Future Window. Once the buffered data reaches to an appropriate level, QA increases the stream bandwidth by adding a new layer. In contrast, when the aggregate bandwidth is lower than the stream bandwidth ($nC > T_{ewma}$), it first drains the buffered data to compensate the bandwidth deficit. If the buffered data is inadequate to absorb bandwidth deficit, it drops the top layer. In order to react the long-term and short-term mismatch between available bandwidth and stream bandwidth, QA provides coarse-grained adaption and fine-grained adaption to adjust the number of layers and control evolution of buffer state respectively. After selecting the required packets, the receiver assigns these packets to be delivered by the appropriate sender. It is accomplished by Packet Assignment (PA) mechanism. PA divides selected packets into disjoint subset and sends a separate request to each sender. The number of packets to be delivered by each sender is based on its EWMA bandwidth. Unlike the previous approach that the sender is required to determine which packet to be sent next, each sender simply transmits requested packets in PALS.

CoolStreaming [ZHANG05] uses a data-centric design to provide live media streaming. The idea of this scheme is simple. Each peer periodically exchanges data

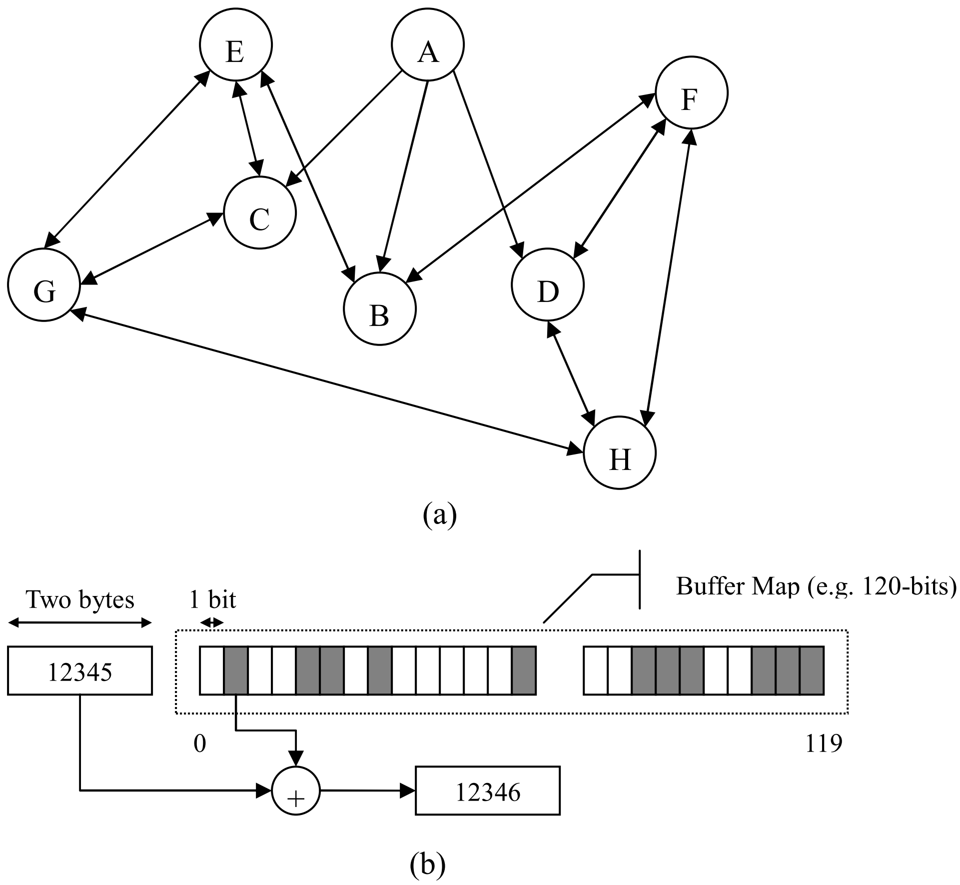


Figure 2.10. (a) Illustration of the Partnership in CoolStreaming (b) Idea of BM

availability information with a set of partners and retrieves unavailable data from one or more partners, or supplies available data to partners. The role of the peer and its partners are equal such that this partnership relation allows video data transmit from late-coming peer to early-coming ones or in reverse direction. In other words, it is the availability of data that guilds the flow directions. Each peer running CoolStreaming has a unique identifier and maintains a membership cache denoted *mCache*. This cache contains a partial list of the identifiers for the active peers in the CoolStreaming network. To keep the updated membership information among peers, each peer should periodically distribute membership message following Scalable Gossip Membership protocol [GANES03].

Figure 2.10(a) shows an example of the partnership in CoolStreaming. Similar to the previous schemes, a video stream is divided into a number of fixed-length segments in CoolStreaming. Each peer deploys a circular bit map called Buffer Map (BM) to represent the availability of the segments in its local buffer. An example of the size of BM is 120 bits. Each bit in BM indicates whether the corresponding segment is available or not. The sequence number of the first segment in BM is recorded by another two bytes. Figure 2.10(b) illustrates the idea of BM. Each rectangle represents the availability of each segment (white and dark rectangle indicates available and unavailable segments of between 12345 and 12464 in peer's local buffer respectively). Then, each peer continuously exchanges its BM with its partners and then schedules which segment is to be fetched from which partner accordingly. Therefore, under this scheme, each peer tries to retrieve the unavailable segments from other partners which indciate these segments are available from their BMs. CoolStreaming uses a heuristic algorithm to determine the best suppliers for a specific segment from a number of potential suppliers (i.e. the corresponding bit for this specific segment is set in BM) based on their bandwidth and the available time for transmission. For example, refer to Figure 2.10(a) again, peer *G* receives BMs from peer *C,* peer *E* and peer *H*. Peer *G* first determines the deadline of segment *i* and counts the number of peers containing segment *i* based on the received BMs. Then, it computes the transmission time of segment *i* from each peer. Assume peer *C h*as the highest bandwidth and the shortest transmission time for segment *i*, peer *G* sends a request message to this peer to fetch this segment. This segment is delivered through a real-time transport protocol which adopts the TCP-friendly rate control protocol [PADHY98, FLOYD99, FLOYD00, BANSA01]. With the same principle, other peers retrieve their desired segment from their partners based on the contents of their BMs.

2.5 Hybrid Transmission Approach for Video-on-Demand Systems

In Section 2.3.3 and Section 2.4, we have reviewed a number of transmission strategies for VoD systems. However, these approaches have their own problems for video delivery. Proxy caching and CDN are expensive to deploy and maintain. P2P requires a sufficient number of supplying peers to jumpstart the distribution process [XU06] because each peer may only able to contribute limit resources to the system. For example, the outbound bandwidth of the peer may be lower than the playback rate of the video. In addition, the dynamic nature of P2P application is another flaw that a peer can leave the system at any time without notice. Broadcasting protocols such as HB are impractical to support insensitive (less-than-minutes) startup delay services since the central server needs to manage a large number of concurrent channels for a single video. Therefore, to compensate for their disadvantages, a number of hybrid approaches have been proposed recently.

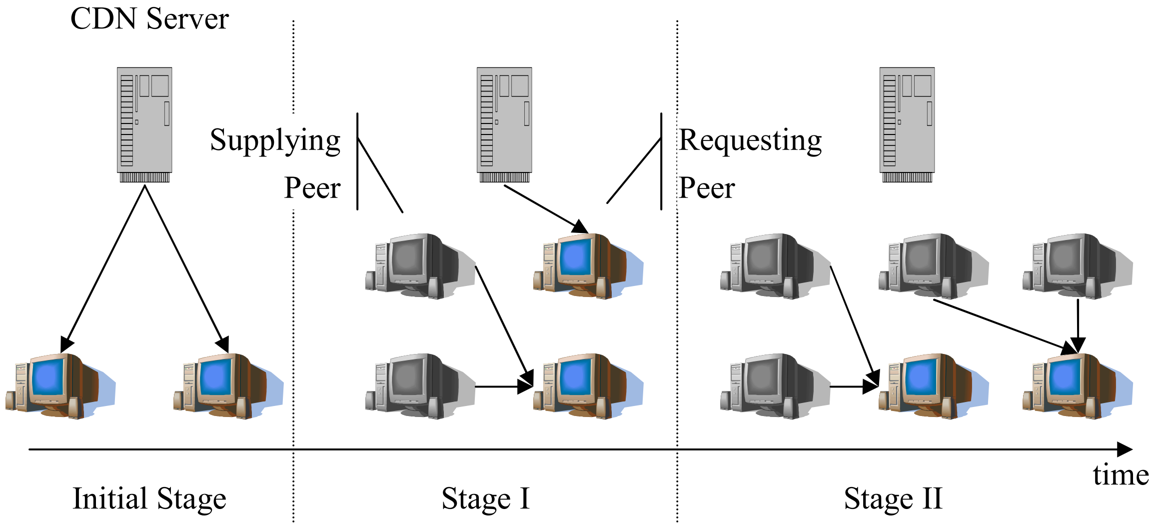


Figure 2.11. Operation of CDN-P2P Architecture

Xu *et al.* [XU06] proposed a CDN-P2P architecture that integrates both P2P and CDN to provide a cost-effective solution for video services. In this architecture, these two technologies complement each other. As shown in Figure 2.11, the distribution process of video data in this framework involves three stages. When the system is launched, the system first enters an *initial stage*. In this stage, the requesting peers are served by the CDN server directly. During obtaining the video data from the CDN server, a number of supplying peers are also created. The CDN server can then divide the workload between itself and the supplying peers such that any newly admitted peers are served by the CDN server and the supplying peers simultaneously. This is the second stage when the CDN and P2P delivery co-exist to support the service. Once the aggregated bandwidth of the supplying peers is sufficient to support the subsequent peers, the system goes into the third stage that the reserved CDN server resources are released and let the supplying peers

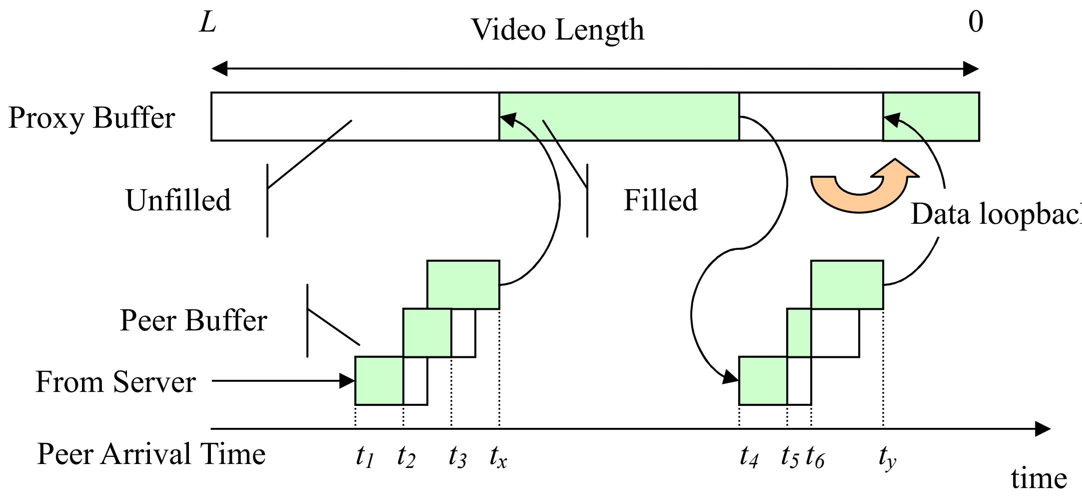


Figure 2.12. Operation of Loopback

maintain the services. But, the CDN should take over the service again if the bandwidth contribution of the P2P network is lower than the demands. Therefore, customers can guarantee to obtain the service without collapsing by the dynamic nature of P2P framework and the CDN server resources can be used more effectively as well.

Kusmierek *et al.* [KUSMI06] exploits the idea of chaining and proxy caching techniques to support video services in their proposed system called *Loopback*. In Loopback, customers arriving close to each other in time form a forwarding ring with the first customer obtaining data from a proxy and the last customer returning data to the proxy. Therefore, a loop is formed. Whenever a customer buffer fills up before the next customer arrives, a new loop is created for the next customer. Figure 2.12 illustrates the operation of Loopback. The arrow in the figure indicates the direction of the video data flows between the proxy and each loop. Assume that the buffer size of the proxy is the same as the length of video (L seconds). The first peer arrived at t_1 which has played the leading part of the

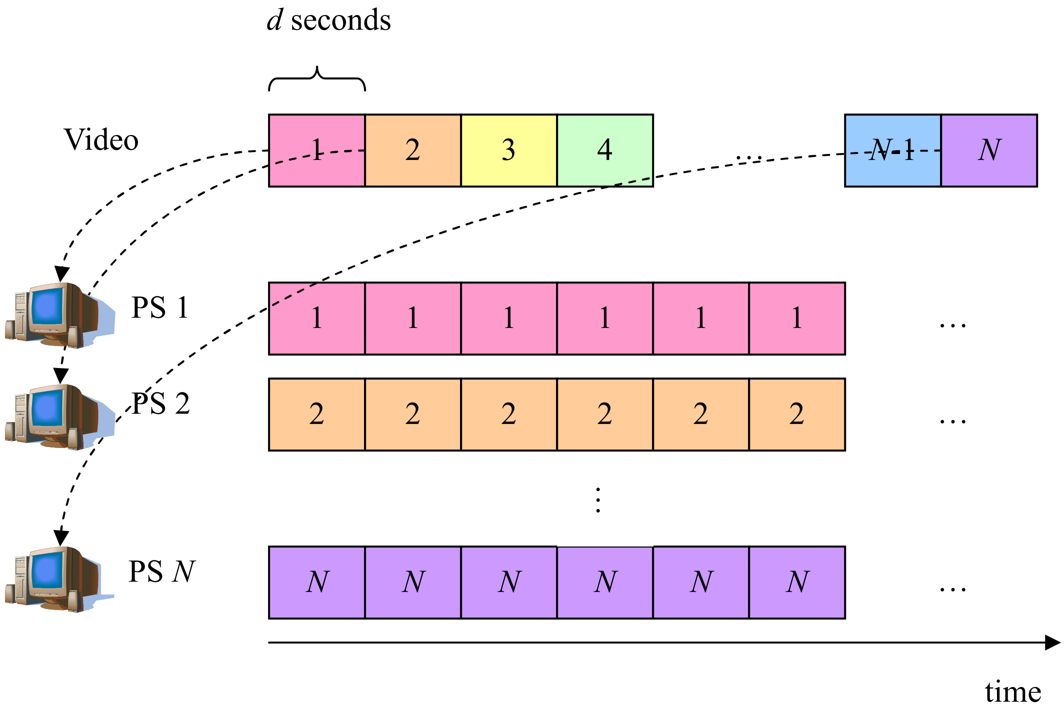


Figure 2.13. DPCS Transmission Scheme

video and is downloading the current part of the video from the central server. Later, the second and third peer arrived which were close to the first peer and thus they chained together. After t_x, there is no more peer that can be chained and the third peer should pass the video data to the proxy. When the forth peer arrives, it obtains the video data from the proxy server directly and then forwards it to the subsequent peers (i.e. the fifth and sixth peer). The sixth peer then starts to return the video data to the proxy at t_y. In this mechanism, the proxy server only requires to cache the length of the video in the time gap between two loops. Thus, Loopback can reduce storage and bandwidth requirement of the proxy as well as the workload of the central server.

To *et al.* [TO05] proposed a hybrid scheme called Distributed Probabilistic Clock Synchronization (DPCS) in which the existing SB is modified to adapt the P2P transmission. Unlike the original SB that a new video session is created every d seconds (i.e. the startup delay of the system) from the central server and a whole video is transmitted on the video channel periodically. As shown in Figure 2.13, in DPCS, a video with a length of L seconds is first divided into N equal segments (i.e. $N = L / d$), each of which is assigned to one end-point machine denoted peer server (PS). Each PS then transmits its assigned segment to its own video channel periodically. Each PS is required to contribute a buffer size of d seconds. While the bandwidth required for the system is still kept as the original SB, the workload of the system is dispersed among PSs.

Yang *et al.* [YANG05] developed Dynamic Distributed Collaborative Merging (DDCM) which comprised of two stream managers, Patch Stream Manager (PSM) as well as Complete Stream Manager (CSM), for P2P streaming. PSM uses peers' unused buffer to form a collaborative buffer to store the suitable video data. When a new video session starts up, the central server first allocates a new multicast channel for the first peer of a new peer group. Other subsequent peers for the same group then join this multicast channel and establish a unicast channel to obtain the leading part of the video from either the central server or other peers in the system if the video contents are available in the collaborative buffer. With CMS, a number of peers are selected, each of which holds a portion of the video and streams its cached content to the multicast channel. Based on these two mechanisms, the workload of the central server can be significantly reduced.

Because the hybrid approach is much suitable for providing large scale VoD services, we also bases on this hybrid framework to exploit the multicast/ broadcast capability of the network and P2P paradigm to efficiently deliver video data to the clients, which will be described in Chapter 5 and Chapter 6. In the proposed architecture, similar to CDN-P2P, a central server is deployed in order to avoid the disruption of the service caused by the dynamic nature of P2P applications. But, we also consider the use of broadcast capability in the network as DPCS and DDCM. However, unlike DDCM that the system will use the client buffer space as much as possible to reduce the use of the central server resources, our proposed framework intends to determine the optimal resource allocation on the unicast transmission and multicast delivery in the whole system such that the overall transmission cost of the system is minimized. Furthermore, we also consider fault exception which has not been studied in DDCM. Similar to DPCS that the duty of the broadcasting is dispersed among a number of peer servers. But, we also address the issues of reliability which has not been considered in DPCS. In addition, we also explore the relationship between the number of peer servers required and the bandwidth requirement of the central server.

In the following work, we will perform a number of simulations in order to verify the correctness of our model. The simulation program is developed in C++ using GSL software package (it can be found in http://www.gnu.org/software/gsl/). The simulated environment models a commercial video-on-demand system composed of thousands of users. The inter-arrival and the inter-departure time of clients are modeled by Poisson distribution as well as exponential distribution respectively which are generated by GSL. The simulator is an event-driven based. Each arrival or departure triggers an event. The event may cause to occupy the system resources or to release the system resources. If the system does not have enough resource to handle the event, this event will be blocked. We count the number of blocked events during the pre-defined simulation time to calculate the system blocking probability.

Chapter 3

Performance Analysis of Hierarchical Video-on-Demand Systems in Heterogeneous Environments

3.1 Introduction

With the advances in digital video technology, Video-on-Demand (VoD) systems have come into practice in recent years. Nevertheless, such systems have not yet been commercial success because of the high cost of implementation. The server and network requirements are still the limiting factors in the wide deployment of VoD services. Many works [LIU01, SERPA00] have thus tried to minimize the resources requirements as well as increase the system scalability. Currently, data broadcasting and proxy caching are the two orthogonal approaches to provide a cost-effective VoD service.

To support large-scale video streaming services, people exploited the broadcast capability of a network to share the system resources. Staggered broadcasting [WONG88] is the simplest broadcasting protocol proposed in the early days. Since the staggered broadcasting scheme suffered from a long start-up latency, some efficient broadcasting protocols such as skyscraper [HUA97], harmonic [JUHN97] and consonant [LIU03] were then developed to minimize the .start-up delay. In such broadcasting schemes, customers are required to receive data from several channels simultaneously and a buffer should be installed in each receiver. Taking the bandwidth capacity of the user into consideration, Yan et al. [YAN03] proved that the generalized Fibonacci broadcasting achieved the best performance among the known schemes. The results showed that efficient broadcasting protocols can support a nearly true VoD service and the waiting time can be reduced to as little as a few seconds. To implement a true (zero-delay) VoD system in a broadcast environment, patching [HUA98] and hierarchical stream merging (HSM) [TAN02] schemes were proposed. The idea of patching is that a client first downloads data on two channels simultaneously. After receiving the leading portion of the video, the client is then able to merge into one of the broadcasting channels. For the HSM scheme, the clients hierarchically merge with the broadcasting groups so that the bandwidth requirement can be further reduced compared with the patching protocol.

In addition to the broadcasting techniques, hierarchical architectures have also been explored to provide cost saving as well as increased quality of service to end users in a VoD system. In such hierarchical frameworks, video data can be temporarily stored in proxy servers so that the workload of the central server can be greatly alleviated. In [LI96], Li et al. developed a queuing model with the two-tier architecture to decide which video and how many copies have to maintain at each distributed server. Instead of storing the video programs as entity in the local servers, the "server caching" scheme [CHAN01] in a distributed system was proposed to pre-cache a portion of the video for the local customers. Wang et al. [WANG04] also addressed the problem of streaming videos from a remote server through a proxy and developed a generalized allocation technique to minimize the transmission cost.

Most of the previous works, however, mainly focused on providing the streaming services in a homogeneous environment, i.e. all users have the same traffic characteristics such as downstream bandwidth. In practical situations, access to the Internet is highly heterogeneous. Clients can connect to the network with different communication technologies such as modem, ASDL and wireless links. Their downstream rates may vary from 56kbps to 10Mbps. Different systems were thus proposed to deal with the problem of heterogeneity of the receiver capability. One of the approaches called replication [JIANG98] is that the servers support multiple quality video streams with identical content but at different data rates. The clients can therefore receive the appropriate video streams according to their network conditions. For example, the video encoded into low quality will be delivered to low bandwidth clients such as mobile users. On the other hand, the high quality video will be streamed to the high capacity receivers. Nevertheless, multiple versions of the same video can cause large increases in the amount of storage. Thus, some researchers argued that layered encoded videos [KANGA02, REJAI01] should be used to create multiple quality video streams. Although the storage requirement of the layering approach is much less than that of the replication technique, creating video layers generates additional bandwidth overhead [KIMUR99, KIM01, HARTA02]. In particular, for the same video quality, layered encoding typically requires more transmission bandwidth than does a replication.

In this chapter, we investigate possible solutions for building a large-scale VoD system in a heterogeneous network environment using both the broadcasting technique and hierarchical architecture. We compare video streaming of replication with that of layering in the proposed framework. The difference of this work from [KIM01] is that Kim et al. did not consider the proxy servers sitting between the central server and the clients. In [KANGA02], delivering layered videos using caches was investigated. However, the authors did not take into account replication and therefore did not provide any comparative results. Similar work was also presented in [HARTA02] but the authors neither explored the broadcast capability in the network nor verified the results by simulation. The main contribution of this chapter is that we explore the impact of the broadcasting schemes coupled with proxy caching and different coding schemes. In addition, we develop an analytical model to evaluate the system performance. The model can be applied to different system configurations such as centralized/distributed, unicast/broadcast as well as replication/layering. In addition, an extensive simulation is performed to verify the correctness of the model. We develop guidelines for resources allocation and identify the combination of transmission strategies and caching schemes that provide the best performance under different scenarios with heterogeneous requesting patterns. It is believed that the model can assist the system designers to study various design options and to perform system dimensioning.

This chapter is organized as follows. In Section 3.2, we discuss the system architecture for video streaming. In Section 3.3 and 3.4, we propose how the video replication approach and layered encoded videos can be applied to support the heterogeneous clients respectively. In addition, the performance model is derived as well. The simulation is then built and the results of both simulation and analytical models will be shown in Section 3.5. Finally, a summary of this chapter will be given in Section 3.6.

3.2 System Architecture

In this section, we describe how the proposed system provides video streaming services in the heterogeneous environment such as Internet. Figure 3.1 illustrates a typical two-tier VoD system which consists of one central server and several proxy servers. The central server, which has a large storage space to store all the available videos for clients, is connected to the proxy servers that are physically located closer to the clients. To meet clients' bandwidth requirements, video *m* will be encoded into *n* different quality levels. It

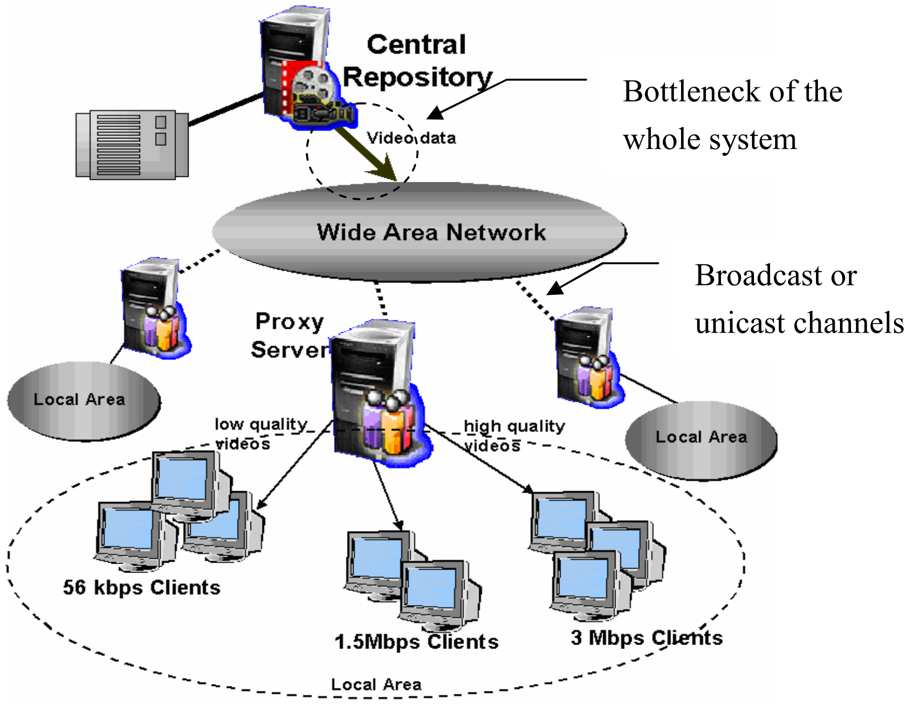


Figure 3.1. Hierarchical of VoD Architecture

is assumed that the proxy servers are independent and a large group of heterogeneous clients is served by a single proxy server.

In general, the proxy server caches the most popular videos for users' repeating requests in order to minimize the transmission cost. Upon the user's request received by the proxy server, it will acknowledge the request if the video has been already cached. Otherwise, it will bypass the request to the higher level. To cater for the heterogeneous requirement, the system will deliver different qualities of video streams to the clients according to their capacity constraints. If the clients have a low bandwidth connection such as 56Kbps, they will receive the videos encoded at a low bit rate. On the other hand, the high quality video will be streamed to the customers having the broadband access capability.

Because the storage capacity of the proxy server is limited, some popular videos cannot be cached and eventually should be delivered from the central server. It is clearly seen that the system is not scalable since the bandwidth requirement will linearly increase with the number of clients or the arrival rate. To further enhance the system performance, we also exploit the broadcasting capability in such a hierarchical architecture. Apart from storing the popular videos in the proxy server, some videos will also be broadcasted to the clients over the backbone network. Therefore, it is assumed that a generic network infrastructure that supports broadcasting operations is used to implement the broadcasting protocols.

In the proposed architecture, we assume that the I/O and network bandwidth of the proxy server is sufficient to serve all the video requests from the local area. Therefore, the bottleneck of the proposed system is the access bandwidth between the central server and the wide area network (as illustrated in Figure 3.1). For a given access bandwidth, only a finite number of requests can be served by the system. When a client issues a request for the system, the system first checks whether the requested video is stored in the proxy cache. If so, this request is accepted. Otherwise, the system further examines whether the requested item is delivered over the broadcasting channels. When it is not satisfied by the broadcasting channels, the central server will open a unicast channel for this request

Symbol	Meanings
M	Number of videos in the system
B	Bandwidth between the central and proxy servers (bits/s)
K	Proxy size (in bits)
λ	System arrival rate (reqs/s)
p_m	Popularity of video *m*
l_m	Number of quality levels of video *m*
r_j	Probability of customers requesting j^{th} quality of videos
λ_s^R	Arrival rate for the dedicated streams (reqs/s), replication and no broadcast
λ_s^{RB}	Arrival rate for the dedicated streams (reqs/s), replication and broadcast
c_{mj}^R	Streaming rate of replicated video m having j^{th} quality level (bits/s)
s_{mj}^R	Size of replicated video *m* encoded into j^{th} quality (bits)
d^R	Average rate of the dedicated streams (bits/s), replication and no broadcast
d^{RB}	Average rate for the dedicated streams (bits/s), replication and broadcast
χ^R	Bandwidth for broadcasting (bits/s), replication
λ_s^L	Arrival rate for the dedicated streams (reqs/s), layering and no broadcast
λ_s^{LB}	Arrival rate for the dedicated streams (reqs/s), layering and broadcast
c_{mj}^L	Streaming rate of layered video m having j^{th} quality level (bits/s)
η_{mj}^L	Streaming rate of layer *j* of video *m* (bits/s)
s_{mj}^L	Size of layer *j* of video *m* (bits)
d^L	Average rate for the dedicated streams (bits/s), layering and no broadcast
d^{LB}	Average rate for the dedicated streams (bits/s), layering and broadcast
χ^L	Bandwidth for broadcasting (bits/s), layering

Table 3.1. Summary of Notations

directly if there is available access bandwidth between the central server and the network. Otherwise, the request will be blocked. It is also assumed that the system will not check whether the resources can support a lower quality level of the requested video. In the following analysis, the blocking probability, which is measured as the probability that the user's request cannot be served due to insufficient access bandwidth of the central server, will be used as the performance metric for the system. To facilitate our discussion, we define notations listed in Table 3.1.

3.3 Replication Approach

In this section, we are going to describe the system using the replication approach to provide VoD services and develop an analytical model to evaluate the performance in terms of the blocking probability. Using replications, video m with j^{th} quality (v_{mj}) is encoded at a rate C_{mj}^R. Without a loss of generality, $C_{m1}^R < C_{m2}^R < \cdots < C_{ml_m}^R$. We first consider how to determine which video replicas are stored in the proxy server. It is assumed that r_j is the probability of the users requesting $v_{mj} \forall m$ where $\sum_{j=1}^{l_m} r_j = 1$. In the proposed system, the proxy server simply stores the most popular videos to maximize the cache hits. Define b_{mj} as the proxy map that is used to describe the subsets of video replicas in its cache. b_{mj} is set to 1 if v_{mj} is in proxy. Otherwise, it is set to 0. Therefore, with the objective of maximizing the cache hits, the optimization problem is then formally

1. Sort v_{mj} in the ascending order of $p_m r_j$ into stack where m=1, 2, 3, …, M and j=1, 2, 3, …, l_m.
2. Set *temp* = 0 and *cached* = 0
3. fetch the first element from the top of stack
4. while (*cached* < *K* and *all elements in the stack are not fetched*)
5. Set m = the video id referred in the fetched element
6. Set j = the replica id referred in the fetched element
7. Set *temp* = the data size of the corresponding video replica referred in the fetched element
8. if ($temp + cached \le K$)
9. *cached = cached + temp*
10. Set b_{mj} = 1
11. end if
12. fetch next element from the top of stack
13. end while

Table 3.2. Algorithm for Proxy Caching

stated as follows.

$$\text{Maximize :} \quad \sum_{m=1}^{M}\sum_{j=1}^{l_m} p_m r_j\, b_{mj}$$

$$\text{Subject to} \quad \sum_{m=1}^{M}\sum_{j=1}^{l_m} s_{mj}^{R} b_{mj} \le K$$

To efficiently determine which video replicas should be cached such that the cache hits can be maximized, we develop an algorithm for proxy caching as shown in Table 3. 1. The algorithm starts with sorting v_{mj} in the ascending order of popularity $p_m r_j$ into the stack. Each element in the stack is composed of (v_{mj}, $p_m r_j$) pair. For each iteration, the algorithm fetches one element from the top of stack. If the proxy cache has enough space to store the video replica indicated in the fetched element, the cache map b_{mj} for this video replica is set to 1. Otherwise, the algorithm fetches the next element until all the proxy cache space is occupied.

Once b_{mj} has been found by maximizing the cache efficiency, we can determine the fraction of requests that goes up to the central server for the dedicated streams. Since a portion of requests is satisfied by the proxy server, the arrival rate of the requests going to the central server can be computed by eqn. (3.1)

$$\lambda_s^R = \lambda(1 - \sum_{m=1}^{M} \sum_{j=1}^{l_m} p_m r_j b_{mj}) \tag{3.1}$$

Since multiple qualities of video streams are delivered at different data rates from the central server to the clients, the average streaming rate can thus be calculated by eqn. (3.2).

$$d^R = \frac{\lambda}{\lambda_s^R} \sum_{m=1}^{M} \sum_{j=1}^{l_m} p_m r_j c_{mj}^R \overline{b}_{mj} \tag{3.2}$$

where

$\overline{b}_{ij}$: complement of b_{ij}

To deliver a large number of video streams, the central server is generally the bottleneck of the whole system (refer to Figure 3.1). Thus, we particularly focus on the performance of the central server. Denote B as the available bandwidth between the central server and the proxy server. On average, the central server can thus simultaneously support N^R number of video streams where $N^R = \left\lceil \frac{B}{d^R} \right\rceil$. Assume that the service time of each video stream is exponentially distributed with mean T (service rate $\mu = \frac{1}{T}$) by considering the varying length of different videos. The system can be modeled as an $M/M/N^R/N^R$ queueing system [PRABH97] and the blocking probability is equal to

$$P^R = \frac{(\lambda_s^R / \mu)^{N^R} / N^R!}{\sum_{j=0}^{N^R} (\lambda_s^R / \mu)^j / j!} \tag{3.3}$$

If the bandwidth from the proxy server to the clients is large enough and no requests will be blocked, the overall blocking probability of the system is given by eqn. (3.4).

$$P_{overall}^R = \frac{\lambda_s^R P^R}{\lambda} \tag{3.4}$$

Apart from storing the popular videos in the proxy server, some replicated videos will also be broadcasted to the clients over the backbone network. For example, a low quality video is delivered over the broadcasting channels but the same video encoded at higher data rate is transmitted to the clients through the dedicated streams. Then, we should determine which video can be delivered over the broadcast channels. Since our focus is on the performance of the whole architecture, the broadcasting techniques are not our major concern. In general, any efficient protocols, such as [HUA97, JUHN97, LIU03], can be applied to the system framework. Because the bandwidth requirement of the broadcasting protocols depends on the transmission schedule and user bandwidth constraints, H^x is denoted as the number of channels required for the protocol x to broadcast a video such that the start-up delay is insensitive to the clients. It is further assumed that the receiver buffer is large enough to implement the efficient broadcasting protocol. To determine which video replicas should be sent over the broadcasting channels, w_{mj} is used to indicate whether v_{mj} is broadcasted or not. Then, the bandwidth required for broadcasting can be calculated by eqn. (3.5).

$$\chi^R = \sum_{m=1}^{M}\sum_{j=1}^{l_m} c_{mj}^R H^x \overline{b}_{mj} w_{mj} \tag{3.5}$$

Similar to proxy caching, we assign v_{mj} to the broadcasting channels according to their popularity. For example, v_{mj} with the highest popularity will be first broadcasted. Given the bandwidth reserved for broadcasting (B_{rsv}), w_{mj} can be found such that the broadcasting bandwidth does not exceed the reserved capacity, i.e. $\chi^R \leq B_{rsv}$. Because some of the replicated videos are being broadcasted, the arrival rate for the dedicated channels is given by eqn. (3.6) which is equal to the arrival rate to the system minus the arrival rate to the proxy server as well as the arrival rate to the broadcast channels.

$$\lambda_s^{RB} = \lambda(1 - \sum_{m=1}^{M}\sum_{j=1}^{l_m} p_m r_j b_{mj} - \sum_{i=m}^{M}\sum_{j=1}^{l_m} p_m r_j \overline{b}_{mj} w_{mj}) \tag{3.6}$$

The average streaming rate of the dedicated channels can thus be found by eqn. (3.7).

$$d^{RB} = \frac{\lambda}{\lambda_s^{RB}} \sum_{m=1}^{M}\sum_{j=1}^{l_m} p_m r_j c_{mj}^R \overline{b}_{mj} \overline{w}_{mj} \tag{3.7}$$

As B is the available bandwidth, the number of streams that can be concurrently supported by the server is found by $N^{RB} = \left\lceil \frac{B - \chi^R}{d^{RB}} \right\rceil$. Similar to eqns. (3.3) and (3.4), the $M/M/N^{RB}/N^{RB}$ queue can be applied and the overall blocking probability can be found accordingly.

3.4 Layering Approach

It is argued that layering can achieve better performance than replication. In this section, we extend the model to investigate the system using layered encoded videos. In a heterogeneous environment, layered encoding can also provide multiple qualities of video streams. For instance, if the clients have a low bandwidth connection, they can simply receive the base layer of the videos. On the other hand, both base and enhancement layers of the videos will be delivered to the clients who have the broadband access capability. Refer to Figure 3.1 again, when the proxy server cannot support the requested quality of video streams, it will deliver the cached layer(s) to the client and the missing enhancement layer(s) is retrieved from the central server directly. Similar to the replication approach, if the central server does not have sufficient bandwidth to stream the missing layer(s), the client will be blocked from the system.

The reports in [KIMUR99, KIM01] showed that a non-layered stream has better video quality than a layered stream at the same data rate. Specifically, for the same quality level, the layered video will incur around 20%-30% overhead compared with the non-layered video and as a result the system requires more transmission bandwidth. With a view to making a fair comparison between layering and replication, we assume β to be the overhead of the layered videos where $\beta \geq 0$. The relationship of the streaming rate of

v_{mj} between these two approaches is given by eqn. (3.8).

$$\sum_{k=1}^{j}\eta_{mj}^{L} = c_{mj}^{L} = (1+\beta)c_{mj}^{R} \tag{3.8}$$

Different from the replication approach, the proxy server will cache the layers of the videos instead of the video replicas at different quality levels. Considering the property of the layered video, i.e. all the lower quality layers must be stored before caching the enhancement layers, we define q_{mj} as the fraction of users requesting the j^{th} layer of video *m* such that $q_{mj} = \sum_{k=j}^{l_m} r_k$ [GUO01]. Because the proxy server stores the most popular layers of the videos to increase the cache hits, the problem is to maximize $\sum_{m=1}^{M}\sum_{j=1}^{l_m} p_{mj}q_{mj}b_{mj}$ subject to $\sum_{m=1}^{M}\sum_{j=1}^{l_m} s_{mj}^{L}b_{mj} \le K$. It is noted that $b_{mj} = 1$ here represents layer *j* of video *m* that is cached in the proxy server and b_{mj} can also be determined by the algorithm shown in Table 3.2. Once b_{mj} has been found, eqn. (3.1) can be used to compute λ_s^L. Because the client for v_{mj} will receive *j* layered video streams simultaneously, the average streaming rate is equal to

$$d^{L} = \frac{\lambda}{\lambda_s}\sum_{m=1}^{M}\sum_{j=1}^{l_m} p_m r_j \left(\sum_{k=1}^{j}\eta_{mk}^{L}\bar{b}_{mk}\right) \tag{3.9}$$

To determine which layers of the videos should be sent over the broadcasting channels, g_m is denoted as the highest layer of video *m* using the broadcasting scheme such that the j^{th} layer of video *m*, where $j \le g_m$, is either broadcasted to the customers or stored in the proxy server. Then, the broadcasting bandwidth can be calculated by eqn.

(3.10).

$$\chi^L = \sum_{m=1}^{M}\sum_{j=1}^{g_m}\eta_{mj}^L H^x \overline{b}_{mj} \tag{3.10}$$

g_m can therefore be found on condition that $\chi^L \le B_{rsv}$ and λ_s^{LB} is equal to

$$\lambda_s^{LB} = \lambda(1 - \sum_{m=1}^{M}\sum_{j=1}^{l_m} p_m r_j b_{mj} - \sum_{i=m}^{M}\sum_{j=1}^{g_m} p_m r_j \overline{b}_{mj}) \tag{3.11}$$

Since some video requests only retrieve the enhancement layers from the central server, the average bandwidth of the dedicated streams can thus be computed by eqn. (3.12).

$$d^{LB} = \frac{\lambda}{\lambda_s^{LB}}\sum_{m=1}^{M}\sum_{j=g_m+1}^{l_m} p_m r_j (\sum_{k=g_m+1}^{j}\eta_{mk}^L \overline{b}_{mk}) \tag{3.12}$$

Similar to the replication approach, the number of streams that can be supported by the server with and without broadcasting is found by $N^{LB} = \left\lceil \frac{B-\chi^L}{d^{LB}} \right\rceil$ and $N^L = \left\lceil \frac{B}{d^L} \right\rceil$ respectively. The $M/M/N^L/N^L$ and $M/M/N^{LB}/N^{LB}$ queue can therefore be applied and the overall blocking probability can be calculated by eqns (3.3) and (3.4). It should be noted that, with the layering approach, the clients may be required to delay the start of video playback because some layers retrieved from the broadcasting channels may not be synchronized with other layers streamed dedicatedly from the central server or the proxy server. However, such delay is negligible as the start-up latency of an efficient broadcasting protocol is insensitive to the clients.

3.5 Experimental Results

Computer simulations are performed to study the performance of the proposed hierarchical framework and verify the results from the analytical model. The simulated environment models a commercial VoD system serving thousands of users with five different bandwidth capacities. Based on their bandwidth capacity, we define two video quality requesting patterns which will be stated in Table 3.4 to simulate the heterogeneous environment. In the simulation, there are 200 videos for which each of them is encoded into five quality levels and stored at the central database. The client requests are modeled as the Poisson arrival process and the video popularity is followed by Zipf's distribution [ZIPF49] with the skew parameter $\theta = 0.271$. Then, $p_m = \frac{\Omega}{m^{1-\theta}}$ where $\Omega = \frac{1}{\sum_{i=1}^{M} \frac{1}{i^{1-\theta}}}$, Since the Erlang's loss formula [MEDHI94] holds for any distribution of service time (having mean 1/μ) provided that the input is Poisson, i.e. it holds for the model *M/G/N/N*, it is assumed that the length of each video is fixed as 90 minutes. The environment is simulated as long as 240 hours. The blocking probability is defined as the ratio of the number of rejected requests to the total number of video requests. For the replication approach, the client for v_{mj} will be served by the proxy server or the broadcasting channels if the requested video has been cached or broadcast. Otherwise, the client will retrieve the video from the central server. For the layering approach, when the client issues a video request, he/she will be served by the proxy server if all the requested layers of the video are cached. Otherwise, he/she will

Parameter	Nominal Value	Range
No of videos (M)	200	-
System arrival rate (λ)	0.3, 0.8	0.1 – 1 reqs/s
Proxy size (K)	5%	0% – 10%
No of broadcast channels (H^x)	10	4 – 14
Access bandwidth of the central video server (B)	100Mbps	-
Layering overhead (β)	0.25	0 – 1
Proportion of bandwidth reserved (p_{rsv})	0.1	0 – 1
Layer stream rate (η_{mj}^{L})	η_{m1}^{L}	-

Table 3.3. Parameters of the Simulation

listen to the broadcasting channels for the missing video layers. If the missing layers are not being broadcasted, the client will open a dedicated stream from the central server. Assume that the streaming rate of the base layer of all the videos is $\eta_{m0}^{L} = 56Kbps$. In order to provide the high quality of video streams for the high capacity receivers, some video information is encoded in the enhancement layers. It is assumed that all layers that have the same rate [REJAI00], i.e. $\eta_{mj}^{L} = \eta_{m1}^{L}$. As the backbone bandwidth is fixed, it is further assumed that the proportion of bandwidth, p_{rsv}, is reserved for broadcasting, i.e. $B_{rsv} = p_{rsv}B$. The results in [YAN03] reported that less than 10 broadcasting channels are sufficient to provide delay insensitive VoD services. Therefore, H^x is set to 10 for the following experiments unless other specified. From the results in [KIMUR99], the amount of overhead incurred by the layered encoded videos is varied from 0 to 30%. Table 3.3 summarizes the parameters used in the simulation.

In order to evaluate the performance of the system with heterogeneous clients, two

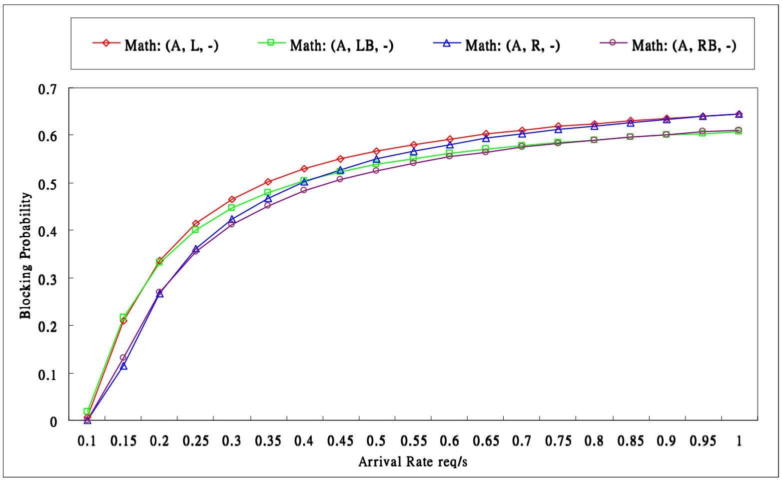
Scenario A (S-A)	*2^{nd} and 5^{th} qualities of videos are requested equally ($r_2 = r_5 = 0.5$, $r_1 = r_3 = r_4 = 0$)*
Scenario B (S-B)	*All videos qualities are requested uniformly ($r_1 = r_2 = r_3 = r_4 = r_5 = 0.2$)*

Table 3.4. Requesting Patterns of Clients

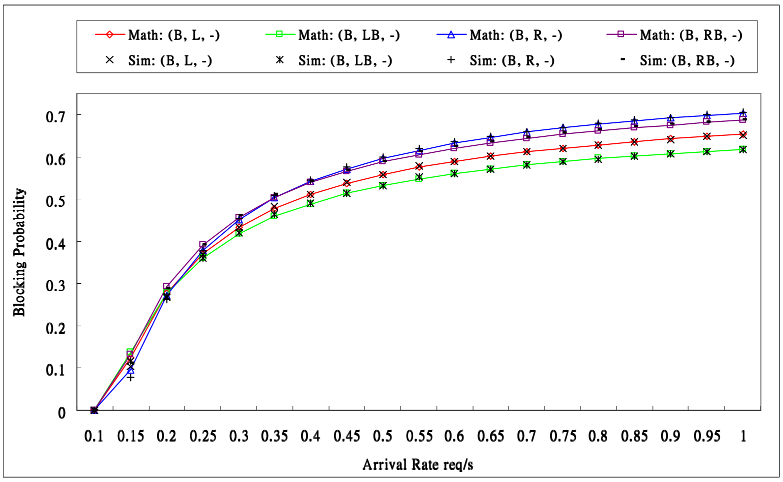
requesting patterns will be considered as shown in Table 3.4. S-A is to model the less heterogeneity environment (i.e. the system serves only two types of client) while S-B focuses on considering for highly heterogeneity environment (i.e. the system serves five types of client):

In the following, each curve is represented by "Model[2]: (x, y, z)" where "x=requesting scenario", "y=system configuration[3]" and "z=arrival rate". We first evaluate the performance impact of system arrival rates to the blocking probability when *B* is 100Mbps and the result is illustrated in Figure 3.2. The figures show that our mathematical model is closely matched with the simulation results under various system configurations. As we expected, more customers are blocked by the system when the arrival rate is increasing. It can be found in Figure 3.2(a) that the performance of the layering approach is worse than that of the replication approach when the arrival rate is low even though layering requires less storage capacity. When the arrival rate is low, the system does not take much advantage from proxy caching and thus most of the requests will be handled by the

[2] Math – results obtained from mathematics model, Sim – results obtained from simulation model
[3] R=Replication, L=Layering, B=Broadcasting



(a) S-A

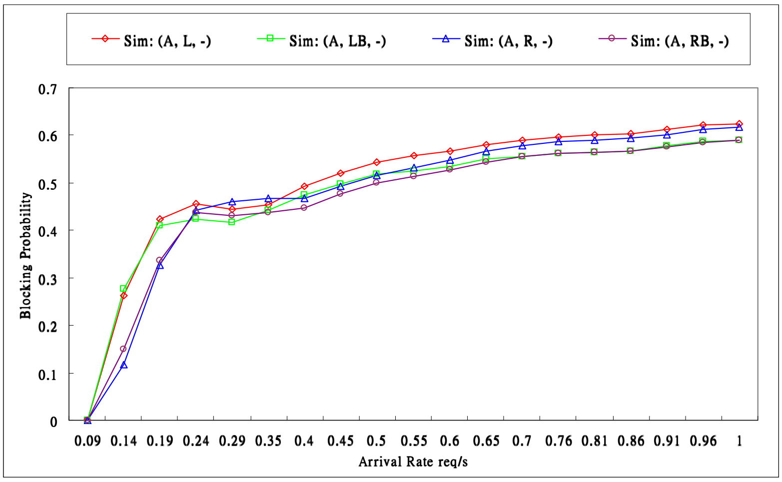


(b) S-B

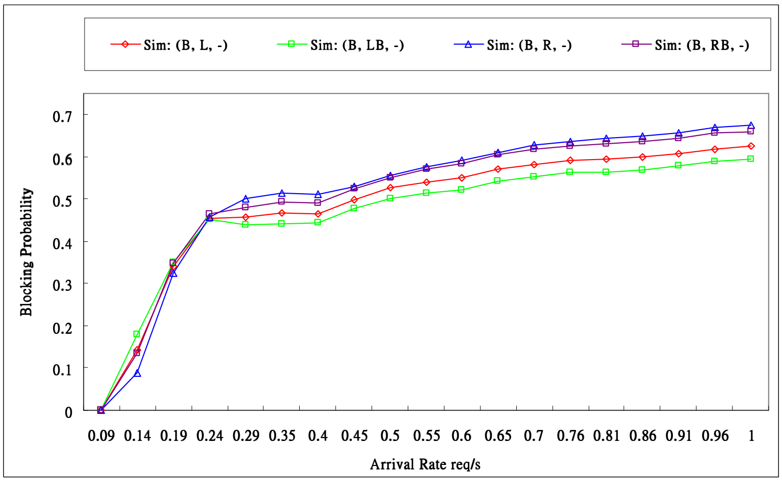
Figure 3.2 Blocking Probability against Arrival Rate

central server. Because layering requires more transmission bandwidth than replication for the same quality level, fewer video channels can be allocated by the server simultaneously and thus more customers are blocked. However, when the arrival rate is further increasing, the cache hit rate is increased. Since more videos can be cached with the layering approach, the performance between layering and replication is then getting close. Figure 3.2(b) shows the blocking probability of the system under “S-B”. It is found that layering can achieve better performance than replication for different arrival rates. It is expected that, under the layering approach, more videos can be cached in the proxy server. Hence, most of the requests can be satisfied by the proxy server. If the video quality can be adapted to the customers' environment, the system can serve more customers. These results are consistent with [HARTA02] that replication is favorable to less heterogeneous environment while layering is suitable for highly heterogeneous environment. In the figures, the impact of broadcasting is also examined. It is observed that the performance cannot be significantly improved under light traffic such as 0.2 requests/s. When the arrival rate is further increased, the system with broadcasting can obtain better performance than the system without broadcasting because broadcasting favors the popular videos. On the other hand, it can be found that layering with broadcasting can perform better than replication without broadcasting when the arrival rate is higher than 0.45 req/s in S-A. Therefore, layering with broadcasting can still obtain better performance under less heterogeneous environment when the arrival rate is high.

Figure 3.3 shows the simulation results with dynamic load, where the arrival rate is dynamically changed during the simulation but the overall arrival rate is kept constant. From the results, we can see that the same conclusion can be drawn (i.e. layering approach performs better than replication approach in highly heterogeneous network environment).



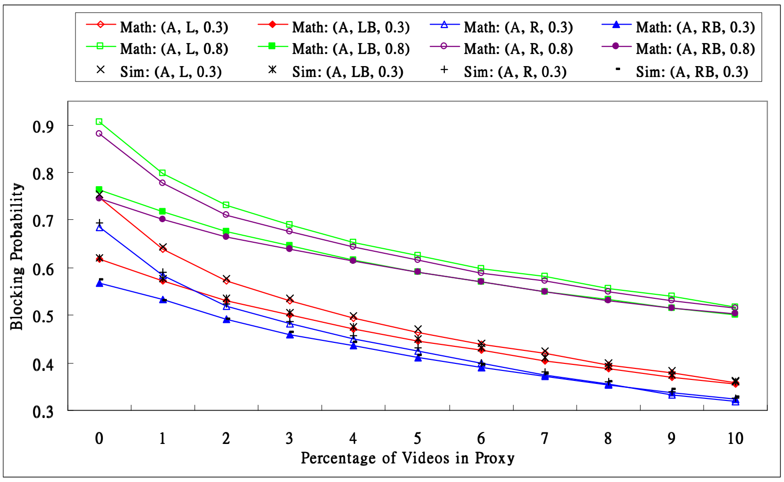
(a) S-A



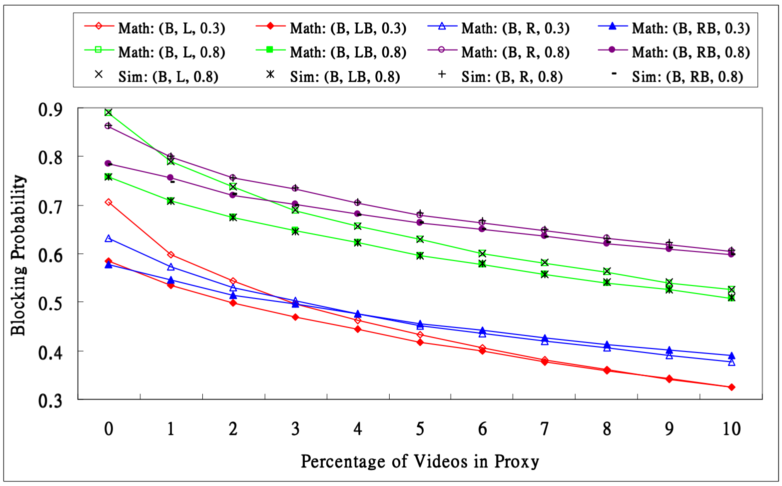
(b) S-B

Figure 3.3 Blocking Probability against Arrival Rate with Dynamic Load

From the model, apart from the arrival rate, it can be observed that the proxy size (*K*), the efficiency of the broadcasting scheme (H^x) and the bandwidth reserved for broadcasting (B_{rsv}) also bring a great impact on the system performance. In order to have a close look on the effectiveness of the broadcasting protocol to the system, Figures 3.4 to 3.6 show the blocking probability of the systems when these parameters are varied. Figure 3.4 illustrates the system blocking probability against the proxy size. Increasing the proxy size results in fewer video requests passed to the central server and thus more customers can be served. In “S-A”, the video requests for replicated video streams can obtain better performance when the proxy size is small. The reason is that when the proxy size is small, most of the requests should be handled by the central server but the layering overhead results in higher streaming rate of the videos. We can see that the performances of replication and layering are getting close under high traffic when the proxy size is further increased. The similar trend can be observed in “S-B” which is shown in Figure 3.4(b). However, in this case, the layering approach can obtain better performance when the proxy size is just larger than 2% since the layered videos can use the storage capacity in a more efficient way. The results show that the blocking probability of the layering approach can be less than that of the replication approach up to 10% when the proxy capacity is further increasing. It should be noticed that replication obtains better performance than layering when the proxy size is zero (i.e. no proxy). This result is consistent with [KIM01] when there is no proxy server sitting between the central server and the clients. On the other hand, it is also found that the system with either replication or layering performs better if the broadcasting scheme is incorporated. However, the gain is insignificant when the proxy size is too large as all the popular videos are locally cached and the system only broadcasts the less popular videos.



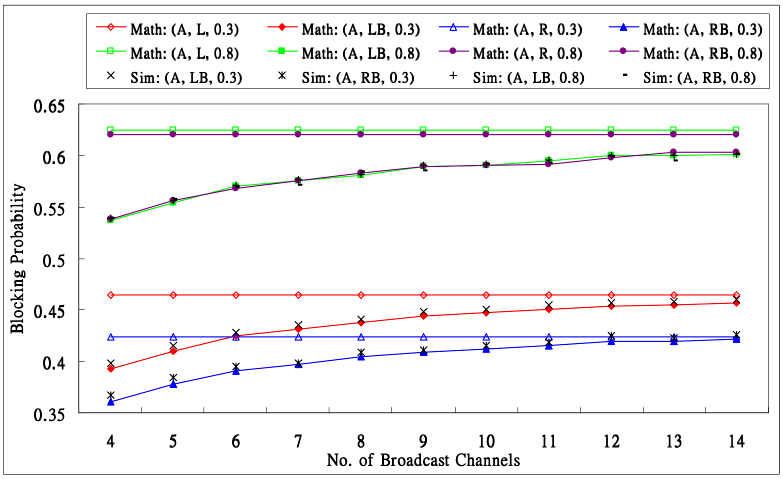
(a) S-A



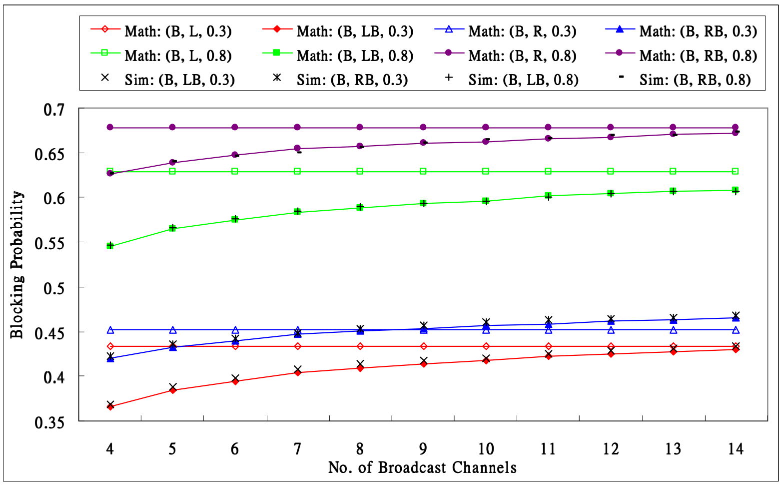
(b) S-B

Figure 3.4 Blocking Probability against Proxy Size

Next we consider how broadcasting affects the system performance. Figure 3.5 illustrates the blocking probability against the number of broadcasting channels. The more efficient the broadcasting scheme is, the fewer the broadcasting channels are required. Therefore, the blocking probability is lower when fewer number of broadcasting channels is used. Similar to the previous results, the performance of replication is better than that of layering in “S-A” because layering consumes more transmission bandwidth than replication when the environment is not highly heterogeneous. In contrast, as shown in



(a) S-A

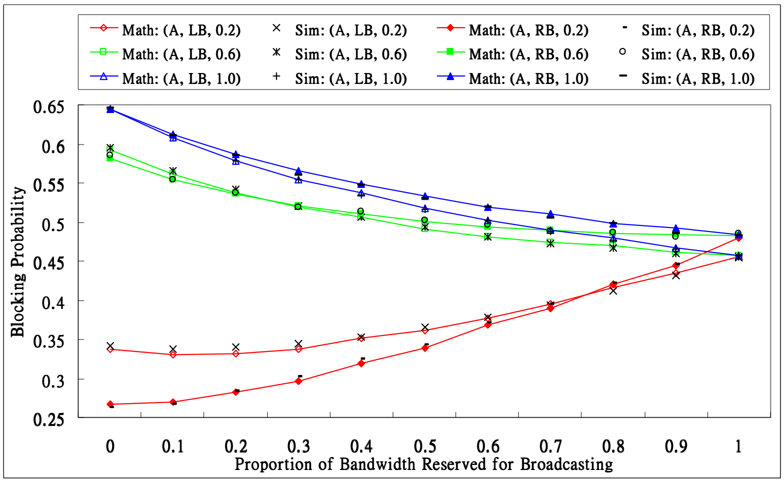


(b) S-B

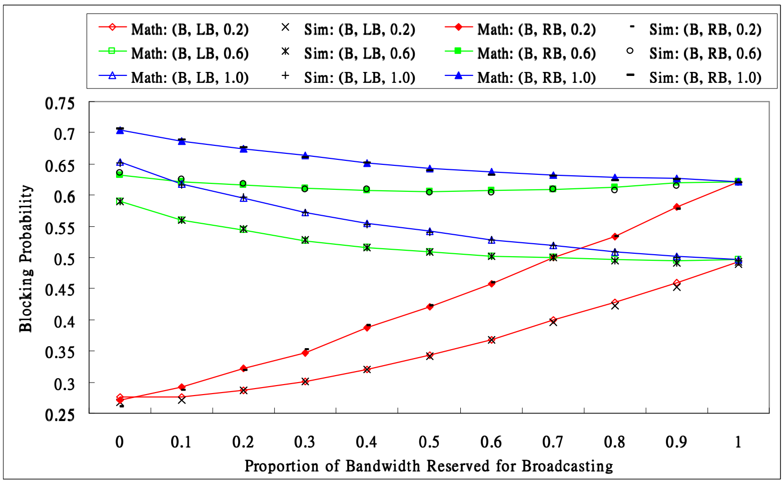
Figure 3.5 Blocking Probability against Broadcasting Efficiency

Figure 3.5 (b), the layering approach performs better if the system needs to provide five different qualities of video streams. It can be found that the blocking probability goes convergence when the number of broadcasting channels is further increased. It is because the reserved bandwidth for broadcasting is fixed (i.e. B_{rsv}) and more broadcasting channels assigned to a video layer or a replicated video results in reducing the number of videos that can take the benefit from broadcasting. It is also the reason that the blocking probability of the system with broadcasting is higher than that of the system without broadcasting approaches if the number of broadcasting channels is further increasing (e.g. Math(B, RB, 0.3)). In addition, as we expected, the system performance is better when the arrival rate is high. From the results in Figure 3.5(b), the broadcasting scheme can still reduce the blocking probability when the arrival rate is 0.8 requests/s even if more than 10 channels are used to broadcast a video. It is noted that, by using some recently proposed broadcasting protocols, 6 to 10 channels are sufficient to provide delay insensitive VoD services.

Now we plot the blocking probability when the proportion of bandwidth reserved for broadcasting is changed from 0 to 1 in Figure 3.6. To demonstrate the results with low, medium and high traffic, we now set the arrival rate as 0.2, 0.6 and 1.0 requests/s respectively. If p_{rsv} is 0, there is no bandwidth reserved for broadcasting. On the other hand, all the available bandwidth will be used for broadcasting if p_{rsv} is 1. It can be seen that the systems are getting worse with the increase of bandwidth reserved for broadcasting at low traffic because broadcasting does not favor to the systems with low arrival rate. At high traffic, the system blocking is strictly decreasing. However, when all the bandwidth is reserved for broadcasting, the performance of various system configurations converges. It is because only a portion of the videos can be delivered to the customers and some less popular videos will never be served. In Figure 3.6(a), we can observe that the blocking



(a) S-A

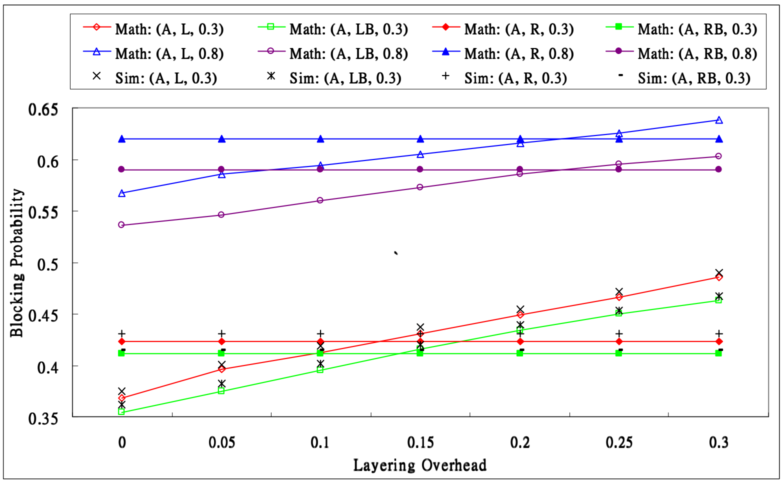


(b) S-B

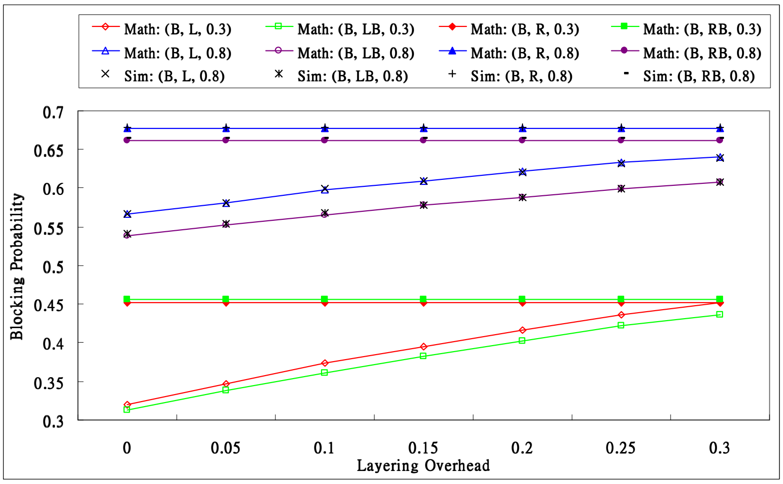
Figure 3.6 Blocking Probability against Reserved Broadcasting Bandwidth

probability of the layering approach is higher than that of replication but it performs better when the bandwidth reserved for broadcasting is increasing. One of the reasons is that most of the requests are still handled by the central server when p_{rsv} is low. Layering overhead is the main drawback of end-to-end delivery. However, since the layering approach can utilize the broadcasting bandwidth more effective than replication when p_{rsv} is high such that the layering approach can assign more videos on the broadcasting channels than replication does, more clients can download the video data from the broadcasting channels. It is found that when all the bandwidth are reserved for broadcasting, the blocking probability of "Math: (A, LB, 0.2)" is about 3% lower than that of "Math; (A, RB, 0.2)". In contrast, when no bandwidth are reserved, the blocking probability of "Math: (A, LB, 0.2)" is about 7% higher than that of "Math: (A, RB, 0.2)". It is noted that the gain from broadcasting is more obvious if the storage capacity of the proxy server is not large enough to store all the popular videos.

Finally, we look into the effect of layering overhead. The overhead is varied from 0% to 30%. Figure 3.7 demonstrates the blocking probability of the systems against layering overhead. As expected, more clients will be blocked from the systems with higher layering overhead. It is shown that the layering approach should be used in a highly heterogeneous environment. In the case of "S-B", layering is still better than replication even if 30% of layering overhead is incurred. However, for the system with only two types of clients like "S-A", the replication approach should be adopted if the overhead is larger than 20%. It should be noticed that the system still has superior performance when the broadcasting protocol is implemented for various layering overhead factors.



(a) S-A



(b) S-B

Figure 3.7 Blocking Probability against Layering Overhead

3.6 Summary

Video transmission over Internet is one of the hottest research topics in recent years. One of the challenges to provide VoD services is how the video streams can be delivered in a heterogeneous network environment. In this chapter, we consider using the replication and layering approach to create multiple qualities of video streams. We developed the large-scale video streaming system using both the proxy caching and data broadcasting techniques. In the proposed system, either replicated or layered videos are streamed to the clients according to their bandwidth constraints. An analytical model was developed to compare their performance in terms of the blocking probability. From this model, it has been found that the proxy size, the efficiency of the broadcasting scheme, the bandwidth reserved for broadcasting as well as the layering overhead have significant impacts on the system performance. The results showed that the blocking probability of layering is lower than that of replication when the environment is highly heterogeneous, i.e. the system needs to support five different qualities of video streams at the same time. In addition, it has been observed that the system performance can be further improved by exploring the broadcast capability of the network if the proxy server cannot store all the popular videos.

Chapter 4

Hierarchical Video-on-Demand Systems with Hybrid Coding Scheme and Quality Renegotiation

4.1 Introduction

In chapter 3, we investigated possible solutions for building a large-scale VoD system in heterogeneous network environments using the hierarchical architecture. In such architecture, the proxy server serves most of the popular videos to clients directly such that the workload of the central server can be alleviated. In order to effectively use the server resources as well as improve the scalability of the system, the broadcast capability of a network is also exploited that video contents are distributed along a number of video channels shared among clients. To meet different clients' bandwidth constraints, videos are encoded into a number of different quality levels with replication or layered encoding. We investigated the efficiency of the system by developing an analytical model to effectively compare the performance of the replication approach and that of layering for video streaming in the proposed framework under different scenarios and parameter settings. From the results, it can be found that the layering approach is suitable for proxy caching and video broadcasting while replication is favorable to end-to-end transmission. The results introduce an interesting question whether the system performance can be further improved if both coding schemes are deployed in different parts of the system based on their natures. Thus, in this chapter, we are going to investigate a complementary approach using both video replication and layering for video streaming such that the blocking probability of the system can be reduced. On the other hand, the client is blocked when the system resources cannot satisfy client's request in the previous proposed architecture. However, this request may still be accepted if the system resources can support a lower quality level of the requested video. Therefore, we also investigate the benefit of renegotiation about the video quality under limited system resources in this chapter.

The remaining portion of this chapter is organized as follows. In Section 4.2, we describe how video replication and video layering are deployed in the system. In Section 4.3, we consider renegotiation about video quality that the client is no longer blocked immediately when the system can support a lower quality of the requested video. A mathematical model is derived to show the efficacy of these approaches analytically. Computer simulations are then performed to evaluate the performance and the results of both simulation and analytical models will be presented in Section 4.4. Finally, some concluding remarks of this chapter are given in Section 4.5.

Symbol	Meanings
λ_s^{LBR}	Arrival rate for the dedicated streams (reqs/s), layering, replication and broadcast
d^{LBR}	Average rate for the dedicated streams (bits/s), layering, replication and broadcast
χ^{LBR}	Bandwidth for broadcasting (bit/s), layering, replication and broadcast
z_m	An indicator that indicate whether video *m* is layered-encoded
$\underline{Q}$	The expected proportion of the requested qualities of the video that can be delivered to the clients

Table 4.1. Summary of Notations

4.2 Hybrid Coding Strategy

In this section, we describe a complementary approach using both video replication and layering for video streaming. The system architecture remains unchanged which has been described in Section 3.2 and depicted in Figure 3.1. In this section, we assume that clients are able to decode the video for playback no matter what coding scheme is used for the video. To facilitate our discussion, the notations defined in Table 3.1 are still applied and we also define new symbols listed in Table 4.1.

The problem here is how to determine which coding scheme should be used for a particular video. Since layered-encoded video is favorable to proxy caching and video broadcasting, details of Section 3.4 can be applied to decide which layers of the videos are cached in proxy server or broadcasted over broadcasting channels. Once b_{mj} and g_m are found, the broadcasting bandwidth χ^{LBR} and the arrival rate to the central server λ_s^{LBR}

can thus be calculated by eqn. (3.10) and eqn. (3.11) respectively. When λ_s^{LBR} has been computed, we can determine the number of the dedicated streams that can be supported by the central server. However, as mentioned before, replication is favor to end-to-end transmission. Therefore, rest of the videos should be encoded by the replication approach if it is neither cached in the proxy server nor delivered over the broadcasting channels. To indicate whether video *m* is layered-encoded, we denote z_m as an indicator which is set to 1 if $\sum_{i=1}^{l_m} b_{mj} > 0$ or $g_m > 0$ and 0 otherwise. The average bandwidth of the dedicated streams can thus be computed by

$$d^{LBR} = \frac{\lambda}{\lambda_S^{LBR}} \sum_{m=1}^{M} \left(z_m \sum_{j=g_m+1}^{l_m} p_m r_j \left(\sum_{k=g_m+1}^{j} \eta_{mk}^L \overline{b}_{mk} \right) + \overline{z}_m \sum_{j=1}^{l_m} p_m r_j c_{mj}^R \overline{b}_{mj} \right) \tag{4.1}$$

The first term of eqn. (4.1) shows the average bandwidth requirement of the video which is layered-encoded while the second term illustrates that of the video which is encoded by replication approach. Therefore, the number of the streams that can be sustained by the server is $N^{LBR} = \left\lceil \frac{B - \chi^{LBR}}{d^{LBR}} \right\rceil$. Then, the blocking probability of the central server is equal to

$$P^{LBR} = \frac{(\lambda_s^{LBR} / \mu)^{N^{LBR}} / N^{LBR}!}{\sum_{j=0}^{N^H} (\lambda_s^{LBR} / \mu)^j / j!} \tag{4.2}$$

and the overall blocking probability of the system can be found by eqn. (4.3).

$$P_R^{LBR} = \frac{\lambda_s^{LBR} P^{LBR}}{\lambda} \tag{4.3}$$

4.3 Quality Renegotiation

In this section, we investigate the benefit of renegotiation about the video quality when the system resources are limited. In the original framework, the client is blocked immediately when the system resources cannot satisfy the client's request. However, this request may still be accepted if the system resources can support a lower quality level of the requested video. It is first assumed that the client will always accept the lower quality level of video when the original request is rejected due to lack of system resources. Define *Quality Perception* (*Q*) as the expected proportion of the requested quality of the video that can be delivered to the clients, which can be expressed by

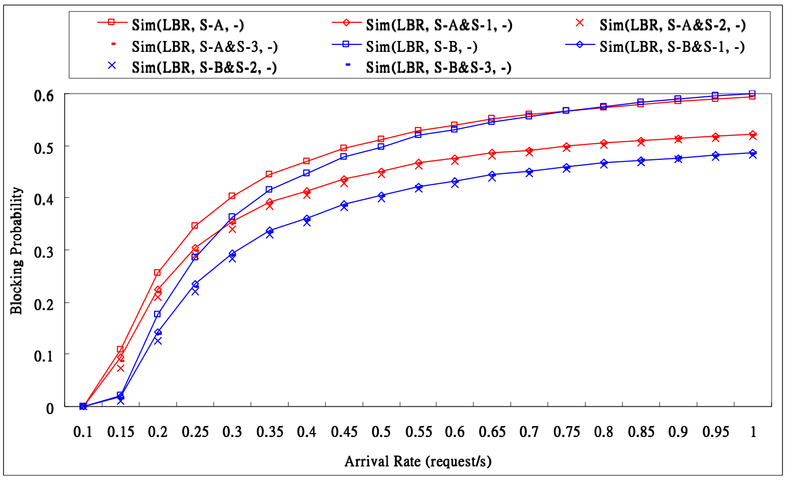
$$Quality\ \ Perception\ \ (Q) = \frac{number\ of\ \ quality\ \ level\ \ received}{number\ of\ \ quality\ \ level\ \ requested}$$

Therefore, Q is equal to 1 if the system can completely satisfy what the client requests. There are several strategies to be considered for the renegotiation mechanism, i.e.

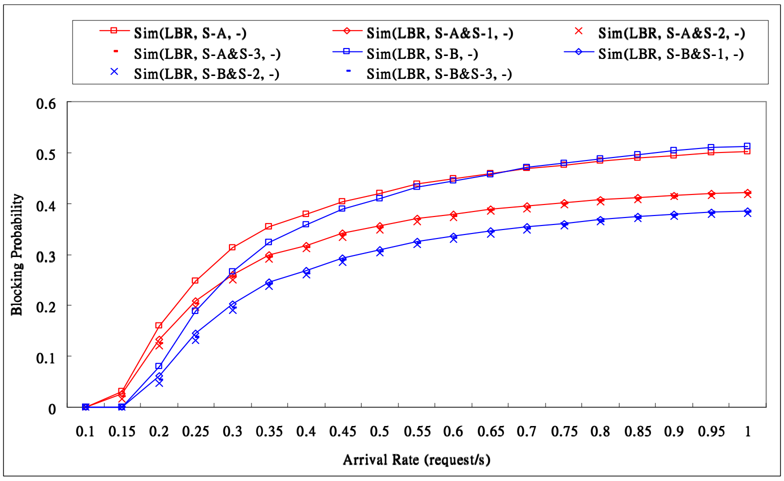
S-1. Provide the video which is cached in proxy or delivered over the broadcasting channels only.

S-2. Provide the lowest quality level by the central server if the requested layers are neither cached in proxy nor broadcasted (S-1 is still applied).

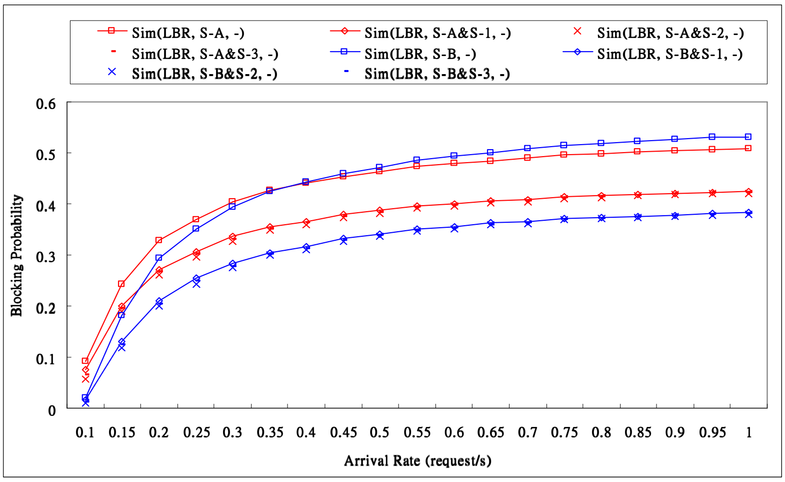
S-3. Based on the current resources, provide the highest quality level of video to the requested client (S-1 is still applied).



(a) Proxy Size = 5%, Broadcast Efficiency = 0.1



(b) Proxy Size = 10%, Broadcast Efficiency = 0.1



(c) Proxy Size = 5%, Broadcast Efficiency = 0.5

Figure 4.1. System Performance in various Renegotiation Mechanisms

We first use the simulation to show the efficacy of these strategies in various scenarios as shown in Figure 4.1. It can be found that S-1 dominates the major improvement and the others only provide less than 1% further enhancement in various system parameters. Based on this finding, we use S-1 to approximate the system performance with renegotiations. Since the request will be relayed to the central server only when the original request is rejected, meanwhile, it cannot be satisfied by the proxy and broadcast channels, the overall blocking probability of the system with quality renegotiation ($P_{overall}^{Q}$) can thus be determined by

$$P_{overall}^{Q} = \left(\sum_{i=1}^{M} \sum_{j=1}^{l_i} p_i \, r_j \prod_{k=1}^{j} \overline{b}_{ik} \mid \overline{\eta}(k) \right) P^{LBR} \tag{4.4}$$

where

$$\eta(k) = \begin{cases} 1 & , k \le g_i \\ 0 & , otherwise \end{cases}$$ and "|" is an OR operation

Since the system allows the customers to start playing back the video even if the given video quality is lower than what they requested, we should determine the probability that the customer can obtain all requested layers from the system (or the expected proportion of the requested layers that can be delivered to the clients). Because the expected proportion of clients that can be admitted to the system is $1 - P_{overall}^{Q}$, Q can be calculated by

$$Q = \frac{\sum_{i=1}^{M}\sum_{j=1}^{l_i} p_i r_j \left((b_{ij} \mid \eta(j)) + (\bar{b}_{ij} \times \bar{\eta}(j))(1-P^H) + (\bar{b}_{ij} \times \bar{\eta}(j)) \left(\sum_{k=1}^{j} \frac{b_{ik} \mid \eta(k)}{j} P^H \right) \right)}{1 - P_{overall}^{Q}} \tag{4.5}$$

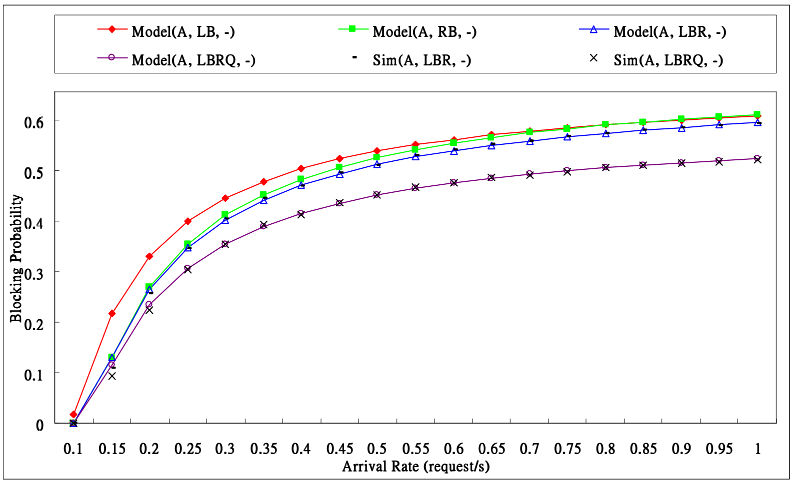
where "$\times$" is an AND operator.

Therefore, Q is equal to 1 if the requested layers can be obtained from the proxy server or broadcast channels (i.e. $(b_{ij} + \eta(j)\bar{b}_{ij})$) or can be streamed directly from the central server (i.e. $(\bar{b}_{ij} \times \bar{\eta}(j))(1-P^H)$). The third term indicates the proportion of clients that can still be satisfied by the system even if the resources cannot support the desired quality levels.

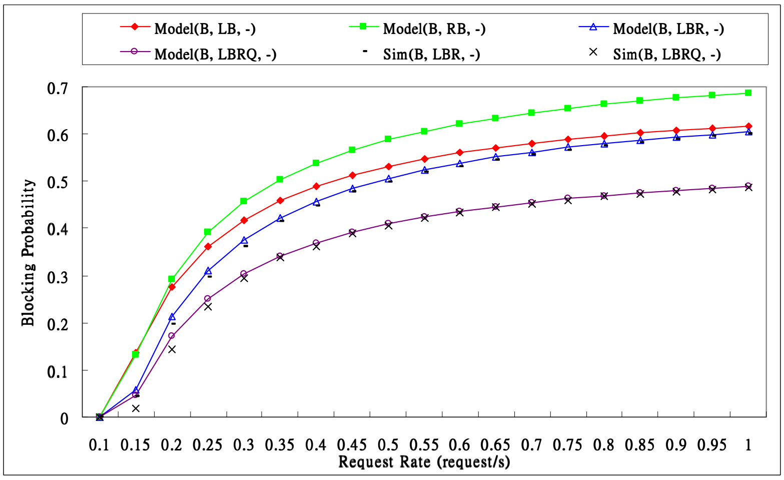
4.4 Experimental Results

In this section, we evaluate the performance of the proposed schemes in terms of blocking probability as well as quality perception. Computer simulation is also performed to verify the correctness of the analytical model. The parameters used for the evaluation are defined in Table 3.3. The two scenarios of requesting quality pattern stated in Table 3.4 will be used again to evaluate the performance of the system.

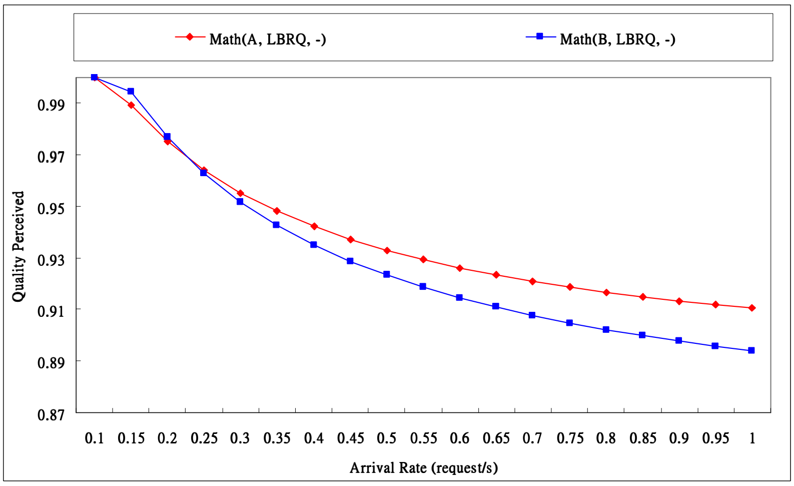
We first evaluate the performance impact of various arrival rates to the blocking probability. Figure 4.2 illustrates the performance of the proposed hybrid approach (LBR) and that with renegotiation (LBRQ) for various arrival rates. The figures show that our mathematical model is closely matched with the simulation results under various system configurations. As we expected, the blocking probability is increasing when the arrival rate is increased for various configurations. It can be seen that the systems with complementary approach can obtain further improvement compared to the sole approach (i.e. the system only uses layering with broadcasting (LB) or replication with broadcasting (RB) approach) for video transmission in both scenarios. As we know, LB performs worse than RB when the system environment is homogeneous, especially in low arrival rate. We can find from the result in Figure 4.2(a) that LBR can have a reduction of blocking probability up to 8% and can perform better than RB about 2% when the replication approach is taken the place of layering for end-to-end transmission in LB. On the other hand, it can be observed in Figure 4.2(b) that LBR can also obtain up to 9% further improvement when replication is deployed for unicast transmission in LB. However, it can be seen that the improvement is reducing when the arrival rate is further increased. It is because proxy caching is favor to high arrival rate and most of requests can be satisfied by the proxy server directly. In the figure, the performance of renegotiation is also presented. It can be observed that the system performance can be improved significantly when renegotiation mechanism is applied. We can see that the gain is increasing when the arrival rate is increased. It is because renegotiation increases the cache hit such that more requests can be satisfied by the proxy server. Although the renegotiation process can improve the system performance, it may reduce the customers' perception because this process cannot satisfy the customer's



(a) Blocking Probability (S-A)



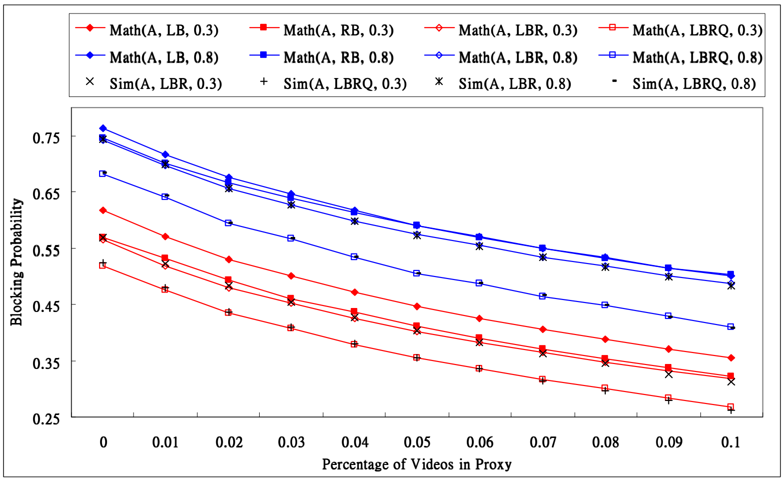
(b) Blocking Probability (S-B)



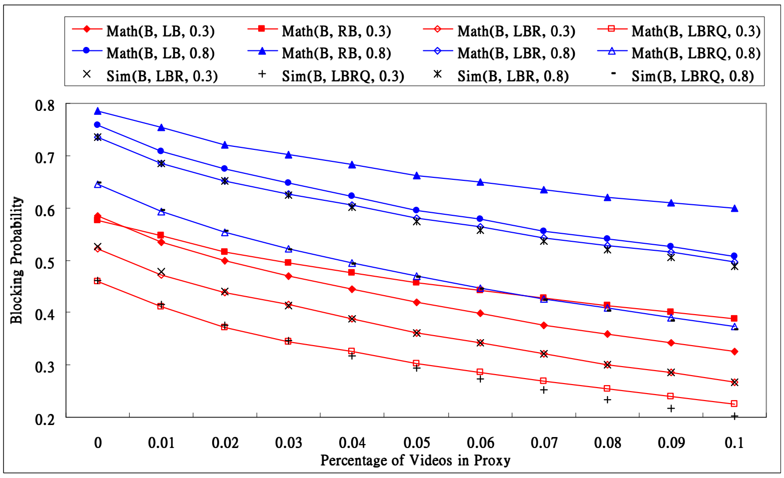
(c) Quality Perception

Figure 4.2 System Performance against Arrival Rate request completely when lacking of system resources. Therefore, we look at the probability that the customer can obtain all requested layers from the system (or the expected proportion of the requested layers that can be delivered to the customers). Figure 4.2(c) shows the quality perception against the arrival rate. We can observe that the quality perception is decreasing when the arrival rate is increased. When the arrival rate is increasing, λ_S^H is also increasing. Because the number of dedicated channels supported by the central server is limited, more requests should be accomplished by the renegotiation mechanism. Therefore, quality perception is reducing.

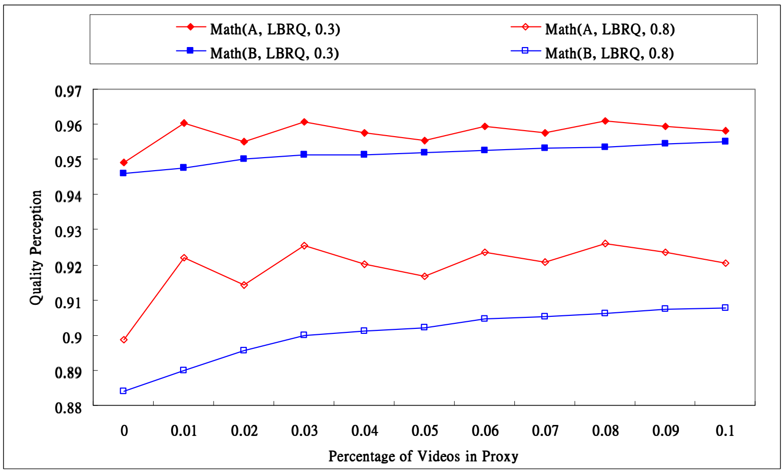
From the model, apart from the arrival rate, it can be observed that the proxy size (*K*), the efficiency of the broadcasting scheme (H^x) and the bandwidth reserved for broadcasting (B_{rsv}) bring a great impact on the system performance. In order to have a close look on the effectiveness of the broadcasting protocols to the system, Figures 4.3 and 4.4 show the blocking probability of the systems when these parameters are varied. We first investigate the impact of the proxy size when *K* is varied but B_{rsv} is fixed. Figure 4.3 illustrates the system blocking probability as the proxy size is changed. Increasing the proxy size results in fewer video requests passed to the central server and thus more customers can be served. Therefore, it can be found that all the blocking probabilities are decreasing along with the growth of the proxy size. In "S-A", it can be more clearly seen that LBR can improve the drawback of LB in homogeneous environment. In "S-B", LBR



(a) Blocking Probability (S-A)



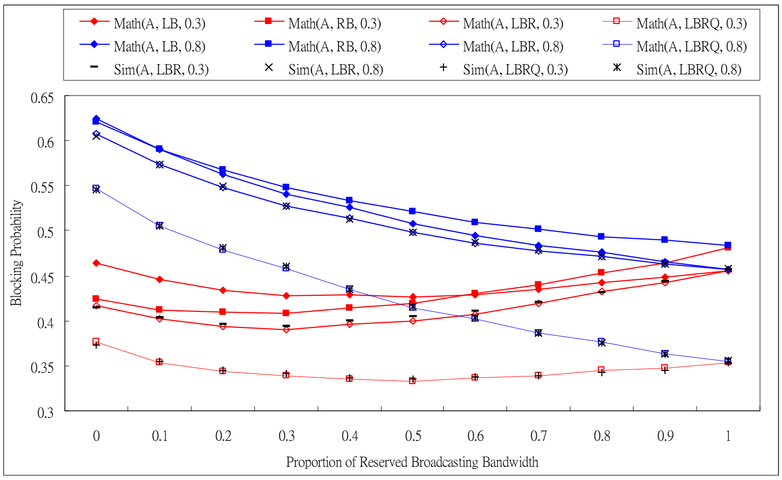
(b) Blocking Probability (S-B)



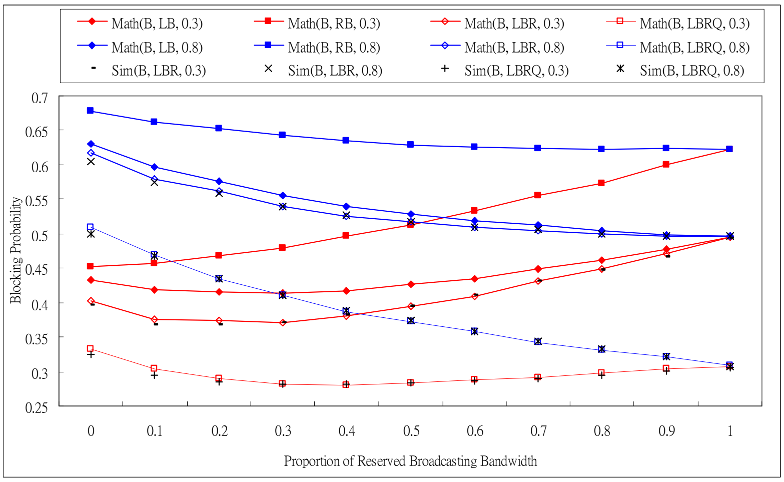
(c) Quality Perception

Figure 4.3 System Performance against Proxy Size can also further enhance the system performance. It is because LBR is benefited from both layering and replication for proxy caching and end-to-end transmission respectively. In addition, the system performance can have a further improvement up to 10% when LBRQ is used. The system performance in terms of the video quality is depicted in Figure 4.3(c). It can be observed that the video quality is improving when the proxy size is increased. In various proxy sizes, we can see that the system can provide more than 80% quality perception to the customer. Therefore, the customers are still able to obtain the quality of video close to their expectation under this approach.

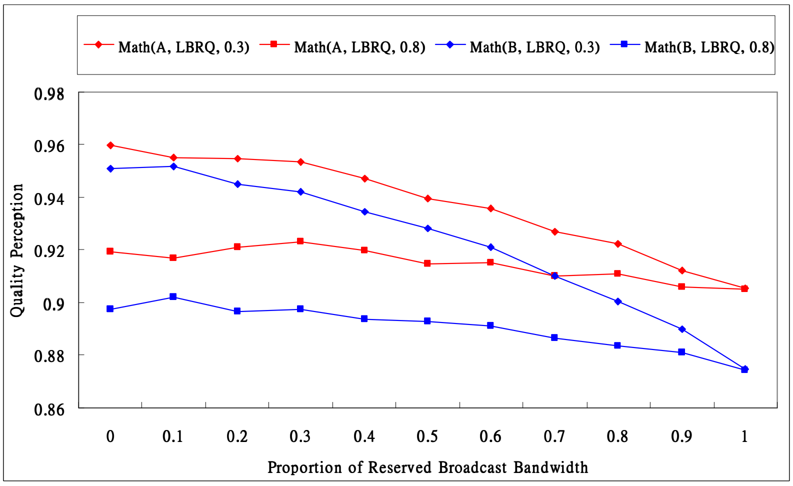
Now we plot the bandwidth reserved for broadcasting. The proportion of broadcasting bandwidth is changed from 0 to 1 as shown in Figure 4.4. If p_{rsv} is 0, there is no bandwidth reserved for broadcasting. On the other hand, all the available bandwidth will be used for broadcasting the videos if p_{rsv} is 1. It can be seen that the systems are getting worse with the increase of bandwidth reserved for broadcasting at low traffic. It is expected because broadcasting does not favor to the systems with low arrival rate. At high traffic, the system blocking is strictly decreasing. However, when all bandwidth is reserved for broadcasting, the performance of the systems is convergence. It is because only a portion of videos in the systems can be served to the customers and some less popular videos will never be served. The results show that the performance of LBR still performs better than LB and RB but the performance of LBR is getting close to the performance of



(a) Blocking Probability (S-A)



(b) Blocking Probability (S-B)



(c) Quality Perception

Figure 4.4 System Performance against Broadcast Efficiency

LB in various system configurations. One explanation for this is that the system bandwidth for end-to-end transmission is shrinking and thus the operation of LBR and LB becomes identical. Figure 4.4(c) illustrates the average video quality to serve the admitted customers when the proportion of reserved bandwidth for broadcasting is increased. It can be observed that increases in reserved bandwidth will result in decreases in quality perception of customers. When the reserved bandwidth is increased, the number of dedicated channels provided by the central server is reducing and hence the flexibility of the system will be reduced. As a result, the system only provides a restrictive quality level of videos to customers from the proxy cache and broadcast channels. However, the system can still deliver the video streams to provide customers more than 87% of video quality that they request.

4.5 Summary

By using the findings in Chapter 3, we investigate a complementary approach using both video replication and layering for video streaming in this chapter. We then analytically explore the benefit of renegotiation about video quality when the system resources cannot support the requested quality levels. From the results, we have found that the system performance can obtain an improvement up to 15% when complementary coding schemes (i.e. layering approach is used for proxy caching and video broadcasting while replication is used for end-to-end transmission) for video transmission and renegotiation mechanism are used.

Chapter 5

Peer-to-Peer Batching Policy for Video-on-Demand

5.1 Introduction

In Chapter 3 ad Chapter 4, we have investigated possible solutions for building a large-scale VoD system in a heterogeneous network environment using both the broadcasting technique and hierarchical architecture. We also compared video streaming of replication with that of layering in the proposed framework. In such architecture, we deploy broadcast/multicast to improve the scalability of the system. In general, any efficient protocols can be applied to the system framework. Among existing broadcast protocols, Staggered is the earliest broadcasting scheme proposed in [DAN94] that we can be applied. As it introduces very long start-up latency, some efficient broadcasting protocols [VISWA96, JUHN97] were then proposed to reduce the start-up delay and at the same time minimize the bandwidth requirement. Nevertheless, these broadcasting protocols such as Harmonic [JUHN97] are impractical to support insensitive (less-than-minutes) start-up delay services since the system needs to manage a large number of concurrent channels for a single video. Moreover, the requirement on client bandwidth and buffer is strict high, making the overall system very expensive [HUA94]. Different from the broadcasting system, in a multicast VoD system, customers arriving closely enough can be grouped together and served by a single multicasting channel. Time between the consecutive multicasting channels is known as batching time/start-up delay. To implement a true (zero-delay) VoD system in such a multicast environment, a video server needs to allocate unicast channels for late-arriving customers until they can catch up the ongoing multicast video session. Patching [HUA98, CAI99, GAO99] is one of the representative protocols based on this idea. Although these schemes can improve the scalability of the system, this approach still inherits from the client-sever model such that the bottleneck of the system is still at the server side. Recently, researchers have turned their focus to another approach to address the issues of system scalability – peer-to-peer (P2P). In such P2P architecture, each end-point called peer is operated as client and server simultaneously. It retrieves what it requests from the system and forwards/relays it to the system as well in such a way that an application layer multicast is constructed [SHEU97, XU02, GUO03a, GUO03b, HEFEE03, NICOLO03, TRAN04,]. To watch a video, a peer establishes one/several dedicated channels to other peers in the system. The number of channels established for one video session is therefore proportionally increased with the arrival rate.

As mentioned, the broadcasting schemes may not be feasible to support the zero-delay VoD services. Instead, it is believed that both multicasting and P2P are the two promising alternatives. It is observed that a classic P2P mechanism requires small server resources as well as provides an insensitive delay. However, the bandwidth requirement inside the network will be rapidly increased when more peers joins the video session. On the other hand, if the system simply uses a multicast scheme to deliver a video, the peers will wait for a noticeable delay before watching the video whereas the overall bandwidth requirement will not be significantly increased. In this chapter, we consider the trade-off between the network bandwidth requirement for P2P transmission and multicast delivery. We exploit the multicast capability of the network and P2P paradigm to efficiently deliver video data to the clients. A Peer-to-Peer Batching (PPB) policy is proposed to minimize the system requirement. Unlike traditional multicast approaches that use extra server resources to meet the true VoD requirement, in PPB, the central server shifts this duty to peers for providing the initial portion of the video to the subsequent peers by means of P2P transmission during the video session. If the size of the P2P network is large enough, any subsequent peer will be served by a new video session. Hence, the size of the P2P network using PPB will not proportionally grow with the arrival rate. One of the objectives of this chapter is to determine the optimal size of the P2P network such that the overall transmission cost is minimized. In addition, because peers can leave the system at any time without notice and the departure or failure of any peer may corrupt the service, fault tolerance and recovery procedures are developed to take the peers departure behavior into account for better management of the system resources.

This chapter is organized as follows. We first describe our proposed PPB policy in Section 5.2. We then develop an admission blocking probability model for PPB in Section 5.3. The fault recovery mechanism of PPB will be presented in Section 5.4. In Section 5.5, analytical results will be presented to demonstrate the efficacy of the proposed system. In addition, simulations are also performed to verify the correctness of the analytical model. Finally, some concluding remarks are given in Section 5.6.

5.2 Peer-to-Peer Batching (PPB) Policy

In this section, we describe PPB and then derive an analytical model to evaluate the system performance of PPB. In our proposed architecture, it is first assumed that a generic infrastructure of network is multicast-enabled. All peers in the system have an identical upstream and downstream bandwidth. In addition, the downstream bandwidth is at least double of the playback rate. It is further assumed that the caching buffer is enough to store a small portion of the video and data sharing can only be accomplished within the same video session. All the peers are assumed to leave the system when the playback has been completed. To facilitate our discussion, we define the notations in Table 5.1.

Symbol	Meanings
λ	System arrival rate (reqs/s)
μ	System departure rate (departure/s)
R	Data rate of the video (bit/s)
W	Length of the batching time (s)
L	Length of the video (s)
B	Buffer size of peer (s)
M	Number of videos in the system
p_i	Popularity of video i
λ_E	The effective arrival rate of the system
μ_E	The expected service rate of the system
c_s	Transmission cost of the central server for one video channel
c_p	Transmission cost of the P2P network for one video channel
S_S, S_S^F	Number of multicast channel demand for the central server without and with fault exception
S_P, S_P^F	Number of PPC demand for the system without and with fault exception
S_S^R	Number of unicast channel demand for the central server during fault exception
C_S, C_S^F	Transmission cost demanded for the central server without and with fault exception
C_P, C_P^F	Transmission cost demanded for the P2P network without and with fault exception
C_r^F	Transmission cost demanded for the system for fault recovery
W_{opt}, W_{opt}^F	Optimal batching time of the system without and with fault exception

Table 5.1. Notations of Symbol

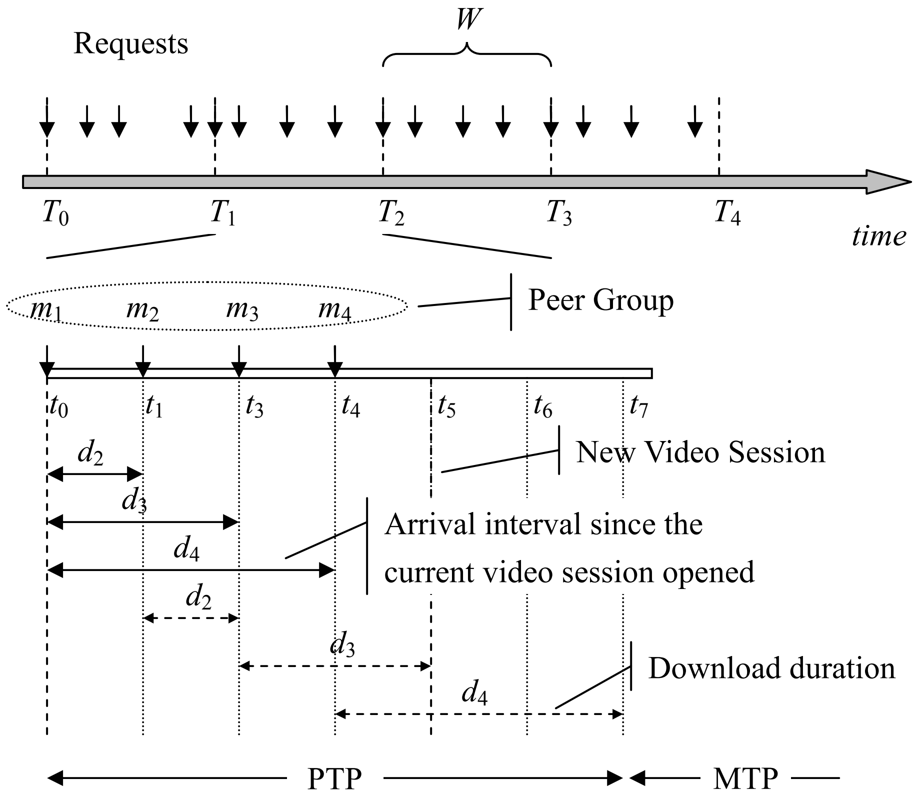


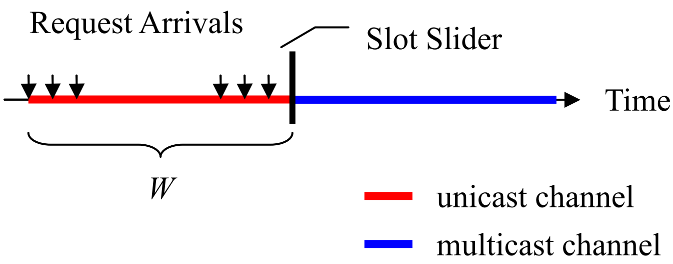
Figure 5.1. The Scheme of the Proposed PPB Policy

In PPB, each peer needs to go through two phases to retrieve the whole copy of the video for playback, namely, P2P transmission phase (PTP) and multicast transmission phase (MTP). PTP enables the peers to start watching the video without a long waiting time while MTP is to reduce the network bandwidth requirement when more peers join the system. In this policy, a new video session for one particular video is created every W seconds called batching time and peers arriving within W for video session i form a peer group (G_i). As shown in Figure 5.1, in PTP, any peer arriving after the beginning of the video session (denoted patch-caller) is served by a unicast channel called peer-to-peer channel (PPC) opened from a previously arrived peer (denoted patch-callee) to retrieve the initial portion of the video (denoted video patch). The duration of the video patch to be retrieved (d_i) can be computed as:

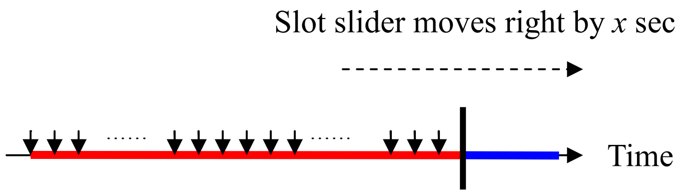
$$d_i = t_i \mod W \tag{5.1}$$

where t_i is the arrival time of peer i (m_i). Therefore, the peers in the same video session are interconnected by PPCs and a P2P network is formed. While concurrently downloading the video patch from the P2P network, any subsequent patch-caller is also required to join the multicast channel to retrieve the current video content. After d_i seconds, m_i has completed downloading the video patch and PPC for m_i is in synchronization with the multicast channel. This PPC is thus released. If all PPCs are released, the P2P network for the peer group is dismissed and this peer group goes into the MTP phase. During this phase, no new peer can join this peer group and all members in this group simply retrieve the video data from the multicast channel until the end of the video session. Therefore, the time to switch from PTP into MTP during a video session is governed by W. Obviously, this value is critical on affecting the system performance in terms of overall bandwidth requirement or transmission cost.

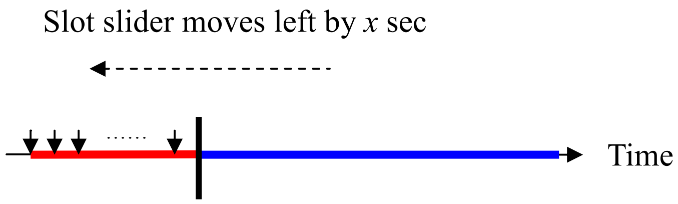
Now, we look at how W affects the system performance in terms of bandwidth requirement. As shown in Figure 5.2(b), when the slot slider moves to the right by x seconds, the size of the P2P network is expanded to $\lambda(W+x)-1$. However, the number of multicast channels is reduced to $\frac{L}{W+x}$. Interestingly, the situation is opposite when



(a)



(b)



(c)

Figure 5.2. The Effect of Slot Width W

the slot slider moves to the left by x seconds as shown in Figure 5.2(c). In this case, the group size is reduced to $\lambda(W-1)-1$ and the number of multicast channels is increased to $\frac{L}{W-x}$. Thus, the change of W increases the bandwidth demanded for one transmission phase but decreases another. This behaviour is defined as *sliding effect*. There is obviously a tradeoff between the number of channels required by the P2P transmission phase and multicast delivery phase. Our objective is therefore to determine the optimal value of W (W_{opt}) such that the overall bandwidth requirement is minimized.

As mentioned, a new video session for one particular video is created every W seconds. To calculate the average number of PPCs for the system, it is assumed that all P2P groups are statistically identical. In addition, the peer arrival process is assumed as a Poisson process with rate λ. The mean number of peers (α) arriving within W can then be approximately given by

$$\alpha = \max(\lfloor \lambda \times W \rfloor, \ 1) \tag{5.2}$$

Assume that the time between two consecutive arrivals is x seconds. Since the i^{th} peer of this peer group retrieves ix seconds of the video patch from its patch-callee, the total number of bits of the video patch demanded for one peer group (D_I) during PTP can be calculated by the following equation

$$D_I = \sum_{i=1}^{\alpha-1} ixR = \frac{\alpha(\alpha-1)xR}{2} \tag{5.3}$$

where R is the data rate of the video. Under the assumption of Poisson arrival process, the probability density function of x is $f(x)=\lambda e^{-\lambda x}$. The expected number of PPCs (S_P) is thus given by

$$S_P = \int_0^{W-1} \frac{D_I}{RW} f(x)dx \approx \frac{\alpha - 1}{2} \tag{5.4}$$

Then, the average number of concurrent video channels allocated by the central server is thus given by

$$S_S = \frac{L}{W}, \tag{5.5}$$

where L is the length of the video. Denote that the transmission cost of the central server and that of the P2P network are equal to $C_S = S_S$ and $C_P = cS_P$ respectively, where c is ratio of the transmission cost of PPCs to that of the video channels from the central server. Then, the total system transmission cost is $C_T = C_P + C_S$. If λ and L are fixed, to find W_{opt}, we define

$$g(W) = S_S + cS_P \tag{5.6}$$

By setting the derivative of the *g(W)* with respect to W to zero, W_{opt} is then obtained as

$$W_{opt} = \sqrt{\frac{2L}{c\lambda}} \tag{5.7}$$

As the last-coming peer in the peer group misses at most ($W_{opt} - \lambda^{-1}$) seconds' leading portion of the video, the minimum buffer size (B) of the peers is thus equal to $B = W_{opt} - \lambda^{-1}$.

5.3 Admission Blocking Probability

Apart from calculating the average resource requirement of the system, the system blocking probability is another important factor to the design of a VoD system. Similar to other batching-based VoD systems [AGGAR96a, DAN96, POON00, TANG04], our proposed policy also serves a group of customers only when the resource is available. Therefore, customers are blocked when there is no multicast channel available for video delivery.

Assume that the system provides M videos for customers. Denote p_i and L_i be the popularity and the length of video i respectively. Because the arrival process is modeled as a Poisson process with rate λ, the effective request rate of video i (λ_i) can be determined as

$$\lambda_i = \sqrt{\frac{c\lambda p_i}{2L_i}} \qquad (5.8)$$

Thus, the effective request rate for the system (λ_E) can be computed by

$$\lambda_E = \sum_{i=1}^{M} \lambda_i \qquad (5.9)$$

Then, the expected service rate of the system (μ_E) is given by

$$\mu_E = \left(\sum_{i=1}^{M} \frac{\lambda_i L_i}{\lambda_E} \right)^{-1} \qquad (5.10)$$

From the viewpoint of queuing theory [PRABH97], the system can be considered that there are K video channels supported by the central server simultaneously and the service time is followed a general distribution with rate μ_E. Since the arrival rate of the system for batching is λ_E, the system can be modeled as M/G/K/K queuing system and the blocking probability (P^B) is equal to

$$P^B = \frac{(\lambda_E / \mu_E)^K / K!}{\sum_{i=0}^{K} (\lambda_E / \mu_E)^j / i!} \qquad (5.11)$$

5.4 Fault Tolerance and Recovery Mechanism

In Section 5.2, it is simply assumed that the peers leave the system when the playback has been completed. However, similar to any P2P applications, the peers may leave from the system at any time without notice and the departure/failure of any peers may affect other

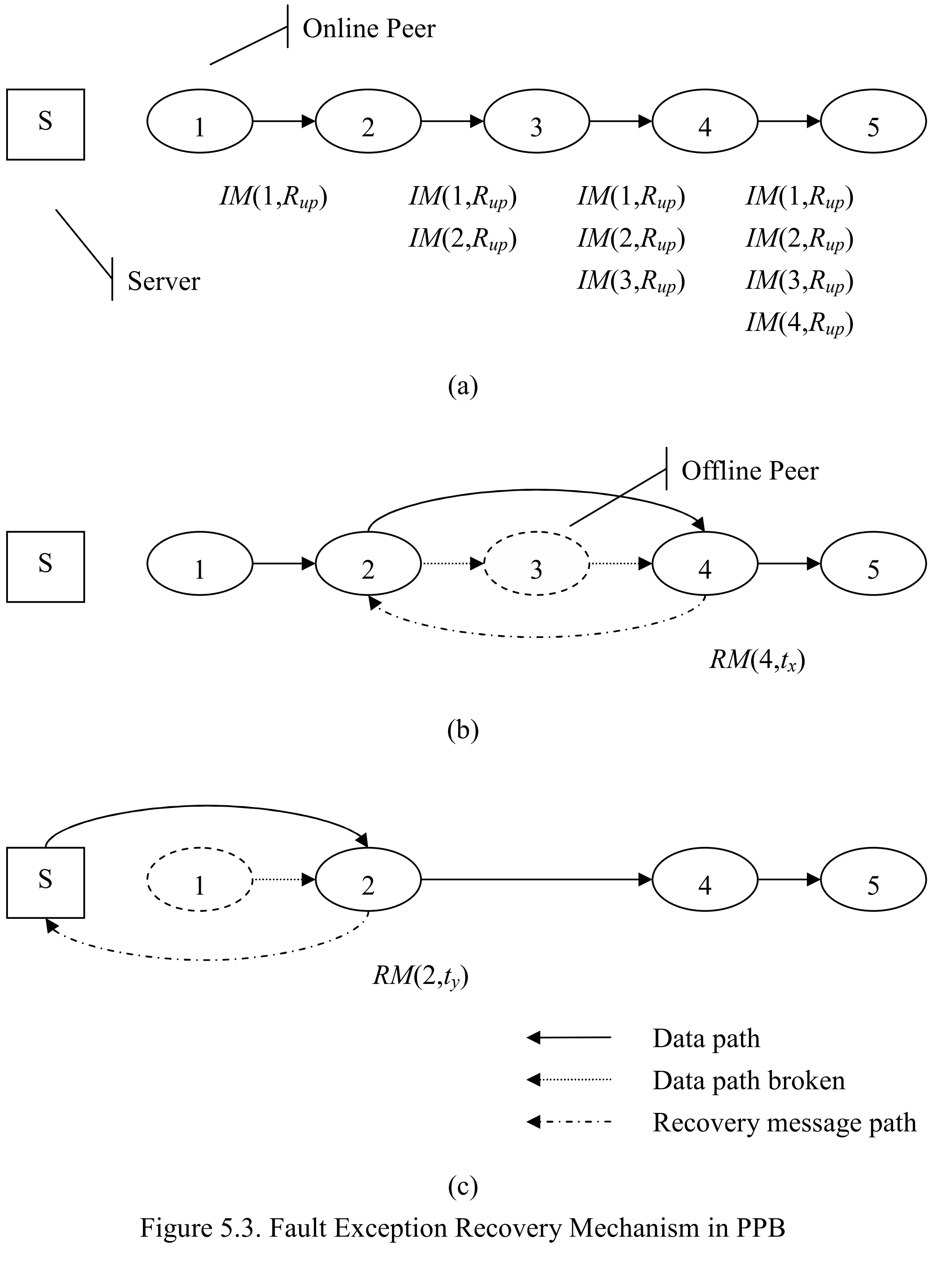


Figure 5.3. Fault Exception Recovery Mechanism in PPB

peers in the system. We thus investigate such early departure behavior of the peers and propose a fault tolerance and recovery mechanism in our proposed framework.

In PPB, the recovery procedure is handled by other peers in the same video session (i.e. a peer is never redirected to other video sessions to recover the service). The reason is that, under our assumption, data sharing can only be accomplished within the same video session. From the results in Section 5.5, it can be found that this limitation only introduces insignificantly additional server resources but it can increase the efficiency of the recovery process. As shown in Figure 5.3(a), during PTP, the peer should send information message (IM) including its identifier along the video chain periodically such that all late-coming peers have their ascendants' information in the same video session. It can be seen from the figure that peer 1 (m_1) sends an information message IM(1) to m_2. Then, m_2 relays this message to m_3 and so on. The time duration to send IM can be self-clocked or triggered by the arrival of IM from the other peers. When a peer receives an IM, it updates its IM table which is sorted according to the descending order of peers' identifier. Based on this information, the peer can therefore determine which ascendant should connect to without the aid of the central server when a fault is detected.

To detect a normal fault exception, a peer should send a "leave" message with its identifier to its descendants along the video chain when it is going to offline. Then, its identifier will be removed from other peers' IM table. However, to handle the unexpected departure/failure, each identifier in IM table is also associated with a count down timer, which is reset whenever the corresponding peer's identifier is arrived. Thus, if the timer is expired, the corresponding peer is considered as failure and its identifier will also be removed from the IM table. Whenever the peer finds that its ascendant's identifier (i.e.

patch-callee) is removed from the IM table due to normal or unexpected failure, it will launch the fault recovery process. If there is no ascendant to recover the fault, the peer sends a recovery message (RM) to the central server which then opens a recovery channel immediately. RM is composed of the peer's identifier as well as the resumption point of the video. Formally speaking, when m_i departs, the video chain is split such that m_{i+1} and its descendant(s) cannot retrieve the video data from the system. To continue the service, m_{i+1} should look for m_r ($r \neq i+1$) which should be with the highest identifier from the IM table. If the desired m_r is found, m_{i+1} sends RM to m_r and the recovery process is successful (see Figure 5.3(b)). Otherwise, the central server has to create a recovery channel for m_{i+1} (see Figure 5.3(c)).

To take the early departure behavior of the peers into account in the performance model, the central server may not need to deliver the whole video to one peer group. Therefore, we first determine the time duration between the start and the end of the video session in one peer group. For simplicity, we assume that the transition time for recovery is very short and can be neglected. In the proposed policy, a peer retrieves the video patch and current portion of video content from the unicast channel and the multicast channel respectively at the same time. Since the video patch is delivered to the late-coming peers either from the P2P network or from the central server, the peer can admit into the system without blocking. From the viewpoint of queuing theory, the system can be considered that there are infinite serving PPC for peers and the mean time to departure (*MTD*) follows an exponential distribution with rate μ. As the arrival process is Poisson, the system can be viewed as an $M/M/\infty$ queuing system. Since the peers arriving after *W* are served by another video session, the lifetime of one video session depends on the number of active peers in such session. Thus, we investigate the transient behavior of the system and, from [PARLA00], the transient probability of $M/M/\infty$ can be expressed by

$$p_n(t) = \frac{1}{n!}\left[\frac{\lambda}{\mu}(1-e^{-\mu t})\right]^n \cdot P_0(t), \tag{5.12}$$

where $p_0(t) = e^{-\frac{\lambda}{\mu}(1-e^{-\mu t})}$. The expected number of peers (*K*) that is still in the system at *t* is then given by

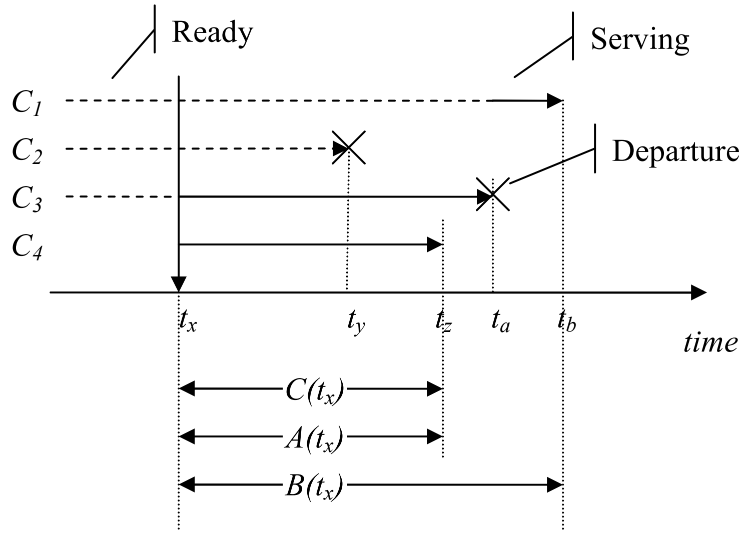
$$K(t) = \min\left(\left\lfloor\frac{\lambda}{\mu}(1-e^{-\mu t})\right\rfloor, 1\right) \tag{5.13}$$

With *K(t)*, we can compute the average serving time of the central server for each video session. The time starting from *W* until all the peers leave is called the peer preserving time (T_P) which can be modeled as a parallel system of *K(W)* independent exponential distributed components with rate μ. Hence, we have

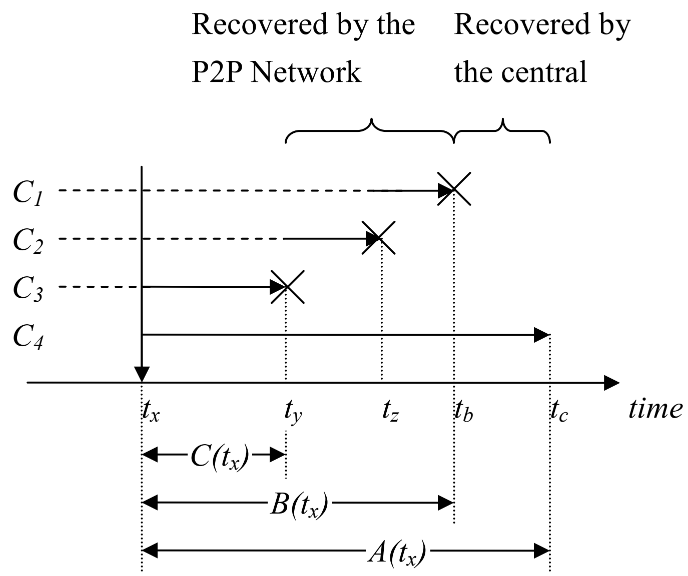
$$T_P = \int_0^{L-W} (1-e^{-\mu t})^{K(W)} dt \tag{5.14}$$

The expected time duration for serving each video session by the central server is $T_P + W$ and thus the expected number of server channels to be used for multicasting is

$$S_S^F = \frac{T_P + W}{W} \tag{5.15}$$



(a)



(b)

Figure 5.4. Illustration of Recovery Mechanism in PPB

We now calculate the bandwidth required for the P2P network and the recovery process. As the expected inter-arrival time between two consecutive arrivals is λ^{-1} seconds, m_i needs to download the video patch for $i\lambda^{-1}$ seconds. Therefore, the total number of bits (D_I^F) transmitted over the P2P network for one peer group can be calculated by

$$D_I^F = \sum_{i=1}^{\alpha-1} R \cdot A(i/\lambda) \tag{5.16}$$

where $A(t) = \int_0^t e^{-\mu x} dx$, which is the expected duration of each peer to download the video patch.

As mentioned before, the data sharing involves two peers (i.e. one is patch-callee and one is patch-caller). Obviously, this retrieval process between patch-callee and patch-caller forms a series system and either of them who leaves early will terminate the normal download process and thus the recovery procedure should be started. Therefore, for one video session, the total expected number of bits downloaded successfully of each patch-caller from its first patch-callee (D_C^F) can be determined by

$$D_C^F = \sum_{i=1}^{\alpha-1} (1 - p_0(i/\lambda)) C(t), \tag{5.17}$$

where $C(t) = R\int_0^t e^{-2\mu x} dx$, which is the expected duration that the patch-caller can obtain the video patch from its first patch-callee. $1 - p_0(t)$ is the probability that at least one peer is ready to serve in the peer group at time *t*. Hence, the expected number of bits required for fault recovery (D_F^R) is $D_F^R = D_I^F - D_C^F$. Actually, D_F^R is contributed by the central server and the P2P network. When a new peer enters the system, the number of peers in the current video session determines how long this peer can retrieve the video patch from the P2P network. If this time is longer than its expected duration of retrieving the video patch, there is no recovery channel allocated by the central server. Otherwise, the central server has to open a recovery channel to deliver the rest of the video patch to this peer. As shown in Figure 5.4(a), C_4 arrives at t_x and three peers in the system can become the patch-callee of C_4. Let $B(t)$ be the expected serving time of these patch-callees which can be expressed as $B(t) = \int_0^t (1-e^{-\mu x})^{K(t)} dx$. If C_4 can completely download the video patch or leave the system before $B(t)$, no recovery channel will be opened from the central server. Otherwise, as shown in Figure 5.4(b), if C_1, C_2 and C_3 depart from the system, the central server should allocate a video channel for C_4 until it leaves or completely receives the video patch. Therefore, the expected number of bits delivered from the central server for fault recovery (D_S^R) can be expressed by:

$$D_S^R = \sum_{i=1}^{\alpha-1} \left(\left(p_0(i/\lambda) A(i/\lambda) + (1-p_0(i/\lambda))\eta(i/\lambda) \right) \cdot R \right), \tag{5.18}$$

where $\eta(t) = \begin{cases} 0 & , B(t) \ge A(t) \\ A(t) - B(t) & , otherwise \end{cases}$.

The first term in eqn.(5.18) is the expected number of bits transmitted from the central server if there is no peer ready to serve at time t. The second term is the expected number of bits transmitted from the central server when all the patch-callees depart from the system. Then, the expected number of bits delivered from the P2P network (D_P^F) can be given by $D_P^F = D_I^F - D_S^R$.

Parameter	Nominal Value	Range of Value
λ	0.1 req/s	0.05-0.15 req/s, 1 req/s
μ^{-1}	1200 seconds	300-3600 seconds
L	7200 seconds	-
c	0.5	0.1-1
B	10% of L	5%-50% of L
X	1	-
M	50	-
K	-	50, 100, 150

Table 5.2. Summary of the System Parameters

The expected number of server channels (S_S^R) and PPC (S_P^F) required for the system during PTP can thus be determined by $S_S^R = D_S^R / WR$ and $S_P^F = D_P^F / WR$ respectively. Then, the transmission cost of the system for recovery (C_r^F) can be computed by $C_r^F = S_S^R + c(S_P^F - D_C^F / WR)$. Finally, the transmission cost of the central server and that of the P2P network is equal to $C_S^F = S_S^F + S_S^R$ and $C_P^F = cS_P^F$. Similar to PPB, the system performance highly depends on *W*. In order to determine the optimal value of *W* (W_{opt}^F), the objective function can be formally stated as to minimize $C_S^F + C_P^F$ subject to $0 < W \leq L$. As the range of *W* is bounded, this optimization problem can be solved numerically.

5.5 Experimental Results

In this section, we present the performance of our proposed policies in terms of transmission cost. Computer simulations are performed to verify the correctness of the analytical models. We assume that all videos have the identical length of 7200 seconds (2 hours). Unless otherwise specified, $c = 0.5$ [WANG04] and R=1. To illustrate the performance of our system, we compare the proposed policy with DirectStream [GUO03a] and PSM [YANG05]. The parameters used for evaluation are summarized in Table 4.2. In the following results, each curve is represented by "Model[4]:($a_{b,c}$, p)" where $a_{b,c}$ is the output of the curve which is equivalent to a_c^b in our notations and p is the parameter for the curve. For example, $C_{f,s}$ is used to represent C_S^F in the graph.

5.5.1 Performance of the PPB Policy without Fault Exception

We first investigate the transmission cost against W in Figure 5.5 when the arrival rate is fixed as 0.1 req/s for a single video. It is found that the mathematical model closely matches with the simulation results. As expected, C_S is decreasing when W is increased. However, C_P is increasing because the size of the P2P network is increasing with W. As mentioned, our goal is to determine W_{opt} in order to minimize the total transmission cost of the system. From Figure 5.5, it is observed that W_{opt} is about 550 seconds and the corresponding total transmission cost is about 25.

Figure 5.6 shows W_{opt} and its corresponding transmission cost with various arrival rates. As we expected, the total transmission cost is increasing when the arrival rate is increased. But, it can be found that W_{opt} is decreasing. When the arrival rate is high, the

[4] Math – results obtained from analytical model, Sim – results obtained from simulation

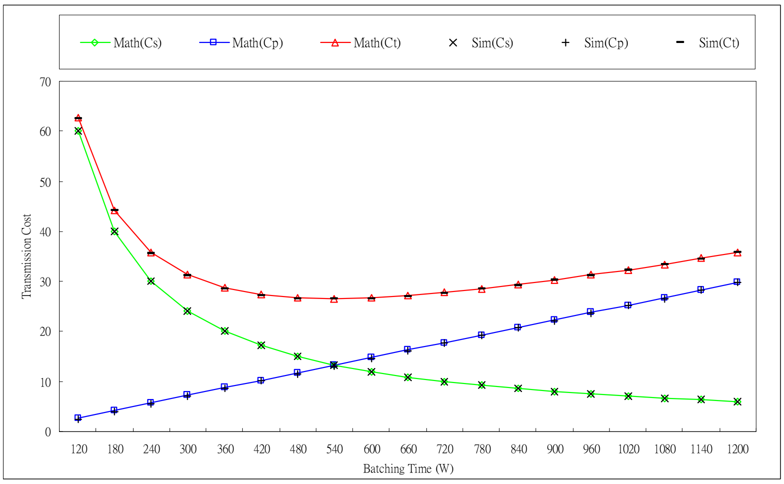


Figure 5.5. Transmission Cost against Various Batching Time W

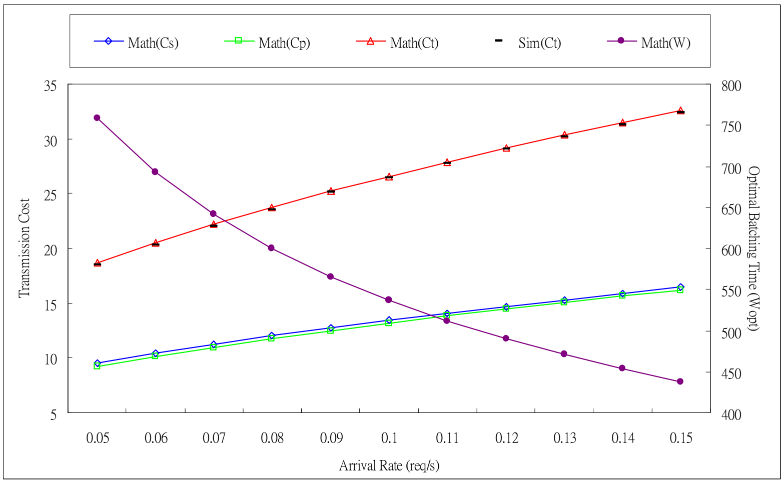


Figure 5.6. Optimal Batching Time W_{opt} and Its Corresponding Transmission Cost Requirement against Arrival Rates

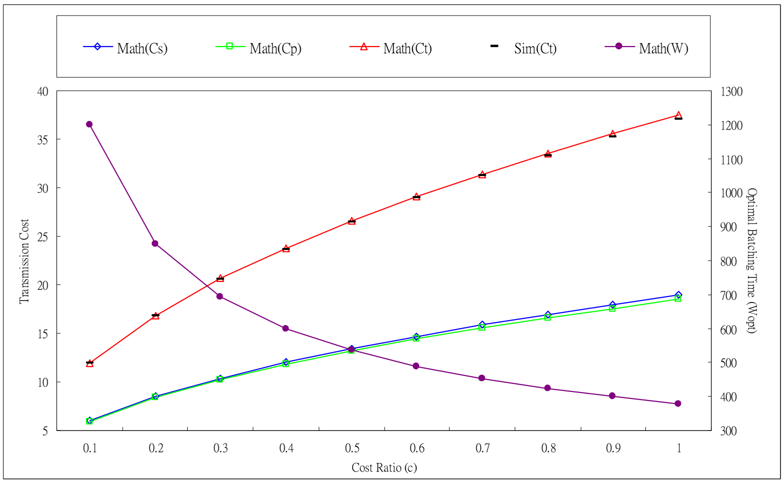
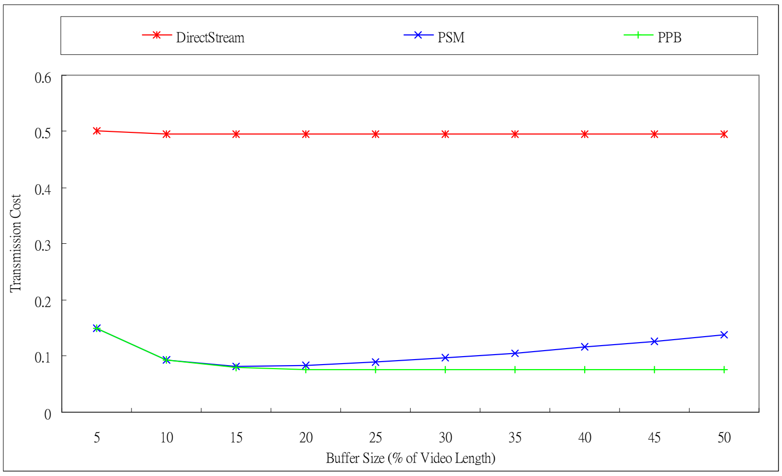


Figure 5.7. Optimal Batching Time W_{opt} and Its Corresponding Transmission Cost Requirement against Cost Ratios

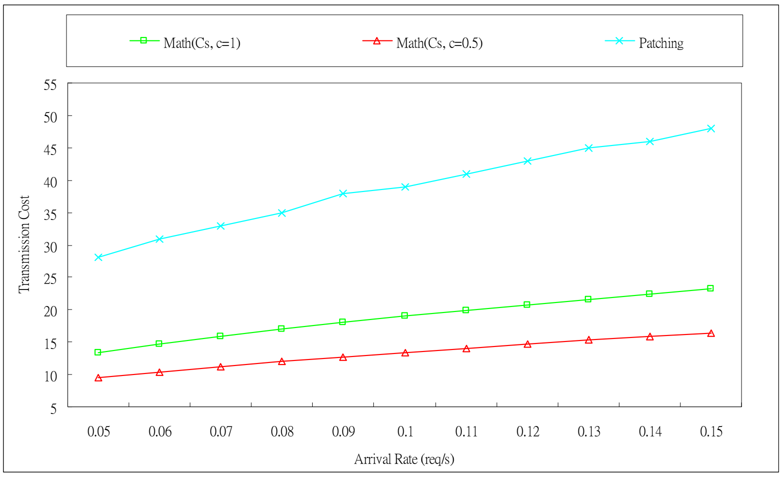
size of the P2P network is reduced but more multicast channels are allocated. It is intuitive that multicast is suitable for high arrival rate while unicast is favor to low arrival rate. Our system adjusts the slot slider (i.e. W) to make a trade-off between the use of multicast delivery and unicast transmission for various arrival rates to minimize the total transmission cost. On the other hand, it can be found that the transmission cost for the central server and the P2P network are nearly the same when the system operates in the optimal state.

Figure 5.7 shows the transmission cost and W_{opt} when the cost ratio is changed. It can be seen from the results that, when the cost ratio is increased, W_{opt} is decreased but the system requires higher transmission cost. The reason is that when the cost ratio is low, the transmission cost for the server channels is high so the system enlarges the size of the P2P network (i.e. increase W) to reduce the number of multicast channels. It is worth noting that the central server and the P2P network also roughly share the same amount of transmission cost from the total transmission cost when the system operates in the optimal state.

We then compare the performance of PPB with DirectStream and PSM. We assume that the system supports 50 videos with the identical length and the arrival rate of the system is 1 req/s. The video popularity follows the Zipf's distribution with a skew factor of 0.271 [ZIPF49]. Figure 5.8(a) shows the total transmission cost required for PPB,



(a) Comparison of PPB, DirectStream and PSM in various buffer sizes



(b) Comparison of PPB and Patching

Figure 5.8 Comparisons of Various Schemes with PPB

DirectStream and PSM with different buffer size. The transmission cost is normalized by λL (i.e. the total transmission cost of the system where each request is served by the central server with unicast stream directly). It is found that both PPB and PSM require less transmission cost than DirectStream because of the use of multicast transmission solution. From the results, the performance of PPB and PSM is similar when the buffer size is less than 15% of L. However, when it is over 15%, PPB performs better than PSM.

When the buffer size is small, both schemes fully utilize the buffer space. Nevertheless, PSM and PPB manage the buffer space in different ways when the buffer size is increased. In PSM, the system will use the buffer space as much as possible to reduce the central server resources. On the other hand, in PPB, min(W_{opt}, B) is occupied to minimize the overall system transmission cost. Therefore, when the buffer size is increased, PSM will obtain less benefit from multicasting because of the unrestricted growth of the P2P network (refer to Figure 5.5). In fact, we will see in next section that the bound buffer management will be more efficient for fault recovery. We also compare the performance of PPB with Patching [HUA98] (a native multicast approach). The result of Patching is computed from the model in [CAI99]. Figure 5.8(b) shows the expected transmission cost required for the central server in PPB and Patching with different arrival rates. It is observed that the transmission cost required for the central server in PPB is significantly lower than Patching in various arrival rates. The server resource requirement of PPB is less than Patching by half when the arrival rate is of 0.1 req/s even the cost ratio of the system is equal to 1. It is because each peer in PPB contributes its resources to the system to share the workload of the server. It can be found that PPB can have further reduction of transmission cost up to 30% when the cost ratio is 0.5. We would like to point out that PPB can serve more video to clients than Patching if their hardware settings are the same on server.

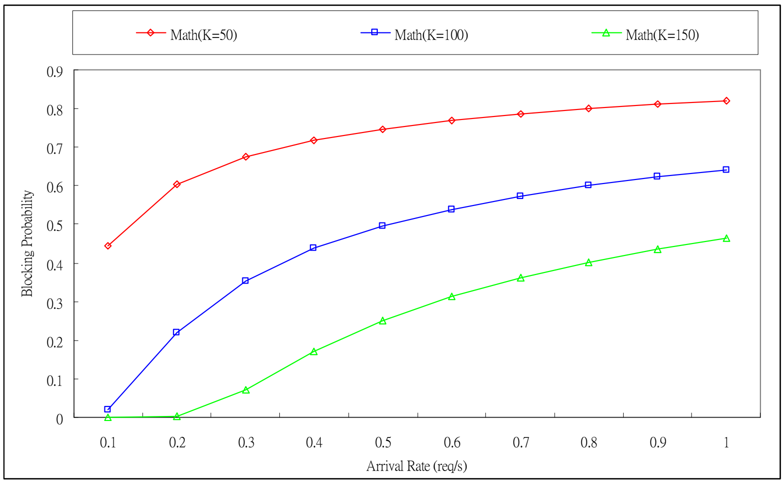


Figure 5.9. Blocking probability against arrival rate

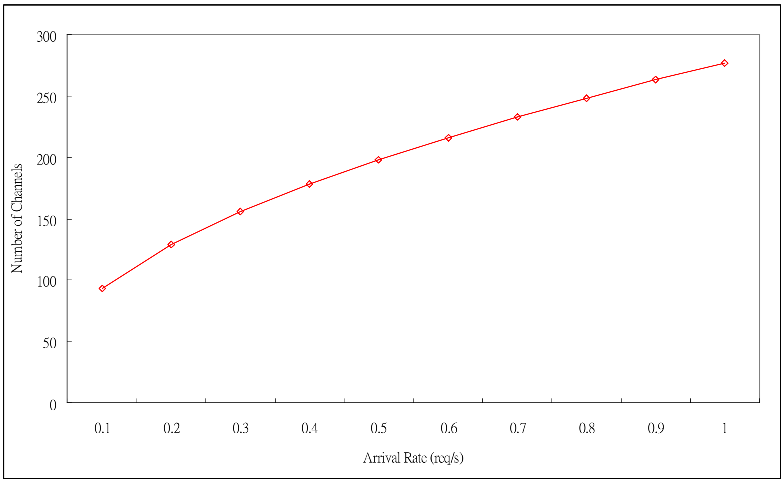


Figure 5.10. Channels required to achieve less than 5% blocking probability against arrival rate

5.5.2 System Blocking Probability

Now, we are going to investigate the blocking probability of the system in various system configurations. Again, there are 50 videos with identical length and the video popularity is followed by Zipf's distribution. Figure 5.9 shows the blocking probability as a function of arrival rate. It can be seen from the results that the blocking probability of the system is increasing when the system arrival rate is increased. It is because more multicast channels are required when the arrival rate is high. In additional, the impact of the number of channels reserved for multicasting (i.e. K) is also presented in this figure. It can be found that the system performance can be improved significantly when K is increased.

Figure 5.10 depicts the number of channels required to support the system with less than 5% blocking probability in various arrival rates. As we expected, more channels are needed when the arrival rate is high. From the results, we can see that about 300 channels are enough to support the arrival rate of 1 req/s with less than 5% blocking probability.

5.5.3 Performance of PPB Policy with Fault Exception

We now investigate how the early departure behavior of the peers affects the system performance. Unless otherwise specified, the peer's *MTD* is assumed to be 1200 seconds. Figure 5.11 shows the transmission cost against W when the arrival rate is 0.1 req/s. It can be observed that the mathematical model also closely matches with the simulation results. By comparing with Figure 5.5, W_{opt}^{F} is smaller (i.e. about 480 seconds) and the transmission cost is reduced as well. On the other hand, the transmission cost for recovery process (i.e C_r^F) is also presented in this figure. It can be seen that longer W requires more resources for recovery since more peers join the same video session and

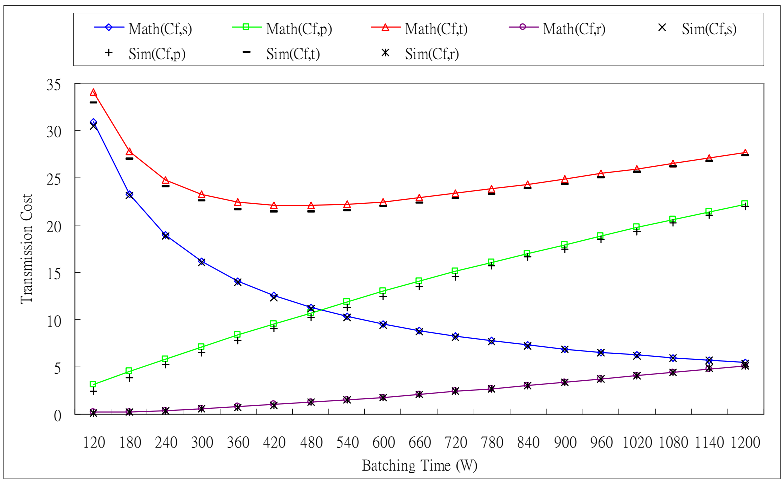


Figure 5.11. Transmission Cost against Various Batching Time W

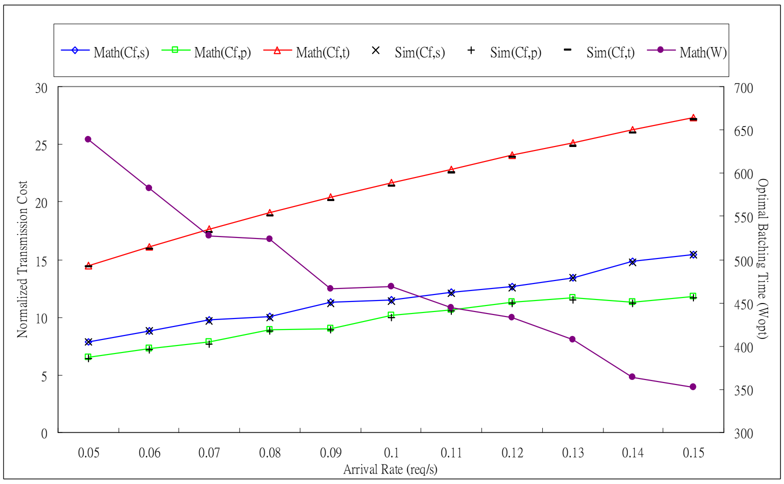


Figure 5.12. Optimal Batching Time and Its Corresponding Transmission Cost Requirement against Arrival Rate

the video patch for subsequent peers is longer. The P2P network formed by a peer group can be dismissed early when W is short and thus peers leaving within W is infrequent. Conversely, when W is increased, more peers depart before completely sending/receiving the video patch. Therefore, more resources for recovery are required. The optimum value of W and the corresponding transmission cost against the arrival rate is plotted in Figure 5.12. The trend is similar to that of Figure 5.6. It can be seen that the transmission cost

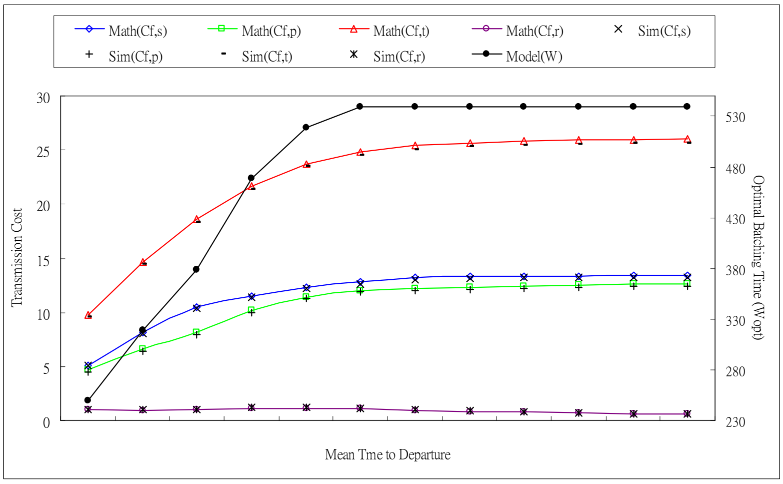


Figure 5.13 Transmission Cost and Its Corresponding Transmission Cost Requirement against *MTD*

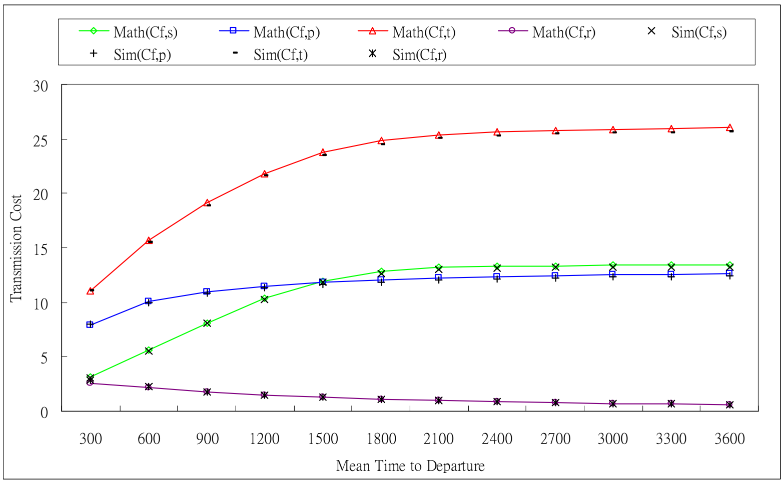


Figure 5.14 Transmission Cost again *MTD* with W_{opt}

required for fault recovery is slightly increasing when the arrival rate is increased.

In Figure 5.13, the transmission cost and W_{opt}^{F} are plotted against *MTD* when the arrival rate is 0.1 req/s. Longer *MTD* requires more transmission cost since more peers stay in the system and W_{opt}^{F} should be deflate to enlarge the size of the P2P network. However, when *MTD* > 1800s, it can be found that the transmission cost is slightly increasing because fewer peers leave early and the recovery process is rarely triggered. In

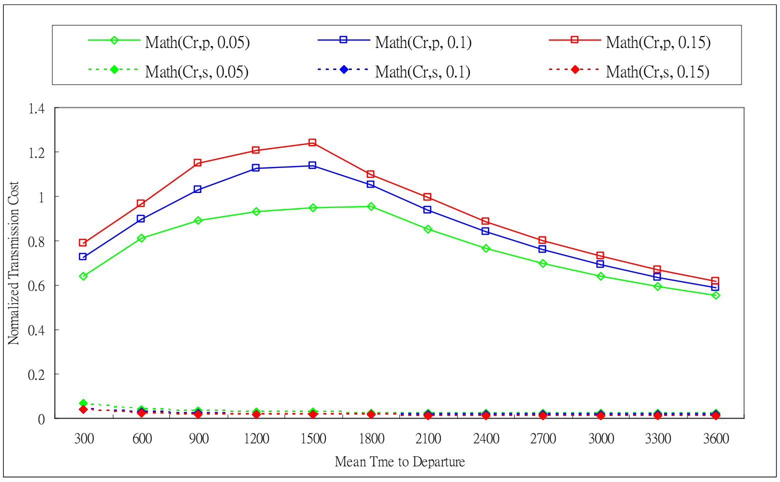


Figure 5.15. Distribution of Transmission Cost for Recovery between the P2P Network and the Central Server

Figure 5.14, we also investigate the impact of *MTD* but W_{opt} is used (W_{opt} is obtained from eqn.(5.7)). We can see that the total transmission cost with W_{opt} is higher than that with W_{opt}^{F} when the departure rate of the system is high. Although the transmission cost required for the central server with W_{opt} is slightly lower than that with W_{opt}^{F}, it is evident that the P2P network and fault recovery with W_{opt} require higher transmission cost because most of the peers leave the system before *W*. The results show a strong relationship between the batching time and the departure rate, which bring a great impact on the system performance. Therefore, these results bring us important information that *W* should be adjusted in respond to the variation of *MTD* such that the resources can be utilized efficiently and the overall system transmission cost is optimal.

As mentioned in Section 5.3, the bandwidth allocation for fault recovery is contributed by the P2P network as well as the central server. Figure 5.15 shows how the transmission cost is shared between the P2P network and the central server for fault recovery when *MTD* is varied. It can be seen that the resources for fault recovery are mainly contributed by the P2P network. From the results, the transmission cost is first increasing and then decreasing when *MTD* is increased. The reason is that when *MTD* is small, the size of the P2P network is small and thus the bandwidth required for the video patch for one peer group is low. As a result, the transmission cost required for fault exception is low. When *MTD* is increased, the size of the P2P network is increased as well. The bandwidth required for the video patch for one peer group is also increased. However, the transmission cost is decreasing when *MTD* is beyond certain value. It is because peers are rarely to leave the system and thus fewer recovery operations are launched.

5.6 Summary

In this chapter, we propose a new batching policy called peer-to-peer batching (PPB) to explore the multicast delivery coupled with P2P transmission strategy for video streaming. To provide the zero-delay services, customers form a P2P network and use a cache-and-relay manner to deliver the initial portion of the video for the late-coming customers during the beginning of the video session. When the size of the P2P network is large enough, they will be grouped together and served by a multicast channel so that the network bandwidth will not be exhausted rapidly. We also consider the issues of fault exception in our system framework. It is found that that the resources can be utilized more effectively when the departure rate of customers is taken into account. In addition, a mathematical model is developed to determine the optimal batching time such that the overall transmission cost is minimized. From the results, we can find that this policy can leverage the workload of the central server about 50%.

Chapter 6

Peer-to-Peer Video-on-Demand Systems in Broadcast Environment

6.1 Introduction

In Chapter 5, we proposed a new PPB policy to take both benefit of multicast delivery and P2P paradigm for VoD services. It is shown that PPB can leverage the workload of the central server about 50%. Although the network resources can be utilized more effectively using PPB when compared with other existing schemes, there are still rooms for enhancement. With the PPB scheme, the central server still needs to reserve quite a number of resources for video multicasting. In this chapter, we develop a generic VoD system model that uses P2P paradigm coupled with video broadcasting and CDN-like approach for video delivery. Unlike PPB that the delivery of the leading portion of a video should be accomplished in chaining manner and each peer should have identical bandwidth, the proposed framework in this chapter allows peer to select a set of peers with different bandwidth capacity for video transmission. In addition, we also propose a distributed scheme to disperse the workload from the central server to the peer side.

In the proposed framework, a video is first divided into two parts. The first part of the video is transmitted among peers by P2P manner such that customers can obtain the service in a short time. The second part of the video is periodically broadcasted by a number of high-end machines called peer server which has a higher storage space and bandwidth capacity than normal peer. The peer server is assumed to preload the video in its storage given that the video access pattern changes slowly. Therefore, customers will obtain the whole copy of video from the P2P network as well as broadcasting channels simultaneously. This content delivery strategy allows the workload of the system disperse over the network and also takes a benefit from video broadcasting. However, as mentioned, in real situation, the outbound bandwidth of a normal peer is limited that may not be able to support a full playback rate for video transmission. Also, both the normal peer and peer server may leave the system at arbitrary time. Therefore, a central server is still deployed to avoid the disruption of the service as PPB. The responsibility of the central server has twofold. It provides the first part of the video to the customers until the aggregated bandwidth of the supplying peers is sufficient to support the subsequent peers. It takes over the duty of video broadcasting when any peer server leaves the system. In order to increase the system reliability, a fault tolerant mechanism based on replication as well as erasure correcting codes are proposed for peer servers. The focus of this work is to study the features of the proposed VoD architecture by analytical model. We first examine how the partition of the video impacts on the system performance in terms of bandwidth requirement. Then, we investigate how various system parameters affect the proliferation of the system capacity as well as the usage of the central server resources. We also explore the relationship between the number of peer servers required for the system and the server resources such that the minimum number of peer servers required for the system can be determined when the central server resource is given. This study allows us better understanding of the system dynamics and provides guidelines for the management of design resources and realization of VoD services based on our proposed architecture.

In the following, this chapter is organized as follows. In Section 6.2, we discuss our proposed system architecture for video streaming using video broadcasting technique coupled with P2P paradigm. The analytical model of the proposed architecture is developed in Section 6.3. In Section 6.4, the results will be presented to demonstrate the efficacy of the proposed system. In Section 6.5, we describe how erasure correcting code can be applied to the proposed framework. Finally, some concluding remarks are given in Section 6.6.

6.2 Description of the Proposed Policy

Figure 6.1 illustrates a generic architecture of the proposed VoD system. The central server has a large storage space to store all the available videos for clients. It is assumed that a generic network infrastructure that supports broadcasting operations is used to implement

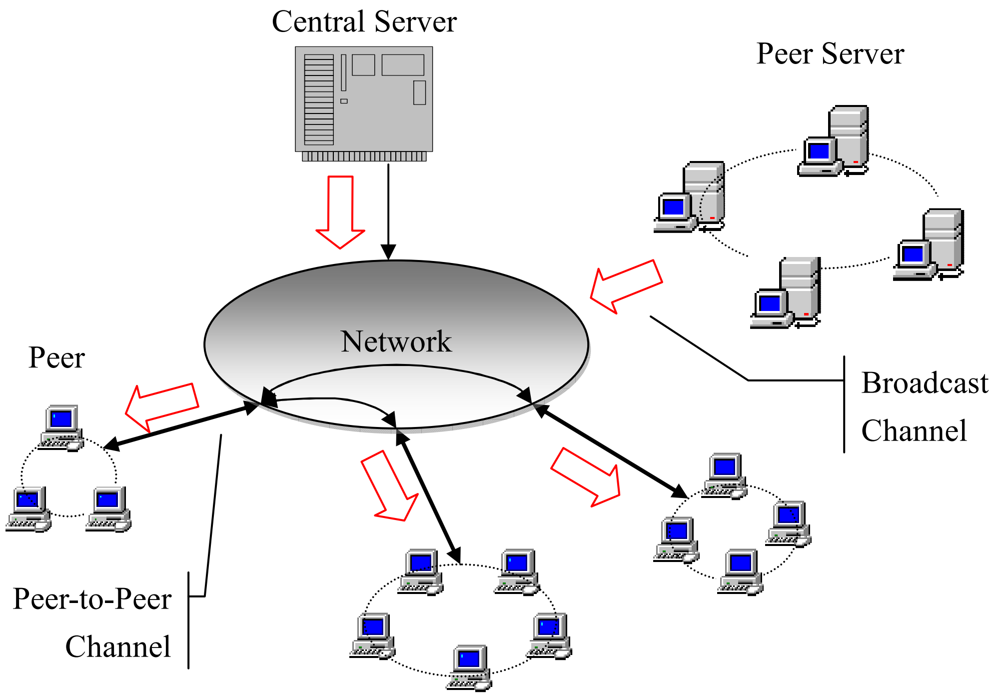


Figure 6.1. The Architecture of the Proposed Policy

the broadcasting protocols. In general, most of the existing broadcasting protocols can be applied to the proposed system framework. In this chapter, Staggered Broadcasting (SB) scheme, in particular, is used as an example to illustrate the design issue. In the proposed architecture, we define two classes of peer. The first class named Peer Client (PC) which only contributes a little storage and bandwidth capacity for the system and shares its cached content to other PCs such that a P2P network is constructed. The PC which is currently ready to stream its content to other PCs in the system is named Serving Peer Client (SPC). In addition, PC is assumed to perform data sharing only when the video is playback. Once the playback has been completed, it will leave the system and the cached content is assumed to be erased. The second class of peer called Peer Server (PS) which supports a full copy of video and has a bandwidth of at least the playback rate. PS is

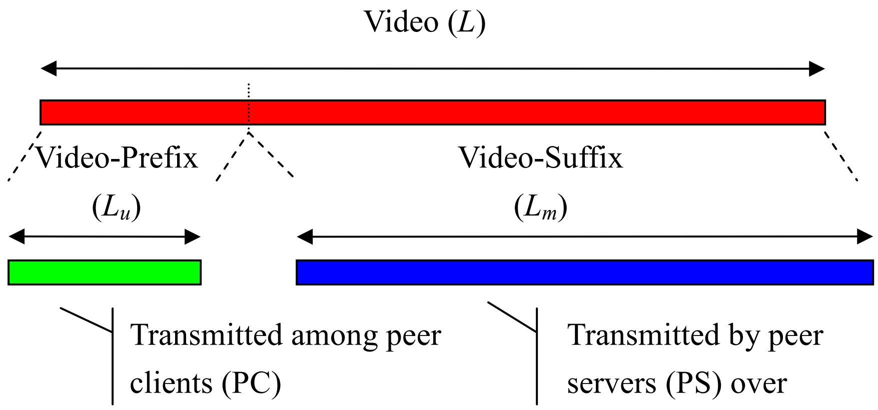


Figure 6.2 Video Partitioning Scheme

assumed to preload a part of the video in its storage given that video access pattern changes slowly. Each PS in the system is allowed to leave and enter the system at arbitrary time. Specifically, as shown in Figure 6.2, a video is divided into two pieces. The first piece of the video named video-prefix is shared among PCs over the P2P network while the second piece called video-suffix is transmitted by PSs over the broadcast channels periodically. Depending on the underlying broadcasting scheme, the video-suffix may be further partitioned into a number of segments [HUA97, JUHN97, LIU03], each of which is transmitted over the dedicated broadcast channel periodically. The size of the video-prefix and video-suffix have great impact on the system performance in terms of overall bandwidth requirement or transmission cost and their optimal size will be determined in Section 6.3.

When a new-coming peer admits into the system, it first searches for the video-prefix from the P2P network. One possible way to accomplish this purpose is to employ the well-known Distributed Hash Table. Once the requesting peer has been found that E serving peer clients (SPC) have its requested video content, it should schedule which SPC(s) should take part in the data transmission. It can be achieved by using the rate allocation algorithm (RAA) and the packet partition algorithm (PPA) defined in [NGUYE02]. The video channel established between the requesting PC and SPC is denoted as Peer-to-Peer Channel (PPC). However, if the request cannot be satisfied by the P2P network or the aggregated upload bandwidth among E SPCs is insufficient to support the request, this peer should make a new request to the central server. Then, the central server will open a dedicated channel for this request directly. The video frames received are played and cached in its local buffer. When the PC has completed downloading the video-prefix, it becomes a SPC and it is ready to serve other subsequent PCs. In general, the capacity of the P2P network is self-amplification. Therefore, the workload of the central server only persists for a short period. Apart from receiving the video-prefix from the P2P network or the server, peers are also required to download the video-suffix from the broadcasting channels. PC starts fetching data from the broadcasting channels right after it can start downloading the video-suffix according to the policy of a particular broadcasting scheme. When the playback has been completed, PC leaves the system and it is no longer to contribute its capability to the system.

As mentioned before, the video-suffix is handled by PSs. According to the SB protocol, the system requires to allocate X video channels for video broadcasting. Therefore, if each PS can contribute B video channels ($B \geq 1$), each of which has the same rate of the playback rate, the system should employ X/B PSs. However, similar to other P2P applications, each PS in the system is allowed to leave and enter the system at arbitrary time. Therefore, each video channel may require more than one PS in order to increase the system reliability. All PSs handling the identical video channel form a peer server group (PSG) and only one of them denoted active serving peer (ASP) will be responsible for broadcasting the video. When the system is launched, one of PSs in each PSG will be selected as an ASP to broadcast the stored video over the network. The simplest way to determine which PS provides the service in the same group is described in the following. When a PS is online, it generates a random number and its access time (referenced by the reference peer) as its identifier and then exchanges this identifier with other PSs in the same PSG periodically. A list of identifier is then collected and stored in a PSG table which is arranged in *first-come-first-list* manner according to the access time. When PS is offline, it broadcasts a "leave" message with its identifier to other peers. Then, its identifier will be removed from other PSs' PSG table. On the other hand, each identifier in the PSG table is also associated with a count down timer, which is reset whenever the corresponding PS's identifier is arrived. If the timer has elapsed, the corresponding PS is considered as departure or failure and thus its identifier will be removed from the PSG

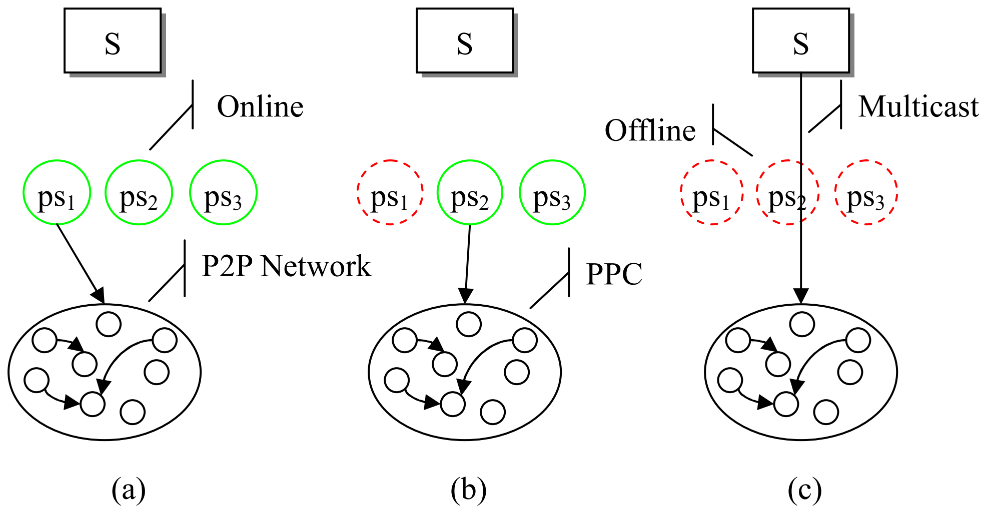


Figure 6.3. Idea of Video-Suffix Delivery

table. Therefore, when a PS finds that its identifier is listed at the top of the PSG table, it will become an ASP. As shown in Figure 6.3, there are three PSs forming a PSG and ps1 is selected to broadcast the video. After some time has elapsed, the current ASP (ps1) departs. Another PS in the same PSG is then picked to take over the service (i.e. ps2, see Figure 6.3b). However, if all PSs for this video leave, the central server should handle the service in order to avoid disruption of service (see Figure 6.3c). Once one of the PSs in the PSG is online again, the central server will return the duty to this PSG. It should be noticed that this PS should obtain the synchronization information from the central server before starting the service. In general, the central server provides the services to clients only when all PSs in the PGS are offline. Therefore, in order to leverage the workload of the central server, we should determine the minimum number of replica for each video channel which will be described in Section 6.4.

On the other hand, in order to ensure that all PSs in the same group can synchronize with each others, a clock synchronization procedure is required. This synchronization step can be accomplished by the Probabilistic Clock Synchronization (PCS) algorithm [ARVIN94]. In the proposed architecture, ASP will act as a reference peer in PCS. To allow other PSs and the central server estimate the current time on ASP's clock, ASP is required to broadcast a sequence of synchronization messages to other PSs in the same group periodically. If other PSs and the central server have received n synchronization messages during the progress, the ASP's clock can be estimated by using the following equation

$$T_e = R_n - \overline{R}(n) + \overline{T}(n) + d \tag{6.1}$$

where

$$\overline{T}(n) = \frac{1}{n}\sum_{i=1}^{n} T_i \quad \text{and} \quad \overline{R}(n) = \frac{1}{n}\sum_{i=1}^{n} R_i$$

T_i is the timestamp recorded by ASP on the i^{th} synchronization message transmitted, R_i is the time at which the i^{th} synchronization message is received by other PSs and d is an estimate of the expected value of the message delay. It should be mentioned that, the central server will become the reference if all PSs depart in one of the PSGs. On the other hand, the current frame number of the video should be associated with the synchronization message. Therefore, by using this clock information, each PS can estimate the how many frames have been transmitted when the current ASP has left.

Symbol	Meanings
λ_p	System arrival rate (reqs/s)
γ_p	PC departure rate (departure/s)
μ_p	Download rate of video-prefix
μ_s	Download rate of video-suffix
γ_{up}	Mean up time of the PS
γ_{down}	Mean down time of the PS
B	Number of video channels contributed by the PS
R	Data rate of the video (bit/s)
L	Length of the video (s)
L_u	Length of video-prefix (s)
L_m	Length of video-suffix (s)
S_u	Expected number of PPCs required for the system
S_m	Expected number of broadcast channel required for the system
L_u^{opt}	Optimum length of video-prefix (s)
L_m^{opt}	Optimum length of video-suffix (s)
G_{down}	Number of PCs obtaining the video-prefix
G_{up}	Number of SPS serving the video-suffix
$\overline{U}$	Average number of channels contributed by PC
J	Number of PSG required for the system
A	Availability of PS
Z^j	Mean serving time of PSG j
D^j	Mean serving time of the central server for PSG j
T^j	Mean time of renewal period of PSG j
B_S^j	Bandwidth requirement of the central server for PSG j
B_S	Total bandwidth requirement of the central server for video broadcasting
N	Total number of PSs required for the system

Table 6.1. Notations of Symbol

6.3 System Modeling

In this section, we develop an analytical model to evaluate the performance of the proposed architecture. We first examine how the length of video-prefix and video-suffix affects the system performance so that their optimal length can be determined. Then, with the use of a simple fluid model and queuing model, we can show that the proposed architecture is scalable. Finally, we find out the optimal number of PSs such that the workload of the central server can be reduced to a target level. To facilitate our discussion, we define the notations in Table 6.1.

6.3.1 Video Partitioning

In the proposed architecture, the video is partitioned into video-prefix as well as video-suffix, which is handled by the PCs in the P2P network and the PSs respectively. Obviously, their size is critical on affecting the system performance in terms of overall bandwidth requirement or transmission cost. If the video-prefix is too short, more multicast channels will be allocated. If it is too large, more PPCs will be opened. Therefore, an analytical model is developed to determine the optimal value of L_u (L_u^{opt}) and L_m (L_m^{opt})

In order to calculate the number of video channels required for the system including both PCC and multicast channel, we first assume that the customer arrival process is a

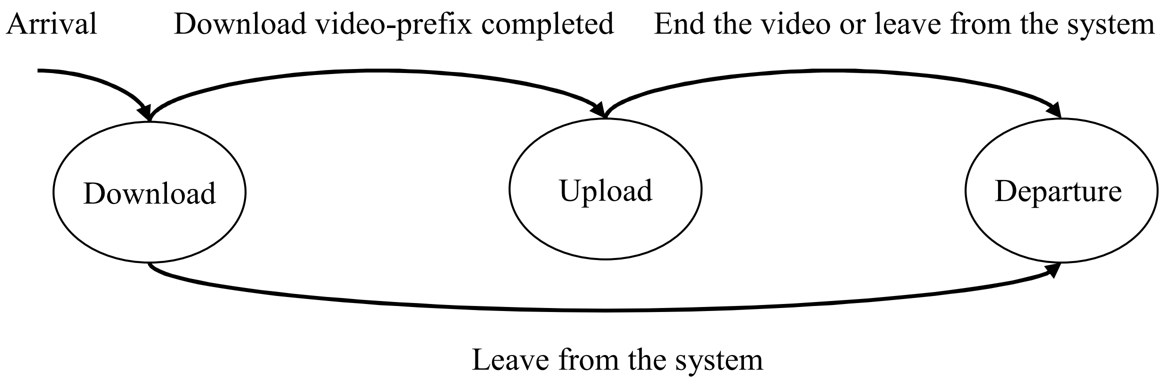


Figure 6.4. State Diagram of PC

Poisson process with rate λ_p. Then, the expected number of PPCs required for the P2P network (S_u) can be calculated by

$$S_u = \lambda_p L_p \tag{6.2}$$

Next, we consider the total number of concurrent video channels allocated for multicast delivery (S_m). As mention in Section 6.2, most of the efficient broadcasting protocols can be applied to the system framework to deliver the video-suffix over the broadcasting channels. Depending on the particular broadcasting scheme, B^x is denoted as the number of multicast channels required for the protocol x to broadcast the video-suffix and S_m is thus equal to B^x. For SB, a new broadcast channel should be opened at fixed interval of L_u seconds. Thus, S_m is given by

$$S_m = \frac{L - L_u}{L_u} \tag{6.3}$$

Therefore, the total bandwidth required for the system (S_T) is $S_T = S_u + S_m$. If λ_p and L are fixed to find the optimal value of L_u, we define $g(L_u) = S_u + S_m$. By setting the derivative of $g(L_u)$ with respect to L_u to zero, L_u^{opt} is then obtained as

$$L_u^{opt} = \sqrt{\frac{L}{\lambda}} \tag{6.4}$$

6.3.2 Modeling of PC

Figure 6.4 shows the state-space diagram of the proposed system. When a PC admits into the system, it first enters the "Download" state to obtain the video-prefix from the P2P network. When the retrieval of video-prefix has been completed, it goes to the "Upload" state. In this state, it acts as a SPC to serve other subsequent arriving PCs in the system. As mentioned before, the PC is no longer used to provide services to the system when its playback has completed and thus it moves to the "Depart" state. Similar to other P2P applications, PC is allowed to leave the system at any time. Therefore, it is also possible to move to the "Depart" state from either the "Download" state or the "Upload" state. With this state-space diagram, a queuing network model is developed to explore several system parameters, such as the average number of PCs which is currently downloading the video-prefix from the P2P network, the average number of SPCs which is serving the system as well as the workload of the central server.

In the proposed architecture, a PC should download the video-prefix for L_u^{opt} seconds from the P2P network. The total streaming rate required for $G_{down}(t)$ PCs should be supported by the central server and/or SPCs. Then, the transition rate of the system from "Download" state to "Upload" state can be expressed as $\mu_p G_{down}(t)$, where

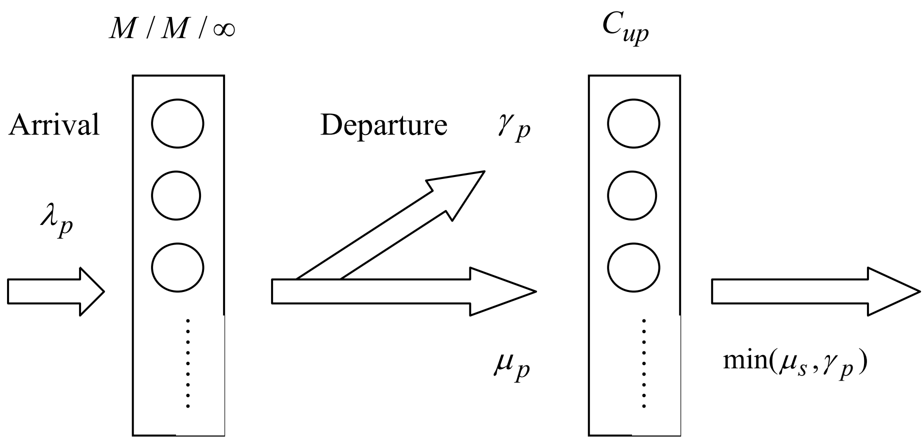


Figure 6.5. Queuing System of the Proposed Framework

$\mu_p = \dfrac{1}{L_u^{opt}}$. Occasionally, the PC may leave the system when it has completely played back the video. Assume that each PC independently leaves the system after a certain amount of time which is exponentially distributed with mean γ_p. Therefore, the transition rate of $G_{down}(t)$ PCs from "Download" state to "Depart" state is $\gamma_p G_{down}(t)$. Similar to "Download" state, a PC may not stay in the system before the playback has been completed when it is in the "Upload" state. Therefore, the transition rate of $G_{up}(t)$ SPCs from "Upload" state to "Depart" state is given by $\min(\mu_s, \gamma_p) G_{up}(t)$, where $\mu_s = \dfrac{1}{L_m^{opt}}$. With the above information, we are going to determine the number of PCs in the "Download" state and SPCs in the "Upload" state. When a PC admits into the system, it first retrieves the video-prefix from the P2P network and then acts as SPC to serve the system. Therefore, the system can be viewed as a tandem queuing network as shown in Figure 6.5. As mentioned before, if the aggregated bandwidth of the P2P network is insufficient to provide service to this PC, it should be served by the central server.

Therefore, the PC can admit into the system without blocking. From the viewpoint of queuing theory, this operation can be considered that there are infinite serving PPCs for PCs. As the arrival process is Poisson, the system can be approximated as an $M/M/\infty$ queuing system. From [PARLA00], the number of PCs of an $M/M/\infty$ at time *t* can be expressed by

$$G_{down}(t) = \frac{\lambda_p}{\gamma_p + \mu_p}\left[1 - e^{-(\gamma_p + \mu_p)t}\right] \tag{6.5}$$

In steady-state, the expected number of PCs in the system ($\overline{G_{down}}$) can be computed by $\overline{G_{down}} = \frac{\lambda_p}{\gamma_p + \mu_p}$ [PARLA00]. In order to determine the number of SPCs which is currently serving at time *t* (i.e. $G_{up}(t)$), we develop a simple fluid model for the evolution of $G_{up}(t)$, which is given by

$$\begin{aligned} \dot{G}_{up}(t) &= \mu_p G_{down}(t) - \min(\mu_s, \gamma_p) G_{up}(t) \\ &= \frac{\lambda_p \mu_p}{\gamma_p + \mu_p}\left[1 - e^{-(\gamma_p + \mu_p)t}\right] - \min(\mu_s, \gamma_p) G_{up}(t) \end{aligned} \tag{6.6}$$

By introducing the Laplace transform of the eqn.(6.6) and setting the initial value of $G_{up}(0) = 0$, we have

$$\begin{aligned} s\overset{*}{G}_{up}(s) - sG_{up}(0) &= \frac{\psi}{s} - \frac{\psi}{s + \gamma_p + \mu_p} - \min(\mu_s, \gamma_p)\overset{*}{G}_{up}(s) \\ \overset{*}{G}_{up}(s) &= \frac{\psi}{s(s + \min(\mu_s, \gamma_p))} - \frac{\psi}{(s + \gamma_p + \mu_p)(s + \min(\mu_s, \gamma_p))} \end{aligned} \tag{6.7}$$

where

$$\psi = \frac{\lambda_p \mu_p}{\gamma_p + \mu_p}$$

By using the inverse Laplace transform, $G_{up}(t)$ can be solved by

$$G_{up}(t) = A_1(1 - e^{-\min(\mu_s,\gamma_p)t}) - A_2 e^{-\min(\mu_s,\gamma_p)t} - A_3 e^{-(\gamma_p+\mu_p)t} \tag{6.8}$$

where

$$A_1 = \frac{\psi}{\min(\mu_s, \lambda_p)},$$

$$A_2 = \frac{\psi}{\gamma_p + \mu_p - \min(\mu_s, \gamma_p)}, \text{ and}$$

$$A_3 = \frac{\psi}{\min(\mu_s, \gamma_p) - \gamma_p - \mu_p}$$

Thus, the expected number of SPCs in the system in steady-state ($\overline{G_{up}}$) can be expressed as $\overline{G_{up}} = \dfrac{\lambda_p \mu_p}{\min(\mu_s, \gamma_p)(\gamma_p + \mu_p)}$. Assume that each SPC can contribute a bandwidth of $\overline{U}$ to the system on average. Then, the aggregated bandwidth of the P2P network at time *t* is equal to $\overline{U}G_{up}(t)$. Therefore, the bandwidth requirement of the central server at time *t* ($S_s(t)$) can be computed by

$$S_s(t) = \max(0, G_{down}(t) - \overline{U}G_{up}(t)) \tag{6.9}$$

Therefore, if $\overline{U}G_{up}(t) \geq G_{down}(t)$, the central server becomes idle. The time that the system reaches this state is called transition time. However, if $\overline{U}\,\overline{G_{up}} < \overline{G_{down}}$, the central server should reserved at least $\overline{G_{up}} - \overline{U}\,\overline{G_{up}}$ channels for PCs.

6.3.3 Modeling of PS

Since the number of broadcast channels required for the system is S_m, the number of PSG

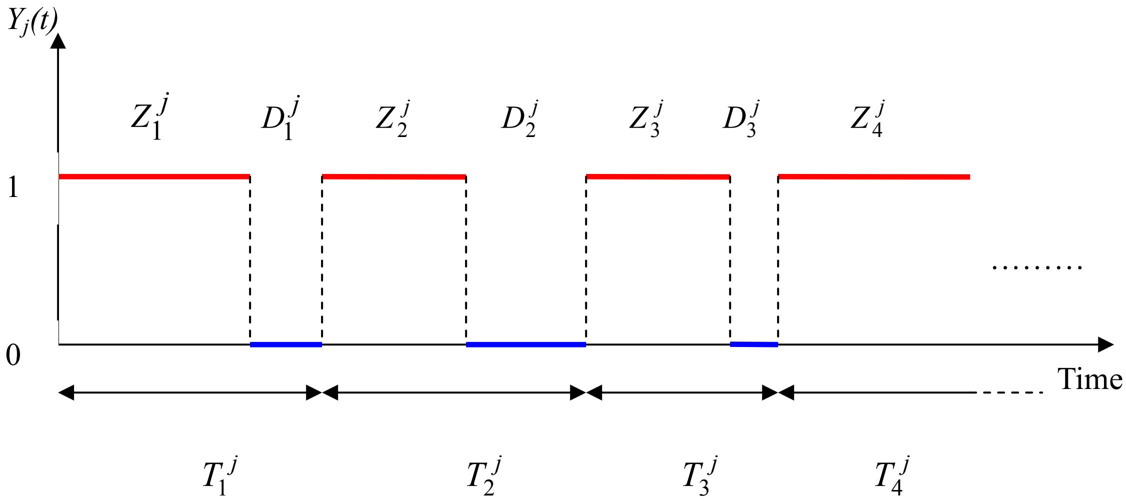
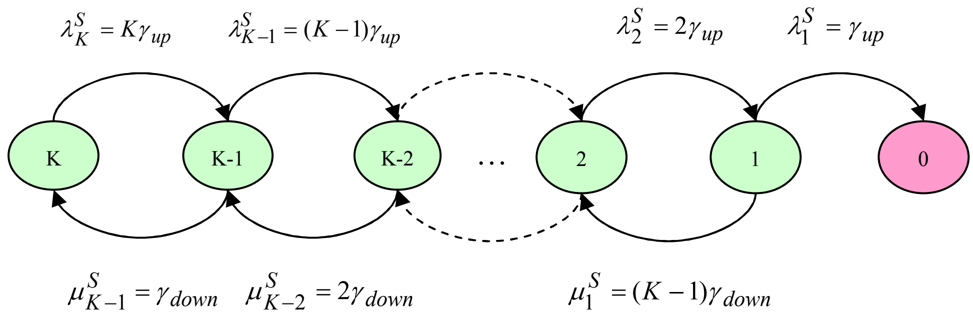
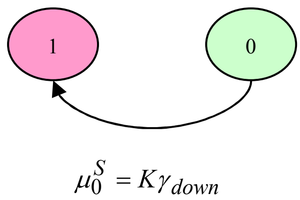


Figure 6.6. Alternative Renewal Process for PSG *j*

for the video (*J*) can be expressed by $J = \left\lceil \frac{S_m}{B} \right\rceil$ if each PS can contribute *B* channels, where $B>0$. Therefore, each PSG has to handle *B* video channels. Denote C_j as the set of the assigned channels for PSG *j*, where $j=1,2,3\ldots,J$. Since each PS can leave and enter the system at anytime, it is assumed that the mean up time and mean down time of each PS are independent and identically distributed with exponential function with the rate γ_{up} and γ_{down} respectively. Then, the availability of each PS can be defined as $A = \frac{\gamma_{up}^{-1}}{\gamma_{up}^{-1} + \gamma_{down}^{-1}}$ [HOYLA94]. As mentioned before, the central server takes over the service only when all PSs in the PSG are offline and it returns the duty to PSG when one of the PSs in the group is online. Because C_j is served alternatively by PSG *j* and the central server, it forms an alternating renewal process [HOYLA94] as shown in Figure 6.6. Denote $Y_j(t)$ as the state variable of the system for PSG *j* at time *t*. It is equal to 1 if the PSG *j* is carrying the video broadcasting operation. When the central server is taking over the service, it is equal to 0.



(a) Peer Server Group



(b) Central Server

Figure 6.7. State-Space Diagram

Let Z_1^j, Z_2^j, … denote the successive serving time of PSG *j* for C_j and also let D_1^j, D_2^j, … denote the corresponding successive serving time of the central server. Therefore, we have a renewal process with renewal periods $T_i^j = Z_i^j + D_i^j$ for i = 1, 2, 3,…. Obviously, more PSs for PSG *j* results in longer $Z_k^j, \forall k$ and thus fewer resources from the central server are required. Denote N_j as the number of PSs deployed for C_j. To determine the mean serving time of PSG *j* (Z^j) with N_j PSs, we use a continuous-time Markov Chains (CTMC) model the PSG and the state-space diagram for this model is shown in Figure 6.7(a). It is assumed that all PSs are independent to each others and all PSGs also operate independently to other PSGs in the system. Thus, we can simply consider one particular PSG with *K* PSs. Denote λ_k^s and μ_k^s be the transition rate from state *k+1* to state *k* and from state *k-1* to *k* respectively. When the system is at state *K*, all PSs are available for broadcasting the segments. State 0 here denotes that all PSs in PSG leave and is assumed to be an absorbing state. We first define $P_k(t)$ to be the probability of the PSG in state *k* at time *t* (*k*=0, 1, 2, 3, … *K*). From Figure 6.7(a), the state equations of the CTMC are given by

$$\begin{aligned} \dot{P}_k(t) &= -(\lambda_k^s + \mu_k^s)P_k(t) + \mu_{k-1}^s P_{k-1}(t) + \lambda_{k+1}^s P_{k+1}(t) \quad &, k \geq 1 \\ \dot{P}_0(t) &= \lambda_1^s P_0(t) \quad &, k = 0 \end{aligned} \tag{6.10}$$

Since the transition rate matrix does not have full rank, we can remove the equation of *k=0* without loosing any information [HOYLA94]. Therefore, we obtain

$$\dot{P}_k(t) = -(\lambda_k^s + \mu_k^s)P_k(t) + \mu_{k-1}^s P_{k-1}(t) + \lambda_{k+1}^s P_{k+1}(t) \quad , k \geq 1 \tag{6.11}$$

By introducing the Laplace transform of the eqn.(6.11), we have

$$s\overset{*}{P}_k(s) - P_k(s) = -(\lambda_k^s + \mu_k^s)\overset{*}{P}_k(s) + \mu_{k-1}^s \overset{*}{P}_{k-1}(s) + \lambda_{k+1}^s \overset{*}{P}_{k+1}(s) \ , k \geq 1 \tag{6.12}$$

In the proposed architecture, the PSG will take over the duty when one of PSs in the group is online. Hence, the initial state of the PSG at time *t*=0 is defined as

$$\begin{aligned} &P_1(0) = 1 \\ &P_k(0) = 0 \qquad for\ k \neq 1 \end{aligned}$$

By inserting *s=0* in eqn.(6.12), we obtain

$$\begin{aligned} &-(\lambda_k^s + \mu_k^s)\overset{*}{P}_k(0) + \mu_{k-1}^s \overset{*}{P}_{k-1}(0) + \lambda_{k+1}^s \overset{*}{P}_{k+1}(0) = -1, k = 1 \\ &-(\lambda_k^s + \mu_k^s)\overset{*}{P}_k(0) + \mu_{k-1}^s \overset{*}{P}_{k-1}(0) + \lambda_{k+1}^s \overset{*}{P}_{k+1}(0) = 0, k > 1 \end{aligned}$$

(6.13)

Because the expected time to reach state 0 from state 1 is equal to $\frac{1}{\gamma_{up}}$, eqn.(6.13) can be solved for $\overset{*}{P}_k(0)$ for $k \geq 1$. Then, the mean time to failure (*MTTF*) of the CTMC can be determined by [HOYLA94]

$$MTTF(K) = \sum_{k=1}^{K} \overset{*}{P}_k(0) \tag{6.14}$$

Therefore, the mean serving time of PSG j (Z^j) with N_j PSs can be expressed as:

$$\begin{aligned} Z^j(N_j) &= MTTF(N_j) \\ &= \frac{1}{\gamma_{up}} \sum_{m=1}^{N_j} \left(\frac{{\gamma_{down}}^{m-1}(N_j - 1)!}{m{\gamma_{up}}^{m-1}(N_j - m)!(m-1)!} \right) \end{aligned} \tag{6.15}$$

To compute the corresponding mean serving time of the central server (D^j), we use another CTMC model as shown in Figure 6.7(b). When the system is at state 0, the central server is taking over the broadcasting operation. If one of the PSs in the PSG is online again, the system will move to state 1 so that the broadcasting operation is returned to this PSG. Therefore, D^j can be computed by

$$D^j(N_j) = \frac{1}{N_j \gamma_{down}} \tag{6.16}$$

Then, the average bandwidth reserved by the central server for PSG j can be given by

$$B_S^j(N_j) = \frac{BR \cdot D^j(N_j)}{D^j(N_j) + Z^j(N_j)}, \tag{6.17}$$

where R is the streaming rate of the video. Therefore, the mean time of renewal period of PSG j is equal to $Y^j(N_j) = D^j(N_j) + Z^j(N_j)$. Then, the overall bandwidth required for the central server (B_s) is thus computed by

$$B_S = \sum_{j=1}^{J} B_S^j(N_j) \tag{6.18}$$

In general, we want to determine the minimal number of PSs required for the system such that the central server resources for this video can be reduced from S_mR[5] to rS_mR, where $0 < r \le 1$. Therefore, the optimization problem is defined as below:

Minimize N

Subject to $B_S \le rS_mR$ and $N_i \ge 1,\ i = 1,2,3\ldots,J$

$$\text{where } N = \sum_{i=1}^{J} N_i \tag{6.19}$$

The second constraint implies that at least one PS should be deployed in each PSG. It should also be noted that N_i, where $i = 1,2,3\ldots,J$, should have the same value because all PSGs have equal importance.

6.4 Experimental Results

In this section, some numerical results from our analytical models and computer simulations are presented. Unless otherwise specified, the following settings are used for our evaluation. We assume that the video length is 7200 seconds (2 hours) and R=1. The average bandwidth contribution of each PC (i.e. $\overline{U}$) is 0.5. It is also assumed that the availability of each PS is 0.4, each of which reserves one video channel for video

5 In general VoD system, this bandwidth allocation is supported by the central server and we want to determine the number of PSs such that this workload can leverage to a particular value.

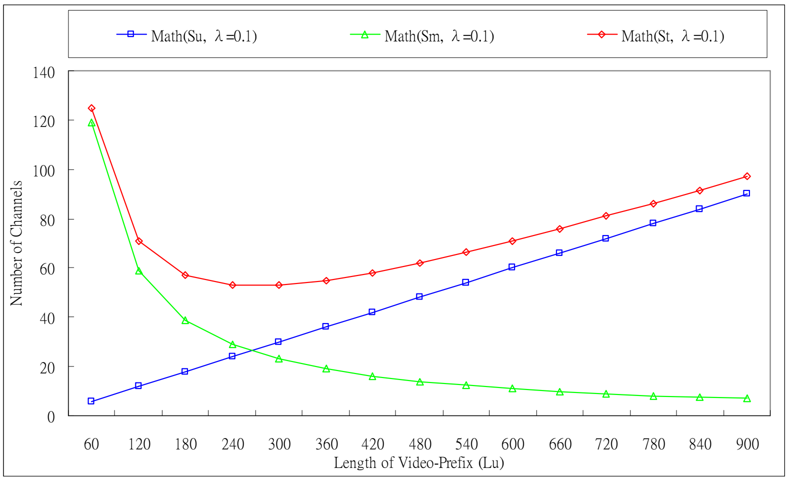
Parameter	Nominal Value	Range of Value
λ_p	0.1 req/s	0.05-0.15 req/s
$1/\gamma_p$	7200 seconds	300-7200 seconds
L	7200 seconds	-
$\overline{U}$	0.5	0.5, 1, 1.5
R	1	-
B	1	1, 3
r	-	0.01-1
A	0.4	0.1-1

Table 6.2. Summary of the System Parameters

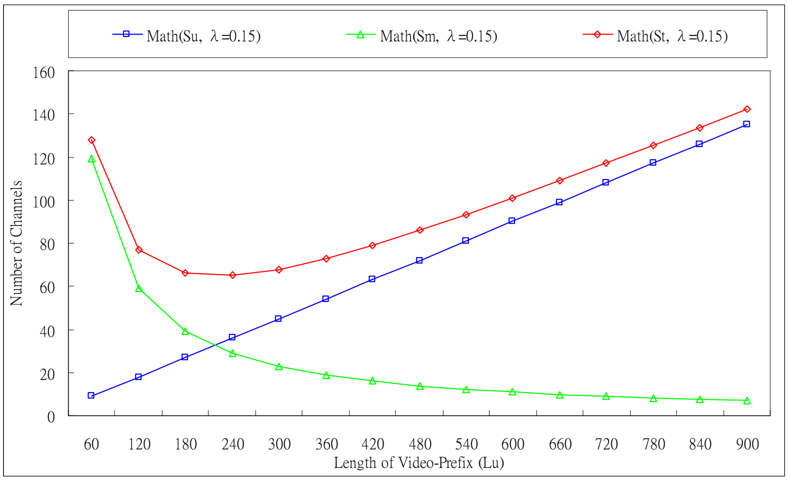
broadcasting. The mean up time and mean up time of each PS is γ_{up}=*36000A* seconds andγ_{down}=*36000(1-A)* seconds respectively. Therefore, the online/offline time of each PS is ranged between 1 to 10 hours governed by *A*. The parameters used for evaluation are summarized in Table 6.2. In the following results, each curve is represented by "*Model*[6]:(*a*, *b*)", where *a* is the output of the curve and *p* is the parameter for the curve.

We first investigate the bandwidth requirement against L_u with the arrival rate of 0.1 req/s and 0.15 req/s for a single video in Figure 6.8. From the results, we can see that S_m is decreasing when L_u is increased. However, S_u is increasing because the size of the P2P network is increased. It can be found that the minimum bandwidth requirement of the system for low popular video (0.1 req/s) can be achieved when L_u is 250 seconds. For the video with a higher arrival rate (0.15 reqs/s), it is as long as about 200s. To show the optimal length of the video-prefix (i.e. L_u^{opt}) in various arrival rates such that the overall

6 Math – results obtained from analytical model, Sim - results obtained from simulation



(a) λ_p=0.1



(b) λ_p=0.15

Figure 6.8. Number of Channel Requirement against Various L_u

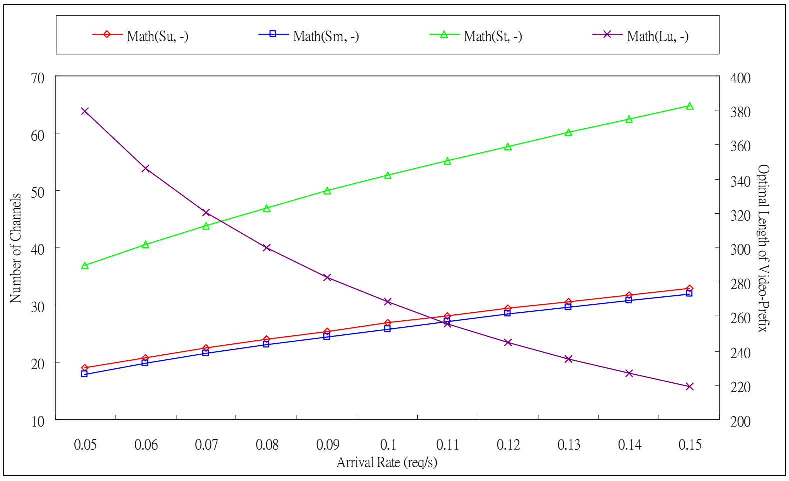
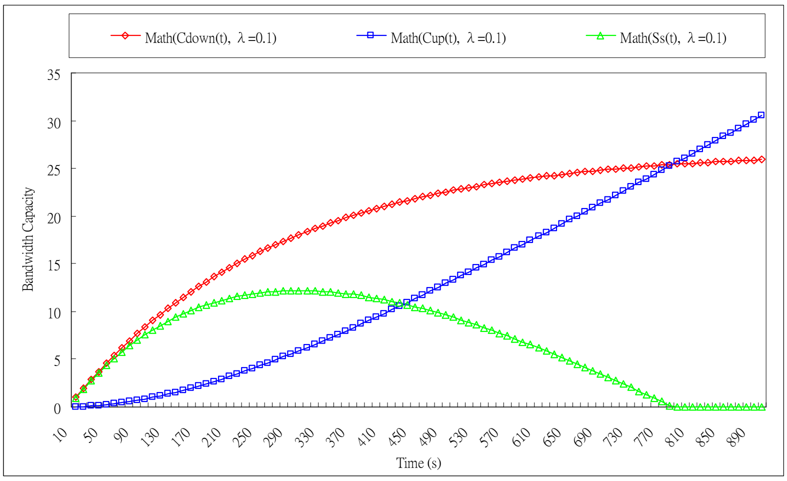


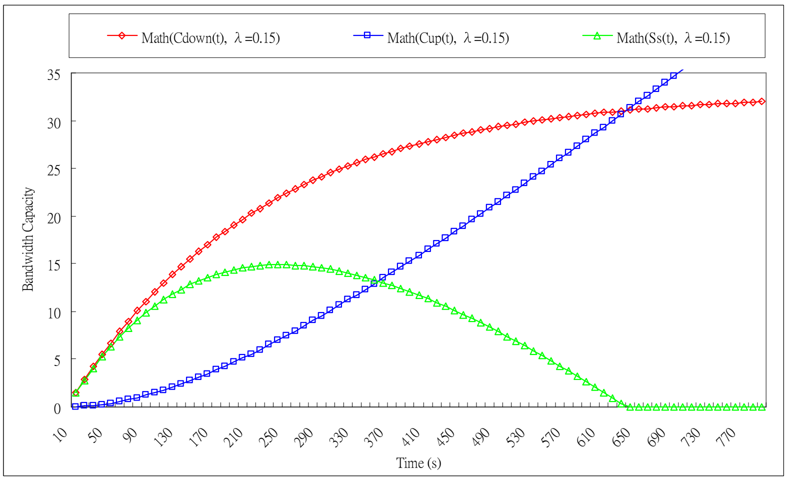
Figure 6.9. Optimum Length of the Video-Prefix and its Corresponding Bandwidth Requirement against Arrival Rate

bandwidth requirement of the system can be minimized, we plot the relationship of the arrival rate and L_u^{opt} in Figure 6.9 As we expected, more channels are required for higher arrival rate. The results show that L_u^{opt} is decreasing with the increase of the arrival rate. It can be observed that the bandwidth required for multicast transmission and P2P delivery are nearly the same when the system operates in the optimal state.

After examining how the length of the video-prefix affects the system performance, we are going to look at the evolution of the bandwidth capacity of the system in Figure 6.10 with $\lambda_p = 0.1$ and $\lambda_p = 0.15$. As shown in Figure 6.10(a), it can be observed that the bandwidth required for the central server is first increasing and then decreasing gradually during the operation when the system arrival rate is $\lambda_p = 0.1$. After the system is launched, only a few numbers of SPCs contribute to the system and thus most of the PCs will be served by the central server directly at the initial stage. When the operation is going on, the number of SPCs is increasing and the corresponding aggregated bandwidth capacity of the P2P network is expanding accordingly. Therefore, most of the subsequent requests can be supported by the P2P network directly and fewer server resources are involved as time goes by. From the results, we can see that the transition time of the system is about 800s and the maximum bandwidth requirement of the central server ($S^{\max}$) during the operation is about 12 channels. Similar observation can be found when the system arrival rate is $\lambda_p = 0.15$. But, the transition time of the system is reduced to about



(a) λ_p=0.1



(b) λ_p=0.15

Figure 6.10. Evolution of the bandwidth capacity as a function of time

650 seconds.

Figure 6.11 shows the transition time of the system for various arrival rates. With this, we can schedule different videos at different time with the use of the same resources. It can be found that the transition time is decreasing when the arrival rate is increased. From the graphs, we also present how the bandwidth contribution of a peer affects the system performance. It can be seen from the results that the transition time are decreased and when

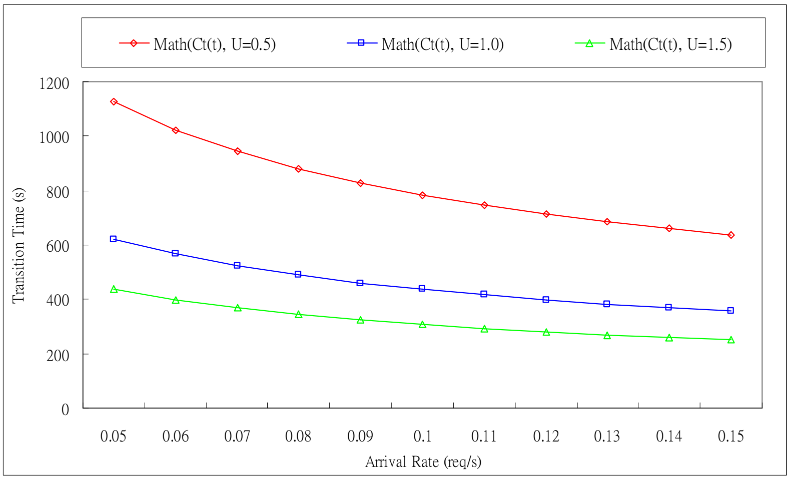


Figure 6.11. Transition time against various arrival rates

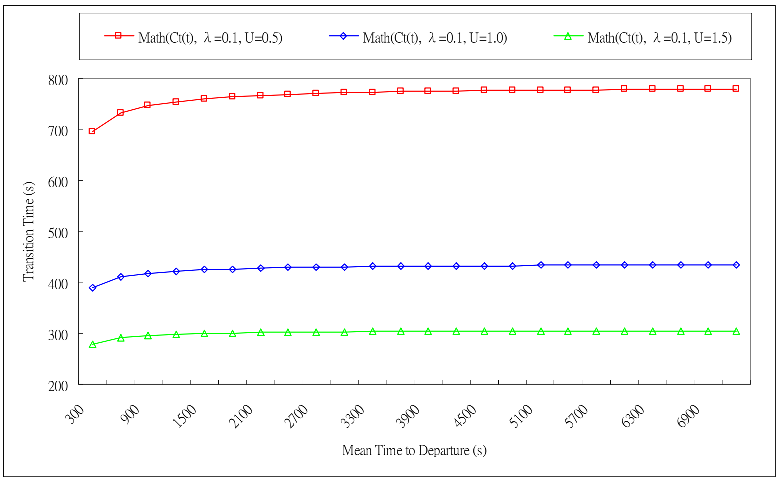


Figure 6.12. Transition time against *MTD*

$\overline{U}$ is increased. As mentioned, a peer is allowed to leave the system at arbitrary time. We now investigate how the early departure behavior of the peer affects the system performance. In Figure 6.12, the transition time of the system is plotted against mean time to departure (*MTD*) (i.e. $\gamma_p{}^{-1}$) . Longer *MTD* requires longer transition time since more peers stays in the system. However, when *MTD* > 1800s, it can be found that the transition time is slightly increasing only because fewer peers leave early.

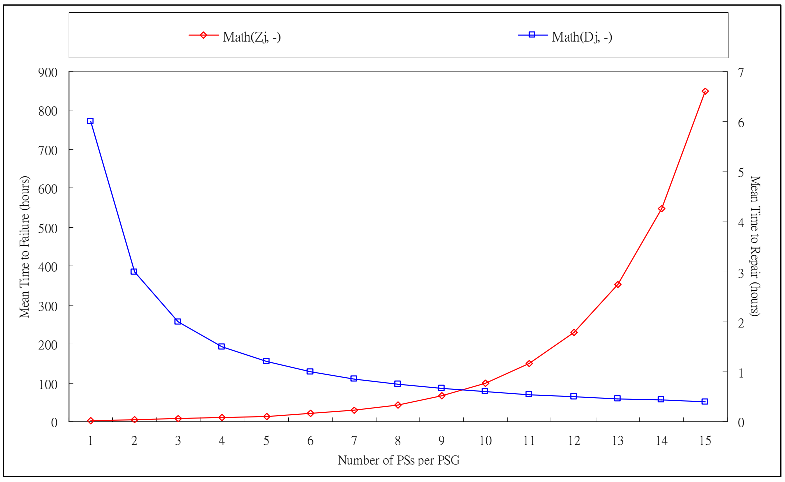


Figure 6.13. The *MTTF* of the System and its corresponding *MTTR* against the Number of PSs per PSG

Finally, we look at how the setting of PS affects the system performance. It is again assumed that $\gamma_{up} = 36000A$ seconds and $\gamma_{down} = 36000(1 - A)$ seconds. Therefore, the online/offline time of each PS is ranged between 1 to 10 hours governed by A. We first look at the performance of the system without the central server involved, i.e. the operation is identical to the DPCS [TO05]. Therefore, the service is disrupted when all PSs in the system are offline and it can be resumed again only when one of the PSs in the PSG is online. Figure 6.13 shows the *MTTF* of the system and its corresponding *MTTR* against the number of PSs deployed. As we expected, the serving time of the system is increasing and its corresponding repairing time is decreasing when the number of PSs is increased. It can be found from the result that the customers should wait for about 20 minutes before the service can be started again once the service is suspended if there are 15 PSs used. Thus, more PSs should be required in order to reduce the suspension duration of the service.

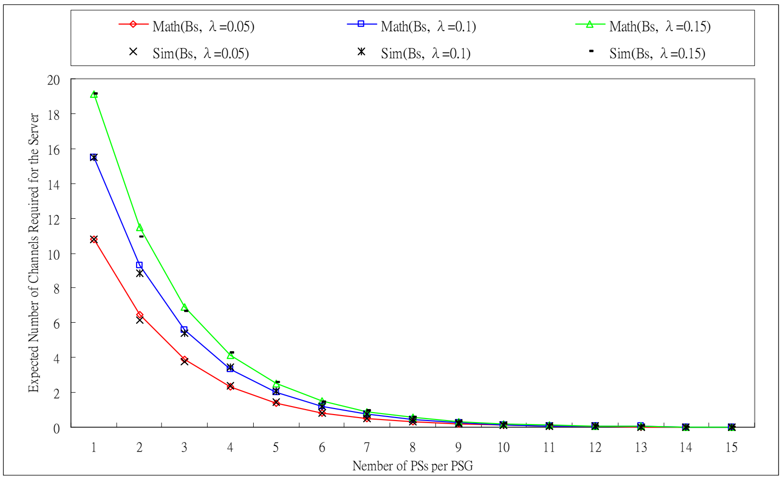


Figure 6.14. Server Bandwidth Requirement against the Number of PSs per PSG

However, such framework cannot still prevent from the disruption of the service. In addition, enormous number of PSs also increases the cost of the system. Therefore, a central server is still used in our framework to make a tradeoff between the number of PSs required and the central server resources needed.

Figure 6.14 depicts the average bandwidth requirement of the central server against the number of PSs required for each PSG with different arrival rates. As we expected, more PSs requires fewer central server resources. More PSs are deployed in the system, Z^j can be extended longer and the frequent of redirect from PSG to the central server is thus decreased. On the other hand, with the same number of PSs, higher arrival rate requires more resources because more PSG required for the video. However, it can be observed that the change is not significant even if more PSs are deployed into the system when $N_j>10$. It is because Z^j is less responsive to the extra PSs (c.f. eqn.(6.15)).

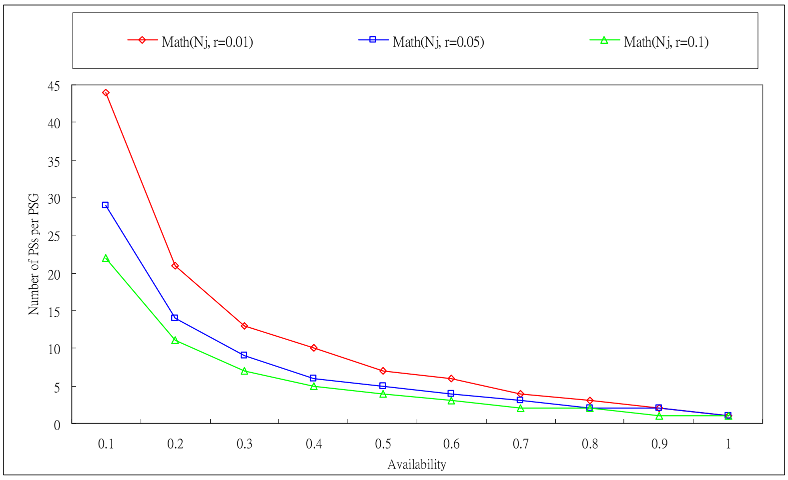


Figure 6.15 Number of PSs per PSG against the Availability of PS

As mentioned, a large amount of server bandwidth is still required for video broadcasting in general VoD system. Then, we have proposed to shift this duty to a number of PSGs. However, the central server still requires to take over the broadcast delivery when all PSs in the same group depart. Thus, the central server should reserve a portion of bandwidth for video broadcasting. Specifically, we should determine the number of PSs required for the system such that the server bandwidth required can be reduced to the desired level from that in general broadcast VoD system. Therefore, the number of PSs required for each PSG against the PS availability is plotted such that the target reduction of the central server resources can be achieved. In Figure 6.15, it is found that the number of PSs is decreasing when the availability is increased. For the same value of r, when the availability is increased, the lifetime of each PS is also increased and thus fewer PSs are required for standby. From the results, the workload of the central server can be reduced by 95% compared with the original PPB when there are 156 PSs (Note that there are 26 PSG

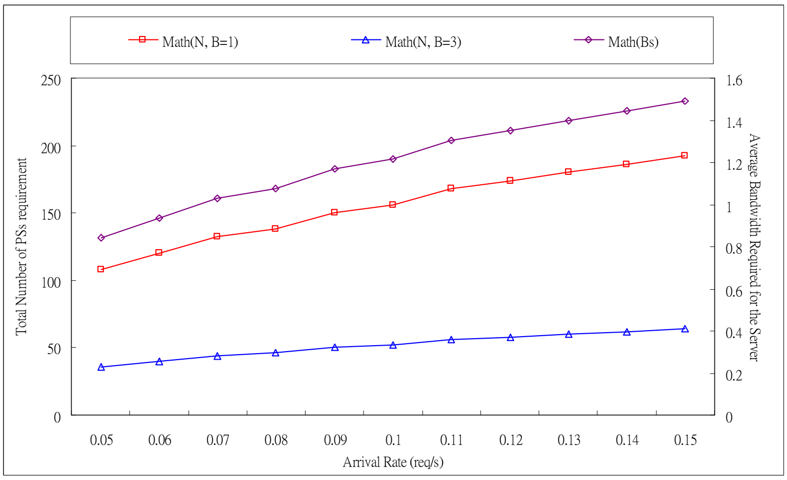


Figure 6.16 Number of PSs Requirement and Its Corresponding

in the system) with the availability of 0.4.

Figure 6.16 shows the total number of PSs and its corresponding server bandwidth requirement against the arrival rate when the availability of each PS is 0.4 and $r=0.05$. It can be observed that the system requires less than 200 PSs to achieve 95% reduction of the server resources. From the graph, we also present the system performance when the bandwidth contribution of each PS is increased. It can be seen that the total number of PSs required for the system can be significantly reduced to about 60 when $\overline{U}$ is increased to 3 channels.

6.5 Other Reliable Mechanism

In the original proposed framework, the video-suffix of a video is replicated to a number of PSs in order to support a certain level of system reliability. In this section, we use an alternative approach to achieve the same purpose based on erasure correcting codes (ECC)

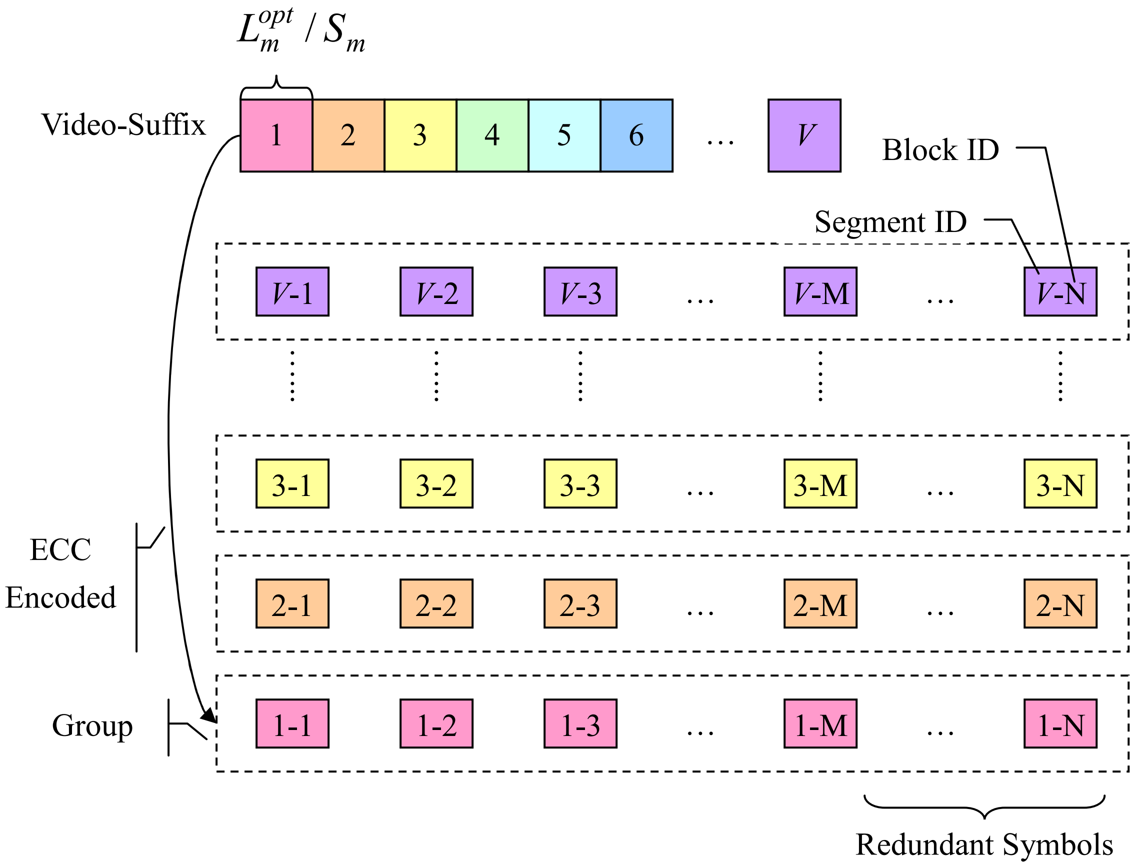


Figure 6.17. Idea of ECC

[MCAUL90] and compare its effectiveness to replication. Rather than producing duplicated contents in all PSs in the same PSG, the video-suffix is first partitioned and then applied to ECC encoder to generate a number of message and redundant symbols. Each PS holds one of the symbols and clients only need to have sufficient symbols that the original video content can be reconstructed.

6.5.1 Peer Server with Erasure-Correcting Codes

Similar to the original policy, the video-suffix is handled by a number of PSs. Instead of using the replication approach to produce a whole copy of the video in each PS in the PSG, an alternative approach based on ECC is used to improve the reliability of the system. The principle of ECC is to add extra information to the video-suffix. In ECC, the video-suffix is first partitioned into a number of segments, each of which has the size of L_m^{opt} / S_m seconds as shown in Figure 6.17. Each segment is packetized into *M* blocks to form a group which is then applied to an ECC encoder to generate *N* blocks output (*N*>*M*) and thus an ECC(*N, H*) codeword is produced, where *H*=*N*-*M* which is the redundant blocks in the codeword. Therefore, the overhead of the codeword (*O*) can be expressed as *O*=*H*/*M*. To reconstruct the original group of blocks, the receiving side only needs to receive any *M* blocks out of the *N* blocks correctly. In order words, it can correct up to *H* missing blocks per group during transmission. The transmission rate of each video session will be increased to ηR, where η is known as the code rate of the codeword which can be expressed as $\eta = N / M$. In the ECC approach, each PS is required to store at least one of *N* blocks in its local storage and contributes a bandwidth of $\eta R / N$ in contrary to the replication approach that each PS has to hold a complete video-suffix and to have a bandwidth of *R*. Although the transmission rate of each video session is increased by the factor of η, it should be mentioned that symbols are distributed to a number of PSs in our proposed system and thus it is not necessary to broadcast all symbols over the network in contrast with the conventional ECC transmission that a dedicated server should transmits

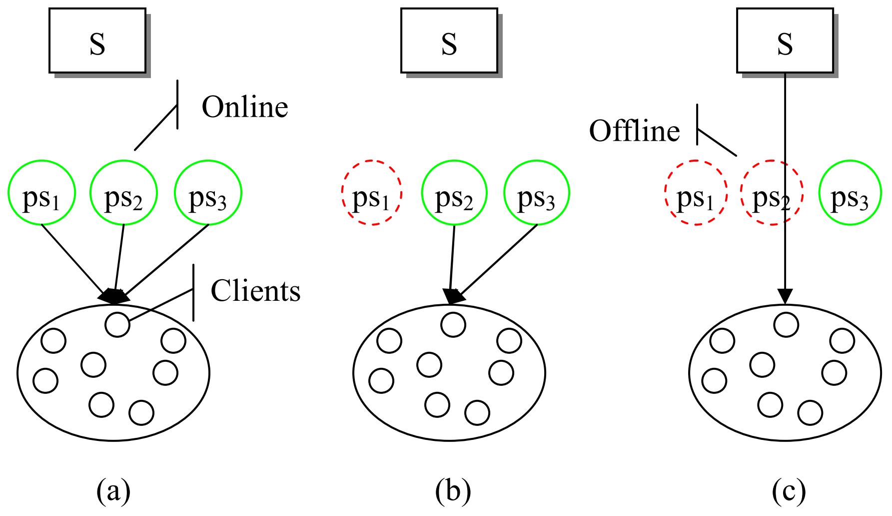


Figure 6.18. Idea of Video-Suffix Delivery Using ECC

all symbols over the network. The system only is required to select M out of H PSs to broadcast the symbols. Therefore, there is no additional overhead for transmission in teams of bandwidth usage.

N PSs comprising a complete ECC(N, H) codeword and handling one or more video sessions form a PSG. One of the online PSs will be selected as a group leader (GL) which has the same duty of ASP. As shown in Figure 6.18, there are three PSs forming a PSG with a ECC(*3, 1*) codeword. Therefore, the PSG can provide the service without disruption when any two of the PSs in the PSG is online. As shown in Figure 6.18(b), ps1 departs after some time has elapsed. Since the clients can still receive 2 blocks out of 3 blocks from the PSG, the service is kept going on. However, if one more PSs (i.e. ps2) for this video leave, the clients cannot reconstruct the video correctly. Thus, the central server should take over the service in order to avoid disruption of service (see Figure 6.18(c)). It should be noticed that all online PSs in a PSG should suspend their transmission when the central server is handling the service. Once any one of the PSs in the PSG is online again, the central server will return the duty to this PSG. In general, the central server provides the service to clients only when less than *M* PSs in the PGS are online. Therefore, in order to evaluate the workload of the central server in this policy, we should determine the relationship between ECC(*N, M*) and the bandwidth requirement of the central server for each video which will be described in Section 6.5.2.

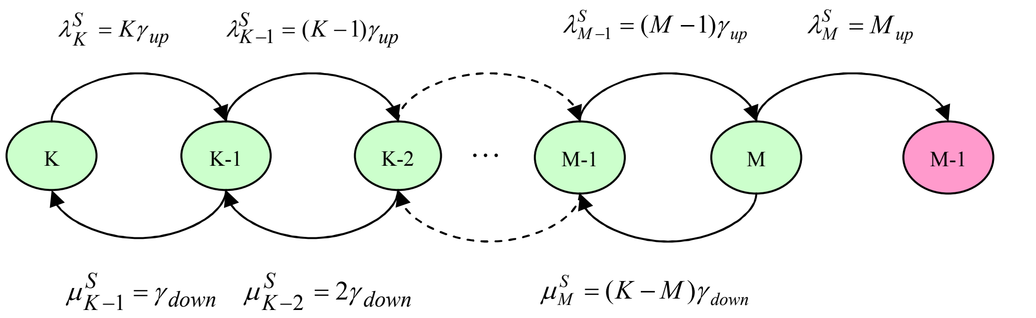
6.5.2 Reliability Modeling

Since the number of broadcast channels required for the system is S_m, the number of PSG using ECC for the video (J^{ECC}) can be expressed by $J^{ECC} = \lceil \eta S_m / NB \rceil$ if each PS can contribute *B* video channels[7], where *B>0*. Therefore, the number of video sessions handled by PSG j (G_j) can be expressed by

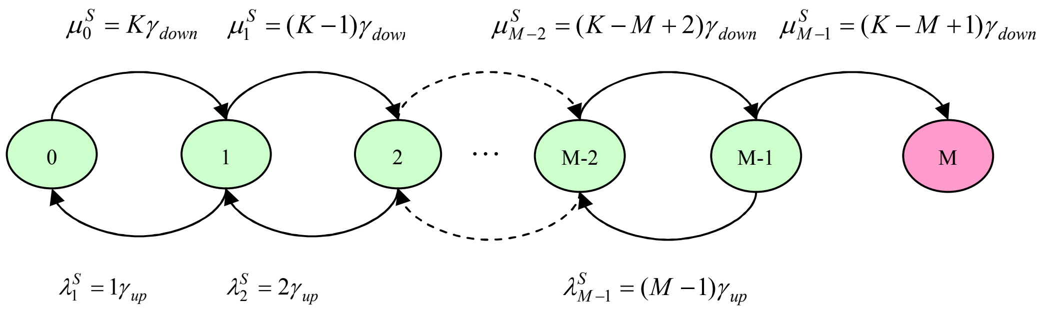
$$G_j = \begin{cases} \min(S_m, \lfloor MB \rfloor) & , j = 1,2,3,...,J-1 \\ \min(S_m, \lfloor MB \rfloor)\% M & , j = J \end{cases}$$

Again, denote C_j as the set of the assigned video sessions for PSG *j*, where *j=1,2,3…,J*. As mentioned before, similar to the replication approach, the central server takes over the service only when more than *M* PSs in the PSG are offline and it returns the duty to that PSG when more than *M* PSs in the group is online. Because C_j is served alternatively by

[7] To simplify our analysis, we assume that each PS has unlimited local storage space.



(a) Peer Server Group using ECC



(b) Central Server

Figure 6.19. State-Space Diagram

PSG *j* and the central server, it also forms an alternating renewal process. Again, let Z_1^j, Z_2^j, …be the successive serving time of PSG *j* for C_j and also let D_1^j, D_2^j, …as the corresponding successive serving time of the central server. Obviously, there is a relationship between the ECC(*N*,*M*) codeword and the bandwidth requirement of the central server. To determine the mean serving time of PSG *j* using ECC (Z_j^{ECC}) with N_j PSs, we use another set of CTMC to model the PSG using ECC and the state-space diagram for this model is shown in Figure 6.19(a). It is assumed that all PSs are independent to each others and all PSGs also operate independently to other PSGs in the system. Thus, we can simply consider one particular PSG with *N* PSs. When the system is at state *N*, all PSs are available for broadcasting. State *M-1* here shows that the PSG does not have sufficient PSs to sustain the service and is assumed to be an absorbing state. We first define $P_k^{ECC}(t)$ to be the probability of the PSG using ECC in state *k* at time *t* (*k=0, 1, 2, 3, ... K*). From Figure 6.19(a), the state equations of the CTMC are given by

$$\begin{aligned}
&\dot{P}_k^{ECC}(t) = -(\lambda_k^s + \mu_k^s)P_k^{ECC}(t) + \mu_{k-1}^s P_k^{ECC}(t) + \lambda_{k+1}^s P_{k+1}^{ECC}(t) && ,k \geq M \\
&\dot{P}_{M-1}^{ECC}(t) = \lambda_1^s P_{M-1}^{ECC}(t) && ,k = M-1
\end{aligned}$$

(6.20)

Since the transition rate matrix does not have full rank, we can remove the equation of *k=M-1* without loosing any information [HOYLA94]. Therefore, we obtain

$$\dot{P}_k^{ECC}(t) = -(\lambda_k^s + \mu_k^s)P_k^{ECC}(t) + \mu_{k-1}^s P_{k-1}^{ECC}(t) + \lambda_{k+1}^s P_{k+1}^{ECC}(t) \qquad ,k \geq M$$

(6.21)

By introducing the Laplace transform of the eqn.(6.21), we have

$$\begin{aligned}
&s\overset{*}{P}_k^{ECC}(s) - P_k^{ECC}(s) = \\
&-(\lambda_k^s + \mu_k^s)\overset{*}{P}_k^{ECC}(s) + \mu_{k-1}^s \overset{*}{P}_{k-1}^{ECC}(s) + \lambda_{k+1}^s \overset{*}{P}_{k+1}^{ECC}(s) \qquad ,k \geq M
\end{aligned}$$

(6.22)

In the proposed architecture, the PSG will take over the duty when *M* PSs in the group is online. Hence, the initial state of the PSG at time *t=0* is defined as

$$\begin{aligned}
&P_M^{ECC}(0) = 1 \\
&P_k^{ECC}(0) = 0 \quad for\ k \neq M
\end{aligned}$$

By inserting $s=0$ in eqn.(6.22), we obtain

$$-(\lambda_k^s+\mu_k^s)\overset{*}{P}{}_k^{ECC}(0)+\mu_{k-1}^s\overset{*}{P}{}_{k-1}^{ECC}(0)+\lambda_{k+1}^s\overset{*}{P}{}_{k+1}^{ECC}(0)=-1 \quad ,k=M$$
$$-(\lambda_k^s+\mu_k^s)\overset{*}{P}{}_k^{ECC}(0)+\mu_{k-1}^s\overset{*}{P}{}_{k-1}^{ECC}(0)+\lambda_{k+1}^s\overset{*}{P}{}_{k+1}^{ECC}(0)=0 \quad ,k>M$$

(6.23)

Because the expected time to reach state *M-1* from state *M* is equal to $1/\lambda_M^S$, eqn.(6.23) can be solved for $\overset{*}{P}{}_k^{ECC}(0)$ for $k \ge M$. Therefore, with eqn.(6.14), the mean serving time of PSG *j* using ECC (Z_j^{ECC}) with N_j PSs can be expressed as:

$$Z_j^{ECC}(N_j)=\frac{1}{\lambda_M^S}+\sum_{i=0}^{N_j-M-1}\left(\frac{\prod_{j=0}^{i}\mu_{M+j}^S}{\prod_{k=0}^{i+1}\lambda_{M+k}^S}\right) \tag{6.24}$$

To compute the corresponding mean serving time of the central server (D_j^{ECC}), we use another CTMC model as shown in Figure 6.19(b). When the system is at state *0, 1, 2, ..., M-1*, the central server is taking over the broadcasting operation. Once there are *M* PSs online, the broadcasting operation is returned to this PSG. Therefore, using the similar deviation from eqn.(6.20) to eqn.(6.23), D_j^{ECC} can be computed by

$$D_j^{ECC}(N_j)=\frac{1}{\mu_{M-1}^S}+\sum_{i=0}^{M-2}\left(\frac{\prod_{j=0}^{i}\lambda_{M-1-j}^S}{\prod_{k=0}^{i+1}\mu_{M-1-k}^S}\right) \tag{6.25}$$

Then, the average bandwidth reserved by the central server for PSG *j* is given by

$$B_S^j(N_j)=\frac{G_jR\cdot D_j^{ECC}(N_j)}{D_j^{ECC}(N_j)+Z_j^{ECC}(N_j)} \tag{6.26}$$

Then, the overall bandwidth required for the central server (B_S^{ECC}) is thus computed by

$$B_S^{ECC} = \sum_{j=1}^{J^{ECC}} B_S^j(N_j) \tag{6.27}$$

It should be noticed that eqn.(6.24) and eqn.(6.25) will be reduced to eqn.(6.15) and eqn.(6.16) when M is equal to 1. Similar to the replication approach, eqn.(6.19) can be used to determine the minimum number of PSs required for the system in order to achieve the desired bandwidth requirement of the central server.

6.5.2 Results

In this section, we evaluate the performance of the proposed protocol. Computer simulation is also performed to verify the correctness of the model. Again, it is first assumed that the video length is 7200 seconds long and R is equal to 1. The startup delay of the system (W) is 300 seconds such that there are 24 video sessions (i.e. $S_m=24$). Unless other specified, the availability of the PS is set as 0.6. For simplicity, we also assume that each PS with $\gamma_{up}=36000A$ seconds and $\gamma_{down}=36000(1-A)$ seconds contributes a bandwidth of R. Therefore, the online/offline time of each PS is ranged between 1 to 10 hours governed by A.

We first investigate how the number of PSs affects the system performance. Figure 6.20 plots the total server bandwidth versus the number of PSs per PSG. The number of PSs in each PSG is determined by the value of M as well as the redundant overhead O. For example, if the value of M is 5 and O is 0.2, the number of PSs per PSG is 6. It is found

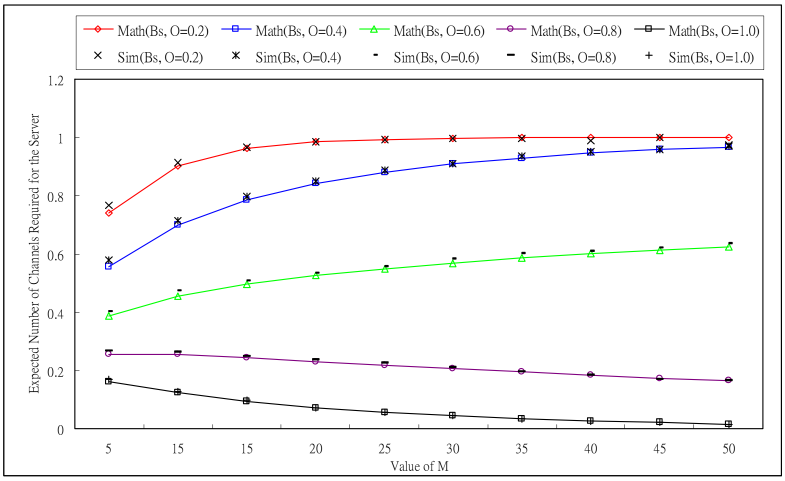


Figure 6.20. Total Server Bandwidth Requirement against the number of PSs per PSG

that the mathematical model closely matches with the simulation results. From the result, it can be first found that the bandwidth requirement of the central server is decreasing when O is increased for all M. In case $M=5$ and $O=0.2$, B_s is equal to 18. If O is increased to 1.0, B_s can be reduced to about 4 channels. As the redundant of the PSG is increased (i.e. O is increased), the lifetime (or Z^j) of each PSG is extended. Therefore, the duration of the central server taking over the broadcasting operation is reduced and thus utilizes fewer server resources. On the other hand, it is found from the result that the bandwidth requirement of the central server is increasing when O is low. But, that is decreasing when it is high. When the redundant is low (e.g. $O=0.2$), Z^j is decreasing with the increase of M but D^j is increased. However, when the redundant is high (e.g. $O=1.0$), the situation is opposed that Z^j is increasing with the increase of M but D^j is decreased.

Then, we look at the system performance against the availability of the PS. Figure

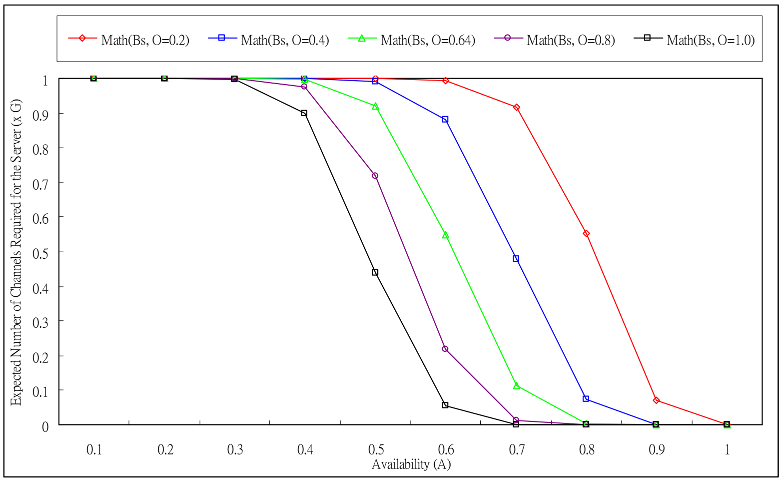


Figure 6.21. Total Server Bandwidth Requirement against the Availability of PS

6.21 shows the bandwidth requirement of the central server when the availability of the PS is changed where *M=25*. As we expected, B_s is sharply decreasing when the availability of each PS is increased. For the same A, the performance can be also improved when *O* is increased. From the results, it can be seen that the system performance cannot get any benefit from the proposed policy when the availability of the PS is less than 0.4. It is because Z^j is short when A is low and thus the central server has to take over the broadcasting operation frequently. One possible solution to compensate this flaw is to further increase the redundant overhead. However, it can be found that the central server only requires to allocate less than 80% resources compared with the original SB for various value of *O* when *A>0.5*.

Finally, we are going to investigate the number of PSs required for the system in order to achieve certain level of reduction of the central server resources compared to the

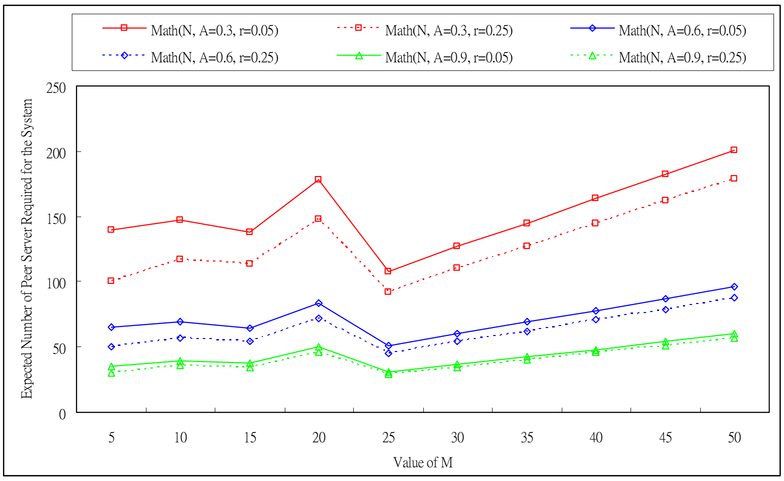


Figure 6.22. Total Number of PSs required for the System against the Value of M

original SB. As shown in Figure 6.22, the number of PSs required for the system is decreased when the availability of PS is increased for all M. For the same M and A, it can be seen that the system with $r=0.05$ requires more PSs than the system with $r=0.25$. However, we can see that the performance with different r does not have significant difference when the availability is increasing. It is because Z^j can be extended longer by each additional PS. From the results, the workload of the central server can be reduced by 95% compared with the original SB when there are 51 PSs with the availability of 0.6 where M is 25. On the other hand, it is interesting to observe that the number of PSs required for system is irregularly changed when M is increasing with the same r and A. It is because their number of PSG required for the system is not the same for different M. With the same group size, the number of PSs required for system is increasing with the increase of M. For example, the case of $M \geq 25$ has the same PSG size of 1. It should be reminded that the requirement of the PS in terms of bandwidth and storage is decreasing when M is increased.

6.6 Summary

In this chapter, we develop a hybrid VoD system framework that uses a P2P paradigm coupled with video broadcasting and CDN-like approach for video delivery. In the proposed architecture, the video is first partitioned into two parts. The first part of the video is transmitted among customers in P2P manner while the second part of the video is broadcasted periodically by the peer server. In order to avoid the disruption of the service caused by the unpredictable departure/failure of peers, a central server is still deployed. This content delivery strategy allows the workload of the system disperse over the network while customers can still guarantee to obtain the service without collapsing by the dynamic nature of P2P framework. In addition, a fault exception mechanism based on replication and error correcting codes are deployed in peer servers. Analytical model is also developed to allow system designers better understanding of system dynamics and provide guidelines for the management of design resources and realization of VoD services based on this architecture. It is noted from the results that, using the replication approach, the workload of the central server can be reduced by 95% when the system arrival rate is 0.1 req/s and there are 156 peer servers are deployed, each of which has the availability of 0.4.

Chapter 7

Conclusion

7.1 Conclusion of this Dissertation

With the advances in digital video technology and high speed networking, video-on-demand (VoD) services have come into practice in recent years. The VoD service allows geographically distributed users to interactively access video files from remote VoD servers. Users can request videos from a server at any time and enjoy the flexible control of video playback. Nevertheless, such systems have not yet been commercial success because deployment of a large-scale VoD system requires an enormous amount of server and network resources. Therefore, one of the most challenging aspects of a VoD design is how to deliver videos to a large number of clients economically. On the other hand, heterogeneity of the network environment is another design issue that should be addressed in order to meet the different capability of the clients' devices in the system. In this dissertation, we focused on investigating a feasible solution for building a cost-effective and scalable VoD system in heterogeneous network environment. We have made a number of contributions in various system design issues including the development of an analytical model to efficiently compare the replication coding with the layered coding for video transmission in heterogeneous environment based on the hierarchical VoD framework. According to the findings of this model, we then explore the complementary coding approach to further improve the system performance. In this dissertation, another unified model is also developed that exploits the multicast/ broadcast capability of the network and P2P paradigm to efficiently deliver video data to the clients. With these models, we can have a better understanding of the system dynamics. Such models also provide guidelines for the management of system resources and realization of VoD services based on the proposed frameworks.

In this thesis, we first develop a unified model to explore the impact of the broadcasting schemes coupled with proxy caching for video transmission in hierarchical network architecture. This model also considers using the replication or layering approach to create multiple qualities of video streams to meet different requirements of clients. Using this model, we have first found that the proxy size, the efficiency of the broadcasting scheme, the bandwidth reserved for broadcasting as well as the layering overhead have significant impacts on the system performance. Based on this model, we also analytically compare the performance of the video replication approach with that of layering for video streaming under different scenarios and parameter settings. From the result, it can be concluded that the layering approach is suitable for proxy caching and video broadcasting while replication is favorable to end-to-end transmission.

This finding motivates us to examine whether the system performance can be further improved if both coding schemes are deployed in different levels of the system based on their natures. Therefore, we propose a complementary coding scheme using both video replication and layering for video streaming (i.e. the layering approach is used for proxy caching and video broadcasting with replication is used for end-to-end transmission). In addition, we also analytically explore the benefit of renegotiation about video quality when the system resources cannot support the requested quality levels. From the results, we found that the system performance can have a further enhancement up to 15% when the complementary coding scheme and renegotiation mechanism are used.

Although the hierarchical architecture approach can greatly improve the system performance, such client-server architecture does still not scale well because the bottleneck of the system is still at the server side. Thus, we turn our focus to P2P paradigm to address the issues of system scalability. However, most of the traditional approaches were designed for streaming applications in a unicast infrastructure. As the successful deployment of IP multicast/broadcast delivery, it is believed that the system could have a further improvement in terms of cost effective and scalability when the broadcasting scheme can be coupled with P2P paradigm in a suitable manner. Thus, a new batching policy called peer-to-peer batching (PPB) has been proposed. The objective of this policy is to consider the trade-off between the network bandwidth requirement for P2P transmission and multicast delivery such that the overall transmission cost is minimized. It is found that this policy can leverage the workload of the central server about 50%. In addition, the result also shows that there is a strong relationship among the batching time, the arrival rate and the departure rate, which bring a great impact on the system performance.

To further reduce the server's workload, we develop another unified model that uses a P2P paradigm coupled with video broadcasting for video delivery. Similar to CDN-P2P, a central server is still deployed in order to avoid the disruption of the service caused by the dynamic nature of P2P applications. Unlike PPB that the delivery of leading portion of the video should be accomplished in chaining manner and each peer should have identical bandwidth, this framework allows peer to select a set of peers with different bandwidth capacity for video transmission. On the other hand, while PPB is required that video broadcasting should be done by the central server, the new framework disperses the duty of video broadcasting among a number of peer servers. In order to increase the system reliability, a fault tolerant mechanism based on replication as well as erasure correcting codes are applied to the peer server. In addition, the relationship between the number of peer servers required and the bandwidth requirement of the central server are also investigated. In the proposed framework, with the engagement of the peer servers and the contribution of the normal peers, the workload of the system can be highly dispersed along the network and customers can also be guarantee to obtain the service without collapsing by the dynamic nature of P2P framework given that the finite server resources. From the results, using the replication approach, the workload of the central server can be reduced by 95% when the system arrival rate is 0.1 req/s and there are 156 peer servers are deployed, each of which has the availability of 0.4.

7.2 Future Directions

Robustness, reliability, scalability, ability to deal with heterogeneity, real-time or low latency service, security and interactivity with the clients are all important and desired properties of the T-VoD service. In this dissertation, the first five of these properties have been already considered.

Since video contents are mainly distributed by the self-proliferation of peers in P2P applications, customers may not contact the central agent of the system directly during watching the video. Security is a very important design issue in P2P applications. In order to avoid the illegal duplication of video contents as well as unpaid users admitted to the system, a system should provide efficient support for copyright protection in transmitting video streams to multiple peers. To allow a user to have a control right for a particular video session, VoD with VCR like features should also be supported. Users can request and view any video at any time with full VCR capabilities including forward, reverse, freeze and random positioning.

On the other hand, other than video replication and layering encoding, Multiple Description Coding (MDC) is another video coding technique that can be used to support multiple qualities of videos to clients. It also provides error resilience to media streams such that it can resist the network congestion as well as packet loss which are common in best-effort networks such as the Internet. In addition, our performance models are mainly developed for constant-bit-rate (CBR) videos. Therefore, it is also worth to study the proposed architecture with variable-bit-rate (VBR) videos as well as MDC in the future development.

Besides delivering video data over the wired network environment, video transmission over wireless environment has also brought a great attention recently. Video call is one of the most common mobile video streaming applications, where users do not only hear the call partner’s voice as a conventional voice phone call but also can see the call partner’s activities through the display. However, there are a number of technical challenges in providing video services over wireless environment. One of these is the unreliable nature of wireless channels. Wireless links are error-prone and varying connection quality in nature, which results in burst packet corruptions which bring a great impact on transmitted video quality. The other arises from the heterogeneity of end-to-end systems with different capabilities such as processing power and bandwidth limitations. In order to tackle these problems, scalable video coding (SVC) as well as network coding (NC) have been developed.

The new scalable video coding (SVC) extension of H.264/AVC standard [SCHWA07] provides network-friendly scalability at a bit stream level with a moderate increase in decoder complexity compared to single-layer H.264/AVC. It supports functionalities such as bit rate, format, and power adaptation, graceful degradation in lossy transmission environments as well as lossless rewriting of quality-scalable SVC bit streams to single-layer H.264/AVC bit streams. These functionalities provide better performance to streaming and storage applications. SVC has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to the scalable profiles of prior video coding standards.

In conventional video multicast, video data are carried by store-and-forward mechanisms with the FEC protection and the intermediate nodes in the system forward an exact copy of video data what they have received. A network coding (NC) [AHLSW00] theory has been developed which provides alternative approach that encodes packets at intermediate nodes. There are several advantages of NC for video multicast. First, it enhances the throughput of the network. Seconds, robustness of video transmission can be easily achieved as compared with erasure coding in erroneous wireless channel. Third, it also simplifies the construction of multicast tree and routing which use broadcasting to construct multiple paths. Therefore, it can improve the overall received video quality significantly in wireless environment.

References

[AGGAR96a] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optimal batching policies for video-on-demand storage servers," *in Proc. IEEE Int. Conf. Multimedia Computing and Systems,* pp. 253-258, 1996.

[AGGAR96b] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," In Proc. of the IEEE Int'l conf. Multimedia Computing and Systems '96, Hiroshima, Japan, June 1996.

[AGARW06] V. Agarwal and R. Rejaie, "Adaptive Receiver-Driven Streaming from Multiple Senders," *Proceedings of ACM Multimedia Systems Journal*, Volume 11, Issue 6, page 1-18, Springer-Verlag, April 2006.

[AHLSW00] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[ARVIN94] K. Arvind, "Probabilistic Clock Synchronization in Distributed Systems," *IEEE Trans. On Parallel and Distributed Systems*, vol.5 no.5, pp. 474-487, May 1994.

[BANSA01] D. Bansal and H. Balakrishnan, “Bionomial congestion control algorithms”, *In Proceedings INFOCOM 2001*, April 2001.

[BERME96] T. Bermers-Lee, R. Fielding and H. Frystyk, “Hypertext Transfer Protocol -- HTTP/1.0,” RFC 1945, Network Working Group, May 1996.

[CAI99] Y. Cai, K.A. Hua and K. Vu, "Optimizing Patching Performance,” *Proc. IS&T/SPIE Conf. on multimedia Computing and Networking 1999 (MMCN '99)*, San Jose, CA, pp. 204-215, Jan. 1999.

[CHAN01] G. S. H. Chan and F. Tobagi, “Distributed Servers Architecture for Networked Video Services”, *IEEE/ACM Transactions on Networking*, vol. 9, No. 2, pp. 125-136, April 2001.

[CHEN04] S. Chen *et al*., “Designs of High Quality Streaming Proxy Systems,” *Proc. IEEE INFOCOM’04*, Hong Kong, Chain, Mar, 2004.

[CUI03] Yi Cui and Klara Nahrstedt, “Layered Peer-to-Peer Streaming”, *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Monterey, CA, June, 2003.

[DAN94] A. Dan, D. Sitaram and P. Shahabuddin, “Scheduling policies for an on-demand video server with batching,” *IBM Research Report RC*

20206, T.J. Watson Research Center, Yorktown Heights, NY, 1994.

[DAN95] A. Dan, P. Ahahabuddin, D. Sitaram and D. Towsley, "Channel allocation under batching and VCR control in video-on-demand systems," *Journal of Parallel and Distributed Computing*, vol. 30, no. 2, pp.168-179, Nov. 1995.

[EAGER01] Eager, M. Vernon and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, issue 5, pp. 742-757, Sept.-Oct. 2001.

[FLOYD99] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in Internet", *IEEE/ ACM Trans. On Networking*, Vol. &., No. 7, August 1999.

[FLOYD00] Floyd, S., Handley, M., Padhye, J., and Widmer, J. "Equation-based congestion control for unicast applications", *In Proceedings of the 2000 ACM SIGCOMM Annual Technical Conference*, August 2000.

[GAO99] L. Gao and D. Towsley, "Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast," *Proc. 1999 IEEE Int'l. Conf. On Multimedia Computing and Systems (ICMCS'99)*, Florence, Italy, vol.2, pp. 117 – 121, June 1999.

[GANES03] A. J. Ganesh, A.-M. Kermearrec and L. Massoulie, "Peer-to-Peer Membership Management for Gossip-based Protocols," *IEEE Transactions on Computers*, 52(2), Feb. 2003.

[GOLUB96] L. Golubchik, J. C. S. Lui, and R. R. Muntz, "Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers", *Multimedia Systems*, 4, (3), pp. 140-155, 1996.

[GUO01] C. Guo, Z. Xiang, Y. Zhong and J. Long, "Optimal caching algorithm based on dynamic programming", *Proceedings of SPIE International Symposium on Convergence of IT and Communications (ITCom'01)*, vol. 4519, pp. 285-295, 2001.

[GUO03a] Y. Guo, K. Suh, J. Kurose and D. Towsley, "A peer-to-peer on-demand streaming service and its performance evaluation," *in Proc.IEEE Int. Conf. Multimedia and Expo (ICME'03),* pp. 649-652, 2003.

[GUO03b] Y. Guo, K. Suh, J. Kurose and D. Towsley, "P2Cast: Peer-to-Peer Patching Scheme for Vod Service," *Proc. of the 12th World Wide Web Conference (WWW-03)*, Budapest, Hungary, May, 2003

[HASKE97] B.G. Haskell, "Digital Video: An introduction to MPEG-2,: Chapman & Hall, New York, 1997.

[HUA97] K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems", *Proceedings of SIGCOMM 97*, pp. 98-100, 1997.

[HUA98] K. A. Hua, Y. Cai and S. Sheu, "Patching: A multicast technique for true video-on-demand services", *Proceedings of the 6th ACM International Conference on Multimedia*, pp. 191-200, 1998.

[HUA00] K. A. Hua, D. A. Tran, and R. Villafane, "Caching multicast protocol for on-demand video delivery," in Proc/ ACMSPIE Conf. Multimedia Computing and Networking (MMCN 2000), pp. 2-13, 2000.

[HUA04] K.A. Hua, M.A. Tantaoui and W. Tavanapong, "Video Delivery Technologies for Large-Scale Deployment of Multimedia Applications," *In Proc. of The IEEE*, vol. 92, no. 9, pp. 1439-1451, Sept. 2004.

[HARTA02] Felix Hartanto, Jussi Kangasharju, Martin Reisslein, and Keith W. Ross, "Caching Video Objects: Layers vs. Versions," *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '02)*, vol. 2, pages 45-48, Lausanne, Switzerland, August 2002.

[HEFEE03] M. Hefeeda, A. Habib, B. Boyan, D. Xu and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast.," *Technical report*, CS-TR 03-016, Purdue University, Aug. 2003.

[HOYLA94] A. Hoyland and M. Rausand, "System Reliability Theory: Models and Statistical Methods," *John Wiley and Sons*, vol. 518, 1994.

[ISO93] ISO/IEC, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbps," *ISO/IEC 11172*, 1993.

[ISO96a] ISO/IEC and ITU-T, "Information technology – Generic coding of moving pictures and associated autio – Part 1: Systems," *ISO/IEC 13818-1*, *ITU-T Recommendation H.222.0*, 1996.

[ISO96b] ISO/IEC and ITU-T, "Information technology – Generic coding of moving pictures and associated autio – Part 2:Video," *ISO/IEC 13818-2*, *ITU-T Recommendation H.262*, 1996.

[ISO00] ISO/IEC, "Coding of Audio-Visual Objects, Part-2 Visual, Amendment 4: Streaming Video Profile," *ISO/IEC 14 496-2/FPDAM4*, July 2000.

[JUHN97] L. Juhn and L. Tseng, "Harmonic broadcasting protocols for video-on-demand service", *IEEE Trans. on Broadcasting*, vol. 43, no. 3, pp. 268-271, 1997

[JIANG98] T. Jiang, M. H. Ammar and E. W. Zegura, "Inter-receiver fairness: a novel performance measure for multicast ABR sessions," *Proceedings of ACM SIGMETRICS' 98*, Madison, WI, pp. 202-211, June, 1998.

[KANGA02] J. Kangasharju, F. Hartanto, M. Reisslein and K.W. Ross, "Distributing layered encoded video through caches," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 622–636, 2002.

[KIMUR99] J. I. Kimura, F. A. Tobagi, J. M. Pulido and P. J. Emstad, "Perceived quality and bandwidth characterization of layered MPEG-2 video encoding," *Proceedings of SPIE Internation Symposium Voice, Video and Data Communications*, Boston, MA, Sep. 1999.

[KIM01] T. Kim and M. H. Ammar, "A comparison of layering and stream replication video multicast schemes," *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, New York, pp.63-72, 2001.

[KIM04] Y.G. Kim, J.W. Kim, C.-C.J. Kuo, "TCP-friendly Internet video with smooth and fast rate adaptation and network-aware error control", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.14, Issue: 2, pp. 256-268, Feb. 2004.

[KUSMI06] E. Kusmierek, Y. Dong, and D. Du, "Loopback: Exploiting Collaborative Clients for Large-Scale Streaming," *IEEE Transactions on Multimedi*a, vol. 8, issues 2, pp. 233-242, April 2006.

[LI96] V.O.K. K. Li, W.J. Liao, X. X. Qiu, and W. M. Wong, "Performance model of interactive video-on-demand systems", *IEEE Journal Selected*

Areas in Communication, vol.14, no.6, pp.1099-1109, Aug. 1996.

[LI01] W.P. Li, "Overview of fine granularity scalability in MPEG-4 video standard." *IEEE Trans. Circuit and System for Video Technology*, vol. 11, no. 3, pp. 301-317, Mar. 2001.

[LIAO97] W.J. Liao, and V.O.K. Li, "The Split and Merge (SAM) Protocol for Interactive Video on Demand systems," *IEEE Trans. on Multimedia*, vol. 4, no. 4 pp. 51-62, Oct-Dec 1997.

[LIU01] J. C. L. Liu and D. H. C. Du, "Continuous media on demand", *IEEE Computer*, vol. 34, no. 9, pp.37-39, Sep. 2001

[LIU03] W. C. Liu and Jack Y. B. Lee, "Constrained Consonant Broadcasting - A Generalized Periodic Broadcasting Scheme for Large Scale Video Streaming," *Proceedings IEEE International Conference on Multimedia & Expo 2003*, Baltimore, U.S.A., July 6-9, 2003.

[LIU04] J.C. Liu and J.L. Xu, "Proxy caching for media streaming over the Internet," *IEEE Communications Magazine*, vol. 42, issue 8, pp. 88-94, Aug. 2004.

[LU00] G. Lu, "Issues and technologies for supporting multimedia communications over the Internet," *Computer Communications*, vol.23, pp.1323-1335, Aug. 2000.

[LITTL96] T. D. C. Little and D. Venkatesh, “Prospects for interactive video on demand,” *IEEE J. Select. Areas Commun.*, vol. 14, pp.1099-1109, Aug. 1996.

[MCAUL90] A. J. McAuley, “Reliable Broadband Communication Using a Burst Erasure Correcting Code,” *Proceeding of the ACM SIGCOMM 90*, Philadelphia, PA, pp. 287-306, Sept. 1990.

[MEDHI94] J. Medhi, Stochastic Process, Second Edition, New York, Wiley, pp. 439-440, 1994.

[MA02] H. Ma and K. G. Shin, “Multicast video-on-demand services,” *ACM Commun. Rev.*, 00. 31-42, Jan. 2002.

[NGUYE02] T. Nguyen, A. Zakhor, “Distributed video streaming”, *in Proc. SPIE-The International Society for Optical Engineering, Multimedia Computing and Networking (MMCN)*, vol 4673, San Jose, CA, pp. 186-195, Jan. 2002.

[NGUYE04] T. Nguyen, A. Zakhor, “Multiple Sender Distributed Video Streaming”, *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 315-326, April 2004.

[NICOLO03] A. Nicolosi and S. Annapureddy, “P2PCast: A peer-to-peer multicast scheme for streaming data,” *First IRIS Student Workshop(ISW’03)*, Available at: http://www.cs.nyu.edu/~nicolosi/papers/P2PCast.ps, 2003.

[PADHY98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling "TCP throughput: A simple model and its empirical validation". *Proceedings of SIGCOMM '98*, 1998.

[PRABH97] N.U. Prabhu, Foundations of Queueing Theory, pp.43-56, Kluwer Academic Publishers, 1997.

[PARIS98a] J. F. Paris, S. W. Carter, and D. D. E. Long, "Efficient broadcasting protocols for video on demand," *In 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, pp 127-132, July 1998.

[PARIS98b] J. F. Paris, S. W. Carter, and D. D. E. Long, "A low bandwidth broadcasting protocol for video on demand, " *In Proc. of IEEE Int'l Conference on Computer Communications and Networks (IC3N'98)*, 1998

[PARLA00] M. Parlar, "Interactive operations research with Maple: methods and models," Boston: Birkhäuser, 2000.

[POON00] W.F. Poon, K.T. Lo and J. Feng, "Batching policy for video-on-demand in multicast environment," *Electronics Letters*, vol.36, pp.1329-1330, July 2000.

[POON03] W.F. Poon, J. Feng and K.T. Lo, "Video data broadcast protocol for video on demand." *IEICE Trans. Communications*, vol.E86, no.8, pp.273-275, Aug. 2003.

[REJAI00] R. Rejaie, M. Handley and D. Estrin, "Layered Quality Adaptation for Internet Video Streaming", *IEEE Journal on Selected Areas in Communications*, vol 18, Issue 12, pp. 2530-2543, Dec. 2000.

[REJAI01] R. Rejaie, J. Kangasharju, "Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming", *11th International workshop on Network and Operating Systems support for digital audio and video*, pp. 3-10, Jan. 2001.

[ROWST01] A. Rowstron and P. Druchel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware), pp. 329-350, 2001.

[SCHUL98] H.Schulzrinne, A. Rao and R. Lanphier, "Real-time streaming protocol (RTSP)," IETF, RFC 2316, Arp. 1998.

[SCHWA07] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, " *IEEE Trans on Circuits and Systems for Video Technology*, vol. 17, no. 9 pp.

1103-1120, Sept. 2007

[SEN99] S. Sen, J. Rexford and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," *Proc. IEEE INFOCOM'99*, New York, NY, Mar. 1999.

[SERPA00] D.N. Serpanos and A. Bouloutas, "Centralized versus distributed multimedia servers", *IEEE Transactions on Circuit Systems and Video Technology*, vol. 10, no. 8, pp. 1438-1449, Dec. 2000.

[SHEU97] S. Sheu and K. Hua and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," *in Proc. IEEE Int. Conf. Multimedia Computing and Systems'97*, CA, pp. 110-117, 1997.

[SINCO91] W.D. Sincokie, "System architecture for a large scale video on demand Service," *Computer Networks and ISDN Systems*, vol.22, pp.155-162, Sept. 1991.

[SRIPA04] K. Sripanidkulchai et al., "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," *Proc. ACM SIGCOMM*, ACM Press, pp.107-120, 2004.

[STOIC03] I. Stoica et al.,"Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Apllications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp.

17-32, 2003.

[TAN02] H. Tan, D. L. Eager and M. K. Vernon, "Delimiting the range of effectiveness of scalable on-demand streaming", *Proceedings of IFIP International Symposium on Computer Performance Modeling*, Measurement and Evaluation (Performance 2002), Rome, pp. 387-410, Sept. 2002.

[TEWAR98] R. Tewari et al., "Resource-Based Caching for Web Servers," *Proc. MMCN'98*, San Jose, CA, Jan. 1998.

[THOUI07] F. Thouin and M. Coates, "Video-on-Demand Networks: Design Approaches and Future Challenges," *IEEE Network*, vol. 21, issue 2, pp. 42-48, Mar. 2007.

[TO05] K.-K. To, J. Lee and S.-H. Chan, "Decentralized Periodic Broadcasting for Large-Scale Video Streaming," *in Proceedings of International Conference on Multimedia & Expo (ICME)*, pp. 285-288, Amsterdam, The Netherlands, 6-8 July, 2005.

[TRAN04] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE Journal on Selected Areas in Communications*, Vol.22, Issue 1, pp. 121-133, Jan. 2004.

[TU05] Y.C. Tu, J.Z. Sun, M. Hefeeda and S. Prabhakar, "An Analytical Study of Peer-to-Peer Media Streaming Systems," *ACM Trans. Multimedia Computing, Communication and Application*, vol. 1, no. 4, pp. 354-376, Nov. 2005.

[VISWA96] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Syst.*, vol. 4, no. 4, pp. 197-208, Aug. 1996.

[WONG88] J.W. Wong, "Broadcast delivery", *Proceedings of IEEE*, 76(12), pp. 1566-1577, 1988.

[WANG04] B. Wang, S. Sen, M. Adler and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 366-374, April 2004.

[TANG04] W.K.S. Tang, E.W.M. Wong, S. Chan and K.T. Ko, "Optimal video placement scheme for batching VOD services," *IEEE Transactions on Broadcasting*, vol. 51, no. 1, pp. 16-25, Mar. 2004.

[XU02] D. Xu, M. Hefeeda, S. Hambrusch and B. Bhargave, "On Peer-to-Peer Media Streaming", *in proceedings of ICDCS2002*, vol. 1, (Vienna), pp. 363-371, July 2002.

[XU06] D. Xu, S. Kulkarni, C. Rosenberg, H.K. Chai, "Analysis of a CDN-P2P Hybrid Architecture for Cost-Effective Streaming Distribution", *ACM/Springer Multimedia Systems Journal*, vol. 11, no. 4, 2006.

[YAN03] E. M. Yan and T. Kameda, “An efficient VoD broadcasting scheme with user bandwidth limit”, *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking*, vol. 5019, pp. 200-208, Santa Clara, CA, Jan. 2003.

[YANG05] X. Yang, P. Hern´andez, F. Cores, A. Ripoll, R. Suppi, and E. Luque, “Distributed p2p merging policy to decentralize the multicasting delivery.,” *In 31st EuroMicro Conference on Software Engineering and Advanced Applications*, pp. 322–329, Porto, Portugal, Aug.-Sept. 2005.

[ZHANG00] Z.L. Zhang *et. al*, “Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks,” *IEEE/ACM Trans. Net.*, vol. 8, no. 9, pp. 429-442, 2000.

[ZHANG05] X.Y Zhang, J.C. Liu, B.Li and Y.-S.P. Yum, “CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming,” *INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE*, vol. 3, pp. 2102 – 2111, 13-17 Mar. 2005.

[ZIPF49] G. Zipf, Human Behavior and the Principle of Least Effort, Reading, MA: Addison-Wesley, 1949.