uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dafu Lou

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineerng

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Analysis and Design of Peer-to-Peer Video On-Demand Streaming

TITRE DE LA THÈSE / TITLE OF THESIS

Tet Yeap

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Ahmed Karmouch

Amiya Nayak

Victor Leung

Dorina Petriu

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Analysis and Design of Peer-to-Peer Video On-Demand Streaming

by

Dafu Lou

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the Ph.D. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-59548-0
Our file Notre référence
ISBN: 978-0-494-59548-0

Abstract

The problem of video on-demand (VoD) streaming via peer-to-peer (P2P) overlay networks is still challenging. A novel resilient P2P VoD streaming system is proposed in this thesis based on a multi-video-block transport framework with user interactivity supports. The proposed system, namely resilient VoD (rVoD), consists of a distributed encoded video content caching scheme and a transport protocol based on rateless codes.

Notions of *production* and *saturation time*, naturally arising in this context, are introduced to characterize the dynamics and performance of the P2P streaming system. Quantified in terms of production, the throughput of the presented framework is shown analytically to increase with time in a manner no worse than an exponential function prior to the saturation time. The notions of production and saturation time may also serve as fundamental concepts for the study of quality of service (QoS) for the proposed system.

A necessary condition of *saturation time* and the relationship between *production* and *server stress* are derived for an optimal P2P streaming system. We then study the definition of QoS metrics and their tradeoff. A constraint equation of QoS metrics tradeoff is derived, and the numeric results are demonstrated.

A simulation system is developed to measure the performance of rVoD in terms of production, saturation time, several QoS metrics, server stress, latency, and overhead. Results of the simulations demonstrate the advantages of the proposed system compared to previous works and confirm our theoretical analysis.

Acknowledgements

I am deeply indebted to my supervisor, Dr. Tet H. Yeap, for his support, encouragement, and invaluable advice during my Ph.D. study. Professor Yeap has been an excellent supervisor in communication, network security, and networking. I feel fortunate to have been introduced to the research area of video communication via overlay networks at a very early stage. I really loved the freedom of exploring my favorite research directions during my research at the Bell University Laboratories at the University of Ottawa.

I would like to thank Dr. Yongyi Mao for teaching me knowledge of coding theory and for his invaluable suggestions for this thesis. I would also like to thank Prof. Dorina Petriu, Prof. Amiya Nayak, and Prof. Ahmed Karmouch for serving on my thesis committee.

I have also enjoyed pleasant and fruitful collaborations with friends at the University of Ottawa, which helped shape the structure of this thesis. I am particularly thankful to Hongyu Guo, Hanxi Zhang, and Dongmei, Jiang.

This thesis is dedicated to my wife and my parents, whose love and unconditional support provide constant inspiration in my life.

Contents

List of Tables

List of Figures

List of Abbreviations

ϵ ..... Overhead of rateless coding

γ ..... Overlay network loss rate

λ ..... Arrival rate of users

$\overline{T}_j$ ..... Average jitter time over all users

$\overline{T}_w$ ..... Average waiting time over all users

a ..... Average input bandwidth of users

b ..... Average output bandwidth of users

s ..... Total output bandwidth of server

$\widehat{T}_m$ ..... Saturation time of block m

$\widetilde{\Delta_m}$ ..... Delay in which users with the video block-m become requesting users for the video block-$m+1$

$\widetilde{N_m}(t)$ ..... The potential number (production) of users with the video block-m at time t

A ..... Size of the video block in bytes

C_i ..... Capacity of network link

i ..... Index of arrival users

M ..... Number of blocks of the video file

m ..... Index of video blocks

$N(t)$ ..... Number of arrived users at time t

$N_m(t)$ ..... Number (production) of users with the video block-m at time t

q_m ..... Factor of allocated server output bandwidth for users requesting the

	video block-m
$R(t)$	Requested bandwidth of arrival users at time t
r_m	Factor of allocated output bandwidth of users with the video block-m for supplying
$R_m(t)$	Requested bandwidth of users for the video block-m at time t
$S_m(t)$	Potential supplying bandwidth of the video block-m at time t
$X_m(t)$	Number of users who have stored the video block-m at time t
$\mathcal{U}_0$	Video server
$\mathcal{U}_i$	Users that arrive in the system ordered by i
CDN	Content Distribution Networks
DPCC	Distributed rateless code-based encoded video content caching
DVD	Digital Video Disc
rVoD	resilient P2P VoD system

Chapter 1

Introduction

1.1 Content Distribution over P2P Network

Content distribution is the service of dispersing content from a server to many receivers. For example, web applications, software dissemination, and video file distribution are typical applications of content distribution over the Internet. In this thesis, we mainly focus on high quality of service (QoS) content distribution. High QoS content refers to video streams here. It must meet two requirements: high playback rate (i.e. high transmission bandwidth rate) and continual playback (i.e. without experience of jitter). The known applications are IPTV, Video On-Demand (VoD), and live-video broadcast. Recently, such applications have increasingly attracted significant attention from academia and industry [1, 2, 3]. Many believe that the rise in content distribution and the related entertainment industry may greatly impact many industrial sectors centered around or based on information technologies.

The proposed approaches or methodologies to content distribution can be categorized into three classes by the infrastructure: native IP multicast [4], client-server mode-based content distribution networks (CDN) [1], and peer-to-peer (P2P) overlay networks.

A native IP multicast system is the best solution for content distribution in terms of bandwidth cost, but it relies on the underlying network, such as routers and switches,

having multicast functions. Such a requirement becomes hard to meet in the current Internet, as network nodes do not support IP multicast.

CDN based on the client-server model does not rely on the underlying network, for it is at the application level, but it needs multiple servers or proxies to handle a high rate of user requests. The cost to maintain and deploy such systems is high. Moreover, the system resources have lots of redundancy at most times to meet the requirements of the highest usages. Thus, this methodology is also not feasible in current IP networks.

In P2P overlay networks, receivers can obtain content directly via other receivers rather than from the server (the source node) over overlay networks. An overlay network [5], in contrast to underlying networks, is built on top of it. All nodes are completely connected by virtual links without information from the underlying network topology. P2P architecture has become a modern approach recently for content distribution, since P2P file sharing on the Internet has been successful since it was introduced to share music by Napster in 1999 [6]. There are many P2P file-sharing commercial systems available now, the most famous one being BitTorrent [7]. Some also reported that P2P file-sharing accounts for as much as 70% of Internet traffic. The P2P network has become increasingly popular recently over the Internet as one of the architectures of overlay networks.

We are now witnessing the emergence of a new class of content distribution based on peer-to-peer overlay networks, namely P2P video streaming [8, 9, 10, 11], with the widespread adoption of broadband residential access and the increasing demand of multimedia service over the Internet. In P2P video-streaming systems, a video file (or multimedia data) can be played while being received based on peer-to-peer overlay networks. As reported in [12], a P2P video streaming system (PPLive) successfully broadcasted the 2006 Chinese New Year's celebration to over $200,000$ users over the Internet at a bit rate of $400 - 800$ kbps range. We expect that thousands of live video streaming channels and video on-demand channels will be available on the Internet in the near future.

However, there still exist many significant challenges to implement high-QoS content streaming for overlay networks, such as on the Internet, even though much research

effort has been put into content distribution (the details are reviewed in Chapter 3). In this thesis, we will examine this problem and investigate the possibility of implementing high-QoS content distribution, particularly in the on-demand model.

1.2 Challenges

First, the video content needs to be received *in order* so that real-time online viewing, namely playing back while downloading, is possible. In other words, to keep the continuity of playing back the video content, the receiver should download the video content (or video segment) in its order of playing frame. In P2P file-sharing systems, the aim of the receiver is to download the whole file; thus the order issue does not exist. Such constraint causes two new problems: the potential number of peers (or users) that can supply content in buffer to the receiver becomes less, and the time (or transmission delay) becomes more sensitive for multi-session cases (i.e. the synchronous problem of many-to-one communication). However, many proposed results for P2P streaming handle this problem by minimizing throughput or maximizing the downloading rate [13, 14, 15, 16, 17, 18]. We argue that those are just average performance metrics in time that cannot replace the measurement of continual viewing. In this thesis, we treat it in a different way that measures jitter by each small video block, based on the proposed multi-block (or clip) transport framework.

Second, arranging the synchronous content of playing back with the video streaming system becomes more critical, i.e. we need to carefully arrange at which part of the video content (or which frame of the video content, denoted the "view point" in this thesis) to start to download from the system for a newly arrived user so that the user can continually view and wait for the minimum time to start to view. At the same time, there are specific requirements for this for different applications. For example, users always want to view the video from the beginning in video on-demand systems. But in live-streaming systems, such as live-news broadcasting, users wish to watch the program

without delay with the news centre. Clearly, such problems will not be encountered in current P2P file-sharing systems. In fact, this is one of the most critical problems for video streaming based on P2P networks. The problem has never been formalized or been explored in previous works [19, 20, 14, 15, 16, 17, 21, 22]. In this thesis, we explore the problem based on the proposed multi-block transport framework.

Third, the characteristics of users in P2P streaming networks are more critical than in P2P file-sharing systems. For example, users' departure and jumping operations are critical in P2P streaming networks. In P2P file-sharing systems, users are willing to wait for a long time to finish downloading the whole file, but users in P2P streaming systems will quit viewing if they cannot view the video smoothly. Particularly, the operation whereby the user jumps to different parts of the content to view does not exist in P2P file-sharing systems. Thus, this thesis will address the design of users' storage management and transportation networks to handle this problem.

Finally, the bandwidth required for P2P streaming systems is high when we know that most video-streaming applications require a high transmission rate. Particularly, more people like to enjoy high-quality video, such as DVDs/videos of DVD quality, in which case the basic required video rate is 3.75 Mbps, and so far we have not seen that the existing P2P video-streaming systems could successfully support such a rate.

In summary, we mainly focus on implementing streaming high-QoS content via P2P overlay networks. A multi-video-block transport framework based on rateless codes to handle the above challenges in a novel way is proposed.

1.3 Rateless Coding as an Enabling Technology

Rateless codes are a set of special codes that can be used to achieve the performance of linear network coding [23], but they are not linear codes. Considering that our research is interested in reducing the sophistication of the scheduling algorithms and in increasing the robustness of the dynamic network environment, rateless codes will really help us

to solve the problem of designing good linear network codes with lower and constant overhead.

Designed for erasure channels, a fountain code (LT code [24] or Raptor code [25]) is a mapping of k information symbols to an infinite stream of codeword symbols. The symbol here can be a bit or a binary vector of any prescribed length. The transmitter continuously sends codeword symbols until the receiver, upon receiving a sufficient number of symbols, is able to decode the intended k symbols. The receiver then sends an `ACK` to the transmitter to terminate the transmission of the current codeword. Taking LT codes, for example, the mapping of the k-symbol information vector to the codeword stream is via a pseudo-random selection of an integer d from $\{1, 2, \ldots, k\}$ followed by another pseudo-random selection of d information symbols and then an `XOR` operation of the d symbols. The family of fountain (LT) codes may be in principle constructed by the specified distribution from which d is drawn. Each realization of the i.i.d. sequence $\{d_i\}$ defines a code.

The performance of fountain codes shows in that they can achieve the capacity of erasure channels without any knowledge of the probability law of erasure events. That is, without such knowledge, as long as $k(1+\epsilon)$ symbols are received, the decoder can recover the k-symbol information vector, and ϵ, referred to as the *overhead*, is quite small (usually smaller than 5% when k is relatively large). In addition, the complexity of encoding and decoding fountain codes is low and practicable, and commercial products based on fountain codes for large-file transport have been developed [26, 3].

It is natural to utilize rateless codes for P2P networks as the presented features. When we apply rateless coding to each user for P2P networks, two major problems of P2P networks are completely solved: the reconciliation problem between multiple transmitters and one receiver for the same content and content synchronization between the receiver and the transmitter for unreliable overlay connection. Figure 1.1 gives a simple example of how rateless codes, such as Raptor codes [25], can be deployed to solve the above two problems without complex algorithms. In this example, each user

just needs to be initialized with a unique Raptor codes seed. Then the user keeps generating encoded information bits $(e_i^1, e_i^2, e_i^3, \ldots)$ based on its seed from its buffered content (here we suppose that the buffered video block B_m only has four information bits—a, b, c, and d.) for receivers. For a receiver, the encoded information bits received from different senders are totally independent, since their encoding seeds are unique; thus the distribution of received bits from different senders is the same as from one sender. In other words, the overhead ϵ of rateless coding remains the same for multiple senders and one sender. In this example, we assume the overhead is zero; thus each user can successfully decode as long as it receives four encoded information bits. Another fact is that it is not important if every encoded information bit transmitted can be received because of rateless coding; thus it will not increase overhead, for the connections among senders and receivers are broken and reconnected randomly, i.e. the transmitted information bits are lost. Since the overhead of Raptor codes is small and constant, the achievable downloading rate from the multiple transmitters can be easily computed as a summation of each transmitter uploading rate. It likes a fluid mode in which a mass water flow can be arbitrarily separated into many small flows. And many small water flows can be added together to form a mass water flow.

The example also shows that the architecture of applying encoding/decoding to each user for P2P networks based on rateless codes shares the idea of network coding [27]. The benefits discovered by network coding could also be applicable for the proposed framework; for example, each user could achieve their maximum transmission rate from the source without affecting others. In this thesis, we will name such P2P video-streaming networks based on rateless coding P2P video rateless streaming networks.

1.4 Contributions

In this thesis, the problem of VoD streaming via P2P overlay networks is examined. A novel resilient P2P VoD streaming system, where we propose a novel distributed encoded

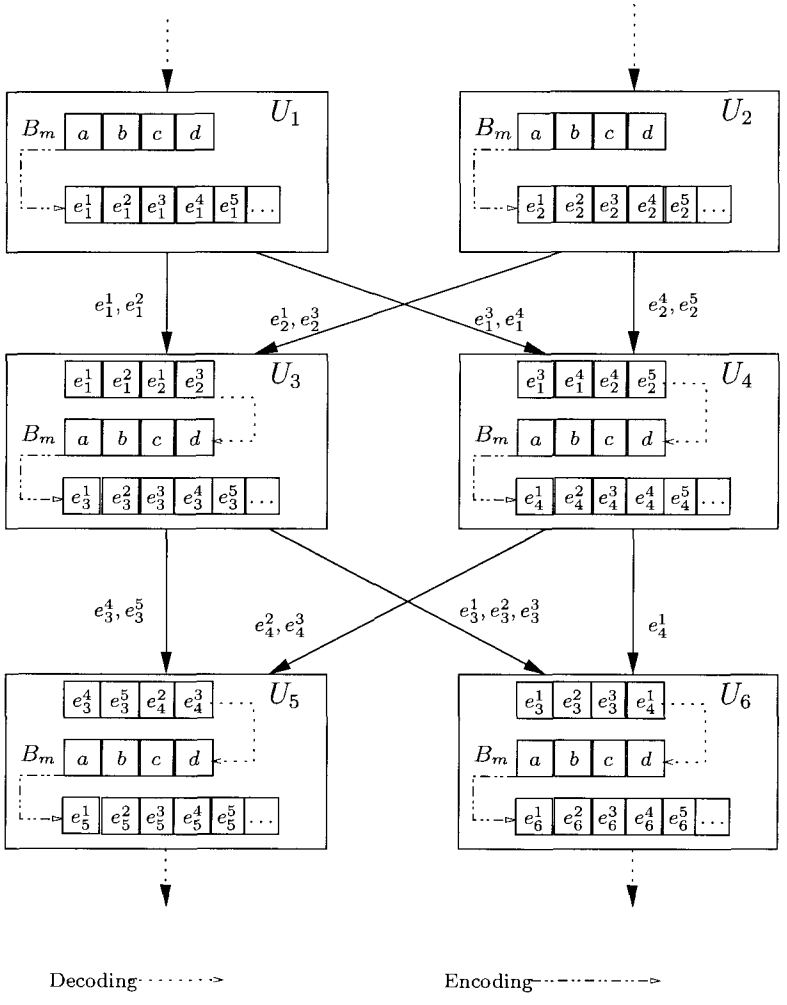


Figure 1.1: A simple example of using rateless coding for P2P streaming.

video content caching scheme and a transportation protocol based on rateless codes is designed. We first introduce the concept of production to measure the dynamics of the system and the concept of distributed partial complete-video content caching to allow each user to store the partial information of the complete video content using its buffer with limited size. We also derive important properties of P2P streaming networks in terms of bounds of P2P streaming network capacity, QoS control, and traffic control, through both theoretical and experimental studies. We summarize the major contributions of this thesis in the following list.

- A multi-video-block transport framework for streaming high-QoS content based on P2P networks is proposed. In this novel framework, the video content (or file) is divided into small blocks (or clips) and then transmitted in sequence. Each small block distribution becomes a sub-P2P streaming system. All such sub-P2P streaming systems are linked together in the order of blocks. The relationship between them is that the output (the number of peers that received the block completely) of a sub-P2P streaming system is the input (number of requesting peers) of its next linked sub-system. The advantage of the framework is that we can model and measure the dynamic performance of a P2P streaming system. If we review the previous works completely, no one has addressed a similar idea yet.

- A rateless coding-based transport protocol for the proposed multi-video-block transport framework is proposed. Fountain codes [25] are a class of realistic and optimal rateless codes in terms of the complexity of implementation. The benefit of applying rateless coding is that the content-scheduling problem among multiple senders is completely solved so that we can treat the bandwidth assignment linearly for each sub-P2P streaming system based on the multi-video-block transport framework.

- Notions of *production* and *saturation* have been introduced to measure the dynamics of the system capacity of P2P streaming networks at any time. We believe that such definitions can capture the performance of a realistic P2P streaming system.

- The dynamics of the system and QoS performance within the proposed framework are theoretically analyzed. The throughput of the presented framework is shown analytically to increase with time in a manner no worse than an exponential function prior to the saturation time. The notions of *production* and *saturation time* may also serve as fundamental concepts for the study of QoS for the proposed framework.

- A distributed partial complete-video content storage management scheme is proposed to realize the system that allows each user to store the partial information of the complete video content using its buffer with limited size. The proposed design efficiently solves the problems of supporting user jumping so that every user can maintain connections with both its parents and its children while users perform VCR operations.

- A novel resilient P2P VoD system, namely rVoD, which supports VCR operations with low startup latency and low overhead is proposed. In rVoD, the requesting-access popularity-based storage management algorithm and allocation bandwidth allocation algorithm have been designed. We also design a user locating scheme based on an index server so that the system can be managed efficiently with low overhead.

- A packet-based P2P simulation system is developed to evaluate our proposed P2P streaming system. We examined the dynamic performance of the system in terms of system throughput, saturation time, and several QoS metrics, given the limited uploading capacity of the server. We also measured the server stress, latency, and overhead when QoS requirements are given. The results have also been compared to a previously proposed VMesh [28]. Results of the simulations demonstrate the advantages of our proposed P2P VoD system and confirm our theoretical analysis.

- A necessary condition of *saturation time* for an optimal P2P streaming system and the relationship between *production* and *server stress* is derived. We first

introduce the concept of cross-over design problems of user storage management and content transportation for P2P systems and derive a condition under which the two procedures can be optimized independently.

- We have studied the definition of QoS metrics and their tradeoff relationship for a given P2P system. A constraint equation that QoS metrics hold for a given P2P system is derived. The numeric results are also demonstrated.

1.5 Organization

The thesis is organized as follows. We present background material in Chapter 2 and related research in Chapter 3. We propose a rateless coding-based P2P content streaming framework and address the theoretical analysis based on average cases in Chapter 4. The theoretical analysis results for general P2P streaming system design guidelines are further addressed in Chapter 5. Then we design a P2P video on-demand system (rVoD) based on rateless coding. The QoS performance analysis and simulation results are discussed in Chapter 7. Conclusions and the direction of further research are presented finally in Chapter 8.

Chapter 2

Technical Background

2.1 Overlay Network

An overlay network is a virtual topology on top of the physical network topology, such as many peer-to-peer networks [29,30,28]. Nodes in an overlay network can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links in the underlying network. Overlay networks have two common features: guaranteed data retrieval and self-organization. When the physical network is an IP network, the overlay network refers to the overlay network over IP. The overlay network over IP shares the properties of the IP network. According to the Internet standards, there are four forms of IP communication, each with its own unique properties.

Unicast: IP packets are sent from host to host in a unicast [31,32] network. Usually, a unicast IP address is associated with a single device or host, but it is not a one-to-one correspondence. Some individual devices have several distinct unicast addresses, each for its own distinct purpose. In a unicast network, sending the same data to multiple unicast addresses requires the sender to send the data many times over, once for each recipient.

Broadcast: Broadcast [31, 32] is when a single device transmits a message to all

other devices in a given address range. Broadcast permits the sender to send the data only once, and all receivers can copy it. In the IP protocol, broadcast packets have the host (and/or subnet) portion of the address set to all ones. For example, to send to all addresses within a network with the prefix 192.168.1, the directed broadcast IP address is 192.168.1.255 (assuming the netmask is 255.255.255.0). By design, most modern routers will block IP broadcast traffic and restrict it to the local subnet.

Multicast: A multicast address is associated with a group of interested receivers. Multicast enables a single device to communicate with a specific set of hosts, not defined by any standard IP address and mask combination. Over a multicast network, data is sent from the source only once, and the network must transmit the data to multiple destinations. Usually multicasting is more efficient for many multimedia applications. According to RFC 3171 [33], addresses 224.0.0.0 to 239.255.255.255 are designated as multicast addresses. This range was formerly called **Class D**. The sender sends a single datagram (from the sender's unicast address) to the multicast address, and the routers take care of making copies and sending them to all receivers that have registered their interest in data from that sender.

Anycast: Anycast [34] is a one-to-many routing protocol. It is similar to broadcast and multicast, but the data stream is not transmitted to all receivers, just the one which the router decides is the **closest** in the network. Anycast is useful for balancing data loads. It is used in DNS [35].

2.2 Linear Coding

2.2.1 Rateless Codes

Rateless codes are a type of code designed for unknown channel characteristics. Usually codes are characterized by a rate and a distance to ensure correct data transmission in a particular channel setting. However, rateless codes would like to achieve data transmission with small overhead in an unknown channel setting, while maintaining efficient

encoding and decoding. LT codes [24] and Raptor codes [25] are known as two classes of rateless codes, which are proposed by the Digital Fountain.

In general, a fountain code is a mapping of k information symbols to an infinite stream of codeword symbols. The symbol here can be a bit or a binary vector of any prescribed length. The transmitter continuously sends codeword symbols until the receiver, upon receiving sufficient many symbols, is able to decode the intended k symbols. The receiver then sends an `ACK` to the transmitter to terminate the transmission of the current codeword. LT codes and Raptor codes are two particular classes of fountain codes. Both of them have good performance for erasure channels.

LT codes are the first class of practical fountain codes with near optimal erasure correcting codes, which were invented by Michael Luby in 1998 [24]. The distinguishing characteristic of LT codes is that a particularly simple algorithm based on the exclusive `OR` operation ($\oplus$) is designed to encode and decode the message.

LT encoding first divides the uncoded message into k blocks (k-symbol information) and randomly selects a degree d from $\{1, 2, \ldots, k\}$ via a pseudo-random selection for the next encoded block. That is followed by another pseudo-random selection of d blocks and an `XOR` operation of the d blocks. Finally, the encoded block is formed by appending a prefix of coding information (i.e. k, d, and the list of indices) and some form of error-detecting code. The family of LT codes may be in principle constructed by the specified distribution from which d is drawn. Each realization of the i.i.d. sequence $\{d_i\}$ defines a code. Luby himself proposed the ideal soliton distribution for the degree distribution defined by

$$\rho(d = 1) = \frac{1}{k},$$

$$\rho(d = i) = \frac{1}{i(i-1)}, (i = 2, 3, \cdots, k).$$

However, the ideal soliton distribution does not work well in practice, and a modified distribution, the robust soliton distribution [24], is then addressed. The effect of the

modification is to produce more packets of degree 1 and fewer packets of degree greater than 1. The robust soliton distribution is $\mu(\cdot)$, defined as follows. Let $R = c \cdot \ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$, where δ is the allowable failure probability of the decoder to recover the data for a given number k of encoding symbols. Define

$$\mathtt{T}(i) = \begin{cases} R/ik & i = 1, \cdots, k/R - 1 \\ R\ln(k/\delta)/k & i = k/R \\ 0 & i = k/R + 1, \cdots, k \end{cases}$$

and add the ideal solition distribution to obtain $\mu(\cdot)$:

$$\beta = \sum_{i=1}^{k} \rho(i) + \mathtt{T}(i). \tag{2.1}$$

$$\mu(i) = (\rho(i) + \mathtt{T}(i))/\beta, \quad for \;\; i = 1, \cdots, k. \tag{2.2}$$

The LT decoding process uses the `XOR` operation to retrieve the encoded message through the following steps:

- If the received encoded block is not clean, i.e. the block is incomplete (CRC32 checking is failed at TCP/UDP layer), or if it replicates a block that has already been received, this block is discarded.

- If the clean received encoded block is of degree $d > 1$, it is first processed against all the decoded blocks in the message queuing area that are encoded in this encoded block using `XOR`, then stored in a buffer area if its reduced degree is greater than 1.

- When a new, clean encoded block of degree $d = 1$ is received (or the degree of the current processed block is reduced to 1), it is successfully decoded and moved to the message queuing area.

- Then the new decoded block is `XOR` operated against all the blocks of degree $d > 1$ residing in the buffer that are encoded using this decoded block. The degree of

those matching blocks is decremented, and the list of indices for those blocks is adjusted to remove the new decoded block ID. The matching blocks are reduced to degree 1 and are in turn moved to the message queuing area, and then processed against the blocks remaining in the buffer following the above procedure.

- When all k blocks of the message have been moved to the message queuing area, the message has been successfully decoded.

If the output symbol degrees are chosen according to the robust soliton distribution, k input symbols can be recovered from any $k + O(\sqrt{k}\log^2(k/\delta))$ output symbols with probability $1 - \delta$. The average encoding and decoding cost is $O(log(k/\delta)$.

Raptor codes [25] are one of the first known classes of fountain codes with linear time encoding and decoding, and were invented by Amin Shokrollahi in 2000/2001. The key idea of Raptor codes is to relax the condition that all input symbols need to be recovered in LT codes. A traditional erasure correcting code concatenated with LT codes is introduced to recover all input symbols. An LT decoding graph needs only $O(k)$ edges, allowing for linear time encoding if it needs to recover only a constant fraction of its input symbols. Thus, Raptor codes are formed by the concatenation of two codes: a fixed-rate erasure code and LT codes. The fixed-rate erasure code, named as **pre-code** or **outer code**, can be a concatenation of multiple codes. For example, a high-density parity check code is concatenated with a simple, regular low-density parity check code [36, 37] in the code standardized by 3GPP. LT codes, called the **inner code** in Raptor codes, take the result of the pre-coding operation and process the symbols in the same way as the above LT codes procedure.

The performance of fountain codes is amazing in that they can achieve the capacity of erasure channels without any knowledge of the probability law of erasure events. That is, without such knowledge, as long as $k(1 + \epsilon)$ symbols are received, the decoder can recover the k-symbol information vector, and ϵ, referred to as the *overhead*, is quite small (usually smaller than 5% when k is relatively large). In addition, the complexity of

encoding and decoding fountain codes is low and practicable, and commercial products based on fountain codes for large-file transport have been developed.

2.2.2 Multiple Description Coding (MDC)

Multiple description coding (MDC) [38,39] is a coding technique that fragments a single source stream into n independent sub-streams ($n >= 2$) referred to as descriptions. The basic idea of MDC is to generate multiple independent descriptions such that each description independently describes the source with a certain fidelity, and when more than one description is available, they can be synergistically combined to enhance the quality. The ideal MDC is to decode the original stream from an arbitrary subset of descriptions. The quality of the decoded stream is determined by the size of the received subset of descriptions, which is expected to be the data rate sustained by the receiver in a lossy network.

MDC is also a form of data partitioning compared to layered coding. Layered coding mechanisms generate a base layer and n enhancement layers. The base layer is necessary for the media stream to be decoded, and enhancement layers are applied to improve stream quality. However, the first enhancement layer depends on the base layer and each enhancement layer $n + 1$ depends on its subordinate layer n. Thus, MDC can only be applied if n was already applied. Hence, media streams using the layered approach are interrupted whenever the base layer is missing, and as a consequence, the data of the respective enhancement layers is rendered useless. The same applies for missing enhancement layers. In general, this implies that in lossy networks the quality of a media stream is not proportional to the amount of correct received data.

Besides increased fault tolerance, MDC allows for rate-adaptive streaming: content providers send all descriptions of a stream without knowing the download limitations of clients. Receivers can only subscribe to a subset of these streams to recover partial information.

The difference between MDC and fountain codes is that MDC only generates the

limited number of encoded symbols and can recover the original message from an arbitrary subset of encoded symbols with a certain fidelity, but fountain codes can generate almost an unlimited number of encoded symbols and recover the exact original message from any large enough subset of encoded symbols. MDC is a type of source coding, and fountain codes are a type of channel coding.

Currently available MDC coders, however, are mostly limited to the generation of two descriptions. Only some works [38, 40] have proposed mechanisms to transform a scalable source bit stream into a robust MD packet stream by encoding source layers of decreasing importance with progressively weaker forward error correction (FEC) channel codes. Based on priority encoding transmissions [41], unequal protections are assigned in each layers of the proposed works.

2.2.3 Network Coding

The concept of network coding is formally introduced by Ahlswede *et al.* in [27]. The idea of network coding is to apply encoding and decoding operations at intermediate nodes as well as at the source and receivers in the network, as contrasted to source coding, where only the source encodes and the receivers decode. In general, network coding can achieve optimal utilization of the resources of a network topology.

Figure 2.1 is a well-known example that shows how network coding achieves the maximum flow. In the example, the source (S) needs to multicast two bits (a and b) to two receivers (R_1 and R_2). Each link is undirected and has a unit capacity of 1 bps. Figure 2.1(a) depicts a way to transmit without network coding. We can obtain end-to-end throughput of 1.5 bps. When the replay node in the middle receives bits a and b from the source s, it encodes them into one bit $a + b$. Here $+$ is the exclusive **OR** operation. As shown in Figure 2.1(b), receiver R_1 can receive a and $a + b$. Then it can recover two bits, a and b, by the exclusive **OR** operation, i.e. $b = a + (a + b)$. Similarly, R_2 can also recover a and b at the same time. Now we can achieve the maximum throughput of 2 bps by applying network coding.

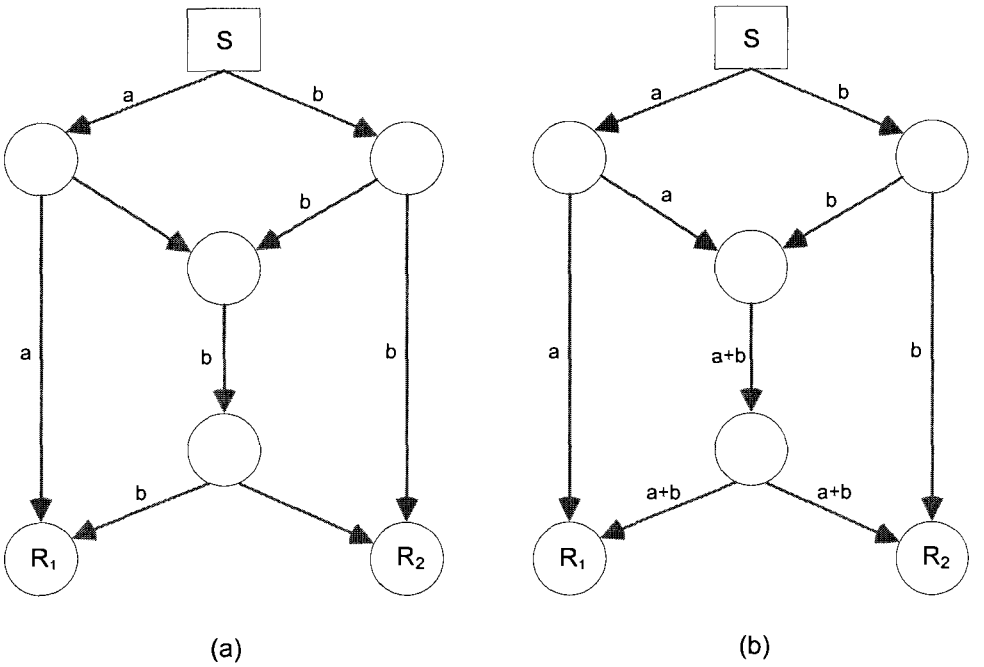


Figure 2.1: An example showing the principle of network coding.

Ahlswede *et al.* in [27] and Koetter *et al.* in [42] prove that every receiver can achieve its maximum flow rate from the multicast source in a directed network with network coding support. Here the maximum flow can be computed for every receiver in the network from the source at a time. In other words, we can assume there is only one receiver to utilize the network resource when computing the maximum flow rate in the original network. Their great work tells us that more than one network flow to multiple receivers can share the bandwidth of a network edge by applying coding without reducing the achievable network flow of each one. Then Li *et al.* in [23] prove that we can achieve the network maximum flow by linear coding, namely, linear network coding.

Ho *et al.* [43] presented a random network coding that relies on using random codes at nodes for multicast or unicast networks. They gave a practical way to construct linear network codes for dynamic networks so that linear network coding is applied into the applications; for instance, it was used by Christos *et al.* [3] to share a large file for P2P content-distribution systems. Jaggi *et al.* [44] then studied the efficient code construction.

They designed polynomial time algorithms that determine the coding operations to be applied at each node. Their results improved the previous algorithms presented in [23]. Wu *et al.* [45] have also presented the relation between distributed source coding and network coding.

All research results reveal that coding technology is of great use in content distribution systems, such as P2P networks. In our research, we mainly want to apply coding technology such as rateless coding and network coding to help reduce the complexity of the scheduling algorithm of information across overlay networks and increase the robustness for the user's random departure.

2.3 Peer-to-Peer Networks

P2P is a communications model, in contrast to the client-server model, where each party has the same roles and capabilities. The peer-to-peer model is used not only in communication areas, but also in business areas. In our research, P2P has come to describe applications in which users can use the network to exchange data, such as video content, with each other directly. The P2P network is a type of transient network on the Internet that allows a group of computer users with the same networking program to connect with each other and transmit data to one another directly or via others, which is a completely decentralized network. Napster [6], Gnutella [46], and BitTorrent [7] are typical examples of peer-to-peer softwares.

The P2P network has attracted significant attention and has been widely adopted to share files on the Internet since it was introduced by Gnutella in 1999. However, P2P technology appeared earlier than this. At least, the idea of P2P was developed in USENET [47] and FidoNet [48] – two very successful, completely decentralized P2P networks. USENET, created in 1979, is the distributed application that provides most of the world with its newsgroups. FidoNet, created in 1984 by Tom Jennings, is a decentralized, distributed application for exchanging messages between users of different

BBS systems.

In recent usage, the P2P network is an overlay network on the Internet, as shown in Figure 2.2. All peers (or users) of the P2P network are connected in the overlay (level), of which a full connected graph is comprised. The P2P network has the following features that are different from the traditional networks:

- All nodes (users) in the P2P network do not know underlying (physical) network topology.
- All nodes do not have the ability to control the underlying path (or routing).
- The size of the P2P network (i.e. the number of users in the P2P network) can be from zero to millions (very large).
- Each node (peer) joins and leaves the network in random.

The P2P network has two typical types of architecture: the pure peer-to-peer network shown in Figure 2.3(a) and the hybrid peer-to-peer network shown in Figure 2.3(b). The pure P2P network is a completely decentralized network in which all peers have an equal role. The current famous P2P file-sharing system, BitTorrent [7], is an example of a pure P2P network. The hybrid P2P network still has a central server that does some network maintenance, but the main data are still transmitted in a peer-to-peer model. An example of a hybrid P2P system is PPLive [8], a P2P based video broadcast system.

P2P streaming refers to distributing video streaming based on the P2P network. Since the great success of file-sharing systems such as BitTorrent on the P2P network, P2P streaming has attracted significant attention from researchers. According to the type of service of the video-streaming systems, they can be categorized as live streaming systems and video on-demand systems. For technological purposes, the practical implemented VoD systems can be classified as *true* VoD (referred to as tVoD) or *near* VoD (referred to as nVoD), where tVoD is the system in which each user can immediately watch the

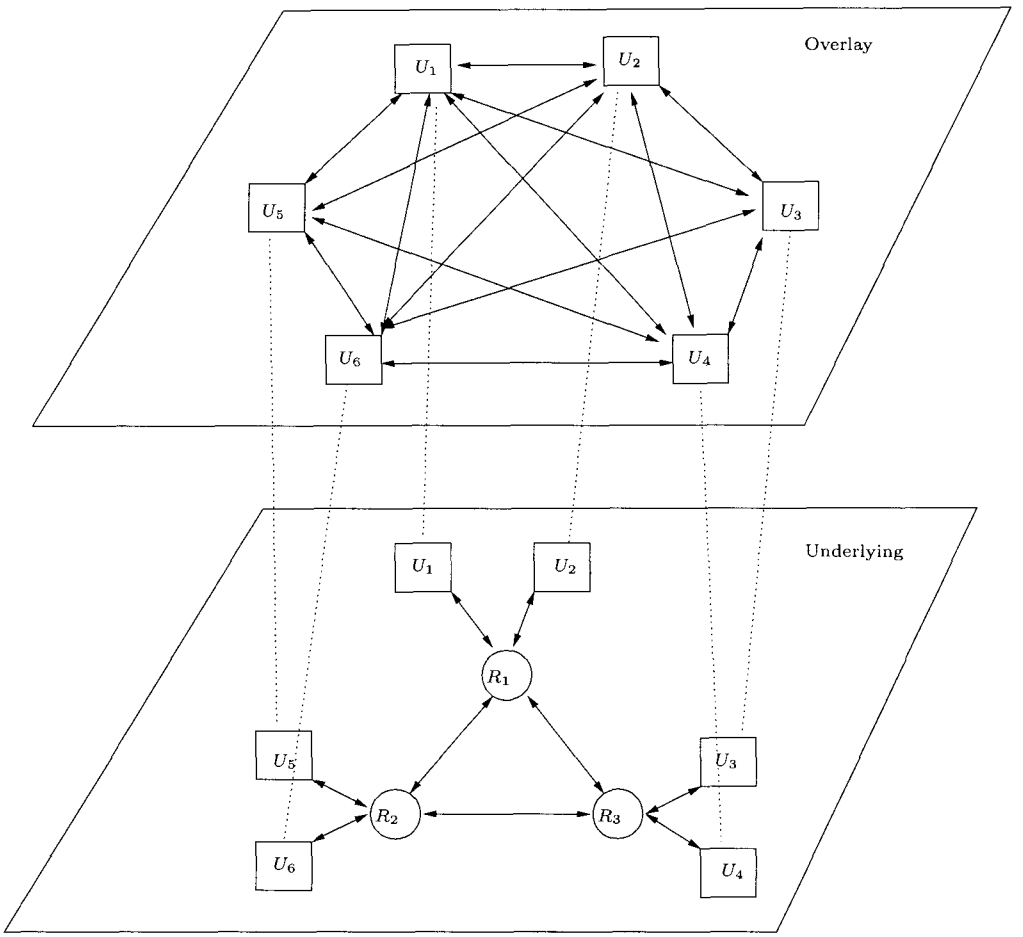


Figure 2.2: The relationship between overlay P2P networks and underlying networks.

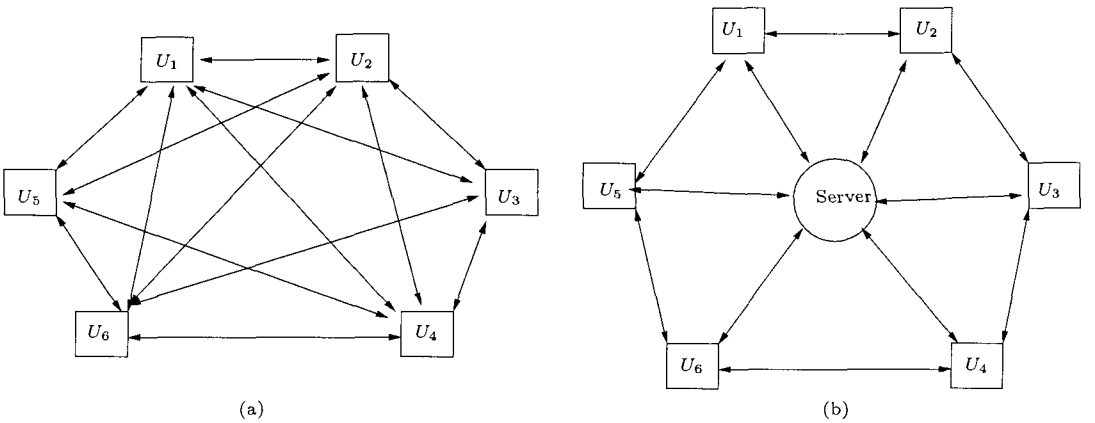


Figure 2.3: Pure P2P networks and hybrid P2P networks.

video from the beginning after it joins the system at any time, and in an nVoD system, the user may wait for a short time, such as a few minutes, to watch the content from the beginning after the user selects the content. What makes live streaming systems different from VoD is that every user plays the content that is synchronous with the server (or the content source). In fact, due to the transmission delay, live streaming is not really synchronous with the server for all users. It is also not necessarily synchronous for most streaming applications, even those that require real time, such as watching TV programs. From the point of technique, we can categorize video streaming systems into three classes. First, viewing content for all users is always from the beginning of the content. Second, viewing content is synchronous with the server playing. And finally, there is no specific common synchronous point in the system. We would like to raise a question here as to whether the theoretical bounds of the service capacity for the three cases are the same. However, to the best of our knowledge, no one has pointed out this question yet. In terms of classification of the video-streaming systems, we note that most proposed P2P streaming systems are P2P live streaming systems or non-P2P on-demand streaming systems. There are not many research results found for VoD systems based on P2P networks, since there are more significant challenges to guaranteeing the quality of service.

The technical problems of the P2P streaming can be categorized as three classes: information searching (locating problems), media data transportation, and security-related problems. Locating problems refer to finding the needed information or updating the information to other peers in the distributed P2P network environment. The problem becomes hard because a management centre does not exist in the P2P network. The proposed solutions for the problem can be grouped into structure-based schemes and unstructure-based schemes, such as Chord [29] and Gossip [49]. It is hard to implement media streaming in P2P networks, since users' behaviour is unknown and unpredictable. The available solutions mainly fall into two types: tree-based protocol [16, 50, 51] and mesh-based protocol [52, 28, 53]. The security problems include privacy, authorization,

authentication, and audit, which become hard, since the distributed environment does not allow us to fully control all users (peers). The typical problems [54] are free ride, anonymity, censorship attack and sybil attack [55], etc.

2.4 Quality of Service Problems

The quality of service problem is how to implement application-layer QoS control in P2P streaming networks. The objective of QoS control is to avoid congestion and maximize video quality in the presence of packet loss and user failure.

For the traditional networks, such as the Internet, the techniques of QoS control include congestion control and error control [56]. Congestion control typically controls rates for video-streaming applications. The purpose of rate control is to determine the sending rate of video traffic based on the estimated available bandwidth. The schemes of rate control have three classes: source-based, receiver-based, and hybrid rate control [56]. Another method for congestion control is rate shaping, which matches the rate of a pre-compressed video bit stream to the target rate constraint. The typical techniques for rate shaping are the codec filter, frame-dropping filter, layer-dropping filter, and re-quantization filter.

Error control mechanisms include forward error correction (FEC), retransmission, error-resilient encoding, and error concealment. The principle of FEC is to add redundant information so that original information can be recovered with packet loss. Three kinds of coding, channel coding, source coding, and joint source-channel coding, can be used for FEC. Fountain codes can also be considered as one kind of optimal channel coding for FEC. Error-resilient encoding is to enhance the robustness of compressed video to packet loss. Multiple descriptions coding [38] is presented for robust Internet video transmission. Error concealment is a technique that is performed by the receiver to conceal the lost data and make the presentation less displeasing to human eyes. The basic approaches for error concealment are spatial and temporal interpolation [57].

On P2P networks, considering the feature of user departure or failure in random, QoS control becomes more difficult. The techniques addressed above are not sufficient. A good solution we can find easily is to quickly find more available users who have the data needed to maintain the downloading rate. The problem is similar to the link failure in the optical network, in which we need to find another link to replace the failed link. However, searching for available users is also another difficult problem in completely decentralized networks. Characterizing QoS in P2P networks and to addressing an efficient methodology for QoS control in P2P networks are still open problems.

Chapter 3

Related P2P Works

The problem that we are interested in is content distribution via P2P networks, which can be classified into three groups: theoretical analysis based on modeling methodology, architecture (or protocol) design, and performance measurement. In this chapter, we mainly review related research results.

3.1 Theoretical Performance Analysis

Many efforts characterize the P2P file-sharing network for both its transient and steady states. Yang *et al.* [58] propose a branching model and a Markov chain model to measure the system dynamics of a BitTorrent-like file-sharing network. They find that the capacity of such systems grows exponentially in transience and stabilizes at a steady state. Their work is extended in Qiu *et al.* [59], where a fluid model is presented to quantify system capacity at a steady state. Closed expressions of performance metrics are obtained from numerical results. Ramachandran *et al.* [60] present an analytic framework based on a queuing model to evaluate the performance of P2P file-sharing systems in terms of the downloading rate. Clevenot *et al.* [61] propose a multi-class fluid model for BitTorrent-like content-distribution systems. The model leads toward a non-linear system of differential equations with special structure. Clevenot *et al.* prove that the system

of differential equations admits a unique stable equilibrium and computes in closed form. There is also recent work in modeling the time for file-sharing systems using the stochastic model in [62]. Authers in [62] have studied the problem for heterogeneous peers with an infinite download bandwidth, both for chunk-based and fluid-based systems. They also derived an explicit expression for the minimum download time in a general heterogeneous fluid system. However, all valuable theoretical results and methodologies are based on file-sharing systems. They could tell us that P2P networks can be suitable for file sharing in a large scale environment with low costs, but they cannot directly point out whether the approach is also feasible for P2P streaming for those particular challenges discussed in Chapter 1.

To consider modeling P2P streaming systems, only a few results have come out so far. In the work of [13], without explicitly mentioning fountain codes or rateless codes, the authors present an analytic framework for video streaming over peer-to-peer networks. However, the work of [13] ignores the requirement that video content needs to be received and decoded in playback sequence and treats the video-streaming system as being equivalent to a video file-distribution system. Additionally, in [13], the authors set up their problem as finding the "rate-saturating point" of the system, i.e. finding the time at which the user arrival rate equals the "supplying rate" of the system. When the system does not reject any requesting users, we argue that the "rate-saturating point" is not the true "saturating point" of the system. Multi-file scenarios are also discussed in [13]. And this work lastly presents how the system achieves optimal allocation of server bandwidth among different media objects in terms of the "rate-saturating point" and extends the results for peer failures by using the concept of peer lifespan. The work gave a very good point for analyzing the dynamics of the P2P streaming system. However, the following assumptions made in the work are not practical.

- Video streaming sessions are treated as video file distribution. The key difference between the two systems is that playing while downloading is in the video streaming system, but it is not the case in the file-distribution system. In other words, peers

may exit the system if the downloading rate is too low to play in the video-streaming system, but peers may stay to finish downloading a file even if the downloading rate is low.

- The "rate-saturating point" is not accurate because there are peers that are on the waiting list.
- An average uploading rate is used, which is not enough for heterogeneous peers.

In 2007 Kumar *et al.* [14] developed a simple stochastic fluid mode for P2P streaming systems. The authors classify peers into two classes by peer uploading bandwidth and treat each class of peers as two independent Poisson processes. Then they use a fluid model to characterize P2P streaming systems with infinite downloading bandwidth. In [14] they define the concept of universal streaming based on the achievable downloading rate and derive the closed-form expressions of the system performance in terms of the probability of universal streaming. This is the first model to consider peer churn for P2P streaming systems; however, it only gives analysis for the steady state of the system. Further, the work treats video streaming as one session for every connection or flow likely; in fact, video streaming is usually transmitted by blocks, i.e. there may be two different connections (sessions) for two video blocks sent to a peer. Thus, the analysis results could not characterize one important QoS measurement jitter. Also, the proposed model did not mention the help of the buffer. Kumar *et al.* have not investigated the limits of the system in this work. Considering the target of the thesis, the work has not provided an effective approach.

3.2 P2P Live Streaming Systems

The Narada [63] at Carnegie Mellon University is the first generation overlay multicast system. The Narada protocol takes two steps to implement an overlay transmission network: first it establishes an overlay mesh connecting nodes in the multicast group;

then it builds a data multicast tree by using one of the distance vector routing algorithms to select paths between the multicast source and the receivers. The routing algorithm is similar to the approach taken in DVMRP [64]. Therefore, Narada is based on single-tree multicast to distribute content and on an overlay mesh to maintain the tree. The distance vector routing module may take rate, delay, and a customer-tailored weight as the metrics to achieve the optimal performance in throughput. However, it completely ignores video content distributed in blocks and sequence. The problems caused are no QoS such as jitter control and no ability to support VoD applications.

NICE [65] has been developed at the University of Maryland for large-scale communication groups over the Internet. The major target of NICE is to achieve scalability and robustness, otherwise known as data throughput and delay. The idea that is different from Narada is that NICE organizes peers into a hierarchy of clusters, then forms a multicast tree on top of a cluster hierarchy. Therefore, the data multicast tree also belongs to a single tree-based multicast system. However, NICE can manage the control overhead of handling data routing and nodes joining and leaving between $O(k)$ and $O(k \log(N))$, where k is a parameter of the size of the cluster and N is the total number of peers. The size of a cluster is between k and $3k - 1$. Clusters are grouped into different levels (or layers) in the hierarchy. The total number of layers in a NICE network is $\log(N)$. The way layers are formed is that all clusters form layer 0, then all head nodes from layer i form layer $i + 1$ of the hierarchy, as each cluster has a head node. Then a data path from the multicast sender transmits up and then down the topology hierarchy. It may have better performance than Narada, but clearly, the basic method is similar. It does not have any QoS control, and it is not suitable for VoD either.

Zigzag [20] also clusters peers into a hierarchy, namely the administrative organization, then forms the data multicast tree atop this hierarchy. The idea is similar to NICE [65], but the methods to construct and maintain the data multicast tree is different. In a Zigzag network, each cluster has a head node and an associate head node. And the associate head is used to forward the content to other members. In a NICE [65]

network, there is not an associate head node in each cluster; the data are forwarded by the head of a cluster. As compared in [20] to NICE, Zigzag has better performance in terms of the control overhead. However, Zigzag is also a single tree-based multicast system considering its data multicast tree. Zigzag has a good ability for live streaming applications, but it still cannot guarantee QoS (jitter) control since the video stream is still treated by a flow-based model. Each peer only does relays or forwards; no block sequence concepts have been discussed. Thus, it is not feasible for VoD applications.

After considering that a single multicast tree may not be feasible due to the fact that an internal tree node (end host) is not stable to forward content, Castro *et al.* propose SplitStream in [50], which addresses a balanced multi-tree overlay multicast design. SplitStream splits the source data into multiple blocks and employs a multicast tree for each block. SplitStream uses protocols such as Pastry [30] to select paths from the sender to receivers and then combines them as a multicast tree by Scribe [66] so that it ensures that each multicast tree is interior-node-disjoint to the others. Here interior-node-disjoint means that each node is an interior node in at most one tree and a leaf node in the other trees for a set of trees. This work has considered dividing the video stream into multi-blocks, but the purpose is to use multi-trees to broadcast each block to achieve maximum throughput. In terms of continuity, this type of mechanism is not really feasible since being synchronous among multi-trees causes big problems.

CoopNet [67] is another example of employing multiple multicast trees in application layer content distribution. CoopNet aims to stream live or on-demand multi-media to a large group of end users on a lossy network. It uses a centralized algorithm (running on the server) to build the trees that is different than SplitStream and further utilizes the bandwidth contribution of peers to decrease the bandwidth burden at the server. Similar to SplitStream, CoopNet also sets up multiple distribution trees. However, it uses MDC (Multiple Description Coding) codes to encode data, then transmits a subset of encoded data along each tree. A nice feature of MDC is that a node with restricted incoming bandwidth may choose to receive a subset of all the descriptions while still being able to

play the media stream at a decreased quality level. Its design purpose is to support a certain media data rate with good scalability and robustness, similar to that of Narada and NICE. This is different than the target of this thesis, where we try to achieve the maximum rate to support high-quality video stream with jitter control. We believe that CoopNet would be greatly improved by simply increasing the buffer size of each peer. However, CoopNet does not offer this advantage for its network topology.

Bullet has been proposed by Kostic *et al.* [52] to improve end-to-end multicast throughput by employing a multicast mesh instead of a multicast tree. In a Bullet network, the data of the source is partitioned into multiple blocks at the sender. Then distinct data blocks are sent out to neighbours. To recover the original data from different neighbours at a receiver, the Bullet first locates disjointed data blocks using RanSub [68]—an approach to distribute uniform random subsets of global state to all users, and then exchanges missing data blocks between the pair of users by applying the approximate data reconciliation techniques proposed in [69]. The proposed algorithm of data reconciliation is complex and inefficient. Although the authors set "maximizing the amount of bandwidth delivered to receivers" as one of the design objectives, they do not clearly demonstrate this point, since the design lacks of a maximum-flow infrastructure. Considering rateless coding and network coding, the approach should be improved. Also, the focus of this work is to distribute live streaming; it is not practicable for VoD applications.

3.3 P2P on-Demand Streaming Systems

An on-demand media streaming solution is proposed in [70] by combining media segmentation and rateless encoding with Raptor codes. The focus of the authors is on reducing coding complexity by minimizing server bandwidth consumption, but they do not explore the useful properties related to the ratelessness to improve overlay topology.

Cui *et al.* propose an asynchronous multicast scheme for overlay video on-demand,

namely oStream in [16]. The focus of oStream is to design a distributed algorithm to construct/maintain a media multicast tree to minimize server bandwidth cost and link cost by using each peer's capability of buffer and uploading, but not employ any encoding/decoding. Therefore, it belongs to the single tree model to implement video-on-demand. In [71], Cui *et al.* also propose a layered-encoding streaming scheme for on-demand systems over the application layer to solve heterogeneous receivers in the request rate. Authors show that it is an NP-complete problem to find an optimal solution to maximize every receiver's downloading rate with constrained input bandwidth. Therefore, the solution proposed in [71] is a heuristic algorithm to construct a multicast tree based on layered-encoded media.

Annapureddy *et al.* [22] propose a mesh-based P2P on-demand scheme. They also employ network coding in their works. However, their works are mostly based on empirical studies. The focus of their work is to evaluate such P2P systems for near video-on-demand applications to understand some key design problems for such systems.

Dana *et al.* [72] also present a hybrid server/P2P streaming system called BitTorrent-Assisted Streaming System (BASS) for large-scale video-on-demand services. The main idea is to reduce the server bandwidth cost by using peer helping. Some recent works are also proposed in [17,18]. All of those works have not demonstrated that they are feasible for distributing high-quality video service with QoS—jitter control under heterogeneous receivers, which is the key challenge of this thesis.

P2Cast [73] uses patching techniques to form all users into a base streaming tree and a patching session for each user. The single parent approach simplifies the above two problems in P2Cast, but it is unable to cope with the network bandwidth fluctuation and the parent failure problem.

P2VoD [74] has proposed generation concepts to organize supplying users into different generations by their playback position. User departure or link failure is efficiently handled by finding another supplying user in the older generation. With each user having fixed bound on the amount of buffered most recent video content, each newly arrived user

can quickly join the system by either joining the youngest generation or creating a new generation. However, it assumes that all new users play the video from the beginning. It does not provide any efficient mechanisms for user VCR operation and random access either. Moreover, the cost of locating and managing generations is not small at all.

Recently proposed VMesh [28] seems to be one of the most practical protocols for P2P VoD services with VCR operation support. The key idea of VMesh is that each user in VMesh stores a few segments of the video content, which are independent of what the user is playing, so that the user's VCR operation does not affect its children for content delivery. Based on the distributed storage approach, all supplying users have been formed into an overlay network, namely VMesh. When a user joins in or performs VCR operation, the user can quickly locate supplying users by using distributed DHT technology [29] in $O(\log N)$, where N is the number of users in the overlay network. To play the continued video segment, the user needs to switch supplying users, because the current supplying users may not have stored the next video segment. The authors also proposed a list of neighbours who possess the next/previous/current segments to improve searching for the frequently switching supplying users issue. A segment population-based storage scheme and the distributed population estimation algorithm have been proposed to improve the performance in terms of the server stress. However, every user has to often change its suppliers in VMesh, which is an issue when the network is not stable. It may cause two problems: every user may suffer network jitter problems and the cost of maintaining content delivery network is high. Another ignored fact about VMesh is that the appropriate segments the user can store are limited to the segments that the user has played. This limits every user's choice to store segments when the segment population based scheme is applied. As a result, VMesh still does not provide an efficient way to support user interactivity.

3.4 P2P Streaming Using Coding Technology

In [75], authors apply network coding for P2P content distribution. The focus of the work is to minimize the downloading time for large data file dissemination. A similar project has been developed in the Avalanche project [3] by a Microsoft research team. By applying network coding to each peer, it aims at low scheduling overhead among peer receivers for data reconciliation and hence high scalability and cost effectiveness. Based on the exact knowledge of network topology, Chou *et al.* [76] propose a practical transmission scheme with network coding, where no topology information is assumed. Their work focuses on detailed networking system design, including packet buffering and coding support in packet headers.

Zhu *et al.* in [77] apply network coding to the application layer content distribution protocol design. In [77], the authors propose to construct a multicast graph instead of a multicast tree to multicast stream. They also take two steps to form the multicast graph: first they construct a density-connected overlay network (called a rudimentary graph); then they form it to the multicast graph (called a k-redundant graph) within the rudimentary graph, where k means that each receiver input degree is k. To apply the advantage of networking coding, they introduce a set of intermediate nodes into the constructed multicast graph. They prove that encoding/decoding are only needed at source nodes and intermediate nodes to apply network coding. Consequently, the authors design a distributed algorithm to assign linear network codes to the overlay nodes so that every receiver is able to recover all original data flows.

Recently, Wang *et al.* [78] implemented an overlay multicast infrastructure based on network coding to achieve maximum end-to-end throughput. The idea is that they first employ an ϵ-relaxation-based [79] distributed algorithm to compute a *minimum-cost* maximum-rate multicast topology and then apply random linear network codes to each node to assemble incoming flows for each outgoing link. Their work shows that the infrastructure can achieve high throughput with the knowledge of link capacity.

However, both works are not feasible for P2P streaming when we note that every user is the receiver without extra intermediate nodes helping and each user may join and leave randomly. As Wang *et al.* [78] propose, data transmission topology and network codes become dynamic, since users join and depart. Therefore, the target of the works is different than ours, through their algorithm may help us design.

In [15] and [80], the authors propose using fountain codes and, more generally, rateless codes for video transport in a peer-to-peer network, namely rStream. They demonstrate the performance advantage of rateless codes in such applications. However, they primarily consider a fixed number of peer users who *have already arrived* in the system, and the dynamics of user arrival and its impact on network formation are ignored. In addition, the work of [80] also investigates and presents distributed algorithms for optimal flow assignment in the network subject to bandwidth constraints —- mainly under the live streaming setting. We note that the problem considered in this thesis is in the setting when users are arriving in the system and demand a video for nearly instantaneous playback. This problem appears to be more difficult compared with the setting of [15] and [80], where only a static scenario is considered.

Similar to rStream, PeerStreaming [81] also employs a high-rate erasure code based on overlay mesh topologies. In PeerStreaming, every peer selects a set of peers with complete or partial cached media (namely serving peers) based on heuristic algorithms and then downloads the encoded packets of media to decode using erasure coding instead of the segment-scheduling algorithm. The high-rate erasure code is a modified Reed-Solomon code on the Galois Field GF(216) for content reconciliation among peers. For a PeerStreaming network, the key step is to locate a proper set of serving peers to guarantee the downloading rate, but the method presented does not clearly show how to find serving peers without the global knowledge of peers. Another problem is that it is unclear how to set serving peers. It seems that the set of serving peers is pre-configured or a peer becomes a serving peer after it receives a complete media file, which is not feasible for live streaming systems or video on-demand systems with large video files.

3.5 Commercial Systems

CoolStream [19] is one of the commercial P2P streaming systems. An Internet-based implementation of CoolStream was released on May 30, 2004. Its service had stopped on June 10, 2005, due to copyright issues. However, as reported in [19], the system has attracted over 30000 distinct users with more than 4000 being online simultaneously at some peak times. CoolStream is a data-driven overlay network for live media streaming. In a CoolStream network, a partnership management algorithm based on SCAM (Scalable Gossiping Membership Protocol) in [49] and a heuristic scheduling algorithm have been introduced to achieve real-time and continuous distribution of streaming contents. The idea is that each peer periodically exchanges media availability with its partners, and the media segments to be retrieved from each partner are dependent upon the number of potential suppliers for each segment and the available upload capacities of partners, but no coding techniques are employed at both the source side and peers. With regard to performance measurement in [19], CoolStream has good performance in terms of playback continuity for TV quality streaming, with a large number of users attending. And the control overhead is also reasonably low, that is, about 1% of total streaming traffic. However, the QoS has not been considered in the system.

PPLive [8] is another recently popular P2P IPTV system that is from China. As reported in [12], it successfully broadcasted the 2006 Chinese New Year's celebration to over 200000 users over the Internet at a bit rate of $400 - 800$ kbps range. However, its detailed design is not public yet. Several researchers have done measurements of it by monitoring the Internet and have discovered that its efficiency is not good in terms of control overhead and neighbour peer organizations. More similar commercial P2P IPTV broadcasting systems can be found on the Internet; some of them are TVants [9], VVSky [82], ppStream [10], and FeiDian [11]. Because there are no valuable published articles related to their design, we cannot analyze them here. We are mostly sure that all of them have a QoS control problem via our experiment.

Chapter 4

P2P Streaming System Modeling and Theoretical Analysis

In this chapter, A multi-video-block transport framework for P2P streaming systems is introduced. Based on the proposed framework, we define two new concepts, *production dynamics* and *saturation time*, to characterize the performance of the system. Then the bounds of the system in terms of the *production* of the system are derived when we note that we could design a protocol to implement the arbitrary allocation of supplying bandwidth using rateless codes with a small constant overhead.

4.1 System Model

The system we consider includes a server $\mathcal{U}_0$ that has a video file and a set of users $\mathcal{U}_1, \mathcal{U}_2, \ldots$. The indices of the users $\mathcal{U}_i$ are in the arrival order of the users; namely, if $i < i'$, then $\mathcal{U}_i$ arrives earlier than $\mathcal{U}_{i'}$. The arrival of users and departure of users are stochastic processes in nature. Every user may arrive or depart at any time. However, we are more interested in the average cases here, not the performance of each individual user. Thus, a deterministic extended "fluid model" based on the model proposed in [59] — rather than a Poisson model — is used to model the arrival of users. Specifically, we

model the number $N(t)$ of users having arrived at the system by time t as a real-valued function $N(t) = \lambda t$ for positive arrival rate λ. The simulation for the proposed Protocol A in this chapter, in which a Poisson process has been used, suggests that the results validate the "fluid model."

On the other hand, to characterize the network formation in close forms, we assume that users will keep serving buffered content to others once they arrive. The assumption is not practical, but we argue that the results can be extended to more realistic network settings because the departure of users will just cause the linear solution to become non-linear. The performance of measurements by the notions will still be valuable.

The objective of the streaming system is to provide video-streaming service to all users so that, shortly after a user arrives in the system and requests the video, it can play the video continuously without interruption.

The video file is divided into M consecutive blocks, each having size A bits. We will use $\{1, 2, \ldots, M\}$ to index these blocks, where block m contains video content preceding block $m+1$, for every $m \in \{1, 2, \ldots, M-1\}$. The playback rate of the video is assumed to be β bits/second, i.e., each block when playing takes A/β seconds to finish. The total size of the video file is then MA, and the total playback duration (or length) of the video is MA/β.

4.2 Multi-Video-Block Transport Framework

We consider that a video file is divided into small blocks to be distributed efficiently, in which the scheme of dividing the video file is flexible, for example, a simple way is to divide the video file into the same small size of blocks. We derive a fluid model-based framework extended to support this multi-block scenario. We note that most proposed analytical frameworks are the fluid model based on the rate saturation [13, 62]. First, based on this, the extended multi-block framework is presented in Figure 4.1. In this framework, the input parameters are the requesting rate (bandwidth)($R(t)$) of arriving

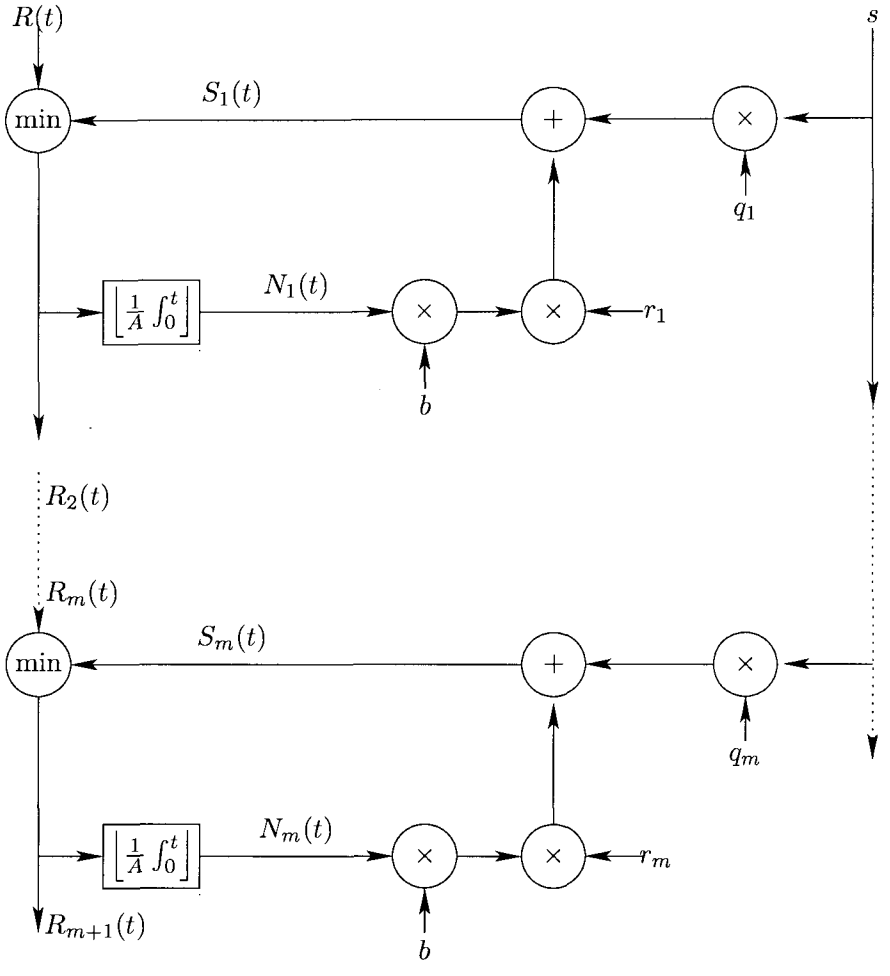


Figure 4.1: Rate saturation-based multi-block video streaming framework.

users and server output bandwidth (s). The output of the model is the available output bandwidth ($R_m(t)$) for each video block. The control parameters are every user's output bandwidth (b) and allocation factors (r_m,q_m) for each video block, which are set by the designed protocol. To present easily, we introduce a symbol here: $S_m(t)$, the potential supplying bandwidth of the video block-m at time t. The framework shows well the principle of relationship between output and input parameters with control parameters.

However, we argue that the rate saturation point is not accurate for the dynamic system. When the system allows arriving users to wait for admittance, the true saturation point should be based on the number of users that finish receiving the video blocks.

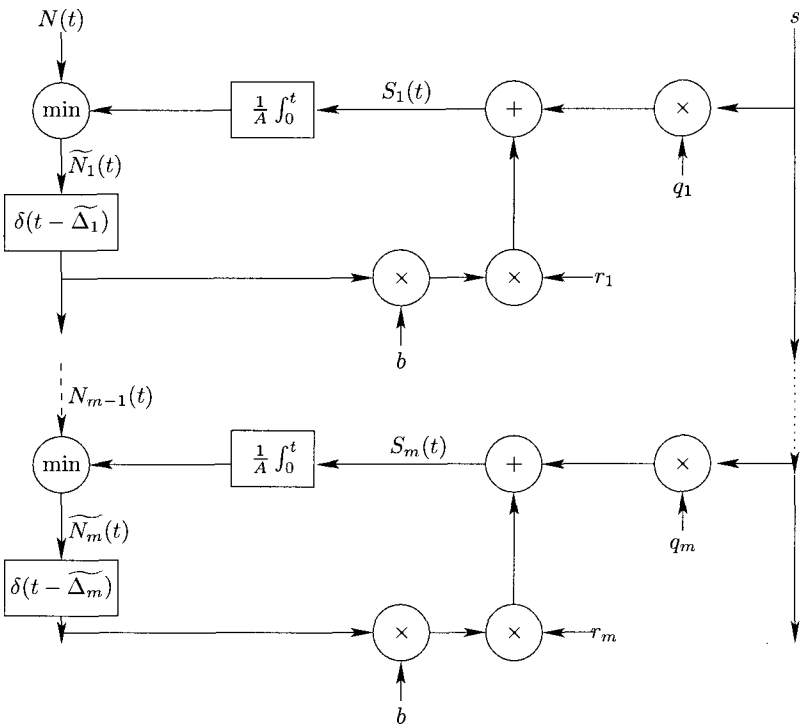


Figure 4.2: Production saturation-based multi-block video streaming framework.

Therefore, we further extend the framework to the *production* (defined later) saturation-based model shown in Figure 4.2. For this framework, the output of the system becomes the number of users that we defined as production for each video block. The input parameters are $N(t)$ the number of arrived users at time t and the server bandwidth. The control parameters are the same as the rate saturation-based framework. For easy presenting, we introduce another notation: $\widetilde{N_m}(t)$, the potential number (production) of users with the video block-m at time t.

4.3 Production and Saturation Time

Borrowing a term from economics [83], we now introduce the notion of *production* to characterize the performance of such a system under any given transport protocol [84].

Definition 1 (Production) *For any given network and under any given video-streaming*

protocol, the production $N_m(t)$ of block m at time t is defined as the number of users that have obtained block m by time t.

Due to the stochastic nature of the network, production $N_m(t)$ is in general a random process. However, as we are mostly interested in the average behaviour of $N_m(t)$, we treat $N_m(t)$ again as a real-valued deterministic function.

It is clear that $N_m(t)$ is upper bounded by $N(t)$ for every m. Thus, how rapidly and closely $N_m(t)$ approaches $N(t)$ indicates the system efficiency for distributing the m^{th} block. We note that, unlike most other performance metrics in literature, which characterize the performance of the streaming system under equilibrium conditions, the notion of production proposed here intends to characterize the performance dynamics of the system as it evolves with the equilibrium.

As the users arrive in the system, there is an overlay network, established on top of the IP network, between each user and the server and between every two users. We assume that , in the overlay network, a user may acquire a demanded video block either from the server or from any other user. The overlay network is characterized by homogeneous loss rate γ. More specifically, each logical link between two users or between the server and a user assumes an independent state for each transmitted packet, such that the transmitted packet suffers from packet loss with probability γ. We note that the homogeneous assumption of loss rate has no particular significance in the forthcoming analysis. In addition, we assume that the loss rate is unknown to the server and to all users.

Definition 2 (Production Potential) *Under any given video-streaming protocol, the production potential $S_m(t)$ of block m at time t is defined by*

$$S_m(t) := \frac{1-\gamma}{A} \int_0^t \left(s_m(\tau) + b_m(\tau) N_m(\tau)\right) d\tau,$$

where $s_m(t)$ and $b_m(t)$, depending on the protocol, are respectively the server's output bandwidth allocated to transmitting block m at time t and the average user output band-

width allocated to transmitting block m at time t, where the average is the overall users having obtained block m.

As each user having obtained block m may become a potential supplier of block m to other users, the notion of production potential, $S_m(t)$, indicates the possible production of block m up to time t if the production were not limited by the number of users needing block m by time t. In this definition, it has been implicitly assumed that each user has a sufficient buffer so that a user having obtained block m can be a "permanent" supplier of block m.

Now, we assume that, after a user has obtained block $m-1$, the user will immediately request block m. That is, $N_{m-1}(t)$ is also equal to the number of users needing block m at time t. Then the following notions of "saturation" and "unsaturation" are natural.

Definition 3 (m-UNSAT and Saturation Time) *At any time T, the system is said to be block-m-unsaturating, or m-UNSAT, if for any $t \in [0, T]$,*

$$N_{m-1}(t) > S_m(t), \text{ or } N_{m-1}(t) = S_m(t) = 0.$$

where $N_0(t) := N(t)$. The saturation time $\widehat{T}_m$ *of block m is defined as*

$$\widehat{T}_m := \sup\{T : \text{ the system is } m\text{−UNSAT at } T\}.$$

When $N_{m-1}(t)$ is interpreted as the number of users that have requested block m since time 0, the rationale of coining the term "block-m-unsaturating" is to indicate, except for the extreme case of $N_{m-1}(t) = S_m(t) = 0$, the state of the system where there have been more users requesting block m than the number of users that the system could have supplied. It is clear by the definition that the system is m-UNSAT at $t = 0$ for all m. Then the saturation time $\widehat{T}_m$ of block m is the phase-transition point for the system to leave the m-UNSAT state. It is then clear that for a well-behaved protocol, one may want the saturation time to have the following properties.

- Saturation time $\widehat{T}_m$ is finite for every m and as small as possible.
- At time $t > \widehat{T}_m$, $N_m(t)$ is close to λt; namely, every user arriving after time $\widehat{T}_m$ obtains every block very quickly.

For the rest of this chapter, we assume that the total output bandwidth of the server is s bits/second and the output bandwidth of each user is b bits/second. No restriction on the downloading bandwidth of each user, which allows for a simplification of analysis, is imposed. Additionally, we ignore all other possible overheads; for example, at every time t, we assume that each user $\mathcal{U}_i$ is instantaneously updated with the information concerning which other users have the video block that user $\mathcal{U}_i$ demands.

All symbols used for this thesis are summarized in Table 4.1.

4.4 A Lower Bound of Production: Protocol A

For the ease of analysis, we assume in this section that the buffer size K of each user is sufficiently large that all decoded video blocks can be stored.

As was discussed in section 4.1, in this section we simplify the setting by considering that users arrive deterministically at a constant rate of λ users/second. That is, we consider $N(t)$ as a deterministic function of time defined as

$$N(t) = \lambda t, t \geq 0, \tag{4.1}$$

and allow $N(t)$ to take non-integer values.

Similarly, for each m, we will consider the production $N_m(t)$ as a real-valued (instead of integer-valued) function, where if a user has obtained αA bits of block m by time t ($\alpha \in [0, 1]$), they contribute to $N_m(t)$ as α users. One implication of this assumption is that we assume the transmission protocol used, such as fountain codes used with very small overhead, has zero overhead, so that as long as A bits of block m are received,

Table 4.1: Notation and symbols

m	Index of video blocks
i	Index of arrival users
M	Number of blocks of the video file
A	Size of the video block in bytes
$\mathcal{U}_0$	Video server
$\mathcal{U}_i$	Users that arrive in the system ordered by i
λ	Arrival rate of users
$N(t)$	Number of arrived users at time t
$N_m(t)$	Number (production) of users with the video block-m at time t
$\widetilde{N_m(t)}$	The potential number (production) of users with the video block-m at time t
$S_m(t)$	Potential supplying bandwidth of the video block-m at time t
$R(t)$	Requested bandwidth of arrival users at time t
$R_m(t)$	Requested bandwidth of users for the video block-m at time t
a	Average input bandwidth of users
b	Average output bandwidth of users
s	Total output bandwidth of server
q_m	Factor of allocated server output bandwidth for users requesting the video block-m
r_m	Factor of allocated output bandwidth of users with the video block-m for supplying
$\widetilde{\Delta_m}$	Delay in which users with the video block-m become requesting users for the video block-$m+1$
γ	Overlay network loss rate

there is no need to receive extra number of bits -- a generalization to arbitrary overhead is a straightforward exercise.

We also assume that users have no restriction in their downloading (or input) bandwidth. This assumption, although invalid for practical systems, simplifies the analysis and design of transport protocols, serving the purpose of finding the lower bounds of production.

4.4.1 Bounds on Production and Saturation Time

Based on our production saturation-based P2P streaming fluid framework, we derive the following analytical results. Concerning saturation time, we assume that, for any reasonable protocols,

$$\lambda \widehat{T}_1 > e. \tag{4.2}$$

This restriction essentially assumes that, by the time the system leaves the 1-UNSAT phase, at least three users have arrived. This assumption evidently holds true for all practical scenarios under any reasonable protocols. In fact, the only cases in which this assumption fails are when the arrival rate is very low and/or the server has sufficient output bandwidth to drive the system out of the 1-UNSAT phase quickly; in both of the cases, one, however, would argue that there is simply no need to use any peer-to-peer approaches.

Under the above assumptions, we have the following theorem, in which $\mathbf{u}(t)$ denotes the unit-step function, namely, $\mathbf{u}(t) = 1$, if $t \geq 0$, and $\mathbf{u}(t) = 0$, if $t < 0$.

Theorem 1 *Under assumption (4.2), there exists a transport protocol such that*

1. $N_m(t) \geq \min\left(\lambda t, e^{\frac{b(1-\gamma)}{AM}(t-T_m)}\mathbf{u}(t-T_m)\right)$,

   for all $m \in \{1, 2, \ldots M\}$, *where* $T_m = mA/(1-\gamma)s + 1/\lambda$;

2. $\widehat{T}_m$ *is finite for every* m.

4.4.2 Proof: Protocol A

We prove Theorem 1 by constructing a protocol, which we call Protocol A, that achieves production functions no lower than the lower bounds in Theorem 1. It will become evident that the essence of Protocol A is to induce linear-system dynamics (over a certain time window) so that analytic solutions can be easily derived.

In Protocol A, after the first user, $\mathcal{U}_1$, has completely obtained block 1, server $\mathcal{U}_0$ transmits block 2 to $\mathcal{U}_1$. The server continues in this way until $\mathcal{U}_1$ has obtained block M. Then the server terminates the transmission and stays idle. Each user $\mathcal{U}_i$ allocates $1/M$ fraction of its output bandwidth for distributing block m for every m. Every user $\mathcal{U}_i$ requests video blocks in sequence, and immediately after receiving a request of the block that it has already obtained, it uses its allocated output bandwidth to transmit the block, coded by its fountain code, to the requesting user. Here we assume that fountain codes are used to implement arbitrary bandwidth allocation because of their amazing features of rateless codes. And we also assume that the coding overhead is small so that we can treat it as no coding overhead. In fact, the assumption here is that the overhead of Protocol A for content reconciliation among receivers from one sender are zero or very small. We will use $N_m^A(t), S_m^A(t)$ and $\widehat{T}_m^A$ to denote the corresponding quantities achieved by Protocol A.

We note that in Protocol A the time by which exactly one user has obtained block m completely is T_m or $mA/(1-\gamma)s + 1/\lambda$ in the theorem. It follows that for every m it is always $\mathcal{U}_1$ who obtains block m first, that it takes $1/\lambda$ seconds for $\mathcal{U}_1$ to arrive in the system, and that it takes $A/(1-\gamma)s$ seconds for $\mathcal{U}_1$ to obtain each block. The following observations are in order.

1. The system is m-UNSAT at every $t \leq T_m$. It simply follows that $N_m^A(t) = 0$ at any $t < T_m$ and $S_m^A(t) = 0$ at every $t \leq T_m$.

2. If the system is m-UNSAT at time t, then

$$N_m^A(t) = S_m^A(t). \tag{4.3}$$

This is due to an idealistic assumption that if the number of users that the system can supply with block m up to time t is more than the number of users that have requested block m up to time t, all output bandwidth allocated to block m will be completely used for supplying users requesting block m. Thus, at the m-UNSAT phase, the number of users having received block m by time t is equal to $S_m^A(t)$.

These observations give rise to the following lemma.

Lemma 1 *If the system is block-m-UNSAT at time t, then*

$$N_m^A(t) = e^{\frac{(1-\gamma)b}{AM}(t-T_m)}\mathbf{u}(t-T_m).$$

Proof: Clearly for $t < T_m$, $N^A(t) = 0$. At $t \geq T_m$, if the system is m-UNSAT, applying (4.3), we have

$$N_m^A(t) = \frac{(1-\gamma)b}{MA}\int_0^t N_m^A(\tau)d\tau. \tag{4.4}$$

Solving this equation with the initial condition $N_m^A(T_m) = 1$, we get

$$N_m^A(t) = e^{\frac{(1-\gamma)b}{AM}(t-T_m)},$$

for $t \geq T_m$. Combining the case of $t < T_m$ and the case of $t \geq T_m$, the lemma is proved. □

We note that, during the time period in which the system is m-UNSAT, $N_m^A(t)$ behaves according to linear-system dynamics following (4.4).

Lemma 2 *For all* $m \in \{1, 2, \ldots, M\}$, $\widehat{T}_m^A$ *is finite, and*

$$N_m^A(t) = \begin{cases} 0, & t < T_m \\ e^{\frac{(1-\gamma)b}{AM}(t-T_m)}, & T_m \leq t \leq \widehat{T}_m \\ \lambda t, & t > \widehat{T}_m^A(t) \end{cases}$$

Proof: We will prove this result by induction.

First consider the case of $m = 1$. Clearly $T_1 < \widehat{T}_1^A$. During time $[T_1, \widehat{T}_1^A)$, the system is 1-UNSAT. Thus

$$N_1^A(t) = e^{\frac{(1-\gamma)b}{AM}(t-T_1)},$$

during $[T_1, \widehat{T}_1^A)$. In addition, $1 = N_1^A(T_1) < \lambda T_1$, i.e., the exponential curve at $t = T_1$ lies below line λt. Then as t increases, the exponential curve $e^{\frac{(1-\gamma)b}{AM}(t-T_1)}$ and line λt must intersect. Furthermore, by definition of saturation time, $\widehat{T}_1^A$ is precisely the intersecting point of the two curves. Then we conclude the $\widehat{T}_1^A$ is finite.

Let $\Delta > 0$ be arbitrarily small. Since it is always true that

$$N_0^A(t) \geq N_1^A(t),$$

$$N_1^A(t) = N_0^A(t) = \lambda t$$

over interval $[\widehat{T}_1^A, \widehat{T}_1^A + \Delta]$.

Then when $t \in [\widehat{T}_1^A, \widehat{T}_1^A + \Delta]$,

$$S_1^A(t) = \lambda \widehat{T}_1^A + \frac{(1-\gamma)b}{MA} \int_{\widehat{T}_1^A}^{t} \lambda \tau d\tau$$

gives rise to

$$\frac{dS_1^A}{dt}(t) = \frac{(1-\gamma)b\lambda t}{MA}$$

at every $t \in [\widehat{T}_1^A, \widehat{T}_1^A + \Delta]$.

Now it is possible to show that

$$\begin{aligned}\frac{dS_1^A}{dt}(\widehat{T}_1^A + \Delta) > \frac{dS_1^A}{dt}(\widehat{T}_1^A) &= \frac{(1-\gamma)b}{MA}\lambda\widehat{T}_1^A \\ &= \lambda\left(\ln(\lambda\widehat{T}_1^A) + \frac{bA}{sM} + \frac{(1-\gamma)b}{MA\lambda}\right)\end{aligned}$$

when considering $\lambda\widehat{T}_1^A = e^{\frac{(1-\gamma)b}{MA}(\widehat{T}_1^A - T_1)}$, which results in

$$\frac{(1-\gamma)b}{MA}\widehat{T}_1^A = \ln(\lambda\widehat{T}_1^A) + \frac{(1-\gamma)b}{MA}T_1,$$

where $T_1 = A/(1-\gamma)s + 1/\lambda$.

Then following the assumption of equation 4.2, we conclude that

$$\frac{dS_1^A}{dt}(\widehat{T}_1^A + \Delta) > \lambda(\widehat{T}_1^A + \Delta) = \frac{dN_0}{dt}(\widehat{T}_1^A + \Delta) = \lambda.$$

This suggests that, after saturation time $\widehat{T}_1^A$, the potential growth rate of the number of users supplied with block 1 always tends to exceed the growth rate of the number of users having requested block 1. As a consequence, $N_1^A(t)$ is bounded by and stays on $N_0(t)$ or λt.

Now as the inductive assumption, assume the claims of the lemma hold true for a given $m = k$. That is, $\widehat{T}_k^A$ is finite and

$$N_k^A(t) = \begin{cases} 0, & t < T_k \\ e^{\frac{(1-\gamma)b}{AM}(t-T_k)}, & T_k \le t \le \widehat{T}_k^A \\ \lambda t, & t > \widehat{T}_k^A(t) \end{cases}$$

When $m = k+1$, the number of users having requested block-$(k+1)$ is $N_k^A(t)$, which equals λt for $t > \widehat{T}_k^A$. By definition of $\widehat{T}_m^A$, it is clear that the system is $(k+1)$-UNSAT over duration $[0, \widehat{T}_{k+1}^A)$, whether or not $\widehat{T}_{k+1}^A$ is finite. Thus $N_{k+1}^A(t)$ (and $S_{k+1}^A(t)$) takes value 0 when $t < T_{k+1}$ and assumes the exponential curve $e^{\frac{(1-\gamma)b}{AM}(t-T_{k+1})}$, when $T_{k+1} \le t < \widehat{T}_{k+1}^A$. By noting that exponential curves $e^{\frac{(1-\gamma)b}{AM}(t-T_{k+1})}$ and $e^{\frac{(1-\gamma)b}{AM}(t-T_k)}$ will never intersect after

time T_{k+1}, we see that curve $e^{\frac{(1-\gamma)b}{AM}(t-T_{k+1})}$ must intersect with $N_k^A(t)$ when the latter assumes the values on line λt, namely, at some time after time $\widehat{T}_k^A$. This intersection point is, by definition of saturation time, at time $\widehat{T}_{k+1}^A$. Thus, $\widehat{T}_{k+1}^A$ is finite, and clearly $\widehat{T}_{k+1}^A > \widehat{T}_k^A$.

By the same argument, as in the proof for the case of $m = 1$, it can be shown that after time $\widehat{T}_{k+1}^A$, $N_{k+1}^A(t)$ stays on $N_k^A(t)$ or λt. □

Now we are ready to prove the theorem.

Proof: As Protocol A is only a particular protocol using fountain codes, there exists a protocol that performs no worse than Protocol A in terms of production functions. By identifying the lower bound in claim (1) of the theorem as $N_m^A(t)$, claim (1) is proved. It is also obvious that a protocol satisfying claim (1) of the theorem will also give rise to claim (2), since all production curves $N_m(t)$ of such a protocol are on the left side of $N_m^A(t)$, giving rise to saturation time $\widehat{T}_m$ no later than $\widehat{T}_m^A$. This proves claim (2). □

Immediately after proving the theorem, the following result is implied.

Corollary 1 *There exists a fountain-coded transport protocol such that* $\widehat{T}_m \leq \widehat{T}_m^A$, *for all* $m \in \{1, 2, \ldots M\}$.

We argue that Theorem 1 gives a lower bound of performance for such fountain codes-based transport protocols, since it is clear that the protocol A is not optimal, because every user uses a fixed amount of output bandwidth for each block. A transport protocol with a dynamic output bandwidth allocation algorithm should have better performance than Protocol A in terms of production. To verify the results, the simulation has been done in the next section. We also simulated an enhanced protocol (namely Protocol B) that allocates the output bandwidth for each block to compare the bounds dynamically.

4.5 Simulation Results and Discussion

Simulations for Protocol A was performed. In our simulations, to meet the setting of the practical system, the user arrival process is modeled as a Poisson process with λ

users per second. The video server is configured with a high quality of video and can continuously supply to all peers, where the server provides a video on-demand service. The user buffer size K is configured as the same size of the video content. DVD-quality video content is considered, where we set the playback rate β to 3.75 Mbps. The video block size A is fixed to 14 Mbytes or the equivalent to 30 seconds of playback time. The number of video blocks M is simulated to up to 30. Every fountain codeword symbol is assumed to be a UDP packet. The overhead of fountain codes is fixed to 0.03.

The underlying network topology is assumed as a completely connected network. On all links of the network, the packet-loss rate is a tunable parameter γ. For every packet sent on a link, packet loss is simulated as an independent Bernoulli event (among all links and across all packets) parameterized by γ.

As Protocol A mainly serves as a theoretical construction for bounding the system performance, Protocol A is simulated only for the purpose of confirming the bounds presented in Theorem 1 and validating the flow model used in establishing the theorem. We also designed another heuristic protocol, Protocol B, with a dynamic output bandwidth allocation rather than the fixed bandwidth allocation in Protocol A. Simulation results are presented in terms of production and saturation time in the next chapter.

The bounds of Theorem 1 and the simulated production functions achieved by Protocol A are plotted in Figure 4.3 (top), and their derivatives are plotted in Figure 4.3 (bottom). The derivative plots in Figure 4.3 (bottom) serve to demonstrate more sharply the saturation times (corresponding to the sudden drop in the derivative plots) and the instantaneous "supplying capability" of video blocks (equivalent to the instantaneous total output bandwidth, up to scale). From the figure, we see that the theoretical results and simulated results match well, confirming the validity of the theorem and flow model of user arrival. Specifically, the following behaviours are observed. First, during the UNSAT phase, the production grows exponentially until it hits the straight line $N(t)$ characterizing the arrival process whereby the UNSAT phase is terminated, and then the production grows according to $N(t)$. Second, uniform spacing between saturation times

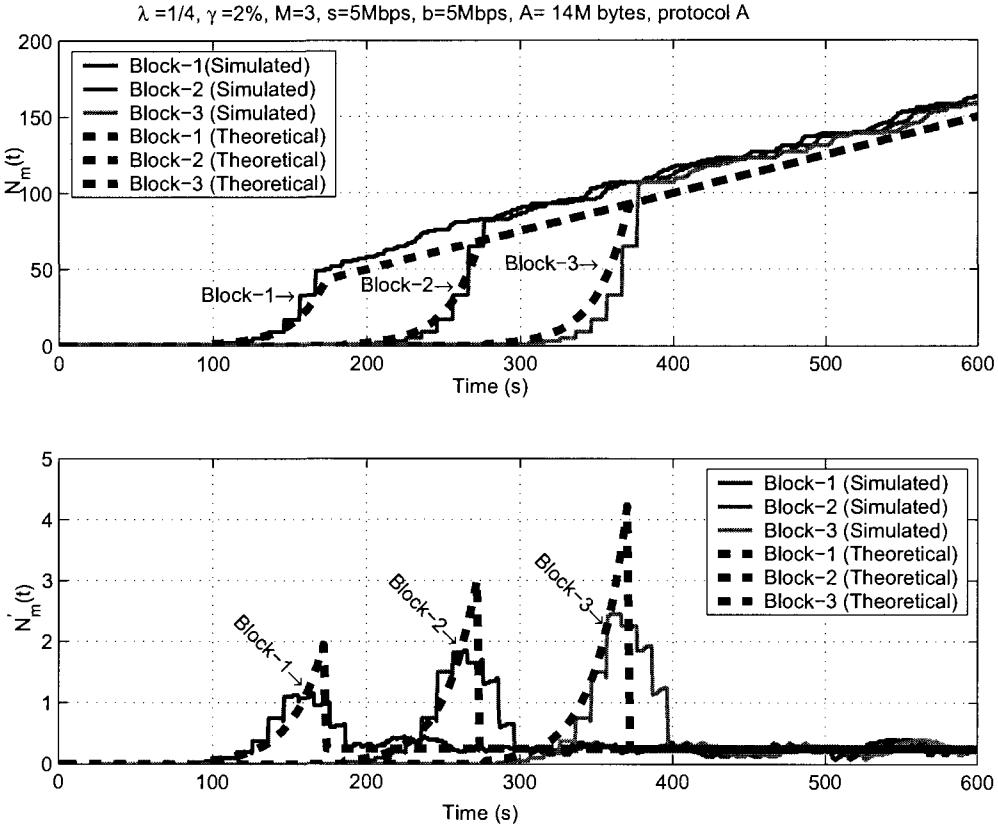


Figure 4.3: Theoretical and simulated production achieved by Protocol A (top) and the derivative of theoretical and simulated production (bottom)

of consecutive blocks is observed.

We also consider the realistic network scenario, where the user departs freely with a limited buffer size. We found it difficult to analytically derive the closed forms of Theorem 1. We believe that the proposed notions of production and saturation time can be extended to realistic network settings. Most of the characters we have derived under our theoretical model are still suitable for realistic networks. Further research will focus mainly on this.

Chapter 5

General P2P Streaming System Design Guidelines

In this chapter, P2P network capacity and QoS control will be discussed to give some guidelines for designing a general P2P system. Several preliminary results are derived.

5.1 P2P Streaming Network Capacity

From our preliminary study on the dynamics and QoS analysis of the P2P video streaming systems, we realized that it is still a hard and open problem to find a general P2P network capacity (i.e. the limitation of the P2P system). However, we found a few preliminary results based on a specific P2P network, which can be guidelines for general P2P system design.

5.1.1 System Throughput

In [85, 86], the authers propose the results of the P2P system throughput (defined in Definition 4) for a simple P2P network, i.e. a star network (shown in Figure 5.1). The P2P network includes a server and N peers, which are connected together by an uplink and some downlinks. The server's downlink has the capacity C_0 (no uplink). Each peer i

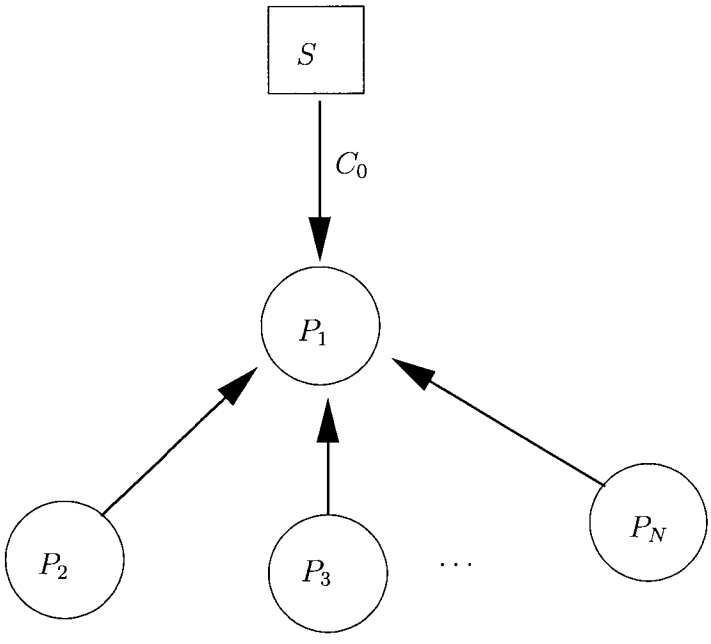


Figure 5.1: Star model of P2P.

($i \in [1, 2, \ldots, N]$) has an uplink of capacity C_i and downlinks with infinite capacity. The maximum system throughput (Theorem 2) has been presented and proved in the works under the following assumptions:

1. The server has infinite content and continuously sends content to the peers.
2. The peers can help the server only by forwarding what they received from the server, i.e. the peers cannot replicate (or generate) information.
3. The peers can unicast only to other peers. They cannot multicast content to the other peers, i.e. every link is independent among peers.

Definition 4 (System throughput) *The amount of content received by all peers per unit time.*

Theorem 2 *Given the star network and the fluid workload, the maximum system throughput is:*

$$C = \min\{C_0, \frac{C_0 + \sum_j C_j}{N}\}.$$

And there exists a two-hop strategy that achieves this throughput.

Proof:

First, we need some notations.

1. Spanning tree usage rate: Let $(s_0(k), s_1(k), \ldots, s_n(k))$ be the unique normalized resource usage ratio for each spanning tree k, where $s_i(k)$ is an integer that represents the usage of link C_i corresponding to all other links $(C_0, C_1, \ldots, C_n)$, and $\sum_i s_i(k) = n$. This definition implies that the system throughput of using a spanning tree is normalized to unity. For instance, if the system throughput is 1, the one-hop spanning tree is $(n, 0, 0, \ldots, 0)$ and the n-hop spanning tree is $(1, 1, \ldots, 1)$.

2. The set of spanning trees are denoted by S, and λ_k is the receiving rate of spanning tree k from the source.

3. A class of spanning trees is the set of spanning trees with the same rate λ_k, and the i^{th} class of spanning trees is denoted by $\lambda_{\{i\}}$. For a simple star network with n nodes, there are n classes of spanning trees in total.

Now we have the following resource constraints equation:

$$\sum_{k \in S} \lambda_k s_i(k) \leq C_i, \forall i \tag{5.1}$$

So the maximum throughput of the system is

$$\max \sum_{k \in S} \lambda_k,$$

for all $k \in S$ under the constraints in Figure (5.1).

In the simple star network, all the uplinks of the peers can be used to provide other peers interchangeably. So we can use another server (server 2) to aggregate all the peers together. The capacity of server 2 is $C = \sum_{j=1}^{n} C_j$.

Then the resource usage rate patterns become

$$(i, n-i) \quad i = 1, 2, \ldots, n$$

The maximum throughput of the system optimization problem thus reduces to:

$$\text{maximize} \quad \sum_{i=1}^{n} \lambda_{\{i\}}$$

$$\text{subject to} \quad \sum_{i=1}^{n} i\lambda_{\{i\}} \leq C_0, \text{ and}$$

$$\sum_{i=1}^{n} (n-i)\lambda_{\{i\}} \leq C,$$

where C_0 is the uplink capacity of the source server.

This is a linear programming problem. The solution can be obtained by the constraints. The maximum throughput is :

$$R = \begin{cases} C_0 & \text{if } C_0 < \frac{C}{n-1}, \\ \frac{C_0+C}{n} & \text{if } C_0 \geq \frac{C}{n-1}. \end{cases}$$

This proves the first part. The existence of a two-hop strategy can be proved by the construction for each case ($C_0 < \frac{C}{n-1}$ and $C_0 \geq \frac{C}{n-1}$). □

The authors found that any coding applied at the peers cannot improve the above bound. The result is presented as Theorem 3 below. The proof is very simple for a star network by considering min-cut to all peers.

Theorem 3 *Given the star network with no coding or multicast (replication) in the network, any coding applied at the peers cannot improve the throughput bound given by Theorem 2.*

The results are good based on the given network setting; however, the assumption is not practical, because each peer in fact stores the content and re-transmits to other peers (i.e. replicates the content) in the P2P network. Also, we are more interested in the server stress that is the outgoing bandwidth required at the video server to support the system and production defined in chapter 4 than to the system throughput. In the study of our proposed lower bound, we observed some results for a P2P VoD streaming system without the above assumption.

5.1.2 Server Stress and Production

We introduce the new term *production* in Chapter 4 to characterize the performance of peer-to-peer video streaming systems. The *production* $N_m(t)$ of the system depends on the communication and transport protocol used. It is also evident that $N_m(t)$ is upper bounded by the number of users that have arrived in the system by time t, which is denoted by $Y_m(t)$ for every m. Thus, how rapidly and closely $N_m(t)$ approaches $Y_m(t)$ indicates the system efficiency for distributing the m^{th} block.

$X_m(t)$ is the number of users who have stored the video block-m at time t. Thus, for any time, $X_m(t)$ is upper bounded by $N_m(t)$.

Now we can examine the relationship between the server stress and the production in P2P video streaming systems. With the same assumptions for deriving the lower bound, the video content at the server denoted by $\mathcal{U}_0$ is divided into M consecutive blocks, each having size A bits. We will use $\{1, 2, \ldots, M\}$ to index these blocks, where block m contains video content preceding block $m+1$, for every $m \in \{1, 2, \ldots, M-1\}$. The playback rate of the video is assumed to be R bits/second, i.e., when playing, each block takes A/R seconds to finish. The total size of the video file is then MA, and the total playback duration (or length) L of the video is MA/R seconds.

We use $\mathcal{U}_i$, $i \in \{1, 2, \ldots\}$ to denote all users in the order of the arriving time, which means user $\mathcal{U}_i$ joins the system earlier than user $\mathcal{U}_j$ if $i < j$. The arrival of users and departure of users are stochastic processes in nature. Every user has a buffer,

whose maximum size is worth K_m units of the video blocks (i.e. A bits) to cache the video content it received. And every user has the maximum uploading bandwidth O bits/second and the maximum downloading bandwidth I bits/second, where there must be $I > R$ to play the video continuously.

We assume the streaming system time t starts at zero, i.e. users start joining at $t = 0$. Then the P2P video streaming system can be presented as the two procedures at any time for every video block m :

The storage allocation process,

$$X_m(t) = F_s(N_m(t));$$

And the transport process,

$$N_m(t) = F_t(X_m(t), Y_m(t)),$$

where F_s refers to the storage allocation protocol and F_t refers to the transportation protocol.

We can note that the two procedures are related. The larger $N_m(t)$ is, the more $X_m(t)$ can be obtained. And the larger $X_m(t)$ is, the more $N_m(t)$ can be generated. However, $N_m(t)$ is upper bounded by $Y_m(t)$, and $\sum_{m=1,\ldots,M} X_m(t)$ is upper bounded by the total number of video blocks $K_s N(t)$, since each user has the limited buffer size K_s units of the video blocks. The purpose of optimizing F_s and F_t is to achieve the minimum server uploading bandwidth (or server stress) while obtaining the maximum production $N_m(t)$ by allocating the storage size in all M video blocks and assigning the uploading bandwidth for all M video blocks.

For each $m = 1, 2, \ldots, M$, we also define

$$S_m(t) := F_t^o(X_m(t)),$$

when $Y_m(t) = \infty$ and given F_t^o is an optimal protocol.

$S_m(t)$ can be interpreted as the total number of users that could have been supplied with block m up to time t if there had been an infinite number of users requesting block m since time 0.

For ease of discussion, we introduce the term "block-m-unsaturating," or m-UNSAT in short: at any time T, the system is said to be m-UNSAT if for any $t \in [0, T]$,

$$Y_m(t) > S_m(t), \text{ or } Y_m(t) = S_m(t) = 0.$$

We define the *saturation time* $\widehat{T}_m$ of block m by

$$\widehat{T}_m := \sup\{T : \text{ the system is } m-\text{UNSAT at } T\}.$$

That is, the saturation time $\widehat{T}_m$ of block m is the first instance of time at which the system is *not* m-UNSAT.

Therefore, for any P2P video streaming systems, the following theorems can be proved.

Theorem 4 *Given the user arrival process $Y_m(t)$, if the given protocols F_s and F_t are such that the P2P video streaming system achieves the maximum production $N_m(t)$ for a given limited uploading bandwidth of the server, then the saturation times $\widehat{T}_m$ are equal for every $m \in [1, M]$.*

Proof: Proof through counter example. We suppose that $\widehat{T}_m$ are not equal for every m when the P2P video streaming system achieves the maximum production $N_m(t)$ for given protocols F_s^1 and F_t^1. Then we can have $\widehat{T}_i^1 > \widehat{T}_j^1$, where $(i, j) \in [1, M]$. The P2P system can achieve the maximum production $N_m^1(t)$ for given protocols F_s^1 and F_t^1. Now that we have other protocols F_s^2 and F_t^2, the system achieves production $N_m^2(t)$ such that $\widehat{T}_m^2 = \widehat{T}_j^1$ for every m. Then we have $N_m^2(t) \leq N_m^1(t)$ for all m, according to the assumption.

However, at $\widehat{T}_i^1 > t > \widehat{T}_j^1$, by the definition of the saturation time, we know that the block-i is in the un-saturation stage, then $N_i^1(t) < Y_i(t)$ for given protocols F_s^1 and F_t^1. But for given protocols F_s^2 and F_t^2, the block-i is in the saturation stage; thus $N_i^2(t) \geq Y_i(t)$. So we have $N_i^2(t) > N_i^1(t)$, which is conflicts with the above results. The theorem is proved. □

Theorem 5 *Given the user arrival process $Y_m(t)$, the P2P video streaming system must obtain the minimum server stress for a given unlimited uploading bandwidth of the server, if the given protocols F_s and F_t are such that the system has the maximum $N_m(t)$ for all m for any given uploading bandwidth of the server.*

Proof: The theorem can also be proved in a contrapositive way. If given a protocol-1 (F_s^1 ,F_t^1) for any given uploading bandwidth of the server such that the system obtains the maximum production $N_m^1(t)$, the protocol-1 cannot achieve the minimum server stress for the given unlimited uploading bandwidth of the server. Then there is another protocol-2 (F_s^2 , F_t^2) for any given uploading bandwidth of the server such that the system production $N_m^2(t) < N_m^1(t)$, the protocol-2 achieves the minimum server stress for the given unlimited uploading bandwidth of the server.

However, we have the fact that, if given a fixed uploading bandwidth of the server, the protocol-2 must enter into the saturation stage earlier than the protocol-1 because the protocol-2 achieves the minimum server stress. By Theorem 4, we know that $\widehat{T}_m^1$ are equal, so there exists $i \in [1, M]$ such that $\widehat{T}_i^2 < \widehat{T}_m^1$. Then at $\widehat{T}_i^2 < t < \widehat{T}_m^1$, we have $N_i^2(t) > N_i^1(t)$, which is conflicting. □

Theorems 4 and 5 tell us that the optimal P2P streaming protocol (F_s^o , F_t^o) must have $\widehat{T}_1^o = \widehat{T}_2^o ... = \widehat{T}_M^o$ for achieving both minimum server stress and maximum production of the system for a given user arrival process. And the protocol should achieve the maximum production to achieve the minimum server stress. Thus, we could derive the following corollary to instruct us how to design an optimal P2P streaming system. The corollary shows that we can independently design a storage allocation protocol F_s and a transport protocol F_t under the given condition.

Corollary 2 *Given user arrival process $Y_m(t)$ for any P2P video streaming systems, if the protocol can guarantee that the saturation time $\widehat{T}_m$ are equal for all m, we can optimize F_s or F_t by assuming one of them is optimal.*

5.2 QoS Control

QoS control or measurement is one of the important requirements for implementing a P2P video streaming system. However, there are not many research results about it. Even we have not found the formal definitions of QoS metrics for P2P streaming system so far. In this session, we will introduce our definitions of QoS metrics, and then present some studies based on our definitions.

First, we define QoS metrics as a measurement of P2P streaming systems.

Definition 5 (waiting user) *The waiting users are the users who have arrived by time t, but they are not admitted to join in downloading the video stream. $N_w(t) :=$ the number of waiting users at time t.*

Definition 6 (jitter user) *Jitter users are those users that have experienced jitters during the time $(0, t)$. $N_j(t) :=$ the number of jitter users at time t.*

Definition 7 (average waiting time) *Waiting time: the time from the time the user arrives to the time at which the user is admitted to receive data. $T_w(i)$ denotes the waiting time of user $\mathcal{U}_i$. $\overline{T}_w$ is defined as the average waiting time for all users, i.e.*

$$\overline{T}_w = \frac{\sum_{i=1}^{N} T_w(i)}{N}.$$

Definition 8 (average waiting rate) *Average waiting rate: denoted to ω_w, which is defined by*

$$\omega_w = \frac{\overline{T}_w}{L}.$$

Definition 9 (average jitter time) *Jitter time: the actual time that the user spends downloading the video file minus the video length, which denotes $T_j(i)$ for user $\mathcal{U}_i$, i.e user $\mathcal{U}_i$ takes the time $T_j(i) + L$ to download the video file. Average jitter time is denoted by $\overline{T}_j$ over all users, i.e.*

$$\overline{T}_j = \frac{\sum_{i=1}^{N} T_j(i)}{N}.$$

Definition 10 (average jitter rate) *Average jitter rate is denoted to ω_j, which is defined by*

$$\omega_j = \frac{\overline{T}_j}{L}.$$

5.2.1 QoS Tradeoff

For a video-on-demand system, video length L is limited. The average waiting rate and average jitter rate imply how long a user waits to start to view a video and how smoothly they watch the video. However, for a video-streaming system, L is supposed to be infinite. Based on the proposed multi-block transport framework, we can create each sub-P2P streaming system independently. After we know the performance of each sub-system, then we can compute the whole system. Thus, we only need to consider the video length of each video block as L. To measure VoD applications, we need to further compute M sub-systems linked together. For live-streaming systems, we can take a sub-system to measure, since M becomes infinite, or we can take the average number of M by considering the average time that users stay in the system.

Lemma 3 *For a given set of users that want to download the video file from the server, the average download rate for every user is*

$$\overline{r} = \frac{1}{N} \sum_{i=1}^{N} r_i,$$

Then we have

$$\overline{r} \geq \frac{r}{\omega_w + \omega_j + 1}$$

if the average waiting rate and average jitter rate are given.

Proof: By the definition of waiting rate and jitter rate, we note that every user actually needs to take $(\omega_w + \omega_j + 1)L$ seconds on average to download the video file when it arrives. We denote t_i to the download time of the user-i, then we have

$$(\omega_w + \omega_j + 1)L = \frac{1}{N}\sum_{i=1}^{N} t_i,$$

and

$$\overline{r} = \frac{1}{N}\sum_{i=1}^{N} r_i = \frac{1}{N}\sum_{i=1}^{N} \frac{rL}{t_i}.$$

To prove the lemma, we just need to prove the inequality:

$$\sum_{i=1}^{N} \frac{1}{t_i} \sum_{i=1}^{N} t_i \geq N^2.$$

We note that the above inequality can be proven by induction. □

Theorem 6 *For any protocols of P2P video streaming, if the server uploading bandwidth and the user's average uploading rate are given, the maximum number of users that can be admitted at either the waiting or the receiving stage is*

$$\frac{s}{r/(\omega_w + \omega_j + 1) - \overline{b}} \geq N_{max} \geq \frac{s - \overline{b}}{r/(\omega_w + \omega_j + 1) - \overline{b}},$$

where we assume each user will leave the system after they obtain the video file. And $\overline{b}$ *is the average uploading rate of all admitted users.*

Proof: For a set of users that are viewing the video under an optimal transport protocol, at any time we have

$$s + \sum_{i=1}^{N} b_i - \min_{i\in[1\ldots N]}\{b_i\} \geq \sum_{i=1}^{N} r_i,$$

where b_i is the uploading rate of user $\mathcal{U}_i$, r_i is the downloading rate of $\mathcal{U}_i$, and N is the number of the set of users at any time. By the definition of $\bar{b}$ and Lemma 3, we know that every user has the same average download rate over all time under an optimal protocol. Otherwise, the system does not support the maximum users under the same total available uploading bandwidth. Thus, we have

$$s + N\bar{b} - \min_{i\in[1\ldots N]}\{b_i\} \geq N\frac{r}{\omega_w + \omega_j + 1},$$

which can be presented as

$$N \leq \frac{s - \min_{i\in[1\ldots N]}\{b_i\}}{r/(\omega_w + \omega_j + 1) - \bar{b}},$$

when we note that $\bar{b} \geq \min_{i\in[1\ldots N]}\{b_i\} \geq 0$. The result is proven. □

Theorem 6 implies that the maximum number of users can be accepted in a sub-P2P streaming system, with a given server upload bandwidth and performance requirement (in terms of average waiting rate and average jitter rate), i.e. there must be an optimal protocol so that the system can support all users meeting the performance when the number of users in the system is less than N_{max}. In other words, the theorem also tells us that there is no protocol by which the system can support a number of users greater than N_{max} for a given server upload bandwidth and performance requirement. In fact, any practical protocols can only support a number of users that is less than or close to N_{max} simultaneously.

Figure 5.2 gives the maximum number of users that the system can support under a different average upload rate of users for a given QoS performance requirement. When the average upload rate of users is zero, the system becomes the traditional client-server model, which implies that the system can only support the users that directly connect to the server. Thus, the number of admitted users is s/r. However, when the average upload rate almost equals the video playback rate or greater, it tells that the number of users approaches infinite, which indicates that the P2P system has good scalability. This figure also shows that the required download rate becomes smaller when we allow

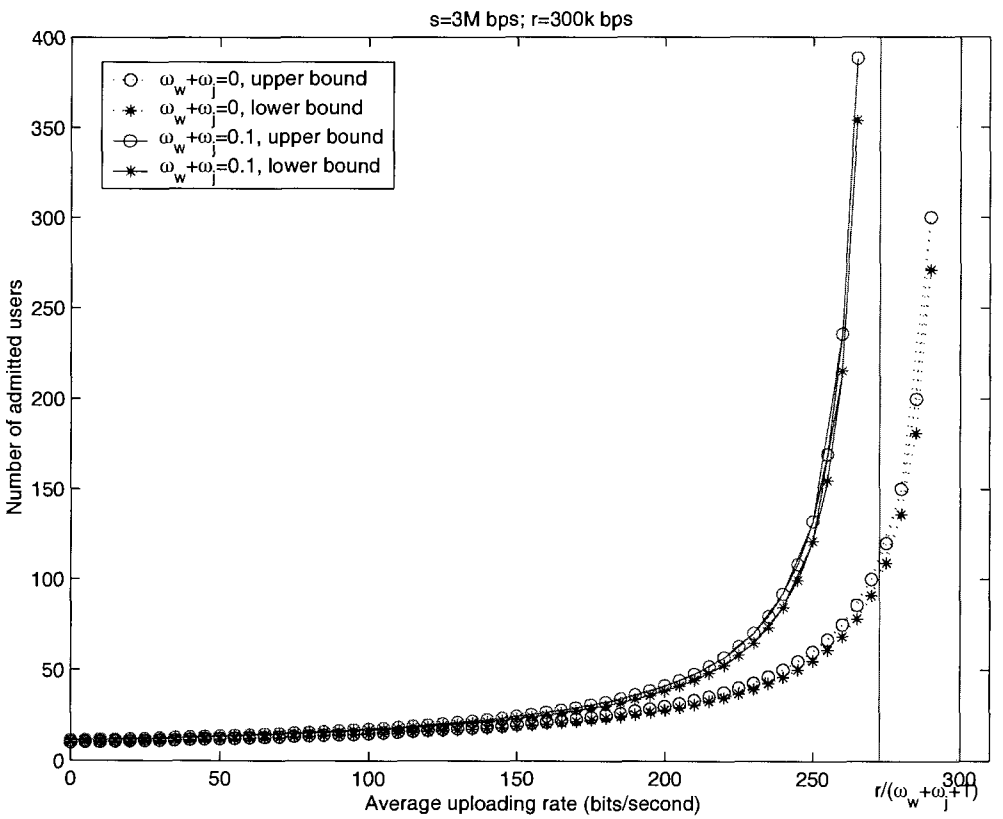


Figure 5.2: Maximum number N of admitted users against the different average uploading rate b.

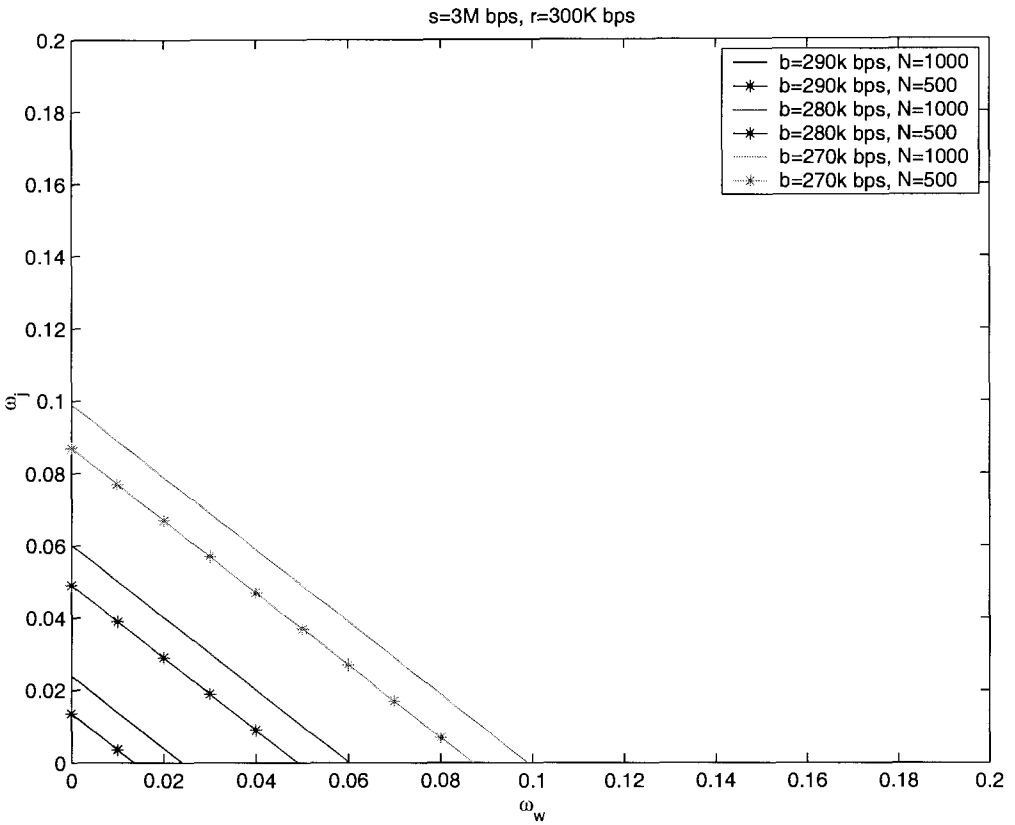


Figure 5.3: Waiting rate and jitter rate against average uploading rate and the number of admitted users.

users to wait some time to start to play the video or during playing, i.e. the waiting rate and jitter rate are greater than zero.

From the theorem, we can get the following corollary that gives the relationship among the performance measure metrics and average uploading rate of users and the number of admitted users.

Corollary 3 *For a given server uploading bandwidth s and video playback rate r, there exists*

$$\omega_w + \omega_j = \frac{(r - \bar{b})N - s}{s + \bar{b}N},$$

where N is the number of admitted users.

Figure 5.3 demonstrates the corollary. It shows that the waiting rate and jitter rate have a trade-off relationship for a given setting system. In other words, the jitter rate could be zero as long as the waiting rate is big enough. Further, it tells that both the average uploading rate of users and the number of admitted users mainly affect QoS performance. A larger average uploading rate and a lower number of admitted users will increase the QoS performance.

Chapter 6

P2P Video on-Demand System (rVoD)

In this chapter, based on the design guidelines on general P2P systems, a novel P2P video on-demand system to implement resilient VoD streaming supporting free user interactivity, called rVoD is proposed [87]. The system includes two layers: video content transportation protocol and user's storage management. The novel idea is the application of rateless coding technology [84, 25] to both content streaming and storage distribution so that rVoD has significant benefits compared to the previous proposed works [51,73,28]. The chapter is organized as follows: The network setting is first presented. The transportation protocol is then introduced. The storage distribution scheme is presented last. The system analysis and simulation results are discussed in the next chapter.

6.1 P2P VoD Network

A P2P video on-demand system consists of a video server denoted as $\mathcal{U}_0$ and an index directory server over a lossy network. The video server stores the videos, and the index directory server maintains the list of online users who are playing the videos. The video content at the video server is divided into M small blocks, each having size A bits. Each

user could start to view the video at any position (i.e. any video block) and could perform VCR operation (i.e. watching video blocks non-consecutively).

Users in the system can leave and join at any time. And every user can store the partial data of the video blocks that it has played. The amount of partial data stored from each video block depends on the user's buffer size K_m. If the buffer size is bigger than the total size of video blocks that the user has played, the user can store the full information of the video blocks. If not, obviously, an algorithm is needed to decide the amount of the partial data of each block to be stored dynamically based on time. However, in most of the proposed works that we discussed in Chapter 3, they use only the storage in FIFO (first in first out) to relay the video content. We only found one work (VMesh) [28] so far where authors proposed a method by which every user stores some specific video blocks based on the block population. In fact, every user's storage is one of the major resources of the P2P network, which cannot only help every user itself but also help others if the storage can be managed in an efficient way. To look for an optimal buffer allocation is an open topic that needs more research work. One of our research works in this thesis is to investigate an efficient algorithm for this. Our creative idea is to make every user store the partial information of the complete video content using rateless coding. To the best of our knowledge, we are the first group to propose this method.

We assume that there is an index server that contains the list of online users in the P2P VoD system for providing user administration, because it is hard to administrate the user and implement accountability in a pure distribution P2P network [54]. Our purpose is to design a P2P VoD system that can be deployed in a commercial environment, so it is practical and efficient to have an index server in the system. Also, the index server can be collocated physically with the video server; it does not really increase the failure point of the system. To reduce the load of the index server for centralized management, we proposed a hybrid online user-list-maintaining mechanism.

Thus, every user stores the encoded video content using rateless codes [84, 25] in our

proposed rVoD, instead of just storing a few segments in VMesh [28]; i.e. each user could have the partial information of the complete video content with a small-size buffer (the buffer size can be smaller than the size of the complete video content). Each user can arbitrarily request the video content from any other users who have played the requested content earlier in rVoD. And the user can keep using the currently connected supplying users, if most of users play the video blocks in order. The user's VCR activities would not affect either its associated children or its associated suppliers if the user's jumped viewing point is in the video content the suppliers played. In realistic networks, above assumptions are practical in general. Quickly locating supplying users and managing online users rely on our proposed hybrid online user-list-maintaining mechanism based on a centralized directory server and a local distributed updating scheme. Video content transportation is based on mesh topology using rateless coding. The rVoD has the following favorable properties:

- Quick join: Any users who arrive earlier can serve the users who arrive later. This will give the best possible download bandwidth.
- Resilience: Mesh-based content delivery overlay network (multi-user to multi-user) and rateless coding transportation features suggest the robustness of transmission packet loss and link failure without single parent problems.
- Efficiency: The handling of user's asynchronous requests and failure recovery for each user having distributed partial information of the complete video content are not complex.
- Small overhead: Only the online user list needs to be maintained. Each user just updates the list once with the index server and locally updates with its children.

Figure 6.1 illustrates the idea of rVoD. The video server (S) has a video file with a length of 4 blocks. Three users (U1, U2, and U3) already exist in the network. They are playing the different blocks of the video, which is shown in the box of **P1**. The

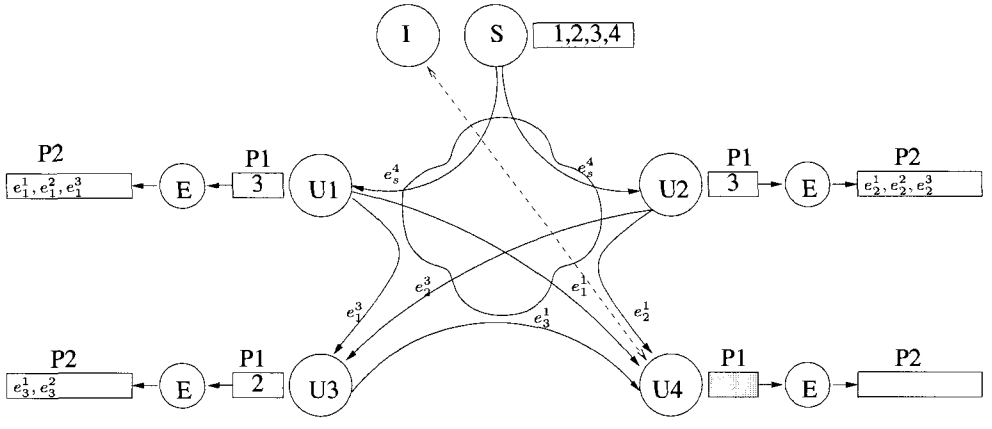


Figure 6.1: rVoD with distributed encoded video content storage

P2 indicates the storage of every user. The circle **E** is the encoding function of rateless coding. Every video block that the user has played will be stored in **P2** after it is encoded by the function of **E**. U1 has played block 1, block 2, and block 3 so that it has the encoded packets e_1^1, e_1^2, and e_1^3 in **P2**, where e_i^m means the encoded packets of the video block-m at the user $\mathcal{U}_i$. When U4 arrives and requests block 1, it registers with the index server (I) (shown as the dashed line) and obtains the current online user list. Then U4 connects to (U1,U2,U3). U1, U2, and U3 transmit their encoded packets of block 1 to U4 until U4 decodes block 1. U4 only needs the total downloaded encoded packets to be slightly bigger than the size of block 1 to successfully recover block 1 using the rateless coding feature, and the overhead can be less than 2%.

6.2 Video Content Transport

Designed for erasure channels, a fountain code (LT codes [24] or Raptor codes [25]) is a mapping of k information symbols to an infinite stream of codeword symbols. The symbol here can be a bit or a binary vector of any prescribed length. The transmitter continuously sends codeword symbols until the receiver, upon receiving sufficient symbols, is able to decode the intended k symbols. The receiver then sends an `ACK` to the

transmitter to terminate the transmission of the current codeword. Taking LT codes, for example, the mapping of the k-symbol information vector to the codeword stream is via a pseudo-random selection of an integer d from $\{1, 2, \ldots, k\}$, followed by another pseudo-random selection of d information symbols and then an XOR operation of the d symbols. The family of fountain (LT) codes may be in principle constructed by the specified distribution from which d is drawn. Each realization of the i.i.d. sequence $\{d_i\}$ defines a code.

The performance of fountain codes is in that they can achieve the capacity of erasure channels without any knowledge of the probability law of erasure events. That is, without such knowledge, as long as $k(1+\epsilon)$ symbols are received, the decoder can recover the k-symbol information vector, and ϵ, referred to as the *overhead*, is quite small (usually less than 5% when k is relatively large). In addition, the complexity of encoding and decoding fountain codes is low and practicable, and commercial products based on fountain codes for large-file transport have been developed.

Now we propose a generic fountain-coded transport framework for peer-to-peer video streaming. What motivates us to use fountain codes is precisely the universality of the codes. That is, if the receiver is allowed to simultaneously receive symbols from two different transmitters, each using a different fountain (LT) code with the same distribution, the receiver will be able to decode as long as in total $k(1+\epsilon)$ symbols are received from the two transmitters. That is, the sum rate of the two channels is automatically achieved, and no channel knowledge is needed. In a peer-to-peer video-streaming system, all peers having a common block requested by a user can then simultaneously transmit to the user, thereby achieving the sum capacity of the links to the user, as shown in Figure 6.2.

Specifically, in this framework, every user $\mathcal{U}_i$ (including server $\mathcal{U}_0$) has a *signature*, which uniquely specifies the fountain code that it uses for encoding. Every $\mathcal{U}_i$ may request and receive a video block from any other users as well as from the server. Each $\mathcal{U}_i$ also keeps a list of users requesting the video block that $\mathcal{U}_i$ has in its buffer. Whenever $\mathcal{U}_i$ has

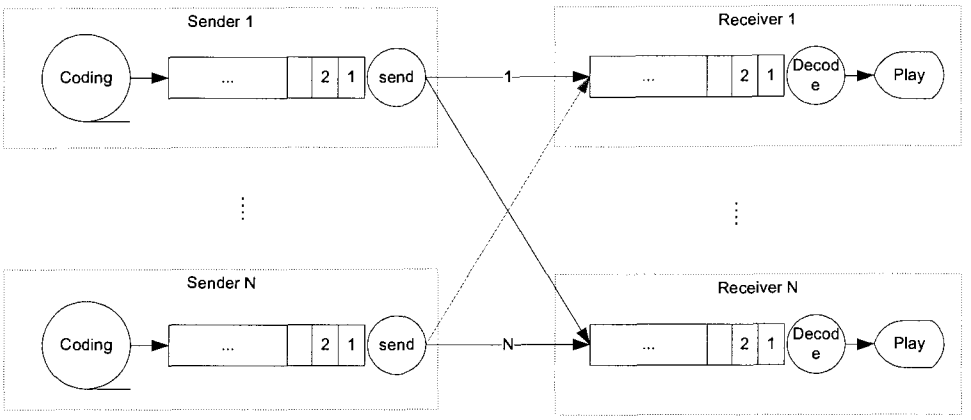


Figure 6.2: Fountain codes-based P2P transport model

free output bandwidth, it picks one user, say $\mathcal{U}_j$, from the list — according to certain rules — and sends to $\mathcal{U}_j$ the block encoded by $\mathcal{U}_i$'s fountain code. It will terminate transmitting as soon as it receives the acknowledgment from $\mathcal{U}_j$ that has decoded the block.

Each user $\mathcal{U}_j$, as a receiver, after receiving fountain-codeword symbols from possibly multiple users, performs joint decoding of all received symbols, based on the codebooks of these senders. It continuously attempts to decode until it can decode the block. It then sends back to all senders an acknowledgment `ACK`. It stores the block in its buffer if there is room in the buffer. If there is no room in the buffer, it removes a block from the buffer based on certain rules and stores the newly received block. It then sends out a request for the next block.

6.3 Distributed Storage Management

The advantages of P2P approaches to distribute video content rely on the availability of each user's cached video content to serve others. In this section, we formally introduce the rateless code-based encoded video caching framework (DPCC) to maximize the contribution of each user for the streaming service.

First, when the user joins the system, it divides its buffer (K_m blocks) into two parts:

part 1 (P1) for playback when its size is K_p blocks, and part 2 (P2) for serving others, whose size is K_s blocks. Then we have $K_m = K_p + K_s$. Here we assume K_m must be greater than 1. If $K_m = 1$, users are only consumers. We would not count such users in the system as normal users.

For the buffer P1, the user uses the traditional rule FIFO to cache video blocks that were recently played. The main purpose of buffer P1 is to protect the network fluctuation. In other words, buffer P1 keeps the most recently received video blocks from block m to block $m+K_p$ while the user is watching video block m and is downloading block $m+K_p$.

Then, when a video block (block m) is ready to leave buffer P1, the user uses its own identity rateless codes (fountain codes) to encode block m and keeps the encoded packets in the buffer P2 in its available memory. Here the amount of memory that can be used to store the encoded packets of block m depends on the buffer size of K_s and the number of video blocks stored. The same fixed size of memory for every block, i.e. storing $\frac{K_s}{M}$ bits information for each block, can be allocated. We refer to this method as DPCC-FIX, as shown in Figure 6.3.

Clearly, the method of DPCC-FIX is not optimal in terms of minimizing the server stress, because the requesting block probabilities for each block is different at all times, where the requesting block probabilities refer to the requesting block popularities for each block. In [88], the authors model the video segment popularities, i.e. the requesting popularities follow a Zipf distribution like the distribution of Web objects' popularities on the Internet. In fact, the requesting block probabilities may change over time. Thus to optimize the usage of information stored, the user needs to dynamically allocate the size of memory to store different blocks according to its requesting popularities. At the same time, we also note that the size of the stored video blocks decreases over time because most users will not play the same video blocks twice. Considering all of the conditions, an adaptive mechanism for estimating the requesting block probabilities in a distributed manner is needed. And an adaptive memory allocation algorithm is needed to meet both the current requesting block popularities and the future requesting block

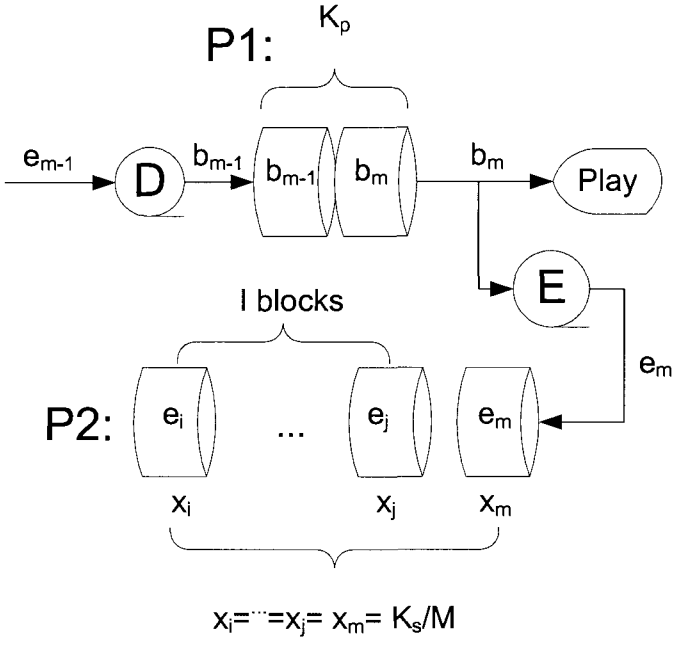


Figure 6.3: DPCC-FIX

popularities. At this point, we propose using a requesting block popularity-based storage allocation scheme, as described in the section 6.6. This scheme is referred to DPCC-PB.

6.4 Requesting Block Popularity Estimation

It is not easy to obtain the global requesting block popularities for every user over the P2P network, because the user does not have the global information. However, it is possible to estimate it using the local information or local information exchange. For instance, the proposed distributed average algorithm in [89] can be used to estimate the global requesting block popularities. But this algorithm will increase the communication overhead over the whole network, since every user needs to exchange information with its neighbours frequently. In our proposed architecture, we propose a new way to estimate the requesting block popularities by using the local block accessing rate, because every user has stored all video blocks and accepts any user's request in our proposed P2P system. Through the simulation, we confirmed that our method is practical and efficient

for our system. The simulation results of comparing the estimated rate with the global rate are shown in Figure 6.4. The formal estimation algorithm is given next.

For every user, at time t, let C_m denote the number of requesting connections of video block m (including the associated links for downloading block m), and let x_m indicate the size (bits) of the cached video block m, where m varies from 1 to M. The user has I blocks of stored video in buffer P2. The local block accessing rate-based requesting block popularity estimation $\hat{p_m}$ is defined as follows:

$$\hat{p_m} = \frac{\frac{1}{C_m+1}}{\sum_{k=1}^{M} \frac{1}{C_k+1}}, \quad m = 1 \cdots M,$$

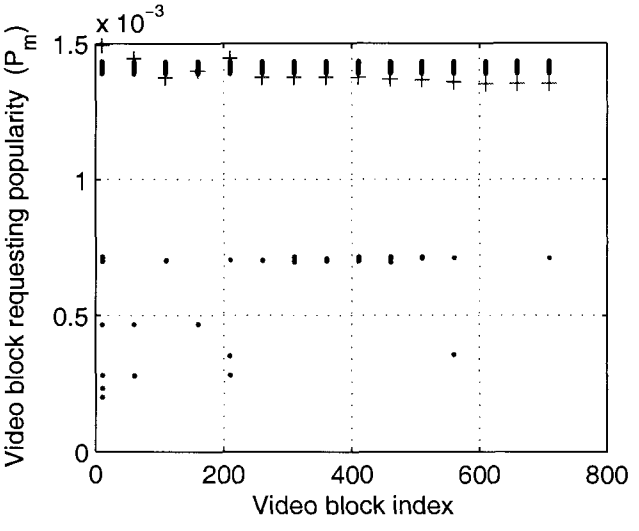
and thus,

$$\sum_{m=1}^{M} \hat{p_m} = 1.$$

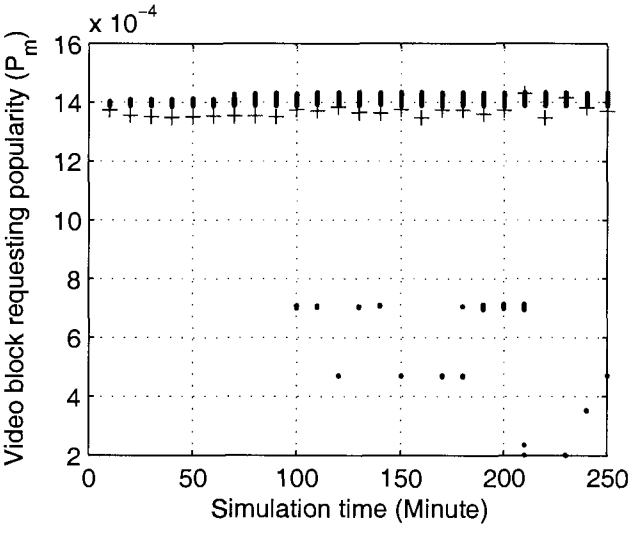
6.5 Transport Scheme: Protocol B

Based on the requesting popularity, we proposed the video content transport scheme, which is named Protocol B [84]. In Protocol B, every user allocates a different uploading bandwidth for different requesters, and a generic rateless-coded (or fountain-coded) is used to encode video streaming. Each user $\mathcal{U}_i$ (including the server $\mathcal{U}_0$) has a *signature*, which uniquely specifies the fountain code. Users may request and receive a video block from any other users as well as from the server. The design of fountain codes allows the receiver to decode as long as a total of $k(1+\epsilon)$ symbols are received from multi-supplying users, as discussed before. The user does not need any further costs to deal with the loss of packets. The transport protocol naturally has the ability to handle link failure or network jitter issues.

Each user $\mathcal{U}_i$, as a receiver, may continuously send requests for the block they need. The user continuously updates a quantity R_e, the rule of which will be specified shortly. The physical meaning of R_e is an estimate of the total rate of receiving the block in need.



(a) Against each video block at the simulation time of 100 minutes.



(b) Against the simulation time for video block 350.

Figure 6.4: Local estimated popularity (P_m) compared to global popularity. Each blue dot stands for a user's estimated value; Each red plus (+) stands for the global value.

The user compares R_e with a threshold $R_{th} = \beta(1+\delta)$ — the playback rate with a small margin — for some small δ (fixed at 0.1 in our simulations). If $R_e < R_{th}$, the user sends a request for the block together with the "demanded rate," $R_{th} - R_e$. The detailed user procedure is described in the following four steps and shown in flow chart 6.5:

- Step 1: Sends the requesting command to currently associated supplying users for a new video block, say block m.
- Step 2: Decodes the encoded packets of block m while receiving an encoded packet. If the user successfully decodes block m, goes to Step 1 for the next video block (block $(m+1)$); otherwise, goes to the next step.
- Step 3: Runs the downloading rate estimation equation 6.1 periodically, such as in Δt. If the estimated total downloading rate R_e is less than a threshold R_{th}, adds a new supplier (the rule to select a new supplier follows the same rule given in 6.5.1). Otherwise, goes to Step 4.
- Step 4: Removes one supplier if the estimated downloading rate from this supplier is less than a threshold D_{th}. Then goes to Step 2.

$$R_e = \frac{\text{the total number of the received bits in } \Delta t}{\Delta t} \tag{6.1}$$

Each user $\mathcal{U}_j$, as a transmitter, if they have free output bandwidth, randomly selects a block that they have in their buffer, where the probability of choosing the block m is proportional to the estimated requesting popularity $\hat{p_m}$ of block m. Then $\mathcal{U}_j$ randomly chooses a user from the users requesting the selected block. If the chosen user's "demanded rate" is smaller than $\mathcal{U}_j$'s available free bandwidth, $\mathcal{U}_j$ uses a portion of its bandwidth equal to the "demanded rate" to transmit the selected block to the chosen user. If the chosen user's "demanded rate" is larger than $\mathcal{U}_j$'s free bandwidth, $\mathcal{U}_j$ uses all its free bandwidth to transmit the selected block to the chosen user. $\mathcal{U}_j$ also informs the

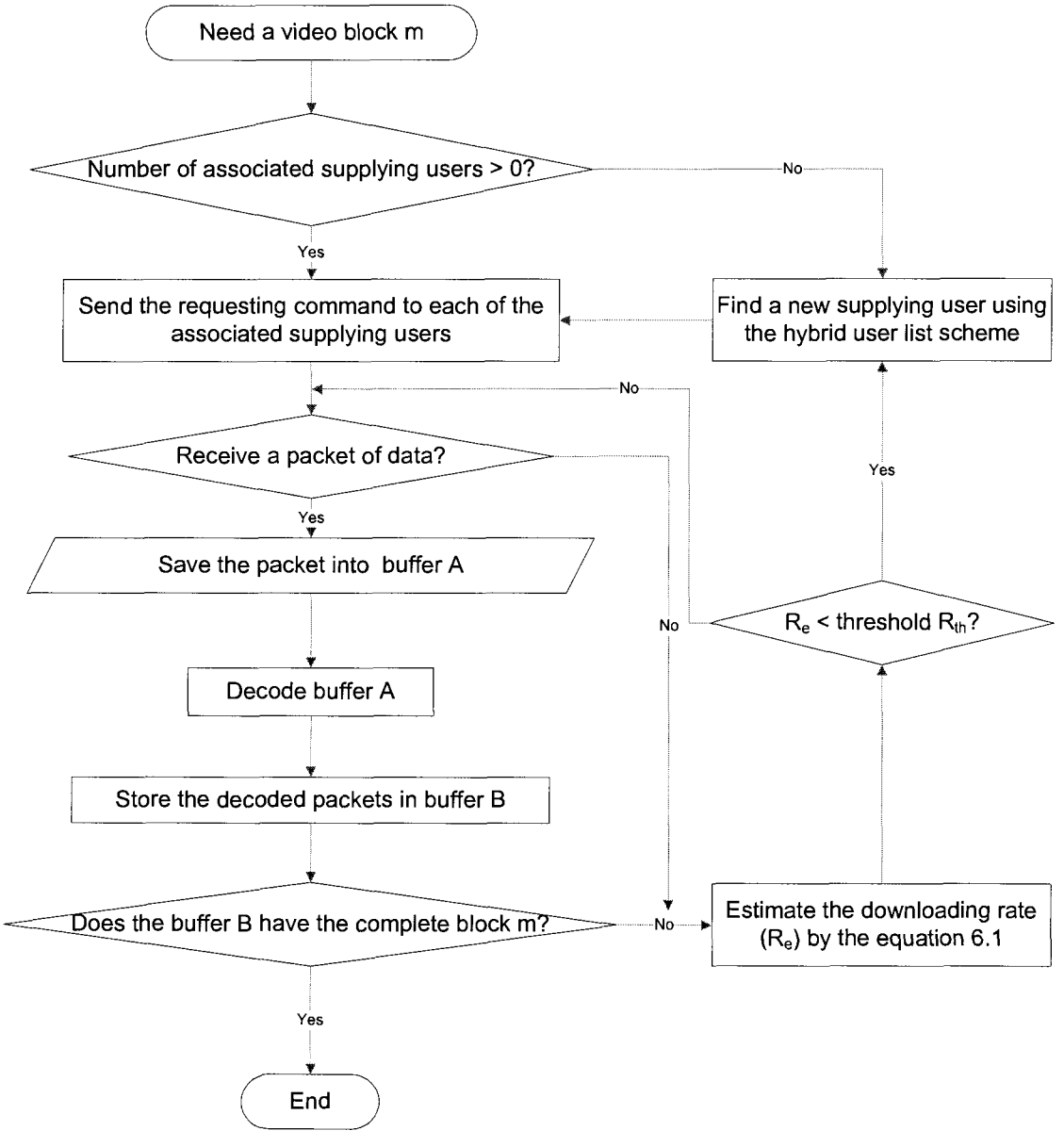


Figure 6.5: Flow chart as a receiver in rVoD

chosen user of the rate that $\mathcal{U}_j$ is using to transmit to the chosen user. We call this rate the "promised rate." $\mathcal{U}_j$ repeats this process until there is no free output bandwidth. The detailed flow chart is presented in Figure 6.6 and the user-admitting algorithm is given in Algorithm 6.1.

Algorithm 6.1 Admit a new user by requesting popularity

1: $b^f = b - \sum_{all\ connected\ user\ i} r_i^p$; //b refers to the maximum uploading bandwidth of the user (pre-configured; r_i^p refers to the "Promised rate" for $\mathcal{U}_i$; b^f refers to the free uploading bandwidth.
2: **if** $b^f \geq 0$ **then**
3: Order the requesting users in the descending order of $\hat{p_m}$ of the block-m they requested;
4: Choose the user $\mathcal{U}_k$ at the top of the ordered list;
5: **if** $\mathcal{U}_k$ "demanded rate" $< b^f$ **then**
6: Send `ACK` to $\mathcal{U}_k$ with the "Promised rate" = its "demanded rate;"
7: **else**
8: Send `ACK` to $\mathcal{U}_k$ with the "Promised rate" $= b^f$;
9: **end if**
10: **end if**

In Protocol B, the server also uses the same algorithm to allocate output bandwidth. The servers will never be idle anymore, since it is always helpful for supplying more sources to the system, although the server's output bandwidth is not the key factor for the system after more and more users arrive. Since the server has the complete video content, it acts in the same way as a provider at the admitting steps, but it does it differently at the transmitting steps, at which the server continuously generates the encoded packets of video block m using its own fountain codes, then transmits the encoded packets to the receiver until it gets a new requesting command for the next video block or disconnection command. The detailed flow chart is given in Figure 6.7.

We note that the proposed Protocol B dynamically allocates user output bandwidth and adapts to the evolution of the system from zero users to an arbitrary number of users.

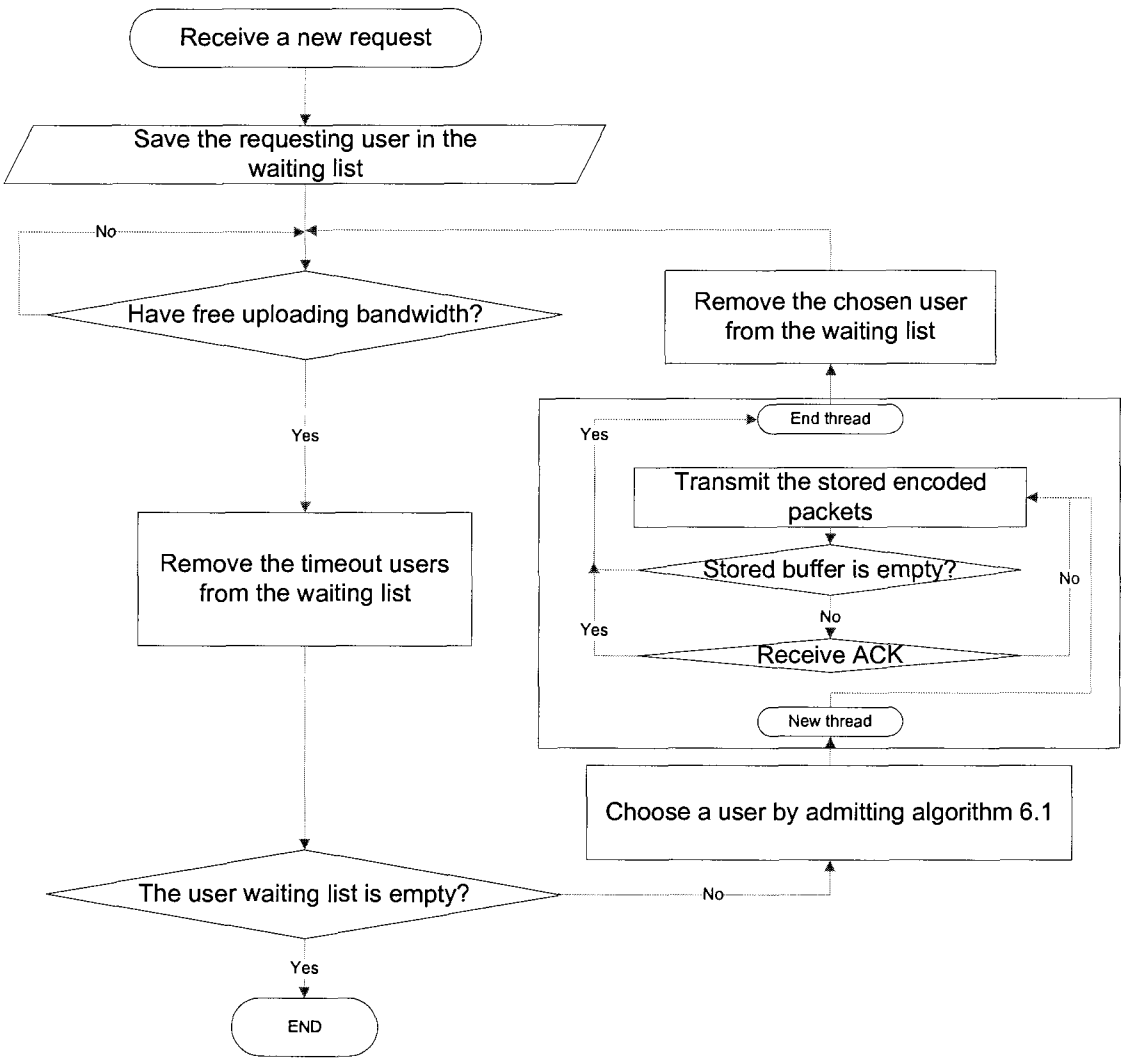


Figure 6.6: Flow chart as a transmitter in rVoD

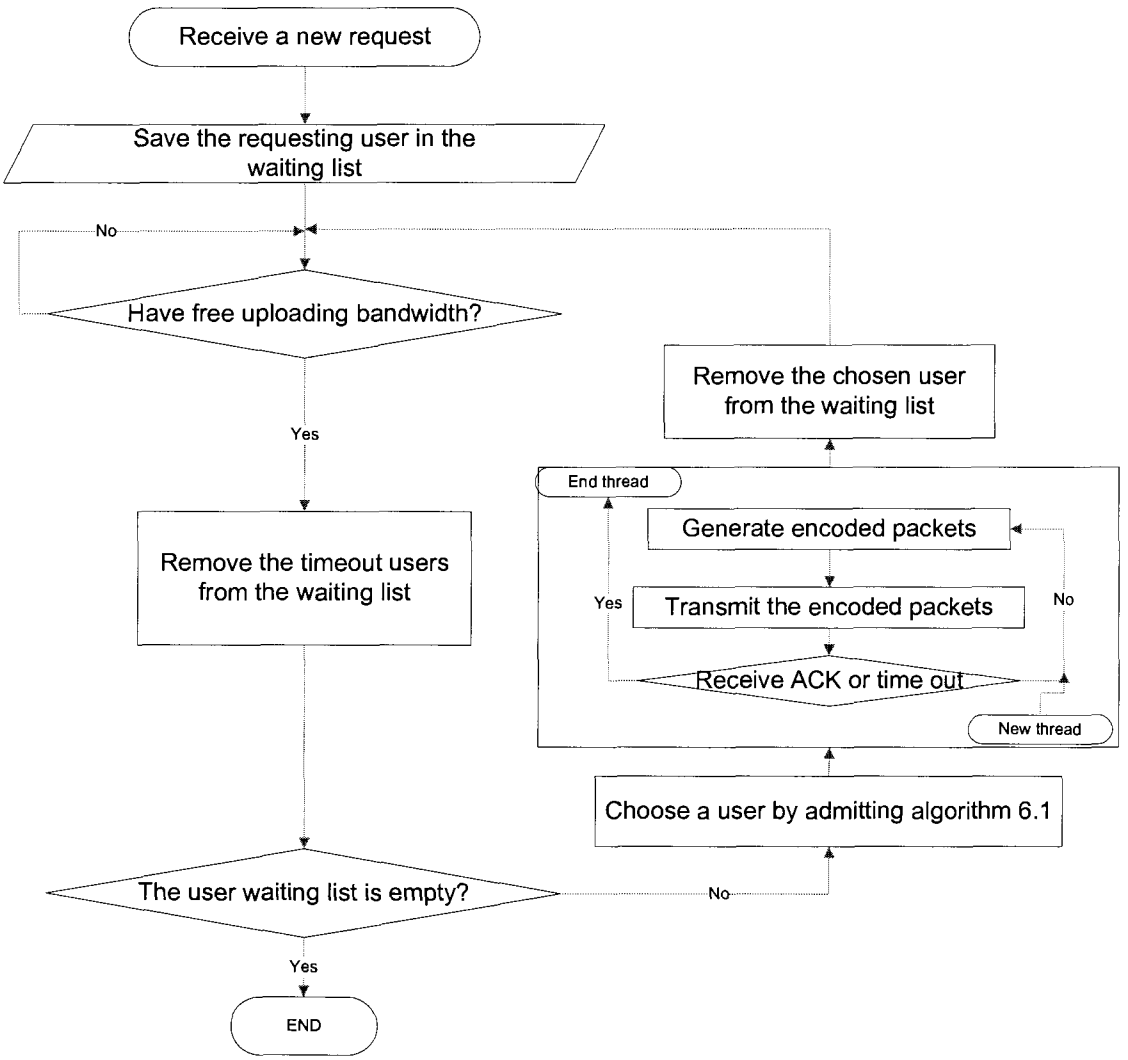


Figure 6.7: Flow chart as the video server in rVoD

6.5.1 User Join or Leave

When user $\mathcal{U}_j$ newly joins in the system, it performs the following procedures:

1. It generates its own identity (*signature*) for the seed of fountain codes.

2. It obtains the requested video information, including playback rate R, the number of blocks M, and the size of block A, from the index server.

3. It registers to the index server and obtains the distributed user list according to its interested video position. Forming the distributed user list for the user follows two rules: (1) choose those users who stay online long enough; for instance, longer than $L/2$; (2) choose users whose video index (defined as $V_t = A_t - B_t$) is close to the requester for users who stay online for a short time, where close means that user $\mathcal{U}_i$'s video index V_t^i is smaller than the requester user $\mathcal{U}_j$'s video index V_t^j, and $V_t^j - V_t^i$ is not big.

4. On the distributed user list, order the list by video index V_t from small to large, and the video server will always be last on the list. Then the users at the top of the list have a higher priority to serve than the users at the bottom of the list. To improve transmission efficiency further, we can also order the lists by the network location parameters such as the round-trip times using Ping.

5. The user follows the transport protocol presented above to maintain the video content downloading paths.

6. If the user finds that the number of available supplying users in its own distributed user list is less than a threshold, it asks its children to update the distributed user list.

7. If the user wants to leave, it can simply quit or notify the index server.

6.5.2 VCR Operation

rVoD supports VCR operations, i.e. the user can have pause, jump forward/backward, etc., functions. Because the caching of the encoded video blocks is never discarded completely for any of the user's activities, the VCR operation does not affect its children when continuing to download video content. Further, the user does not need to locate the new supplying users either. It can keep currently associated suppliers to download the new requested video block. That is the key difference compared to VMesh architecture. In VMesh, the user needs to search new suppliers using the DHT method, which requires searching the whole network. However, the user in rVoD just needs to estimate the downloading rate locally. If the downloading rate is lower than the threshold, it replaces some of the suppliers or locally updates the distributed user list with the children.

6.6 Storage Allocation Scheme: DPCC-PB

Based on the distributed storage framework using rateless coding, every user stores partial or full information of the video blocks it has played in our proposed P2P VoD system. However, we note that a storage allocation algorithm for each cached video block is needed to optimize the performance of the system in terms of reducing the video server load or improving the production of the system. In this section we present an efficient solution: a requesting popularity-based storage allocation scheme (DPCC-PB). The experiment results are discussed in the next chapter.

The popularity-based adaptive storage allocation algorithm is indicated in Algorithm 6.2. When a new video block m is ready to move out from the play buffer P1 at every user, the user will first update the estimated requesting block popularity $\hat{p}_i$, then call this algorithm to re-compute the size x_i of the allocated memory for each video block-i, $i = 1, 2, \cdots, M$, including x_m for the new video block-m. Here, we assume that the user already stores I blocks in its buffer P2 when the new video block-m is ready. The proposed scheme refers to DPCC-PB here, which can be presented as Figure 6.8. The

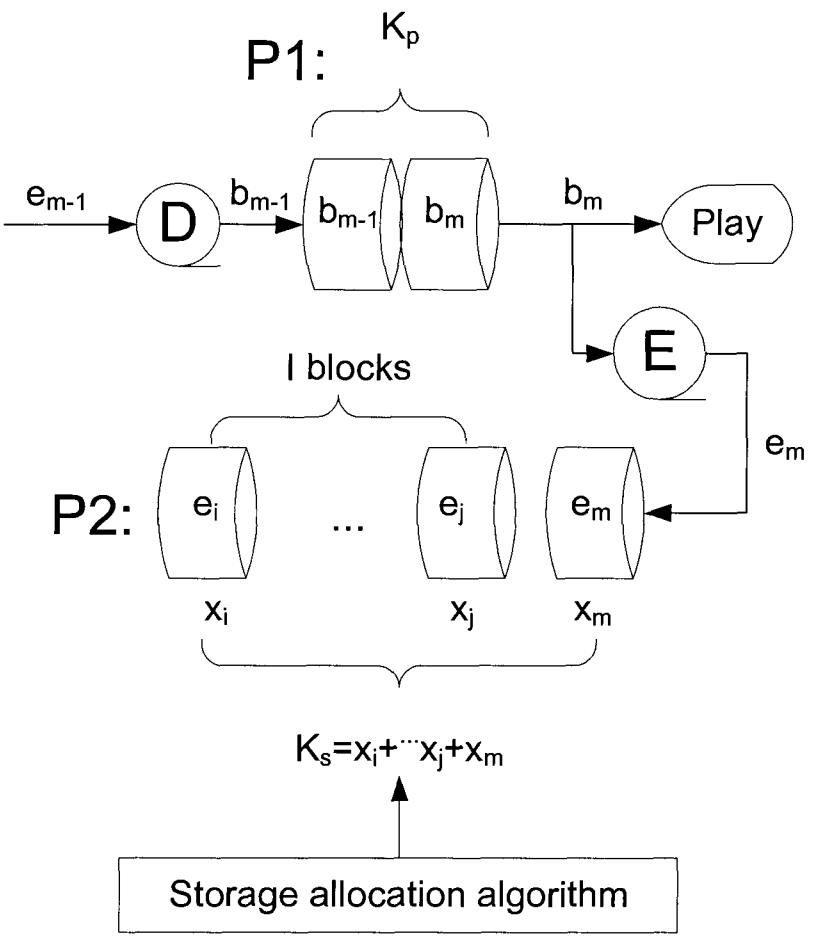


Figure 6.8: DPCC-PB

proposed algorithm has the following features:

- It guarantees that the user stores all blocks that it has played, at least X_{min} bits.
- It always allocates more memory size for the video blocks with higher requesting block popularity.
- It always tries to allocate memory to store the complete video block.
- The algorithm is only based on the user's local information; it is totally distributed.

Algorithm 6.2 Popularity-based storage allocation

Require: Set $x_i = 0$, for $i = 1, \cdots, M$ when the user joins in the system.

$X_{avg} = K_s/(I+1)$; //I refers to the total number of blocks that the user stores now; A refers to the size of video block.
if $X_{avg} \geq A$ **then**
  return $x_m = A$; //Store the full information of new block-m.
else if $X_{avg} \leq x_m$ **then**
  return x_m; //The size of memory for caching each block keeps same.
else
  $x_m = X_{avg}$;
  while $X_{avg} \neq 0$ **do**
    Define a set: $S = \{i | x_i > X_{min}, i = 1, 2, \cdots, M\}$; //$X_{min}$ is a parameter, we choose the value of K_s/M
    $x_i = x_i - X_{avg} \frac{\hat{p_i}}{\sum_{k \in S} \hat{p_k}}$, for $i \in S$;
    $X_{avg} = \sum_{k \in S \text{ and } x_k < X_{min}} X_{min} - x_k$;
    $x_i = X_{min}$, if $x_i < X_{min}$ for $i \in S$;
  end while
end if

6.7 Hybrid User List Maintaining Mechanism

A hybrid architecture to implement a fast method to locate the supplying users, which includes having a centralized directory at the index server and keeping the distributed partial directory for every user is proposed. Introducing a centralized index server may cause a single point failure problem, but we argue that the index server will cause the

single failure problem for the index server can be located together with the video source server physically. Further, the centralized server gives the capability of managing distributed users, such as membership management, media copyright management, and service quality control. Those kinds of management capability are very important requirements for the service provider. In fact, poor management capability is one of the main reasons why the completely distributed systems cannot be deployed as commercial service applications.

To reduce the heavy bandwidth load problem of the centralized index server, we introduce the distribute partial directory for every user. Every user once obtains its own partial directory from the index server, then it updates the directory with its connected users locally. The directory updating at the index server relies on the feed-back mechanism, i.e. every user only updates the directory information to the server when its own information changes or another user's state becomes offline.

In rVoD, the directory information (i.e the online user list) consists of the user's IP address, the user's requested video position B_t with the requesting time A_t, and its state (online/offline). The detailed information shown in Table 6.1. The video index V_t is introduced as $V_t = A_t - B_t$ for improving the probability of locating the supplying users with needed video blocks. That is based on two facts: most users play the video in the sequence of the video position, i.e. from small video block id to big video block id; and the user stores all the video blocks that it has played as either partial or whole information. Thus, we can know that the user with the smaller V_t has the stored video block that the user with bigger V_t needs in most cases, specially, in cases where the difference between the two users' V_t is not big. If users play from the beginning of the video and never jump to play forward/backward, the rule above always holds true. The proposed scheme has the advantages of both the pure centralized directory and the distributed directory: quick locating, low overhead, and a balanced server load.

The detailed procedures for every user to maintain/update the online user list is summarized as follows:

Table 6.1: Online user directory information

IP	A_t	V_t	S_t

IP: The user's IP address

A_t: The system identity time (in seconds) when the user registers/updates its B_t

B_t: The user requesting video position B_t in seconds, and the value is taken from 0 to L

V_t: $V_t = A_t - B_t$

S_t: The user's state, $S_t = 0$, means it is online; otherwise it is offline

1. When the user $\mathcal{U}_i$ joins in the system, it registers with the index server with its B_t. Then the index server organizes a distributed user list (for a specific number of users in the list, such as 20) for the user according to the following two rules: (1)choose those users who stay online long enough for instance longer than $L/2$; (2) choose users whose video index V_t is close to the requester for users who are online for a short time. Where close means that the user $\mathcal{U}_i$'s video index V_t^i is smaller than the requester user $\mathcal{U}_j$'s video index V_t^j and $V_t^j - V_t^i$ is not big.

2. When the user performs VCR operations, i.e. the user changes its viewing position B_t, the user updates the new B_t to the index server.

3. When the user finds the state of the users in its own distributed user list changes to offline, the user updates the state to the index server.

4. If the user finds that there are not enough online users in its own distributed user list, the user asks its connected children to update it. If no children are associated, the user connects to the index server to update it.

Chapter 7

rVoD System Analysis and Simulation

In this chapter, the performance of our proposed rateless codes-based transportation protocol (Protocol B) without a specific storage management is first analyzed. The purpose of Protocol B is to measure the dynamic performance of the system in terms of QoS metrics. Then the performance of the proposed rVoD in terms of server stress is presented. The simulation results have shown that rVoD has better performance compared to the previous works, such as VMesh [28].

7.1 Performance Analysis of Protocol B

To measure the dynamic performance of P2P video streaming systems, we have performed simulations of transport protocol-Protocol B. In this experiment, we mainly focus on measuring the QoS performance in terms of the jitter rate and the waiting rate when the uploading bandwidth of the video server is given. Since we have not proposed any specific users' storage allocation methods in Protocol B, we assume each user has enough buffer size to store all of the video blocks that it played, or the user stores the most recently played video blocks if the buffer size is not enough. We will investigate the user's buffer

allocation problem in rVoD.

QoS metrics of jitter rate and waiting rate in the simulation are defined as follows. For user $\mathcal{U}_i$, we will denote by $T_{i,m}$ the time at which it has obtained video block m. We will use $T_{i,0}$ to denote the arrival time of $\mathcal{U}_i$.

Definition 11 (Waiting Rate) *For some prescribed minimum time $T_{\min}$, the waiting rate of the system at time t is defined by $W(t) := \frac{N_w(t)}{N(t)}$, where $N_w(t)$ is the number of "long-waiting" users up to time t; the user $\mathcal{U}_i$ is said to be a "long-waiting" user if $T_{i,1} - T_{i,0} > T_{\min}$.*

Definition 12 (Jitter Rate) *Under any given video-streaming protocols, the jitter rate $J(t)$ of the system at time t is defined as $J(t) := \frac{N_j(t)}{N(t)}$, where $N_j(t)$ is the total number of users that have experienced block jitter by time t; the user $\mathcal{U}_i$ is said to have experienced block jitter if, for at least one $m \in \{1, 2, \ldots, M-1\}$, $T_{i,m+1} - T_{i,m} > A/\beta$.*

Clearly, both the waiting rate and the jitter rate are measures of QoS, and one expects a good system will have these measures quickly diminish to zero with time t.

7.1.1 Simulation Setup

As most Internet users have ADSL or Ethernet access, to be conservative, we use the following set of values for the simulation as the default values, unless otherwise specified in the figures presented: the server's output bandwidth s is set to 250 Mbps, and each user's input and output bandwidth are respectively 5 and 1 Mbps.

The user arrival process is simulated according to a Poisson process with tunable parameter λ ranging from 1/32 to 1/4. The user buffer size K ranges from 1 to 30 units of the video block.

DVD-quality video content is considered, where we set the playback rate β to 3.75 Mbps. The video block size A is fixed to 14 Mbytes or the equivalent to 30 seconds of playback time. The number of video blocks M is simulated to up to 30. Every fountain

codeword symbol is assumed to be a UDP packet. The overhead of fountain codes is fixed to 0.03.

The underlying network topology is generated using GT-ITM [90]. The whole network consists of 200 routers and more than $1,000$ links. We assume that each router in this network represents a local area network with the ability to host an unlimited number of users. The routing between two routers is determined using the shortest path algorithm. On all links of the network, the packet-loss rate is a tunable parameter γ. For every packet sent on a link, packet loss is simulated as an independent Bernoulli event (among all links and across all packets) parameterized by γ.

7.1.2 Dynamics Measurement

The simulated production achieved by Protocol B and the lower bounds of Theorem 1 are plotted in Figure 7.1 (top), together with their derivatives (bottom).

First, one observes that the saturation time $\widehat{T}_m$ is much earlier in Protocol B than that in Protocol A presented in Chapter 4. Second, these results suggest that the production for a different block m does not follow the same growth curve, demonstrating much more complex dynamics. Third, it is important to realize that the difference between Protocol B and Protocol A is primarily in how each user's output bandwidth is allocated across the video blocks, suggesting the importance of bandwidth allocation in peer-to-peer video-streaming systems. Finally, it is worth noting that Protocol B performs quite well and is already a nearly working protocol for high-quality (DVD) video streaming, particularly when one notices that the parameters chosen are quite close to the realistic settings.

7.1.3 QoS Measurement

Plotted in Figure 7.2 through Figure 7.5 are the QoS of the system measured in terms of jitter rate $J(t)$ and waiting rate $W(t)$ and the dependency of these QoS measures on buffer size, link-loss rate, and user-arrival rate.

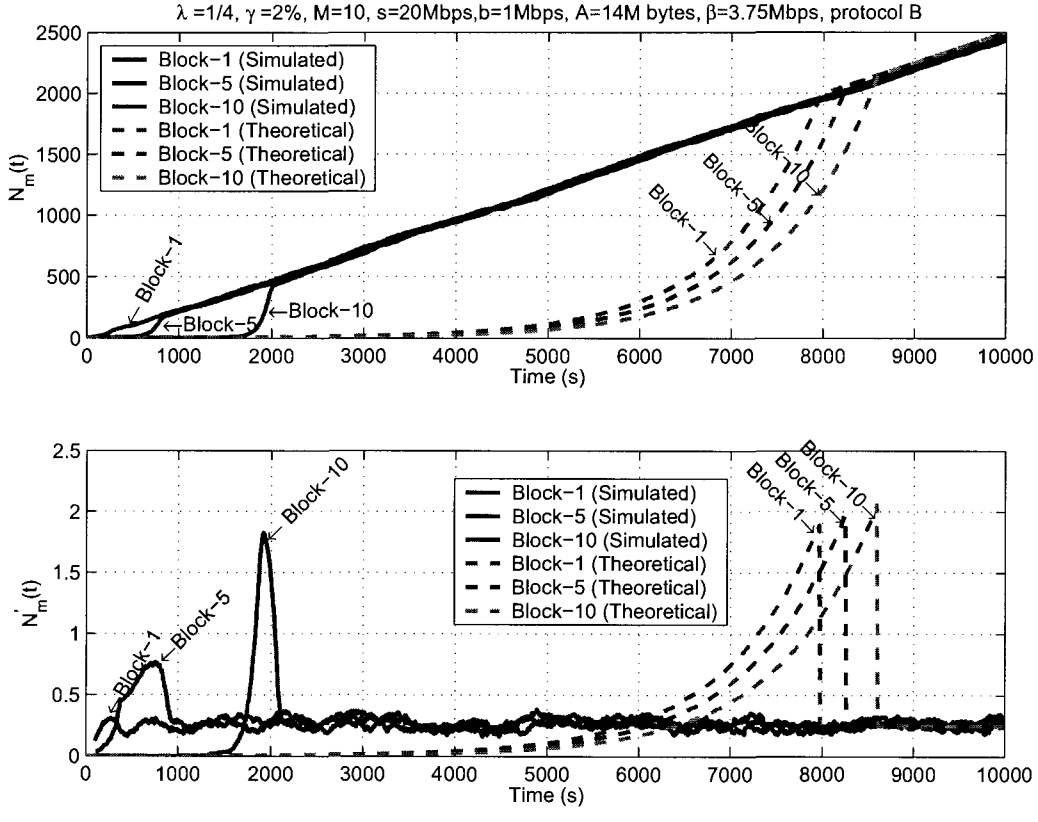


Figure 7.1: Simulated production achieved by Protocol B in comparison with the lower bounds of Theorem 1 (top); the derivative of simulated production achieved by Protocol B and that of the lower bounds of Theorem 1 (bottom).

In Figure 7.2, we see that these two QoS measures are well correlated. In particular, with the arrival rate $\lambda = 1/16$, the waiting rate for different $\gamma = 2\%,\ 5\%$, measured up to $t = 10$ hours, is virtually zero, while the buffer size increases. It is worth noting that, since the $J(t)$ and $W(t)$ are accumulated measures during time interval $[0, t]$, $J(10$ hours$)$ and $W(10$ hours$)$ being close to zero do not exclude the possibility that earlier arriving users may still suffer from severe jitters and a long waiting time. Nevertheless, from the plots one expects that this only happens in the short initial period during which the system is UNSAT. We will return to the time-varying characteristics of jitter rate and waiting rate in a later discussion. The usual trend that QoS degrades with packet-loss rate is also demonstrated in Figure 7.2.

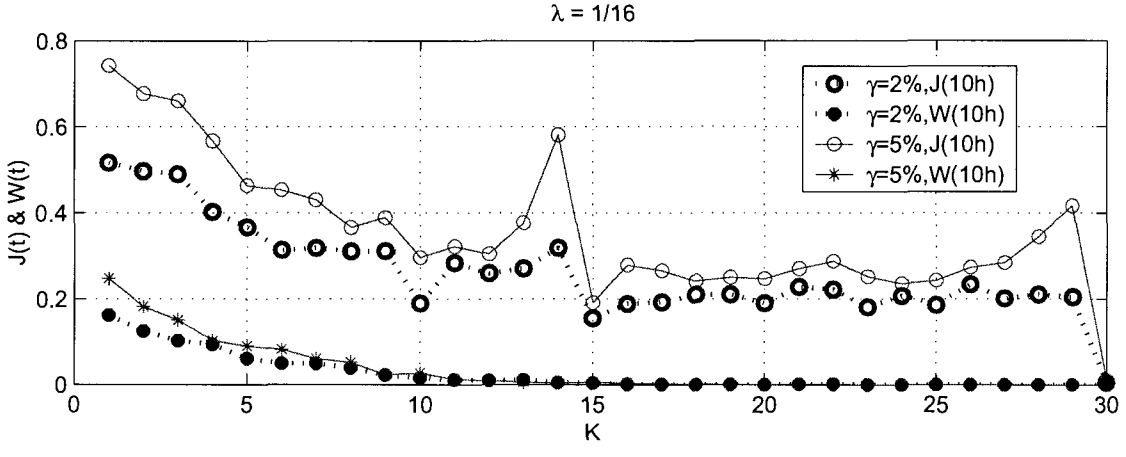


Figure 7.2: Jitter rate $J(t)$ and waiting rate $W(t)$ of Protocol B measured at time $t = 10$ hours plotted against user buffer size K.

In Figure 7.3, we see that with sufficient buffer size ($K = 30, M = 30$, essentially enough to store all blocks) and when the arrival rate is 1/16 or lower, the system QoS stays in an acceptable range across a large range of link-loss rates. In Figure 7.3, one notices that the waiting rate can be much better than the jitter rate, since a long waiting time may be understood as a special case of jitters, i.e. one defined only for the first block.

In Figure 7.4, we see the expected behaviour that QoS improves as the user-arrival

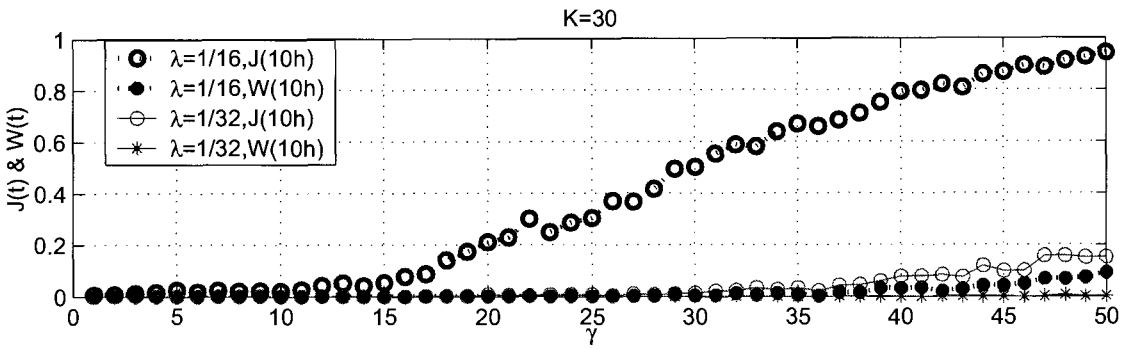


Figure 7.3: Jitter rate $J(t)$ and waiting rate $W(t)$ of Protocol B measured at time $t = 10$ hours plotted against link-loss rate.

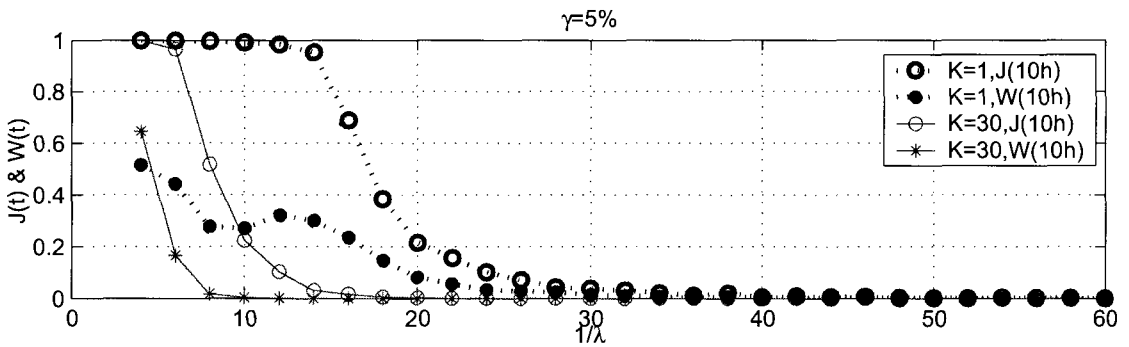


Figure 7.4: Jitter rate $J(t)$ and waiting rate $W(t)$ of Protocol B measured at time $t = 10$ hours plotted against user-arrival rate.

rate decreases. This is because the decrease of arrival rate decreases the growth rate of requests for every block. Additionally from Figure 7.4, we observe a sharp drop in the QoS measures, particularly for large K, as the arrival rate decreases. One can then reason that there exists a maximum arrival rate that the system can support. In fact, we conjecture that such behaviour exists universally and is a fundamental aspect of the studied problem. We believe it is impossible to design a protocol that achieves a strictly zero jitter rate and a waiting rate for *all* arrival rates at the starting phase of the system unless the server has virtually unlimited bandwidth and can serve the arriving requests by itself. This is justified by the fact that, if the system starts from 1-UNSAT state, it must take some time to leave the state. It is then of great interest to characterize the threshold arrival rate below what the system is able to support. It is also worth investigating how much earlier an optimal protocol can push the system, with a given arrival rate, to depart from the 1-UNSAT phase while maintaining small spacing between saturation times $\widehat{T}_m$ for later consecutive blocks.

We also investigate the dynamics of both the jitter rate and the waiting rate. The results in Figure 7.5 demonstrate the behaviours of $J(t)$, $W(t)$, and their derivatives for $K = 10$. The peaks of $J(t)$ and $W(t)$ and the corresponding dropping of the derivatives occur after the system leaves m-UNSAT phases for the first several m.

It is remarkable that these performance measures achieved by Protocol B are for the settings of finite buffer size and limited user input bandwidth. This makes fountain-coded transport protocols an appealing option for real-time video streaming for practical purposes. As Protocol B is only heuristic, one expects further improved performance to be attainable by the fountain-coded protocols. Even with Protocol B, we see that the system already demonstrated good performance for arrival rates lower than 1/16, under very stringent QoS measures.

To further understand the effects of the input bandwidth of users, we also chose four sets of requested downloading bandwidth r setting of users ($3.75Mbps$, $5Mbps$, $10Mbps$, and $25Mbps$) to investigate the QoS performance of Protocol B. The results are presented

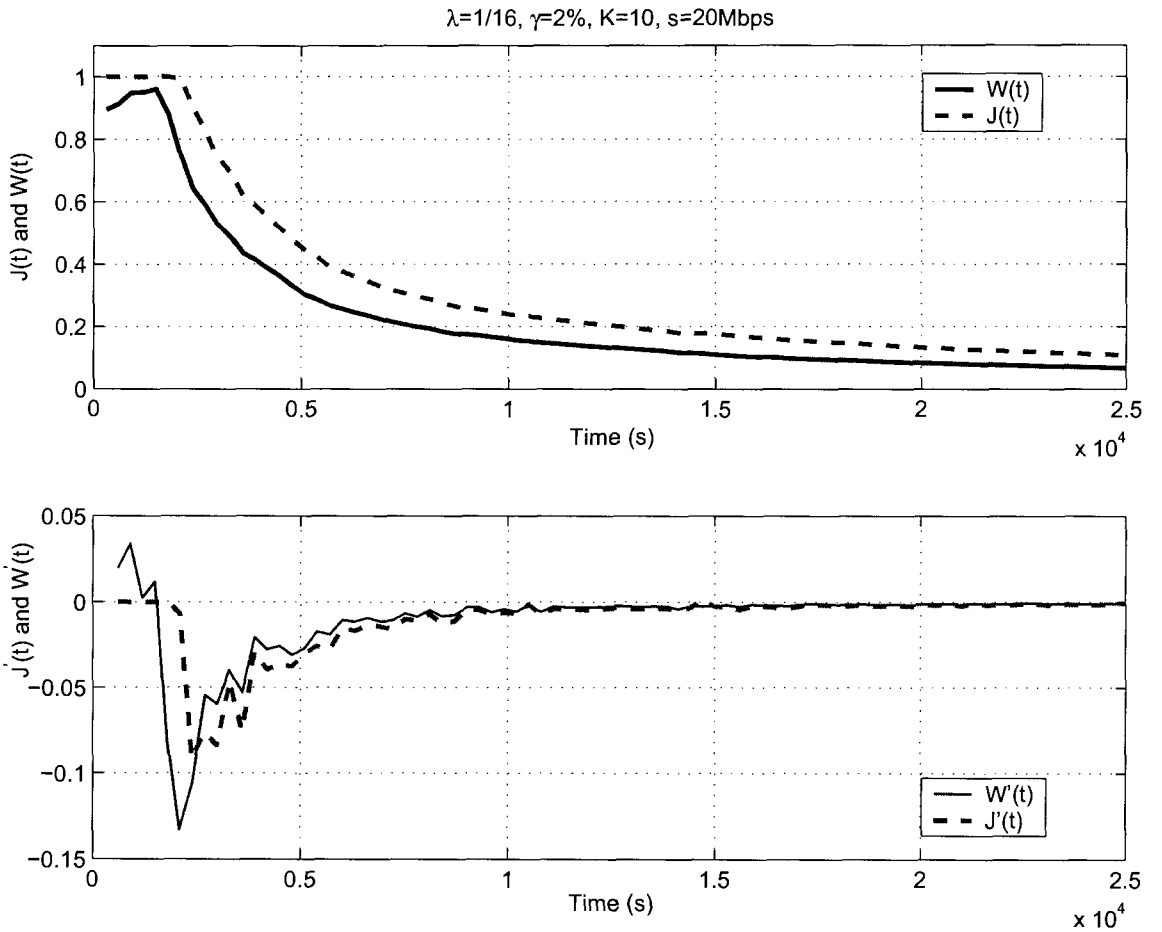


Figure 7.5: Jitter rate $J(t)$ and waiting rate $W(t)$ of Protocol B as functions of time (top) and their derivatives (bottom).

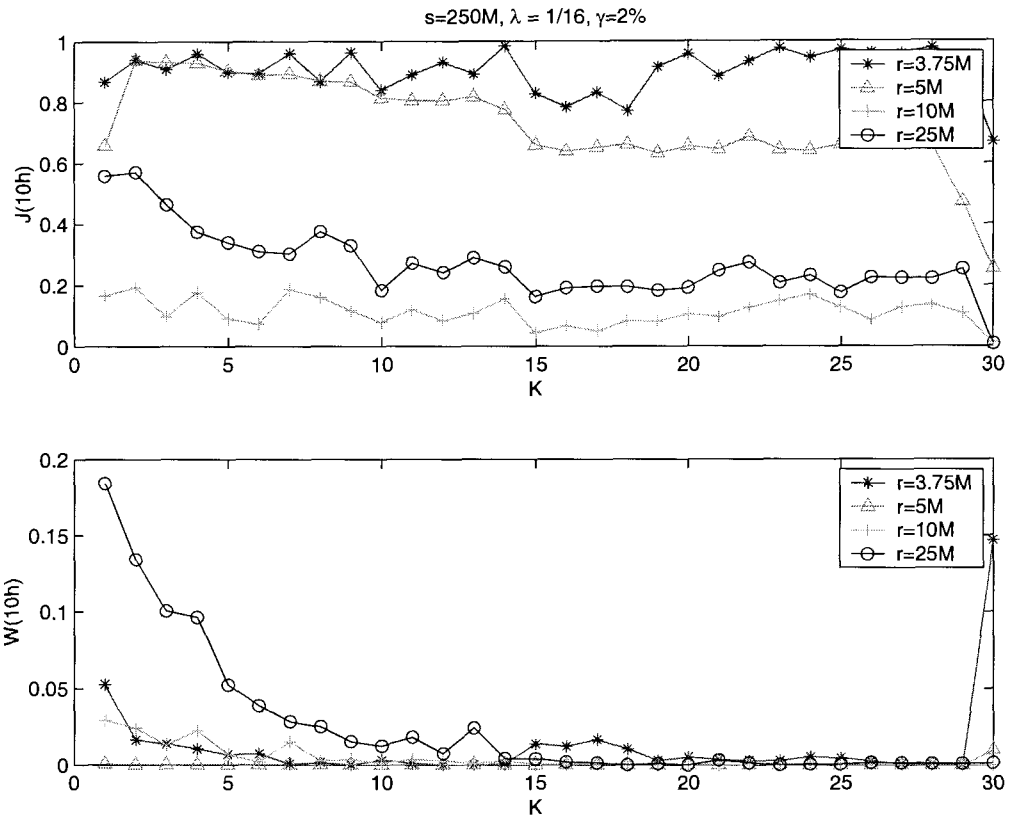


Figure 7.6: Jitter rate and waiting rate comparison for different requesting downloading bandwidth.

in Figure 7.6.

From the $J(10h)$ with four different sets of r, we can find that Protocol B has the best performance when $r = 10Mbps$. It indicates that the performance becomes better at the beginning while the r increases, but it becomes poor again after r is greater than $10Mbps$ or a value between $10Mbps$ and $25Mbps$. The reason is that the total of the system output bandwidth of the system has not been completely used when $r < 10Mbps$. During this stage, more output bandwidth will be used while r increases. We also note that $W(10h)$ remains the same during this period, which also implies this fact. But when r is greater than $10Mbps$ or a value after which the total output bandwidth already has been completely used, the increased input bandwidth does not help users to obtain more

resources. On the other hand, it will cause $W(t)$ to increase, since a newly arrived user needs to wait for the free output bandwidth available at this time. The result of $W(10)$ at $r = 25Mbps$ matches this, and $J(10)$ of $r = 25Mbps$ is in the same situation, because users wait for available output bandwidth not only for the first block, but for the latter block.

Also, we can note another fact: the server output bandwidth mainly affects the system at the beginning or before the system leaves the unsaturated stage, since the server output bandwidth ($250Mbps$) is much larger than the users' output bandwidth ($1Mbps$). In our theoretical analysis for the lower bound section, we assume the server transmits to only one user. If this is the case, the user input bandwidth will not affect the performance, because the output bandwidth has always been completely used at all times, for the input bandwidth is larger than the output bandwidth for every user, i.e. $r > b$. However, in practical Protocol B, at time 0, the results show that the system is in the unsaturated stage. The user input bandwidth needs to be considered to improve performance. It is the only way to let the system pass the unsaturated stage with QoS control that relies on the server's output bandwidth. In other words, an algorithm is also needed to set an input bandwidth in terms of QoS.

7.2 Performance Evaluation of rVoD

Based on the results of Protocol B, we realize that (1) the rateless coding is efficient to improve the performance of the P2P streaming system compared to the lower bound in Theorem 1; (2) Protocol B is efficient to achieve high QoS of the system; (3) a storage management scheme is needed to further improve the performance of the system; and (4) the large uploading bandwidth of the video server is only needed when the number of online users is small. Thus, we proposed rVoD. In rVoD, we apply the rateless coding for the storage allocation so that every user can store encoded video information that it played with its small memory size. After performing the simulation of Protocol B, we are

more interested in investigating the performance of the P2P system under a controlled QoS setting for commercial reasons. Thus, we assume the uploading bandwidth of the video server is large enough that the system is not in the unsaturated stage all time so that the QoS is guaranteed. Then we mainly measure the consuming of the video server to investigate how practical the system is. To confirm that the performance of the proposed system is efficient, the results of the simulation have been compared to VMesh [28].

7.2.1 Network Setup

We have evaluated rVoD using packet-level event-driven simulation. We have implemented uniform distribution-based random memory allocation (DPCC-RAND) and fixed-size memory allocation (DPCC-FIX) as well as the requesting popularity-based storage allocation algorithm (DPCC-PB) to compare the performance. We use Raptor codes [25] for rateless coding and take the overhead of Raptor codes $\epsilon = 2\%$. The underlying network topology is generated using GT-ITM [90]. The whole network consists of 600 routers and more than $2,000$ links. We assume that each router in this network represents a local area network with the ability to host an unlimited number of users. The routing between two routers is determined using the shortest path algorithm. We assume the link loss rate varies from 0 to 3%.

In our simulations, the length and playback rate of a video are $L = 7200$ seconds, and the video playback rate is $R = 1$ Mbps. Every user has five minutes of video storage size or $K_s = 300$ seconds. The user arrival process follows a Poisson distribution with an average arrival rate λ. To measure the system performance of different user populations, we take λ as a function that increases exponentially with time t, and we measure online users from zero to 3200. Every user is randomly attached to a router node and is assigned an uploading bandwidth of at most $2R$ Mbps.

We model the online time of each user exponentially with a mean of T seconds. Each user continues serving others until it is offline, and it even finishes playing the last video

block. According to the statistics in [91], each user randomly performs a VCR operation zero to seven times in its lifetime. In order to compare to VMesh [28], we also implement a VMesh population-based scheme under our network setting and model the user jump distance as a Pareto distribution, which is the same as VMesh.

We use the following performance metrics to evaluate our system:

- **Server stress** - the outgoing bandwidth required at the video server to support the system. In our results, we measure the server stress by the number of streams, and one stream equals the playback rate of the video (R Mbps). For a specific number of requesting users online, the lower the server stress, the more scalable the system is.

- **Startup latency** - the time period starting from the instant that a user joins the system to the instant the user starts playing the video. In our results, we measure it by how many contacts the user tried since it joined the system till it obtains the required downloading rate that equals the video playback rate R.

- **VCR latency** - the time period starting from the instant that a user performs a VCR operation to the instant that the user starts playing the video in the new position. We measure it by the contacts that the user tried after the user submits a VCR operation to obtain the required downloading rate R for the new position of the video.

- **Block seeking overhead** - the overhead of the control protocol to locate the next video block for the normal playing of the video. In our results, we measure it by the contacts that the user made to locate the next video block for normal playing of the video. More contacts indicates that it costs the system more communication to maintain content distribution paths.

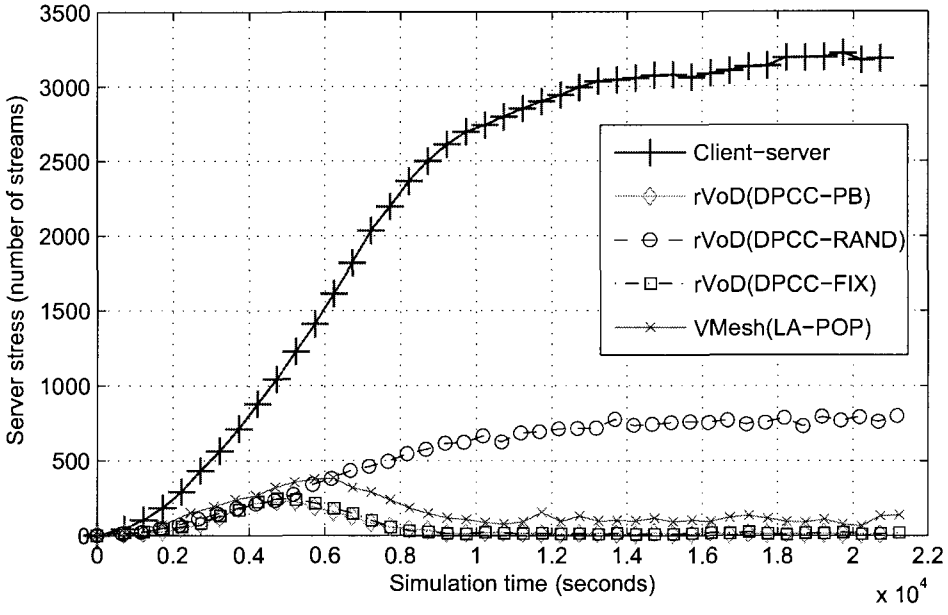


Figure 7.7: Server stress against simulation time (in seconds).

7.2.2 Small Server Stress

Our experiments evaluated the performance of our system in saving server resources and bandwidth by server stress. We measured the server stress using the number of streams it can handle, where one stream equals the playback rate of the video (R Mbps). We continually ran our system for eight hours under varied user arrival rates and measured the video server stress defined above for average online users from 0 to 3,200. Figure 7.7 shows how scalable our rVoD schemes are in terms of server stress compared to the traditional client-server model. The results tell us that rVoD can support a large number of users by using a small server uploading bandwidth, and the server uploading bandwidth can be very small when there are enough users in the system.

The detailed server stress reduction between the proposed rVoD based on three different memory allocation schemes and VMesh is presented in Figure 7.8. The results show that both rVoD (DPCC-PB) and rVoD (DPCC-FIX) have better performance compared to VMesh for all simulation times when the average user population is increasing from 0 to 3,200. The results also indicate that the server stress remains quite small after enough

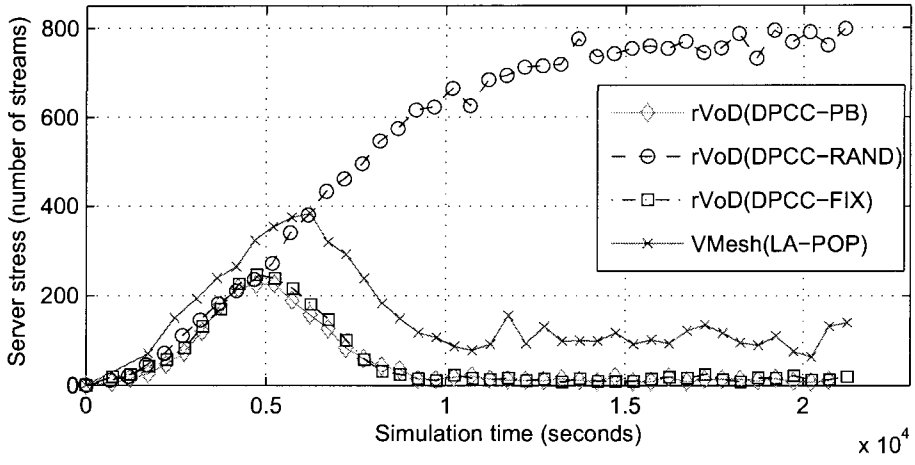


Figure 7.8: Server stress compared to VMesh.

average online users are in the system, which is after about 8,000 simulation seconds in the figure. Clearly, VMesh reaches that time point later than rVoD (DPCC-PB) and needs a great deal of server resources to maintain the system. The results also show that the rVoD (DPCC-RAND) scheme has better performance at the beginning, but has the worst performance when the number of online users is large compared to VMesh. That is because the rVoD (DPCC-RAND) may allocate zero storage memory size for some video blocks randomly; the result is similar to the VMesh (RAND) scheme [28]. Both rVoD (DPCC-PB) and rVoD (DPCC-FIX) have similar performance for all simulation times, and rVoD (DPCC-PB) only achieves better performance at the beginning compared to rVoD (DPCC-FIX). That is because we took the fixed storage size of rVoD (DPCC-FIX) for rVoD (DPCC-PB)'s X_{min} parameter.

7.2.3 Small Latency

We measured the latency related to user arrival operation (startup latency) and VCR operations (VCR-operation latency). VCR-operation latency is measured by the contacts that the user tries after the user submits a VCR operation to obtain the required downloading rate R for the new position of the video. The results in Figure 7.9 show

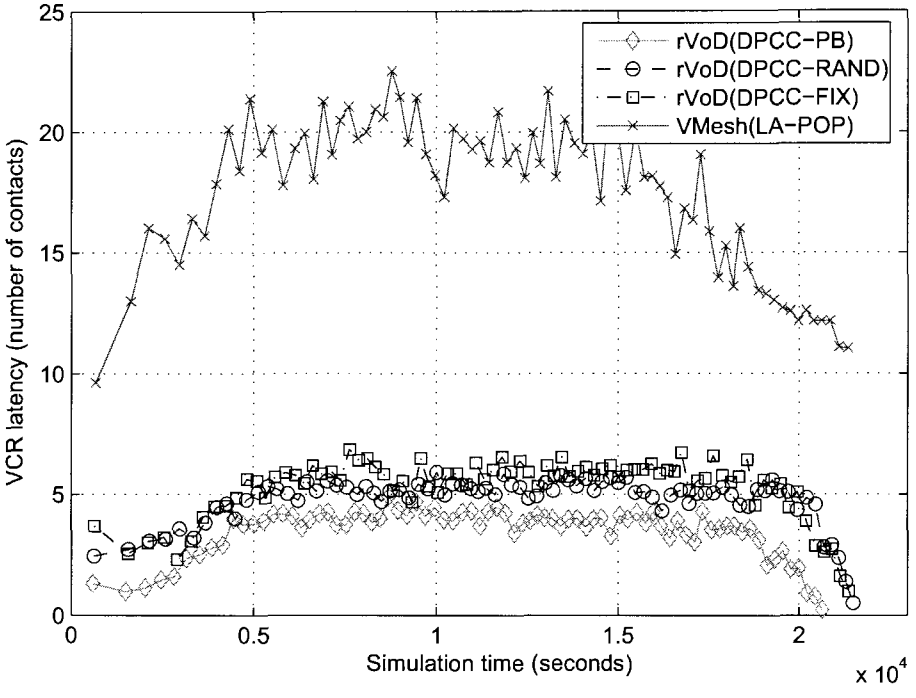


Figure 7.9: VCR latency against simulation time (in second).

that all schemes of rVoD have much better performance than VMesh in terms of VCR-operation latency. The results show key improvement compared to VMesh. In rVoD, the user stores the partial information of all blocks that it played, but the user only stores some of the blocks in VMesh. Thus, our method reduces the VCR-operation latency by saving a DHT searching operation of VMesh because a supplying user can be a supplier of the receiver for its lifetime.

Startup latency is measured by how many contacts the user tried since it joined the system till it obtains the required downloading rate that equals the video playback rate R. The performance of the system in terms of startup latency is presented in Figure 7.10. The results show that rVoD (DPCC-PB) has a small number of contacts to start to play video content. It is also smaller compared to VMesh, since rVoD uses the centralized server for locating the first video block.

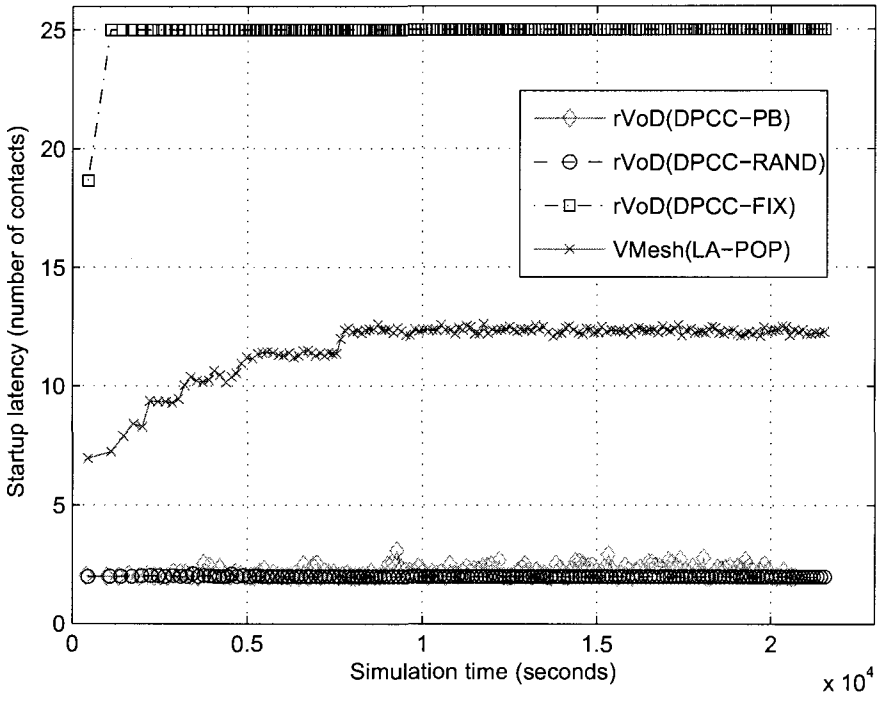


Figure 7.10: Startup latency against simulation time (in second).

7.2.4 Low Control Overhead

The overhead of a protocol is a very important parameter to tell whether the system is efficient. In general, the overhead of the P2P system includes coding overhead and all kinds of control messages, such as block seeking, data scheduling, and peer locating.

We mainly measure the block-seeking overhead in our experiment, which is defined as the number of contacts that the user made to locate the next video block for normal video playing. The metric indicates the overhead cost to maintain content streaming paths. The simulation results, depicted in Figure 7.11, show that all three schemes of rVoD have a very low overhead, almost zero. VMesh has at least one contact cost, and its peak is almost six contacts, when the number of online users is not big.

For other kinds of overhead, coding overhead is fixed at about 2% of the video content. The control message of data scheduling becomes almost zero for applying rateless coding. With regard to the control message of user locating, rVoD definitely has a low control

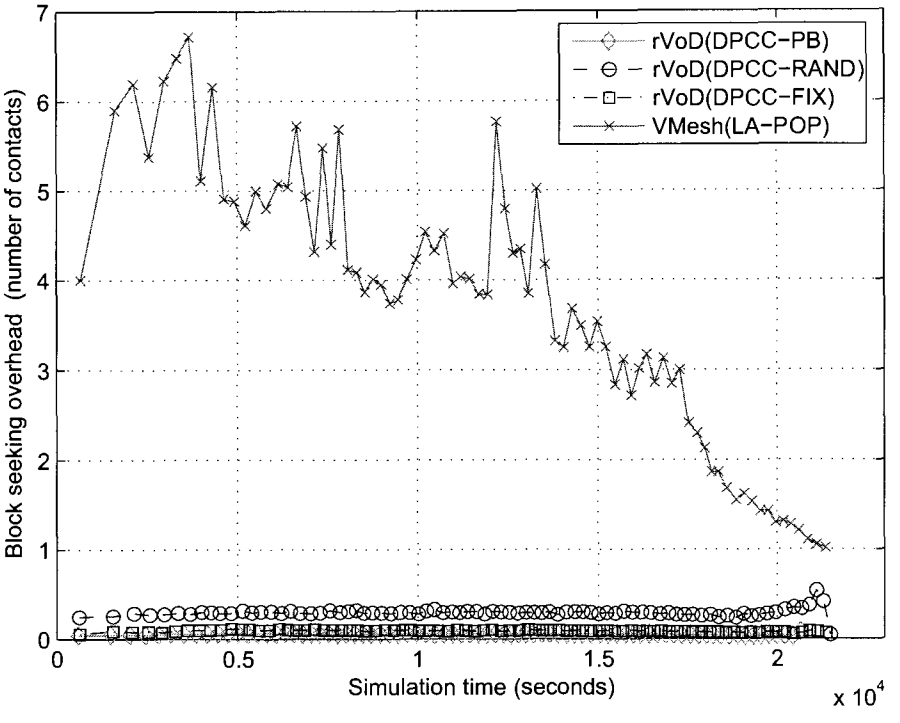


Figure 7.11: Block seeking overhead against simulation time (in seconds).

overhead compared to VMesh, because rVoD does not need to search for suppliers in the whole network using a DHT table but VMesh uses a DHT table. Also, rVoD does not need to change supplying users for downloading different video blocks in most cases, but VMesh need to find new supplying users for each video block.

Thus, rVoD has smaller overhead by adding the above overhead together, compared to VMesh. Particularly, we note that rVoD's overhead is not increasing quickly, while the number of online users grows quickly.

Chapter 8

Conclusions and Future Directions

8.1 Conclusions

The problem of video on-demand (VoD) streaming via peer-to-peer (P2P) overlay networks has been studied in this thesis, where we have proposed a novel distributed encoded video content caching scheme and a transportation protocol based on rateless codes. Some important properties and theoretical results of P2P streaming networks are derived.

First, a multi-video-block transport framework for streaming high-QoS content based on P2P networks is proposed. In this novel framework, the video content (or file) is divided into small blocks (or clips) and then transmitted in sequence. Each small block distribution becomes a sub-P2P streaming system. The advantage of the framework is that we can model and measure the dynamic performance of the P2P streaming system.

Second, notions of *production* and *saturation* are introduced to measure the dynamics of the system capacity of P2P streaming networks at any time. We believe that such defined indices can adequately catch the performance of a realistic P2P streaming system. We theoretically analyze the dynamics of the system and QoS performance within the proposed framework. The throughput of the presented framework is shown analytically to increase with time in a manner no worse than an exponential function

prior to the saturation time. The notions of production and saturation time may also serve as fundamental concepts for the study of quality of service (QoS) for the proposed framework.

Third, a novel resilient P2P VoD system, namely rVoD, which supports VCR operations with low startup latency and low overhead is proposed. In rVoD, a novel distributed encoded video content caching scheme and a transportation protocol based on rateless codes are proposed. rVoD can efficiently solve the problems of supporting VCR operations and data-scheduling among multi-senders.

Fourth, a packet-based P2P simulation system to evaluate our proposed P2P streaming system is developed. We examined the dynamic performance of the system in terms of system throughput and QoS metrics given the limited uploading capacity of the server. We also measured the server stress, latency, and overhead when QoS requirements are given.

Fifth, the experiment results are also compared to the previously proposed VMesh [28]. Results of the simulations demonstrate the advantages of our proposed P2P VoD system and confirm our theoretical analysis.

Finally, a necessary condition-based *saturation time* and the relationship between *production* and *server stress* are derived for a general optimal P2P streaming system. And the theoretical analysis results of QoS tradeoff are provided. The results give us guidelines to designing a general P2P streaming system.

8.2 Future Directions

The research field of P2P streaming is interdisciplinary in nature, involving communication theory, algorithms, information theory, coding theory, optimization theory, etc. Through this thesis, we mainly explore the P2P on-demand streaming problem. We design a P2P VoD system by applying rateless coding and derive some properties of the proposed system on system throughput, QoS, and optimization etc. In general, there

are many interesting topics about P2P streaming. In the following, we list several future topics.

- It would be very interesting to extend our proposed system for multi-source on-demand systems. Are the derived theoretical results, such as the lower bound of production in Chapter 3, still correct? If not, how can we extend them?
- Our theoretical analysis is mostly based on the deterministic model for average cases. Is it possible to extend the stochastic model and derive similar results?
- Designing a P2P VoD streaming system that supports different video quality levels for different users would be also an interesting work. To implement this, is it possible to use join coding technology, such as MDC and rateless coding?
- In Chapter 5, we derive a necessary condition to implement an optimal system and the relationship of maximum production and minimum sever stress for an optimal system. Is it possible to find necessary and sufficient conditions for an optimal system? For example, we may explore a specific stage of the system.
- Lots of interesting topics in the P2P streaming area are open, such as security problems, source locating, and P2P accountability.

Bibliography

[1] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm, "CDN brokering," *Proc. of WCW*, June 2001.

[2] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai, "Analysis of a CDN-P2P hybrid architecture for cost-effective streaming distribution," *ACM/Springer Multimedia Systems*, vol. 11, no. 4, pp. 383–399, 2006.

[3] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," *Proc. of IEEE INFOCOM*, pp. 2235–2245, 2005.

[4] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans." *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, 1990.

[5] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," *Symposium on Operating Systems Principles*, pp. 131–145, 2001.

[6] "Napster," *www.napster.com.*

[7] "BitTorrent," *www.bittorrent.com.*

[8] "PPLive," *www.pplive.com.*

[9] "TVants," *www.tvants.com.*

[10] "ppStream," *www.ppstream.com.*

[11] "FeiDian," *tv.net9.org.*

[12] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," *Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[13] Y. Tu, J. Sun, M. Hefeeda, and S. Prabhakar, "An analytical study of peer-to-peer media streaming systems," *ACM TOMCCAP*, vol. 1, pp. 354–376, November 2005.

[14] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," *Proc. of IEEE INFOCOM*, 2007.

[15] C. Wu and B. Li, "rStream: Resilient peer-to-peer streaming with rateless codes," *Proc. of ACM Multimedia*, pp. 307–310, November 2005.

[16] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE JSAC special issue on Recent Advances in Service Overlay*, vol. 22, no. 1, pp. 91–106, January 2004.

[17] C. Huang, J. Li, and K. W. Ross, "Peer-Assisted VoD: Making Internet video distribution cheap," *IPTPS*, 2007.

[18] Z. Liu, Y. Shen, S. Panwar, K. W. Ross, and Y. Wang, "Efficient substream encoding and transmission for P2P video on demand," *Packet Video 2007*, pp. 143–152, November 2007.

[19] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay," *Proc. of IEEE INFOCOM*, March 2005.

[20] D. A. Tran, K. Hua, and T. Do, "A peer-to-peer architecture for media streaming," *IEEE J. Select. Areas Commun.*, vol. 22, pp. 121–133, January 2004.

[21] H. Jenkac, T. Stockhammer, W. Xu, and W. A. Samad, "Efficient video-on-demand services over mobile datacast channels," *Journal of Zhejiang University SCIENCE A, Special Issue for Selected Papers (Part I) of 15th International Packet Video Workshop*, vol. 7, no. 5, pp. 873–884, May 2006.

[22] S. Annapureddy, C. Gkantsidis, and P. Rodriguez, "Providing video-on-demand using peer-to-peer networks," *Proc. Internet Protocol Television (IPTV) workshop in conjunction with WWW '06*, 2006.

[23] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, pp. 371–381, February 2003.

[24] M. Luby, "LT codes," *IEEE Symposium on Foundations of Computer Science*, pp. 271–282, 2002.

[25] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[26] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and X. Wen, "Raptor codes for reliable download delivery in wireless broadcast systems," *Proc. of IEEE CCNC*, vol. 1, pp. 192–197, January 2006.

[27] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, July 2000.

[28] W.-P. Yiu, X. Jin, and S.-H. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE J. Select. Areas Commun.*, vol. 25, no. 9, pp. 1717–1731, December 2007.

[29] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM: Applications, technologies, architectures, and protocols for computer communications*, pp. 149–160, 2001.

[30] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *In Proceedings of IFIP/ACM Middleware*, 2001.

[31] S. Deering, "Host extensions for ip multicasting," *RFC 1112*, August 1989.

[32] F. Baker, "Requirements for ip version 4 routers," *RFC 1812*, June 1995.

[33] Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper, "IANA guidelines for IPv4 multicast address assignments," *RFC 3171*, August 2001.

[34] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," *RFC 1546*, November 1993.

[35] P. Mockapetris, "Domain names - concepts and facilities," *RFC 1034*, November 1987.

[36] R. G. Gallager, "Low density parity-check codes," *MIT Press, Cambridge, MA*, 1963.

[37] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, March 1997.

[38] R. Puri and K. Ramchandran, "Multiple description source coding using forward error correction codes," *Conf. on Signals, Systems, and Computers*, October 1999.

[39] V. Goyal, "Multiple description coding: compression meets the network," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 74–93, September 2001.

[40] H. Bai, Y. Zhao, and C. Zhu, "Priority encoding transmission based multiple description video coding over packet loss network," *Data Compression Conference*, pp. 504–504, March 2008.

[41] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *Information Theory, IEEE Transactions on*, vol. 42, no. 6, pp. 1737–1744, November 1996.

[42] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, 2003.

[43] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *Proc. IEEE Int. Symp. on Inform. Theory*, June 2003.

[44] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, pp. 1973–1982, 2005.

[45] Y. Wu, V. Stankovi, Z. Xiongy, and S. Kung, "On practical design for joint distributed source and network coding," *Proc. of the First Workshop on Network Coding(NetCod)*, 2005.

[46] "Gnutella," *www.gnutella.com.*

[47] "USENET," *www.usenetnewsgroup.net.*

[48] "FidoNet," *www.fidonet.org.*

[49] A. J. Ganesh, A. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Trans. Comput.*, vol. 52, no. 2, pp. 139–149, 2003.

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

[51] T. Do, K. Hua, and M. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," *IEEE International Conference on Communications*, vol. 3, pp. 1467–1472, June 2004.

[52] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," *Proc. of ACM SOSP*, 2003.

[53] Y. Guo, S. Yu, H. Liu, S. Mathur, and K. Ramaswamy, "Supporting VCR operation in a mesh-based P2P VoD system," *Proc. of IEEE CCNC*, pp. 452–457, January 2008.

[54] D. S. Wallach, "A survey of peer-to-peer security issues," *International Symposium on Software Security*, pp. 42–57, 2002.

[55] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the sybil attack," University of Massachusetts Amherst, Amherst, MA, Tech report 2006-052, October 2006.

[56] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. M. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 282–300, March 2001.

[57] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proc. of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

[58] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," *Proc. of IEEE INFOCOM*, 2004.

[59] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *ACM SIGCOMM*, August 2004.

[60] K. K. Ramachandran and B. Sikdar, "An analytic framework for modeling peer-to-peer networks," *Proc. of IEEE INFOCOM*, pp. 215–269, March 2005.

[61] F. C. Perronnin, P. Nain, and K. W. Ross, "Multiclass P2P networks: Static resource allocation for service differentiation and bandwidth diversity," *Perform. Eval.*, vol. 62, no. 1-4, pp. 32–49, 2005.

[62] J. Mundinger, R. R. Weber, and G. Weiss, "Analysis of peer-to-peer file dissemination amongst users of different upload capacities," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 2, pp. 5–6, 2006.

[63] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *IEEE J. Select. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, 2002.

[64] S. E. Deering, "Multicast routing in internetworks and extended lans," *ACM SIGCOMM: Symposium proceedings on Communications architectures and protocols*, pp. 55–64, 1988.

[65] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *ACM SIGCOMM*, August 2002.

[66] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE J. Select. Areas Commun.*, vol. 20, no. 8, pp. 1489–1499, 2002.

[67] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *ACM/IEEE NOSSDAV*, May 2002.

[68] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, "Using random subsets to build scalable network services," *Proc. of USITS*, 2003.

[69] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," *ACM SIGCOMM*, August 2002.

[70] C. Huang, R. Janakiraman, and L. Xu, "Loss-resilient on-demand media streaming using priority encoding," *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 152–159, 2004.

[71] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," *ACM/IEEE,NOSSDAV*, 2003.

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

[73] 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*, May 2003.

[74] T. T. Do, K. A. Hua, and M. A. Tantaoui, "Robust video-on-demand streaming in peer-to-peer environments," *Comput. Commun.*, vol. 31, no. 3, pp. 506–519, 2008.

[75] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding P2P system," *ACM SIGCOMM/USENIX IMC'06*, October 2006.

[76] P. Chou, Y. Wu, and K. Jain, "Practical network coding," *51st Allerton Conf. Communication, Control and Computing*, 2003.

[77] Y. Zhu, B. Li, and J. Guo, "Multicast with Network Coding in Application-Layer Overlay Networks," *IEEE J. Select. Areas Commun.*, vol. 22, no. 1, pp. 107–120, January 2004.

[78] M. Wang, Z. Li, and B. Li, "A high-throughput overlay multicast infrastructure with network coding," *Proc. of the Twelfth IEEE International Workshop on Quality of Service (IWQoS)*, 2005.

[79] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, 1989.

[80] C. Wu and B. Li, "rStream: Resilient and optimal peer-to-peer streaming with rateless codes," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, pp. 77–92, January 2008.

[81] J. Li, "PeerStreaming: A practical receiver-driven peer-to-peer media streaming system," *Microsoft Research MSR-TR-2004-101, Tech. Rep.*, September 2004.

[82] "VVSky," *www.vvsky.com.cn.*

[83] M. Fuss and D. L. McFadden, *Production Economics: A Dual Approach to Theory and Applications*, Amsterdam: North-Holland, 1978.

[84] D. Lou, Y. Mao, and T. Yeap, "The production of peer-to-peer video-streaming networks," *ACM SIGCOMM: P2P-TV*, August 2007.

[85] D. M. Chiu, R. Yeung, J. Huang, and B. Fan, "Can network coding help in P2P networks?" *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium*, pp. 1–5, April 2006.

[86] J. Mundinger, R. Weber, and G. Weiss, "Analysis of peer-to-peer file dissemination," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 3, pp. 12–14, 2006.

[87] D. Lou and T. Yeap, "Resilient video-on-demand streaming over P2P networks," *IEEE HotWeb 2008*, October 2008.

[88] L. Guo, S. Chen, and X. Zhang, "Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 5, pp. 669–682, May 2006.

[89] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray, "Asynchronous distributed averaging on communication networks," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 3, pp. 512–520, June 2007.

[90] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," *Proc. of IEEE INFOCOM*, vol. 2, pp. 594–602, March 1996.

[91] C. Zheng, G. Shen, and S. Li, "Distributed prefetching scheme for random seek support in peer-to-peer streaming applications," *P2PMMS'05: Proceedings of the ACM workshop on advances in peer-to-peer multimedia streaming*, pp. 29–38, 2005.