THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學
Pao Yue-kong Library
包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

Pao Yue-kong Library, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

http://www.lib.polyu.edu.hk

The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Efficient Video Streaming in Peer-to-Peer Networks

by

ZHANG Lei

A thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Philosophy

August 2004

Pao Yue-kong Library
PolyU · Hong Kong

CERTIFICATE OF ORIGINALITY

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

_____ (Signed)

ZHANG Lei (Name of student)

Abstract

of the thesis entitled

"Efficient Video Streaming in Peer-to-Peer Networks"

submitted by ZHANG Lei

for the degree of Master of Philosophy

at The Hong Kong Polytechnic University in August 2004

Video streaming is one of the rapidly emerging applications on the Internet. Most of the current streaming solutions are using the client/server architecture in which a dedicated server is used to serve a number of clients over the network. With this model, the only way to increase the system capacity is to use multiple servers and increase their resources such as network bandwidth and storage capacity. To increase the scalability of a video streaming system, many different approaches have been proposed, of which the peer-to-peer (P2P) architecture has gained much attention in recent few years. The P2P model in file sharing/ downloading is very successful in Internet world. However, the work on P2P video streaming is on-going and there are still many problems to be solved before it can be widely deployed in the Internet. Hence, it motivates our work to study how to provide an efficient and reliable video streaming service in a P2P network in this thesis.

In this thesis, we first investigate the performance of the hybrid ARQ (Automatic

Repeat Request) technique in providing reliable data transfer in a video streaming system. To enhance the performance of the hybrid ARQ method, an adaptive scheme that can adjust the number of parity packets sent according to different video frames is proposed to fully utilize the transmission bandwidth. With the idea of hybrid ARQ, a new P2P architecture is developed for providing on-demand video streaming. In the proposed system, clients will normally receive video data through the P2P forwarding chain. The video server containing both original data packets and FEC parity packets is used not only as the parent of the first peer in the chain but also as a backup source responding for emergency request of peers when their buffer content is dropped to a low level. A parent-child exchange (PCX) routing mechanism is also developed to tackle the weak node problem in such a chain structure. Simulation results demonstrate the efficiency of the proposed P2P streaming system.

As the chain structure is not suitable for live video streaming due to the long server-to-peer path delay. A tree structure P2P system is also studied in the thesis. One important concern of such a system is the searching of the parent location for a newly arrived peer. To handle this problem, a new fast parent searching method based on network triangle inequality is then developed. Simulation results show that the complexity of the parent searching procedure is greatly reduced using the proposed method.

Acknowledgements

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

I would like to give special thanks to my supervisor, Dr. K.T. Lo. He gave me enough flexibility to search for the interesting research topics for my MPhil study. Help and advice can be received from him all the way in my research. He has also given supports to help me complete this Thesis.

I would like to thank Dr. Morris M. Z. Wang and Mr. HO King Man for their support providing advice on my work. I would also like to thank Mr. Phil Karn for providing the Reed-Solomon codec for my simulations. I would also thank all the members and staff in the Department of Electronic and Information Engineering and the Centre of Multimedia Signal Processing for providing me a comfortable and pleasant research environment, and the Hong Kong Polytechnic University for the financial support to carry out my research work.

Last but not the least, I want to express my truly thanks to my family, my friends and especially my love Yeye for their encouragement and support throughout these years.

Table of Contents

Abstract i

Acknowledgment iii

Table of Contents iv

List of Figures ix

List of Tables xi

List of Abbreviations xii

List of Publications on which this Research is Based xiv

Chapter 1 Introduction 1

1.1 Introduction 1

1.2 Motivations and Research Objectives 4

1.3 Thesis Organization 4

Chapter 2 Technological Background on Video Streaming and Peer-to-Peer (P2P) Networks 6

2.1 Issues on Internet Media Streaming 6

2.1.1 Bandwidth limitations 7

2.1.2 Jitter 8

2.1.3 Loss 9

2.1.4 TCP and UDP 9

2.1.5 RTP and RTCP 10

2.2 Video Streaming Using Client/Server Model 11

2.2.1 Architecture for Internet Video Streaming 11

2.2.2 TCP-Friendly Streaming Protocols 13

2.2.3 Error Concealment 15

2.2.3.1 Automatic Repeat Request (ARQ) 15

2.2.3.2 Forward Error Correction 17

2.2.3.3 Reed Solomon FEC coding 18

2.2.3.4 Multiple Description using Forward Error Correction (MD-FEC) 19

2.3 Video Streaming Using Peer-to-Peer (P2P) Model 21

2.3.1 Introduction 21

2.3.2 Peer-to-Peer Files Sharing 22

2.3.2.1 Existing P2P File Sharing Architectures 23

2.3.3 P2P media streaming 25

2.3.3.1 Multicast based P2P Media Streaming Approaches 26

2.3.3.2 Application layer multicast and network layer Multicast 27

2.3.4 Related Works on P2P Media Streaming 28

2.3.4.1 Zigzag 28

2.3.4.2 NICE 31

2.3.4.3 P2cast 34

2.3.4.4 Narada 36

Chapter 3 Performance Study on Hybrid ARQ for Video Streaming 39

3.1 Introduction 39

3.2 Hybrid ARQ Techniques 42

3.2.1 Operations of the Hybrid ARQ Technique 42

3.2.2 Throughput Improvement on Hybrid ARQ 44

3.2.3 Scenarios in Hybrid ARQ 45

3.3 Performance Study of Hybrid ARQ for Video Streaming 49

3.3.1 Designs of Hybrid ARQ Video Streaming System 49

3.3.2 Simulations 50

3.3.2.1 Frame Loss Rate Simulation 50

3.3.3 Performance Test on an Implemented Hybrid ARQ Video Streaming System 56

3.4 An Adaptive Hybrid ARQ Scheme 59

3.4.1 Bandwidth Vs Frame Loss rate 59

3.4.2 Adaptive Hybrid ARQ 61

3.5 Chapter Conclusion 64

Chapter 4 A New Peer-to-Peer Architecture for On-Demand Video Streaming 66

4.1 Introduction 66

4.2 A New P2P On-Demand Video Streaming Scheme 69

4.2.1 P2P forwarding Chain 69

4.2.2 Standby Server and FEC Stream 70

4.2.3 Hybrid P2P architecture 72

4.2.4 Client Departure 73

4.3 Parent-Child Exchange Routing Mechanism 74

4.4 Simulations 78

4.4.1 Simulation methodology 78

4.4.2 Simulation Results 79

4.4.2.1 Standby Server Performance 79

4.4.2.2 PCX routing mechanism performance 81

4.4.2.3. System Performance with Variable B_{P2P} and B_{S2P} 82

4.4.2.4 Server FEC Sending Threshold 84

4.4.2.5 PCX Shift 86

4.4.3 QoS Mechanisms for the Proposed System 87

4.4.3.1 QoS Mechanisms 87

4.4.3.2 Charging for Commercial VoD system 88

4.5 Chapter Conclusions 90

Chapter 5 A Parent Search Method for Peer-to-Peer Media Streaming Network 91

5.1 Introduction 91

5.2 Prim-Dijkstra algorithm in P2P Network 93

5.3 Fast Parent Search in P2P network 96

5.3.1 Network Triangle Inequality 97

5.3.2 Network Triangle Inequality in P2P Network 98

5.4 Simulations 101

5.4.1 Simulation Methodology 101

5.4.2 Simulation Results 102

5.4.2.1. Prim-Dijkstra algorithm 102

5.4.2.2 Fast Parent Searching 104

5.4.2.3 Fast Searching in Peer Contact List 105

5.5 Chapter Conclusion 107

Chapter 6 Conclusions and Future Work 108

6.1 Conclusions 108

6.2 Suggestion for Future Work 110

References 113

List of Figures

Figure 2-1: Layered architecture for transporting multimedia data 12

Figure 2-2: Stop-and-wait ARQ protocols 15

Figure 2-3: Lost-information frame recovery using SR-ARQ 16

Figure 2-4: Parity data in data link layer 18

Figure 2-5: Progressive bitstream from the source coder partitioned into N layers or quality levels 19

Figure 2-6: N-description generalized MD codes using forward error correction codes 21

Figure 2-7: Simplified, High-Level View of Peer-to-Peer versus Centralized (Client/Server) Approach. 22

Figure 2-8: Network bandwidth saving by P2P 25

Figure 2-9: Network-layer and application layer multicast 27

Figure 2-10: Administrative organization of peers 29

Figure 2-11: The multicast tree atop the administrative organization 30

Figure 2-12: Hierarchical arrangement of hosts in NICE. 31

Figure 2-13: Control and data delivery paths for a two-layer hierarchy 33

Figure 2-14: Data forwarding operation at a host, *h*, that itself received the data from host *p* 34

Figure 2-15: A snapshot of P2Cast at time 40 36

Figure 2-16: Examples to illustrate the mesh-based approach in Narada 37

Figure 3-1: FEC, ARQ and Hybrid ARQ scheme 42

Figure 3-2: Hybrid ARQ with SR-ARQ 45

Figure 3-3: No Transmission Error 46

Figure 3-4: Sender to Receiver Transmission Error (Recoverable) 47

Figure 3-5: Sender to Receiver Transmission Error (Unrecoverable) 47

Figure 3-6: ACK Transmission Error 48

Figure 3-7: Hybrid ARQ simulation system diagram 49

Figure 3-8: Simulation results for $N=M$, (N=3~10, 10%~30% packet loss) 51
Figure 3-9: Simulation results for different M/N ratio (N=1~5) 52
Figure 3-10: Two-state Markov Chain Model 53
Figure 3-11: Markov Chain Loss model Simulation results ($M=N$, N=3~8) 55
Figure 3-12: Markov Chain Loss model Simulation results (N=5, M=1~6) 55
Figure 3-13: I frame and P frame 59
Figure 3-14: Adaptive Hybrid ARQ Scheme 63
Figure 4-1: The proposed chain structure P2P media streaming system 69
Figure 4-2: Hybrid P2P media streaming architecture 73
Figure 4-3: Client Departure 74
Figure 4-4: PCX routing mechanism 76
Figure 4-5: Client discontinuous playback time 79
Figure 4-6: Standby server bandwidth burden 81
Figure 4-7: Standby server bandwidth burden 82
Figure 4-8: Server bandwidth burden under different peer-to-peer bandwidth 83
Figure 4-9: system performance under different server-to-peer bandwidth 84
Figure 4-10: Client discontinuity with different server FEC sending threshold H 85
Figure 4-11: Server burden with different server FEC sending threshold H 86
Figure 5-1: Network spanning tree with different α 95
Figure 5-2: Network Triangle 97
Figure 5-3: P2P Network Triangle 98
Figure 5-4: Results on average hop and path delay 104
Figure 5-5: Optimal Network Spanning Tree (α = 0.5, connection =8) 104
Figure 5-6: Searching cost for parent search of a newly arrived peer 105
Figure 5-7: Searching cost for different searching orders 106
Figure 6-1: VCR function for P2P Video Streaming 111
Figure 6-2: Nearby peers cluster together and are taken as one node 112

List of Tables

Table 3-1: Simulation results for N=5 (20% packet loss) 52

Table 3-2: Simulation results for N=M, (20%packet loss) 57

Table 3-3: Simulation results for N=5, M=1~6 (20% packet loss) 58

Table 3-4: Frame loss rate and server bandwidth consumption 60

Table 3-5: Bandwidth underutilization 62

Table 3-6: Adaptive Hybrid ARQ scheme utilizes server bandwidth 64

Table 4-1: PCX shift with different B_{S2P} (B_{P2P} = 1) 86

Table 4-2 PCX shift with different B_{P2P} (B_{S2P} = 1) 87

Table 4-3 QoS and Different B_{S2P} 88

Table 5-1: Parent location results 103

Table 5-2: Parent location results (with a limit of 8 for the direct server-peer connection) 103

List of Abbreviations

ACK	:	Acknowledgement
ARQ	:	Automatic Repeat Request
CBR	:	Constant Bit Rate
CPU	:	Central Processing Unit
CVG	:	Complete Virtual Graph
DNS	:	Domain Name System
FEC	:	Forward Error Correction
GOF	:	Group of Frame
ISP	:	Internet Service Provider
MD-FEC	:	Multiple Description using Forward Error Correction
MST	:	Minimum Spanning Tree
P2P	;	Peer to Peer
PCX	:	Parent Child Exchange
QoS	:	Quality of Service
RS	:	Reed-Solomon
RTCP	:	Real-time transport control protocol
RTP	:	Real-time Transport Protocol
S2PPD	:	Server-to-Peer Path Delay
SPT	:	Shortest Path Tree
SR-ARQ	:	Selective Reject Automatic Repeat Request

SW-ARQ	:	Stop-and-Wait Automatic Repeat Request
TCP	:	Transmission control protocol
UDP	:	User Datagram Protocol
URL	:	Uniform Resource Locator
VCR	:	Video Cassette Recorder
VLC	:	Variable Length Coding
VoD	:	Video on Demand

List of Publications on which this Research is Based

1. Lei. Zhang and K. T. Lo, "A peer-to-peer architecture for on-demand video streaming on Internet", *Proceedings International Conference on Communications, Circuits and Systems (ICCCAS'04)*, Chengdu, China, July, 2004.
2. Lei. Zhang and K. T. Lo, "A Parent Search Method in Peer-to-Peer Media Streaming Networks", *Proceedings of 2004 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'04)*, pp 897-900, Tainan, Taiwan, 2004.

Chapter 1

Introduction

1.1 Introduction

Before the advent of streaming media [WU2001, CONK2001], people had to wait for a media file to be completely downloaded from the Internet or a network server in order to experience it. Using streaming media, users can view the media almost instantly. With streaming, media data are first divided into packets and then sent over a network connection according to their timing order. The user receives the information packets and plays simultaneously the media data piece by piece. The process is almost invisible to the user, except for a small delay at the beginning due to buffering or pre-fetching.

In general, multimedia streaming applications can be classified into two categories based on their general requirements [LU2000]: broadcasting services and on-demand applications. For broadcasting, a multimedia stream is first captured and compressed in real-time at encoder and then transmitted through the network to the clients. For on-demand applications, the multimedia streams are pre-compressed and archived on the storage system of the streaming server, and then transmitted to clients at any time upon clients' request. Live broadcasting

radio and video-on-demand (VoD) are the most well-known examples for these two modes of operation respectively.

Streaming media occupies quite a large share of today's Internet applications. It is also believed that the demand for streaming applications will rapidly increase in the coming few years. Compared with conventional data communications, transmission of media data has more stringent requirements on network bandwidth, packet delay and loss. However, the current Internet provides only the best effort services, which have no provision on quality of service (QoS) guarantee for a data connection such as providing minimum throughput or bounded delay. For multimedia applications, an unknown delay time is generally unacceptable. As there is inherent timing relationship among media data packets, any alternation on the timing sequence of original data will affect the playback quality. Therefore, lately arrived data will be useless for the playback and be considered as loss. Besides, the lately arrived data will probably affect the decoding of other data because of the error propagation problem in compressed media data. As a result, designing protocols and mechanisms for multimedia streaming pose many challenges.

Most of the current media streaming solutions are using the client/server model, in which a dedicated media server is used to serve a number of clients. In this architecture, the number of users that can be supported by the system is limited

by the resources of the server such as network bandwidth, storage and CPU power. To increase the system capacity, the only way is to use multiple servers and increase their resources. Obviously, this approach is not scalable and cost-effective. One of the possible solutions to increase the scalability and reduce the system cost is to provide streaming services in a distributed network environment. Most recently, the peer-to-peer (P2P) architecture has gained much attention for media streaming. In such architecture, a peer (or a client) stores the streamed media data after receiving it, and then shares the cached content to other requesting peers. As it makes use of the resources of client machines to provide services, the burden on the media server is greatly reduced. The P2P approach in file sharing/downloading was very successful in Internet world. Typical examples include Napster [NAPS], BitTorrent [BT], eDonkey [ED], Freenet [FREE], Gnutella [GNUT], and Kazaa [KAZA]. However, the work on peer-to-peer media streaming [HUA1998, SEN1999, TRAN2004, BANE2002] is on-going and there are still many problems to be solved before it can be widely deployed in the Internet.

One problem of P2P media streaming is the network construction and maintenance for P2P data forwarding/broadcasting. When a newly arrived peer enters the P2P network, it probably cannot get media stream directly from the media server/seed. In this regard, the new peer has to locate its parent before receiving the data from the forwarding tree/chain. Some methods [XIAN2004,

BANE2002] have been proposed to solve this problem recently. Besides, when a peer departs from the P2P network, the child peers of the departing peer have to be rerouted. Hence, rerouting is another important issue for providing streaming services on a P2P network.

1.2 Motivations and Research Objectives

As mentioned, designing protocols and mechanisms for multimedia streaming over the existing Internet is quite challenging. The work on Internet multimedia streaming has been actively investigated by different people in recent few years and many different approaches have been proposed. However, there are still many open issues to be solved especially for the peer-to-peer streaming model. Hence, it motivates our work to study how to support video streaming using the P2P network model. In this thesis, we aim at solving the major problems in providing efficient and reliable data transfer for both on-demand and live video streaming services in a P2P network.

1.3 Thesis Organization

Following the introduction of the thesis, a literature survey and some technical reviews on multimedia streaming are given in chapter 2. There are three major parts in this chapter: background on media streaming, client/server based video

streaming and peer-to-peer (P2P) video streaming.

In Chapter 3, we focus the work on performance study of hybrid ARQ in a video streaming system. In this chapter, the detailed operation of the hybrid ARQ error protection mechanism and the simulation study are presented. An adaptive hybrid ARQ scheme which can maximize the bandwidth utilization is also developed in this chapter.

Deploying the idea of Hybrid ARQ, we introduce a new P2P architecture for on-demand video streaming in Chapter 4. A parent-child exchange (PCX) routing mechanism is also developed to further improve the performance of the system in this chapter.

Considering live video streaming, a tree structured P2P streaming system is more appropriate than the chain structured approach described in Chapter 4. In Chapter 5, a new fast parent searching method is derived to help newly arrived peers to locate their parent in a tree-structured P2P network efficiently. Computer simulations are performed in this chapter to evaluate the performance of the proposed method.

Finally, Chapter 6 presents an overall conclusion for this dissertation and gives some suggestions for future research.

Chapter 2

Background on Video Streaming and Peer-to-Peer (P2P) Networks

Media streaming is one of the fast emerging applications on the Internet in recent few years. Many different streaming architectures, such as Microsoft's Windows Media, Real Network's RealPlayer and Apple's QuickTime Player, have been introduced to transmit media data over the Internet. In this chapter, an overview and literature survey for different technological aspects in media streaming and peer-to-peer (P2P) networks are presented.

2.1 Issues on Internet Media Streaming

Media streaming is an enabling technology for providing real-time multimedia data delivery between (or among) end users over the Internet. Before the advent of media streaming, people had to wait for a media file to be completely downloaded from the Internet or a network server in order to experience it. Using streaming media, users can view the media almost instantly. With the streaming approach, the media data are divided into small packets which are sent over a network connection. The user receives the data packets and plays simultaneously the media piece by piece. The process is almost invisible to the

user, except for a small delay at the beginning due to buffering or pre-fetching.

Streaming media occupies a large share of today's Internet applications. However, there are still many technical challenges that make multimedia streaming over the Internet difficult. Transmission of multimedia data has stringent bandwidth, delay and loss requirement. However, the current Internet is inherently a best effort network and was not designed to handle isochronous (continuous-time based) traffic such as real-time audio and video streaming. The current Internet provides only the best effort services, which have no guarantee on the quality of service (QoS) such as providing a minimum throughput or a maximum delay. For multimedia applications, an unknown delay time is generally unacceptable. As there is inherent timing relationship among media data packets, lately arrived data is completely useless and considered as loss. Besides, the lately arrived data will probably affect the decoding of other data because of the error propagation problem in compressed media data. Hence, designing protocols and mechanisms for multimedia streaming poses many challenges. In this section, we will give an overview on several major technical issues of media streaming.

2.1.1 Bandwidth limitations

Bandwidth is the portion of the network that is available to an application to transfer information on the network. The level of reliability and playback quality that is acceptable among users has not yet been reached because of the

bandwidth limitations. Multimedia data involves a huge data volume traffic imposing the use of a large bandwidth on the network. That probably will result in packet loss when network congestion occurs. In audio-video communications, packet loss shows up in the form of gaps or periods of silence in the conversation, thus leading to a "clipped speech" effect that is unsatisfactory for most users and unacceptable in communications. Similarly, a non-smooth playback will result in video application due to packet loss. So, the multimedia data need to be compressed before transmission to reduce its data rate and hence the loading on the network.

2.1.2 Jitter

During real-time data transmission, the receiver would like to have regular arrival of packets so that the data can be played back in its particular time. However, packets that are sent by the server at a regular interval tend to arrive at an irregular rate due to the variation of delays, which is caused by different congestion levels at the routers along the path. This phenomenon is called jitter, which may result in a non-smooth playback at the receiver. The jitter problem is usually resolved by using a playback buffer at the receiver and by assigning timing information to each packet (normally carried by RTP header). Using such a buffer, the receiver stores the first several packets into the buffer before starting the playback. As new packets arrive, they are queued into the buffer and then played back according to the scheduling of the buffer. As long as the maximum

jitter is not greater than the capacity of the buffer, a smooth playback can be achieved. With this approach, some initial delay may be observed by the user. One should note that this playback buffer cannot be arbitrarily large as increasingly large playback buffer will introduce a long playback delay that may annoy the user.

2.1.3 Loss

The Internet provides a guarantee that packets will be delivered at all, much less in order. Packets will be dropped under peak loads and during periods of network congestion. But due to the time sensitivity of real-time transmissions, packet loss results in a non-smooth playback that may annoy the user. Error resilience process has to be incorporated to overcome the loss problem for transmission of video over the Internet.

2.1.4 TCP and UDP

Transmission control protocol (TCP) and user datagram protocol (UDP) [BACC1997] are two low-level transport layer protocols used in the current Internet. TCP provides a completely reliable (no data duplication or loss), connection-oriented, full-duplex transport service that allows two application programs to setup/terminate a connection, and send data in either direction. Each TCP connection is started reliably and terminated gracefully, with all data being delivered before the termination occurs. TCP is used mainly in the applications

requiring reliable delivery but no strict delay time limit, such as file transferring, web browsing, E-Mail delivery, and remote login. However, TCP is not suitable for real-time media streaming since the delay time in a TCP application is hard to control and the lately arrived data is useless at all.

UDP is a transport layer protocol that provides connectionless services. Since UDP only provides best effort services, packets may be lost before they reach the destination. Hence, UDP is useful only when TCP would be too complex, too slow, or just unnecessary. The main applications of UDP include domain name system (DNS), multimedia services and network multicasting. Those applications are loss tolerant but delay sensitive.

2.1.5 RTP and RTCP

RTP and RTCP [RFC 791, RFC1889] are protocols designed for real-time data transmission. Real-time transport protocol (RTP) provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio and video, over multicast or unicast channels. Real-time transport control protocol (RTCP) allows monitoring of the data delivery in a manner scalable to large multicast networks, and providing minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layer protocols. However, in most applications of RTP and RTCP, they are used under the UDP environment.

2.2 Video Streaming Using Client/Server Model

2.2.1 Architecture for Internet Video Streaming

In the previous section, we have stated the challenges on multimedia streaming over the current Internet. To address the above technical issues, two general approaches have been proposed [WU2000]. The first approach is network-centric, in which the routers/switches in the network are required to provide QoS support. For example, the integrated services model (IntServ) [RFC1633] and differentiated services model (DiffServ) [RFC2475] proposed by IETF. The second approach is solely an end system-based and does not impose any requirements on the network for QoS support. Extensive research work based on the second approach has been conducted and various solutions have been proposed. These solutions can be presented through two perspectives: transport as well as compression [WU2000]. By transport perspective, the use of control/processing techniques without regard of the specific audiovisual semantics is employed to control the amount of data traveling over the network. By compression perspective, the signal processing techniques with consideration of the audiovisual semantics on the compression layer are used to meet the rate requirement that the transport layer prefers. These two perspectives are closely operated together in order to provide a framework to meet the QoS requirements for the applications.

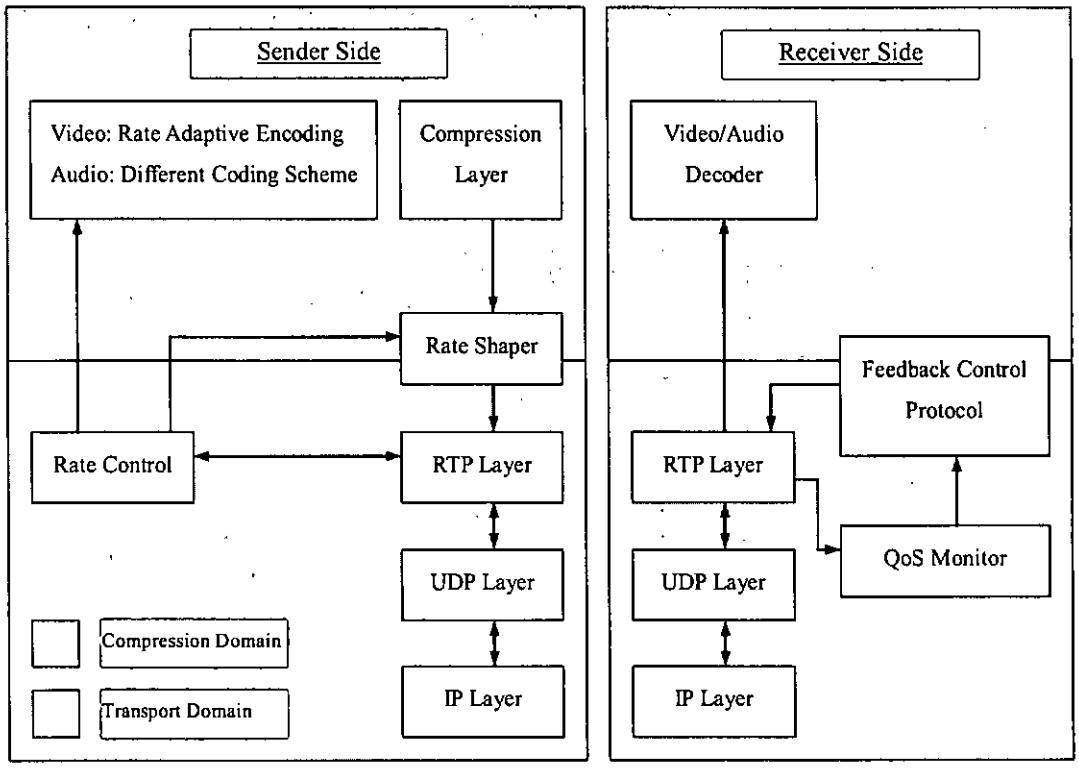


Figure 2-1: Layered architecture for transporting multimedia data

Figure 2-1 presents the architecture for transporting real-time multimedia data. The architecture includes two domains, the compression domain and transport domain, which are used to meet their relative perspective stated above. At the sender side, the compression domain consists of two components, compression layer and rate adaptive encoding. Compression layer compresses the raw audiovisual data passed from application to the rate estimated for the network. This rate can be achieved by applying a rate-adaptive encoding algorithm for audiovisual data to adapt the rate constraint in the transport domain. As the network conditions vary over time, the compressed data may exceed the current allowable rate. So, the compressed data will first be filtered by a rate shaper. Then, it is framed with media information, such as timestamp, at the transport

layer and submitted to the network. At the receiver side, the arriving packet is decapsulated into media information and compressed data. The former is utilized by the QoS Monitor to infer the network conditions based on the behavior of the arriving packet. The compressed data is decoded and passed to the application for playback. To notify the network conditions to the sender, the feedback control protocol obtains the information provided by the QoS monitor and sends it back to the sender. The rate control module at the sender utilizes this feedback information to estimate the available network bandwidth and conveys it to the rate-adaptive encoder or the rate shaper such that the output rate can be regulated to adapt the network conditions.

2.2.2 TCP-Friendly Streaming Protocols

We consider a flow is TCP-friendly "if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances" [FLOY1999]. Thus, a TCP-friendly flow should have a sending rate that is close to TCP throughput. In any case, the sending rate has to be in the same order of magnitude as the corresponding TCP throughput. The test of whether or not a flow is TCP-friendly assumes that TCP can be characterized by a congestion response of reducing its congestion window at least by half upon indications of congestion, and of increasing its congestion windows by a constant rate of at most one packet per round-trip time otherwise. The benefit that a flow is TCP-friendly is that it can fairly share the network resources with other TCP flows since it has a similar

behavior with TCP flows; hence it will not cause congestion collapse in the network.

Previous works are normally based on the rate-based algorithms for providing TCP-friendly congestion control on the top of UDP. The transmission rate is regulated attempting to achieve the same throughput as a TCP flow operating under the same conditions such that it can compete fairly the network resources with other TCP flows. These approaches are based on modeling the characteristics of TCP congestion control [PADH1998, MAHD1997, FLOY2000, JACO1996]. The model assumes that TCP can be characterized by a congestion response of reducing its congestion window at least by half upon indications of congestion and of increasing its congestion window by a constant rate at most one packet per round-trip time otherwise [FLOY1999]. The application periodically measures the values of packet loss rates and round-trip times applying for these models to estimate the throughput of TCP on the same conditions. Such applications typically are implemented with RTP/RTCP for obtaining these parameters. However, there are still several potential drawbacks for such approaches [RAME1999] that can result in under or over-allocation of bandwidth to non-TCP flows caused by estimating the packet loss rate inaccurately. Also, these approaches only achieve the congestion behavior of TCP but not the reliability. The lost packets cannot be recovered and need other mechanisms such as error resilience for handling this problem. Therefore, the

complexity of the protocol will be increased.

2.2.3 Error Concealment

2.2.3.1 Automatic Repeat Request (ARQ)

Automatic Repeat Request (ARQ) [REED1999] is a technique commonly used by many communication systems to ensure a reliable information delivery. With ARQ, the sender will receive an acknowledgement from the receiver to determine whether the transmitted data packets are correctly received or not. If the sender does not get the acknowledgement from the receiver, it is assumed that the data are lost and normally a packet retransmission is required.

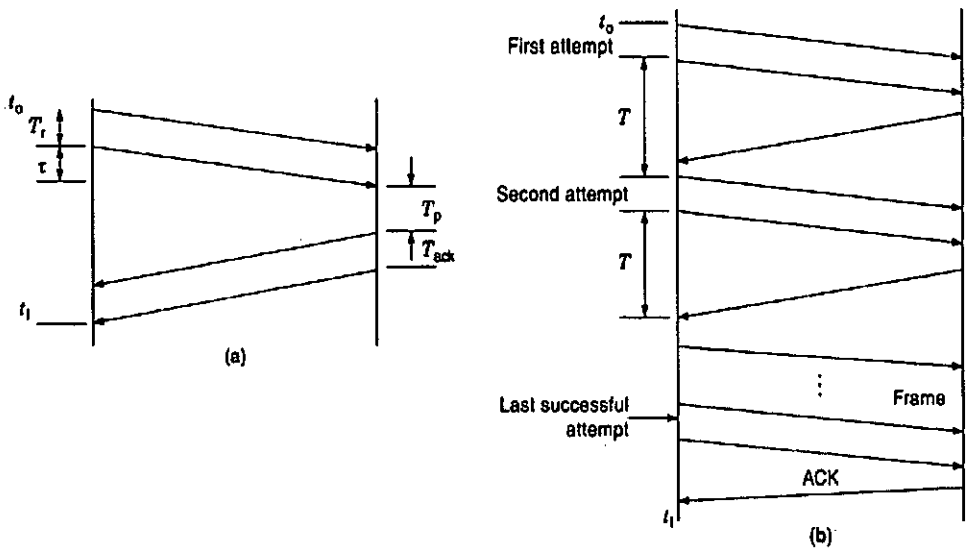


Figure 2-2: Stop-and-wait ARQ protocols: (a) no errors, (b) with errors. (Data transmitted by sender and ACK by receiver. Retransmission takes place if ACK is not received by timeout.)

The Stop-and-Wait (SW) ARQ (Figure 2-2) is one of the most commonly used

ARQ techniques. Under ideal conditions (no channel errors, no network congestion, etc.), the SW ARQ's efficiency or maximum throughput is given by $\eta = \frac{T_f}{T_f + 2\tau + T_p + T_{ack}}$, where T_f is the data frame duration, τ is the propagation time in each direction, T_{ack} is the acknowledgement frame duration, and T_p is processing time. The overhead or delay time is $T_p + T_{ack} + 2\tau$.

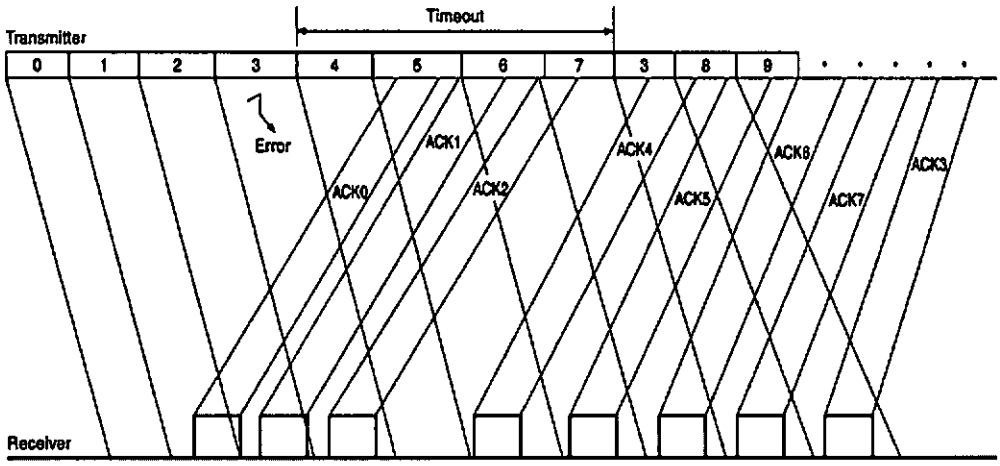


Figure 2-3: Lost-information frame recovery using SR-ARQ, where Frame 3 is received in error and no ACK is transmitted to sender, so sender after timeout retransmits frame 3, after having transmitted frame 7.

Another type of ARQ is Selective Reject (SR) ARQ. As illustrated in Figure 2-3, the information frames in SR ARQ are continuously sent by the transmitter and acknowledged by the receiver, without waiting for acknowledgements from the receiver. If the transmitter gets the ACK for a packet before the deadline of ARQ waiting time, the packet is claimed to be successfully received by the receiver. Otherwise, a packet will be sent out to replace the lost packet. Only delay is increased in the case of packet loss. Such an ARQ scheme is used in the

performance study of the Hybrid ARQ system in Chapter 3.

2.2.3.2 Forward Error Correction

Forward error correction (FEC) [BURR2001] has been a technique used in communication systems to perform error protection and recovery for a long time. The main idea of FEC is to add redundancy to data sent over an unreliable channel. The redundancy is used to correct the error at the receiver without any direct feedback to the sender. The redundant data is normally referred as parity data. Using this technique, there's no need to request a retransmission of missing data since the already delivered parity data should be adequate to correct the errors. The creation of parity data is accomplished by using a channel coder to derive the redundancy from the original data. When using FEC, there is always a trade-off between reliability and performance. FEC can dramatically improve the reliability of data delivery, but it requires extra bandwidth for the necessary parity data in addition to the processing overheads for sending and receiving coded data.

Traditional applications of FEC have been implemented at the link layer as illustrated in Figure 2-4. Redundant parity bits are included with the data bits and any bits in error are corrected by a calculation between the original data and additional parity bits.

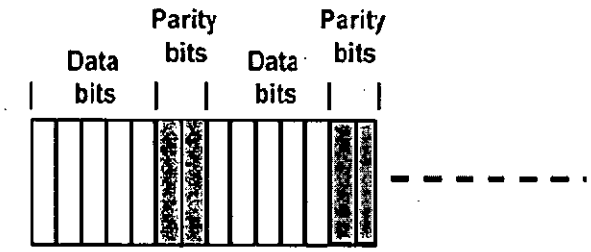


Figure 2-4: Parity data in data link layer

2.2.3.3 Reed Solomon FEC coding

One popular FEC example is Reed-Solomon code [RILE], which has been implemented on many storage devices and communication protocols. A standard RS coder has three parameters as listed below:

MM - the code symbol size in bits

KK - the number of data symbols per block, KK < NN

NN - the block size in symbols, which is always (2^MM - 1)

Each RS "symbol" is actually a group of MM bits. In the literature, RS code parameters are normally given in the form "RS(255,223) ". The first number inside the parentheses is the block length NN, and the second number is KK. In case of NN = 255, MM is 8 indicating an 8-bit symbol, one byte.

The error-correcting ability of a Reed-Solomon code depends on NN-KK, the number of parity symbols in the block. Without known erasures, the decoder can correct up to (NN-KK)/2 symbol errors per block and no more. If all the error

locations are known in advance, the decoder can correct as many as NN-KK errors, the number of parity symbols in the code block.

2.2.3.4 Multiple Description using Forward Error Correction (MD-FEC)

Multiple description using forward error correction (MD-FEC) [PURI2001] is a transcoding mechanism to convert a prioritized multi-resolution bitstream into a non-prioritized multiple description bitstream by an effective FEC coding. Each description in the MD stream occupies an entire network packet, thus the terms "description" and "packet" are used interchangeably.

The quality profile reflects the target quality (or equivalently distortion *d*: lower distortion implies higher quality and vice versa) when any *k* out of N descriptions are received. We will use the notation ***d(k)*** to describe the quality profile where the ith entry in ***d(k)*** represents the target quality when *i* descriptions are received.

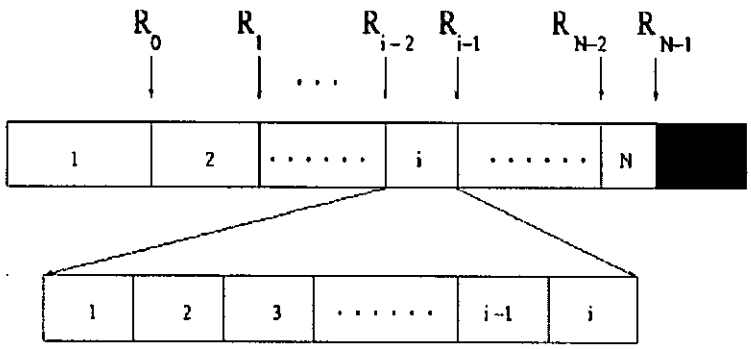


Figure 2-5: Progressive bitstream from the source coder partitioned into N layers or quality levels. [PURI2001]

Given N and ***d(k)*** and a progressive bitstream, the stream is marked at N

positions (see Figure 2-5) which correspond to the attainment of the distortion levels ***d(k)*** and is thus partitioned into **N** sections or resolution layers. The goal is to enable the ith layer to be decodable when i or more descriptions arrive at the receiver (i.e., when the number of erasures does not exceed **N** – i.). This can be attained using the Reed-Solomon family of erasure-correction block codes (An (n,k,d) block code is defined by a length n code with k user symbols and a minimum distance of d, i.e. it can correct $(d-1)$ erasures. Reed Solomon block codes have the property of maximum distance $(d=n-k+1)$ i.e. the whole data can be recovered from *any* k out of n symbols.), which are characterized by the optimal code parameters (**N**, i, **N**-i+1) and can correct any (**N**-i) erasures out of **N** descriptions. We split the ith layer into i equal parts, and apply the (**N**, i, **N**-i+1) Reed Solomon code to get the contribution from the ith layer or section to each of the **N** descriptions. The contributions from each of the **N** levels or sections are then concatenated to form the **N** descriptions (see Figure 2-6). This packetization strategy thus provides the property that the more the number of packets received, the better the received quality.

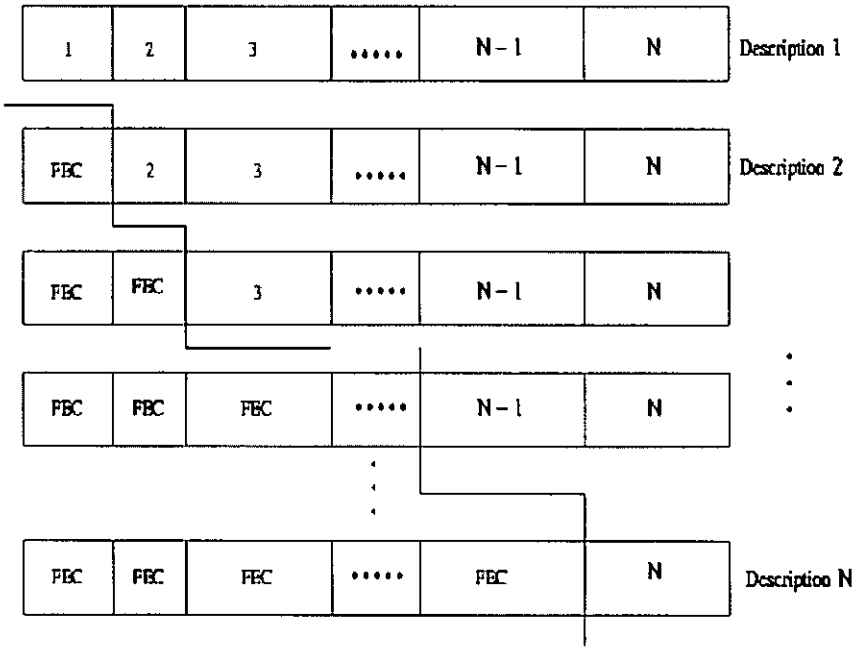


Figure 2-6: N-description generalized MD codes using forward error correction codes. [PURI2001]

2.3 Video Streaming Using Peer-to-Peer (P2P) Model

2.3.1 Introduction

Peer-to-peer, or abbreviated P2P, is referred to a type of network in which each workstation has equivalent capabilities and responsibilities. This differs from the client/server architectures, in which some computers are dedicated to serving the others. P2P uses existing desktop computing power, disk storage and networking connectivity to exchange information or data or do computing process. Figure 2-7 shows the difference between the two architectures.

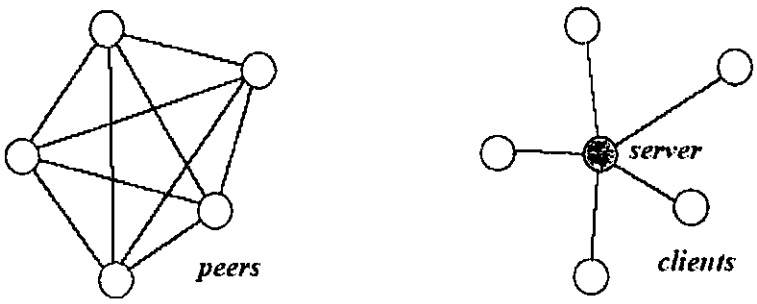


Figure 2-7: Simplified, High-Level View of Peer-to-Peer versus Centralized (Client/Server) Approach.

2.3.2 Peer-to-Peer Files Sharing

Content storage and exchange is one of the areas where the P2P technology has been most successful. Multimedia content, for instance, inherently requires large files. Napster [NAPS] and Gnutella [GNUT] have been used by Internet users to circumvent bandwidth limitations that make large file transfers unacceptable with classic mechanisms.

The key to cheap file distribution is to tap the unutilized upload capacity of computers in the networks. It's **free**. Their contribution grows at the same rate as their demand, creating limitless scalability for a fixed cost. The basic idea of P2P file sharing is that a shared file is chopped into pieces and the user simultaneously downloads these pieces separately from different peers who have one or more pieces. When all pieces of a shared file are got, the file can be recovered and ready for use.

One very famous P2P network system is Napster [NAPS]. It supported 20 million people sharing music online. Now eDonkey [ED] and Bit torrent [BT] are another two successful P2P file sharing architectures. By these tools, one can download the missing parts of a file from several sources and upload different parts of the same file to others at the same time. Data are exchanged in the P2P network. A peer downloads its required data and has to upload data other peers need.

Some mechanisms are often used to reward peers who are willing to upload more data to others. For example, in eDonkey, the user can adjust the upload speed limit to peers. By eDonkey2000(v 0.53) setting, the client's download speed limit is proportional to its upload speed limit. In normal situation, the download speed limit is fixed to 5 times of upload speed limit, and if the upload bandwidth limit is set over a threshold like 20KBps, the download speed is not limited any more.

2.3.2.1 Existing P2P File Sharing Architectures

Napster

Napster [NAPS] is the first P2P file sharing application that jump-started the P2P area. Napster was originally developed to defeat the copying problem and to enable the sharing of music files over the Internet. Napster uses the centralized directory model to maintain a list of music files, where the files are added and

removed as individual users connect and disconnect from the system. Users submit search requests based on keywords such as "title," "artist," etc. Although Napster's search mechanism is centralized, the file sharing mechanism is decentralized; the actual transfer of files is done directly between the peers. Napster's centralized directory model inevitably yields scalability limitations. For example, a user's available bandwidth can be tremendously reduced by users downloading songs from his/her machine. Yet, centralization simplifies the problem of obtaining a namespace and enables the realization of security mechanisms.

BitTorrent

BitTorrent [BT] is a new P2P file sharing protocol. BT works a little different with Napster and eDonkey. In Napster and eDonkey, directory servers are required to provide file publishing and searching function. A user of eDonkey select files to share his/her computer and the shared file list can be published automatically to the directory server he/she has connected to. The directory server maintains the list of millions of shared files and eDonkey users who have connected to an eDonkey directory server can search content they want by keywords or file format. The eDonkey publishes to servers that can be set up by anyone. Once the network reaches a certain size these servers become a bottleneck to the system performance. Users can no longer search the entire network for things they are interested in and the servers become more and more

bogged down. In BitTorrent, a file uploader (so called seeder) has to generate a static metainfo (.torrent) file for shared file(s), which include some necessary information (such as file size, date, seeder URL), and put the metainfo file to a web server. A file downloader can search in the web server and download the metainfo file. The BitTorrent tracker will automatically recognize the metainfo file and start download from the seeder and other peers.

2.3.3 P2P media streaming

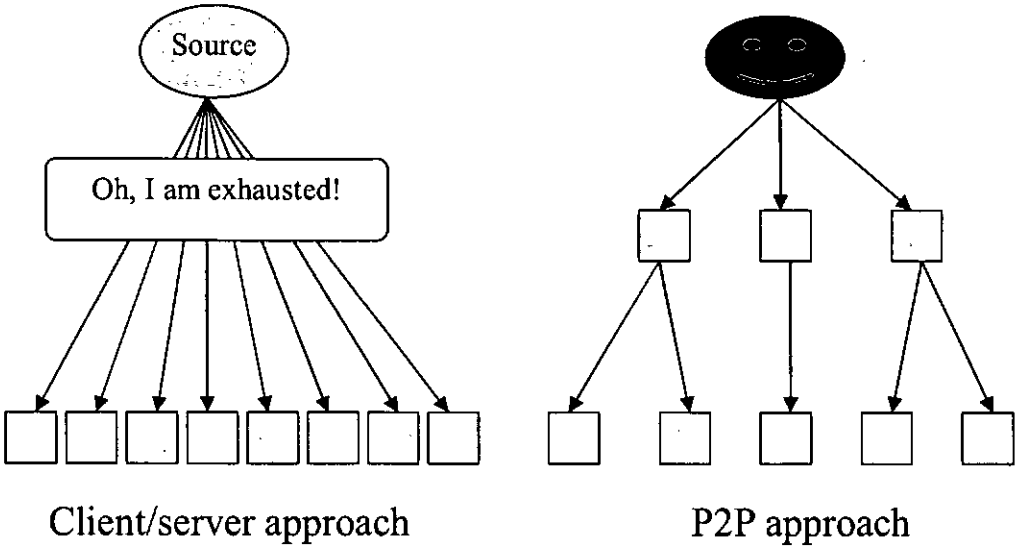


Figure 2-8: Network bandwidth saving by P2P

The biggest concern of P2P media steaming is bandwidth saving as shown in Figure 2-8. In the traditional client/server approach, the bandwidth required for such a VoD server is proportional to the number of clients in the system. It may be workable in a small scale, e.g. tens of clients. However, for a popular movie attracting hundreds or thousands of people, a normal VoD server can hardly

handle the huge network throughput at that way. P2P media streaming makes the bandwidth pressure on the server becomes much smaller than the traditional VoD server. Most clients get media streaming from other peers in the network instead of the server.

P2P media streaming is a little similar to P2P file sharing. The media information is exchanged by peers in a P2P network. Because of the property of media streaming, media viewer needs to view media content in a time sequence order. The media data has to be retrieved by peer in a time sequence order too. A late arrival media data packet is useless. In on-demand media streaming, media data far away from current playback time position is also valueless.

2.3.3.1 Multicast based P2P Media Streaming Approaches

Many existing P2P media streaming approaches are multicast based. The multicast approach achieves better resource utilization by serving multiple clients using the same stream. The basic idea is to establish a multicast session to which clients subscribe. This is done by creating multicast distribution trees. The multicast approach is more natural to live streaming in which clients are synchronized: all clients receive the same portions of the stream at the same time. To cope with the asynchronous nature of the on-demand service, several techniques have been proposed.

One of the key ideas in adapting multicast to on-demand service is patching and its variations [HUA1998, SEN1999]. A good comparison is given in [MAHA2003]. In patching (also known as tapping), a new client arriving within a threshold is allowed to join an on-going multicast session. In addition, the client establishes a unicast connection with the server to "patch" or get the missed part of the file. The two streams run at the full play rate. The patch stream terminates when the client gets the missed part. Patching techniques may require the client to tune into multiple streams during the patching period. This means that the client has to have an inbound bandwidth of at least double the streaming rate. This is quite a stringent requirement for the limited-capacity peers in the target environment.

2.3.3.2 Application layer multicast and network layer Multicast

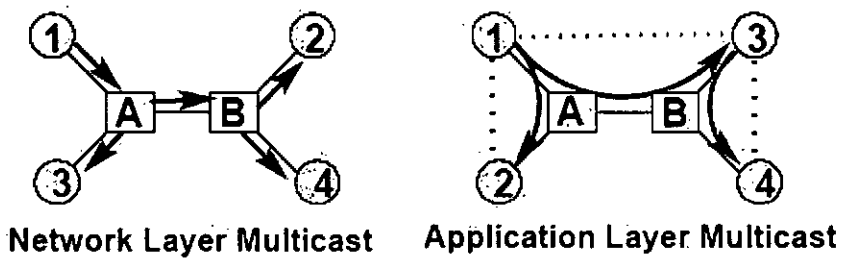


Figure 2-9: Network-layer and application layer multicast. Square nodes are routers, and circular nodes are end-hosts. The dotted lines represent peers on the overlay. [BANE2002]

The basic idea of application-layer multicast is shown above. Unlike network layer multicast where data packets are replicated at routers inside the network, in application-layer multicast data packets are replicated at end hosts. Logically, the

end-hosts form an overlay network, and the goal of application-layer multicast is to construct and maintain an efficient overlay for data transmission. Since application layer multicast protocols must send the identical packets over the same link, they are less efficient than network layer multicast.

2.3.4 Related Works on P2P Media Streaming

2.3.4.1 Zigzag

Zigzag [TRAN2004] is a tree-based P2P architecture where peers are clustered into a hierarchy called administrative organization for easy management. In Zigzag, the multicast tree has a height logarithmic with the number of clients, and a node degree bounded by a constant. This helps reduce the number of processing hops on the delivery path to a client while avoiding network bottleneck. Consequently, the end-to-end delay is kept small. Zigzag consists of two important entities: administrative organization and multicast tree.

A. Administrative Organization

An administrative organization is used to manage the peers currently in the system as illustrated in Figure 2-10. Peers are organized in a multi-layer hierarchy of clusters recursively defined as follows (where H is the number of layers, $k > 3$ is a constant):

(1) Layer 0 contains all peers.

(2) Peers in layer $j < H - 1$ are partitioned into clusters of sizes in $[k, 3k]$. Layer H-1 has only one cluster of size in $[2, 3k]$.

(3) A peer in a cluster at layer j is selected to be the **head** of that cluster. This head automatically becomes a member of layer $j + 1$ if $j < H - 1$. The server S is the head of any cluster it belongs to.

(4) A non-head peer in a cluster at layer j is selected to be the **associate-head** of that cluster. An exception holds for the highest layer where the server is both the head and the associate-head.

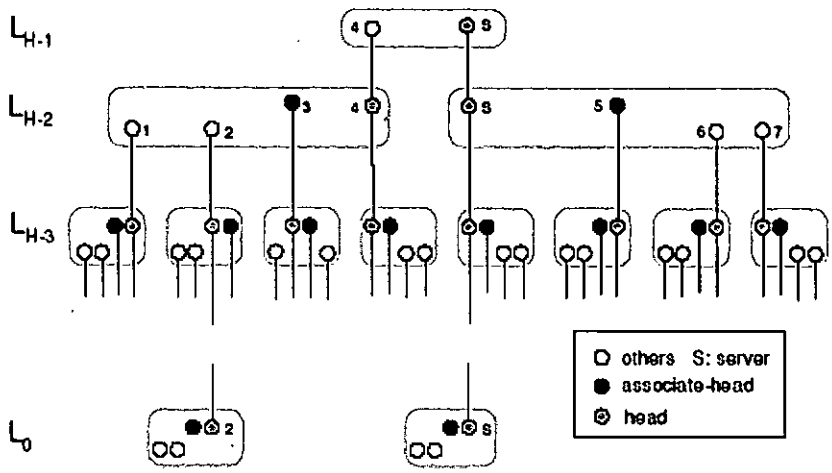


Figure 2-10: Administrative organization of peers [TRAN2004]

As an example, in Figure 2-10, at the highest layer (i.e., layer H-1), the server S is both the head and the associate-head. Peer 3 is the associate-head of a cluster at layer H- 2, and the head of a cluster at layer H- 3. Peer 4 is the head of a cluster at layer H-2, thus also belonging to layer H-1.

B. Multicast Tree

The multicast tree is built based on the administrative organization. The rules to which the multicast tree must be confined are called C-rules and are defined below (demonstrated by Figure 2-11):

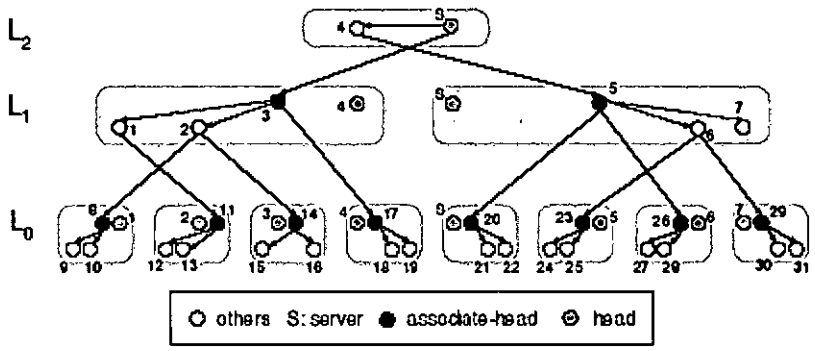


Figure 2-11: The multicast tree atop the administrative organization (H= 3, k= 4) [TRAN2004]

Definition 1: [C-Rules]

- Rule 1: A peer, when not at its highest layer, neither has a link out nor a link in. E.g., peer 4 at layer 1 and layer 0 has neither outgoing link nor incoming link because its highest layer is layer 2.
- Rule 2: Non-head members of a cluster must receive the content directly from its associate-head. In other words, this associate-head links to every other non-head member of the cluster. E.g., in a cluster at layer 1, associate-head 3 links to non-head members 1 and 2; in a cluster at layer 0, associate-head 8 links to non-head members 9 and 10.
- Rule 3: The associate-head of a cluster, except for the server, must get the content directly from a foreign head. E.g., the associate-head 8 of a cluster at

layer 0 has a link from peer 2 who is a foreign head of 8; the associate-head 5 of a cluster at layer 1 has a link from its foreign head 4.

2.3.4.2 NICE

The NICE protocol [BANE2002] (http://www.cs.umd.edu/projects/nice) is a scalable application-layer multicast protocol specifically designed for low-bandwidth data streaming application with large receiver sets. The scheme is based upon a hierarchical clustering of the application-layer multicast peers and can support a number of different data delivery tree with desirable properties.

A. Hierarchical Arrangement of Members

The NICE protocol arranges the set of end hosts into a hierarchy; the basic operation of the protocol is to create and maintain the hierarchy.

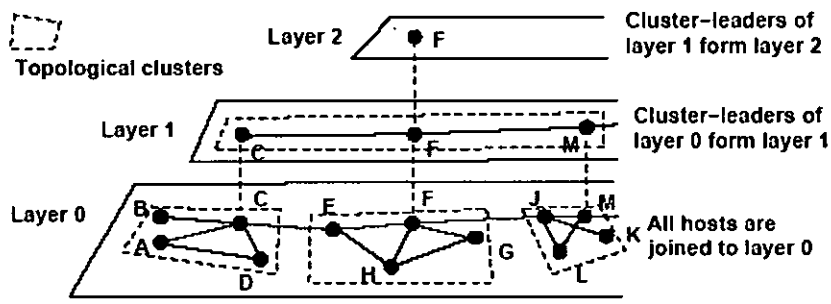


Figure 2-12: Hierarchical arrangement of hosts in NICE. The layers are logical entities overlaid on the same underlying physical network. [BANE2002]

The NICE hierarchy is created by assigning members to different levels (or layers) as illustrated in Figure 2-12. Layers are numbered sequentially with the

lowest layer of the hierarchy being layer zero (denoted by L_0). Hosts in each layer are partitioned into a set of clusters. Each cluster is of size between k and $3k-1$, where is a constant, and consists of a set of hosts that are close to each other. Further, each cluster has a cluster leader. The protocol chooses the (graph-theoretic) center of the cluster to be its leader, i.e. the cluster leader has the minimum maximum distance to all other hosts in the cluster. This choice of the cluster leader is important in guaranteeing that a new joining member is quickly able to find its appropriate position in the hierarchy using a very small number of queries to other members.

Hosts are mapped to layers using the following scheme: All hosts are part of the lowest layer, L_0. The clustering protocol at L_0 partitions these hosts into a set of clusters. The cluster leaders of all the clusters in layer L_i join layer L_{i+1}. This is shown with an example in Figure 2-13, using $k=3$. The layer clusters are [ABCD], [EFGH] and [JKLM]. In this example, C, F and M are the centers of their respective clusters of their L_0 clusters, and are chosen to be the leaders. They form layer L_1 and are clustered to create the single cluster, [CFM], in layer L_1. F is the center of this cluster, and hence its leader. Therefore F belongs to layer L_2 as well.

B. Control and Data Path

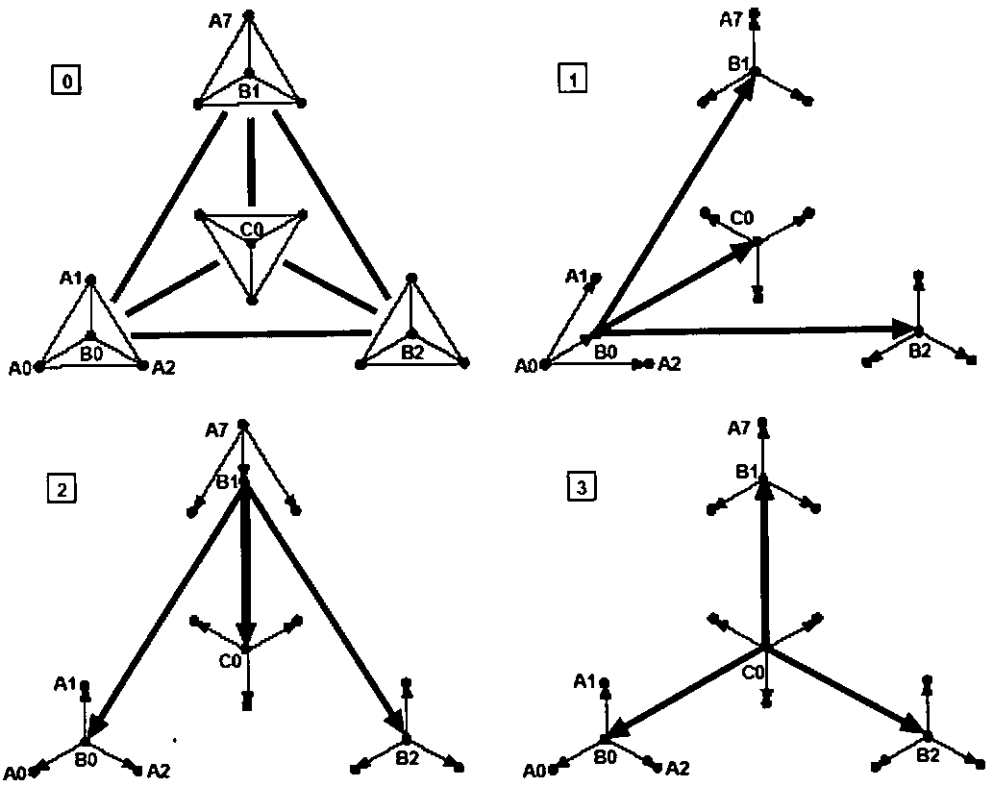


Figure 2-13: Control and data delivery paths for a two-layer hierarchy. All A_i hosts are members of only L_0 clusters. All hosts B_i are members of both layers L_0 and L_1. The only C host is the leader of the L_1 cluster comprising of itself and all the B hosts. [BANE2002]

The control topology for the NICE protocol is illustrated in Figure 2-13, Panel 0. The edges in the figure indicate the peering between group members on the overlay topology. $Cl_j(X)$ denotes the cluster in layer L_j to which member X belongs. It is defined if and only if X belongs to layer L_j. Consider a member, X, that belongs only to layers L_0, ... , L_i. Its peers on the control topology are the other members of the clusters to which X belongs in each of these layers, i.e. members of clusters $Cl_0(X)$, ... , $Cl_i(X)$. Using the example (Figure 2-13, Panel 0), member A_0 belongs to only layer L_0 , and therefore, its control path peers are the other members in its L_0 cluster, i.e. A_1, A_2, and B_0 . In contrast, member B_0 belongs to layers L_0 and L_1 and therefore, its control path peers are all the other

members of its L_0 cluster (i.e. A_0, A_1 and A_2) and L_1 cluster (i.e. B_1, B_2 and C_0). In this control topology, each member of a cluster, therefore, exchanges soft state refreshes with all the remaining members of the cluster. This allows all cluster members to quickly identify changes in the cluster membership, and in turn, enables faster restoration of a set of desirable invariants.

```
Procedure : MulticastDataForward(h, p)
      { h ∈ layers L0, ... , Li in clusters Cl0(h), ... , Cli(h) }
for j in [0, ... , i]
    if (p ∉ Clj(h))
        ForwardDataToSet(Clj(h) − {h})
    end if
end for
```

Figure 2-14: Data forwarding operation at a host, h, that itself received the data from host p. [BANE2002]

The delivery path for multicast data distribution needs to be loop free, otherwise, duplicate packet detection and suppression mechanisms need to be implemented. Therefore, in the NICE protocol the data delivery path is a tree. More specifically, given a data source, the data delivery path is a source-specific tree, and is implicitly defined from the control topology. Each member executes an instance of the Procedure *MulticastDataForward* given in Figure 2-14, to decide the set of members to which it needs to forward the data. Panels 1, 2 and 3 of Figure 2-13 illustrate the consequent source-specific trees when the sources are at members A_0, A_7, and C_0 respectively. It is called the *basic data path.*

2.3.4.3 P2Cast

P2Cast [GUO2003] is an architecture that uses a peer-to-peer approach to cooperatively stream video using *patching* [HUA1998, SEN1999], while relying only on unicast connections among peers. P2Cast clients not only receive the requested stream, but also contribute to the overall VoD service by forwarding the stream to other clients and caching and serving the initial part of the stream. Associated with P2Cast is a *threshold*, ***T***. The clients that arrive within the threshold constitute a *session*. Together with the server, clients belonging to the same session form an application-level multicast tree, denoted as the *base tree*. The server streams the entire video over the base tree. We denote this complete video stream as the *base stream*. When a new client joins the session, it joins the base tree and retrieves the base stream from it. Meanwhile, the new client must obtain a "patch" - the initial part of the video from the start of the session to the time it joined the base tree. It obtains the patch from the server or another client. P2Cast clients behave like peers in a P2P network, and provide the following two functions:

- *Base Stream Forwarding*. P2Cast clients need to be able to forward the received base stream to other clients so that clients and the server can form an application-level multicast tree over which the base stream is transmitted.
- *Patch Serving*. P2Cast clients need to have sufficient storage to cache the initial part of the video. A P2Cast client can then serve the patch to other clients.

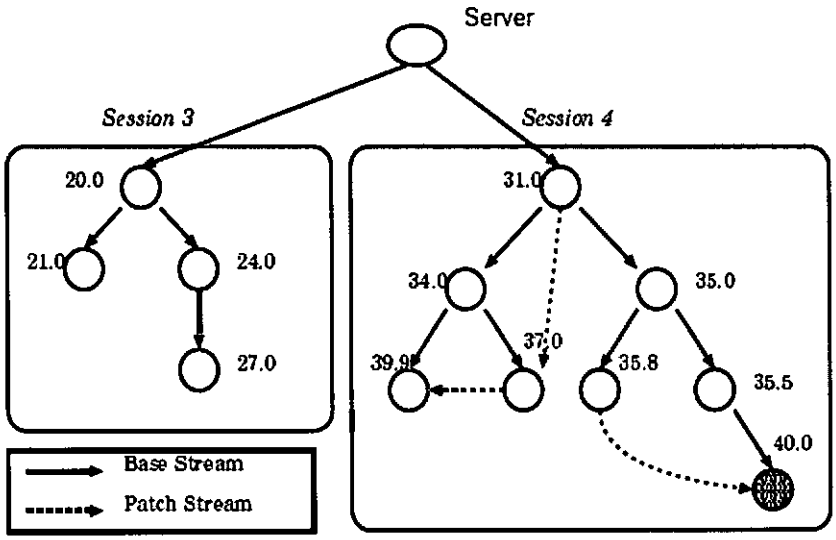


Figure 2-15: A snapshot of P2Cast at time 40 [GUO2003]

Fig. 2-15 illustrates a snapshot of P2Cast at time 40. It shows two sessions, session 3 and session 4, starting at time 20.0 and 31.0, respectively, with the threshold equal to 10. Clients in a session form an application-level multicast tree together with the server. At time 40, all clients in session 3 have finished the patch retrieval; while three clients in session 4 are still in the process of receiving the patch stream.

2.3.4.4 Narada

Narada [CHU2002] is a protocol designed to implement End System Multicast. In Narada, end systems self-organize into an overlay structure using a fully distributed protocol. Further, end systems attempt to optimize the efficiency of the overlay by adapting to network dynamics and by considering application level performance.

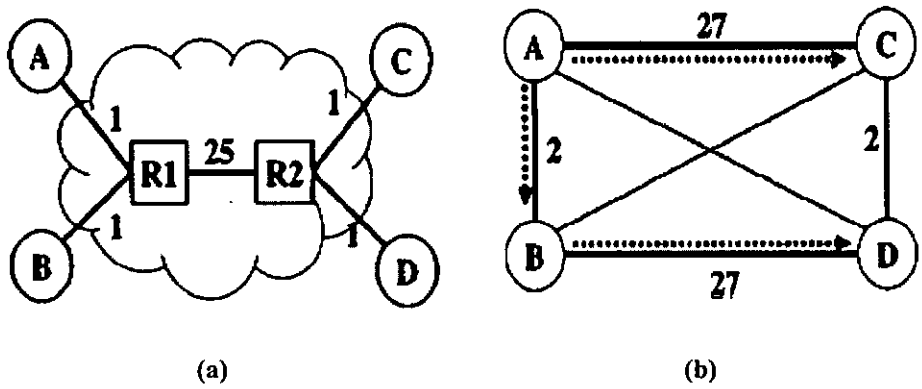


Figure 2-16: Examples to illustrate the mesh-based approach in Narada [CHU2002]

There are two basic methods for construction of overlay spanning trees for data delivery. The first approach is to construct the tree directly - that is, members explicitly select their parents from among the members they know. Narada however constructs trees in a two-step process. First it constructs a richer connected graph that we term mesh. The mesh could in general be an arbitrary connected sub-graph of the Complete Virtual Graph (CVG). In the second step, Narada constructs (reverse) shortest path spanning trees of the mesh, each tree rooted at the corresponding source using well known routing algorithms. Figure 2-16(b) presents an example mesh that Narada constructs for the physical topology shown in Figure 2-16(a), along with the shortest path spanning tree rooted at A. There are several reasons for this two step process. First, group management functions are abstracted out and handled at the mesh rather than replicated across multiple (per source) trees. Second, distributed heuristics for repairing mesh partition and mesh optimization are greatly simplified as loop avoidance is no longer a constraint. Third, we may leverage standard routing algorithms for construction of data delivery trees. Finally, a mesh is more

resilient to the failure of members than a tree and heavy weight partition repair mechanisms are invoked less frequently.

Chapter 3

Performance Study on Hybrid ARQ for Video Streaming

3.1 Introduction

As mentioned before, packet loss is a critical problem for streaming video over the Internet. For transmission of digital video over the Internet, the compressed bit-stream is split up into packets. Video packets can be lost and this would degrade the quality of received video. Assuming each frame of a video is transmitted using multiple packets. If a packet is lost in the middle of a frame, the phase alignment is lost for the rest of the frame. Depending on the synchronization scheme, a packet loss may corrupt a large part of a frame. If a sophisticated video compression technique like MPEG is used, the problem becomes more serious since variable-length and differential coding are used. In such situation, a packet loss will cause error propagation to other frames of the sequence, which in turn leads to perceivable degradation on image quality. Error propagation occurs in both spatial and temporal domain. Spatial error propagation within an image is due to synchronization failure in variable-length coding (VLC) and information loss in differential coded data. On the other hand,

error propagation in temporal domain is due to the use of motion-compensated predictive coding. The accumulated error leads to serious visual degradation on output image quality. Therefore, the demand for error concealment is obvious.

In the last decade, many different methods have been proposed to minimize the adverse effect on output video quality caused by packet loss. These methods can be basically divided into two categories, the protective method and error concealment method. A review on different error control and concealment methods can be found in [OHTA1994], [WANG1998] and [RHEE2000]. The protective method [OHTA1994] tried to prevent a video stream from loss. A typical protection method is to add forward error correction (FEC) coding on the bit-stream at the encoder. Apart from the protective methods, a number of error concealment techniques have also been developed for restoring corrupted video data at the decoder. They can be grouped into two categories, one is interactive error concealment and the other is error concealment by signal processing techniques at the receiver. For interactive error concealment, both encoder and decoder work cooperatively to minimize the impact of packet loss. The cooperation can be realized at either the source coding or transport level. At the source coder level, coding parameters can be altered based on the feedback information. At the transport level, the feedback information can be used to adapt the bandwidth for retransmission when automatic retransmission request (ARQ) is adopted.

If the probability of transmission errors on the link (channel) is very small, ARQ is efficient for concealing these error effects with the cost of some delay. Otherwise, FEC may be more efficient with the cost of some extra bandwidth. Recently, a hybrid ARQ system [MAJU2002, SACH2001] that makes use of both FEC and ARQ techniques is developed. In a normal FEC protective scheme, the parity data is always sent with original data. However, parity data is sent just when some packets are lost and retransmission is required in the hybrid ARQ system. In case of no error, only original data packets are transmitted. When the channel is unstable and some packets are lost, the parity data are sent to recover the lost packets. To investigate the performance of the hybrid ARQ technique in a video streaming system, a detailed simulation study will be performed in this chapter.

Following the introduction, the detailed operations of a hybrid ARQ system are described in section 3.2. Different transmission scenarios are also discussed in this section. A detailed simulation study will be conducted and the results are presented in section 3.3. An adaptive hybrid ARQ scheme that can maximize the bandwidth utilization is then developed in section 3.4. Finally, a conclusion of this chapter is given in section 3.5.

3.2 Hybrid ARQ Techniques

3.2.1 Operations of the Hybrid ARQ Technique

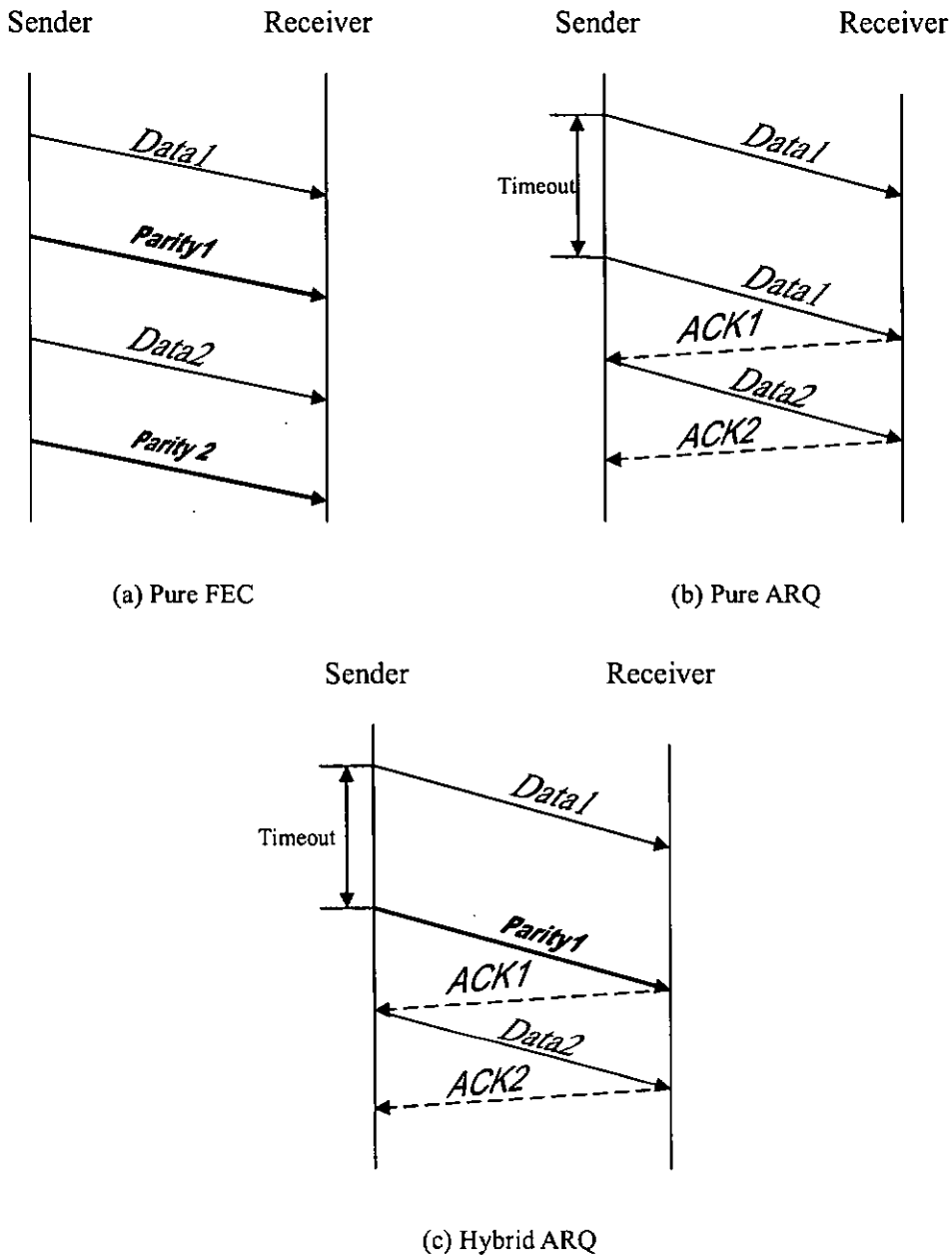


Figure 3-1: FEC, ARQ and Hybrid ARQ scheme

An illustration of the operations of FEC, ARQ and hybrid ARQ is shown in

Fig.3-1. In the figures, "Parity$x$" denotes the parity block of the original data block "Data$x$". The transmitted data is divided into FEC packet groups, each group includes N FEC data packets (forming original data block, such as "Data1" and "Data2") and M FEC parity packets (forming parity data block, such as "Parity1" and "Parity2") created by a FEC algorithm (like Reed Solomon Coding). The receiver has to get any N packets out of $N+M$ packets in the same FEC packet group in order to recover the original data.

In the FEC scheme, as shown in Figure 3-1a, both parity data and original data are sent to the receiver. No feedback/acknowledgement is required. Packet loss can be corrected by a calculation between the original data and additional parity data. However, in the case of no transmission error, parity packets are still be transmitted. Hence, some bandwidth is waste.

As shown in Figure 3-1b, no parity data is involved in the ARQ scheme. The succeed delivery of data is acknowledged by an acknowledgement (denoted as ACK) from receiver. If no ACK is received by the sender before timeout, packets are resent. The ARQ scheme is used in TCP. In Figure 3-1b, the first delivery of "Data1" is with error and the sender has to resend it after the timeout. "ACK1" is sent from the receiver to acknowledge the sender that the corresponding data is successfully received. Then the sender continues to send "Data2". If the channel is unstable (especially when ACK is lost in feedback channel), too much waiting

time for ACK timeout is required and the throughput is degraded.

In the hybrid ARQ technique shown in Figure 3-1c, initially an original data block (including N original data packets) is sent to the receiver. Then the transmitter starts sending parity packets until one of the following two events occur: either an acknowledgment arrives, or all parity packets are sent out. Once at least N packets are received, the receiver sends an acknowledgment. After the acknowledgment is received, the transmitter continues with the FEC data packets of the next original data packet group. As shown in Figure 3-1c, some packets of "Data1" are lost, hence the sender does not receive the acknowledgement before timeout. After the sender sends m parity packets ($m \leq M$), the receiver successfully get N data packets from the same FEC data group. The receiver sends "ACK1" and the sender starts sending "Data2". Data2 is delivered with no error, so "ACK2" is received and no parity packets are required.

Hybrid ARQ uses ACK from receiver to confirm a successful delivery and uses FEC parity data to recover transmission error. Parity data packets are sent only in case of transmission error, so no extra bandwidth is wasted under the no error situation. In the case of an ACK loss, the sender will send other FEC data groups out after sending out the corresponding parity packets.

3.2.2 Throughput Improvement on Hybrid ARQ

The ARQ algorithm used in the hybrid ARQ scheme introduced in Fig.3-1c is stop-and-wait ARQ, in which the sender wastes time to wait for the acknowledgement. To reduce the waiting time and achieve a higher throughput, Selective Reject (SR) ARQ can be used.

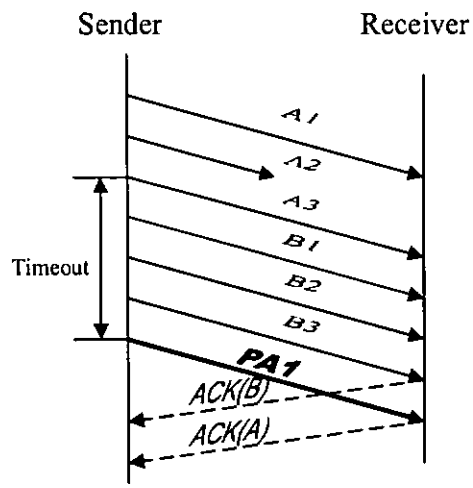


Figure 3-2: Hybrid ARQ with SR-ARQ

As illustrated in Figure 3-2, in the modified Hybrid ARQ, after sending out all original data packets (*A1*, *A2*, *A3*) of a FEC data packet group (Group A), the transmitter does not stop sending and wait for the ACK, but start sending other FEC data packets (*B1*, *B2* and *B3*) of the next FEC data packet group (Group B). An ARQ waiting time is set in the sender. In Figure 3-2, one original data packet (*A2*) of Group A is lost. When the ACK waiting time is passed, a parity packet (P_{A1}) of Group A is sent. Parity packet P_{A1} is sent after *B3*, and the sender has to wait for the ACK of Group A for another ARQ waiting time. After receiving P_{A1}, totally 3 packets of Group A is received by the receiver side, original data of Group A can be recovered by the FEC decoder. An acknowledgement for Group

A, *ACK(A)*, is sent and received by the sender in time. Hence no further parity packet of Group A is required. Fig. 3-2 shows that no error occurs in the transmission of Group B, and *ACK(B)* is received in time with no Group B parity packet sent. In a Hybrid ARQ scheme with SR-ARQ, the sender will never stop the data transmission in order to wait for an ACK. Instead it sends other data group in the ACK waiting period. Therefore, the transmission throughout is improved.

3.2.3 Scenarios in Hybrid ARQ

Scenario 1: No Transmission Error

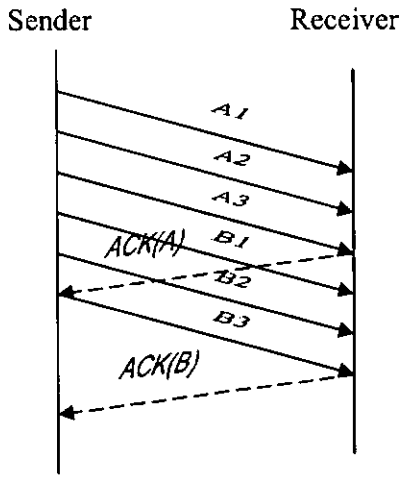


Figure 3-3: No Transmission Error

Figure 3-3 shows the no error scenario, in which the receiver sends *ACK(A)* to acknowledge the sender that FEC data group A is received. Group A is then reconstructed by *A1*, *A2* and *A3*.

Scenario 2: Sender to Receiver Transmission Error (Error Recoverable)

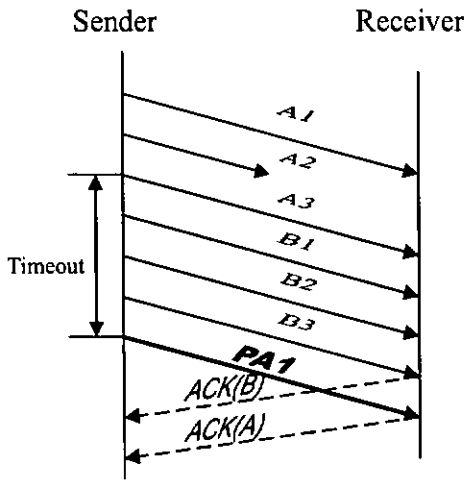


Figure 3-4: Sender to Receiver Transmission Error (Recoverable)

As described in Chapter 3.2.2, in this scenario (Figure 3-4), $A2$ is lost and P_{A1} is sent when the $ACK(A)$ waiting time is passed. Finally Group A is recovered by $A1$, $A3$ and P_{A1}.

Scenario 3: Sender to Receiver Transmission Error (Error Unrecoverable)

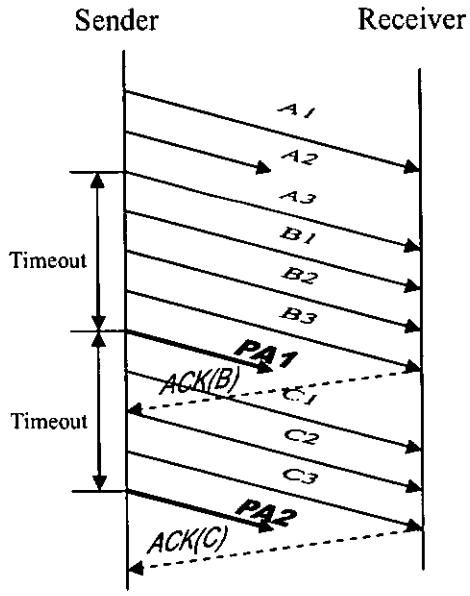


Figure 3-5: Sender to Receiver Transmission Error (Unrecoverable)

Figure 3-5 shows an error unrecoverable scenario, all parity packets (P_{A1} and P_{A2}) of Group A are sent to recover the error, but $A2$, P_{A1} and P_{A2} are lost. In the receiver side, only two packets ($A1$ and $A3$) of Group A are received, and Group A is unrecoverable.

Scenario 4: ACK Transmission Error

Figure 3-6 shows the scenario of ACK transmission error. The first $ACK(A)$ is lost and the sender sends P_{A1} after timeout. Another $ACK(A)$ is sent by the receiver again when it gets P_{A1}. Finally Group A is recovered by $A1$, $A2$ and $A3$. If the feedback channel is very unstable and every ACK is lost in transmission, the sender will send all parity packets to provide maximum reliability. Hybrid ARQ without feedback channel just works as pure FEC scheme.

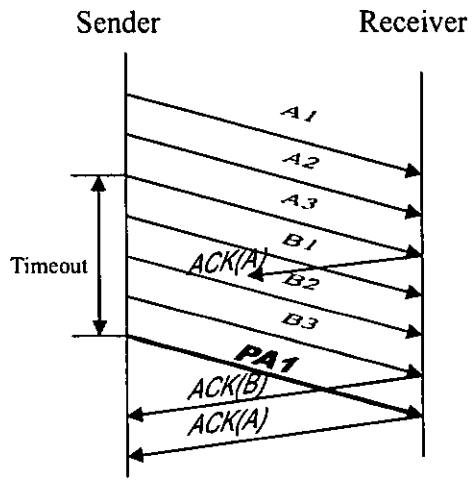


Figure 3-6: ACK Transmission Error

3.3 Performance Study of Hybrid ARQ for Video Streaming

3.3.1 Design of a Hybrid ARQ Video Streaming System

In this section, we are going to investigate the effect of various parameters on the system performance of a hybrid ARQ video streaming system. A simulation platform as shown in Figure 3-7 is set up to conduct the evaluation.

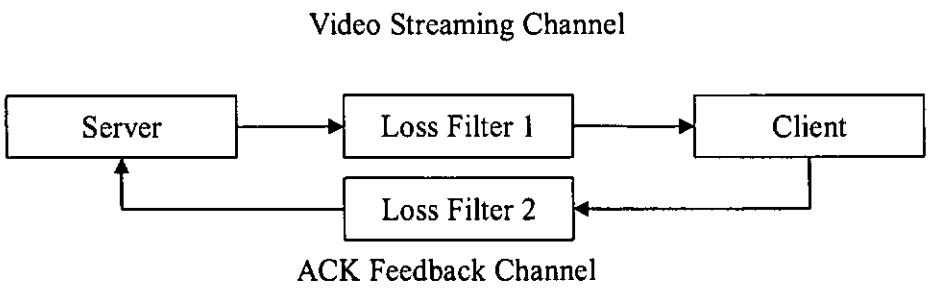


Figure 3-7: Hybrid ARQ simulation system diagram

As shown in Figure 3-7, the server sends FEC coded video data to the client in the forward streaming channel. The client sends acknowledgements to the server in the feedback channel. Loss filters are used to simulate the packet loss and added to both channels.

In our experiments, we use the RS codec developed by Phil Karn (http://www.ka9q.net/) to generate the FEC coded data. The coder creates 1-32 parity symbols for 223 original ones. The symbol size is 8 bits - one byte.

Normally a shortened RS coding like RS (15,5) or RS (10,5) is used by padding some "0"s into the original symbols set and taking only part of the parity symbols. Two patterns of packet loss are simulated. One is uniform distributed loss model and the other is Markov Chain loss model.

In all simulations, we use the hybrid ARQ with SR-ARQ scheme (see Chapter 3.2.2). We encode one video frame into a FEC data group, which includes N original data packets and M parity data packets. When any N packets out of these $N+M$ packets from the same FEC data group are received in the client side, the corresponding video frame is reconstructed. The frame loss rate, which is defined as the number of frames that cannot be recovered at the client, is used as the performance criterion in the study. Different settings on N, M, packet loss rate and pattern of loss filters are simulated to test video frame loss rate in different situations. All simulations are run for one million frames. Since the loss of an ACK in the feedback channel does not affect the frame loss rate much, we assume that no error occurs for ACK transmission in our simulations.

3.3.2 Simulations

3.3.2.1 Frame Loss Rate Simulation

In this part, all video frames in the simulation are I-frames.

Uniform Loss Simulation

The packet loss filter 1 (video streaming channel) is set to uniform loss. The effect on frame loss rate for different settings on forming the FEC data group, i.e. N and M, is investigated. Firstly we set N=M, Figure 3-8 shows the simulation results for the frame loss rate under different packet loss rate. It is seem from the figure that a large number value of N gives a better loss recovery capacity. However, when N gets larger, the complexity of FEC coding will become higher. Therefore, a moderate value of N should be selected to compromise between the error recovery capacity and the system complexity.

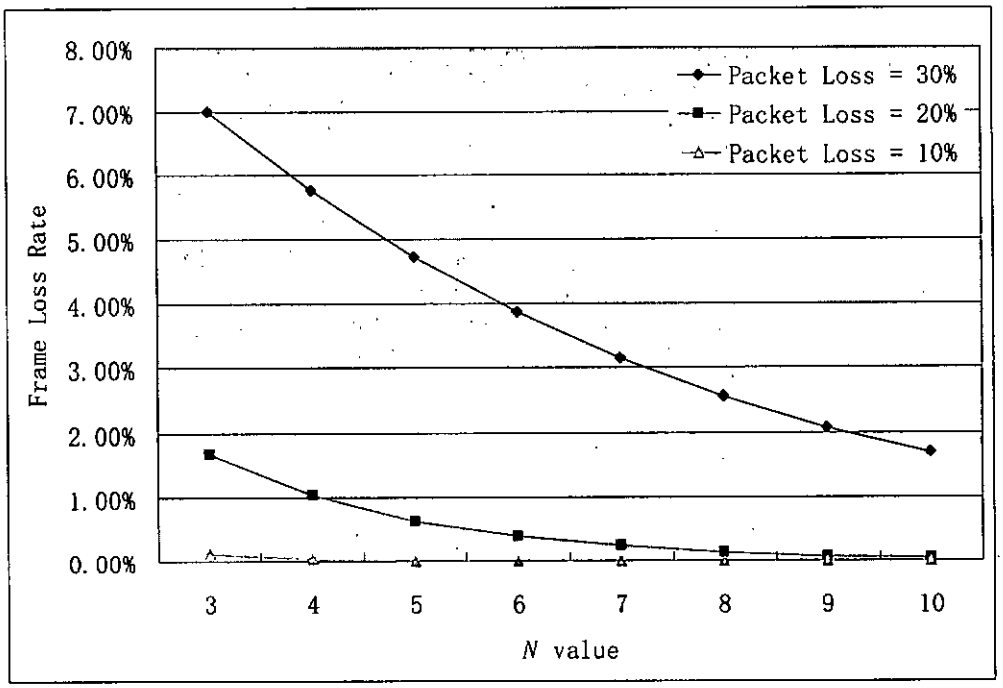


Figure 3-8: Simulation results for N=M, (N=3~10, 10%~30% packet loss)

We then fix N to 5 and investigate the impact of M on the frame loss rate. The results are then listed in Table 3-1. Similar observations as above are obtained. Given a fixed N, a larger value of M will give a smaller frame loss rate, and

hence a better error recovery capacity. However, the system complexity is increased.

N	*M*	Frame Loss Rate (*N*=5)	Frame Loss Rate (*N*=*M*)
5	1	34.29%	-
5	2	14.80%	-
5	3	5.58%	1.69% (N=3, M=3)
5	4	1.96%	1.04% (N=4, M=4)
5	5	0.60%	0.60% (N=5 M=5)
5	6	0.20%	0.39% (N=6, M=6)

Table 3-1: Simulation results for *N*=5 (20% packet loss)

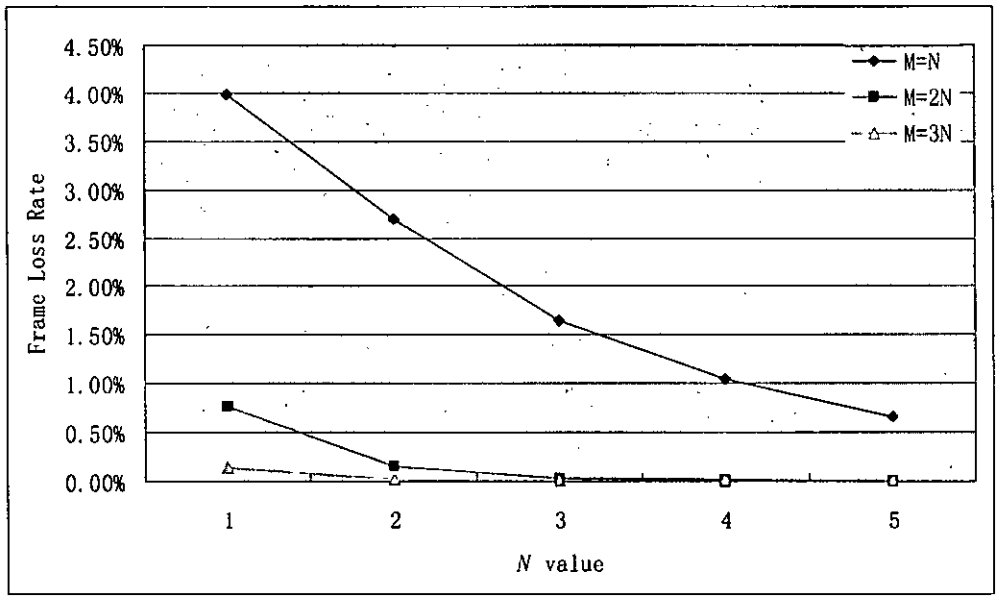


Figure 3-9: Simulation results for different *M/N* ratio (*N*=1~5)

To further investigate the effect *M* and *N*, we study the frame loss rate for different *M/N* ratios. Figure 3-9 shows the results. It is seen that a large *M/N* always gives a better packet loss recovery performance. When the ratio of *M/N* is fixed, a larger *N* (a larger *M* at the same time) makes the system perform better. A

larger ratio of *M/N* can greatly improve the packet loss recovery performance. However, it may create a long frame delay and jitter. Because every parity packet sending requires an ACK waiting time, a long frame delay is created when more parity data packets needs to be sent. In a VoD service, the long frame delivery delay can consume a large buffer. In a real-time application (such as video conferencing) which cannot tolerate a long delay, a large *M/N* ratio may not be a good idea. From results above, it is noted that a moderate value of *N* and *M* is suggested in view of the balance between error recovery capacity and system complexity and delay.

Markov Chain Loss Simulation

In the following, we use another loss filter module to simulate the packet drop using the Markov chain process [FELL1968, KALL1997 and STEW1995].

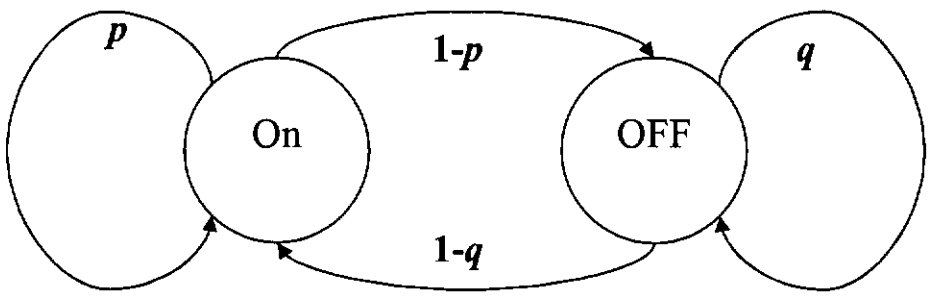


Figure 3-10: Two-state Markov Chain Model

Figure 3-10 shows a two-state Markov chain model we used for packet loss pattern. "On" state presents successful packet delivery and "Off" state means

packet loss. p is the probability of "On" state following an "On" state and q is the probability of "OFF" state following an "OFF" states. The average on probability of the model shown in Figure 3-10 is $\frac{1-q}{2-p-q}$, that is to say the average packet loss rate of the loss filter with above model is $1-\frac{1-q}{2-p-q}=\frac{1-p}{2-p-q}$. The burstiness of such a Markov Chain model is $\frac{(1-p)(q+p)}{(2-q-p)^2}$. A higher burstiness in our packet loss model implies a higher average number of losses that occur in each "loss event", i.e. in each burst of losses. The average number of continuous packet losses is $\frac{1}{1-q}$, and the average number of continuous successful packet delivery is $\frac{1}{1-p}$.

Figure 3-11 and Figure 3-12 show the simulation results with different p, q, M and N while $\frac{1-q}{2-p-q}$=0.8 (average package loss rate is 20%).

In Figure 3-11, we can see that the overall frame loss rate rises as q arises. q is the off→off probability, a large q means a higher burstiness of packet loss. Because the original data packets are transmitted continuously, we may loose too many of them in a long error burst and RS coding may not recover the group.

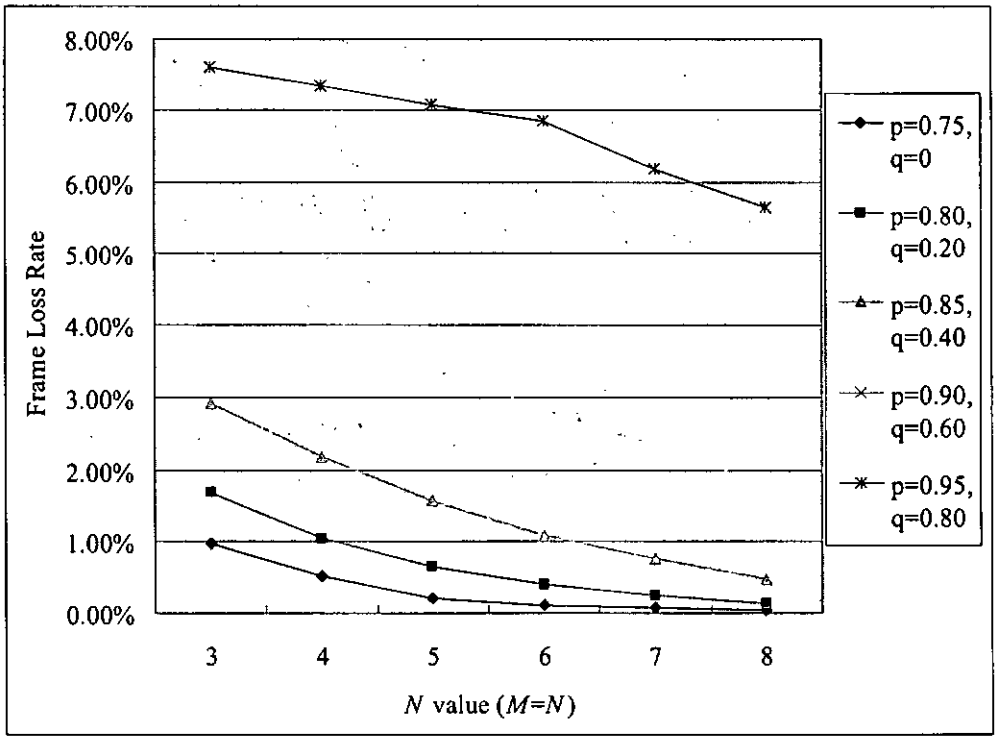


Figure 3-11: Markov Chain Loss model Simulation results (M=N, N=3~8)

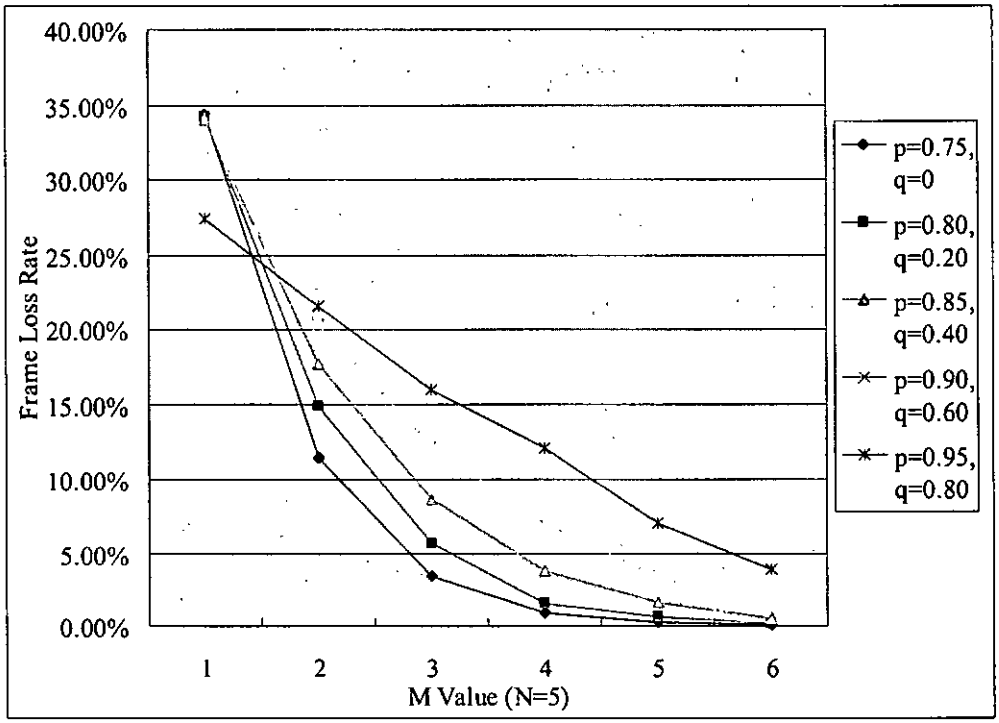


Figure 3-12: Markov Chain Loss model Simulation results (N=5, M=1~6)

Figure 3-12 shows a different result from Figure 3-11. When M=1 and N=5, a higher q leads to less frame loss. When q value is higher, p also got a higher

value, which leads to a longer "On" burst length (continuous successful delivered packets number, also the distance between two "loss event") and a longer "Off" burst length (continuous packet loss). When M=1, two packet losses out of a FEC data packet group make a frame unrecoverable. When p=0.75, q=0, "Off" burst: 1, "On" burst: 4, the short error burst length is enough to drop one or more packets from the same FEC data packets group, and the short burst success cannot guarantee the whole original data block (containing 5 packets) received. In contrast, when p=0.95, q=0.80, "Off" burst: 5, "On" burst: 20, a long burst error length may completely destroy one FEC group or two, but the long burst success will make three or more FEC group recovered.

In Figure 3-11, there are three or more parity packets in one FEC group and they are interleaved. A short error burst length cannot destroy too many packets from the same FEC group. A long error burst is more dangerous. Besides, for a larger N case represented in Figure 3-11, the longer success delivery burst can protect less FEC Group (less video frames) than it does for a small N case illustrated in Figure 3-12.

3.3.3 Performance Test on an Implemented Hybrid ARQ Video Streaming System

A hybrid ARQ Video Streaming System is implemented to test the performance in a practical situation. The system includes a server program and a client

program. The server program encodes video frames captured by a video camera, and transmits the video data to the client on a UDP channel using the Hybrid ARQ scheme. A FEC packet group includes one encoded video frame data and some RS coding parity data. A filter is added to simulate the packet loss. Some third party software can perform the packet loss simulation, such as **"Shunra\Cloud"**[SHUN] and **"NIST Net"**[NIST].

The video server simulation program runs on a P4-2G PC, and the video client program runs on a P3-1G machine. The connection between two PC is a 100Mbps LAN. The embedded packet loss filter fixes the loss rate to 20% (uniformly distributed), video frame rate = 20fps, client buffer time = 0.5 second (10-frame buffer), ARQ waiting time is set to 450ms/M, (N: number of original data packet in one FEC group; M: number of parity data packet in one FEC group). The video server sends 1000 video frames (all I-frames) to the client. The results are summarized in Table 3-2 and 3-3.

N	M	Frame Loss Rate	Useless parity packets client received	Frames dropped (client side)
3	3	1.70%	0	56
4	4	1.20%	5	12
5	5	1.00%	2	0
6	6	0.50%	28	0
7	7	0.40%	41	0
8	8	0.20%	40	0

Table 3-2: Simulation results for N=M, (20%packet loss)

In Table 3-2, the frame loss rate decreases as the values of N and M increase. When N=8 and M=8 the lowest loss rate is 0.20%. When M is large and the ARQ

waiting time becomes small, occasionally one useless parity packet is sent before the ACK comes back from the client. Besides, for a small *N*, the packet size increases and more RS decoding are performed, so the CPU loading increases and some frames are dropped

N	M	Loss	Useless parity package client received	Frames dropped (client side)
5	1	37.10%	0	0
5	2	20.10%	0	0
5	3	8.90%	0	0
5	4	4.90%	0	0
5	5	1.70%	1	0
5	6	0.60%	4	0

Table 3-3: Simulation results for N=5, M=1~6 (20% packet loss)

As shown in Table 3-3, when N=5, M=6 the lowest loss rate is 0.6%. More parity packets give a lower loss rate. Useless parity packets are received when M is large. It is because the short ACK waiting time is occasionally smaller than propagation delay.

From the simulation results, it could be seen that hybrid ARQ can perform quite well with a proper setting (N =5, M =6, loss rate 0.6%; N=M=8, loss rate =0.2%). Since the above simulation is real-time, frame-drop sometimes happens, and with some uncontrollable delay (networking delay and program delay), ARQ function fails when ARQ waiting time is small.

3.4 An Adaptive Hybrid ARQ Scheme

3.4.1 Bandwidth Vs Frame Loss rate

In all simulations described in the above section, all video frames are I-frames. So if one frame fails to be recovered by the hybrid ARQ scheme, it won't affect later frames received. In MPEG standard, P-frames and B frames are used to reduce the size of media date to be transmitted or stored [BASI1996]. In this section, the relationship between the bandwidth required for hybrid ARQ and frame loss rate is studied. A video with the IPPPIPPPIPPPI pattern (Figure 3-13) is simulated.

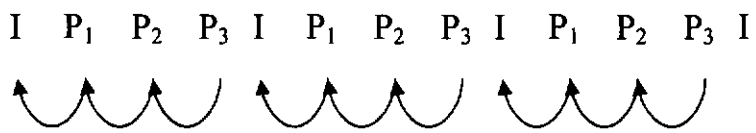


Figure 3-13: I frame and P frame

Because of the I-frame and P-frame properties, an I-frame loss affects the following 3 P-frames (P_1, P_2 and P_3), so totally 4 frames are lost. A P_1 frame loss affects the following 2 P-frames, P_2 and P_3, so three frames are lost. With the same reason, a P_2 frame loss leads to two frames lost and a P_3 frame loss damages itself.

The simulation program simulated 1,000,000 frames including 250,000 I-frames and 750,000 P-frames. We assume the size of an I-frame is three times larger

than the size of a P-frame. I-frame data is divided into three FEC data packets, while P-frame data is packed in one FEC data packet. So every data packet is of equal size whether it is from an I-frame or a P-frame. 3 FEC parity packets is prepared for an I-frame, the parity packet number for a P-frame are sometimes set to 1 and sometimes set to 0. Without parity packet or without packet error, the minimum number of packets the server sends out is 1,500,000. The packet loss is uniform, and loss rate is from 5% to 20%.

Table 3-4 shows the frame loss rate in the client side and the normalized bandwidth required in the server side. Without any packet loss error, the server will send 1,500,000 packets. In the simulations, we record the number of packets sent by the server and normalize the value by dividing 1,500,000. The normalized bandwidth consumption is indicated in the bracket under the frame lost rate.

Packet Loss Rate (uniform)	I(N=3, M=0) P_1(N=1,M=0) P_2(N=1,M=0) P_3(N=1,M=0) Case 1	I(N=3, M=3) P_1(N=1,M=0) P_2(N=1,M=0) P_3(N=1,M=0) Case 2	I(N=3, M=3) P_1(N=1,M=1) P_2(N=1,M=0) P_3(N=1,M=0) Case 3	I(N=3, M=3) P_1(N=1,M=1) P_2(N=1,M=1) P_3(N=1,M=0) Case 4	I(N=3, M=3) P_1(N=1,M=1) P_2(N=1,M=1) P_3(N=1,M=1) Case 5
20%	62.24% (1.000)	27.34% (1.122)	17.85% (1.155)	11.05% (1.187)	7.38% (1.221)
15%	51.26% (1.000)	20.79% (1.087)	12.71% (1.113)	6.94% (1.136)	3.93% (1.162)
10%	36.75% (1.000)	14.11% (1.055)	8.06% (1.073)	3.83% (1.088)	1.63% (1.105)
5%	20.43% (1.000)	7.22% (1.026)	3.88% (1.035)	1.56% (1.043)	0.39% (1.051)

Table 3-4: Frame loss rate and server bandwidth consumption

From Table 3-4 we can see that without hybrid ARQ protection (Case 1), frame loss rate is very high. The hybrid ARQ scheme which gives rare protection against packet loss (Case 2) saves bandwidth but surfers a higher frame loss rate. An intensive packet loss protection scheme (Case 5) gives a much better packet loss protection performance with higher bandwidth consumption. However, it is hard to tell which setting is better, because it is always a tradeoff between higher packet loss protection capability and server bandwidth consumption.

3.4.2 Adaptive Hybrid ARQ

We can see that in hybrid ARQ, more bandwidth consumption always gives a higher packet loss protection performance. But it is hard to determine in a certain environment, how the N and M parameters should be set. The value of N can be determined easily by video frame size and packet size. For example the encoded frame data is 3k bytes long. A data packet is in size of 1k bytes, so N = 3k bytes/1k bytes = 3. On the other hand, the value of M is harder to optimize. A large M always gives a better packet loss protection capability, but requires more bandwidth in the video server side. In normal application the bandwidth provided to the video server is fixed. But it may not be fully utilized in a hybrid ARQ application as the example illustrated in Table 3-5 (part of Table 3-4). If the provided bandwidth is 1.08, only 1.073 is used by N and M settings of Case 3. By adding a little bit more parity packets like in Case 4, the server requires a bandwidth of 1.088, which is larger than the provided bandwidth.

Packet loss rate (Uniform)	I(N=3, M=3) P_1(N=1, M=1) P_2(N=1, M=0) P_3(N=1, M=0) Case 3	I(N=3, M=3) P_1(N=1, M=1) P_2(N=1, M=1) P_3(N=1, M=0) Case 4
10%	8.06%(1.073)	3.83% (1.088)

Table 3-5: Bandwidth underutilization

In this regard, we propose an adaptive hybrid ARQ scheme to solve this problem. The inspiration is got from the shared memory ATM switch architecture. In a common shared memory ATM switch, the buffer size for different queue has a lower bound and an upper bound. The lower bound provides basic buffer size of a queue, and the upper bound limits the buffer size and prevents one queue occupy too much shared buffer memory.

We set a lower bound and an upper bound for M, M_U and M_L. M_U sets the upper limit number of parity packet and M_L is the lower limit of parity packet number. M_U parity packets will be prepared for a FEC group. When the ACK of a FEC group is not received within the waiting time and a parity data packet need to be sent, the server checks the sent parity packets number of that FEC group, K. If $K<M_L$, the server sends the parity packet as normal. If $M_U>K\geq M_L$, it will check current bandwidth consumption and determine whether the parity packet should be sent. If the bandwidth is not fully utilized, the parity packet is sent. If all bandwidth is occupied, the parity packets will not be sent and that FEC group won't be recovered.

A detailed procedure of our adaptive Hybrid ARQ scheme is shown below. M_U is the upper limit number of parity packet and M_L is the lower limit of parity packet number. $M_U \geq M_L \geq 0$. B is the fixed bandwidth allocated for the video server, b is the current bandwidth consumption monitored ($B \geq b$), and K is the already sent parity packet of a FEC group.

```
    When ACK waiting time for a certain FEC group is timeout
if K<M_L
        Send parity packet, and K++
end if
if M_U>K≥M_L,
        if b<B,
                Send parity packet, and K++,
        end if
        if b=B,
                Do not send parity packet, and give up that FEC group.
        end if
end if
```

Figure 3-14: Adaptive Hybrid ARQ Scheme

The server always monitors packets sending speed and make it sure that it doesn't send too many packets that the allocated network bandwidth can not support. The server sends more parity packets when extra bandwidth can support, and the parity packets will be used to protect the most important frames.

Packet loss rate (uniform)	I(N=3, M=3) P_1(N=1, M=1) P_2(N=1, M=0) P_3(N=1, M=0) Case 3	I(N=3, M_L=3, M_U =3) P_1(N=1, M_L=1, M_U =1) P_2(N=1, M_L=0, M_U =1) P_3(N=1, M_L=0, M_U =0) Adaptive Scheme	I(N=3, M=3) P_1(N=1, M=1) P_2(N=1, M=1) P_3(N=1, M=0) Case 4
10%	8.06%(1.073)	**5.99% (1.08)**	3.83% (1.088)

Table 3-6: Adaptive Hybrid ARQ scheme utilizes server bandwidth

By comparing the Table 3-5 and Table 3-6, we can see that the new method can fully utilize the provided bandwidth to send as many parity packets as possible, and correct more packet loss at the same time.

3.5 Chapter Conclusion

Hybrid ARQ is a technique that can effectively improve the communication reliability in a packet loss channel. ACK is sent back through the feedback channel to acknowledge the transmitter that a certain FEC packet group has been received. In the case of packet loss, no ACK comes back before the ARQ deadline, parity packets are sent out to replace the lost packets.

In a 20% uniform packet loss channel, with proper setting on hybrid ARQ, video streaming application can achieve a frame loss rate of less than 1%. The cost of the reliability improving is the extra bandwidth required to send parity packets. Hybrid ARQ overcomes the disadvantage of pure FEC, in which parity packets are always sent whenever packet loss occurs and the unnecessary parity packets are sent out. Another advantage of hybrid ARQ is that the transmitter can recover

the packet loss without knowing which packet is lost. Parity packets can replace any lost packets; no matter the lost packets are original data packets or parity data packets. This property of hybrid ARQ overcomes the disadvantage of pure ARQ, in which the transmitter must wait for some feedback to know which packet is lost, before retransmitting the same packet.

To fully utilize the bandwidth provided in multimedia streaming, we propose an adaptive Hybrid ARQ scheme and set an upper bound and a lower bound to the number of parity packet in a FEC packet group. Such a setting can make use of any chance to send more parity packets within the allocated bandwidth, so more packet losses can be recovered.

Chapter 4

A New Peer-to-Peer Architecture for On-Demand Video Streaming

4.1 Introduction

Providing on-demand video streaming or video on demand (VoD) services over the Internet is a challenging task for not only the huge transmission bandwidth required for a long time but also the wide spread of a lot of clients over the network. The traditional way to do VoD is by simple client/server direct streaming, in which the server makes a connection with each client and streams data directly to them at the video bit rate. The resources required for such a VoD server are proportional to the number of clients in the system. It may be workable in a small scale, e.g. tens of clients. However, for a popular movie attracting hundreds or thousands of people, a normal VoD server can hardly handle the huge network throughput at that way. To increase the system capacity, the only way is to use multiple servers and increase their resources. Obviously, this approach is not scalable and cost-effective. Our objective in this chapter is to tackle this problem using a peer-to-peer (P2P) network model.

The idea of peer-to-peer is not new, many file-sharing systems based on the P2P

concept have been developed [RATN2001, STOI2001]. The basic idea of such approaches is that a shared file is copped into pieces and the user simultaneously downloads these pieces separately from different peers who have one or more pieces. However, for VoD services, users want to watch the video as soon as possible and the video data has to be obtained following the video time sequence.

Most recently, some researches have been done on P2P media streaming. The P2Cast approach [GUO2003] uses the technique of patching [HUA1998, SEN1999] and groups peers based on their system enter time. In the Zigzag [TRAN2004] approach, it clusters peers to a complex administrative organization and build the multicast tree atop of this hierarchy. Both Zigzag and P2Cast requires application level multicast support. Due to limited deployment and access to IP multicast, the multicast P2P is complex.

Shan et al. [SHAN2003] has proposed a hybrid video downloading/streaming scheme (HDS). In this scheme, a client will receive video stream from more than one source. HDS assigns downloading sessions from each source according to the measured bandwidth and video content availability. Another multiple source video streaming coordination scheme is described in [HEFE2004]. In their system, supplying peers are scheduled to send video stream to requesting peers at different rates. In the above coordination schemes, if any source fails to do its assigned works, the requesting peer will fail to get data segment completely. In

this chapter, we propose another solution using the idea of Hybrid ARQ.

In our proposed system, a peer will obtain data from not only a P2P forwarding chain but also a server that contains both FEC coded and original video data. The server is used as the parent of the first peer in the chain and also a backup source responding for emergency request of peers when their buffer content is almost exhausted. Either the peer parent or the backup server can do video streaming work independently. When both sources are working, the process of streaming is speed-up. To further improve the performance, a parent-child exchange (PCX) routing mechanism is also developed to handle the weak-node problem in the P2P chain. Simulation results show that the PCX scheme can greatly improve the system performance and make efficient use of peers' bandwidth. It is also shown that the backup FEC data support from the server can greatly improve the playback quality at the peers with a reasonable cost. Besides, some QoS mechanisms for the video streaming system are discussed.

In the following of this chapter, the proposed P2P video streaming system is firstly described in section 4.2. Section 4.3 presents the parent-child exchange mechanism which is intended to solve the weak node problem in a P2P chain. Simulation results will be discussed in section 4.4. Some QoS mechanisms for the proposed system are also discussed in this section. A conclusion of the chapter is given in section 4.5.

4.2 A New P2P On-Demand Video Streaming Scheme

4.2.1 P2P forwarding Chain

In a chain structure P2P VoD system, video data is delivered from one peer to another. The peer who uploads data is called a parent, whose child is the peer downloading data from it. At the very beginning when a peer enters the VoD system, it is connected to the end of the P2P chain and becomes the child of a peer who enters earlier. In our system, one parent has only one child. The term "peer" is interchangeable with "node" and "client". The proposed P2P system is shown in Figure 4-1. It is seen that the server containing both data packets and FEC packets of the videos is used as the parent of the first peer in the chain.

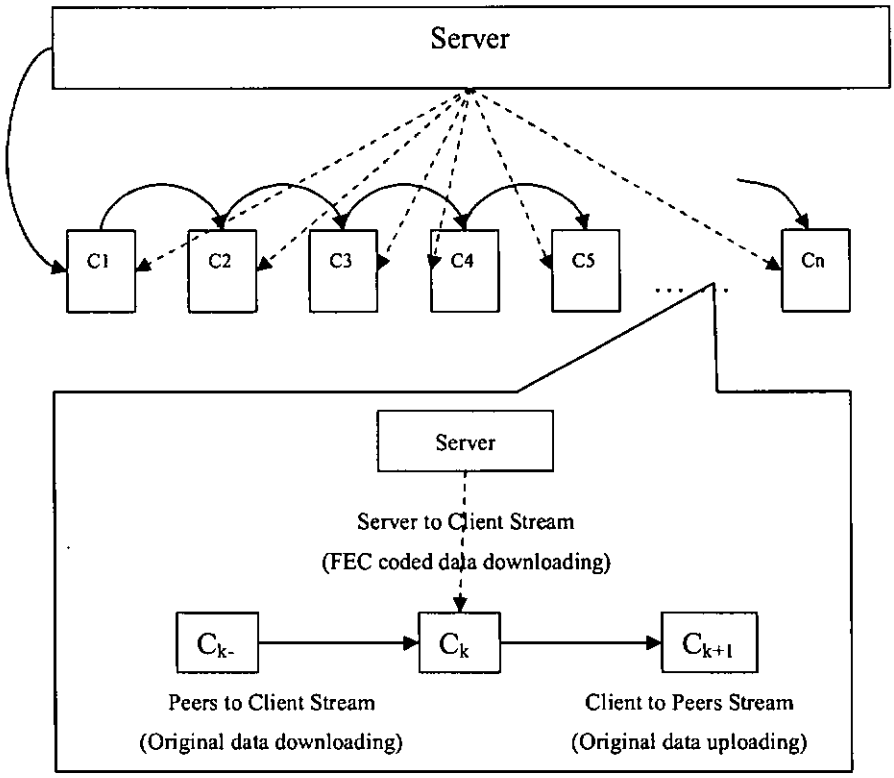


Figure 4-1: The proposed chain structure P2P media streaming system

4.2.2 Standby Server and FEC Stream

In addition to being the first peer's parent, the server in the proposed P2P VoD system mainly acts as a backup media source. When any peer's buffer content is below a threshold denoted as *H*, the server will deliver FEC coded media stream to the peer.

Involving FEC coding in our P2P media stream system is not just for error protection but to avoid packet transmission coordination. When the media server acts as the additional media stream provider to a peer, the peer will receive two streams, one from the server and the other from its parent peer. In this regard, some mechanisms [SHAN2003] are required to control the transmission of content downloaded by the peer from two sources to avoid data duplication. In our proposed scheme, we use a simple way to solve this problem. Since the media server has all sessions of media data, it is easy to generate FEC redundant data derived from the original media data using a FEC coder like Reed-Solomon coding [RILE]. The original data can be FEC encoded in unit of GOF (Group of Frame). For example a GOF contains 10k bytes block and is packed to 10 1k-bytes-size original data packets. These packets are transmitted among peers. Another 10 1k-bytes-size parity data packets is generated in the server and sent to the client when needed. Because of the FEC coding property, the GOF can be recovered by a FEC decoder when any 10 packets out of these 10+10 packets are received by the peer. The client will acknowledge the server and the parent, and

ask them stop sending packets from this GOF and start sending next GOF.

The inspiration of using FEC parity data in server-to-peer transmission is from Hybrid ARQ [SACH2001, MAJU2002], which mixes FEC parity packet with original data packet in one transmission route. For any client who just joins the P2P system, it will receive video stream from its parent peer and the server, because its buffer is empty and definitely lower than the threshold H. This helps reducing the initial playback delay of the user.

The detailed procedure of the backup server operation on FEC streaming to clients is described below. The pre-buffer time of a peer before playback is denoted as K. H is the threshold to invoke server FEC streaming support. T_b is the received video frame timestamp, T_p is the video playback frame timestamp. $T_b - T_p$ is the buffered video content for playback.

1. When a peer just enters the system, it is connected to the end of the P2P chain and becomes the child of a peer who enters earlier. The peer buffer is empty, $T_b = 0 < K$. The peer won't start playback until $T_b \geq K$. The peer receives video stream from the parent and FEC stream from server since $H < T_b - T_p$. When the buffer content is larger than H, the server will stop sending FEC stream and the client receives data only from its parent peer.
2. If a client's buffer content is smaller than the threshold H, i.e. $H < T_b - T_p$, it sends "*need help*" message to the backup server, and tell the server the video

number S_v (the server may concurrently provide several video clips in VoD services) and frame sequence number S_f that it is receiving.

3. After receiving the "*need help*" message form the client, the server may check the client identification. If the client is eligible to get FEC streaming backup support, the server will start to send parity packets for Frame S_f (or Frame S_f+n, considering the delay) of Video S_v.
4. The client receives original data packets of Frame S_f from its parent peer in the P2P chain, and receives parity data packets of Frame S_f from the server. When enough packets for the Frame S_f are received, the clients will acknowledge the server and its peer parent. Afterward, they will proceed to send packets of the next frame S_f+1.
5. The client monitors its buffered content value T_b - T_p and continuously receives FEC stream from the server when T_b - T_p <H. With the FEC stream support, the client's buffered content will increase. When T_b - T_p $\geq H$, the client sends the "*thank you*" message to the server. Then the server stops sending FEC packets.
6. When a peer only receives video stream from its parent, it is assumed that normal RTP/UDP transmission is used and no acknowledgement is required.

4.2.3 Hybrid P2P Architecture

As illustrated in Fig.4-2, the proposed P2P media streaming system combines the traditional client/server approach with a P2P chain structure. The client/server

approach provides backup support to peers who cannot get adequate media data form the P2P chain. In normal situation, peers get all media data from the P2P forwarding chain.

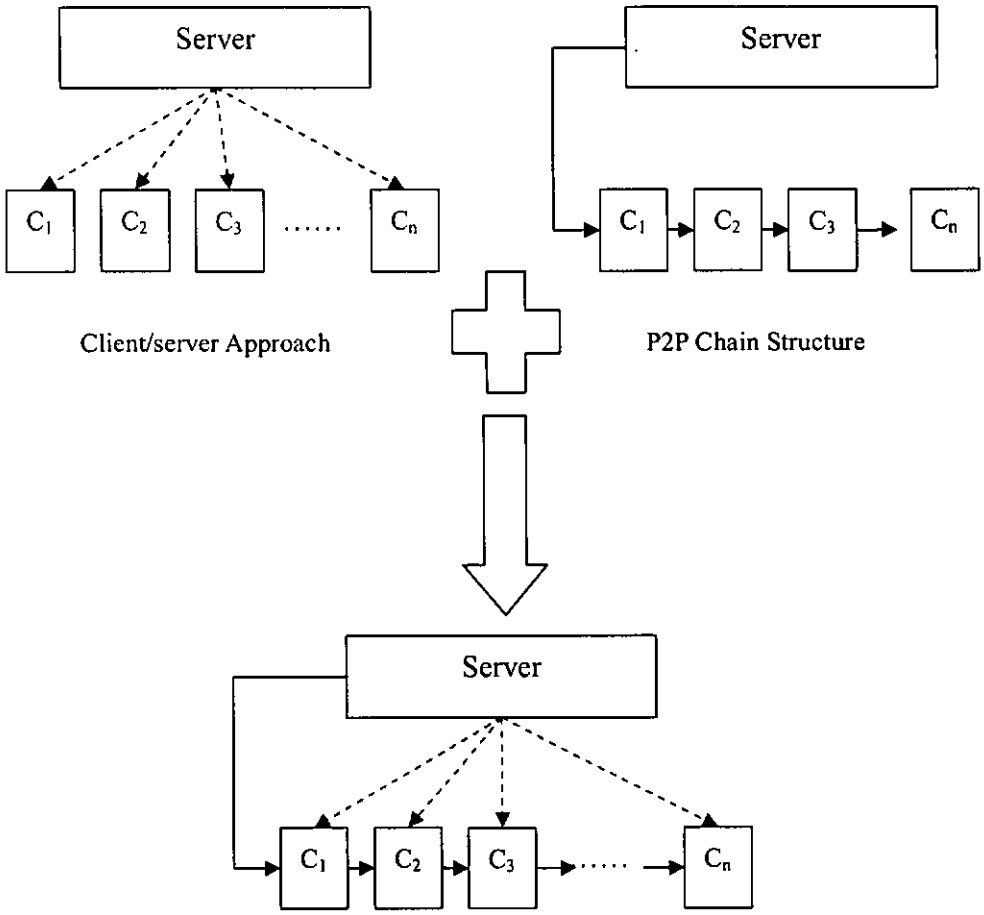


Figure 4-2: Hybrid P2P media streaming architecture

4.2.4 Client Departure

A peer leaves the system either when it has completed what it wants to enjoy or it drops from the chain accidentally. The child whose parent leaves will be redirected. Such a process only involves three peers, and the redirection can be

managed by the server. Figure 4-3 shows the leave process of client C_k. To ensure the continuous playback of C_{k+1}, it may request a FEC stream support from the server when the buffer content is below the threshold *H*.

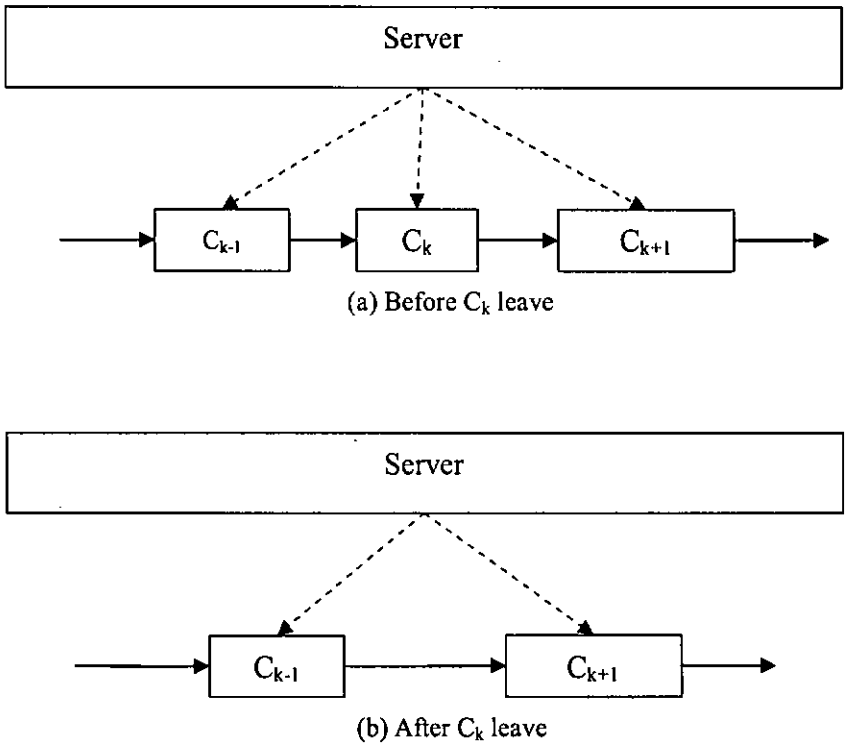


Figure 4-3: Client Departure

4.3 Parent-Child Exchange Routing Mechanism

A chain structure is simple but has the weak node/link problem (a peer with poor network condition). For example, when the download link of peer C_{k-1} gets some problems and it will experience a low download speed. Meanwhile it uploads data to the child peer C_k at a faster speed. Because of the speed difference, sooner

the parent C_{k-1} cannot provide any useful data to its child C_k. After sending out its buffered data, C_{k-1} has to download something from its parent C_{k-2} before it can upload new data to its child C_k. C_k's download speed will be suppressed and its downloading bandwidth is not fully occupied. Besides, such an effect will pass to the following peers like a shockwave movement through a chain. To tackle the problem, we propose a simple but efficient solution. A parent-child exchange (PCX) scheme is proposed, which exchanges the parent and child position in the P2P chain as the name implies. PCX gives priority to peers with better connection. If the child has a better network connection than its parent, it can exchange the position with the parent. When the child has got all its parent has and cannot do full speed download from its parent, PCX is invoked and the parent and child positions are exchanged.

To simulate a real situation, all parts of the P2P chain is set to unstable, and we assume that one peer has an unstable download link and a stable upload link. One part of P2P chain includes a peer's upload link and its child's download link. The parent's stable upload link and the child's unstable download link combine together and make the P2P chain unstable. Without setting both download and upload link unstable, we get an unstable P2P chain in an easier way.

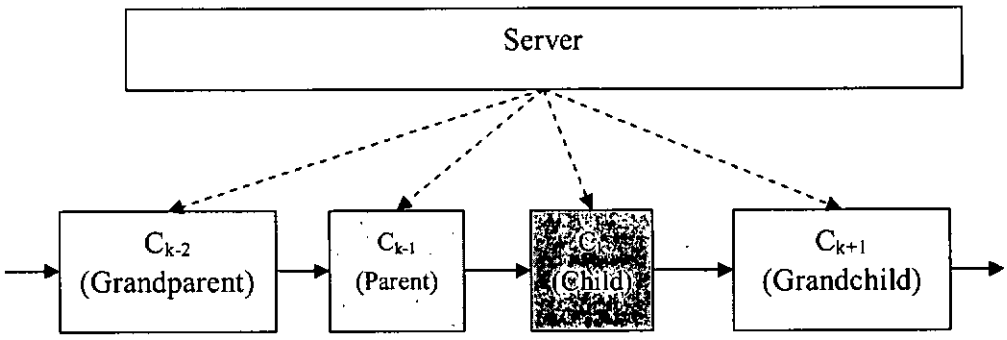
The assumed upload and download bandwidth of a peer is described below:

1) The download bandwidth varies from time to time. The download

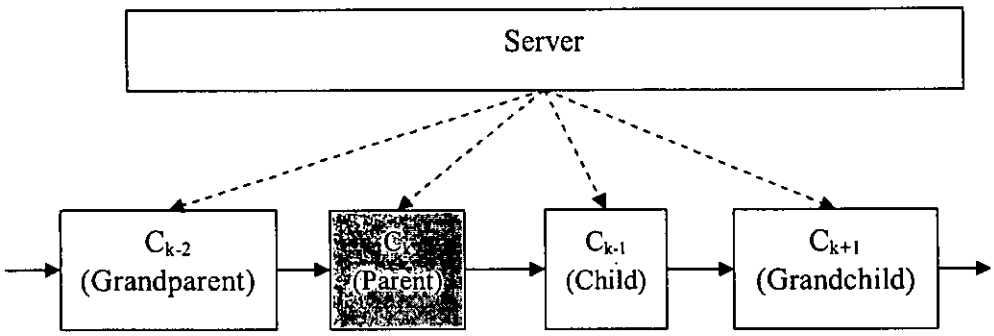
rate is limited by the download bandwidth.

2) The upload bandwidth is not strictly limited, and it is assumed that full speed download of its child can always be supported. A peer's upload rate is always limited by its child's download bandwidth.

When the parent's download rate cannot support the full speed download of the child, the child is probably in a better network condition than the parent. So the child can firstly download data from its grandparent with faster speed and then upload the data to the parent whose download bandwidth is smaller.



(a) Routes between peers before rerouting



(b) Routes between peers after rerouting

Figure 4-4: PCX routing mechanism

One example is illustrated in Figure 4-4. Because C_{k-1}'s download speed is constantly lower than its upload speed, at some moment, C_{k-1} will send out all it has to its child C_k. According to the PCX scheme, C_{k-1} will acknowledge C_k and C_{k-2} (grandparent) that it is out of buffer and prepares to arrange a reroute process. C_k will receive media stream from C_{k-2} instead of C_{k-1} and C_{k-1} receives media stream from C_k (its original child) instead of C_{k-2}. After receiving the acknowledgement from C_{k-1}, C_k acknowledges C_{k+1} (grandchild) for rerouting. The parent peer of C_{k+1} becomes C_{k-1}. The information required for reroute, such as the new parent and child IP address and connection port, is exchanged among these clients. Before rerouting, the P2P chain is shown in Fig. 4-4a. After rerouting, as illustrated in Fig. 4-4b, the position of C_k and C_{k-1} in the chain is exchanged.

The detailed procedures of a PCX reroute operation are listed below:

1) The original route is C_{k-2} to C_{k-1} to C_k to C_{k+1}. C_{k-1} sends out its buffer content to its child C_k.
2) C_{k-1} acknowledges C_{k-2} and C_k for rerouting. After C_k receives an acknowledgement from C_{k-1}, it acknowledges C_{k+1} rerouting.
3) The original media stream is cut, and a new route is established as C_{k-2} to C_k to C_{k-1} to C_{k+1}.

Whenever a peer uploads all its content to its child, it will be downgraded to its child's child. The original child peer with a better network condition will be

upgraded to its parent's parent to download with its full download bandwidth. Each time in such a reroute process, only three routes have been rerouted and four peers are involved, so the delay caused by rerouting is very small. The whole reroute process can be without awareness of server.

4.4 Simulations

4.4.1 Simulation methodology

Computer simulations are performed in this section to evaluate the performance of the proposed P2P VoD system. In our simulations, all videos in the server are of 2-hour-length with a constant bit-rate (CBR) of 250kbps. All simulations are run for 10 hours. The client arrival pattern is assumed to be Poisson. The average client interarrival time is denoted as T. The peer-to-peer bandwidth is simulated as a set of random data with an average value B_{P2P}. Each P2P bandwidth curve is bounded by $B_{P2P} \times (1 \pm \alpha)$, and α is a weighting factor ranging from 0.1 to 0.5 (assumed uniformly distributed). The server-to-peer bandwidth is set to a constant B_{S2P}. We assume that such a constant bandwidth between server and peer is reserved to improve client playback continuity. That will be easier to compare with the bandwidth consumption in a traditional server-client VoD service. T, B_{P2P} and B_{S2P} are variables to test the system performance under different conditions. The early departure rate of the client holds to 10%. The buffer threshold H and client pre-buffer time before playback K are both set to 30 seconds.

4.4.2 Simulation Results

4.4.2.1 Standby Server Performance

Firstly we test the system performance of a normal chain structured P2P video streaming system without PCX scheme. Simulation results are shown in Figure 4-5 and Figure 4-6.

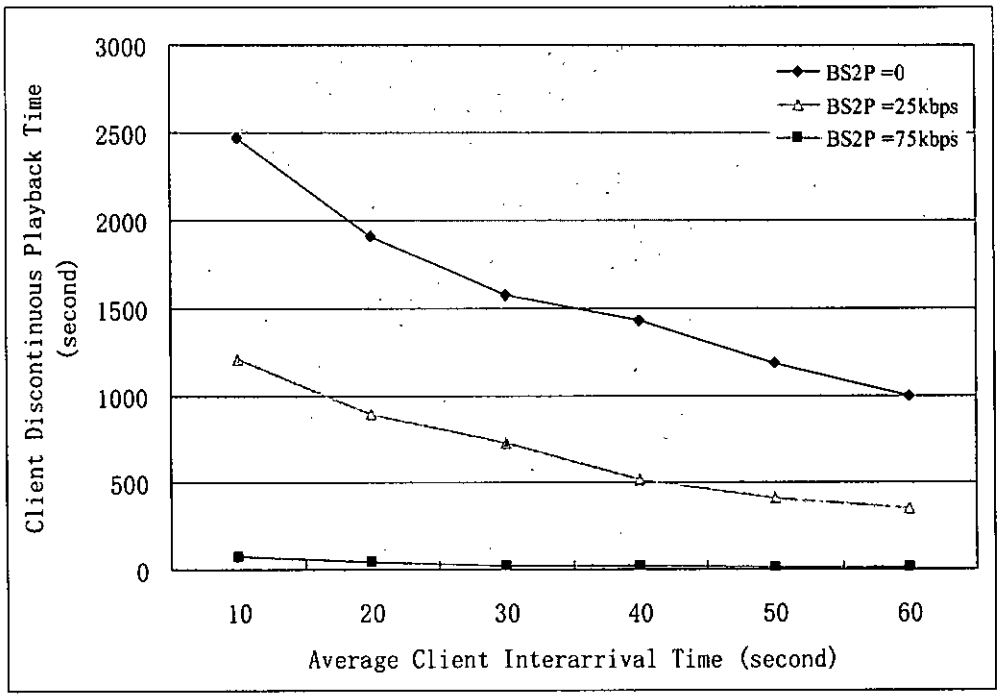


Figure 4-5: Client discontinuous playback time
(B_{P2P}=250kbps, B_{S2P}=0~75kbps, No PCX)

Figure 4-5 shows three curves of client discontinuous playback time with different server-to-peer connection bandwidth. The discontinuous playback time is measured by summing up all the stalled time due to re-buffering at the client. When B_{S2P} =0, i.e. no FEC server support, clients experience a long discontinuous playback. In such a case, clients have experienced about 1000 to 2500 seconds stalled time during the playback, which is quite annoying to the users. When B_{S2P}=25kpbs, i.e. a FEC server support with an average bandwidth

of 10% of video bandwidth, the client discontinuous playback time is much reduced, as indicated by the middle curve. When B_{S2P} =75kpbs, i.e. a FEC server support with an average bandwidth of 30% of video bandwidth, clients seldom experience display freeze and wait for re-buffer, as the lower curve shows. Actually from the simulation results, when $B_{S2P} \geq$125kpbs, i.e. 50% of video bandwidth, there won't be any discontinuous playback at the client.

To reduce the discontinuity suffered by peers, we provide the FEC standby server support to the peer whose buffer content is under the threshold H. To measure the bandwidth required by the server to provide FEC support to clients/peers, a parameter called normalized server burden B is defined as:

$$B = \frac{F}{V \times N}, \qquad (4\text{-}1)$$

where F denotes the average server bandwidth occupied by server-to-peer FEC streaming, which is the sum of all server-to-peer streams. V is video bandwidth (set to 250kbps in the simulations), and N is the average number of the clients in the system. In a traditional server-client VoD system, the occupied server bandwidth is $V \times N$. Server burden is the ratio of server bandwidth consumption in the proposed VoD system over that in a traditional VoD system. A lower server burden value means a larger bandwidth saving by the new approach.

If the standby server can provide constant 250kbps FEC codes video stream to peers when their buffer content is smaller than the threshold H, peers won't experience discontinuity. In Figure 4-6, the server burden varies from 0.052 to 0.037, which indicates that even without PCX, the proposed P2P VoD

architecture saves over 95% server bandwidth than a traditional client/server approach VoD system.

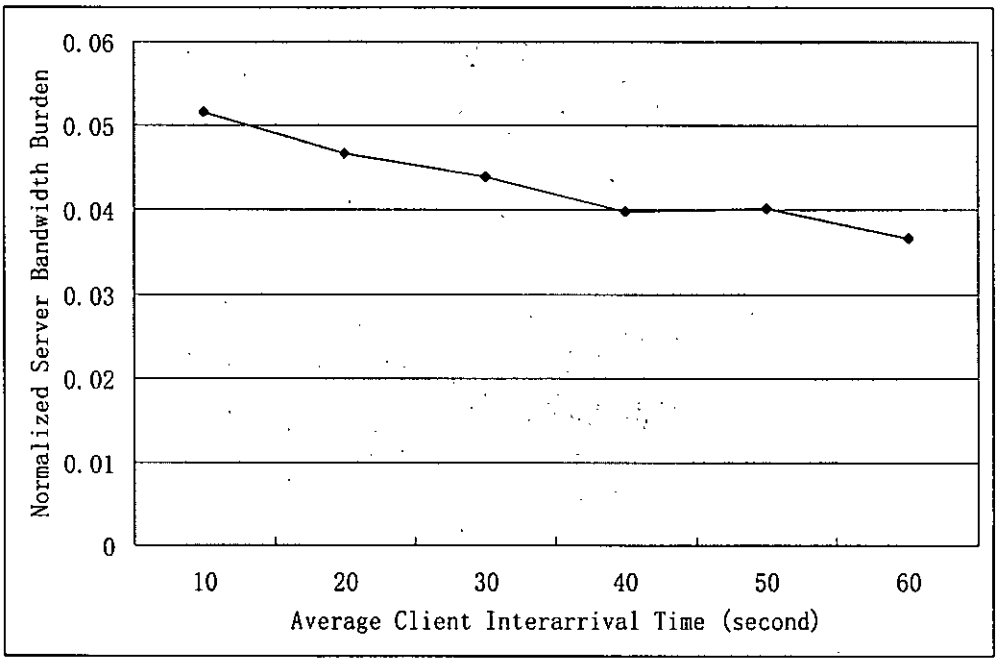


Figure 4-6: Standby server bandwidth burden
(B_{P2P}=250kbps, B_{S2P}=250kbps, No PCX)

4.4.2.2 PCX routing mechanism performance

We then evaluate the performance of the proposed PCX routing mechanism in the proposed architecture. The normalized server burden for the proposed system with and without the PCX method is shown in Figure 4-7. It is seen that a lower server bandwidth loading is resulted when we apply the PCX method in the system. The server burden for the system with PCX varies between 2.6% to 2.9% and the curve does not change much with different client interarrival time. The results reveal that the proposed PCX method can greatly enhance the system performance.

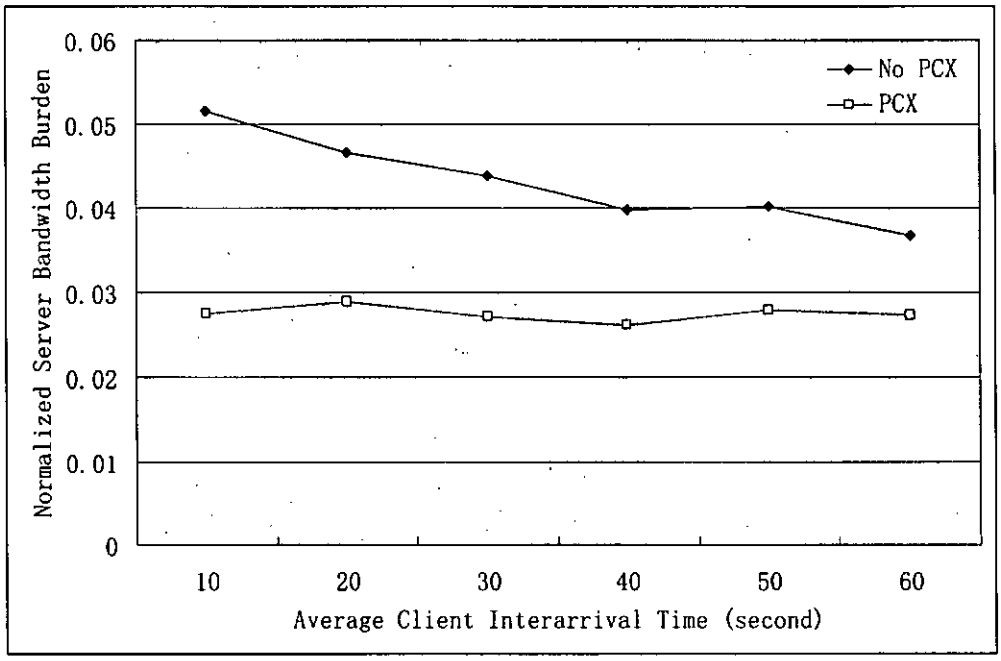


Figure 4-7: Standby server bandwidth burden
(B_{P2P} =250kbps, B_{S2P} =250kbps, PCX Applies)

4.4.2.3. System Performance with Variable B_{P2P} and B_{S2P}

In this part, we are going to see how peer-to-peer bandwidth B_{P2P} and server-to-peer bandwidth B_{S2P} affect the system performance. Simulation results are shown in Figure 4-8 and Figure 4-9. In the simulations, the client interarrival time is fixed to 20 seconds.

If the connection between the standby server and all peers is stable and B_{S2P} constantly remains at 250kbps, full length continuous playback at the peer side can be achieved. As shown in Figure 4-8, the server bandwidth burden increases while B_{P2P} decreases. In the figure, all bandwidth values are normalized by the video bandwidth, i.e. 250kbps. When $B_{P2P} < 1$, peer-to-peer streaming is not enough to provide the continuous playback, the standby server has to stream FEC data to peers from time to time. When B_{P2P} decreases, the server burden will increase rapidly. When $B_{P2P} > 1$, continuous playback is maintained at most time

and the server seldom works.

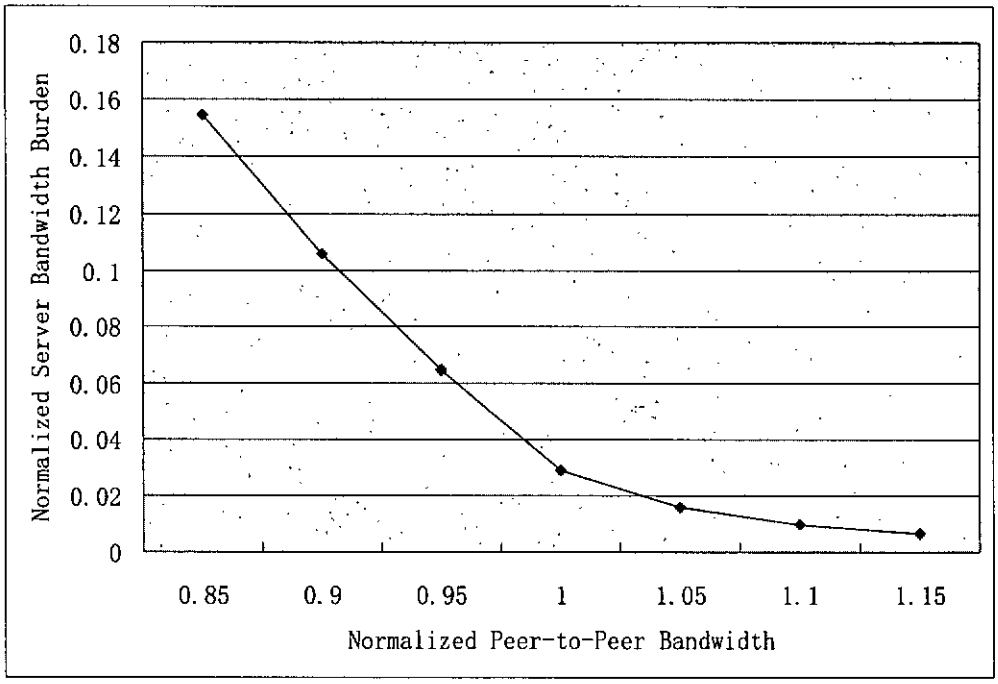


Figure 4-8: Server bandwidth burden under different peer-to-peer bandwidth (B_{S2P} =250kbps, T=20s, PCX Applied)

Figure 4-9 illustrates the system performance when the server-to-peer bandwidth varies. The server bandwidth burden curve and client discontinuous playback time curve are shown in the figure with B_{S2P} varies from 0 to 1. The bandwidth burden curve keeps at a certain level and the client discontinuous playback time remains 0 when $B_{S2P} \geq 0.4$. B_{S2P}=0.4 is a critical point for both curves. It is seen that the client will experience more discontinuous playback when B_{S2P} <0.4, while the FEC server has a sharp decrease on bandwidth burden. From Figure 4-9, we can see that the optimal point is at around B_{S2P}=0.7(175kbps), which is the lowest server burden point for no discontinuous playback. We can see that when the peer-to-peer connection is good (B_{P2P}=250kbps), the full speed FEC stream support is not necessary to provide a smooth playback to clients.

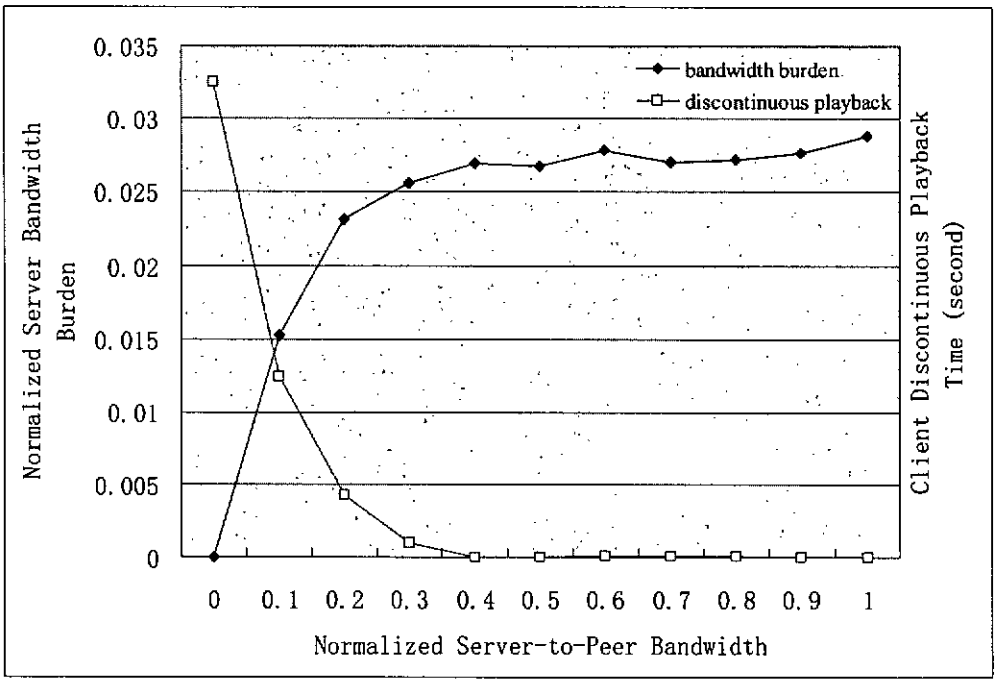


Figure 4-9: system performance under different server-to-peer bandwidth (B_{P2P} =250kbps, T=20s, PCX Applied)

4.4.2.4 Server FEC Sending Threshold

In this section, we are going to study the effect of the FEC sending threshold H on the system performance. In our study, we fix B_{S2P} =50kbps and test the system performance for different threshold H and B_{P2P}. Simulation results are illustrated in Figure 4-10 and 4-11.

From the figures, we see that a larger H will give smaller discontinuity to clients in general. When H is set to a higher value, the chance for a client to request the FEC support from the server will be higher. Then the client will get a better support from the server and hence continuous playback can be maintained in a longer time. On the other hand, the server burden will become higher. It is seen from the figure, when H increases from 10 to 30, the playback discontinuity is

greatly reduced. When H reaches 30s, the playback discontinuous time reduces slowly as H further increases. For the server burden, it is seen in Figure 4-11 that a larger H gives a higher server bandwidth burden. When H increases from 10 to 30, the server burden increases very quickly. It is noted from the results in Fig.4-10 and Fig.4-11, the system achieves a good balance between server bandwidth consumption and clients' discontinuity is achieved when we set H to 30.

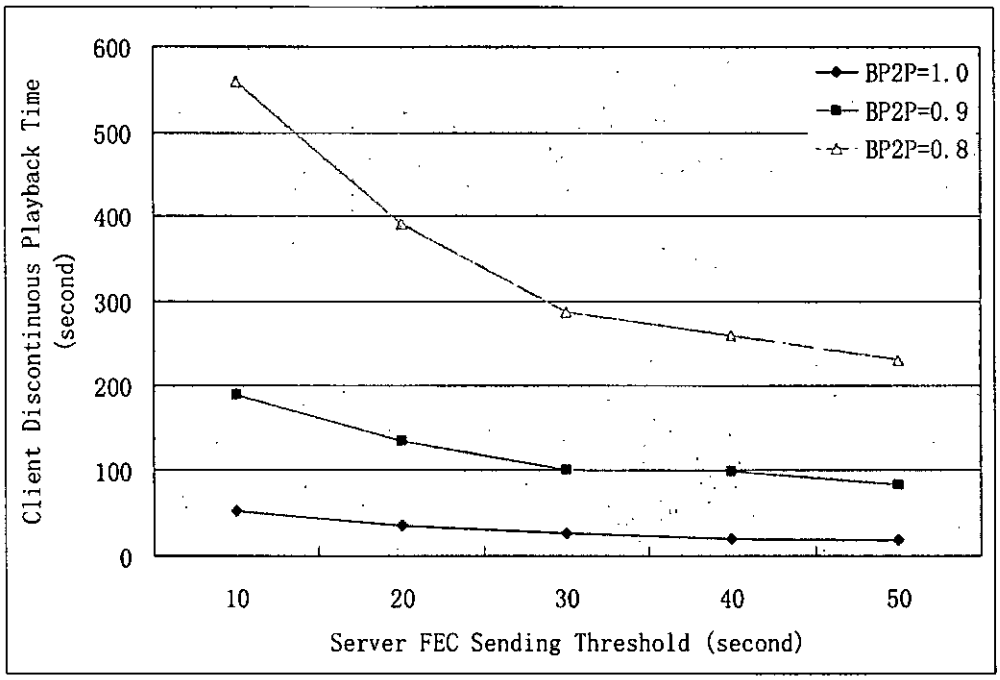


Figure 4-10: Client discontinuity with different server FEC sending threshold H (B_{S2P} =50kbps, T=20s, PCX Applied)

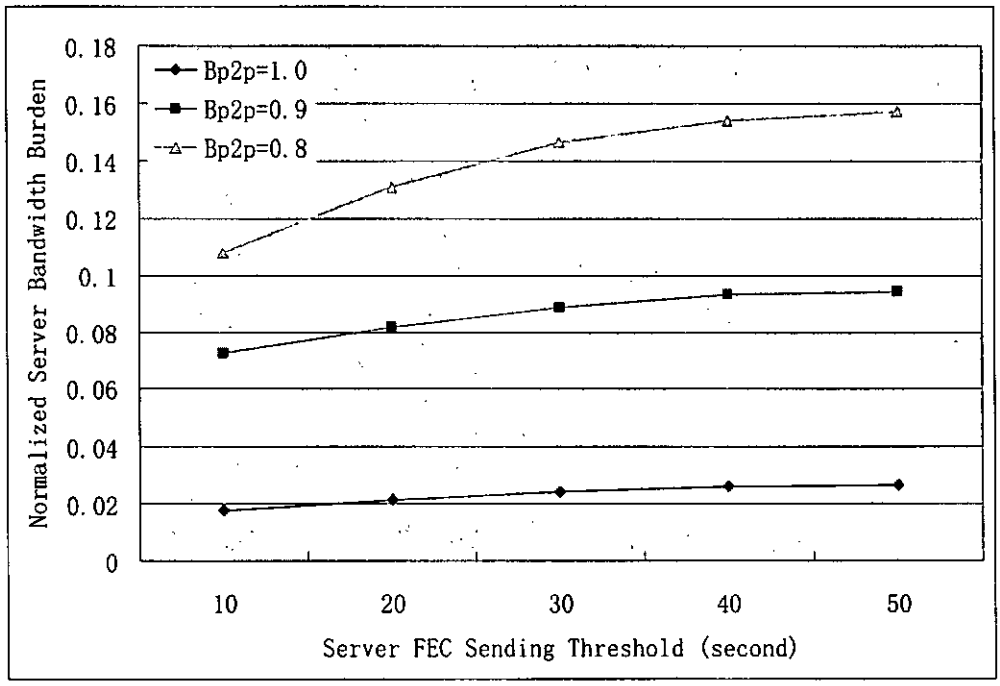


Figure 4-11: Server burden with different server FEC sending threshold H (B_{S2P} =50kbps, T=20s, PCX Applied)

4.4.2.5. PCX Shift

We examine in this section the number of PCX shifts for situations under different B_{P2P} and B_{S2P}. In the study, we record the PCX shift number of peers who completely receive the video in the simulation time. Tables 4-1 and 4-2 summarize the number of downlink shift as the unqualified parent in the PCX scheme for different B_{P2P} and B_{S2P}.

B_{S2P}	1.3	1.2	1.1	1	0.9	0.8	0.7
PCX Shift	17.08	17.1	16.35	16.57	16.32	16	15.89

Table 4-1: PCX shift with different B_{S2P} (B_{P2P} = 1)

As shown in Table 4-1, when B_{P2P} = 1, the P2P chain is enough to support normal streaming operation of the system, and the variation of B_{S2P} cannot affect

PCX shift a lot. When B_{S2P} decreases, the frequency of PCX only decreases slightly.

B_{P2P}	1.5	1.4	1.3	1.2	1.1	1	0.9	0.8
PCX shift	12.02	13.25	14.49	16.04	17.10	16.57	8.75	2.18

Table 4-2 PCX shift with different B_{P2P} (B_{S2P} = 1)

As Table 4-2 shown, the variation of B_{P2P} does affect PCX shift a lot. When B_{P2P} is less than 1, the number of PCX shift decreases sharply. When B_{P2P} is less than the video bandwidth, the backup support from the FEC server becomes more frequent, and that makes the incoming stream bandwidth (B_{P2P} + B_{S2P}) for a peer more stable. Imaging that when B_{P2P} =0, the server FEC stream becomes the only source of a peer, no PCX will be trigged. In Table 4-2, when B_{P2P} decreases from 1.5 to 1, the PCX number increases smoothly. That may be because the parent peer can pre-buffer a little more before its child enters the system.

4.4.3 QoS Mechanisms for the Proposed System

Simulation results in the above sections show that the system performance depends much on B_{P2P} and B_{S2P}. In this regard, we propose three quality of service (QoS) mechanisms to differentiate the services provided to different users depending on the allowed B_{P2P} and B_{S2P}.

4.4.3.1 QoS Mechanisms

In our architecture, the bandwidth on the P2P chain is generally unpredictable.

The only link we can control is the server-to-peer link. By providing different levels on the server-to-peer connection speed, we can provide different quality of service levels. Table 4-3 shows three classes of users, where they get different QoS levels according to their server-to-peer bandwidth.

	Total	First class	Second class	Third class
Peer Number	956	197(20.6%)	574(60.0%)	185 (19.4%)
Normalized B_{S2P}		1	0.4	0
Normalized B_{P2P}		1	1	1
Discontinuous playback number	0	0	0	6
Discontinuous playback time	55	0	6	270

Table 4-3 QoS and Different B_{S2P}

With different B_{S2P}, the users get different discontinuous playback time. In around 1000 peers, 20% of them are belong to the first class, and they receive full speed FEC stream support (B_{S2P} = B_{video}) from the server. Hence they get continuous playback all the time as shown in the table. 60% peers are belonging to the second class, and they get 0.4 B_{video} and around 6 seconds discontinuous playback time only. The left 20% clients are the third class users. They receive no server-to-peer FEC stream, and experience long time discontinuous playback.

4.4.3.2 Charging for Commercial VoD system

We propose 3 charging mechanisms for a commercial VoD system based on our architecture.

1. Charging based on Server-to-Peer FEC stream bandwidth.

That means different rank of clients would receive FEC coded stream from the server in different speed.

- Class A clients, who pay a high price, get the full speed FEC stream. With the high-speed FEC stream from the server, Class A clients definitely get the highest quality for the video.
- Class B clients, who pay a middle price, get middle speed FEC stream support. They may suffer some short time discontinuous playback.
- Class C clients, who pay a low price, get low speed FEC stream support or no FEC stream. They are the lowest rank, and probably get frequent discontinuous playback.

2. Charging based on Server-to-Peer FEC stream data amount.

The user of the VoD system can be charged on Server-to-Peer FEC stream data amount. Because in the first charging mechanism, sometimes the server-to-peer FEC stream may not fully occupy the provided B_{S2P}, and some bandwidth is unused but charged. Hence, if the system charges the users based on the data amount transmitted from server, it may be a little bit fair than the first one.

3. Combination of the first and the second method

Charging based on both the server-to-peer FEC stream bandwidth and data amount maybe more appropriate. Different B_{S2P} will consume different server

resources. Therefore, charging on the data amount is fairer. So we can set different B_{S2P} for different classes of users and charge them on FEC data amount.

4.5 Chapter Conclusions

In this paper, we present a chain structured peer-to-peer media streaming scheme for stored video, in which a FEC server is maintained as a backup data source. The standby server acts as a backup parent when a peer cannot receive data form its assigned parent normally. It also helps to prevent the cascading effect when a peer fails. A parent-child exchange (PCX) routing mechanism is also proposed in the system to adjust the P2P chain. Simulation results show that PCX greatly reduces the effect a peer with poor network conditions to the P2P chain and the client playback continuity is improved by the backup support from the FEC server with little cost. Besides, some QoS and charging mechanisms for such an on-demand streaming system are discussed. By controlling the server-to-peer connection, different quality of service levels can be provided.

Chapter 5

A Parent Search Method for Peer-to-Peer Media Streaming Networks

5.1 Introduction

The P2P streaming system proposed in Chapter 4 is designed mainly to support on demand video streaming, in which all videos are encoded and stored in the server. As a chain structure is used in the system, the long server-to-peer path delay introduced by the P2P chain is not generally acceptable for real-time media streaming services such as live news report and football match live broadcast. In this regard, a tree structured P2P media streaming system maybe more suitable for real-time content streaming.

For a tree structured P2P media streaming system, construction of the network spanning tree that connects all clients/peers together in the network is one of the most important tasks. To construct a network spanning tree, one should consider node traffic, node to node distance, path capacity and other criteria, and should achieve a good balance on all these elements. In a P2P network, when a new peer arrives, it probably cannot get media stream directly from the media server/seed. To receive the media data, the newly arrived peer has to locate a suitable parent

peer in the tree based on some criteria. One simple solution is to assign the previous peer just enters the system as the parent for the new peer. However, the physical distance of them maybe far away. Therefore, it is more natural to use the distance as a measure for a peer to select its parent. With this criterion, a new peer will choose the closest one as its parent. To find an optimal solution, one has to search all the nodes in the network, which of course is very time consuming.

To solve the problem, some people proposed to use the cluster-based hierarchical search [BANE2004, XIAN2004] to locate the parent for a newly arrived peer. In [BANE2004], Banerjee et al. suggested to group nearby peers into clusters and the clusters are then connected in a tree structure. Starting from the highest hierarchical level, the newly arrived peer measures the distance to the cluster leaders and chooses the closest cluster to continue searching in clusters at a lower level until it finds the shortest distance cluster and connected to the closest neighbor. Xiang et al. [XIAN2004] adopts a similar way to do parent searching. However, both schemes didn't consider the node to root distance so that the distance measurement to cluster leaders cannot represent the distance to clusters. Such a search is easily misled by the local minima. The hierarchical parent search is fast but it is hard to locate the best parent to a peer because of the local minimum problem. In this chapter, we use the server-to-peer path delay (S2PPD) as the criterion for finding the parent location for a peer and use the Prim-Dijkstra tree to construct the network spanning tree. With the use of

network triangle inequality [HOTZ1996, GUYT1995], we develop a fast parent search method to locate the best parent for the newly arrived peer.

In the following of this chapter, the Prim-Dijkstra algorithm which is used to construct the network spanning tree is introduced in Section 5.2. The proposed fast parent searching method using the network triangle inequality is described in Section 5.3. Simulations are performed to test the performance of the proposed method and the results are presented in section 5.4. Finally, some concluding remarks are given in section 5.5

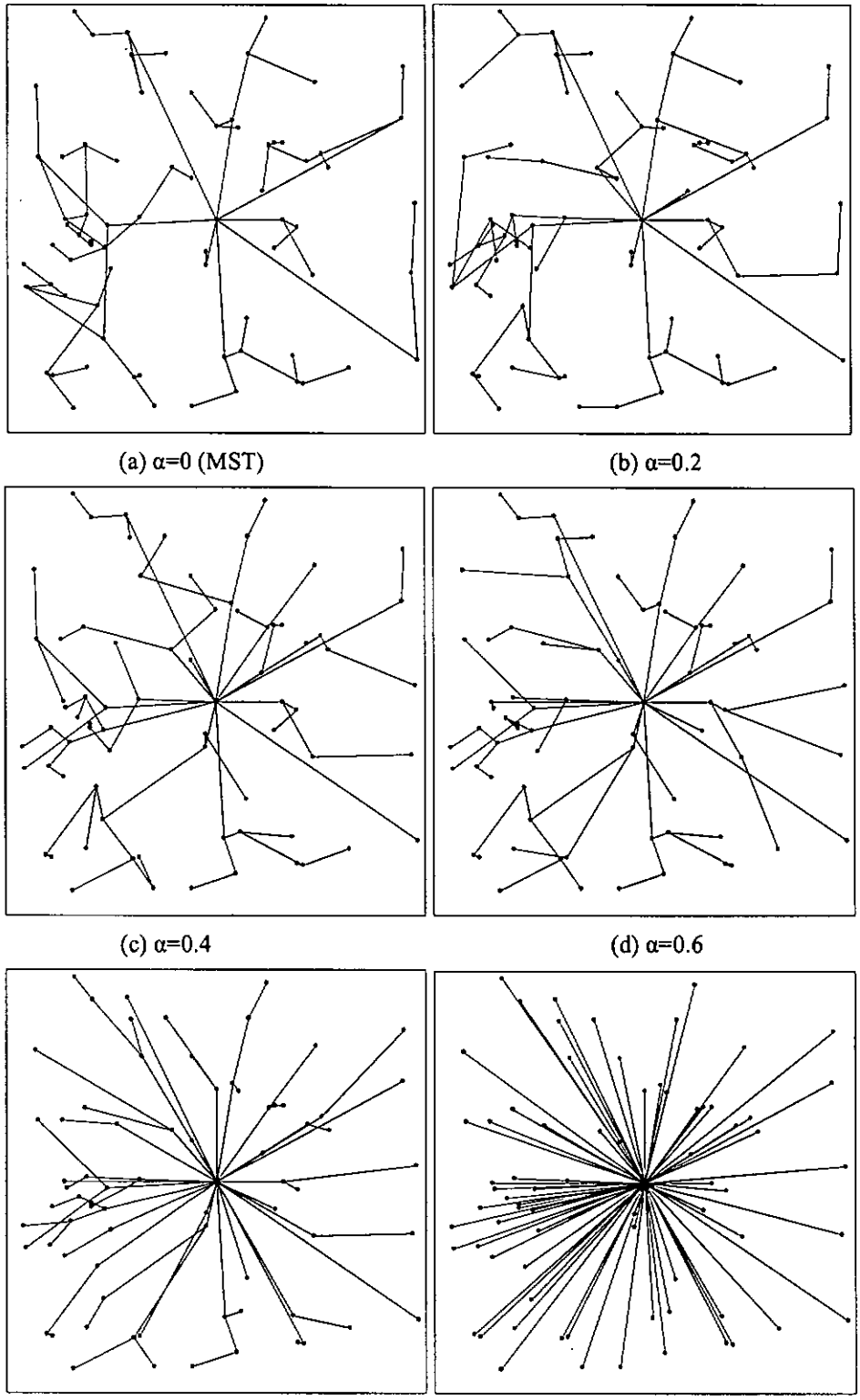
5.2 Prim-Dijkstra algorithm in P2P Networks

We have proposed a chain structured P2P system for on-demand media streaming services [ZHAN2004] in Chapter 4, which is only suitable for downloading mode video streaming that can tolerate a large server-to-peer path delay (S2PPD) generated by a long P2P chain. In the P2P chain proposed in last chapter, as the one parent one child principle is applied, the burden to peers is minimized. In a tree structured P2P network, a parent can have more than one child and thus the S2PPD is greatly reduced. A tree structured network is suitable for live media streaming/broadcasting, which may not require the best video/audio quality but a small delay is important.

Clustering is an efficient method to construct a hierarchical P2P tree. In such an approach, nearby peers form a multicast group and share/forward data within the group. Due to limited deployment in network layer multicast, most of previous works use application layer multicast. In the cluster-based hierarchical approach [BANE2004, TRAN2004], the new peer joins the closest cluster, and the server-to-peer path delay (S2PPD) is not considered. S2PPD measures the total end-to-end delay time for data packet transferring from server to peers.

S2PPD is important for P2P media streaming. To minimize S2PPD, the Prim-Dijkstra tree in graph theory may give a better solution to interconnect peers to the media server. In the Internet, the shortest path tree rooted on a server with a number of end nodes is simple, which is directly connecting from the server to the ends, as illustrated in Figure 5-1. It obviously disobeys the intention of media streaming in a P2P network. So we try applying the Prim-Dijkstra [ALPE1995, CAHN1998] algorithm to construct the network spanning tree for a P2P network.

The Prim-Dijkstra algorithm builds a tree that interpolates between a Minimum Spanning Tree (MST) and a Shortest Path Tree (SPT). The Prim-Dijkstra method considers not only the connection between a node and its neighbor but also the path distance from the root/center to the node.



(a) α=0 (MST) (b) α=0.2

(c) α=0.4 (d) α=0.6

(e) α=0.8 (f) α=1 (SPT)

Figure 5-1: Network spanning tree with different α

A weighing parameter is employed. The Prim-Dijkstra algorithm is labeled:

$$\min_{neighbors} [\alpha \times dist(root, neighbor) + dist(neighbor, node)]$$

where $0 \le \alpha \le 1$ and is used to parameterize the algorithm. When α is 0, we build the minimum spanning tree as shown in Figure 5-1a. When α is 1, we build the shortest path tree (Figure 5-1f) from root.

Using the above algorithm in a P2P network, we are searching for the newly arrived peer's parent who can give the minimum value of the expression below:

$$D(j,S) \times \alpha + D(i,j) \qquad j \in P$$

where

i: the new arrival peer

j: the peer who enters the system earlier than the new comer.

P: peers who has child vacancy, (they are the potential parents of the new comer.)

S: the server.

D(j,S): path delay from nodes *j* and server *S*.

D(i,j): network distance/delay between nodes *i* and *j*.

Later on, in the simulation section we will show that the Prim-Dijkstra algorithm can give a better result in server-to-peer path delay.

5.3 Fast Parent Search in P2P network

In a computer network, most of the nodes are not directly connected. Routers,

gateways, switches are used to interconnect end nodes. The round trip time between two nodes mainly depends on the delay incurred in packet queuing delay on routers through the route between the nodes.

5.3.1 Network Triangle Inequality

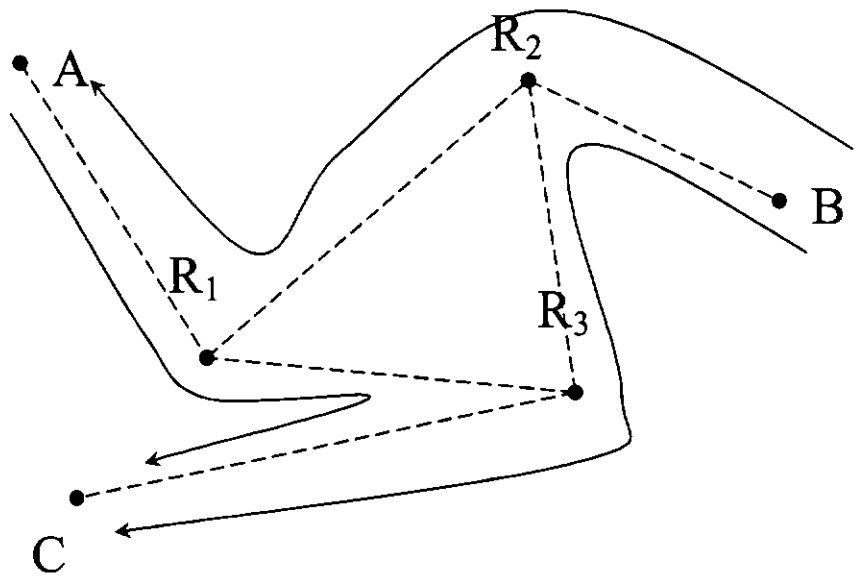


Figure 5-2: Network Triangle

In Figure 5-2, we can see that nodes A, B and C are connected through routers R_1, R_2 and R_3 respectively. These three routes may be interconnected either directly or indirectly through other routers. Assume that $A \to R_1 \to R_3 \to C$ is the fastest route connecting A to C with delay $D(A,C) = D(A,R_1) + D(R_1,R_3) + D(R_3,C)$, $B \to R_2 \to R_3 \to C$ is the fastest route connecting B to C with delay $D(B,C) = D(B,R_2) + D(R_2,R_3) + D(R_3,C)$ and $B \to R_2 \to R_1 \to A$ is the fastest route connecting A to B with delay $D(A,B) = D(A,R_1) + D(R_1,R_2) + D(R_2,B)$.

We can easily prove that:

$$D(A,C) \le D(A,B) + D(B,C) \qquad (5-1)$$

Because $A \to R_1 \to R_3 \to C$ is the fastest route connecting A to C, and the route delay of $A \to R_1 \to R_2 \to R_3 \to C$ should be larger than $D(A,C)$. In the same way, we can get:

$$D(A,B) \le D(A,C) + D(B,C) \qquad (5-2)$$

and

$$D(B,C) \le D(A,B) + D(B,C) \qquad (5-3).$$

By (5-1), (5-2) and (5-3) we can derive that

$$|D(A,B) - D(B,C)| \le D(A,C) \le D(A,B) + D(B,C) \quad (5-4)$$

Such a network triangle inequality [HOTZ1996, GUYT1995] exists whenever these nodes are inter-connectable.

5.3.2 Network Triangle Inequality in P2P Network

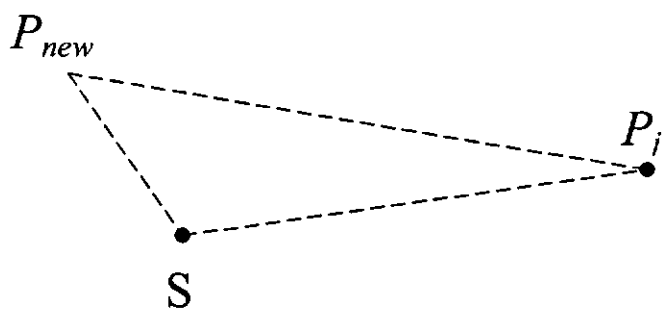


Figure 5-3: P2P Network Triangle

In Figure 5-3, P_j is a childfree peer who has already connected to the P2P tree. S is the media server in the system and knows $dist(S, P_j)$ and $PathDelay\ (P_j, S)$, where $dist(S, P_j)$ is the direct connection delay between S and P_j without any intermediate peer, $PathDealy\ (P_j, S)$ is the path delay of $P_j \rightarrow S$ which has already been fixed. The $P_j \rightarrow S$ route may involve other peers. $dist(S, P_j)$ and $PathDelay\ (P_j, S)$ were obtained by S when P_j entered the system, and $PathDelay(P_j, S) \geq dist(S, P_j)$.

Assume that P_{new} is a newly arrived peer. When P_{new} enters the P2P system, it firstly contacts S. $dist(S, P_j)$ and $PathDealy(P_j, S)$ can be obtained from the database of S. $dist(S, P_{new})$ can be calculated in the contact process.

Using the network triangle inequality described in previous section, without contacting P_j, we know:

$$dist(S, P_j) + dist(S, P_{new}) \geq dist(P_j, P_{new}) \geq \left| dist(S, P_j) - dist(S, P_{new}) \right|$$

So we have:

$$Min\,[dist\,(P_j, P_{new})] = \left| dist\,(S, P_j) - dist\,(S, P_{new}) \right|$$

and

$$Max[dist(P_j, P_{new})] = dist(S, P_j) + dist(S, P_{new})$$

Based on the Prim-Dijkstra algorithm introduced in Section before, we choose $0 \le \alpha \le 1$. Then we have:

$$PathDelay_{P_j}(P_{new}, S) = dist(P_j, P_{new}) + \alpha \times PathDelay(P_j, S)$$

where $PathDelay_{P_j}(P_{new}, S)$ is the $P_{new} \rightarrow S$ route delay through peer P_j.

Therefore, we have:

$$Min[PathDelay_{P_j}(P_{new}, S)] = \left|dist(S, P_j) - dist(S, P_{new})\right| + \alpha \times PathDelay(P_j, S)$$

and

$$Max[PathDelay_{P_j}(P_{new}, S)] = dist(S, P_j) + dist(S, P_{new}) + \alpha \times PathDelay(P_j, S)$$

Up to this step, the path delay range of $P_{new} \rightarrow S$ through peer P_j is obtained. By the same method, the path delay range of $P_{new} \rightarrow S$ through all other peers can be obtained as well. If there are n potential parent peers, we denote $PathDelay_{P_j}(P_{new}, S)$ as $PathDelay_{P_j}$, the following upper and lower bound for $PathDelay(P_{new}, S)$ can be obtained:

$$Min(PathDelay\ ) = Min\{Min[PathDelay_{P_1}], Min[PathDelay_{P_2}], \cdots, Min[PathDelay_{P_n}]\} \tag{5-5}$$

and

$$Max(PathDelay\ ) = Min\{Max[PathDelay_{P_1}], Max[PathDelay_{P_2}], \cdots, Max[PathDelay_{P_n}]\} \tag{5-6}$$

P_{new} can now search for the best parent in a reduced range. Only peers through

which $PathDelay(P_{new}, S)$ possibly falls between the values obtained by (5-5) and (5-6) are added to the contact list of P_{new}.

The server will send the P_{new} information about peers in the contact list, such as their network address, the maximum and minimum value of the server-to-peer path delay through each of them. The new peer will contact those peers one by one. Each time when P_{new} contacts a potential parent, it will update the $Min(PathDelay)$ value. Any non-contact peer through which the minimum server-to-P_{new} path delay is larger than the updated $Min(PathDelay)$ will be deleted in the contact list of P_{new}. When all potential peers in the list are contacted, P_{new} can finally obtain the best suitable peer parent.

5.4 Simulations

5.4.1 Simulation Methodology

To simulate the nodes/peers involved in the P2P network, we generated random nodes in a square on a 2-Dimensional Euclidean space. The square edge length is set to 200. The server is located in the center of square. After one node is generated in the square space, we conduct the parent searching and assign it the best peer according to different methods. In the simulation we always limit one

parent peer can support maximally two child peers. The total number of peers in the simulated P2P network varies from 72 to 720. Since we only focus on the parent seeking method, the simulation only includes the parent search session when generated peers entering the system.

We concern 3 important values: *Avg_Path_Delay*, *Avg_Hop*, and *Avg_Parent_Search_Cost* as follows.

- *Avg_Path_Delay* is the average value of path delay for all peers to get media stream from the server; VoD services require a small value of it.
- *Avg_Hop* is the average number of hops for peer to server. A smaller value of it makes a slighter effect to other peers when a peer leaves or drops.
- *Avg_Parent_Search_Cost* is the average number of potential parent peers contacted by a newly arrived peer before finding its most suitable parent. A smaller cost can provide a smaller initial delay.

5.4.2 Simulation Results

5.4.2.1. Prim-Dijkstra algorithm

Table 5-1 shows the parent location results for 72 peers using the Prim-Dijkstra algorithm.

α	0	0.2	0.4	0.6	0.8	1
Delay	137.8	104.0	86.6	81.7	75.6	73.7
Hop	3.57	2.93	2.43	2.22	1.79	1
Connection	8	11	15	19	28	72
	MST	Prim-Dijkstra Tree				SPT

Table 5-1: Parent location results

If we take a limit of 8 for the direct server-peer connection, the results are shown in Table 5-2. It is noted that the distance between points generated in the square is taken as the network distance between peers in simulated P2P network. The path delay from the server to a peer is the sum of distance of all point-to-point sessions along the server-to-peer route.

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Hop	3.57	3.39	3.31	3.13	3.15	3.13	3.17	3.14	3.15	3.03	3.03
Delay	137.8	127.2	116.3	114.5	114.9	114.3	125.3	125.7	126.6	130.1	130.1

Table 5-2: Parent location results
(with a limit of 8 for the direct server-peer connection)

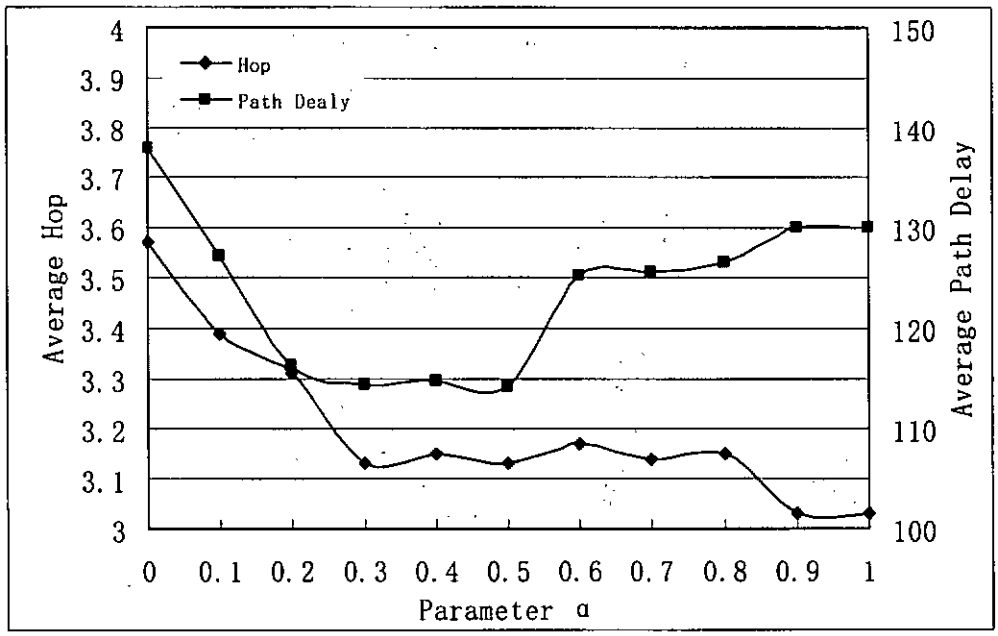


Figure 5-4: Results on average hop and path delay

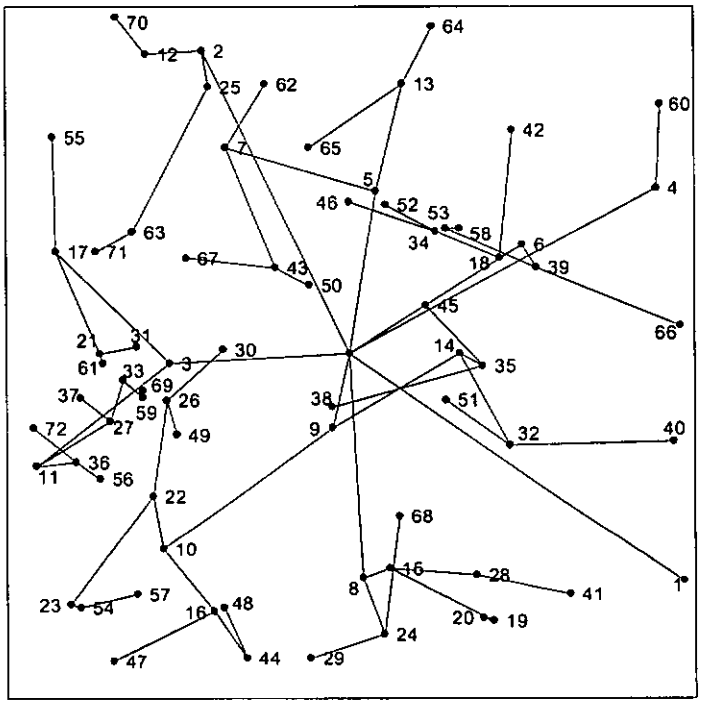


Figure 5-5: Optimal Network Spanning Tree (α = 0.5, connection =8)

From Figure 5-4 and Table 5-2, we can see that the average number of hop and path delay both keep in a low level when $0.2 \le \alpha \le 0.5$. It is noted that the optimal

point is at around $\alpha = 0.5$. The optimal network spanning tree at α=0.5 is shown in Figure 5-5.

5.4.2.2 Fast Parent Searching

Figure 5-6 shows the parent searching cost in terms of the total number of peers searched. In our simulations, α is set to 0.5. We can see from the results that without the filter function of network triangle inequality, the parent search is very expensive, especially when the P2P network consists of a large number of peers. The newly arrived peer will contact every potential parent peer, and the cost is very high. It is shown that the fast parent searching method using the network triangle inequality reduces the searching cost a lot and performs well in a large P2P network.

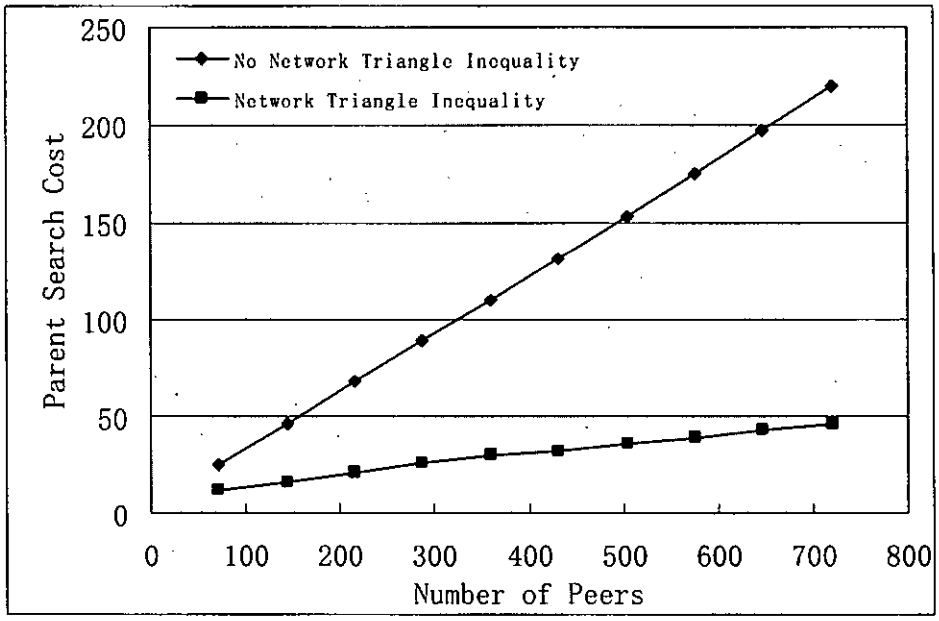


Figure 5-6: Searching cost for parent search of a newly arrived peer

5.4.2.3 Fast Searching in Peer Contact List

Since in the contact list lookup/search, the *shortest_path_delay* value is updated whenever a better parent is found. If the searching order can lead us to the best parent faster, fewer peers has to be contacted. Guyton et al. [GUYT1995] proposed using the following formula to compute the predicted value of $PathDelay_{P_j}(P_{new}, S)$ as:

$$Avg[PathDelay_{P_j}(P_{new}, S)] = \frac{Min[PathDelay_{P_j}(P_{new}, S)] + Max[PathDelay_{P_j}(P_{new}, S)]}{2}$$

However we found in the simulation that it is better to search the potential parent list in the sequence of peer's arrival time rather than search in an increasing $Avg[PathDelay_{P_j}(P_{new}, S)]$ order.

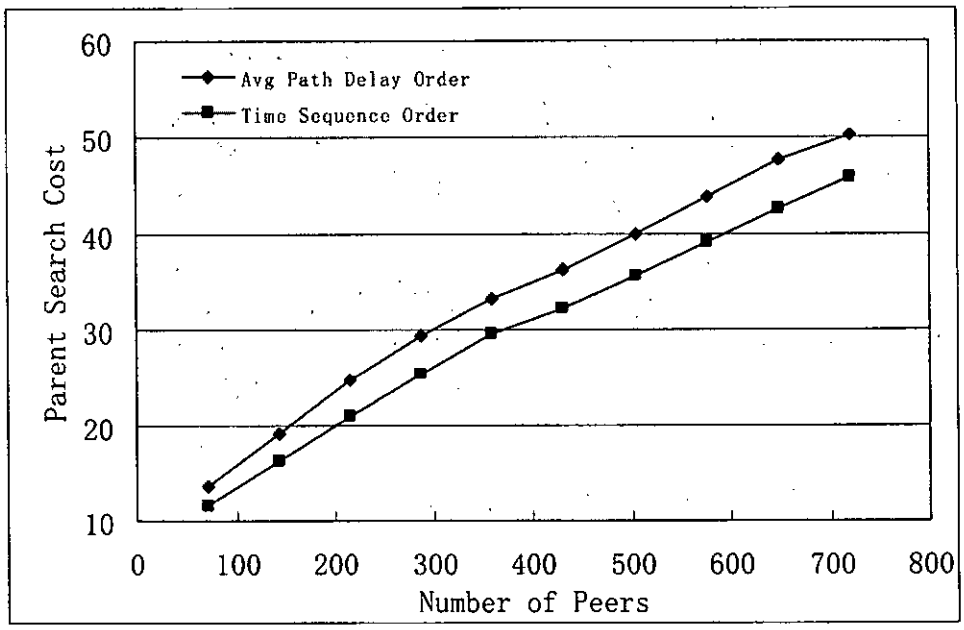


Figure 5-7: Searching cost for different searching orders

Figure 5-7 compares the performance of two searching orders in the potential

parent peer list with the Prim-Dijkstra algorithm (α=0.5), one follows the order the predicted path delay; the other is in order of the sequence of peer arrival time. It shows in a P2P network, the peer arrival time sequence order is a better choice. Earlier arrival potential parent peers has a smaller hop number from the server. They should be contacted firstly.

5.5 Chapter Conclusion

In this chapter, we have developed a new parent searching method for a P2P media streaming system. Newly arrived peers join the P2P network based on the Prim-Dijkstra algorithm. With the aid of a filter function using the network triangle inequality and a reference function of the server, we can conduct a faster parent search. Besides, some searching skills to further reduce the parent search cost are compared. From the simulation results, the Prim-Dijkstra algorithm which keeps the server-to-peer path delay low can construct a better spanning tree for a P2P network than the algorithm simply makes newly arrived peers join the closest parent (minimum spanning tree). It is also shown that the use of network triangle inequality greatly reduces the parent searching overhead and performs well with a large P2P network.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In recent years, the issue on video streaming on the Internet has aroused a massive attention from academia and commercial industry. Most of the current streaming solutions are using the client/server architecture in which a dedicated server is used to serve a number of clients over the network. To increase the scalability of such a system, one of the possible solutions is to use the peer-to-peer (P2P) architecture. The P2P model in file sharing/downloading is very successful in Internet world. However, the work on P2P video streaming is on-going and there are still many problems to be solved before it can be widely deployed in the Internet. This thesis presented our work in supporting efficient and reliable video streaming on a P2P network.

Firstly in Chapter 3, we made a detailed performance study on the hybrid ARQ method in a video streaming system. The study showed that hybrid ARQ is a technique that can effectively improve the communication reliability in a packet loss channel. We also proposed an adaptive scheme to fully utilize the bandwidth and improve the error correction performance. In our proposed scheme, we set an

upper bound and a lower bound to the number of parity packets in a FEC packet group, and send extra parity packets when the server bandwidth is not fully occupied.

Using the idea of hybrid ARQ, we then proposed a new P2P architecture for on-demand video streaming. In the proposed system, all video clients form a P2P forwarding chain for delivering video data from the source/seed. A standby server is maintained for sending FEC coded video stream to clients having difficulty to receive from their parents. A parent-child exchange (PCX) routing mechanism was also developed to adjust the P2P chain and push down the peers with network problems. Together with the standby server, PCX helps to prevent the cascading effect of a peer failure. Simulation results showed that PCX greatly reduces the effect of bad peers to the P2P chain. Also, the client playback continuity is improved by the backup support from the standby server with little cost. Besides, some QoS and charging mechanisms for a commercial VoD system based on the proposed architecture were suggested.

Finally, we derived an efficient parent searching method for a tree-structured P2P media streaming system, which targeted for live video streaming. With the proposed method, newly arrived peers will join the P2P network and search their best parents based on the Prim-Dijkstra algorithm. We made use of the network triangle inequality to estimate the server-to-peer path delay (S2PPD). In faster

parent searching, the server estimates the server-to-peer path delay of the newly arrived peer for all potential parent peers. Only parts of the potential parents who make the new peer a lower S2PPD are searched. From the simulation results, it was shown that the Prim-Dijkstra algorithm can maintain a small server-to-peer path delay and construct a better spanning tree for a P2P network than the algorithm simply makes newly arrived peers join the closest parent (minimum spanning tree). The use of network triangle inequality can greatly reduce the search overhead of peers and perform well with a large P2P network.

6.2 Suggestion for Future Work

For the on-demand video service introduced in Chapter 4, no VCR function is considered. In the system, the peer can freely reverse to a displayed scenario to watch it again since those data are stored in the buffer. However, a peer cannot jump to a scene not received yet, i.e. to do fast forward. To support this function, the standby server can directly stream the required data to the peer and the peer remains its position in the P2P forwarding chain. Such a solution is simple but requires extra server bandwidth resource. Another possible way is illustrated in Figure 6-1. The peer leaves its original chain position and gets a suitable parent with the help of standby server. As an example shown in the figure, to jump from time 14:00 to time 19:00, Peer D has to be pushed upward and assigned a new parent (Peer A) and a new child (Peer B). Peer D receives video stream starting at

19:00 from Peer A. Because Peer D has no suitable media data to feed its child peer (Peer B) instantly, the standby server may provide FEC stream to Peer B when its buffer is under the threshold *H*. Besides, the server has to stream video data between 15:00 and 19:00 to Peer D to fill its buffer gap. When Peer D has buffered enough data to serve Peer B, it streams to Peer B as the parent. In the solution described above, the standby server only temporally steams data to Peer D and Peer B, and the cost of bandwidth is small. In future work, the provision of such VCR functions can be further investigated.

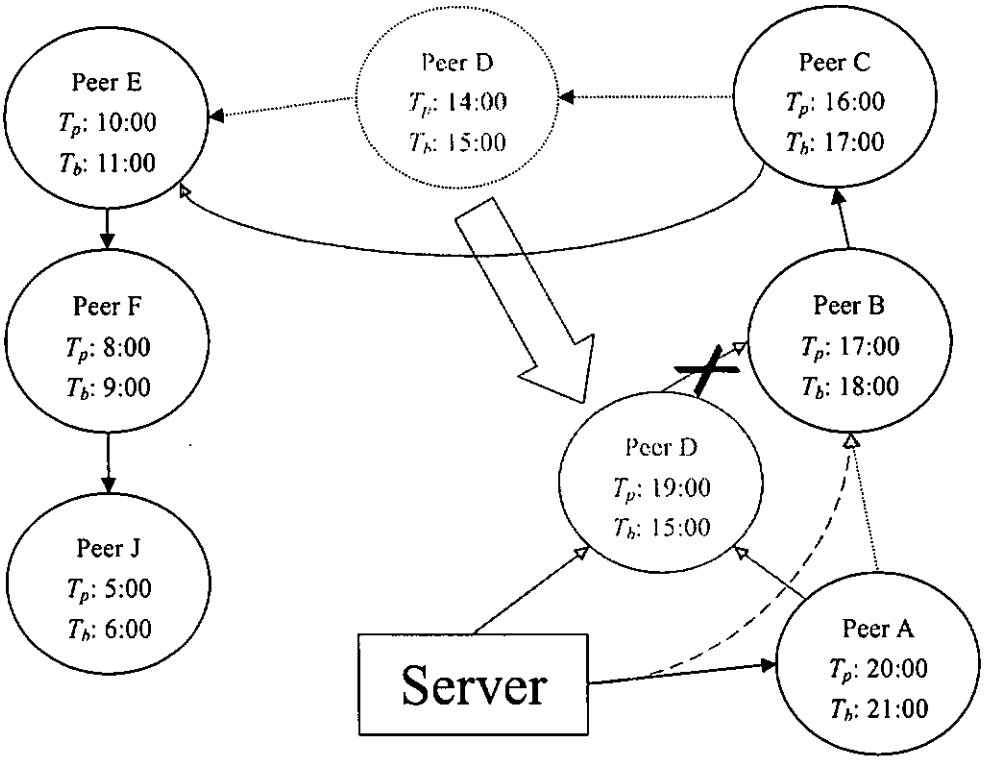


Figure 6-1: VCR function for P2P Video Streaming

In Chapter 5, the parent search method for real-time content media streaming is discussed. The parent search method is used to construct the network spanning

tree for the media streaming system in a P2P network. The maintenance of the tree, such as client departure and tree adjustment, is not studied. In future work, the network spanning tree maintenance and data transmission route has to be included. Besides, clustering may also be integrated to our P2P media streaming tree construction. Nearby peers cluster together and are taken as one node. In real situation, peers in the same local network or under the same ISP network are possibly ideal neighbors of the same cluster. Clustering may speedup the parent search and simplify the network structure. Study of such a hybrid approach can be conducted.

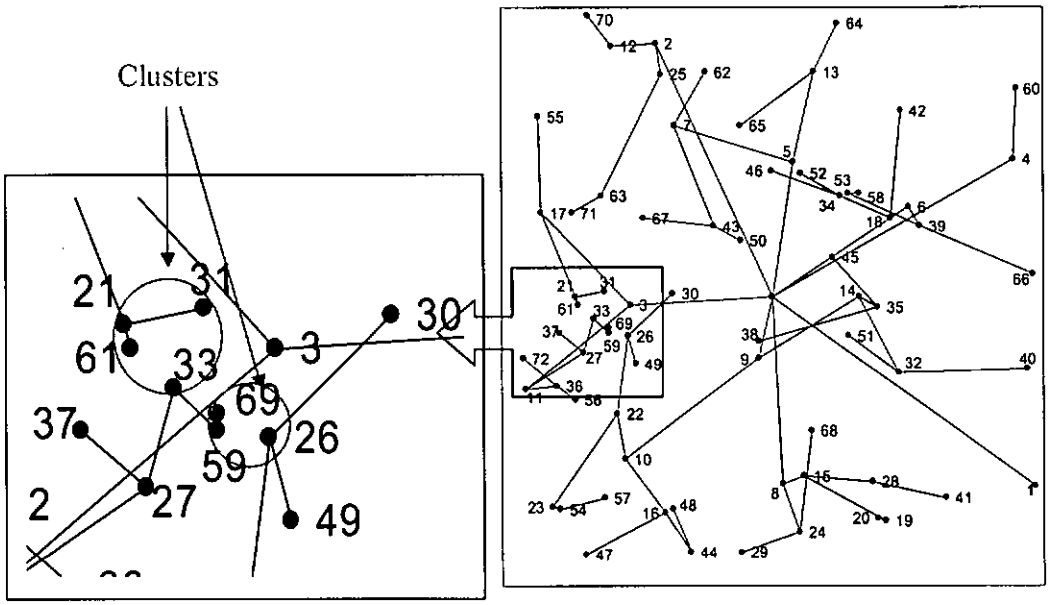


Figure 6-2: Nearby peers cluster together and are taken as one node.

References

[ALPE1995] C.J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger, "Prim-Dijkstra tradeoffs for improved performance driven routing tree design", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems,* vol.14, no. 7, pp. 890-896, July 1995.

[BACC1997] B. Baccala, *Connected: An Internet Encyclopedia*, April 1997.

[BANE2002] S. Banerjee, B. Bhattacharjee and C. Kommareddy, "Scalable application layer multicast", *Proc. SIGCOMM'02*, pp. 205–217, Aug. 2002, Pittsburgh, Pennsylvania, USA.

[BANE2004] S. Banerjee, C. Kommareddy and B. Bhattacharjee, "Efficient peer location on the Internet", *Computer Networks,* vol.45, pp. 5-17, 2004.

[BASI1996] S. Basith, "*MPEG: Standards, Technology and Applications*", June 1996. Available online, <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/sab/article2.html>

[BT] Available from BitTorrent Home Page, <http://bitconjurer.org/BitTorrent>.

[BURR2001] Alister Burr, *Modulation and Coding for Wireless Communications*, Harlow, Prentice Hall, 2001.

[CAHN1998] Robert Cahn, Robert Chan, *Wide Area Network Design: Concepts and Tools for Optimization*, Morgan Kaufmann, 1998.

[CHU2002] Y.H. Chu, S.G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on selected areas in commun.*, vol. 20, no.8 , pp.1456-1471, Oct. 2002.

[CONK2001] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman and Y.A. Reznik, "Video coding for streaming media delivery on the Internet", *IEEE Trans. on Circuits and Systems for Video Technology*, vol.11, No.3, pp.269-281, March 2001.

[ED] Available from eDonkey Home Page, <http://www.edonkey2000.com>

[FELL1968] William Feller, *an Introduction to Probability Theory and Its Applications*, 3rd ed., Vol. 1, New York: Wiley, 1968-1971

[FLOY1999] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in Internet", *ACM/IEEE Transactions on Networking*, vol.7, no.4, pp.458-473, Aug. 1999.

[FLOY2000] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications", *Proceedings of the 2000 ACM SIGCOMM Annual Technical Conference*, pp.43–56, Aug. 2000, Stockholm, Sweden.

[FREE] Available from Freenet Home Page <http://www.freenet.com>

[GNUT] Available from Gnutella Home Page <http://www.gnutella.com>

[GUO2003] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service", in *Proceedings of the 12th World Wide Web Conference (WWW-03)*, pp.301-309, May 2003, Budapest, Hungary.

[GUYT1995] J. Guyton and M. Schwartz, "Location nearby copies of replicated Internet servers", *Proceedings of SIGCOMM*, pp.288 – 298, 1995, Cambridge, Massachusetts, USA.

[HEFE2004] M.M. Hefeeda, B. K. Bhargava, David K. Y. Yau, "A hybrid architecture for cost-effective on-demand media streaming", *Computer Networks*, vol.44, pp.353-382, 2004.

[HOTZ1996] S. Hotz, "Routing information organization to support scalable interdomain routing with heterogeneous path requirement", *PhD thesis*, University of Southern California, 1996.

[HUA1998] K.A. Hua, Y. Cai and S. Sheu, "Patching: A multicast technique for true video-on-demand services", *Proceedings of the 6th ACM International Conference on Multimedia*, pp.191-200, 1998, Bristol, United Kingdom.

[ISO1996] ISO/IEC and ITU-T, "Information Technology - Generic coding of moving pictures and associated audio information - Part 2: Video," ISO/IEC 13818-2 (MPEG-2) - ITU-T Recommendation H.262, 1996.

[JACO1996] S. Jacobs and A. Eleftheriadis, "Providing video services over networks without quality of service guarantees", *Proceedings of the WWW Consortium Workshop on Real-Time Multimedia and the Web*, October 1996, Sophia Antipolis, France.

[KALL1997] Kallenberg, O. *Foundations of Modern Probability.* New York: Springer-Verlag, 1997

[KAZA] Available from Kazaa Home Page, <http://www.kazzaa.com>.

[LU2000] G. Lu, "Issues and technologies for supporting multimedia communications over the Internet", *Computer Communications*, pp.1323-1335, 2000.

[MAHA2003] A. Mahanti, D. Eager, M. Vernon and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery", *IEEE/ACM Trans. on Networking*, vol.11, no.2. pp.195-209, April 2003.

[MAHD1997] J. Mahdavi and S. Floyd. "TCP-friendly unicast rate-based flow control". *Technical Note on end2end-interest Mailing List*, January 1997.

[MAJU2002] A. Majumdar, D. G. Sachs, I. Kozintsev, K. Ramchandran, and M. Yeung, "Multicast and unicast real-time video streaming over wireless LANs," *IEEE Trans. Circuits and Systems for Video Technology,* vol.12, no.6, pp.524-534, June 2002.

[NAPS] Available from Napster Home Page, <http://www.napster.com>

[NIST] Available from NIST Net Home Page, <http://snad.ncsl.nist.gov /itg/nistnet>

[OHTA1994] N. Ohta, Packet video: Modeling and Signal Processing, Artech House, 1994.

[PADH1998] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "Modeling TCP throughput: A simple model and its empirical validation", *Proceedings of SIGCOMM '98*, pp.303-314, 1998, Vancouver, British Columbia, Canada.

[PURI2001] R. Puri, K. Ramchandran, K. W. Lee and V. Bharghavan,

"Forward error correction (FEC) codes based multiple description coding for Internet video streaming and multicast", *Signal Processing: Image Communication*, vol.16, no.8, pp. 745-762, May 2001.

[RAME1999] Sridhar Ramesh and Injong Rhee, "Issues in TCP model-based flow control," NCSU *Technical Report TR-99-15*, Presented also at IRTF RM meeting, Washington DC, Nov 1999.

[RATN2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proc. ACM SIGCOMM*, pp.161-172, Aug. 2001, San Diego, CA.

[REED1999] Irving S. Reed, *Error-control coding for data networks*, Boston: Kluwer Academic Publishers, 1999.

[RFC 791] J. Postel, Internet Protocol, RFC 791, September 1981

[RFC1889] H. Schulzrinne, GMD Fokus, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport protocol for real-time applications," RFC 1889, 1996.

[RFC1633] R. Braden, D. Clark and S. Shenker, "Integrated services in the Internet architecture: An overview," IETF RFC1633, July 1994.

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An architecture for differentiated services", IETF RFC2475, Dec, 1998.

[RHEE2000] I. Rhee, S.R. Joshi, 'Error recovery for interactive video transmission over the Internet,' *IEEE Journal of Selected Areas in Communications*, vol.18, no.6, pp.1033-1049, June 2000.

[RILE] M. Riley and I. Richardson, "An introduction to Reed-Solomon codes: principles architecture and implementation", Available online <http://www.4i2i.com/reed_solomon_codes.htm>

[SACH2001] D. G. Sachs, I. Kozintsev, M. Yeung, D. L. Jones, "Hybrid ARQ for robust video streaming over wireless LANs," *IEEE Proceedings International Conference on Information Technology: Coding and Computing*, pp. 317-21, April 2001, Las Vegas, NV. USA.

[SEN1999] S. Sen, L. Gao, J. Rexford, D. Towsley, "Optimal patching schemes for efficient multimedia streaming", *Proceedings of IEEE INFOCOM'99*, pp.455-463, March 1999. New York, USA.

[SHAN2003] Y.F. Shan and S. Kalyanaraman "Hybrid video downloading/ streaming over peer-to-peer networks", *Proc. International Conference on Multimedia and Expo (ICME)*, vol.II, pp.665 - 668, July 2003. Baltimore, Maryland, USA.

[SHUN] Available from Shunra\Cloud Home Page, <http://www.shunra.com/products/cloud/cloud_1.php>

[STEW1995] Stewart, W. J. *Introduction to the Numerical Solution of Markov Chains.* Princeton, NJ: Princeton University Press, 1995.

[STOI2001] I. Stoica, R. Morris, D. Karger, M. Kaashock, and H. Balakrishman, "Chord: A scalable peer-to-peer lookup protocol for Internet application," in *Proc. ACM SIGCOMM*, pp. 149-160, Aug, 2001, San Diego, CA.

[TRAN2004] D.A. Tran, K.A. Hua, T.T. Do, "A peer-to-peer architecture for media streaming", *IEEE Journal on selected areas in commun.*, vol.22, no.1, pp.121-133, Jan. 2004.

[XIAN2004] Z. Xiang, Q. Zhang, W. Zhu, Z.S. Zhang and Y.Q. Zhang, "Peer-to-peer based multimedia distribution service", *IEEE Trans. on Multimedia*, vol.6, no.2, pp.343- 355, April 2004.

[WANG1998] Y. Wang and Q.F. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE*, vol. 86, no.5, pp.974-997, May 1998.

[WU2000] D.P. Wu, Y.W. Hou and Y.Q. Zhang, "Transporting real-time video over Internet: challenges and approaches", *Proceedings of the IEEE*, vol. 88 issue 12, pp.1855-1877, Dec. 2000.

[WU2001] D.P. Wu, Y.W. Hou, W.W. Zhu, Y.Q. Zhang and J. M. Peha, "Streaming video over the Internet: approaches and directions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.11, no.3, pp.282-300, March 2001.

[ZHAN2004] L. Zhang and K. T. Lo, "A peer-to-peer architecture for on-demand video streaming on Internet", *Proceedings International Conference on Communications, Circuits and Systems (ICCCAS'04)*, pp. 525-528, July 2004, Chengdu, China.