Bandwidth-efficient Video Streaming with Network Coding on Peer-to-Peer Networks

by

Shenglan Huang

A thesis submitted to the University of London for the degree of
Doctor of Philosophy

Department of Electronic Engineering
Queen Mary, University of London
United Kingdom

August 2016

TO MY FAMILY

Abstract

Over the last decade, live video streaming applications have gained great popularity among users but put great pressure on video servers and the Internet. In order to satisfy the growing demands for live video streaming, Peer-to-Peer(P2P) has been developed to relieve the video servers of bandwidth bottlenecks and computational load. Furthermore, Network Coding (NC) has been proposed and proved as a significant breakthrough in information theory and coding theory. According to previous research, NC not only brings substantial improvements regarding throughput and delay in data transmission, but also provides innovative solutions for multiple issues related to resource allocation, such as the coupon-collection problem, allocation and scheduling procedure. However, the complex NC-driven P2P streaming network poses substantial challenges to the packet scheduling algorithm.

This thesis focuses on the packet scheduling algorithm for video multicast in NC-driven P2P streaming network. It determines how upload bandwidth resources of peer nodes are allocated in different transmission scenarios to achieve a better Quality of Service(QoS).

First, an optimized rate allocation algorithm is proposed for scalable video transmission (SVT) in the NC-based lossy streaming network. This algorithm is developed to achieve the tradeoffs between average video distortion and average bandwidth redundancy in each generation. It determines how senders allocate their upload bandwidth to different classes in scalable data so that the sum of the distortion and the weighted redundancy ratio can be minimized.

Second, in the NC-based non-scalable video transmission system, the bandwidth inefficiency which is caused by the asynchronization communication among peers is reduced. First, a scalable compensation model and an adaptive push algorithm are proposed to reduce the unrecoverable transmission caused by network loss and insufficient bandwidth resources. Then a centralized packet scheduling algorithm is proposed to reduce the uninformative transmission caused by the asynchronized communication among sender nodes. Subsequently, we further propose a distributed packet scheduling algorithm, which adds a critical scalability property to the packet scheduling model.

Third, the bandwidth resource scheduling for SVT is further studied. A novel multiple-generation scheduling algorithm is proposed to determine the quality classes that the receiver node can subscribe to so that the overall perceived video quality can be maximized. A single generation scheduling algorithm for SVT is also proposed to provide a faster and easier solution to the video quality maximization function.

Thorough theoretical analysis is conducted in the development of all proposed algorithms, and their performance is evaluated via comprehensive simulations. We have demonstrated, by adjusting the conventional transmission model and involving new packet scheduling models, the overall QoS and bandwidth efficiency are dramatically improved. In non-scalable video streaming system, the maximum video quality gain can be around 5dB compared with the random push method, and the overall uninformative transmission ratio are reduced to 1% - 2%. In scalable video streaming system, the maximum video quality gain can be around 7dB, and the overall uninformative transmission ratio are reduced to 2% - 3%.

Acknowledgments

I would like to express my sincere gratitude to my supervisors and friends who supported my during my PhD project. I would like to thank my supervisor Dr. Pengwei Hao for providing me an opportunity to pursue a PhD degree in Multimedia and Vision Research Group and provide me much guidance in both my research and daily life. Also, I am deeply grateful to Prof. Ebroul Izquierdo for his invaluable guidance as well as the persistent encouragement. The work presented in this thesis would not have been possible without his guidance. I would also like to thank Dr. Qianni Zhang for her many suggestions and encouragements during these years.

I am deeply grateful to Dr. Michele Sanna, who has worked closely with me on some of the topics of this thesis. Without his friendly support I could not have completed this work. I would also like to express my appreciation to Prof. DongDong Zhang, Prof. Jian Zhang, and Dr. Xin Shu for their valuable suggestions and comments on my research, which are indispensable for my research progress.

My study would not have been complete without the help and the friendship of others. Dr. Heng Yang, Dr. Yixian Liu, Dr. Xinyue Wang, Craig Henderson, Fiona Rivera, Xueke lv, Wenxuan Mou, Zhaoyang Xu, Zongyi Xu, Anqi He. I cherish every minute of their delightful companion.

Finally, with my love and gratitude, I would like to dedicate this thesis to my family who always give me the unreserved support and selfishless love. It is my best luck to have them in my life.

Table of Contents

List of Figures

List of Abbreviations

ADS	Asynchronous Distributed Scheduler
APA	Adaptive Push Algorithm
ARND	Advanced Random Network Coding
AVC	Advanced Video Coding
CAN	Content Addressable Network
CDN	Content Distribution Networks
CPS	Centralized Packet Scheduler
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Dynamic Cosine Transform
DPS	Distributed Packet Scheduler
GF	Galois Field
GOP	Group of Pictures
HDS	HTTP Dynamic Streaming
HEVC	High Efficiency Video Coding
HLS	HTTP Live Streaming
HNC	Hierarchical Network Coding
HTTP	Hypertext Transfer Protocol
ILP	Integer Linear Programming
ISP	International Service Provider

JSVM	Joint Scalable Video Model
MDC	Multiple Description Coding
MGS	Multiple Generation Scheduling
MSE	Mean Squared Error
NC	Network coding
ORA	Optimized Rate Allocation
P2P	Peer to Peer
PET	Priority Encoding Transmission
PRC	Practical Network Coding
PSNR	Peak Signal-to-Noise Ratio
QoS	Quality of Service
QoE	Quality of Experience
RLM	Receiver Driven Multicast
RLNC	Random Linear Network Coding
RLNC	Random Linear Network Coding
RTP	Real Time Transport Protocols
RTCP	Real-time Transport Control Protocol
RTSP	Real Time Transport Streaming Protocols
SCM	Scheduling Compensation Model
SGS	Single Generation Scheduling
SVC	Scalable Video Coding
SVT	Scalable Video Transmission
VNI	Visual Network Index
VoD	Video on Demand
WAN	Wide Area Network

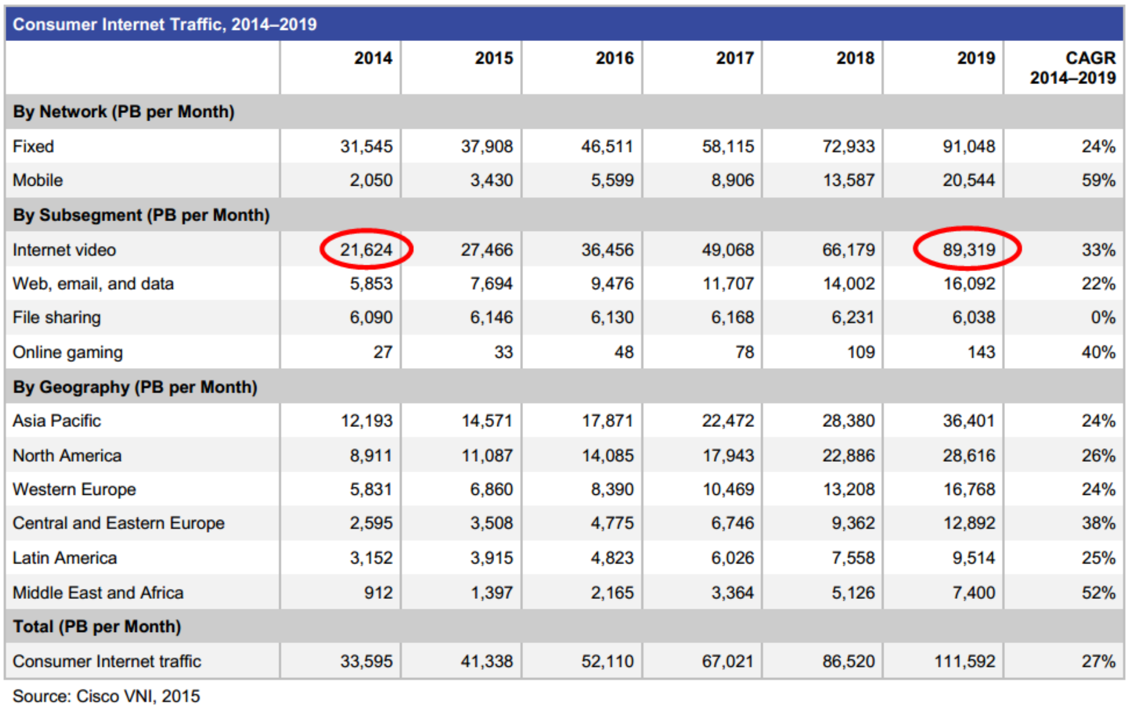
Chapter 1

Introduction

1.1 Background

The sky-rocketing proliferation of multimedia infotainment applications and high-end devices(e.g., smartphones, tablets, wearable devices, laptops, machine-to-machine communication devices) exacerbates the stringent demand for video streaming services. According to the latest Visual Network Index (VNI) report from Cisco [1], the global Internet traffic will increase threefold from 2014 to 2019, reaching 108.98 exabytes per month by 2019, and Internet video traffic will be 89.319 exabytes per month as shown in Fig. 1.1. The consumer Internet video traffic(TV, video on demand, Internet, and P2P) will be in the range of 80 to 90 percent of global consumer Internet traffic in 2019. Furthermore, according to this data and prediction, it would take an individual over 5 million years to watch the amount of video that will cross global IP networks each month in 2019. Every second, nearly a million minutes of video content will cross the network by 2019. It has become a consensus of the researchers in the field of communications that today's video streaming systems cannot meet the streaming demands in the foreseeable future.

Driven by an insatiable appetite for bandwidth in the Internet, advances in media



Consumer Internet Traffic, 2014–2019

	2014	2015	2016	2017	2018	2019	CAGR 2014–2019
By Network (PB per Month)							
Fixed	31,545	37,908	46,511	58,115	72,933	91,048	24%
Mobile	2,050	3,430	5,599	8,906	13,587	20,544	59%
By Subsegment (PB per Month)							
Internet video	21,624	27,466	36,456	49,068	66,179	89,319	33%
Web, email, and data	5,853	7,694	9,476	11,707	14,002	16,092	22%
File sharing	6,090	6,146	6,130	6,168	6,231	6,038	0%
Online gaming	27	33	48	78	109	143	40%
By Geography (PB per Month)							
Asia Pacific	12,193	14,571	17,871	22,472	28,380	36,401	24%
North America	8,911	11,087	14,085	17,943	22,886	28,616	26%
Western Europe	5,831	6,860	8,390	10,469	13,208	16,768	24%
Central and Eastern Europe	2,595	3,508	4,775	6,746	9,362	12,892	38%
Latin America	3,152	3,915	4,823	6,026	7,558	9,514	25%
Middle East and Africa	912	1,397	2,165	3,364	5,126	7,400	52%
Total (PB per Month)							
Consumer Internet traffic	33,595	41,338	52,110	67,021	86,520	111,592	27%

Source: Cisco VNI, 2015

Figure 1.1: Cisco forecasts over 80 percent of global consumer Internet traffic will be used by Internet video

compression technologies, and accelerating user demand, video streaming technologies have received a substantial amount of attention from both academia and industry for a long time. During the 1990s and early 2000s, research attentions mostly focused on the design and implementation of one-to-one streaming protocols, such as the design of the Real-time Transport Protocol (RTP) [2] specifically for streaming media. In recent years, to provide more convenience for users, HTTP video streaming was proposed and widely accepted by the industry. Based on the HTTP streaming protocol, the *Dynamic Adaptive Streaming over HTTP* (DASH) [3] framework was also proposed and adopted by Netflix. Furthermore, HTTP Live Streaming (HLS, Apple), HTTP Dynamic Streaming(HDS, Adobe System), and Smooth Streaming (Microsoft) are also proposed based on the HTTP streaming technology.

However, all the protocols mentioned above are based on the paradigm of one-to-one Internet communication. Maintaining unicast sessions for each user quickly became infeasible as the number of users increased. Many emerging applications, including

Internet TV and live event broadcast, require the support of large-scale video streaming. In 2012, a significant milestone for multimedia multicast was met by the biggest Internet-based TV and live on-demand video platform. With the live streaming of the London Olympic games through NBC, Youtube brought the live streaming of all sport events via the Google servers throughout the United States, reaching 225 million streams globally, with peaks of 500 thousand concurrent connections [4]. This sudden increase of content popularity adds a great burden to the streaming server and the content distribution network. It becomes imperative to construct a better streaming network to improve the overall network efficiency.

Since 1990, to improve the transmission efficiency and reduce the transmission burden in the core network, video multicast technology over IP network was thoroughly researched [5]. However, many Internet Service Providers (ISPs) simply block or disable IP multicast due to various security and economic concerns [6]. To solve this problem, many content provider like YouTube [7] and Netflix [8] employ a Content Delivery Network(CDN) with cache nodes which aggregate the users of a geographical region to a local access point. It is undeniable that CDN brings an excellent service quality. However, the growing demand for video content and popularity of the flash-crowd events like the Live Olympic Games is still a serious burden to these content providers. Therefore, it further motivated a new form of delivery called *peer-assisted* delivery network.

Peer-to-Peer (P2P) networks have been used extensively in multimedia live streaming as an effective transmission platform. In P2P live streaming, peers collaboratively organize themselves into an overlay and contribute their upload capacities to others. Commercially, a Chinese national television has already applied the P2P network to broadcast the Olympic Games in 2008 [9], and a French company and the French national television have already teamed up to provide a browser-based peer-assisted solution for broadcasting the football World Cup[2014] [10]. The European project P2P-Next [11] is developing an internet television standard based on P2P, which marks a high and ethical recognition of P2P technologies for future multimedia communication.

Network coding is a great breakthrough in Information theory and coding theory, and proposals have been made to apply this technique in P2P streaming network. Network coding was first introduced in Information Theory by Ahlswede et al. [12], who proved that the maximum capacity of a network can be achieved by transmitting mixed data at intermediate nodes. Based on the theoretical model, Chou et al. [13] proposed a practical network coding scheme for media streaming. Their work also proves that practical network coding can provide significant gains in throughput and a reduction in delay. Many large-scale applications of utilizing network coding in the field of multimedia streaming [14], [15], [16] have also demonstrated the associated benefits in reducing communications delays and facilitating the cooperation among nodes.

In this thesis, based on the conventional NC-based P2P streaming model. we identify that the conventional NC-based streaming network will generate *unrecoverable* and *uninformative* packets, thereby leading to bandwidth inefficiencies. *unrecoverable* packets are those network-encoded packets that can not be decoded before the playback deadline, and the *uninformative* packets are those received packets that are linear dependent to other previous received packets. To solve this problem, we firstly improve the conventional streaming model, then propose several packet scheduling algorithm to optimize the QoS and bandwidth efficiency based on the new streaming model.

1.2 Research Contributions

The contribution of this thesis are summarized as follows.

- An extensive and detailed overview of the state-of-the-art in P2P network with network coding is carried out. Additionally, open challenges of packet scheduling in this context are highlighted, which sheds lights on the research direction.
- In the NC-driven scalable video transmission (SVT) system, an optimized rate allocation algorithm is developed to achieve the tradeoffs between average video

distortion and average bandwidth redundancy in each generation. It determines how senders allocate their upload bandwidth to different classes in scalable data so that the sum of the distortion and the weighted bandwidth redundancy ratio can be minimized. It is discovered that the inefficient transmission for a single generation at one client node can lead to poor video quality in the subsequent generations of this client node and poor video quality of other client nodes, thereby resulting in poor overall video quality and delivery ratio. Better Quality of Experience (QoE) can be achieved when the sum of video quality and the bandwidth redundancy are both taken into consideration in the rate allocation algorithm for each generation.

- It is observed that the conventional NC-based streaming network generates *unrecoverable* and *uninformative* packets, thereby leading to bandwidth inefficiencies. This is because all transmitted packets are network encoded. When information updates among users are asynchronized, these transmitted packets are likely to be undecodable or redundant. Furthermore, It is discovered that in conventional push-based schemes, it is impossible to avoid the transmission of uninformative and unrecoverable packets because fully intelligent scheduling carries enormous overheads for communication which cannot be compensated by the gains achieved. Therefore, to find a trade-off between the improved transmission efficiency and the information overhead needed to achieve it, we propose a scheduling compensation model (SCM), an adaptive push algorithm (APA), a centralized packet scheduler (CPS), and a distributed packet scheduler (DPS). They are proposed to reduce the unrecoverable and uninformative transmission, and lead to a better QoE in the non-scalable video streaming system. First, the SCM and APA calculate the number of packets that each receiver could receive from its neighboring nodes after taking the dynamic network conditions into account. The two algorithms work together to reduce *unrecoverable* transmissions. Then, the CPS and the DPS construct the centralized and the distributed multi-sender cooperation models separately. They accurately determine the number of packets that should be sent from each sender to each receiver in each generation. The experimental results prove that these proposed algorithms can reduce redundant packet transmissions and achieve better video quality and delivery ratio.

- In the scalable video transmission system, we further propose a novel multiple-generation scheduling (MGS) algorithm which determines the video classes that each receiver node should subscribe to so that the overall perceived video quality can be maximized. The MGS is formulated as a perceived video quality maximization problem in some upload bandwidth constraints. Through solving the maximization problem, the optimal layer subscription policy can be found. The MGS problem is solved using two methods. One transfers the MGS to a single generation scheduling(SGS) algorithm to provide a faster and easier solution. The other solves the MGS directly using a dynamic programming algorithm. Experimental results confirm that the MGS algorithm can bring better QoS in the scalable video transmission system. The research output is declaired in Appendex B.

1.3 Thesis Organisation

The rest of this thesis is organized as follows:

Chapter 2 introduces fundamental concepts of network coding, development of streaming network with particular emphasis on the state-of-the-art peer-to-peer network. This chapter also summarizes some classic algorithms in conventional NC-based P2P streaming network, as well as highlights open challenges of packet scheduling in this context.

Chapter 3 presents a conventional NC-based P2P streaming system. This streaming system describes the construction of the P2P network, the node communication methods, the encoding and decoding processes at the end user, and the construction of transmission region.

Chapter 4 presents a rate allocation optimization algorithm to achieve the tradeoffs between average video distortion and average bandwidth redundancy in each generation. This chapter addresses the problem of streaming scalable packetized media over a loss packetized network by determining the rate allocation among different packet classes such that the distortion and bandwidth redundancy can be minimized. A theoretical analysis and performance evaluation of proposed algorithm are carried out for this proposed algorithm.

Chapter 5 investigates that the conventional NC-based streaming network may generate *unrecoverable* and *uninformative* packets, thereby leading to bandwidth inefficiencies. This chapter solves this problem by proposing a scheduling compensation model (SCM), an adaptive push algorithm (APA), a centralized packet scheduler (CPS), and a distributed packet scheduler (DPS). The theoretical analysis and performance comparison of proposed algorithms are carried out for these two packet scheduling algorithms.

Chapter 6 proposes a novel multiple-generation scheduling (MGS) algorithm to improve the overall perceived video quality. The MGS is formulated as a perceived video quality maximization problem in some upload bandwidth constraints. Through solving the maximization algorithm, the optimal video class subscription policy can be found. The MGS problem is solved using two methods. One transfers the MGS to a single generation optimization(SGS) algorithm to provide a faster and easier solution. The other solves the MGS directly using a dynamic programming algorithm. Experimental results confirm that MGS can bring better QoS in the scalable video transmission system.

Chapter 7 concludes the thesis and provides some thoughts for future work.

Chapter 2

Fundamental Concepts and State-of-the-Art

In this chapter, Network Coding (NC) technologies are first elaborated. In this part, the characteristics of NC and the application of NC to streaming network was analyzed. Then the evolution of streaming network is presented. Through detailed analysis of past streaming network, we evaluate the advantages and disadvantages of existing streaming frameworks, with an emphasis on the introduction of P2P streaming network. The P2P streaming network has been proved to have remarkable advantages over other streaming networks, especially when it works with NC technologies. At the end of this chapter, we comprehensively studied the past and ongoing NC-based P2P streaming network and analyzed their goodness and drawbacks. These thorough studies and analyses help us identify the open challenges in present work and shed light on the research direction.

2.1 Network Coding for Content Delivery

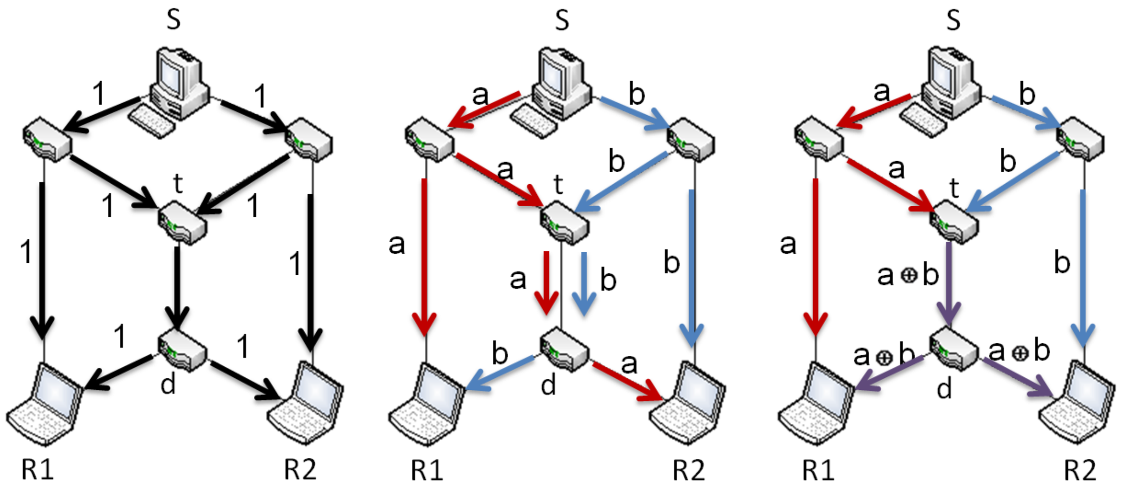
Network coding (NC) is one of the significant breakthroughs in the communication area. It was first proposed in 2000 [12] and later brought an enormous impact on all areas of communications and networking. The conventional network mainly uses routers to relay messages. In this network paradigm, a router only routes, or forwards messages. It means that each message on an output link must be a copy of a message that arrived earlier on an input link. In contrast, network coding allows each node in a network to perform some computations on the packets. Thus, each message sent on a node's output link can be a mixture of messages that arrived earlier on the node's input links. As such, network coding refers to the transmission, mixing(or encoding), and remixing(or re-encoding) of messages arriving at nodes, so that the transmitted messages can be decoded at their final destination.

The benefits of applying network coding to the multimedia application include augmented throughput, reduced vulnerability to packet erasures, minimized transmission delay and ease of deployment in large-scale distributed systems. These benefits make NC very attractive in the communication area, as these benefits are helpful in solving many tough communication questions. Furthermore, NC brings a new transmission mechanism and this new transmission mechanism also entails a radical change in the design of communication networks and applications as well. With the new NC paradigm in networks, the traditional coding and communication protocols can be rethought regarding the way that packets are created, combined and delivered. Therefore, it has attracted great interest in many applications, such as wireless networking, network security, data sharing, data storage, and video streaming.

To review the network coding technology, we first describe the fundamental theory of network coding, then the linear network coding technology. In the end, we present the well-established guideline that allows for putting NC into the practical streaming system.

2.1.1 Basic of Network Coding

In [12], Ahlswed, Cai, Li and Yeung give the definition of network coding: "Refer to coding at a node in a network as network coding". This key idea and the goodness of network coding in a communication network can be shown through the butterfly network depicted in Fig. 2.1.



a) capacity of the edge b) traditional approach c) approach with network coding

Figure 2.1: Butterfly network.

Figure. 2.1 depicts a common communication network. This network is widely known in the network coding literature as the butterfly network. The network is a directed network, and the capacity of each link is 1 as shown in Fig.2.1(a). There are one source node S, and two receiver nodes $R1$ and $R2$. The source S needs to send two different symbols a and b to two receivers separately. In the traditional routing approach, the network need two transmission slots to transmit both symbol a(red) and symbol b(blue) due to the bottleneck link between the intermediate node t and the intermediate node d. However, when the symbols can be combined and forwarded to the node t, one transmission slot is sufficient to transmit both symbol a and symbol b. This combination could be the simplest XOR operation in this toy network. In such a transmission network, each of receivers receives the coded symbol $a + b$ (purple) from node d and one of the original symbol from the side links. The receivers can subsequently decode the original symbol a and b. This simple example demonstrates the most important and incomparable goodness in increasing throughput and reducing transmission delay. The advantages make NC very attractive in current network transmission.

2.1.1.1 Benefits of Network Coding

- **Throughput Benefit**:

The achievable throughput with network coding can be larger than the achievable throughput only with routing. Although their is no general answer about the gain, Chekuri at al. [17] discuss bounds of the throughput benefit of network coding for several regular configurations of acyclic direction networks. They show that when the capacity of minimum cuts of the network is equal to $O(\sqrt{|V|})$, where $|V|$ is the number of nodes, the throughput benefit of NC is about 1.58 over routing. In some other cases, the throughput benefit is bounded by $1 + 1/(e - 1)$. In all, the throughput benefit of NC is remarkable .

- **Robustness Enhancement**:

In addition to the advantages in augmented throughput, NC is also helpful in overcoming the transmission difficulties over lossy channels by increasing the resilience to data loss. This is due to the nature of coding technologies. As some other channel coding techniques, NC allows for adding redundancies to data transmission, thereby providing higher data recovery probability in a lossy network. Furthermore, as data is encoded, it allows receivers to achieve error detection and reconstruction in many cases. We illustrate the advantage of NC in data recovery by a simple example of a server-client relationship in transmission. In Fig. 2.2, a toy example of network transmission is presented. The server node 1 needs to transmit packets p_1 and p_2 to the receiver node 2. In a lossy or an error network, server node 1 could transmit packet p_1 , packet p_2, and packet $p_1 XOR p_2$ to receivers. In this way, the loss of any one packet will not bring any original data loss. The receiver node can decode the packet p_1 and $p2$ if any of two packets arrive at the receiver node. The resilience to data loss is especially useful in satellite transmission and any other transmission with a strict time constraint.

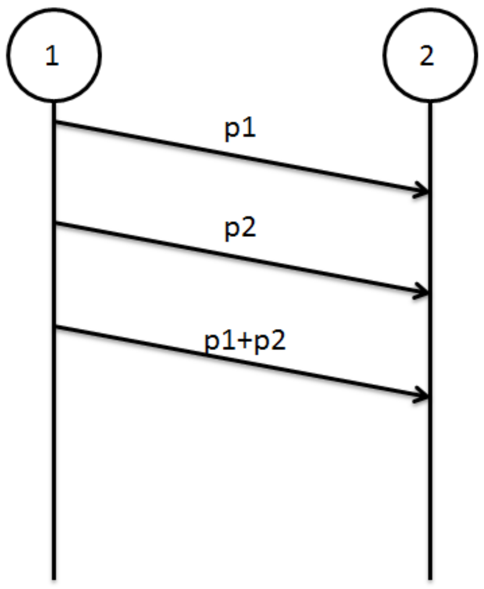


Figure 2.2: Toy example of network with one source node and one receiver node. loss resilience can be increased when NC is implemented in the source node

It is also important to emphasize the goodness of network coding over other rateless codes (also known as Fountain Code) [18]. The rateless codes are defined as a class of erasure codes with the property that a potentially limitless sequence of encoding symbols can be generated from a given set of source symbols such that the original source symbols can ideally be recovered from any subset of the encoding symbols of size equal to or only slightly larger than the number of source symbols[18]. However, the drawback of rateless codes is that they only perform on uncoded data to generate new encoded data. It means that only source nodes encode packets. In contrast, network coding is considered as a generalization of conventional store-and-forward routing techniques, and it was originally proposed in order to achieve multicast data delivery at the maximum data transfer rate in single-source multicast networks [19].

2.1.2 Linear Network Coding

Although the goodness of network coding is already proved in the area of Information theory, a more general coding mechanism is still needed to introduce NC to actual data transmission. Then in [20] and [21], linear network coding was proposed to allow network nodes to code data with linear functions on finite field $\mathbf{GF}(q)$, where q denotes the size of the finite field. Most network coding applications are developed with linear network coding, although non-linear network coding has also been studied, e.g., in Dougherty et al. [22], Kosut et al. [22], Lehman and Lehaman [23], Li et al. [24]. It has been proved that coding with a linear operator is sufficient to achieve the upper bound of the max-flow-min-cut theorem, with single or multiple sources. Considering the complexity of the non-linear coding, linear network coding is the best type of practical coding algorithm used in the practical applications.

In [20], a communication network is considered as a directed graph $G = (V, E)$. In the graph, V represents a set of vertex, and E represents a set of directed edge. We extract a single node from the whole graph such that this node have three incoming edges, and one outgoing edges. It can be expressed in the Fig. 2.3:

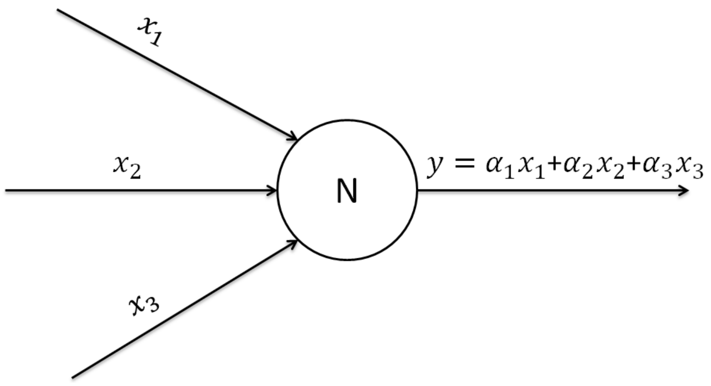


Figure 2.3: Illustration of linear network coding.

Let x_i represents the message in the incoming edge i received by node N, and y represents the message sent by node n. y can be represented by a linear combination of its incoming packets x_i. y can be expressed as:

$$y = \sum_i \alpha_i x_i \tag{2.1}$$

In the equation, the local coding coefficients α_i is an randomly generated coefficient that applied to each packet x_i. The receiver node can decode the linear equations generated based on the transmitted packets. Coded packets received at the receiver node yield a system of linear equations that should be solved so as to retrieve the original packet x_i.

Later, in [25], a distributed random linear network coding (RLNC) approach for transmission and compression of information in general multicast networks is proposed. In RLNC, network nodes independently and randomly select linear mappings from inputs onto output links over some field. It is proved that the RLNC achieves capacity with probability exponentially approaching 1 with the code length. Benefits of this approach are decentralized operation and robustness to network changes or link failures. Usually, the random linear network coding in a practical system is on a finite field F_q of size q, e.g, a Galois Field(GF) of size $q = 2^m$, where m is the number of coefficient bits.

2.1.3 Practical Network Coding

Practical network coding was proposed in [13] to standardize the NC in the area of multimedia streaming. Practical network coding relies on three key ideas: random coding, packet tagging, and data buffering. Random coding allows encoding in a distributed manner. Furthermore, tagging each packet with the coding vector also allows the distributed decoding. Buffering allows for asynchronous packet arrivals and departures with arbitrarily varying rates, delay, and loss. Different from network coding in the pre-mentioned multicast scenario, in practical implementations of multimedia communications over packet networks, the NC operations described in the previous subsection are applied to full data packets and not on bytes or elementary symbols independently. As such, a packet is treated as a vector of symbols (typically one byte each), and the same coding coefficients are applied to all symbols in the packet.

Practical network coding chooses to use RLNC to encode its received packets. The reason for choosing RLNC to develop practical network coding is that it is hard to get the centralized knowledge of the network topology for the purpose of computing, which is treated as a known information in previous theoretical work. To ensure the receiver can perform the appropriate decoding operations, each packet is tagged with the set of coefficients that describe the coding operations, and the set of coefficients are transmitted along with the encoded packets to the receiver. In practice, a header that conveys information about the coding coefficients is added to each packet. The global encoding vector entails a rate overhead, whose size depends on the number of source symbols and the size of the finite field.

Furthermore, in the multimedia communications application, the timing constraints is also an important criterion. Real networks transmit packets asynchronously, and media streaming sources continuously transmit new packets that have to be decoded before their expiration deadlines. To satisfy this requirement, packets belonging to multimedia stream are grouped into *generations* of size L (i.e., sets of L time consecutive multimedia packets) and network coding operations are applied to packets of the same generation [13]. To further explained how practical network coding solves the NC in streaming transmissions, we explain the random coding process, packet format, and the buffering model in the following subsection.

2.1.3.1 Random construction

The centralized knowledge of the network topology is very impractical in a real network. Besides, the construction of network coding needs the cooperation of all nodes in the network, which will bring an unacceptable amount of work between senders and receivers. To solve this problem, a distributed and randomized approach is desired. The key innovation in [13] is transmitting encoding packets with random network coding. As a consequence, receivers will receive a randomly combined version of the source data by a random combination over the whole network. In practical network coding, packets are allowed to be re-encoded at the intermediate nodes. The previous coefficients in the packet overhead are re-encoded as well and forward to the next node. The receiver can decode the original packet after it receives enough independent encoded packets.

The original packets can be fully decoded if the coefficients can form a full rank matrix at the receiver. To keep the innovation of the sent packets, the encoding operation is performed in a finite field of sufficient size. Many research also studies the impact of Field size to the inherent uninformative rate in the network [13], [26]. According to calculation, when the field size is 2^8, the coefficient matrix G is a full rank matrix with probability at least $1 - 2^{-8} = 0.996$. They proved that $m = 8$ is suitable to be used in the large-scale network. Furermore, some authors also proposed network coding over GF(2) by using sliding window to reduce coding complexity [27].

2.1.3.2 Packet tagging

In practical network coding, the receiver nodes do not have any centralized knowledge of the network topology or the decoding function. Therefore, a packet tagging format is necessary to transmit the global encoding vectors within the packet so that the receivers get the encoding functions.

We define the notation of a source packet $x_i = [x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,w}]$. x_i represents the source symbol and x_i means that the source symbol x_i can be further represented with vectors. So every receiver can recover h source vectors $x_1, ...x_h$ from any h received packets. Equivalently, we also defined a received packet as $d_i = [y_{i,1}, x_{i,2}, x_{i,3}, ..., y_{i,w}]$. d_i is the result of several combinations of a set of w packets. It can be represented as Eq. 2.2, where M_t is a $n \times h$ matrix.

$$\begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,w} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,w} \end{pmatrix} = M_t \begin{pmatrix} x_1 \\ \cdots \\ x_h \end{pmatrix} = M_t \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,w} \\ \vdots & \vdots & \ddots & \vdots \\ x_{h,1} & x_{h,2} & \cdots & x_{h,w} \end{pmatrix} \tag{2.2}$$

To include the information of coefficients in the packets, the global encoding vectors should be transformed with the packet data. This can be done by prepending a vector of length h to each original packet $x_i, i = 1, \cdot, h,$. Any receiver can decode the original source packet by using the Gaussian elimination. The prepending process of Eq. 2.2 can be rewritten as Eq.2.3.

$$Y_t = M_t \begin{pmatrix} 1 \cdots 0 & x_{1,1} & x_{1,2} & \cdots & x_{1,w} \\ \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 \cdots 1 & x_{h,1} & x_{h,2} & \cdots & x_{h,w} \end{pmatrix} = \begin{pmatrix} & y_{1,1} & y_{1,2} & \cdots & y_{1,w} \\ M_t & \vdots & \vdots & \ddots & \vdots \\ & y_{n,1} & y_{n,2} & \cdots & y_{n,w} \end{pmatrix} \tag{2.3}$$

According to the formula, the cost of network coding is the length of the pre-pending symbol. For a normal UDP packet, a typical maximum packet size excluding header is somewhat larger than 1400 bytes. We also can say that each packet has about N=1400 symbols if the field size is 2^8. If h is 50 in this matrix, the overhead is $50/1400 \approx 3\%$.

2.1.3.3 Buffering Model: Generation

In real networks, the number of packets per unit time on each edge varies due to loss, congestion, and competing for traffic. To synchronize the transmission, a generation based buffering model is proposed. The streaming source divides whole source information into several *generation* and each generation is assigned an identification number. In the packet header, a small field is used to distinguish the generation to which the packets belong to. The packets are encoded at the intermediate node only if all encoding source packets are in the same generation. These packets are encoded with random coefficients and transmitted to the next hop. Gauss-Jordan elimination is traditionally used to check for non-innovative packets. This also progressively decodes the incoming data, so that when the last incoming packet from a generation is decoded, the original information is available to be consumed at the receiver.

2.1.3.4 Problems of Practical Network coding

Although practical network coding brings many benefits. In real network, it also brings two types of superfluous transmission.

- **Uninformative Transmission**:

Uninformative transmission can be caused through two ways. First, due to the randomized nature of the coefficients, a small probability exists that an incoming packet is linearly dependent from the packets that have already been buffered due to the same coefficients. We call these packets that do not provide new information *uninformative* packets. *Uninformative* packets can also be generated due to asynchronous communication among sender nodes. Senders may push an encoded packet in generation g to a receiver when the receiver already decodes the whole generation g, because the stop signaling message does not arrive at the sender node in time. The first type of *uninformative* packet can not be avoided. However, the second type of *uninformative* packet can be avoided through an accurate transmission.

- **Unrecoverable Transmission**:

Due to the strict playback deadline of the live streaming application, network encoded packet could be unrecoverable due to the lack of encoded packets to decode the generation before the playback point. This kind of transmission is called *unrecoverable transmission.* In contrast, transmission with routing does not have this problem. All transmitted packets are recoverable. When a node can not decode the encoded packets before the playback deadline, these packets become useless and will be discarded. As such, the bandwidth efficiency of the whole streaming system is low, the perceived video quality at the receiver node is low as well. Therefore, the unrecoverable transmission leads to poor performance of the delivery system. Avoiding them through careful transmission system design is necessary.

2.2 Development of Internet Video Streaming

Over the last two decades, video streaming has received a substantial amount of attention from both academia and industry. Research interests start from the design of transport protocols for streaming video and then they have shifted to the peer-to-peer paradigm at the application layer. Besides, some research has focused on building streaming protocols based on the HTTP protocols, like Dynamic Adaptive Streaming over HTTP (DASH) [28]. In this section, we provide a retrospective view of the streaming protocols over the past two decades, with an emphasis on peer-to-peer streaming protocols. This survey can help us to find these open challenges in the design of streaming network [29].

We seek to go back in the time machine and present a retrospective view of the history of Internet video streaming by highlighting a few main research areas in the past two decades of research and development. Following the chronological order, four stages of research development on Internet video streaming are presented in Fig. 2.4.

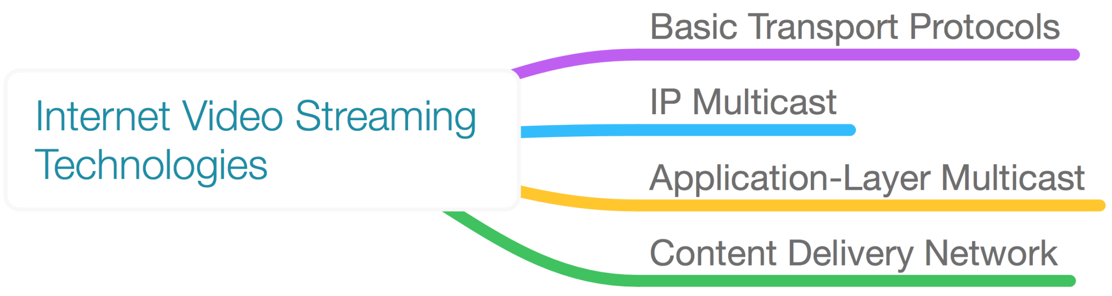

Internet Video Streaming Technologies
Basic Transport Protocols
IP Multicast
Application-Layer Multicast
Content Delivery Network

Figure 2.4: Main Technologies used in Internet Video Streaming

2.2.1 Client-Server Video Streaming

During the 1990s and early 200s, research attention mostly focused on the design and implementation of new streaming protocols. These initial streaming protocols were designed to receive video streams from the streaming servers over the Internet. In order to enhance the performance of the overall network, the Internet Engineering Task Force standardizes the Real-time Transport Protocol (RTP)/Real-time Transport Control Protocol (RTCP)/Real-time Transport Streaming Protocol (RTSP) protocol suite, including the RTP [2], the RTSP [30], [31]. RTP was proposed for the end-to-end real-time data streaming. RTSP was designed to maintain video sessions and to provide VCR-style control functionality, which enables users to pause, resume, or seek in video streaming. However, all these protocols are based on one-to-one Internet communication paradigm. Maintaining unicast sessions for each user quickly became unfeasible as the number of users scaled up. Many emerging applications, including Internet TV and live event broadcast, desire the support for video multicast, which allows delivering a video stream to a large number of clients.

2.2.2 IP Multicast

To support the growing demands for video streaming, IP multicast [5] was proposed as an extension to IP unicast, with the purpose of providing efficient multipoint packet delivery. In theory, IP multicast can be the most efficient multicast scenario because the network topology was best known in the network layer. IP multicast retains the semantics of IP and allows users to join or leave multicast groups dynamically.

As IP multicast scheme is a sender-driven scheme, multicasting scalable data from a content server to different receivers without a common target rate becomes a challenge question [32]. One solution was stream replication, which could be viewed as a trade-off between single-rate multicast and multiple point-to-point connections [33]. It was justified in a typical multicast environment where the bandwidth of receivers usually followed some clustered distribution. As a result, a set of streams could be used to match these clusters to achieve a reasonably good performance. Several receiver-driven layered IP Multicast (RLM) streaming protocols were also proposed. RLM [34] mapped different layers of a scalable video source to distinct multicast groups and let the receivers independently decide the number of layers to subscribe.

However, due to many ISPs directly block or disable IP multicast due to various security and economic concerns[6], the scope and development of IP multicast always remain relatively limited with IPv4. In IPv6, the multicasting has been integrated into the specification. Compared with the IPv4, the multicast in IPv6 is similer and safer.

2.2.3 Peer-to-Peer Network

Due to the limitation of IP multicast, the idea of using the application layer for multicast was proposed around 2000 [35],[36], [37], [38], [39], [40]. *Application Layer Multicast* (ALM) is defined as the implementation of multicast forwarding functionality at end-host. ALM is also known as peer-to-peer multicast, end system multicast. In some reviews like [29], they distinguish the ALM and P2P multicast based on the structure of the network. The authors claim that ALM and P2P multicast share the same design philosophy. However, ALM is largely push-based, and P2P multicast is mostly pull-based. In some other review articles [38], authors think both ALM and P2P represents an overlay multicast system, where end users contribute their upload bandwidth to achieve video multicast. In Section. 2.3, we use P2P network to represent all overlay multicast network, including push-based network and pull-based network. We survey P2P networks from centralized, recursive, unstructured tree-based network, structured tree-based network, and mesh-based network, which covers all type of application layer multicast structure in the following sections.

2.2.4 Content Delivery Network

Another method of supporting media multicast is to build a large-scale unicast system by reusing the client-server model that is widely used before, instead of building a multicast network. It is obvious that the high demand of bandwidth to media server instantly questions the feasibility of this approach. Therefore, the content delivery network was proposed [41]. As a direct improvement to the conventional clientserver service model, CDNs slightly expand the concept of the "server". In the CDN model, clients find a content delivery server that is located nearest to download a video, instead of downloading it directly from a video source server. The video source server acts as the "server" for content delivery servers by pushing video contents to them. Therefore, the CDN model can be seen as a two-layer clientserver model. This not only reduces the distance that content travels but also reduces the number of hops a data packet must make. The benefits are less packet loss, optimized bandwidth and faster performance – improving overall user experience. In fact, YouTube [7] keeps employing CDNs to deliver its most popular videos to its users, and Akamai [42] runs a profitable business as the worlds largest CDN service.

However, CDN servers are expensive to deploy and maintain. The server capacity (including processing power and outbound bandwidth) that can be allocated to the distribution of one media file is limited. The CDN model can not take advantage of the upload bandwidth of the clients, which puts all the load on the CDN infrastructure (the involved servers and networks). In fact, the bandwidth provided by a CDN has to grow proportionally with the number of clients, which makes CDNs an expensive solution for large client populations, although the excellent service quality of an adequately dimensioned CDN is undeniable.

In retrospect, different from the CDN networks, P2P networks store the media data at client nodes after the streaming service, and client nodes act as supplying peers and stream the media data to other requesting clients (peers). In comparison, the P2P network is easier to deploy and requires fewer resources from the server. It is argued that both CDN- and P2P-based architectures have their advantages and disadvantages, and each architecture does not provide a cost-effective and scalable solution for streaming media distribution. Therefore, a hybrid system combining both CDNs and P2P has long been envisioned [43][44]. For instance, LiveSky is a P2P-CDN hybrid streaming system designed and deployed recently [45]. The servers in CDNs are organized in a tree structure, and the clients form a tree-mesh combined overlay to forward the received streams. A better balance between streaming quality and scalability is achieved by this P2P-CDN hybrid architecture, which brings shorter startup latency, and less cross-ISP traffic.

2.3 Peer-to-Peer Streaming Network

A robust real-time video communication service over the Internet in a distributed manner is an important challenge. In this context, Peer-to-Peer (P2P) networks are playing an imperative position for providing efficient video transmission over the Internet [46] [47]. A distributed network architecture is called a Peer-to-Peer network, if the participants share a part of their hardware resources (processing power, storage capacity, network link capacity, etc). These shared resources are helpful in providing the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). The participants of this network are called resource providers and resource requestors respectively [48].

Peer-to-peer (P2P) is a decentralized communications model in which peers collaboratively organize themselves into an overlay and share their upload capacities to serve others. It was proved that the benefits introduced by overlay multicast overshadowed the topological inefficiency introduced, with respect to bandwidth consumption on the backbone network [49]. In P2P networks, clients are both resource providers and resource users. It means that the capacity of peer-to-peer networks can increase as more users begin to access the content. This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small.

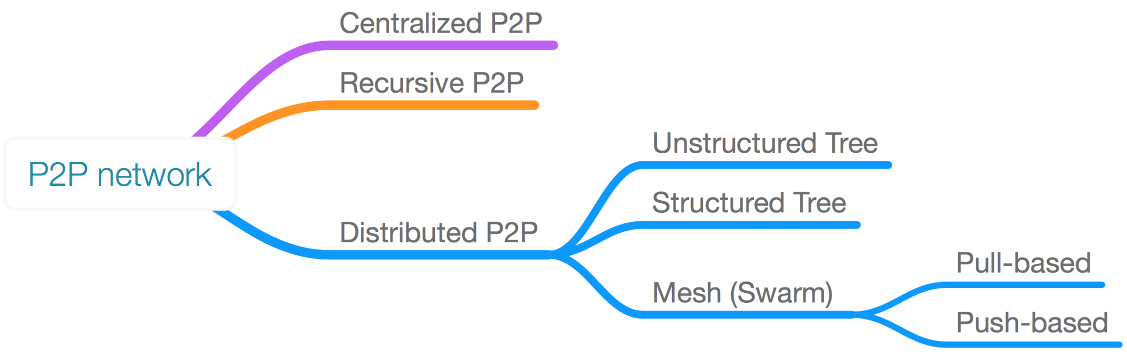


Figure 2.5: Main Technologies used in peer-to-peer streaming network

In this section, we describe these major algorithmic choices to build a P2P live video streaming network as shown in Fig. 2.5. Firstly, the supplier-receiver relationships can be determined as centrally, recursively, or a fully distributed algorithms. It determines how the network is constructed. In the centralized network, a peer searches its parent by querying the central server. In the recursive scheme, nodes form a cluster and organize themselves in a hierarchical way. A node joins the network by recursively sending requests from the root to the leaf node. In a fully distributed algorithm, a node joins the network by contacting a well-known tracker to obtain a list of peers in the system and establishes a neighboring relationship. The fully distributed network can be further divided into tree-based network and mesh-based network. The tree-based network can be further divided into the structured tree and unstructured tree.

In the mesh-based network, according to the sender-receiver relationship, it can be further divided into the pull-based and push-based scheme. Moreover,the overall streaming network can also be classified according to the way that peers choose their parents.

The way that peers choose parents can be divided into network-driven and data-driven. Network driven means that peers choose parents based on the geographic location. Data-driven means that peers choose parents based on data availability. Usually, centralized and recursive network use network-driven scheme and fully distributed network use data-driven scheme [50].

2.3.1 Centralized Schemes

In centralized schemes, a peer queries the central server for parents upon its arrival at the system or departure of its parent. Usually, the controller uses some information (e.g. distance from n other members) to maintain a spanning tree. The main concern of centralized algorithms is scalability. In the study of CoopNet [51], the trace captured on September 11, 2001, at MSNBC is simulated. The study shows that an ordinary server can only handle 18,000 hosts and 1000 churns per second.

The representative centralized schemes include ALMI [52] and CoopNet [53]. ALMI [54] aims to support small group multicast applications with multiple sources. When a peer x wants to join a session, it sends a JOIN message to the controller. Then the controller assigned a member ID to peer x and gives peer x the member ID and IP address of its parent. Then peer x sends a GRAFT message to its parent to attach to the tree. The parent keeps sending packets to the receivers without extra requests. The peer node monitors the status of its parent. If its parent leaves, the peer asks the controller for a new parent. A heuristic degree bounded minimum spanning tree algorithm is used by the controller to build a bi-directional shared tree. It periodically calculates the cost reduction of a new tree, and if the reduction exceeds a threshold, it instructs peers to switch parents. Each member's degree on the tree is bounded by its access link's bandwidth.

The CoopNet [53] targets the flash crowds at web servers with streaming content, live or on-demand, different from ALMI, The controller in CoopNet returns a list of potential parents according to a set of nearby peers. The video is encoded with multiple description coding (MDC). For each substream, a peer selects a parent from the list. When a peer leaves gracefully, it notifies the server, which finds new parents for its orphaned children. Each peer monitors the packet loss rate in each substream. If the rate exceeds a certain level, the peer will wait for its parent to fix the problem if its parent also experiences packet losses; otherwise, it will ask the server for a new parent.

2.3.2 Recursive Schemes

Recursive scheme include Overcast [55], NICE [56], ZIGZAG [57], THAG [58], NHAG [59], TURINstream [60], HMTP [61], Yoid [62], and Island Multicast [63]. It means that each node joins the streaming system through recursively enquiring other nodes, in contrast to informed by tracker nodes directly in centralized schemes. According to whether peers are first grouped into clusters, recursive schemes can be further classified into three sub-categories: without clusters, with clusters, with IP multicast. NICE, ZIGZAG, THAG, NHAG, and TURIN-stream first group peers into clusters and then build trees with clusters of nodes. Clusters are used for different purposes, such as imrpoving scalability, building multiple interior node-disjoint trees, or achieving a low tree cost. Consequently they organize peers into clusters in different ways. All these schemes have a control plane and a data plane. On the control plane, peers organize into clusters, select cluster leaders, and split and merge clusters to keep the cluster size within a range. HMTP, Yoid and ISland Multicast build trees with IP multicast island as nodes and use scoped IP multicast inside each island.

To summarize, centralized schemes can construct short, balanced, and low-cost trees if the central server has the global knowledge of peers virtual network positions and upload bandwidth. Centralized schemes have low join complexity, and provide quick repair of the broken trees. The maintenance overhead is low, and an ordinary server can handle the workload of thousands of peers. In contrast, recursive schemes build low-cost trees without having global knowledge. However, peers close to the tree root may have large processing workload. Grouping peers into clusters can improve scalability and achieve other benefits, but often imposes new constraints. The use of hierarchical clusters will severely increase the processing and uploading workload of peers at the top of the hierarchy. Peer churn incurs a long convergence time. Thus, recursive schemes are not necessarily more scalable than centralized schemes.

2.3.3 Fully Distributed Schemes

Fully Distributed schemes include structured tree-based schemes, unstructured tree-based schemes, and swarm-based schemes.

2.3.3.1 Structured Tree-based Schemes

Structured tree-based schemes is a type of fully-distributed network. It organizes peers with Distributed Hash Tables (DHT). DHTs provide a scalable unicast routing mechanism. Each peer has a routing table, and a peer can reach any other peer within $O(logN)$ hops, where N is the number of peers. Structured tree-based schemes include Ratnassamy et al. [64] (uses CAN [64]), Bayeux [65], SplitStream [66] and Borg [67]. Ratnasamy et al. [64] explore the architecture of Content Addressable Network (CAN) to flood multicast packets to group members. In CAN, the source forwards packets to all neighbors, other nodes only forward packets to certain directions inferred from their positions in the torus. Bayeux uses Tapestry [68], where a new peer x sends a JOIN message to the root, and then the root replies with a TREE message, which is routed back to peer x. Upon receiving the TREE message, intermediate nodes set up a virtual link so as to forward future packets from the root to peer x. When peer x wishes to leave, it sends a LEAVE message to the root. The root replies with a PRUNE message, which is routed to peer x along the same path as the TREE message. Upon receiving the PRUNE message, intermediate nodes close the virtual link.

2.3.3.2 Unstructured Tree-based Schemes

The unstructured tree-based scheme is a type of distributed P2P network. In the unstructured tree-based scheme, the algorithmic choices can be further divided into data-driven and network-driven schemes. In this section, we analyze the basic concept of the unstructured tree-based scheme first. Then we survey the data-driven and network-driven respectively.

In the unstructured tree-based scheme, a child renews the parent-child relationship every T_s. A parent forwards each received packets to its children for a period of length $T_m > T_s$. Then the parent-child relationships in the whole network naturally form a tree. The main difference between the centralized P2P scheme and the unstructured tree-based scheme is the forming process of the tree. Usually, in the unstructured tree-based scheme, the choice of parent nodes is more randomized. The child node contacts a well-known RP for its initial membership table and then exchanges membership table with neighbors periodically. In the table, the child node selects a set of nodes to be its parent nodes randomly. In contrast, the centralized scheme only assigns one particular parent node to each new node until the leave of this parent node.

The construction of unstructured tree can be built with data-driven and network-driven schemes. Data-driven tree-based schemes include the new CoolStreaming [69], SPANC [70], GridMedia [9], and Substream Trading[71]. These schemes split the video into S substreams. CoolStreaming and SPANC use peers positions in the video to build trees. In CoolStreaming, a peer tries to maintain that it advances at a similar pace in each substream and its parent advances at a similar pace as its neighbors. Therefore, it is easier for an orphaned peer to find a parent and fewer packets will be lost when a peer switches parent. In SPANC, a peer selects the neighbor with the latest position in the video to be its parent to reduce delays. GridMedia and Substream Trading use exchange history. In GridMedia, a peer subscribes to the peer that it has received more packets in the last interval with higher probability. The rationale is that such parents are more capable of supplying packets in the future. In Substream Trading, peers reciprocate with contributing neighbors as a tit-for-tat incentive mechanism.

Network-driven schemes include Narada [36], Gossamer [72], OMNI [73], ChunkySpread [74], Treebone [39], Fastmesh [75], and TreeClimber [76]. Narada, Gossamer, and TreeClimber use a distance-vector routing protocol. Narada and Gossamer use RTTs as the metric, and TreeClimber uses hops. Narada allows any peer to be the video source and each peer maintains a routing table of size $O(N)$. In Gossamer, peers only maintain routes to video sources and hence have a routing table of size $O(s)$ and s is the number of sources. TreeClimber allows a single video source and the routing table has only one entry. In OMNI, ChunkySpread, Treebone, and Fastmesh, peers swap positions with other peers to optimize the tree. In Treebone and Fastmesh, peers swap positions only with ancestors. In ChunkySpread and OMNI, the selection of swapping targets is more flexible.

2.3.3.3 Swam-based schemes

The swarm-based (mesh-based) scheme is a type of fully distributed P2P network. In swarm-based schemes, a peer, by contacting a well-known tracker or a peer already in the video channel, acquires the IP addresses of a list of peers in the system and establishes neighboring relationships. Some other swarm-based schemes also use a distributed membership management protocol, such as the scalable probabilistic membership protocol (SCAMP) [77]. The video is split into chunks of size S_k. Each chunk has a unique sequence number. Each peer maintains a sliding window of recently received chunks, which is called buffer-map. A peer advertises its buffer-map to neighbors immediately after obtaining a new chunk or every interval of length T_d. In swarm-pull schemes, upon receiving a bit-map message or every interval of length T_s, a peer uses a scheduling algorithm to determine which chunks should be requested, and from which neighbors these chunks should be requested. In swarm-push schemes, upon receiving a chunk, a sender peer pushes the chunk to the neighbors that do not have the chunk according to a specific packet scheduling algorithm based on the sender peers local copies of neighbors bit-maps.

In **swarm-pull** schemes, peers will not receive duplicated chunks, and if the playback delay is large enough, a chunk will eventually reach all the peers in time. However, with a limited playback delay, it is possible that a chunk can not reach a peer before the playback deadline. In the swarm-pull scheme, the most important question is to determine the chunk selection strategy and the target selection strategy. It will determine how packets are requested according to its limited bandwidth and importance of packets. An algorithm may select chunks randomly, or prefer chunks that are rare among neighbors (rarest-first), or chunks that have the latest sequence numbers (latest-first), or chunks that are approaching their playback deadlines (most-urgent-first). The target selection strategy is more versatile. A peer may select from neighbors randomly, or select neighbors according to their workload or pair-wise bandwidth and delays.

In **swarm-push** schemes, peers are likely to receive duplicated(uninformative) chunks when the update of neighbors' buffer-map are stale. However, notifying neighboring nodes immediately after receiving a chunk will result in a high volume of overhead. R^2 is the most representative swarm-push scheme. R^2 [78] combines random network coding with random push algorithm. In the peer-to-peer system, it uses random network coding to transmit live video streaming on a swarm-push P2P system. The basic idea of combining the two concepts together is to improve the cooperation between peers when transmitting a live multimedia. Another attractive advantage of the mesh-push scheme is that it can avoid missing important blocks which are caused by the inelegant left of peers. Therefore, this scheme can provide fast recovery when suffering peer churn. This scheme can also improve the performance of live streaming servers, as well as reduced bandwidth costs on dedicated streaming servers because more peers can corporate in the same multimedia delivery.

- **Random network coding**:

In R^2, random network coding technology is used. Each segment is divided into n blocks $[b_1, b_2, b_3, \ldots, b_n]$, and each block has fix number of bytes. When a peer node sends packets to peers, instead of sending this blocks directly, it sends a random combinations of blocks $[\mathrm{b}_1^\mathrm{p}, \mathrm{b}_2^\mathrm{p}, ...\mathrm{b}_\mathrm{n}^\mathrm{p}]$ with the coding coefficients $[\mathrm{c}_1^\mathrm{p}, \mathrm{c}_2^\mathrm{p}, ...\mathrm{c}_\mathrm{n}^\mathrm{p}](m \leq n)$ random choosing from Galois Field 2^8:

$$\mathrm{x} = \sum_{i=1}^{m} c_i^p . b_i^p \tag{2.4}$$

The receiver uses the Gauss-Jordan elimination to decode a segment progressively since it receives the first coded block in this segment. When n coded blocks are received, the original blocks can be immediately decoded.

The sender uses random push strategy to push packets to receivers. When the transmission process starts, the sender node will randomly push packets to receivers based on the buffer-map of the receivers without receivers' request. Senders determine which packet should be sent first according to a packet scheduling algorithm based on the buffer-map. Furthermore, a priority region is defined to make sure a prioritized transmission. Priority region is defined as a transmission region close to the playback point. It is used to makes sure that the urgent and lastest segment can be served first. It shares the same principles as the latest-first and most-urgent-first strategy in swarm-pull based scheme.

The experimental results in R^2 [78] show that the NC-based mesh-push streaming network brings a better performance regarding initial buffering delays, resilience to peer dynamics, as well as reduced bandwidth costs on dedicated streaming servers, compared with a typical live streaming protocol (both without and with network coding).

2.4 P2P Streaming with Network Coding

As introduced in all above P2P networks, the key of P2P network is the packet scheduling algorithm. It determines how packets are transmitted through the network, thereby determining the efficiency of P2P system. In this section, we focus on the past packet scheduling algorithm in P2P network with network coding. Through a detailed analysis, the strengths and weaknesses of each scheduling algorithm are evaluated.

2.4.1 NC in Unstructured Tree-based Network

Network coding was applied to video streaming since 2003 [13]. Since 2005, network coding is widely applied to video streaming area. They present a new model for multimedia streaming between servers and clients not only over the Internet but also over the wireless network [79], [80]. NC can improve the throughput of data multicast while generating rateless erasure codes [81]. In [82], the authors proposed a packet scheduling algorithm to combine network coding with the multi-tree scalable streaming network to achieve a large-scale streaming network. They organize the receivers into layered data distribution meshes and send substreams to each mesh using layered coding. Later in [83], the authors propose an optimization algorithm using a greedy algorithm based on the maximum and minimum flow of each link. They build a global optimal algorithm for each layer and each receiver. Then they build a greedy algorithm by scheduling the lower layer first(greedy algorithm). Later in [70], video packets are divided into substreams and pushed to receivers. The transmission of the NC-recovery packet in loss network is formulated to another delay minimization problem. They prove that network coding brings faster recovery in the tree-based P2P network. In [84], the authors proposed a layered multicast with inter-layer network coding for multimedia streaming. Instead of building multicast tree for each layer, they allow inter-coding among different layers. Therefore, greater flexibility is given in optimizing the data flow, and higher throughput is achieved. However, due to the computation complexity, this model is only tested in a two-level multicast tree. In [85], they propose a joint network flow control and performance optimization problem. The centralized flow control problem is decomposed into a two-level optimization problem, and this optimization problem finds the scalable content distribution meshes with minimum path costs for each video coding layer while satisfying the inter-layer dependency.

2.4.2 NC in Mesh-based P2P Network

In [86], Wang and Li apply network coding to a large-scale mesh-pull based P2P network and use the actual network traffic to evaluate meticulously the benefits and tradeoffs involved in using network coding in P2P streaming. They proved that network coding makes it possible to perform streaming with a finer granularity, which reduces the redundancy of bandwidth usage, improves resilience to network dynamics, and is most instrumental when the bandwidth supply barely meets the streaming demand. In 2007, in [78], the same authors applied network coding to a mush-push based P2P network and proved that network coding brings dramatic better throughput and less delay compared with traditional routing method and network coding on mesh-pull based P2P network. Later in [87], the advantages of network coding is examined in an over 200,000 peers network with realistic assumptions of system scale, peer dynamics and upload capacities and under some extreme dynamics such as a flash crowd scenario. They proved that network coding can provide high playback qualities, short initial buffering delays, resilience to peer dynamics, as well as minimal bandwidth costs on dedicated streaming servers. Later in [15], network coding was firstly applied to VoD streaming system for the Summer Olympic Games in August 2008. They propose an "early-braking" mechanism to reduce linear independent block.

2.4.3 NC with Scalable Video Coding

Scalable Video Coding (SVC), [88] also referred to as Layered Coding (LC) responds to a multitude of challenges imposed by media broadcasting. SVC standardizes the encoding of a high-quality video bitstream that is consist of one or more subset bitstreams. A subset video bitstream is derived by dropping packets from the larger video to reduce the bandwidth required for the subset bitstream. The subset bitstream can represent a lower spatial resolution (smaller screen), lower temporal resolution (lower frame rate), or lower quality video signal. The scalable video coding is very desirable for Internet streaming because most networks are typically characterized by a wide range of connection qualities and receiving devices. The variety of end servers with different capabilities ranging from cell phones with small screens and restricted processing power to high-end PCs with high-definition displays. Scalable video coding is widely used in common P2P network as well. The common scheduling methods include layer first, order first, and mixed first [89]. Layer first means that the lower layers always have higher priority than higher layers. Order first means that generations close to playback point have higher priority than subsequent generations. Mixed first means that the priority follows the Zig-Zag order. Many optimization works are proposed to improve the performance of scalable video coding in NC-based streaming network [90], [91], [92], [93],[94]. Furthermore, the network conditions of end users also vary. Therefore, it brings a great challenge to design an efficient and fair scalable video delivery system. In the following paragraphs, some optimization works for scalable video transmission are proposed. These packet scheduling algorithms are classified according to their objectives.

- **Improving transmission efficiency** Some researches are proposed to improve the transmitting coding efficiencies. In [95], authors proposed to apply network coding to allow a hierarchical network coding. They push the hierarchically encoded packets to receivers based on probability. Therefore, lower layer can get a higher recovery probability compared to other enhancement layers. The disadvantages of their work are that sometimes data cannot be recovered at the end user due to insufficient packets at end users. The research in [96] shares the same principle. They transmit packets based on the rank of each encoded packets. Generations and layers are sorted according to the number of packets that they need to decode and the timeliness of the corresponding generations and layers. The generation and layer which have a higher rank will be transmitted first. The similar strategy is also used in [97]. In their work, each packet is sorted according to their rarity and their timeliness. Packets with higher rank are sent first.

The authors in [98] developed a scalable video streaming on P2P with network coding algorithm. In their work, they implement a P2P network as an overlay network, and then the scalable video is transmitted using H.264/SVC over the overlay network. Instead of requesting packets from a different layer, they only request packets to a target layer to minimize the uninformative transmission.

- **Improve QoS in lossy network** Some other researches are proposed to provide extra protections to scalable data in a lossy network. In [99], authors proposed an unequal packet loss protection scheme. They proved that the redundant network coding avoided the re-transmission of lost packets and improved error correcting capabilities of lost packet. It proved that network coding could bring better quality in the lossy network. In [100], the authors address the problem of prioritized video streaming over a lossy pull-based mesh network. They propose to exploit network path diversity via a novel randomized network coding approach that provides unequal error protection to the packets conveying the video content. They design a distributed receiver-driven streaming solution where a client requests packets from different classes from its neighbors. The process of choosing a network coding strategy can be cast as an optimization problem which determines the rate allocation between different packet classes such that the average distortion at the requesting peer is minimized. The algorithm outperforms reference schemes with better video streaming quality. The disadvantages are that their work only considers the video quality and did not consider the network resources.

- **Improve QoS in inter-session scenario** In [101], a distributed rate allocation algorithm that minimizes the average decoding delay for multimedia clients in inter-session network coding systems. In [102], they proposed a decoding delay minimization algorithm with inter-session network coding. They formulate the problem as the minimization of the average decoding delay in the client population and solve it using simultaneous perturbation stochastic approximation algorithm, which is a gradient based stochastic algorithm for finding a good approximation of the global optimum in the multivariate non-convex optimization problem. In [101], the authors proposed a distributed rate allocation algorithm that minimizes the average decoding delay for multimedia clients in inter-session network coding systems, where network users determine the coding operations and the packet rates to be requested from the parent nodes, such that the decoding delay is minimized for all clients.

- **Improve QoS using reinforcement learning approaches** In [103], the authors formulate the optimal coding and scheduling decisions problems with the help of Markov Decision Processes(MDP). After that, reinforcement learning approaches are then proposed to derive reduced computational complexity solutions to the adaptive coding and scheduling problems.

2.5 Evaluation Framework

In the literature, some researches evaluate the Quality of Service (QoS) live streaming system from 2 aspects: the playback quality, and the delivery ratio. Due to the property of live streaming, transmission delay is less used as a criterion to QoS, because the deadline for each GOP has been set in advance. Once the corresponding packets cannot arrive at the receiver node before the playback deadline, they are treated as useless packets and discarded. Some aspects are also important to evaluate the streaming system, including goodput efficiency, user fairness, network scalability. Furthermore, the space complexity and the computation complexity is also very important to the design of the system. In this section, we present several important indexes for system evaluation.

Playback Quality Playback quality is a very important criteria to judge the success of a video transmission. Usually it is clearly related to the visual experience of the end user. We usually measure the following formula. **Average PSNR** The Peak Signal-to-Noise Ratio(PSNR) is defined as [104]

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{2.5}$$

Where, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. For color image with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three.

The Mean Squared Error (MSE) is calculated as followed. Where I(m,n) is the intensity of the luminance components at the pixel(m,n). m,n are the coordinate of the pixel. M and N are the number of pixels in a row and in a column respectively.

$$MSE = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(I'(m,n) - I(m,n)\right)^2 \tag{2.6}$$

While PSNR for a full-color video can be calculated for all the chrominance component, some works only use the luminance component to measure their PSNR value. The average PSNR means that it is calculated over every frame and at all receivers, and then it is averaged over the whole playback period.

Goodput Efficiency The goodput efficiency is evaluated by the uninformative packet ratio. It is calculated as the fraction of uninformative packets over all the received packets. It is an index of coding diversity and overhead that is introduced by uninformative packets, and it is calculated as Eq.2.7

$$\eta = \int_{t_0}^{T} \frac{\mu(t)}{r(t)} dt \tag{2.7}$$

where $[t_0, T)$ is the interval under observation for packet arrival events, and $\mu(t)$ and $r(t)$ denote the uninformative packet and received packet over T. It can be further expressed as the sum of received generation in each generation over the period of time:

$$\eta = \sum_{g=g_0}^{g_n} \frac{\mu_g}{r_g} \tag{2.8}$$

In Eq.2.8, μ_g represents the number of uninformative packets in generation g. r_g represents the number of received packets in generation g. g_0 and g_n represent the start point and the end point of observation.

Delivery Ratio The playback skip rate is usually evaluated using the delivery ratio during the observation window from generation g_0 to generation g_n. For each layer, the delivery ratio H_l is calculated as:

$$H_l = 1 - \frac{\sum_{g=g_0}^{g_n} D_{gl}}{g_n - g_0} \tag{2.9}$$

The D_{gl} denotes if the generation g and layer l is decoded. It returns 1 if (g, l) is decodable and returns 0 if (g, l) is undecodable.

2.6 Summary and Discussions

This chapter presents an overview of network coding and the architecture of streaming networks. Firstly, the developments of network coding are introduced and compared. We survey network coding from the basic of network coding, linear network coding, and practical network coding. We identify the benefits of network coding in multimedia streaming including increased throughput, robustness to loss, and easier communication system deployment. We also identify that practical network coding may lead to the uninformative and unrecoverable transmission, thereby leading to poor video quality and bandwidth inefficiency in NC-based video streaming. Secondly, the development of streaming network is reviewed from the basic client-server model to overlay multicast streaming network, with an emphasis on peer-to-peer streaming network. The extensive and detailed review of recent advances on video streaming in P2P network with network coding are also presented. Through comparison of the existing streaming networks, we find that network coding in a mesh-based P2P network can bring better performance in perceived video quality, scalability, initial playback delay, transmission delay, and user fairness. Therefore, we choose to implement and improve the mesh–based P2P streaming network. Later, based on this network, we proposed several scheduling algorithms to solve the uninformative and unrecoverable problem in Chapter 4, Chapter 5 and Chapter 6.

It has been proved that network coding brings many benefits to the multimedia transmission network, especially in peer-to-peer network including augmented throughput, reduced vulnerability to packet erasures, minimized transmission delay and ease of deployment in large-scale distributed systems. However, the scalable video transmission (SVT) over the lossy network is still an open challenge in the scheduling system design. Some people use the perceived video quality as a criterion to schedule the packet transmission in each generation. However, it is observed that peers will tend to spend a lot of available upload bandwidth in tiny video quality improvement, which leads to bandwidth inefficiency and a low overall video quality. To cope with this, a rate allocation optimization for scalable video streaming in the mesh-pull streaming system is investigated in Chapter 4, which optimizes the sum of the video quality and the weighted bandwidth redundancy.

The mesh-push streaming network mainly uses the buffer-map of neighbors to decide the packet scheduling algorithm. However, this design will lead to uninformative and unrecoverable transmission for the system. First, the buffer-map updating could be delayed due to the transmission delay. Therefore, the scheduling decision is usually made based on the outdated information, which leads to the asyncrhonized communication among senders, and brings uninformative transmission. Second, although it has been proved mesh-push streaming system brings less communication delay compared to the mesh-pull streaming system, it still suffers from the problem of unrecoverable transmission. The random-push and the scheduling policy ranking system are the most used scheduling algorithm in the mesh-push streaming system, but none of them can avoid unrecoverable transmission. In conventional mesh-push based schemes, it is impossible to avoid the transmission of *uninformative* and *unrecoverable* packets because fully intelligent scheduling requires enormous amounts of overheads for communication which cannot be compensated by the gains achieved. Therefore, we propose a new transmission process with a new distributed packet scheduling algorithm in the mesh-push based streaming system to improve bandwidth efficiency for non-scalable video streaming in Chapter 5.

It is also observed that current scheduling algorithms are mostly build based on the current bandwidth instead of a period of bandwidth, including the algorithms proposed in Chapter 4 and Chapter 5, which leads to the sub-optimization to the packet scheduling algorithm. Therefore, instead of scheduling packet transmission for each generation in the priority region one by one, a multi-generation scheduling algorithm is proposed in Chapter 6 to bring better bandwidth utilization, and therefore better bandwidth efficiency.

Chapter 3

Video Streaming System Architecture

In this chapter, we introduce the mesh-based P2P streaming network architecture with network coding. As summarized in Chapter 2, mesh-based P2P streaming network with NC brings a better performance regarding the initial buffering delay, resilience to peer dynamics, as well as reduced bandwidth costs on dedicated streaming servers, compared with typical live streaming protocols (both without and with network coding). The key of the P2P streaming network is the packet scheduling algorithm. It determines the bandwidth efficiency and the QoS at the receiver node. Therefore, to provide a fair comparison of scheduling algorithms, we propose a mesh-based streaming system to simulate the streaming network in the real world. First, we model the real world streaming network into a network model, which is depicted in Fig.3.1. Three types of peers are defined in this model: the tracker node, the streaming server, and the client node. Second, a randomly connected network typology is used to represent the real mesh network. In the simulation, the bandwidth, the loss rate, and transmission delay are considered. Third, we implement the main communication processes between nodes for streaming. Fourth, a buffer model is implemented to storage encoded data. At last, the random network coding and decoding model are presented for non-scalable and scalable data transmission.

3.1 Architecture of the System

In this section, a network model and a block diagram are presented. The network model represents the architecture of the streaming system, and the block diagram shows the main function of the streaming application run at each client node. The network mode is depicted in Fig.3.1. Three types of nodes in the network are defined as bellow.

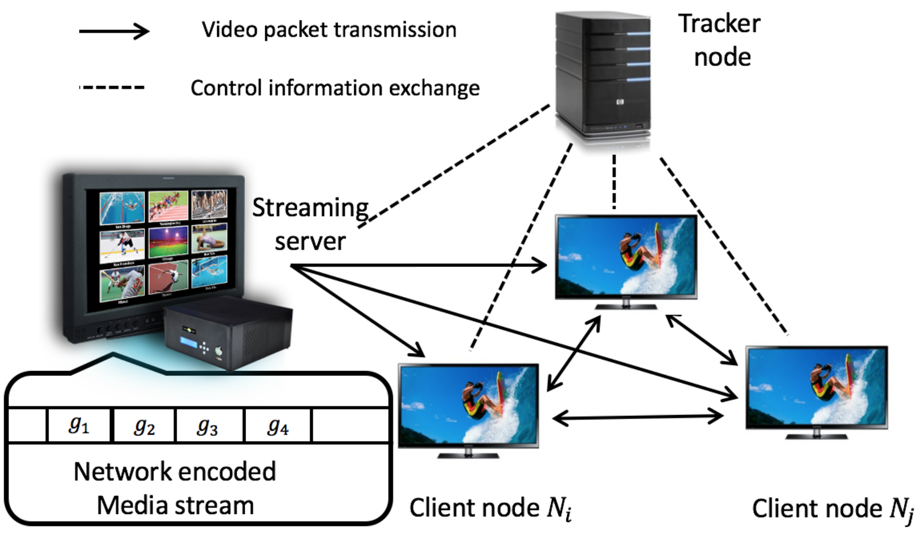


Figure 3.1: The simulation of a swarm network with a tracker server, a streaming source and client nodes.

- **Tracker Node:** The tracker node does not transmit any coded video packet, but only serves the purpose of enabling peers to find each other and exchange control messages.
- **Streaming Server:** The streaming server owns the original video data and transmits network-encoded video packets to client nodes. The streaming server could be any node which holds the original video packets. It could be the video server

or the data center in the CDN network. Before transmission, it divides the video data into several *generation*s and further divides *generation* into several blocks.

- **Client Node:** Client nodes are the end users in the P2P network. They connect the tracker server to form a neighbor relationship. Client nodes download the data from the streaming server and other client nodes at the first place. When client nodes hold a certain amount of data in their data buffer, they become sender nodes to server other client nodes.

The swarm-push P2P communication system is developed based on the UDP protocol. The prototype of the NC-based swarm-push streaming network is proposed in R^2 [78]. R^2 uses *generation*-based streaming. It introduces the use of a priority region and *buffer-map* messages that allow transmission in a given time constraint, thereby leading to low transmission delay. Our streaming system is composed by an overlay of nodes running the application shown in Fig. 3.2. The main parts include the data buffer, the buffer-map, the packet coding and decoding component, and the packet scheduling component.

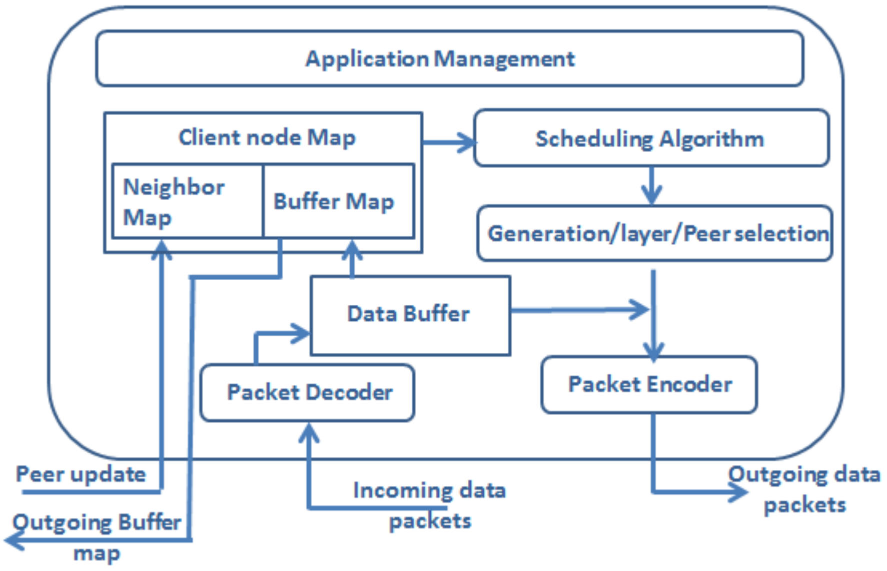


Figure 3.2: Block diagram of the P2P application run on the each client node

- **Data buffer:** When a node receives a video packet, the video packet goes through the packet decoder to check if the packet can be decoded. Then the decoded or undecoded video packet is stored in the data buffer. According to the type of video, the data buffer can be structured as a 1D array and a 2D array. When a client node plays a generation g, all packets belong to the generation g need to be decoded in advance for the successful playback.

- **Neighbour-map:** When a node joins the network, it finds its neighboring nodes by contacting tracker node. Some control information is exchanged among neighbors, including the upload bandwidth of neighbor nodes and the loss rate. The neighboring information is updated every τ second, and when the peer node experiences a big change in its neighboring network, like an inelegant leave of parent nodes and the severe vibration of available bandwidth.

- **Buffer-map:** It holds the buffer-map of its neighboring nodes. The buffer-map updates periodically. For each neighboring node, a 1D array is used to represent the buffer-map of a non-scalable data, and a 2D array is used to represent the buffer-map of a scalable data. The buffer-map updated every T seconds. When T is too long, the system transmits too many uninformative packets. While when T is too short, the system wastes its bandwidth in transmitting these buffer-maps.

- **Packet coding and decoding:** Client nodes decode its received packets in the data buffer using Gauss elimination when a new packet is received. Client nodes and the streaming server also encode the packets in their data buffer before sending a new packet to other client nodes.

- **Packet scheduling component:** The packet scheduling component determines how data is transmitted from this client nodes to other client nodes. It is the most important component to the network performance. It determines which generation and which receiver the packet is addressed to. For scalable video streaming, the packet scheduling algorithm determines which generation, which layer and which receiver the packet is addressed to. In our system, the packet scheduling component uses the neighbors' buffer-map or the neighbors' upload bandwidth to achieve a more accurate delivery.

- **Generation/Layer/Peer Selection:** When the scheduling algorithm determines the target generation, the target layer, and the target peer, this component identifies the corresponding packets in the data buffer, encodes these packets, and transmits the encoded packet to the receiver according to the scheduling result.

Our application is implemented based on two previous data delivery systems. One is a network encoded data delivery system called Zeta-protocol [105], and the other is a live streaming system [98]. ZetaSim extends the GnutellaSim protocol [106]. ZetaSim is implemented in the network simulator(NS2) and uses the messaging, the socket structure, and the UDP datagram for feedback-free transmission. Sanna [98] improves the ZataSim and provides a live streaming system[98]. Based on the ZetaSim, Sanna further implements the use of generations, the synchronized sliding window, and prioritized coding in their work. Based on Sanna's work, the buffering model, packet scheduling component, and communication process are further improved in our work. By modifying these components, the system allows more flexible and efficient packet scheduling algorithm, which brings better system performance.

3.2 Network Setup

A number of random topologies is generated for the system set-up. A geographic model for network growth emulation is used. This model is called the Waxman model [107] and the software implementation is available at [108]. This implementation allows the creation of a flat plane, where a number of nodes are placed according to a 2D Poisson process. Links are then created between any pair of nodes u and v, with probability Eq.3.1:

$$\mathrm{P(u,v)} = \alpha \mathrm{e}^{-\frac{d(u,v)}{bL}} \tag{3.1}$$

where $\alpha > 0$ and $b \leq 1$, d is the Euclidean distance between u and v, and L is the maximum Euclidean distance between any two nodes (given by the plane size). The parameter α drives the probability of having a link between any pair of nodes, whereas the parameter b increases the probability of having long edges on the probability of having links between closer nodes. Fig. 3.3 shows a few examples of randomly generated networks, the top line showing 20 nodes networks, and the bottom line showing a 50 nodes network, as used in our experiments.

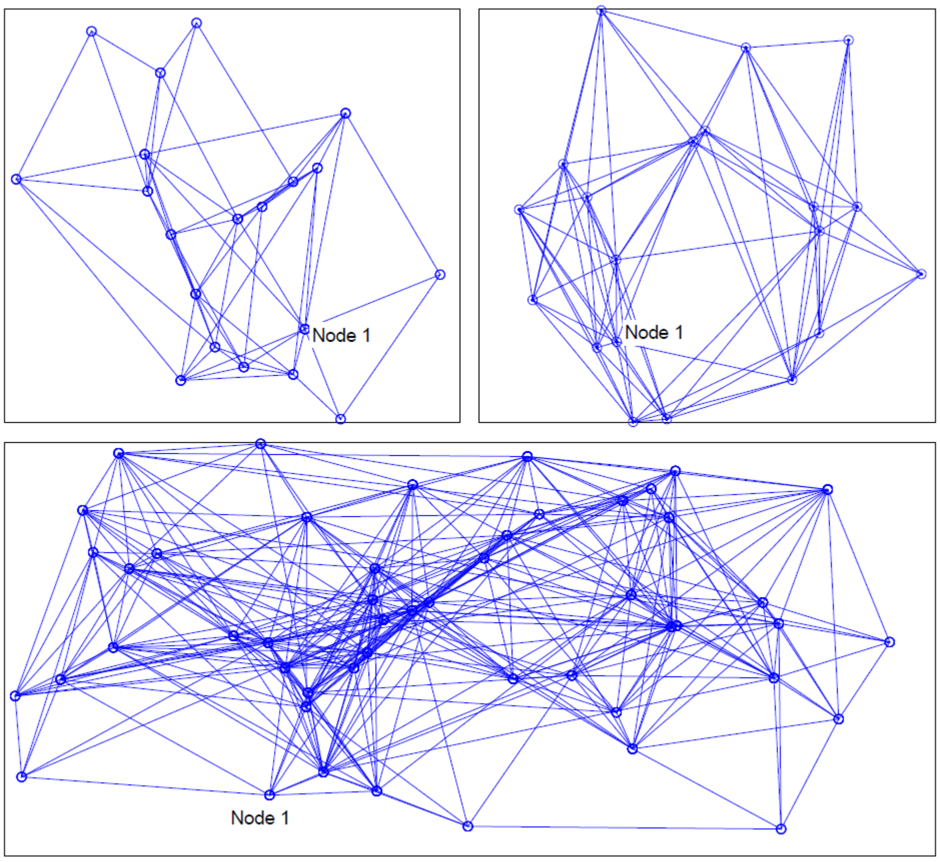


Figure 3.3: Randomly generated network topologies. Top: 20 nodes; left: β = 0.1, α = 5; right: β = 0.1, α = 10 ; Bottom: 50 nodes.

3.3 Main Communication Process

In this section, the main communication processes are presented in the swarm P2P network to achieve the video streaming.

P2P protocol procedure: JOIN The JOIN procedure is shown in Fig. 3.4 for both *client node*s and the *streaming server*. When a node joins, the *tracker node* adds this node to the node list, allowing the user to perform further actions. The *streaming source* then sends an *UPDATE_ STREAMS* message to inform the *tracker node* of the availability of the video. Then the streaming server is put in the list of *SEEDING_ PEERS*.

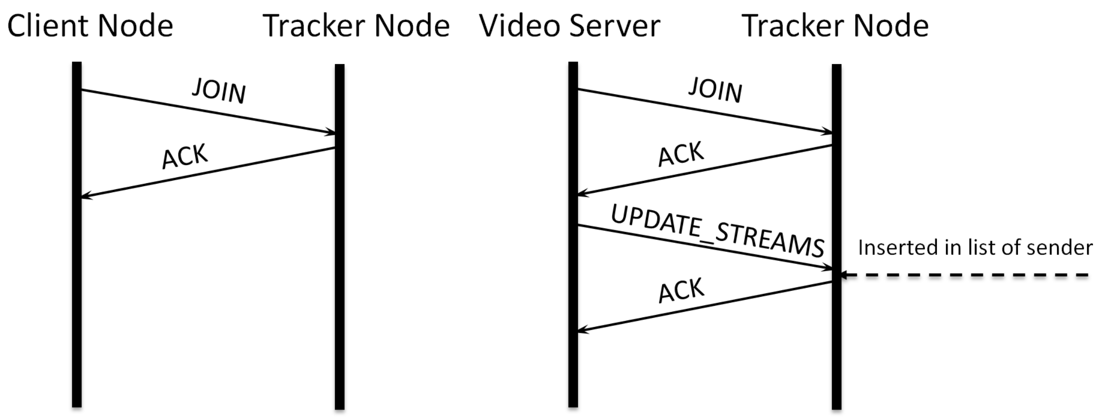


Figure 3.4: P2P protocol JOIN procedure

P2P protocol procedure: QUERY When a client node in the node list wants to receive the video, it sends a QUERY message to the *tracker node*. The query is either replied by a *QUERY_HIT* message, or it is put in the list of unfulfilled queries (in case there are no seeders for the requested video) and is replied at a later stage once seeders become available. When the client node receives the *QUERY_HIT* message, the *client node* becomes a downloader and informs the tracker node. Then the tracker node put the client node in the list of downloaders. This procedure is described in Fig 3.5.

P2P protocol procedure: START Fig. 3.6 shows the process that peers start

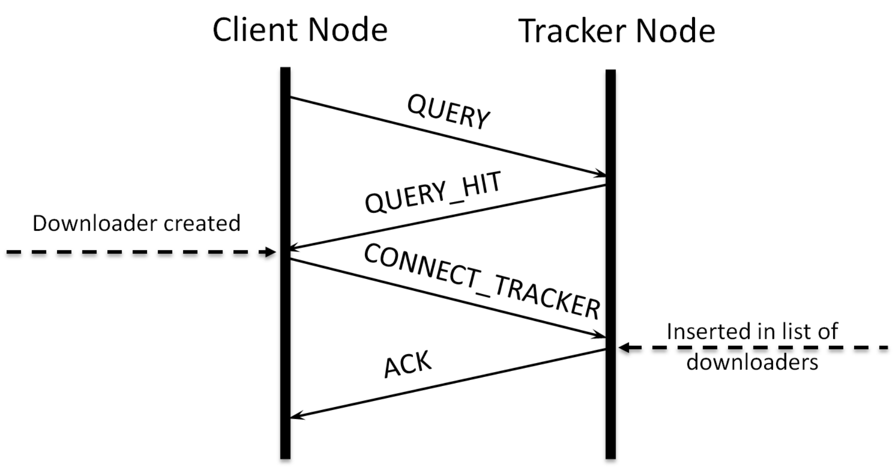


Figure 3.5: P2P protocol QUERY procedure

streaming packets. First, when a node(streaming server or client node) becomes a sender node N_i, N_i will send a message to the tracker node. Then the tracker node will put node N_i into the sender list. When a new downloader N_j joins the network, the tracker node sends the information of N_j to N_i, and the node N_i sends a message to N_j, then the parent relationship between N_i and N_j is established. Based on the information of N_is at N_j, a signaling message is sent from N_j to N_i to trigger the transmission. When the node N_j reaches its seeding buffering threshold, it becomes a sender node automatically.

The information exchanged between N_i and N_j varies from different packet scheduling algorithms. The buffer-map of the sender node N_i, the buffer-map of receiver node N_j, the upload bandwidth between N_i and N_j, and the loss rate between N_i and N_j are some common information exchanged during this process. In the system introduction, we only use **SIGNALLING_MESSAGE** to represent this step. In the later chapters, the details about the information exchanged during this step are elaborated separately.

3.4 Priority Region

A sliding window defines, at each instant of time, a priority region, or playback buffer (Fig. 3.7). Based on a synchronized clock and decoding/playback deadlines defined for

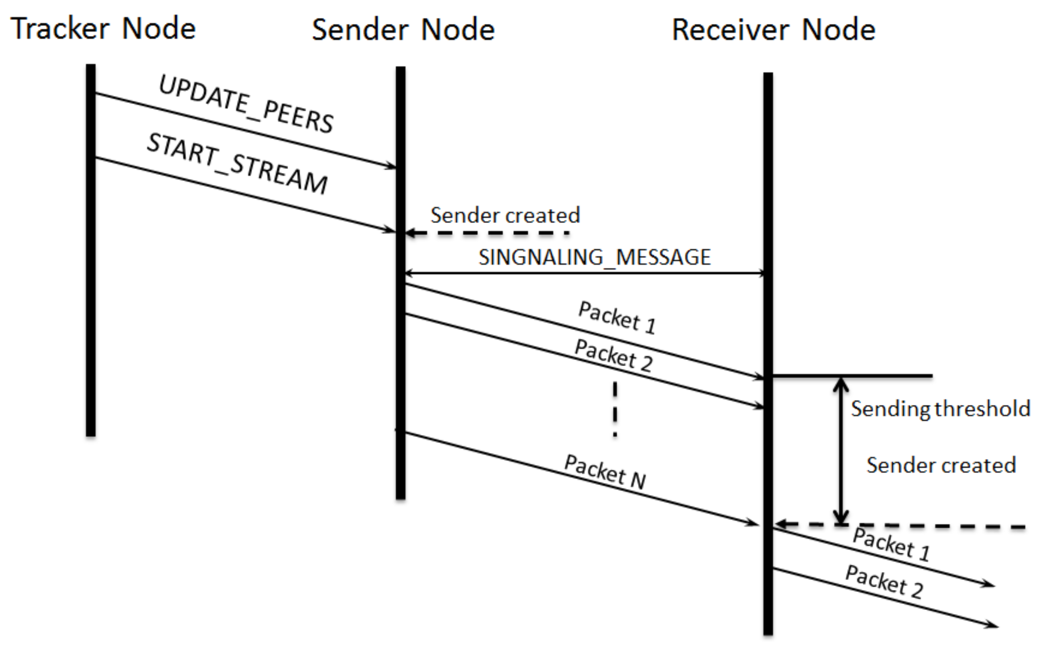


Figure 3.6: P2P protocol START procedure

every group of picutres (GOP), it is possible for peers to determine at every instant which GOPs are to be buffered and retransmitted. We define Γ_s as the start of the priority region, and Γ_e as the end of the priority region, and Γ_l as the length of the priority region in number of generations, so that at each instant in time the sliding window includes all generations with indices:

$$GOP = [\Gamma_s, \dots, \Gamma_e] \tag{3.2}$$

Fig. 3.7 (top) shows a sliding window for non-scalable video. When the deadline for the oldest GOP expires, the window slides forward. If the GOP has been completely received, it can be decoded and played back. Otherwise, it will be skipped. Fig. 3.7 (button) shows a sliding window for scalable video. Each generation is played at the maximum level of quality that has been received. One of the goals of our system is to make sure that as much data as possible is buffered within the sliding window to improve

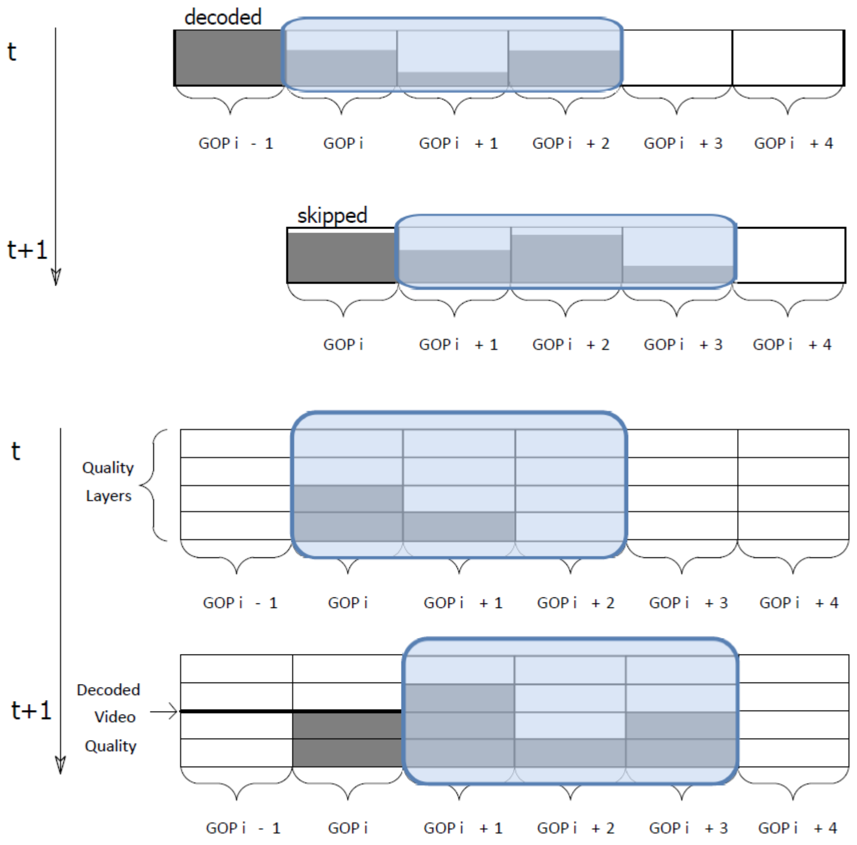


Figure 3.7: Sliding window with non-scalable data(top) and scalable data(bottom) at two successive instants of time

the video quality.

The priority region implicitly defines an allowed playback delay or buffering delay. Such delay is of the order of seconds, which is compliant with the specification of commercial adaptive streaming protocols (2 seconds for HLS, 10 seconds for Smooth Streaming). The buffer size also determines the occupation in the device memory, ranging from 400-700 KBytes for 2 seconds of HD and full HD H.264 video, respectively, to 2-3.5 Mbytes for 10 seconds of the same videos.

3.5 Network Encoding and Decoding

In our system, each packet is re-encoded before forwarding to the next node. Therefore, each packet goes through the packet encoder module before it is sent to the next node, and each packet in its data buffer goes through the packet decoder module before the playback. The whole encoding and decoding system is shown in the Fig. 3.8.

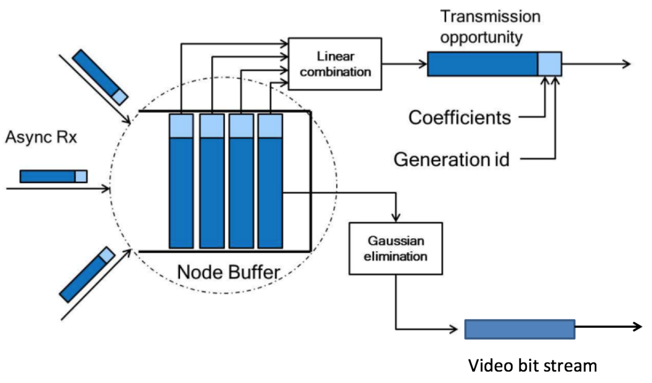


Figure 3.8: Illustration of the packet encoding and decoding process at the client node

3.5.1 Packet Encoder

The transmission opportunity arises at the sender node every T seconds. The transmission interval T is equal to the size of transmitted packet over the upload bandwidth. When a transmission opportunity arises at the peer node, the peer encodes a packet from the buffer and sends it to a chosen destination. The vector of coding coefficients and the generation ID are embedded in the packet header, to allow other receivers to build a matrix with the respective encoding vectors and retrieve the source data. According to the type of video, we used two network coding methods. One is random linear network coding(RLNC), and the other is the hierarchical network coding(HNC). The details about RLNC and HNC are introduced in the next paragraphs.

3.5.1.1 Random Linear Network Coding

Our network coding system, as well as many other systems, uses generation-based buffering and coding [13]. Fig.3.9 shows the segmentation hierarchy from GOP level to data blocks. Network Abstraction Layer (NAL) units are grouped into non-overlapping segments of data, called generations. Each generation is then divided into blocks, which are combined to produce outgoing packets. We use the Galois coding field of size 2^8, which has been proved enough for encoding video packets in [13]. We define a packet length as V Bytes. Having chosen generation size of S and a packet size to accommodate blocks in User Datagrams Protocol (UDP) packets, a generation can be segmented into $\varpi = \lceil S/V \rceil$ blocks.

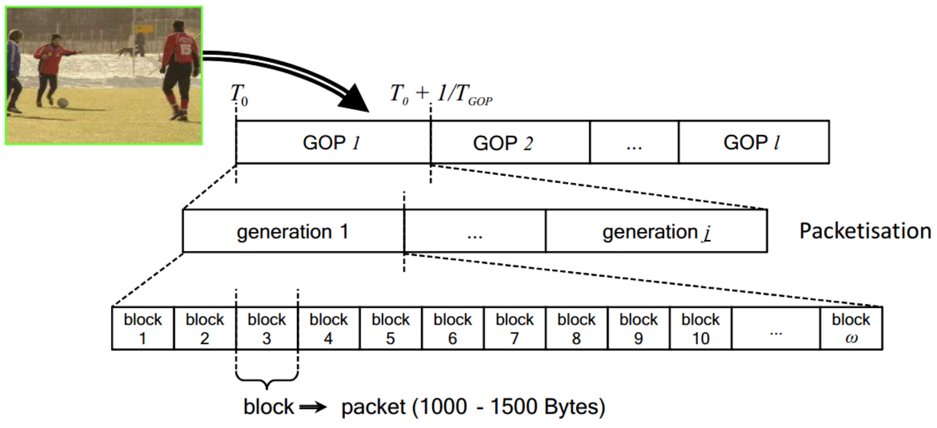


Figure 3.9: The structure of the generation

One packet is generated by combining the the blocks belonging to one generation with some coding coefficients. In other words, considering a generation composed of blocks $b_j, j = 1, ..., \varpi$, a new encoded packet can be written as:

$$y = \sum_{i=1}^{\varpi} c_i.b_i \tag{3.3}$$

Algebraic operations are performed in a Galois Field of size $q = 2^n$, with $n = 8$, and $c_i \in GF(q), i = 1, ..., q$ are the random coefficients generated for the specific outgoing packet.

3.5.1.2 Hierarchical Network Coding

The following formula is the hierarchical network coding function, which is developed based on the characters of scalable video coding. In the scalable video coding, a generation will be further divided into L layers. As shown in the Fig.3.10, in the first hierarchy of HNC, blocks in the first layer are encoded together. In the l layer, packets from the first layer to the l layer are encoded together. The encoding process can be represented using Eq.3.4.

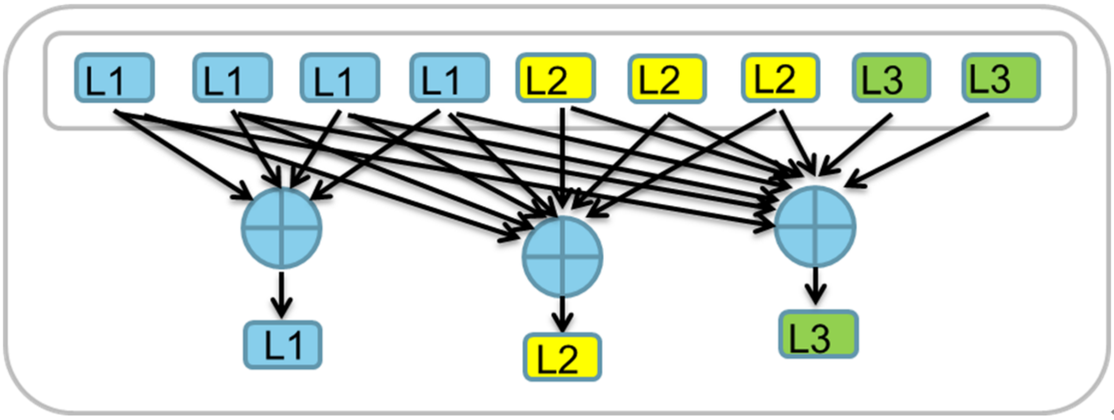


Figure 3.10: Arrangement of blocks from different layers in the buffer

$$\text{Hierarchy layer} \quad l: \qquad y_{gi} = \sum_{j=1}^{N_{g1}+N_{g2}+\ldots+N_{gl}} b_{gj} c'_{ji} \tag{3.4}$$

In this algorithm, N_{gl} is the number of packets in generation g of layer l , b_{gj} is the jth raw blocks in generation g, and y_{gi} is the ith encoded blocks in generation g, and c'_{ji} is the coding coefficient, which are randomly generated from the Galois field F_q (usually $q = 2^8$).

3.5.2 Packet Decoder

To accelerate the decoding process, the system uses the progressive decoding technique. Instead of decoding after collecting enough packets, a downloader tries to pre-decode the received packets each time when it receives a packet. First, the downloader cascades the coefficient vectors of all the received packets. These coefficient vectors form a matrix. Then, the downloader tries to keep this matrix as an upper triangle form. Each time when a new block arrives, the downloader will check whether it is linearly independent with existing packets. If it is, it will add the new coefficient vector as the last row.

If a node owns m linearly independently blocks, it will be able to decode this generation. It first forms a $m * m$ coefficient matrix $\mathbf{C}$, Each row in coefficient matrix $\mathbf{C}$ is the coefficient vector of one received combination. It then will be able to recover the original data using Eq.3.5. Where C^{-1} is the inverse matrix of the received coefficient matrix. Y is the vector of encoded video packets.

$$b = C^{-1}Y \tag{3.5}$$

3.6 Summary

In this chapter, we propose the mesh-based P2P streaming model with network coding. This system model is the P2P mesh network. First, a random network typology is proposed, where any pair of nodes are linked according to a probability. At any node in the network, a streaming application is run to achieve the overall video streaming. The streaming application can be divided into six components. The functionality of each component is introduced, and the connections among these components are presented as well. Though the introduction to these components, the work process of the streaming system is presented. Then we introduce the main communication process among peers in the system. Based on this communication process, a mesh-based streaming system is built. Furthermore, the priority region and the network encoding and decoding component are also elaborated to achieve NC-based streaming.

In the next chapters, we proposed several packet scheduling algorithms based on this streaming system. According to the design of algorithms, the information updating and packet transmission procedures may differ in each chapter. These designs will be introduced separately the following chapters.

Chapter 4

Optimized Scalable Video Transmission with Hierarchical Network Coding

The scalable video transmission with network coding in mesh-based network is attractive, since this scenario can bring better performance in video quality, transmission delay, throughput, network reliability for heterogeneous peers [109]. The key to the transmission model is the packet scheduling algorithm at each client node. Therefore, this chapter proposes an optimized rate algorithm (ORA) to optimize the video quality and the weighted bandwidth redundancy in the NC-based lossy streaming network for scalable video transmission. The proposed ORA is developed to achieve the trade-offs between video quality and the weighted bandwidth efficiency for each generation, which can lead to a better overall video quality for all generations and a relatively low bandwidth redundancy for the transmission system.

4.1 Motivation

The scalable video transmission (SVT) over the lossy network is still an open challenge in the P2P network. When network coding(NC) is combined with SVT, more challenges are brought to the design of the scheduling algorithm. The first problem is the unequal importance of each video layer. In SVT, packets are grouped into layers in each generation, and the lower layers are more important than the enhancement layers. Therefore, the packet scheduling algorithm should give more protection to the lower layers than the enhancement layers. However, this unequal error protection may lead to the unrecoverable transmission of the enhancement layers in the NC-based streaming network, which, in turn, leads to poor overall video quality in the generation. To avoid this problem, the packet scheduling algorithm should carefully schedule upload bandwidth allocation among different layers to achieve the best overall video quality in each generation.

The second problem is the system redundancy. Because the receiver node estimates the video quality based on a probability model, the receiver node will choose to transmit as many redundant packets as possible for a tiny video quality improvement when the video quality is used as the only criterion. However, it will lead to a severe bandwidth inefficiency, which is undesired in the packet scheduling design. First, the bandwidth inefficiency will burden the whole P2P network, and lead to poor QoS for other client nodes. Second, the bandwidth inefficiency in this generation will lead to a poor video quality in the next generations when the cumulative upload bandwidth is used to schedule packet transmission. Therefore, it is necessary to take both video quality and bandwidth redundancy into consideration to achieve a better overall video quality and system performance.

4.2 System Model

The P2P streaming model is the system model proposed in Chapter. 3. Based on this streaming model, some definitions are given to formulate problem. In the network, each receiver node N_j have some sender nodes $N_i \in A_j$, where A_j represents the group of senders of the receiver node N_j. The size of A_j is $|A_j|$. The upload bandwidth between N_i and N_j is donated as U_{ij}, and the estimated loss probability is donated as p_{ij}. The number of packets can be sent from all senders to the receiver is $N_p = \sum_{i=1}^{|A_j|} U_{ij}/V$, where V is the size of encoded video packets. If the last packet is smaller thant V, it will be sent without padding with zeros.

4.3 Hierarchical Network Coding

Our network coding system uses generation-based buffering and network coding [13]. A generation can be further divided into many blocks. A new coded packet is generated by combining the blocks belonging to one generation together with random coefficients. The algebraic operations are performed in a Galois field of size $q = 2^m$, where m is 8 in our experiment. The following formula is the proposed Hierarchical Network Coding (HNC). In each quality layer l of the hierarchy, the blocks from the base layer to the target layer l are encoded together. The HNC equation is in Eq. 3.4.

4.4 Optimized Rate Allocation

In this section, we propose an optimized rate allocation optimization algorithm. In this algorithm, the estimated distortion and the weighted bandwidth efficiency ratio is optimized at the same time. By finding the tradeoff between distortion and transmission redundancy, we can find the optimal rate allocation policy.

The packet delivery protocol works in two steps. Firstly, each sender node N_i allocates its upload bandwidth U_{ij} to the receiver node. Based on the information of $U_{ij}, \forall i \in A_j$, each receiver calculates the ORA to achieve the rate allocation for each generation. The rate allocation determines the rate allocation for each layer in each generation. Secondly, the receiver node informs each sender node the allocation results. Each sender node allocates its upload bandwidth based on the rate allocation distribution and transmits packets to the receiver. The rate allocation algorithm is calculated every Γ_l, where Γ_l is the length of the priority region. In the case that a parent node does not have the request packet, it will uniformly allocate its bandwidth to other quality layers. Furthermore, if the receiver node can not get enough packets before the playback deadline, it will update its buffer-map to all its parent nodes and server node. When its parent nodes have available upload bandwidth, they will reply to this request. When the receiver's request is targeted, it sends a stop message to parent nodes to stop the transmission.

4.4.1 Problem Formulation

We formulate the rate allocation problem by involving two important optimization objectives: distortion and bandwidth redundant ratio. Our objective function is defined as:

$$F(\boldsymbol{a_g}) = \varphi(\boldsymbol{a_g}) + \omega\Phi(\boldsymbol{a_g}) \tag{4.1}$$

Where $\boldsymbol{a_g} = [a_{g1}, a_{g2}, ..., a_{gL}]$ and a_{gl} represents the number of packets allocated to generation g of layer l. The function $\Phi(\boldsymbol{a_g})$ calculates the bandwidth redundancy ratio of the system based on the allocated bandwidth $\boldsymbol{a_g}$. The optimization coefficient ω is a relative weight to balance the tradeoff between the video distortion and the bandwidth redundant ratio. The function $\varphi(\boldsymbol{a_g})$ is the distortion that the end user perceived. It is defined as the difference between the Peak Signal to Noise Ratio(PSNR) of the full rate video and the PSNR gain if $\boldsymbol{a_g}$ are sent. The optimal allocation $\boldsymbol{a_g}^*$ can be solved through finding the solution to minimize the function $F(\boldsymbol{a_g})$ as shown in Eq.4.2. The first constraint means that the number of scheduled packets should be smaller than the available upload bandwidth N_g in the generation g. The N_g is the cumulatively available upload bandwidth. It is equal to N_p at the first generation. The second constraint means that $\boldsymbol{a_g}$ should be nonnegtive.

$$\begin{aligned} \boldsymbol{a_g}^* &= \arg\min_{\boldsymbol{a_g}} F(\boldsymbol{a_g}) \\ \text{subject to} \sum_{l=0}^{N_l} a_{gl} &\le N_g \\ \boldsymbol{a_g} &\ge 0 \end{aligned} \tag{4.2}$$

4.4.2 Distortion Analysis

First, according to the loss rate p_{ij} and upload bandwidth U_{ij} of each sender node N_i, the packet loss probability for this receiver N_j can be calculated by Eq.4.3, where $|A_j|$ is the size of the neighbors of each receiver node N_j:

$$p_j = \frac{\sum_{i=1}^{|A_j|}(U_{ij}p_{ij})}{\sum_{i=1}^{|A_j|}U_{ij}} \tag{4.3}$$

Then, the distortion can be estimated as the difference between the Peak Signal to Noise Ratio(PSNR) of the full rate video d_L and the estimated PSNR gain $R(\boldsymbol{a_g})$ if $\boldsymbol{a_g}$ is sent from senders $N_i \in A_j$ to receiver N_j. d_L is the PSNR gain of the video if all original packets are received and decoded.

$$\varphi(\boldsymbol{a_g}) = d_L - \psi(\boldsymbol{a_g}) \tag{4.4}$$

The $\psi(\boldsymbol{a_g})$ in Eq. 4.4 can be further represented by the sum of the estimated PSNR gain at each layer $G(\boldsymbol{a_g}, l)$ if $\boldsymbol{a_g}$ are sent from senders:

$$\psi(\boldsymbol{a_g}) = \sum_{l=1}^{L} G(\boldsymbol{a_g}, l) \tag{4.5}$$

where $G(\boldsymbol{a_g}, l)$ represents the estimated PSNR gain of each layer l, L represents the total number of layers in the stream.

In the lossy network, although $\boldsymbol{a_g}$ packets are sent to the receiver, only $\boldsymbol{r_g}$ can be received at the receivers due to the network loss. The actual PSNR gain can only be accurately calculated based on the number of received packets at the receiver N_j. The function $G(\boldsymbol{a_g}, l)$ can be further represented as Eq. 4.6

$$G\left(\mathbf{a_g}, l\right) = \sum_{r_{g1}=0}^{a_{g1}} \sum_{r_{g2}=0}^{a_{g2}} \cdots \sum_{r_{gL}=0}^{a_{gL}} h(\mathbf{a_g}, \mathbf{r_g}) d_l(\mathbf{r_g}) \tag{4.6}$$

The Eq.4.6 represents estimated PSNR gain if $\boldsymbol{a_g}$ packets are sent to the receiver. $\boldsymbol{r_g} = [r_{g1}, r_{g2}, ..., r_{gl}]$. r_{gl} represents the number of the received packets in the generation g of layer l. The function $h(\boldsymbol{a_g}, \boldsymbol{r_g})$ represents the probability that $\boldsymbol{a_g}$ packets are sent and $\boldsymbol{r_g}$ packets are received. The values of d_l is the quality improvement or distortion reduction in layer l after the receiver node receive $\mathbf{r_g}$ packets. It depend on the encoding parameters, e.g., the quantization values, employed by the scalable video codec. It also depends on if $\mathbf{r_g}$ is enough for decoding the the layer l in generation g. The function $h(\boldsymbol{a_g}, \boldsymbol{r_g})$ can be further represented by the Eq.4.7

$$h(\mathbf{a_g}, \mathbf{r_g}) = \prod_{l=1}^{L} \begin{pmatrix} a_{gl} \\ r_{gl} \end{pmatrix} (1 - p_j)^{r_{gl}} {p_j}^{a_{gl} - r_{gl}} \tag{4.7}$$

4.4.3 Bandwidth Redundancy Ratio Analysis

The bandwidth redundancy ratio can be represented in Eq.4.8. It is equal to the ratio of the number of redundant packets over the number of actually transmitted packets.

Algorithm 1: Optimized Rate Allocation Algorithm

```
Input: $N_g$, $p_j$
Output: $\boldsymbol{a_g^*}$
$f_{temp1} \leftarrow MAX_{VALUE}$ ; $\boldsymbol{A} \leftarrow 0$; $\boldsymbol{a_g^*} \leftarrow 0$ ; $N_{\Gamma_s} \leftarrow N_p$
for $g \leftarrow \Gamma_s$ to $\Gamma_e$ do
    for $a_{g1temp} \leftarrow 0$ to $N_g$ do
        for $a_{g2temp} \leftarrow 0$ to $N_g - a_{g1temp}$ do
            for $a_{g3temp} \leftarrow 0$ to $N_g - a_{g1temp} - a_{g2temp}$ do
                $f_{temp} \leftarrow \mathrm{F}(a_{g1temp}, a_{g2temp}, a_{g3temp})$
                if $f_{temp} < f_{temp1}$ then
                    $f_{temp1} \leftarrow f_{temp}$, $a_{g1}^* \leftarrow a_{g1temp}$ $a_{g2}^* \leftarrow a_{g2temp}$, $a_{g3}^* \leftarrow a_{g3temp}$
                end
            end
        end
    end
    $N_{g+1} = N_g - a_{g1} - a_{g2} - a_{g3} + N_p$
    Store $\boldsymbol{a_g^*}$ into $\boldsymbol{A}$
end
return $\boldsymbol{A}$;
```

$$\Phi(\mathbf{a_g}) = (\sum_{l=0}^{L} a_{gl} - \sum_{l=0}^{L} s_{gl})/(\sum_{l=0}^{L} a_{gl}) \tag{4.8}$$

where L is the total number of layer of this stream. s_{gl} is the number of actually needed packets to playback the request layers. It is a constant value determined by the coding parameters.

4.4.4 Rate Allocation Optimization

The optimal rate allocation results in a priority region Γ can be found by the searching Algorithm 1. The information needed for searching is the available upload bandwidth N_p and the loss rate p_j of node N_j. Thus, the ORA algorithm is performed for each generation separately. This algorithm use the enumaration method to find the optimal point. The comlexity is $\Gamma_l * {N_g}^L$.

Table 4-A: Average bitrate and PSNR for each video layer

	bitrate [kByte/s]	PSNR [dB]
Base layer	41.6881	35.6259
Enhancement layer 1	78.7615	37.3439
Enhancement layer 2	113.7042	40.12

4.5 Experimental Results

4.5.1 Testing Media

The testing video streaming is Paris sequence encoded with H.264/SVC using Medium Grain Scalability. In our scenario, we have a single source node, streaming a video encoded with H.264/SVC using Medium Grain Scalability at CIF resolution. The source node is the only node that is not consuming the video. We make use of the Joint Scalable Video Model (JSVM) reference software. Three priority classes are selected, exploiting quality scalability, where the base layer has QP equal to 30, and the enhancement layer has QP equal to 22 and Medium Grain Scalability vector partitions equal to 6 and 10, for enhancement layer 1 and 2, respectively. We use the Paris sequence with CIF spatial resolution (352*288) and 30 frames per second. The average stream rate of all layers is defined as S. The compressed video streams account for an average bit rates and PSNR of:

We use an I-frame period of 32 frames (corresponding to 1.067 seconds of video) and a GOP size of 8 frames (equal to 0.2667 seconds of video). The video GOPs have PSNR and sizes shown in Fig. 4.1.

4.5.2 Experiment Settings

All experiments are based on the same P2P streaming network to achieve a fair comparison. We test our experiment in the network we proposed in Chapter. 3. All end nodes independently choose its neighbors and then form a randomly connected network. The

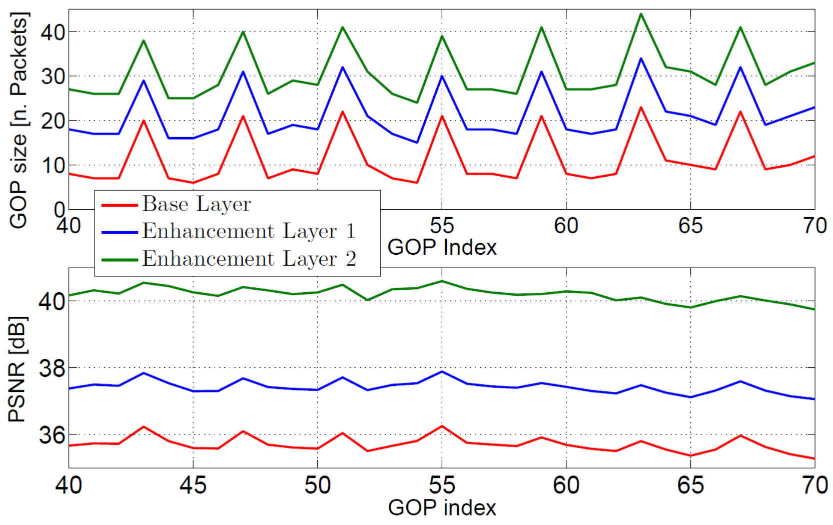


Figure 4.1: Paris CIF video, H.264/SVC with Medium Grain Scalability: Size of GOPs (top) and PSNR (bottom).

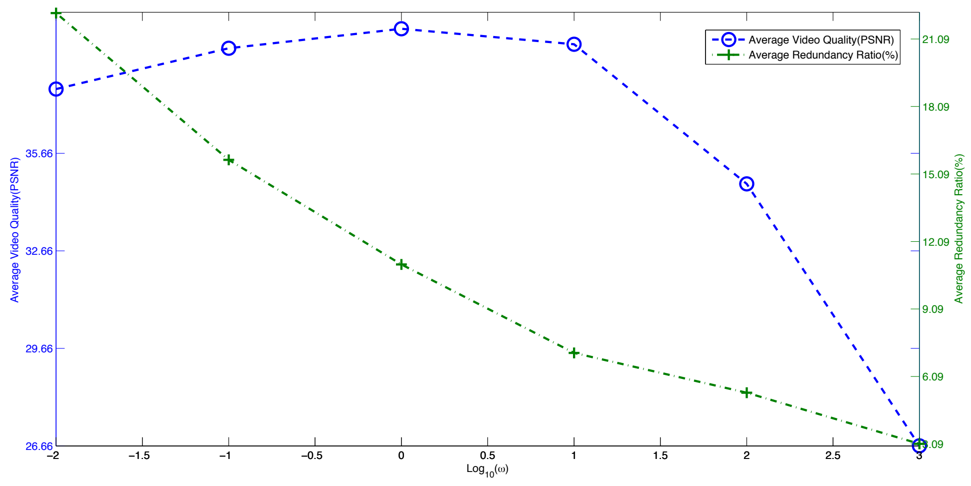
size of the network is set to be 100 nodes. Each node randomly selects 20 nodes as its neighbors. As for the download bandwidth of users, as commonly assumed in other P2P system studies, we simulate a P2P network where only the peer upload bandwidth is the bottleneck. The default upload bandwidth of the streaming server U_1 is set satisfy 15% end-users in the network. The loss rate of the link is set to be 0.02 to simulate the average network loss. In the first experiment, we evaluate the impact of the weight ω. In the first experiment, we evaluate the performance of the packet scheduling system by testing the system with different weight ω. This experiment proves that when ω is 1, the system can have the best performance. Therefore, in the remaining experiments we set ω to be 1. This enabled us to observe the impact of other parameters in the performance of the proposed approach.

Generation size κ is 8 frames, which correspond to a group of pictures covering 0.26 seconds of video. Therefore, κ is 0.26 in this experiment. The number of packets in each generation varies from 19 to 64 packets per generation based on the encoding result of the video, and the average number of packets in each generation is 32 packets in average. The size of priority period is 8κ. It means that the *playback deadline* at each peer is 2.13 seconds and video packets need to arrive at each node in 2.13 second to not expire. The maximum upload bandwidth of streaming source is set to satisfy 15% end users. The actual size of the video block is 1024 Bytes. Coefficients of network-encoded packets are stored in the packet header and transmitted with the video packets as well, whose size depends on the number of encoding packets in the generation. In NS2 simulation, the packet header (e.g. the address of destination peer, and the sequence number of video) is 150 Bytes, and the average size of network coding coefficients is about 32 Byte per packet (1 Byte for the coding coefficient of each video packet in the generation). If a video packet is less than 1024 Bytes, it is encoded directly without padding 0s.

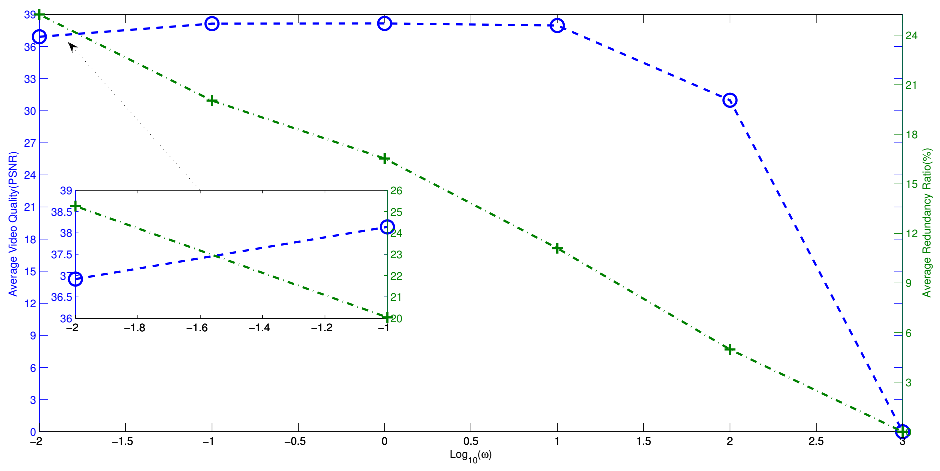
4.5.3 Performance Comparison

In this section, we implement our optimized rate allocation (ORA) and compare it with a hierarchical network coding algorithm (HNC) [110]. The buffer-map updating period is set to be 100ms. It means that the downloaders update their buffer-map to its sender nodes every 100ms. In [78], the buffer-map is embeded into each sent block. In average, each generation have 32 packets, and each generation corresponds to 0.26 seconds. Assuming that the upload bandwidth of each node is the same as the S, and the neighbor size is 20 nodes, the buffer-map updating period will be 160ms for each sender node. In this chapter, we choose 100ms to perform the comparison. In the next chapter, both 100ms and 200ms are are compared. In the experiment, the upload rate of each peer is set to be a fixed value, which is a ratio of the streaming rate S to evaluate the performance.

First the performance of the average video quality and the bandwidth efficiency versus different values of weight ω is demonstrated in Fig. 4.2. Fig. 4.2(a) represents the quality and efficiency performance when the loss rate is 0.02. Fig. 4.2(b) represents the quality and efficiency performance when the loss rate is 0.1. From Fig.4.2(a), we can observe that when the weight ω increases, the bandwidth redundancy decrease. At the same time, the perceived video quality increases first and then decreases. When the value ω is 1000, the average received PSNR is 26.66dB, and the redundancy ratio is 3.09%. When the ω increase, the ORA algorithm tends to transmit less redundancy in each generation and



(a) the loss rate = 0.02



(b) loss rate = 0.1

Figure 4.2: Average video quality and bandwidth redundancy ratio versus different values weight ω when the upload bandwidth is $1.2S$

saves the upload bandwidth to transmit the next generation. Therefore, the overall video quality increases. However, when the value of ω becomes larger, the ORA system puts more emphasis on the bandwidth redundancy ratio instead of the video quality. Thus, the ORA tends to transmit less redundancy in the network. In the lossy network, it may lead to unrecoverable transmission, thereby decreasing the overall video quality. In Fig.4.2(b), the bandwidth redundancy decreases, and the perceived video quality increases first and

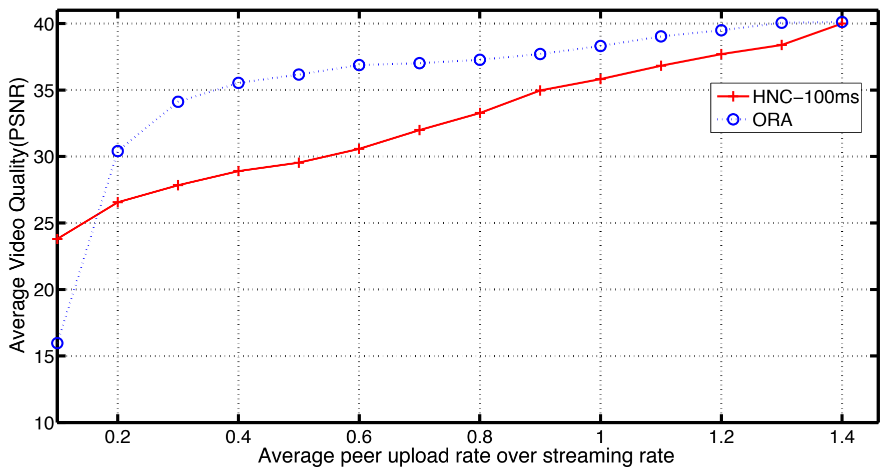


Figure 4.3: Performance comparison of the average delivery ratio between the ORA algorithm and the HNC algorithm

then decreases when the weight ω increases. The difference is that when the ω reaches to 1000, the perceived video quality reaches 0. This is because when the loss rate is 0.1, it becomes impossible that data can be recoverable when no redundancy is transmitted. We conclude that involving the weighted bandwidth redundancy ratio can increase the overall perceived video quality and decrease the redundant transmission. Furthermore, when the value ω is 1, the system can achieve a relatively high video quality and a relative low redundancy ratio.

Then we study the performance of average video quality of each method versus the upload bandwidth when the loss rate is 0.02, the ω is 1. The average video quality is measured by the peak signal-to-noise ratio (PSNR) at the receiver nodes. As depicted in Fig. 4.3, we compare the video quality of these two methods when the peer upload rate ranges from 0.1 to 1.4 times the full rate video. The results in Fig. 4.3 show that the proposed ORA scheme achieves a remarkable improvement in video quality over the whole range of bandwidth values. This improvement is because the HNC transmits packets based on the buffer-map updating mechanism. In our experiment, the time delay between the time that the receiver receives enough packets to the time that senders receive this stop message is 100ms. Therefore, after the receiver successfully decodes a

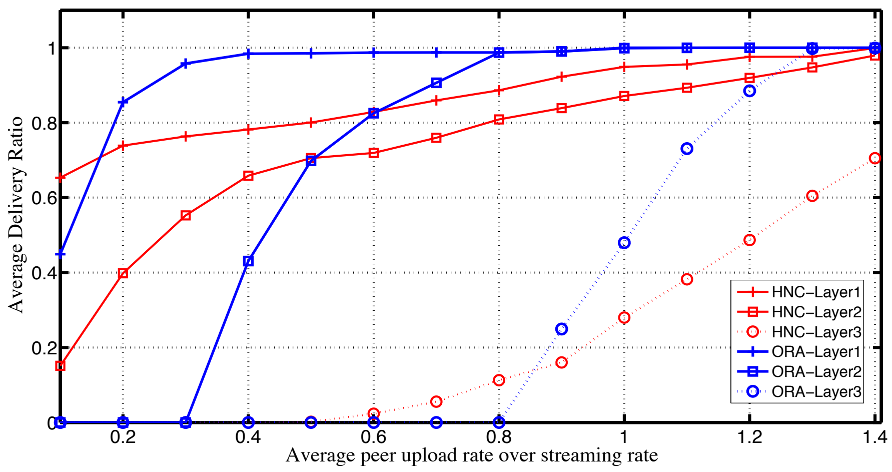


Figure 4.4: Performance comparison of the average delivery ratio between ORA and HNC

layer in a generation, senders may still push packets to the receiver due to the information updating delay. Different from the HNC, the ORA algorithm accurately calculates the number of packets that senders should send in the lossy network. Therefore, the overall received video quality can be improved.

Next in Fig.4.4, we study the performance of packet delivery ratio of each method when the peer upload rate ranges from 0.1 to 1.4 times the full rate video. The delivery ratio is defined as the average received informative packet rate over the streaming packet rate in each layer. Firstly, the ORA has better delivery ratio performance. It achieves full-delivery of the base layer when the peer upload bandwidth is 0.4S. In comparison, the HNC achieves full-delivery of the base layer when the peer upload bandwidth is 1.2 respectively. The experiment also shows that the delivery ratio of ORA is lower than the delivery ratio of HNC when the upload bandwidth is $0.1S$. This is because the ORA algorithm pushes packets based on the scheduling instead of random push. When the available upload bandwidth is too low, the ORA will not transmit any packets. Therefore, the HNC can decode some generations at some client nodes. The ORA performs better than the HNC algorithm when the upload bandwidth ranges from $0.2S$ to $1.2S$, it proves that the scheduling algorithm can bring better video quality compared with the random

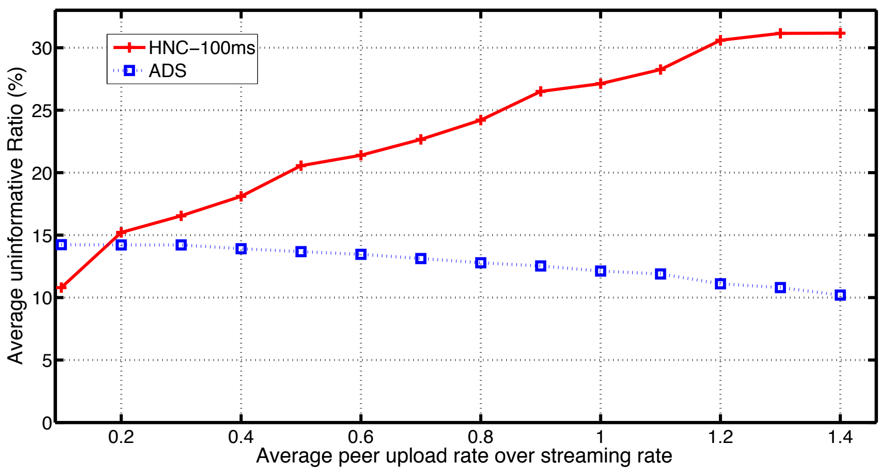


Figure 4.5: Performance comparison of the average uninformative packet ratio between the ORA and HNC scheme

push method.

We further study the influence of the upload bandwidth on the performance of average informative packet ratio in Fig.4.5. The loss rate is set to be 0.02, and the ω is 1. The uninformative ratio is defined as the *uninformative* video packet rate over all transmitted video packet rate. It is observed that when the upload bandwidth increases, the uninformative packet ratio of HNC increases, and the uninformative packet ratio of ORA slightly decreases. The uninformative packet ratio of the ORA algorithm slightly decreases when the upload bandwidth increases mainly because of the unequal importance of the layers. Usually, the ratio of the number of needed packets over the perceived video quality decreases for each layer. Therefore, when the ORA algorithm performs the calculation, it tends to transmit fewer redundancy packets for a tiny improvement in the video quality, which leads to the slightly decrease in the uninformative transmission ratio. The uninformative packet ratio of the HNC algorithm increases because there are more changes to transmit uninformative packets when more layers are decodable. In average, the ORA algorithm transmits less uninformative packets compared with the HNC algorithm. This is because the side effect of the late signaling message is very serious in scalable data transmission, while the ORA algorithm does not suffer from this

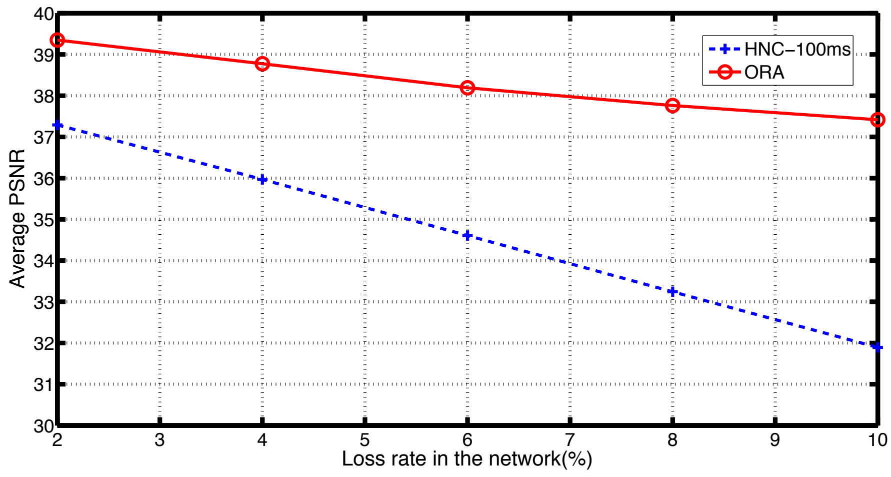


Figure 4.6: Performance comparison of the average video quality as a function of loss rate

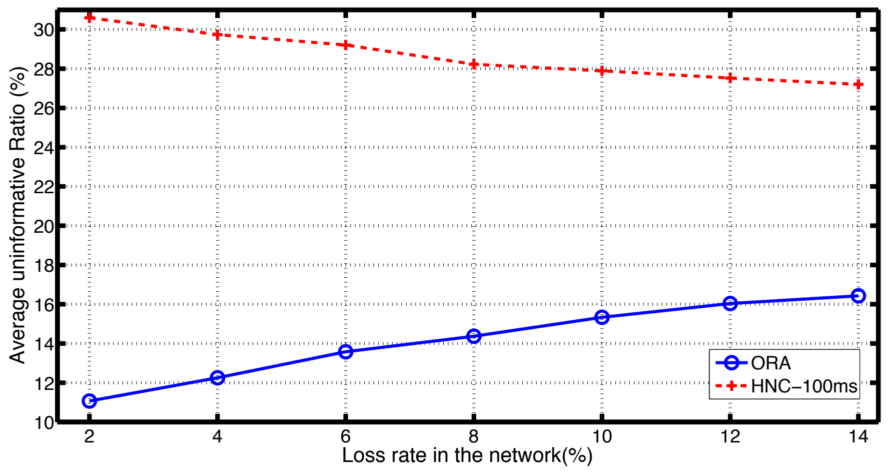


Figure 4.7: Performance comparison of the uninformative ratio as a function of loss rate

problem.

Additionally, the performance of the proposed ORA algorithm is compared with the HNC algorithm when the loss rate ranges from 0.02 to 0.1 in Fig.4.6. We observe that the video quality of both the ORA algorithm and the HNC algorithm decreases when the loss rate increases. The overall performance of the ORA algorithm is better than the HNC-100ms.

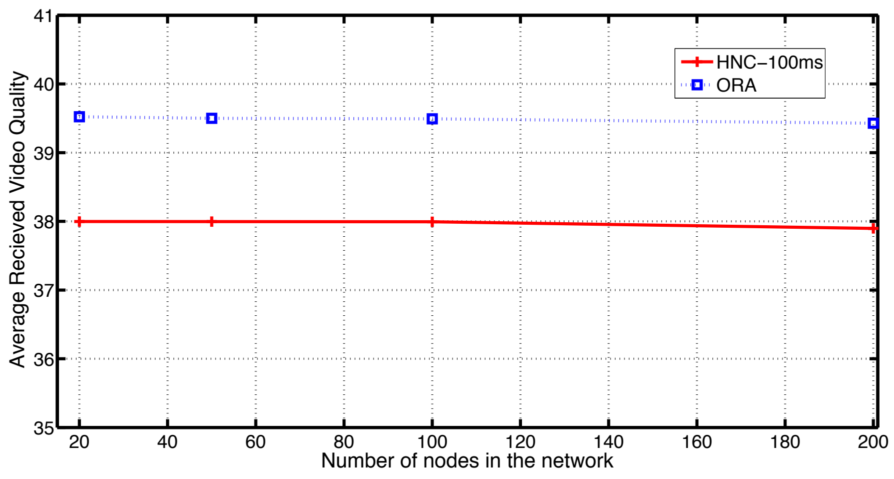


Figure 4.8: Performance comparison of the average delivery ratio as a function of network size

We also evaluate the performance of the uninformative ratio when the loss rate ranges from 0.02 to 0.14 in Fig.4.7. We observe that when the loss rate decreases, the average uninformative ratio of the HNC algorithm decreases, and the uninformative ratio of the ORA algorithm increases. It proves that when the loss rate increases, the ORA algorithm tends to transmit more redundant packets to provide protection to the data. When the packet loss rate increases, the HNC algorithm has less available upload bandwidth to transmit packets. Therefore, less scalable layers can be decoded, which in turn leads to less uninformative transmissions.

In the last experiment, we compare in Fig. 4.8 the average received video quality of the proposed algorithm versus the number of peers. As depicted in Fig. 4.8, we compare the video quality of the two methods when the network size N is 20, 50, 100, 200 separately. Average node upload bandwidth $\bar{U}$ is set to be $1.2S$. ω is set to be 1. The loss rate is set to be 0.02. From the figure, we can see that the perceived video quality of the ORA algorithm is almost the same at different networks. It proves that the ORA algorithm are scalable, and it is suitable for large-scale live streaming.

4.6 Summary

This chapter proposes an optimized rate allocation algorithm (ORA) to optimize the video quality and the weighted bandwidth redundancy in the NC-based lossy streaming network for scalable video transmission. It determines how senders' bandwidth is allocated among different video layers to balance the network loss. In scalable video streaming, different layer in the streaming should be given unequal protection. However, if too much protection are given to the lower layers, the transmission of the enhancement layers will be undecodable. Therefore, it is necessary to use the overall video quality as a criterion to schedule the rate allocation among layers for each generation. Furthermore, the scheduling also needs to consider the bandwidth efficiency. Inefficient transmission for a single generation at one client node can lead to the poor video quality of other client nodes and poor video quality in subsequent generations of this client node. Therefore, it is necessary to take both video quality and bandwidth redundancy into consideration to achieve a better overall video quality and system performance.

We formulate the rate allocation problem as a minimization problem of the weighted sum of distortion and bandwidth redundancy rate. The distortion is represented by the difference between the overall PSNR gain and the estimated PSNR gain. The estimated PSNR gain is the sum of the PSNR gain for each layer, and the estimated PSNR gain for each layer can be presented as a probability function based on the loss rate. The bandwidth redundancy ratio is represented as the ratio of the number of redundant packets over the number of actually transmitted packets. Given some bandwidth constraints to the rate allocation, the optimal rate allocation policy can be found through an exhausting search algorithm. After getting the rate allocation algorithm, the receiver node updates the rate allocation to each sender, and senders allocate their upload bandwidth according to allocation results.

Some experimental results are provided by comparing the ORA algorithm with an HNC algorithm. We prove that the proposed ORA algorithm has better performance than the HNC method when the buffer-map updating interval of the HNC is 100ms.

Chapter 5

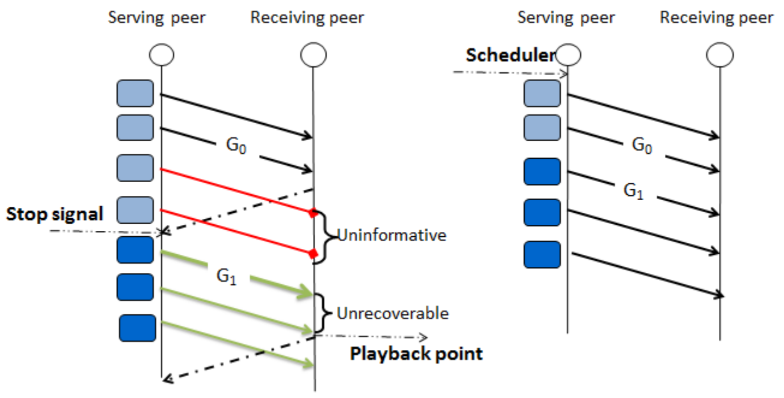
Distributed Packet scheduling with network coding

The mesh-based streaming system is widely used in some commercial P2P file distribution systems and some large-scale streaming systems [15], [87], [111], because it is more robust against peers' departure, and more efficient in using peers' upload bandwidth compared with the multicast tree. However, lack of synchronization among peers usually results in significantly redundant packet transmission, which in turn leads to severe bandwidth inefficiencies. In this chapter, we address the problem of finding a suitable asynchronous packet scheduling policy that greatly helps to overcome this critical redundant transmission problem. We propose a bandwidth cost minimization technique under a full video packet recovery constraint. In order to add a scalability and improved performance, we also further derive a distributed packet scheduling algorithm. Both implementation and analytical considerations of the proposed approaches are described in this chapter. Experimental results confirm that the proposed algorithms deliver higher bandwidth efficiency with reduced redundancy and communication overhead rate, and consequently, better quality-of-service in terms of improved video quality and delivery ratio. This chapter focuses on the packet scheduling for the non-scalable video transmission. Based on the proposed algorithms, packet scheduling algorithms for scalable video are proposed in Chapter. 6.

5.1 Motivation

Despite substantial progress in the field, little attention has been given to the bandwidth efficiency in the *mesh-push* schemes. Improving bandwidth efficiency is critical for video streaming because efficient bandwidth usage means that more useful packets arrive at receivers. This, in turn, results in a better quality of service in any multimedia streaming. A problem with current *mesh-push* schemes is that some transmitted packets are redundant in improving received media quality. Let us consider a transmission process in conventional *mesh-push* P2P networks. Senders actively push encoded packets until the arrival of a stop message from the receiver. The choice of packets may be unintelligent in some cases, due to the delayed stop message or actual network conditions. For instance, some transmitted packets are received after the current content generation has already been decoded. We call such packets **uninformative**. Furthermore in some cases, available packets are insufficient to decode the original content within the given time window. Let us term these packets **unrecoverable**. These two types of packets lead to bandwidth inefficiencies. In conventional push-based schemes, it is impossible to avoid the transmission of *uninformative* and *unrecoverable* packets because fully intelligent scheduling requires enormous amounts of overheads for communication which cannot be compensated by the gains achieved. Thus, it is critical to find a trade-off between the improved transmission efficiency and the information overheads needed to achieve it.

The main difference between the traditional push mechanisms and our scheduled push mechanism is outlined in Fig.5.1. This illustration shows an extremely simplified transmission process in which a sender node sends encoded media packets to a receiver node. Fig. 5.1 (a) shows a random push with random network coding algorithm (RND) [78], which is used to represent the existing push mechanisms. Fig. 5.1 (b) is the



a)Random push with random network coding b)Scheduled push with random network coding

Figure 5.1: Comparison of the transmission mechanism between the RND scheme and the proposed scheme

proposed packet scheduler. The transmissions of these two algorithms are compared to understand their difference. In the illustration, G_0 and G_1 are different generations of a video stream. To decode the original content, the receiver needs to receive two packets of G_0 and three packets of G_1 within the given time window. In the RND algorithm shown in Fig. 5.1 (a), the sender keeps pushing network-encoded packets of G_0 until the arrival of a stop message from the receiver. However, due to the message updating interval, the stop message from the receiver is delayed. Two *uninformative* packets are sent to the receiver. The sender then starts transmitting network-encoded packets of G_1. At this time, the sender does not have enough upload bandwidth to transmit 3 packets of G_1 within the given time window. For this reason, the received packets are not enough to be decoded. Therefore, the received packets are discarded, wasting network resources. In contrast, in our packet scheduling algorithm shown in Fig. 5.1 (b), the sender transmits packets according to the results of pre-scheduling. Firstly, the scheduling compensation model (SCM) calculates the number of actually needed packets for each generation, and the adaptive push algorithm (APA) accurately finds the generations that need to be skipped. Next, the centralized packet scheduler (CPS) and the distributed packet scheduler (DPS) construct linear programming functions to get scheduling results before the actual media transmission. The scheduling results specify how multiple senders cooperatively transmit packets to receivers. Finally, each sender follows the scheduling results to deliver packets. These *uninformative* transmissions caused by the asynchronous communication can be minimized in this way. Furthermore, if some transmitted packets are lost or some senders churn during transmission, the transmission system can achieve fast recovery by pulling packets from other neighboring senders since all transmitted packets are network-encoded.

The key to the transmission model is the accuracy of the underpinning pre-calculation. To achieve an accurate pre-calculation of generation skips and scheduling compensations, the actual network conditions such as loss rate, peer churn rate, inherent *uninformative* packet rate, and the actual upload bandwidth are taken into consideration. Firstly, we propose a scheduling compensation model (SCM), an adaptive push algorithm (APA). The SCM and APA calculate the number of packets that each receiver could receive from its neighboring nodes after taking the dynamic network conditions into account. The two algorithms work together to reduce *unrecoverable* transmissions. Subsequently, to achieve an accurately cooperative packet scheduling among multiple senders, two approaches are proposed: a centralized scheduling algorithm (CPS) and a distributed scheduling algorithm (DPS). In general, both scheduling algorithms achieve accurate calculations by estimating a decoding status and allocating the suitable upload bandwidth to receivers. The difference between them is the amount of required information. CPS calculates the packet scheduling strategy based on the global information while DPS calculates the scheduling strategy based on the local information of neighboring peers. Furthermore, the CPS algorithm formulates the allocation problem as a global integer linear programming (ILP) problem. In contrast, the DPS transforms the global ILP problem into a local ILP problem. It in turn adds a critical scalability property to the packet scheduling model. The experimental results confirm that both algorithms generate less communication traffic, *uninformative* and *unrecoverable* packets compared to conventional random-push schemes.

5.2 Network Model and Media Model

The overall system includes a network model, a network-encoded media streaming model, and a packet scheduling model. This section gives the definitions of the network model and the media streaming model.

5.2.1 Network Model

The overall network model is depicted in Fig.3.1 as described in Chapter 3. Three types of peers are defined in this model: the tracker node, the streaming server, and the client node. In the system, the tracker node does not transmit any coded video packet, but only serves the purpose of enabling peers to find each other and exchange control messages. Each peer contacts the tracker node to join the network. Some control information is exchanged during the procedure, i.e. the upload bandwidth of neighbor nodes and size of video packets. The peer churn rate in the network consists of both the peer arrival and the departure rate. To simplify the model, the peer churn rates ι is set to be constant. When a sender node leaves the network, another available sender is designated to receivers by the tracker node. The streaming server transmits network-encoded video packets to client nodes. The overlay of network nodes is represented as a graph ($\mathcal{N}$, ε) composed for nodes $\mathcal{N} = \{\mathcal{N}_0, ..., \mathcal{N}_N\}$ and the edge ε. In these nodes, $\mathcal{N}_0$ represents the tracker node and $\mathcal{N}_1$ represents the streaming server. The rest nodes from $\mathcal{N}_2$ to $\mathcal{N}_N$ are client nodes. For any nodes in the actual video streaming network, $U = \{U_1, ...U_N\}$ is indicated as the vector of upload bandwidth of streaming nodes. $\bar{U} = \sum_{i=1}^{N} U_i/N$ is defined as the average upload bandwidth of the streaming network. To simplify the discussion, we also define the κ-quantized upload bandwidth as $\tilde{U}_i = U_i * \kappa/V$, where κ is defined as the playback duration of a generation, V is the video packet size. The unit of $\tilde{U}_i$ is the number of packets /κseconds. The average value of the κ-quantized upload bandwidth is also defined for simplicity as $\hat{U} = \sum_{i=1}^{N} \tilde{U}_i/N$. To take fully advantage of NC, the optimization problem would be able to find the suitable peers when each peer have α linear independent segments. Therefore, we define that a client node becomes a sender node when it holds α linear independent segments $(0 \leq \alpha \leq 1)$. It can ensure the content availability at the senders. For any node $\mathcal{N}_j$, $A_j \subset \mathcal{N}$ is defined as the neighborhood of $\mathcal{N}_j$. It is the set of sender nodes that are connected to $\mathcal{N}_j$. The child node $\mathcal{N}_j$ reports the network change to the tracker node when it experiences bandwidth variations, such as the departure of a parent, or the bandwidth change.

5.2.2 Network-encoded Media Streaming

The media stream that distributed to the network nodes is modeled as a single dimensional array of generations. A generation is usually made of one or several group of pictures (GOP) in the media. Each generation is identified within the media stream by a temporal index $g \in [1, G]$. Each generation g is subdivided into P_g blocks of symbols, and the size of each video packet is V bytes. Generations with identical temporal index g have the same playback duration κ. The average streaming rate is termed as S. Instead of transmitting raw video packets, nodes transmit linear combinations of its received packets to other nodes. A new outgoing packet is generated by performing random network coding in a single generation [13].

A new transmission region is proposed to achieve a scheduled and guaranteed transmission. The transmission region can be divided into an *urgent region* and a *priority region* as shown in Fig.5.2. The *priority region* is the scheduled transmission region. In the *priority region*, the transmissions of generations are scheduled firstly according to the CPS and the DPS. Senders then follow the scheduling results to transmit packets. In the *urgent region*, *unrecoverable* generations are requested by receiver nodes from its all neighboring senders $\forall \mathcal{N}_i \in A_j$ according to a request model. A *priority region* Γ is defined as a moving time window next to the *urgent region.* The size of the priority region is Γ_l. The start point and the end point of the *priority region* are Γ_s and Γ_e respectively. An *urgent region* is defined as a moving time window next to the playback point. The start point and the end point of the *urgent region* are ω_s and ω_e. The urgent

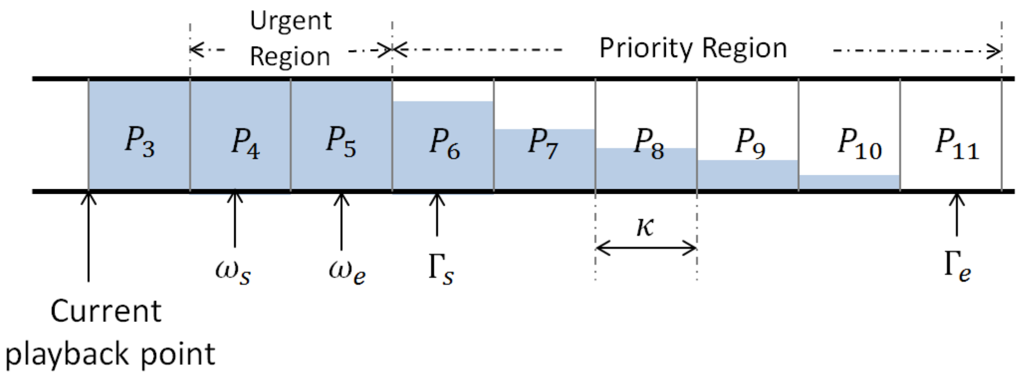


Figure 5.2: Sample buffer status of a client node $\mathcal{N}_j$. The current playback point is g=5, the priority region $\Gamma = [\Gamma_6, \Gamma_7, ..., \Gamma_{11}]$.

region and the priority region move as the playback point moves. The example in Fig.5.2 illustrates that node $\mathcal{N}_j$ is playing the video in generation 3. Its urgent region is from generation 4 to 5 ($\omega_s = 4$ and $\omega_e = 5$). Its priority region is from generation 6 to 11 ($\Gamma_s = 6$, $\Gamma_l = 6$, and $\Gamma_e = 11$). To successfully decode the original media, the client node needs to receive $[P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}]$ informative packets in $g \in (4, 11)$ respectively. For the sake of more clarity, all notations used in the model are summarized in Table 6-A.

5.2.3 Scheduling Compensation Model (SCM)

To reduce the *unrecoverable* generations caused by the unsuccessful packet delivery, we develop a scheduling compensation model. Its main function is to calculate the number of packets $\tilde{P}_g$ that need to be sent from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ such that P_g independent packets can successfully arrive at $\mathcal{N}_j$. In a dynamic network, the number of sent packets $\tilde{P}_g$ needs to be larger than P_g so that $\mathcal{N}_j$ can successfully receive P_g packets to decode the original generation. $\tilde{P}_g - P_g$ packets are used to compensate the unsuccessful packet delivery caused by the dependent transmission, packet loss, and peer churn.

To find the suitable $\tilde{P}_g$, we use the loss rate θ, peer churn rate ι and the inherent independent probability ρ to estimate the successful transmission rate. We also define the

Table 5-A: NOTATION USED IN THE SYSTEM MODEL

$\mathcal{N}_i$	Network node i, $i \in [0, N]$
A_j	Neighborhood of the receiver node $\mathcal{N}_j$
g	Temporal index (generation) in the media, $g \in [1, G]$
κ	Playback duration of each generation
V	Size of video packets
U_i	Upload bandwidth of streaming node $\mathcal{N}_i$, $i \in [1, N]$ [kByte/s]
$\tilde{U}_i$	κ-quantized upload bandwidth of $\mathcal{N}_i, i \in [1, N]$ [packets/κsec]
$\tilde{U}_{ij}$	κ-quantized upload bandwidth allocated from $\mathcal{N}_i$ to $\mathcal{N}_j$ [packets/κsec]
$\bar{U}$	Average upload bandwidth of the streaming network [Kbyte/s]
$\hat{U}$	Average κ-quantized upload bandwidth [packets/κsec]
S	Average streaming rate
α	Independent threshold for a receiver node to become a sender node
Γ	Priority region, $\Gamma = [\Gamma_s, ..., \Gamma_e]$
ω	Urgent region, $\omega = [\omega_s, ...\omega_e]$
μ	Independent transmission rate
θ	Loss rate
ι	Peer churn rate
ρ	Inherent linear independent probability
P_g	Number of real video packets in each generation g
$\tilde{P}_g$	Expected number of packets that $\mathcal{N}_i \in A_j$ must send to $\mathcal{N}_j$ to recover g
$\hat{P}_g$	Number of actually scheduled packets in generation g
$\bar{P}$	Average value of $\hat{P}_g$
H_{ijg}	Number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g
S_{ijg}	Number of sent packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g

successful transmission rate μ ($0 \leq \mu \leq 1$) as the probability that a linear independent packet is successfully received by a node. Therefore, accounting for the unsuccessful transmission rate, the expected number of scheduled packets that $\mathcal{N}_j$ must receive from $\mathcal{N}_i \in A_j$ to recover the generation g can be written as:

$$\tilde{P}_g = P_g/\mu \tag{5.1}$$

The successful transmission rate μ is related to the loss rate θ, the peer churn rate ι and inherent linear dependent rate ρ of random network coding. The inherent linear dependent rate ρ is defined as the dependent transmission probability caused by the inherent property of random network coding. Randomly chosen coefficients may be the same as the coefficients in previously sent packets. According to [13], the lower bound of the inherent linear independent probability is $\rho \geq (1 - 2^{-q})$. The lower bound of the successful transmission rate can therefore be simply given as:

$$\begin{aligned} \mu &= (1-\theta)(1-\iota)\rho \\ &\geq (1-\theta)(1-\iota)(1-2^{-q}) \end{aligned} \tag{5.2}$$

According to Eq.5.1 and Eq.5.2, $\mathcal{N}_i \in A_j$ must send to $\mathcal{N}_j$ at least $\tilde{P}_g$ packets to compensate the unsuccessful transmissions so that receivers can successfully decode P_g packets in generation g.

5.2.4 Adaptive Push Algorithm (APA)

The APA is proposed to reduce *unrecoverable* transmissions caused by the insufficient upload bandwidth. Reducing *unrecoverable* transmissions can lead to a better use of resources, thereby resulting in better bandwidth efficiencies. The APA reduces *unrecoverable* transmissions by determining the number of actually scheduled packets $\hat{P}_g$ from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ considering the actually available upload bandwidth. These *unrecoverable* packets are generated because the receiver $\mathcal{N}_j$ fails to receive P_g informative packets from senders $\mathcal{N}_i \in A_j$ before the playback deadline, due to limited upload bandwidth restricting the packets sent from the neighboring nodes. To avoid *unrecoverable* transmissions, the APA assesses if the available κ-quantized upload bandwidth $\hat{U}_g^{Cum}$ can afford the expected transmission $\tilde{P}_g$. When $\hat{U}_g^{Cum}$ can afford the expected transmission $\tilde{P}_g$, generation g is transmitted ($\hat{P}_g = \tilde{P}_g$). Otherwise, the generation is skipped ($\hat{P}_g = 0$). When a generation g is skipped, the average upload bandwidth of g can be used to deliver generations from $g+1$ to Γ_e. The APA can be described by the following pseudo-code in the Algorithm 2:

Algorithm 2: Adaptive Push Algorithm

```
Input: $\tilde{\boldsymbol{P}}$,$\hat{U}$
Output: $\hat{\boldsymbol{P}}$
for $g$ from $\Gamma_s$ to $\Gamma_e$ do
    $\hat{U}_g^{Cum} = \hat{U}_{g-1}^{Cum} + \hat{U}$
    if $\hat{U}_g^{Cum} >= \tilde{P}_g$ then
        $\hat{P}_g = \tilde{P}_g$
    else
        $\hat{P}_g = 0$
    end
    $\hat{U}_g^{Cum} = \hat{U}_g^{Cum} - \hat{P}_g$
    Store element $\hat{P}_g$ into vector $\hat{\boldsymbol{P}}$
end
return $\hat{\boldsymbol{P}}$;
```

The APA is performed by the *track server* before the transmission. For every generation, the tracker node compares the expected number of packets $\tilde{P}_g$ with the available κ-quantized upload bandwidth $\hat{U}_g^{Cum}$. When $\hat{U}_g^{Cum} \geq \tilde{P}_g$, $\hat{P}_g = \tilde{P}_g$, otherwise, $\hat{P}_g = 0$. In this way only recoverable generations are transmitted, and the bandwidth is better utilized. $\hat{U}_g$ is the average κ-quantized upload bandwidth of the streaming network in generation g. It is equal to the $\hat{U}$ in our system. The unit of $\hat{U}_g^{Cum}$, $\hat{P}_g$ and $\hat{P}_g$ are packets/κseconds. To inform all client nodes of the number of actually scheduled packets $\hat{P}_g$, the tracker node needs to send a message to each client node $\mathcal{N}_i$ every generation. The size of a communication message is 2 bytes. The communication overhead of APA is $2N$ bytes/κseconds, which is $2N/\kappa$ bytes/s. $\bar{P}$ is the average value of $\hat{P}_g$. If the streaming rate is considered as a relatively stable rate, P_g packets can successfully arrive at each receiver when $\hat{U} \geq \bar{P}$.

5.3 Optimized Packet Scheduler

This section proposes two packet schedulers to determine how multiple senders cooperatively contribute their upload bandwidth to different receivers. The schedulers can reduce the *uninformative* transmission caused by asynchronous communications, thereby leading to better transmission efficiencies. The process of the packet scheduling can be summarized as follows: the packet schedulers calculate the number of packets that each sender needs to send to its receiver, and then each sender follows the scheduling results to transmit packets individually. The objective of a streaming network is that senders cooperatively deliver all recoverable generations to all receivers and avoid *uninformative* transmissions at the same time. This requires each sender to accurately determine the number of packets to send to each receiver at each generation. For this purpose, two packet schedulers are proposed. Firstly, a centralized packet scheduler (CPS) is proposed to improve the global bandwidth efficiency. The CPS formulates the global packet scheduling problem as a global integer linear programming problem. A distributed packet scheduler is then derived from the CPS by finding solutions to an approximative optimization objective. Both CPS and DPS accurately schedule packet transmissions, and senders can cooperatively contribute their upload bandwidth to receivers to achieve recoverable and informative transmissions.

5.3.1 Centralized Packet Scheduler (CPS)

The CPS is a global packet scheduler, designed to find the global multi-sender cooperation model to organize the upload bandwidth of all senders to their neighboring receivers. Such a multi-sender cooperation problem is modeled as a redundant transmission minimization function among all senders under a constraint of full recovery. *Uninformative* transmission is caused by the superfluous transmission after the corresponding generation is already successfully transmitted. The streaming network can successfully transmit all recoverable generations with the CPS, and simultaneously minimize *uninformative* transmissions. Any receiver node $\mathcal{N}_j$ needs to receive $\hat{P}_g$ packets to recover generation g, and any other packets are *uninformative* for the client nodes. The objective function of the CPS is to therefore minimize the number of transmitted packets, and the constraints of the CPS are to ensure the full transmission of $\hat{P}_g$.

The number of packets that should be sent from each sender $\mathcal{N}_i$ to each receiver node $\mathcal{N}_j$ in generation $g \in \Gamma$ is denoted as a non-negative integer H_{ijg}. A positive constant C is defined as the transmission cost (the consumed bandwidth of a single packet). For each generation $g \in \Gamma$, the scheduling problem is formulated as a cost minimization problem with some given restrictions in Eq.5.3, and the minimization function is solved as an Integer Linear Programming (ILP) problem by the Simplex method.

$$\begin{aligned}
\boldsymbol{H} &= \arg\min_{H_{ijg}} C \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} H_{ijg} \\
\text{subject to } & \sum_{g=\Gamma_s}^{k} \sum_{j=2}^{N} H_{ijg} \leq \sum_{g=\Gamma_s}^{k} \tilde{U}_{ig} \quad \forall i, \Gamma_s \leq k \leq \Gamma_e \text{ (a)} \\
& \sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \;\; \forall j, g \;\; \text{(b)} \\
& H_{ijg} \leq \alpha \hat{P}_g \;\; \forall i \in (2, N),\; j,\; g \;\; \text{(c)} \\
& H_{ijg} = 0 \quad \forall \mathcal{N}_i \notin A_j,\; j,\; g \qquad \text{(d)}
\end{aligned} \tag{5.3}$$

The optimization objective is to find the minimum bandwidth cost to achieve the full delivery of $\hat{P}_g$. The constraint (a) means that the number of sent packets of each sender node $\mathcal{N}_i$ should be smaller than its available κ quantized upload bandwidth $\tilde{U}_{ig}$ in the priority period, where $\tilde{U}_{ig}$ is equal to $\tilde{U}_i$. The constraint (b) means that senders need to cooperatively send at least $\hat{P}_g$ packets so that P_g packets can arrive at the receiver node. The constraint (c) means that the number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ needs to be smaller than the number of its *linear independent* packets $\alpha\hat{P}_g$. This constraint guarantees that enough contents are available at each client node. The constraint (d) guarantees that only nodes $\mathcal{N}_i \in A_j$ can send packets to its neighboring nodes $\mathcal{N}_j$. The optimization function achieves its optimized solution at $\sum_{i=1}^{N} H_{ijg} = \hat{P}_g$.

This algorithm hypothesizes that a central coordinator who knows all upload bandwidth $\tilde{U}_i$ of each node $\mathcal{N}_i$ exists in the network. It could be the tracker node in our system because it holds all information of client peers in the network. In the global scheduling algorithm, the streaming server is scheduled like other sender peers so that a precise multi-sender cooperation model can be built for each receiver. After the CPS calculates the number of packets that should be sent from each sender $\mathcal{N}_i$ to each receiver $\mathcal{N}_j$, the tracker node sends a control message to each sender $\mathcal{N}_i$. The scheduling results are then stored in the local record of each sender node. The number of sent packets S_{ijg} and H_{ijg} are compared when every transmission opportunity arises at the sender node $\mathcal{N}_i$. If S_{ijg} is smaller than H_{ijg}, a network encoded packet in generation g is sent to node $\mathcal{N}_j$ and the value of S_{ijg} increases. Generations close to Γ_s are those that are initially pushed.

5.3.2 Distributed Packet Scheduler (DPS)

The DPS is proposed to increase the scalability of the system. The DPS is a distributed solution to the centralized optimization function. It solves how multiple senders contribute their bandwidth to each receiver based only on the local bandwidth information. As mentioned in section 5.3.1, the CPS algorithm globally reduces *uninformative* transmission, thereby improving bandwidth efficiencies. However, in a large-scale P2P network, the complexity of the global optimization would be extremely high. Therefore, a distributed bandwidth-efficient packet scheduling algorithm is proposed in this section. In the DPS, we find the symbolic solution to the packet scheduling problem by providing some constraints to an approximate optimization objective of the CPS. The main steps of this algorithm include 1) Each sender allocates its upload bandwidth to each receiver by the handshake procedure; 2) Each sender uses a local weighted quadratic equation to find the most suitable allocation for each receiver in each generation.

5.3.2.1 Handshake Procedure

In the handshake procedure, each sender node $\mathcal{N}_i$ allocates its overall κ-quantized upload bandwidth $\tilde{U}_i$ to its receiver node $\mathcal{N}_j$, denoted as $\tilde{U}_{ij}$. The unit of the allocated upload bandwidth $\tilde{U}_{ij}$ is packets/κseconds. When the receiver node $\mathcal{N}_j$ joins the network, it initially contacts the tracker node to obtain a list of free sender peers $\mathcal{N}_i$ to form its neighboring nodes A_j. When the tracker picks the set of sender nodes, it ensures that $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_i \geq \hat{U}$. The tracker node then calculates the amount of upload bandwidth $\tilde{U}_{ij}$ that each sender $\mathcal{N}_i$ should contribute to support the delivery to this receiver node $\mathcal{N}_j$ according to the Eq.5.4.

$$\tilde{U}_{ij} = \min(\frac{\tilde{U}_i}{\sum_{\mathcal{N}_i \in A_j} \tilde{U}_i} \hat{U}, \alpha \bar{P}) \tag{5.4}$$

If $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_{ij} < \hat{U}$, more senders are allocated to this receiver and the Eq.5.4 is recomputed until $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_{ij} \geq \hat{U}$. Note $\tilde{U}_{ij} = 0$ for any $\mathcal{N}_i \notin A_j$. The scheduled upload bandwidth $\tilde{U}_{ij}$s for any $\mathcal{N}_i \in A_j$ are stored into a vector following its identification order and sent to each sender node $\mathcal{N}_i \in A_j$. At the same time, the tracker node calculates the new available upload bandwidth for each sender $\mathcal{N}_i \in A_j$ by $\tilde{U}_i' = \tilde{U}_i - \tilde{U}_{ij}$. At the end of the handshake procedure, each sender $\mathcal{N}_i$ will get a list of cooperative senders $\mathcal{N}_i \in A_j$, and their corresponding allocated bandwidth $\tilde{U}_{ij}$ for this receiver $\mathcal{N}_j$.

5.3.2.2 Real-time Distributed Scheduler

The real-time distributed scheduler pre-calculates the number of scheduled packets H_{ijg} from node $\mathcal{N}_i$ to $\mathcal{N}_j$ in any generation g. The cooperative transmission model in the local scheduling algorithm can be summarized as a multi-sender and single receiver relationship. A receiver node $\mathcal{N}_j$ has a set of sender nodes $\mathcal{N}_i$, where $\mathcal{N}_i \in A_j$. The optimization problem is formed as a bandwidth cost optimization function under the full recovery constraint in Eq. 5.5. This optimization function calculates the number of packets H_{ijg}

that should be sent from $\mathcal{N}_i$ to $\mathcal{N}_j$ so that $\hat{P}_g$ packets can be successfully decoded at the receiver node.

$$\begin{aligned} \boldsymbol{H} &= \arg\min_{H_{ijg}} \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}} \\ \text{subject to} & \sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j, g \end{aligned} \tag{5.5}$$

The above optimization function finds the suitable allocation H_{ijg} subject to a constraint that the number of scheduled packets must be larger than $\hat{P}_g$. This constraint ensures that this receiver node receives enough packets to recover the original content in generation g. As the transmission allocation to each node $\mathcal{N}_j$ in each generation is independent, Eq.5.5 can be transformed to Eq.5.6.

$$\begin{aligned} \boldsymbol{H} &= \arg\min_{H_{ijg}} \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}} \\ \text{subject to} & \sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j, g \end{aligned} \tag{5.6}$$

The above optimization problem can be solved by finding the solution $\boldsymbol{H}$ that minimizes the expected Lagrangian in Eq. 5.7 since $\sum_{i=0}^{N} H_{ijg}$ approaches to $\hat{P}_g$.

$$J(\boldsymbol{D}) = \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}} + \lambda(\hat{P}_g - \sum_{i=1}^{N} H_{ijg}) \forall j, g \tag{5.7}$$

By solving the above Lagrangian function, the symbolic solution H_{ijg} from sender $\mathcal{N}_i$

to any receiver node for any generation g can be expressed as followed in Eq. 5.8:

$$\begin{aligned} H_{ijg} &= round(\frac{\tilde{U}_{ij}}{\sum\limits_{k=i}^{I} \tilde{U}_{kj}} P_r) \quad (a) \\ P_r &= \hat{P}_g - \sum_{k=1}^{i-1} H_{kjg} \qquad (b) \end{aligned} \tag{5.8}$$

In Eq. 5.8 (a), H_{ijg} is calculated by the product of the bandwidth ratio and the remaining number of packets P_r. The ratio can be expressed as the ith allocated upload bandwidth over all allocated upload bandwidth from sender node $\tilde{U}_{ij}$ to the last sender node $\tilde{U}_{Ij}$. Eq.5.8 (b) means that the remaining number of packets P_r can be represented by the difference between the scheduled result $\hat{P}_g$ and the number of scheduled results for sender node $\tilde{U}_0$ to $\tilde{U}_{i-1}$.

The optimization objective in the DPS is the approximation of the optimization objective in the CPS algorithm. According to the optimization function, it is clear to see that the proposed optimization objective in DPS is the square of the H_{ijg}. Moreover, the solution to Eq.5.5 also satisfies all constraints in Eq.5.3. The details of proof can be found in Appendix A.1.

In general, the distributed packet scheduling algorithm uses the handshake procedure to do the bandwidth pre-allocation for each sender. The multilevel linear programming can therefore be written as a single layer linear programming. By solving the single layer linear programming, the symbolic solution to the ILP problem can be found, thereby reducing the computational complexity. Furthermore, the solution to the quadratic equation is weighted, which means that sender nodes with small $\tilde{U}_{ij}$ have less variation. This helps to reduce the scheduling error. After each sender gets the scheduling results H_{ijg}, it calculates the number of pending transmitting packets by subtracting S_{ijg} from H_{ijg}. If H_{ijg} is larger than S_{ijg}, an encoded packet is sent to the receiver. Generations closed to Γ_s are pushed initially.

In all, the distributed scheduling algorithm gives a symbolic solution to the local optimization function. More, the solution of the distributed algorithm also satisfies the constraints of CPS algorithm in the Eq. 5.3. Both CPS and DPS reduce the redundant transmission by global or local packet scheduling. The network resources can be better utilized, and the bandwidth efficiencies can be improved.

5.3.3 Request Model

A request model is proposed to deal with the *unrecoverable* transmission caused by unpredictable network variations. Although the scheduling compensation model considers the peer churn and loss rate in the network, these may vary in a real network. When the receiver node has an unrecovered generation in its urgent region, it will periodically broadcast its buffer map as a request signal to all neighboring nodes. One or more request signals from different receivers may arrive at the sender nodes. Then, each sender with spare upload bandwidth capacity calls the "random push algorithm" to push packets to the receivers. When a receiver successfully decodes the corresponding generation, it immediately sends its buffer-map as a stop signal to all neighboring nodes. This mechanism aims at improving the recoverability of the streaming in the system. To keep the efficiencies of the system, a local information updating procedure is called, and the packet scheduling algorithm is recomputed for this node if more than 10% of packets in the *urgent region* are requested from senders or more than 10% of packets in the *priority region* are *uninformative.*

5.4 Performance Evaluation

In this section, we report experimental results with our packet schedulers using streaming a network-encoded media sequence over a mesh-push based P2P network. To evaluate the performance of the proposed scheduling algorithms, we compare our proposed packet schedulers with three existing scheduling algorithms with four different sets of considerations. Firstly, the quality-of-service (QoS) at the client nodes is measured for video quality and delivery ratio, with different upload bandwidths. Secondly, the bandwidth efficiencies in the network are measured regarding *uninformative ratio*, communication overhead, and informative packet rate. Thirdly, the performance in a lossy network is evaluated, and finally, the system scalability is evaluated. We present the simulation environment and metrics in Section 5.4.1 and the streaming performance comparison in Section 5.4.2.

5.4.1 Experimental Settings and Metrics

We evaluate our packet schedulers over a mesh-push based P2P network. The P2P network is implemented on an event-based network simulator NS2. All end nodes independently choose its neighbors and then form a randomly connected network. The size of the default test network is set to be 100 nodes. The neighborhood size A_j is set to be 20 nodes. The default upload bandwidth of the streaming server U_1 is set satisfy 15% end-users in the network. The setting of the server upload bandwidth is similar to the settings in [78] to achieve a fair comparison. In the presented three first experiments, the upload bandwidth of each peer is set to be a constant value ranging from $0.8S$ to $1.45S$. Here we aim at evaluating the QoS and bandwidth efficiency of the system. S is the average video bitstream rate. These initial experiments proved that when the peer upload bandwidth is $1.2S$, the delivery ratio can reach 99.9%. Therefore, in the remaining experiments we set the peer upload bandwidth to be $1.2S$. This enabled us to observe the impact of other parameters in the performance of the proposed approach. The default loss rate and the peer churn rate are set to be 0.05. The default value of α is set to be 15%.

The test video stream is the Paris sequence encoded with H.264/AVC. The average bit rate of stream S is 116 Kbyte/s. The average video quality of the media is 40.12dB. The size of a generation κ is eight frames, which corresponds to a group of pictures covering 0.26 seconds of video. On average, there are 32 packets in each generation. The size of the priority region Γ_l is 8κ. The size of the urgent region is 4κ. This indicates that the *playback deadline* at each peer is 3.12 seconds. All packets must arrive at client nodes in 3.12 second to be successfully played. The actual size of a video block is 1024 Bytes. Packets in each generation are encoded using network coding over the Galois Field $\mathbf{GF}(2^8)$. Coefficients of network-encoded packets are stored in the packet header and transmitted with each video packet as well, whose size depends on the number of encoding packets in the generation. In the simulation, the packet header (e.g. the address of destination peer, and the sequence number of video) is 150 Bytes, and the average size of network coding coefficients is about 32 Bytes per packet (1 Byte for the coding coefficient in GF(2^8) per packet).

We compare our proposed packet schedulers with the following approaches:

- Advanced random push with random network coding with 50ms buffer-map updating interval (ARND-50ms): The ARND is an improved version of RND similar to [78]. In RND, each sender randomly pushes its encoded packets to its neighbors according to the buffer-map of its neighbors. In ARND, senders tend to choose those generations that need more packets to be decoded before the playback deadline. The *buffer-map* which represents the data availability of its neighbor peers updates every 50ms.

- Advanced random push with random network coding with 200ms buffer-map update interval (ARND-200ms): The ARND algorithm with 200ms buffer-map updating interval.

- An asynchronous distributed scheduler (ADS) similar to [96]: In ADS, each sender independently chooses the optimal transmission policy from candidate transmission policies. This choice is based on the cost of each candidate policy. This transmission policy determines which generation and which receiver the packet is addressed to. The cost is defined as the product of the number of packets needed to recover the generation g and the corresponding reminding time of this generation

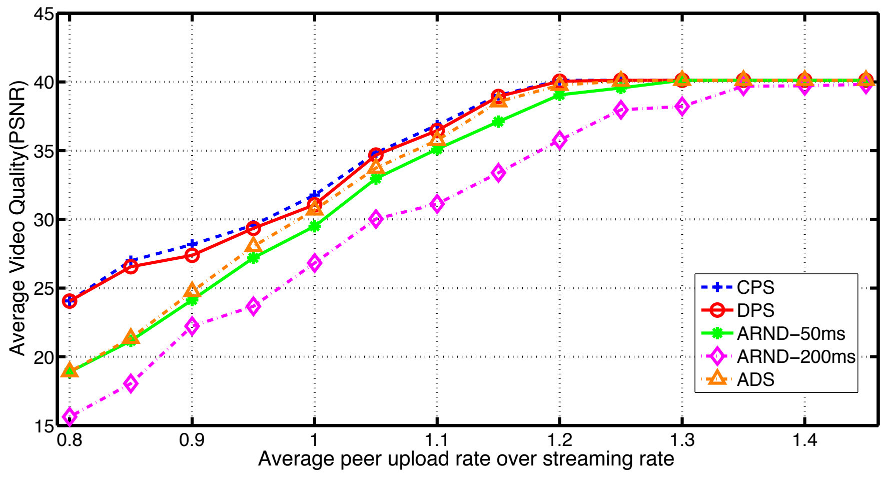


Figure 5.3: Performance comparison of the average video quality with five schemes

g before the playback deadline. All candidate policies are sorted in an increasing order according to the defined cost. The optimal policy is selected from the top 30 policies with uniform probability.

We use the following performance metrics to evaluate the QoS and bandwidth efficiencies: (1) average video quality comparison: the average received peak signal to noise ratio (PSNR) at each client node; (2) delivery ratio: the average fraction of *recoverable* packets that could arrive at the receiver node over the actual streaming packet rate before the playback point at each client node; (3) the *uninformative* packet ratio: the received *uninformative* video packet rate over all received video packet rate; (4) communication overhead ratio: the communication overhead rate over the streaming rate. Communication overhead includes buffer-map exchange messages, scheduling messages and packet request messages; (5) informative packet rate: the number of received informative packets per second.

5.4.2 Streaming Performance Comparison

To evaluate the QoS of the proposed streaming system, the performance of the average video quality is studied first. It is the most visual indicator to the quality of service. The video quality increases when the peer upload bandwidth increases from $0.8S$ to $1.45S$. The results in Fig. 5.3 show that the CPS and the DPS perform better than the other schemes over the whole range of the upload bandwidth, especially when the upload bandwidth is too low for full-rate video delivery. When the peer upload bandwidth is $1.2S$, the CPS and the DPS can achieve 40.12dB, 40.05dB respectively. By contrast, at the same bandwidth, the ADS, the ARND-50ms, and the ARND-200ms can only achieve 39.75dB, 39.06dB, and 35.76dB respectively. This is because the communication overhead and *uninformative* transmission waste network bandwidth. When the peer upload bandwidth is $0.8S$, the DPS shows significant improvements over ADS, ARND-50ms, and ARND-200ms algorithms; 5.14dB, 5.16dB, 8.43dB respectively. This is because the ADS and ARND cannot avoid *unrecoverable* transmissions that waste bandwidth resources and lead to poor transmission of other generations. In contrast, the APA effectively skips some generations so that all transmitted packets are recoverable. The DPS achieves slightly lower video quality compared with the CPS algorithm over the whole range of bandwidth values. That is because the CPS is a global optimization algorithm while the DPS is a local optimization algorithm.

In Fig. 5.4, the performance of packet delivery ratio among these five methods is studied when the peer upload bandwidth ranges from 0.8S to 1.45S. The better delivery ratio will bring smoother playback of videos. In this experiment, when the peer upload bandwidth is $0.8S$, the delivery ratio of CPS, DPS, ADS, ARND-50ms, and ARND-200ms is 61.03%, 60.31%, 47.05%, 47.01% and 38.9% respectively. This demonstrates that our algorithms can improve 13%-22% in delivery ratio compared with those three scheduling algorithms. This improvement is mainly because the proposed APA algorithm can efficiently avoid *uninformative* transmission. Furthermore, the CPS and DPS can

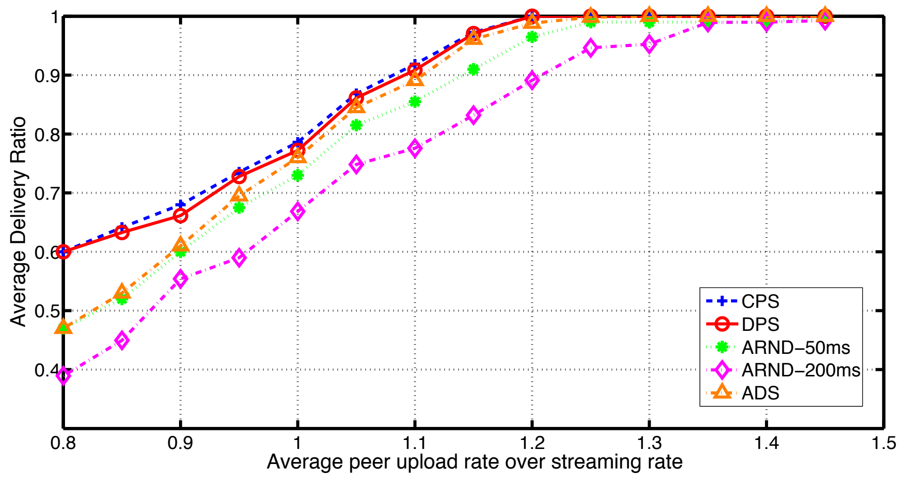


Figure 5.4: Performance comparison of the average delivery ratio with five schemes

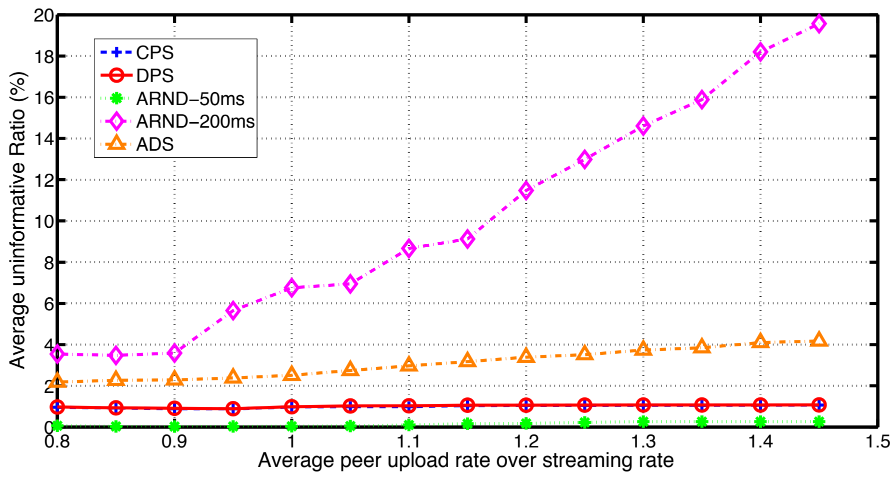


Figure 5.5: Performance comparison of the average uninformative packet ratio with five schemes

achieve about 4% more delivery ratio compared with the ADS algorithm when the upload bandwidth of peers is $1.2S$. This improvement is mainly due to the CPS and the DPS generating less communication overhead and *uninformative* packets. On the whole, this experiment demonstrates that our algorithms can achieve better recoverable packet delivery ratio when the upload bandwidth of peers ranges from $0.8S$ to $1.45S$.

To evaluate the bandwidth efficiencies of the streaming network, we also observe the performance of the *uninformative* packet ratio in Fig. 5.5. The *uninformative* ratio is an

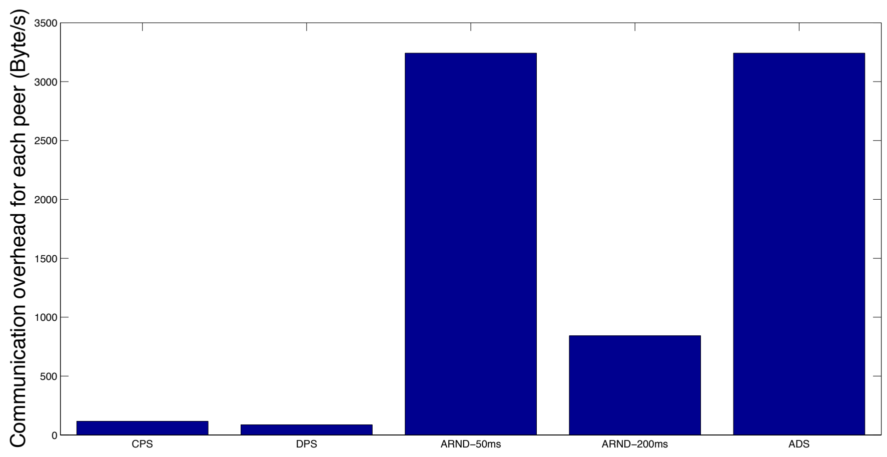


Figure 5.6: Performance comparison of the communication overhead over the streaming rate with 5 schemes

important criterion for the evaluation of bandwidth efficiencies. The experiment shows that the *uninformative* ratio of the CPS, DPS, and ARND-50ms algorithms remains stable under different upload bandwidth, and the *uninformative* ratio of the ARND-200ms increases a lot as the upload bandwidth of peers increases. It can be seen that the ARND-50ms only generates about 0.1% to 0.3% *uninformative* packets, and the CPS and the DPS generate about 1% *uninformative* packets over the whole range of peer upload bandwidth. The ADS generates about 2.2% to 4.1% *uninformative* packets, and the ARND-200ms generates about 3.5% to 19.57% *uninformative* packets. Firstly, it demonstrates that when the buffer-map updating interval of ARND is 50ms, the amount of *uninformative* transmission caused by the lack of synchronization is very small and almost negligible. Secondly, the CPS and the DPS still may generate a small amount of *uninformative* transmission. This is because the request model may bring a small number of *uninformative* transmission. Compared with the ADS and ARND-200ms, the amount of *uninformative* packets is acceptable. Thirdly, the ADS algorithm and the ARND-200ms have a higher *uninformative* ratio. This is because the ADS algorithm has the sub-optimization problem and the long buffer-map updating period of ARND-200ms brings a large number of *uninformative* transmission.

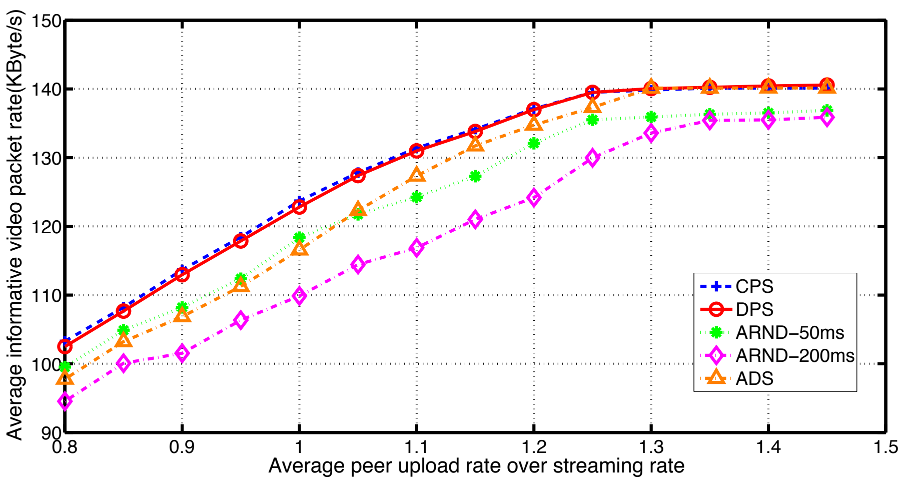


Figure 5.7: Performance comparison of the average informative video packet rate with five schemes

To further evaluate the bandwidth efficiencies of the streaming network, the average communication overhead over the streaming rate is studied in Fig.5.6. The communication overhead of the ARND and the ADS is the bandwidth resources used for buffer-map updating. The communication overhead of the CPS and the DPS is the spent bandwidth resources for packet scheduling and the spend bandwidth resources for peer communications. As depicted in Fig. 5.6, the CPS and the DPS have a relatively low communication overhead while the ARND-50ms and the ADS have a higher overhead. This is because the ARND-50ms and the ADS algorithm frequently exchange buffer-map among neighboring nodes. By contrast, the CPS only transmits the scheduling results from the tracker node to each client node and the DPS only exchanges the information among peers during the handshake procedures and when peer nodes experience bandwidth variations. Therefore, compared with the communication overhead of buffer-map updating, the amount of communication overhead of CPS and DPS is relatively small.

We also evaluate the actual average *informative* packet rate in Fig.5.7 to observe the joint impact of the communication overhead and the *uninformative* transmission when the upload bandwidth of peers increases. It can be seen that the proposed CPS and DPS have a better informative packet rate compared with the other two methods over

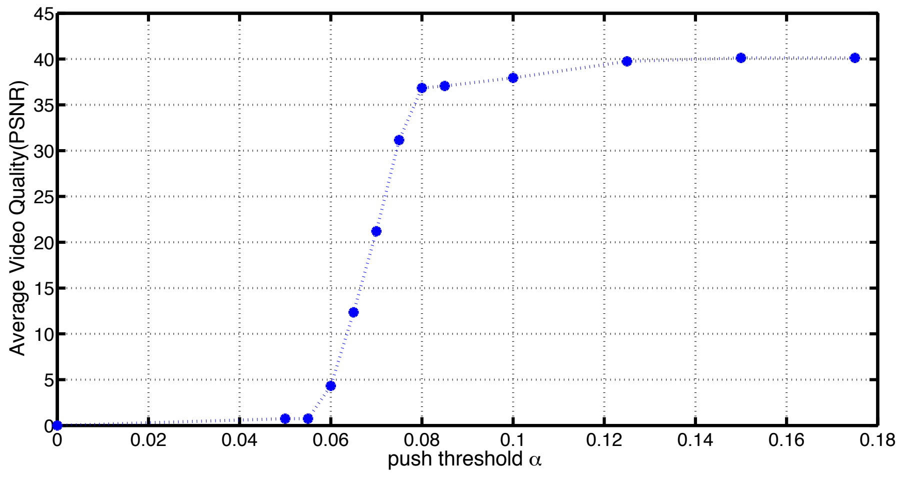


Figure 5.8: The average delivery ratio versus different values of α for each node

the whole range of peer upload bandwidth values. Although the CPS and the DPS have about 1% more *uninformative* packet ratio than the ARND-50ms algorithm, the CPS and the DPS get more *informative* packet rate. That is because that the communication overheads of ARND-50ms use more upload bandwidth of peers.

The impact of the threshold α can be seen in Fig.5.8. This figure shows that a sender node needs to have at least α segments before it becomes a sender of other client nodes. We can see that a lower threshold cannot guarantee that enough linear independent packets are available at the sender nodes, thereby leading to a poor video quality.

In Fig. 5.9, the performance of the delivery ratio is analyzed in a lossy network to evaluate its resistance to network loss. In this experiment, we increase the loss rate and then observe the change in the delivery ratio. Generally, the delivery ratio decreases when the loss rate increases for all schemes. That is because the useful upload bandwidth decreases when the loss rate increases. The results in Fig. 5.9 show that the decreasing rate of both CPS and DPS is similar to the decreasing rate of other scheduling algorithms. It proves that the scheduling compensation model can accurately estimate the number of needed packets for client nodes in a lossy network. The loss in the network will not bring extra scheduling errors to CPS and DPS.

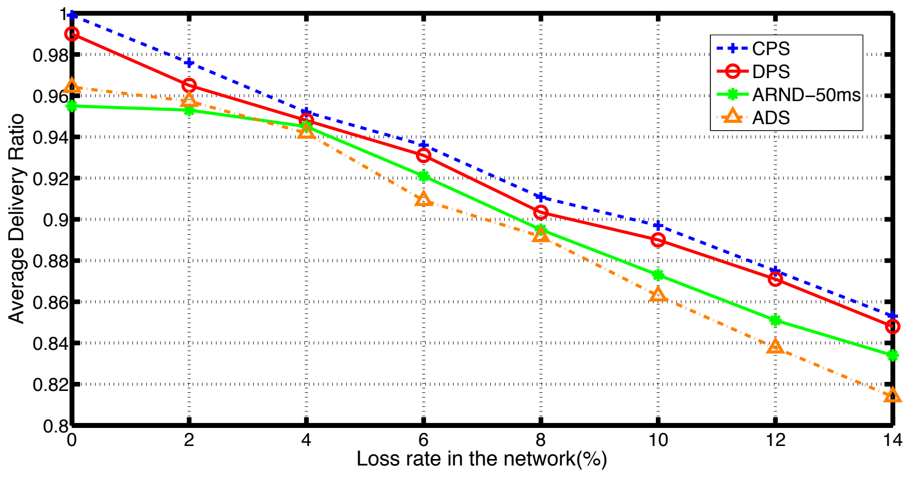


Figure 5.9: Performance comparison of the average delivery ratio as a function of network loss rates when $N = 100$ and upload bandwidth $\bar{U} = 1.2S$

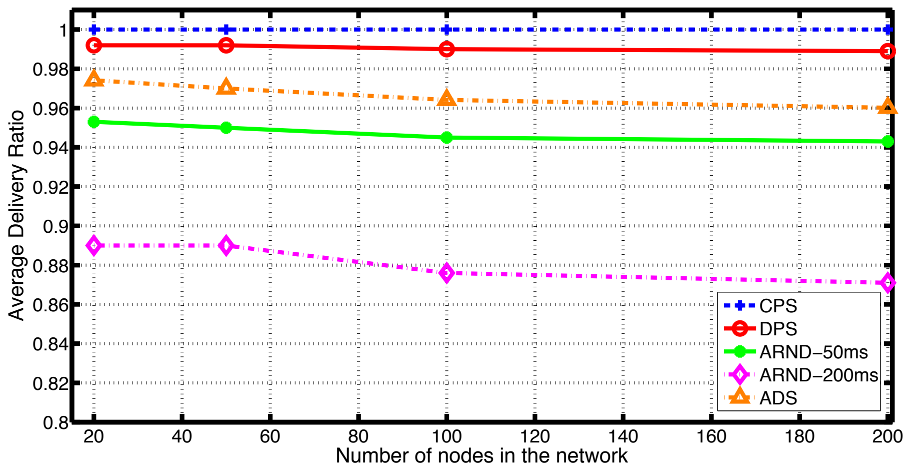


Figure 5.10: Performance comparison of the average delivery ratio in networks of different network sizes when the upload bandwidth $\bar{U} = 1.2S$

In Fig. 5.10, we evaluate the scalability of our proposed packet schedulers by varying the number of peers from 20 to 200 peers. Fig. 5.10 shows that CPS and DPS offer steady delivery ratio, whereas the ADS, ARND-50ms, and ARND-200ms have a slightly decreasing delivery ratio as the number of peers increases. It demonstrates that all these algorithms can be applied to a large-scale P2P network. The property of network coding and push-based P2P network make these packet scheduling algorithms very competitive

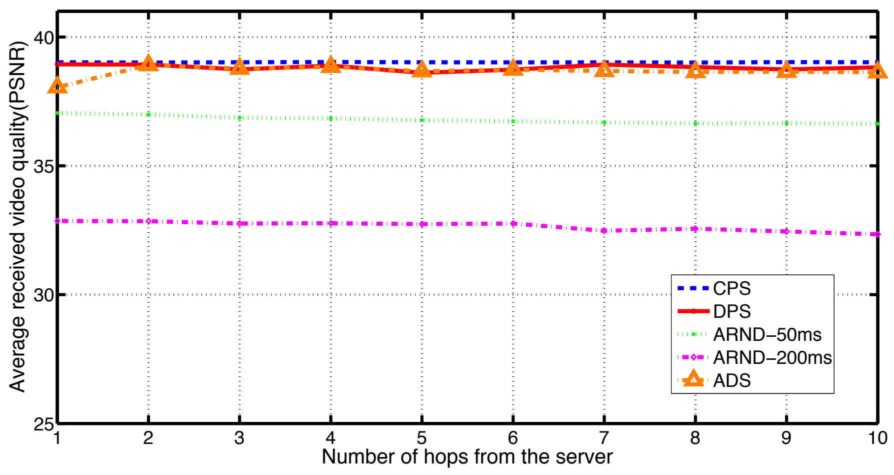


Figure 5.11: The impact of distance from server to the average received video quality in PSNR

regarding the scalability.

In the last experiment, the performance of the video quality in PSNR versus the number of hops from the server is evaluated. This experiment can demonstrate the scalability of the scheduling algorithm and the fairness among client nodes. The hop is defined as the hop distance from the client node $\mathcal{N}_j$ to the streaming server. As depicted in Fig. 5.11, the average video quality remains almost the same as the number of hops increases. This shows that all algorithms are not very sensitive to the distance from the server. That is because that the CPS, DPS, ADS, and ARND algorithms are built in the *mesh-push* network, where the neighbor nodes of each client node are randomly chosen. Therefore, the distance from the streaming server does not have an obvious influence on the delivery ratio.

5.5 Summary

This chapter proposes a centralized and a distributed bandwidth-efficient packet scheduling algorithm for non-scalable live streaming applications. We found that the mesh-push streaming network has the problem of bandwidth inefficiency due to the unsynchronized communication among peers. The bandwidth inefficiency is caused by the uninformative and unrecoverable NC-packet transmission. Furthermore, increasing the communication messages can not solve this problem because the communication overhead will also lead to bandwidth inefficiency. To solve this problem, we propose a new transmission mechanism where packet transmission are scheduled in advance, and peers follow the scheduling results to transmission packets. In this way, we can avoid the problem of unsynchronized communication and too much communication overhead at the same time.

A centralized packet scheduling algorithm is proposed to find the globally optimal packet scheduling policy that minimizes the overall bandwidth cost. It determines how senders cooperatively allocate their upload bandwidth so that the least bandwidth resources can be used to deliver the scheduled generations successfully. To add a critical scalability property to the proposed approach, we propose a distributed packet scheduling algorithm to find the local optimal solution by constructing a pre-allocation procedure and solving the single layer linear programming problem using the Lagrangian algorithm. Our experiments show that our packet schedulers achieve better video quality and delivery ratio, a lower redundant packet ratio and more informative packets when the peer upload bandwidth varies. Furthermore, the communication overhead is relatively low compared with overhead in other push-based streaming networks. The robustness of lossy networks and the network scalability is also verified through different experiments.

Chapter 6

Multiple Generation Scheduling for Scalable Video Streaming with NC

Network coding (NC) brings substantial improvements in terms of throughput and delay in collaborative media streaming applications. A key aspect of NC-driven live peer-to-peer streaming is the packet scheduling algorithm, which organizes the bandwidth allocation of sender nodes to maximize the overall Quality-of-Service. Indeed, unorganized bandwidth resources allocation usually results in significantly inefficient bandwidth utilization, which in turn leads to an unsatisfied video quality. In this chapter, a novel multiple-generation scheduling(MGS) algorithm is proposed to find an optimal class subscription policy that greatly helps to overcome the problem of inefficient bandwidth utilization. This algorithm can make sure that the upload bandwidth is reasonable utilized, and the overall perceived video quality at the receiver nodes are maximized. The algorithm is formulated as a video quality maximization function under the available bandwidth resources constraints and solved through a dynamic programming algorithm. Both implementation and analytical considerations of the proposed algorithm are described in this chapter. Experimental results confirm that the proposed algorithm deliver better quality-of-service regarding improved video quality and delivery ratio, and higher bandwidth efficiency.

6.1 Motivation

In the scalable video transmission system, packets are grouped into different **classes** according to the generation and layer they belong to. Each class needs different bandwidth resources, and brings different gain in video quality. Many existing works adopt to the "random-push", "rarest-first", "priority-first", and "emergency-first" policies to schedule packet transmission [110], [96]. However, in the NC-driven SVT system, it usually will result in unrecoverable and uninformative transmission. Some optimization works are proposed to schedule packet transmission to reach an optimized bandwidth resources allocation [98], [100], [112]. However, these optimizations for packet scheduling among these classes are only performed in each single generation separately, which leads to an unsatisfied overall bandwidth utilization. In [112], Nguyen at al. proposed to use a *drop-threshold* and an *add-threshold* at the receivers' buffer to find the most suitable class subscription policy. However, the bandwidth resource allocation based on the buffer level is fluctuant and inaccurate. In [98], Sanna and Izquierdo proposed to use average upload bandwidth to schedule the class subscription. Furthermore, in [100], Thomos et al. proposed to use the average upload bandwidth to schedule the rate allocation among classes for each single generation in a lossy network. However, both of their work optimize the bandwidth resource based on the single generation, and cannot fully utilize the bandwidth resource.

In Section. 4.4.4 and Section. 5.2.4, we propose to use the cumulative upload bandwidth to perform the packet scheduling. However, these systems still do not fully utilize the bandwidth resource. Different from them, this chapter proposes to schedule multiple generations at the same time to make fully use of the bandwidth resources in the priority region. The advantage is that the bandwidth resources can be more accurately allocated to different classes such that the maximum bandwidth efficiency can be achieved.

The multiple generation scheduling(MGS) problem is formulated as a perceived video quality maximization problem in some upload bandwidth constraints. Through solving the maximization algorithm, the optimal class subscription policy can be found. The MGS problem is solved using two methods. One transfers the MGS to the single generation optimization(SGS) algorithm. The other solves the MGS directly using a dynamic programming algorithm. The receiver node calculates the optimal class subscription policy in the priority region, and subsequently, inform sender nodes the scheduling results. Sender nodes then cooperatively transmit the selected classes to the receiver node according to the distributed packet scheduling proposed in Section 5.3.2. When the network suffers from unpredictable network vitiations, a request model is used to help with the unrecoverable transmission.

Although both Chapter 4 and this Chapter work on the scalable video transmission over the loss network. They still have many differences in the design principles. First, the ORA algorithm proposed in Chapter 4 schedule packet transmission only based on the cumulative available upload bandwidth, and the scheduling is performed for each single generation. While the MGS algorithm in this chapter relies on the multi-generation scheduling results. Second, the ORA algorithm deals with the network loss based on a probability model, while the MGS and the SGS deal with the network loss based on the value of expectation. The probability model can estimate a more accurate video quality gain, but it suffers from a larger computation complexity. Therefore, The ORA is not suitable to be used in the multi-generation scheduling. Third, the ORA algorithm transmits the scheduled layer from the base layer to the top layer subsequently, while the SGS and the MGS algorithm transmit the subscribed class directly. In MGS, the enhancement layers are subscribed only when the lower layers are subscribed first. Therefore, the MGS allows peers to encode packets from the base layer to the highest subscribed layer together, and transmit the highest subscribed layer directly, which can reduce the scheduling complexity.

The remainder of this chapter is organized as follows: In Section 6.2, the general system model and media model used in this chapter are introduced. Additionally, we also detailed describe the overall transmission procedure of the proposed adaptive scheduling algorithm. The scheduling compensation model and the request model proposed in Chapter. 5 is reused in this chapter. Based on these fundamental system design, a multi-generation scalable class subscription algorithm is proposed in Section 6.3. We further proposed two solutions to this problem. Firstly, in Section 6.3.2, we propose a solution by introducing a packet scheduling algorithm based on a single generation optimization(SGS). This algorithm has a very low computation complexity. Then the solution to multi-generation scheduling (MGS) is proposed in Section 6.3.3. The multi-generation scheduling problem is solved by a novel dynamic programming algorithm. A refined distributed packet scheduling algorithm is proposed in Section. 6.4. Finally, the proposed scheduling algorithms are evaluated by comparing the QoS and bandwidth efficiency with other SVC-based packet scheduling algorithms.

6.2 System Model

In this section, a brief overview of the system is given. Firstly, the definitions of the transmission network are given, and the construction process of the transmission network is briefly explained. Secondly, the model for scalable media data is explained to understand the scheduling process and coding process of scalable data.

6.2.1 Network Model

In the network model, three types of peers are defined: *streaming source*, *tracker server*, and *client node*. Each peer contacts the tracker server to join the network. Some control information, i.e., the upload bandwidth of neighboring nodes and the size of video packets, is exchanged during the procedure. The overlay of network nodes is represented as a graph $(\mathcal{N}, \varepsilon)$ composed for nodes $\mathcal{N} = \{\mathcal{N}_0, ..., \mathcal{N}_N\}$ and the edge ε. $\mathcal{N}_0$ represents the *tracker server* and $\mathcal{N}_1$ represents the *streaming source*. The rest nodes from $\mathcal{N}_2$ to $\mathcal{N}_N$ are client nodes. A client node becomes a sender node when it holds at least α linear independent segments $(0 \leq \alpha \leq 1)$. $A_j \subset \mathcal{N}$ is defined as the neighborhood of $\mathcal{N}_j$, which is the set of sender nodes that are connected to $\mathcal{N}_j$.

For any nodes in the network, $U = \{U_0, ...U_N\}$ is indicated as the vector of streaming nodes. To simplify the discussion, we also define the κ-quantized upload bandwidth as $\tilde{U}_i = U_i * \kappa/V$, where κ is defined as the playback duration of a generation, V is the video packet size. The unit of $\tilde{U}_i$ is the number of packets/κ seconds. Each sender $\mathcal{N}_i$ equally allocates its upload bandwidth U_i to its receiving nodes $\mathcal{N}_j$, and the allocated upload bandwidth from node $\mathcal{N}_i$ to $\mathcal{N}_j$ is U_{ij}. The overall available upload bandwidth of $\mathcal{N}_j$ is $\hat{U}_j = \sum_{k=1}^{|A_j|} U_{kj}$. The available κ-quantized bandwidth of $\mathcal{N}_j$ in κ seconds is $\bar{U}_j = \hat{U} * \kappa/V$. As the scheduling algorithm is a distributed packet scheduling algorithm and designed for each receiver node $\mathcal{N}_j$, $\bar{U}$ is used to represent $\bar{U}_j$ for simplicity.

6.2.2 Media Streaming Model

The scalable media stream which is distributed to the network nodes is modeled as a bi-dimensional array of generations shown in Fig.6.1. A generation is made up by one or several group of pictures (GOP) in the media. Each generation is identified within the media stream by a temporal index $g \in [1, G]$. Each generation with identical temporal index g has the same duration κ. Each generation g is subdivided into several layers $l \in [1, L]$. In the following section, the notation (g, l) is used to indicate the **class** with a temporal index g and layer index l. Each class (g, l) has $s_{(g,l)}$ blocks. $Q(g, l)$ is the quality enhancement of each class (g, l). Due to the dependency among layers, the lower layer needed to be decoded first before the higher layer can be decoded. When the highest layer L is decoded, the quality gain of generation g is $\sum\limits_{l=0}^{L} Q(g, l)$.

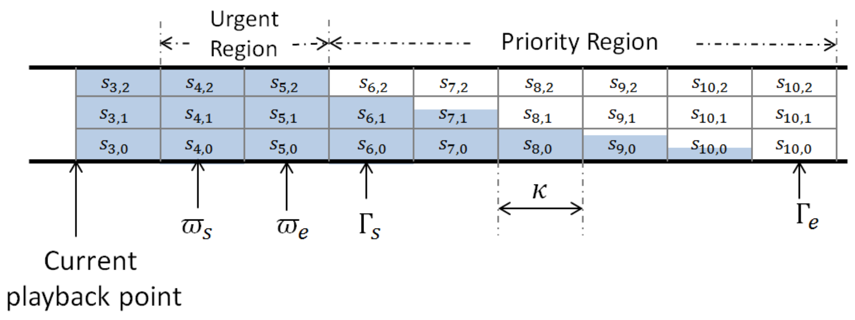

Urgent Region
Priority Region
$s_{3,2}$ $s_{4,2}$ $s_{5,2}$ $s_{6,2}$ $s_{7,2}$ $s_{8,2}$ $s_{9,2}$ $s_{10,2}$ $s_{10,2}$
$s_{3,1}$ $s_{4,1}$ $s_{5,1}$ $s_{6,1}$ $s_{7,1}$ $s_{8,1}$ $s_{9,1}$ $s_{10,1}$ $s_{10,1}$
$s_{3,0}$ $s_{4,0}$ $s_{5,0}$ $s_{6,0}$ $s_{7,0}$ $s_{8,0}$ $s_{9,0}$ $s_{10,0}$ $s_{10,0}$
ϖ_s
ϖ_e
Γ_s
κ
Γ_e
Current playback point

Figure 6.1: Sample media stream model for scalable-coded media stream.

Instead of transmitting raw video packets, the P2P network transmits network-coded packets. A new coded packet is generated by combining the blocks belonging to one generation together with random coefficients. The algebraic operations are performed in Galois field F_q (usually $q = 2^8$). In this system, the blocks are encoded from the base layer to the subscribed layer l together.

$$y = \sum_{i=1}^{s_{(g,1)}+s_{(g,2)}+\ldots+s_{(g,l)}} b_i c_i \tag{6.1}$$

In the Eq.6.1, $s_{(g,l)}$ is the number of packets in generation g of layer l, b_i is the ith raw blocks in generation g and c_i is a random coding coefficient. When a receiver receives y, it checks if the packet is *informative*. If the packet is *uninformative*, it is discarded. Once a client node has collected enough informative packets in a generation, it will recover this generation using the progressive Gauss-Jordan elimination.

To achieve a guaranteed transmission, the whole *transmission region* can be further divided into *urgent region* and *priority region*. In Fig.6.1, $g = 3$ is the playback point. The urgent region ω is a sliding window next to the playback point with fixed size ω_l. The priority region Γ is a sliding window next to the urgent region with fixed size Γ_l. The start point and the end point of the priority region is defined as Γ_s and Γ_e respectively.

In the *priority region*, the data transmission follows the multiple generation scheduling

Table 6-A: NOTATION USED IN THE SYSTEM MODEL

$\mathcal{N}_i$	Network node i, $i \in [0, N]$
A_j	Neighborhood of the receiver node $\mathcal{N}_j$
g	Temporal index (generation) in the media, $g \in [1, G]$
κ	Duration of each generation
V	Size of video packets
U_i	Upload bandwidth of streaming node $\mathcal{N}_i$, $i \in [1, N]$ [kByte/sec]
U_{ij}	Upload bandwidth allocated from $\mathcal{N}_i$ to $\mathcal{N}_j$ [kByte/sec]
$\hat{U}_j$	Overall available upload bandwidth from all senders to the receiver node $\mathcal{N}_j$ [kByte/sec]
U_j	The number of packet can be received from A_j in each generation [packets/ κ sec]
U	Abbreviation of U_j
S	Average streaming rate
α	Independent threshold for a receiver node to become a sender node
Γ	Priority region, $\Gamma = [\Gamma_s, ..., \Gamma_e]$
ω	Urgent region, $\omega = [\omega_s, ...\omega_e]$
l	Layer index in the media, $l \in [1, L]$
$s_{(g,l)}$	Number of video packets in generation g, layer l
$s_{(\hat{g},l)}$	Number of actually scheduled packets in generation g, layer l
$Q_{(g,l)}$	Number of quality gain of generation g, layer l
μ	The unsuccessful transmission rate
d_{ij}	Number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g
$t_{(g,l)}$	Denotes whether generation g and layer l is subscribed by the receiver, $t_{(g,l)} \in (0, 1)$.

algorithm proposed in Sec. 6.3 to achieve an optimized system efficiency. In the *urgent region*, transmission follows the request model proposed in Sec. 6.5 to further improve the delivery ratio. This design can make sure a reliable system performance when the network bandwidth flutters. Any unrecoverable generations out of the transmission region are discarded automatically.

6.2.3 Overall of Transmission Process

The overall of the video data transmission process is summarized in Fig. 6.2. First, each receiver contacts with the tracker node to obtain the video megadata (number of packets in each layer and each generation), link information (the available upload bandwidth U_{ij} between the sender node $\mathcal{N}_i$, and the receiver node $\mathcal{N}_j$ and the unsuccessful transmission rate μ). Then the receiver node calculates the compensated data parameter according to Scheduling Compensation Model (SCM) proposed in [113]. The main function of SCM is to calculate the number of packets $s_{(\hat{g},l)}$ that need to be sent from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ such

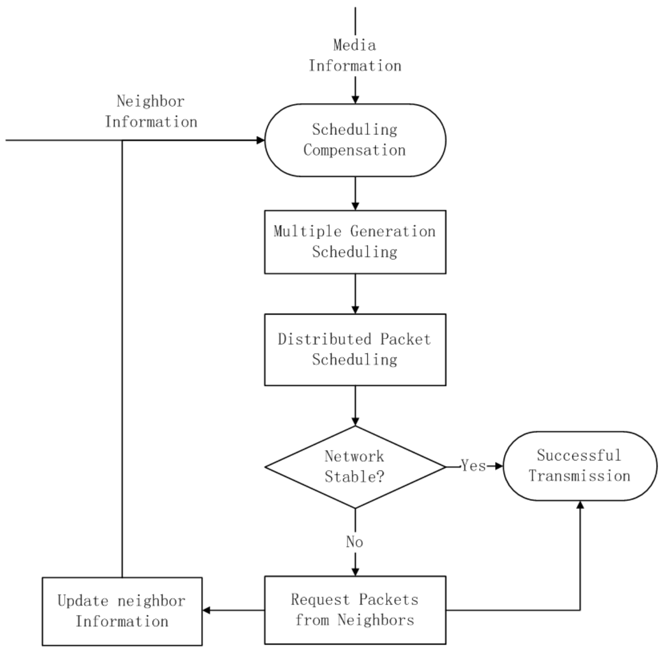

Media Information
Neighbor Information
Scheduling Compensation
Multiple Generation Scheduling
Distributed Packet Scheduling
Network Stable?
Yes
Successful Transmission
No
Update neighbor Information
Request Packets from Neighbors

Figure 6.2: Illustration of the overall transmission procedure at each client node

that $s_{(g,l)}$ independent packets can successfully arrive at $\mathcal{N}_j$. After this, each receiver determines its subscribed layer l for each generation g in the *transmission region* based on the multiple-generation scheduling algorithm(MGS) and single-generation scheduling algorithm(SGS) proposed in Sec. 6.3. Subsequently, according to the request of each receiver, senders cooperatively transmit the selected class (g, l) to the receiver according to the distributed packet scheduling algorithm proposed in Sec. 6.4.

When the network is stable, the subscribed class (g, l) can be successfully transmitted to the receiver. When the network is unstable, the subscribed class (g, l) may not be decoded in the priority region. Then when the class (g, l) is in the *urgent region*, the receiver will actively request packets from its neighbor nodes according to the request model proposed in Sec. 6.5. The *urgent region* and the *priority region* moves as the playback point moves.

6.3 Multiple Generation Scheduling

This section proposes a multiple-generation scheduling optimization problem to make fully use of the bandwidth resources in a given region. It determines which class should be subscribed and transmitted so that the overall video quality in the priority region can be maximized. The problem is formulated as a multi-generation PSNR maximization problem, and the optimal class subscription policy can be found through solving the maximization problem under some bandwidth constraints. To solve the scheduling problem, two algorithms are proposed. First, the problem is reformulated and solved through a single generation scheduling algorithm. Second, the problem is solved directly through a dynamic programming algorithm. As the proposed optimization problem is a variation of the Multiple-Choice Knapsack Problem(MCKP) [114]. Although the MCKP problem has been proved as an NP-complete problem in [115], the optimal solution can be get in pseudo polynomial-time through the dynamic programming algorithm[116]. Therefore, according to the classic dynamic programming for MCKP, we solved the multiple generation scheduling problem using the dynamic programming algorithm.

6.3.1 Problem Formulation

In this section, a multiple-generation scheduling problem is proposed to find the most suitable class subscription policy. This problem finds the optimal layer subscription policy for each generation to maximize the received video quality, which can also reduce the uninformative and unrecoverable transmission. We find the optimal class subscription strategy by proposing a video quality maximization problem under the fully video packet recovery constraint. It chooses a suitable layer for each receiver to subscribe based on the estimated bandwidth $\bar{U}$. The multiple generation scheduling is performed in the priority region Γ. The optimization problem is shown in Eq. 6.2.

For any generation g, and any layer l in the region, $t_{(g,l)} \in (0, 1)$ denotes whether generation g and layer l is subscribed by the receiver. The $s_{\hat{(g,l)}}$ is used to represent the actually needed number of packets to transmit the class (g, l) after considering the scheduling compensation model proposed in [113]. The scheduling algorithm is computed every Γ_l.

$$\begin{aligned}
& t^*_{(g,l)} = \arg\max_{t_{(g,l)}} \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{l=0}^{L-1} Q_{(g,l)} t_{(g,l)} \\
& \text{subject to} \sum_{g=\Gamma_s}^{k} \sum_{l=0}^{L-1} t_{(g,l)} s_{\hat{(g,l)}} \leq \sum_{g=\Gamma_s}^{k} \bar{U}_g \text{ for } \Gamma_s \leq k \leq \Gamma_e \qquad (1) \\
& \qquad t_{(g,l)} \in \{0,1\} \qquad (2) \\
& \qquad t_{(g,m)} \geqslant t_{(g,n)}, \quad \forall m \leqslant n \qquad (3)
\end{aligned} \tag{6.2}$$

The constraint (1) in Eq.6.2 means that the sum of the scheduled packets $s_{\hat{(g,l)}}$ from Γ_s to any instant k should be smaller than the sum of available upload bandwidth from Γ_s to this instant k. $\bar{U}_g$ is equal to the available upload bandwidth in generation g, and it is equal to the $\bar{U}$ because the average upload bandwidth is considered to remain stable during the transmission period. The constraint (2) means that each class (g, l) can only be transmitted or not transmitted. The constraint (3) means that the lower layer needs to be chosen before the higher layer is chosen.

6.3.2 Single Generation Scheduling

The class subscription algorithm selects an optimal layer for each receiver to subscribe for each single generation. To provide a faster and easier solution to class subscription, we propose a single generation scheduling algorithm based on the Eq.6.2. This problem finds the layer subscription policy based on the bandwidth information in a single generation.

Algorithm 3: Single Generation Scheduling algorithm

```
for $g \leftarrow \Gamma_s$ to $\Gamma_e$ do
    for $l \leftarrow 0$ to $L-1$ do
        if $\bar{U} > \sum_{k=0}^{l} s_{\hat{(g,k)}}$ then
            $t_g = l$ ;
        end
    end
end
```

$$
\begin{aligned}
t^*_{(g,l)} &= \arg\max_{t_{(g,l)}} \sum_{l=0}^{L-1} Q_{(g,l)} t_{(g,l)} \\
\text{subject to} & \sum_{l=0}^{L-1} t_{(g,l)} s_{\hat{(g,l)}} \leq \bar{U}_g \qquad (1) \\
& t_{(g,l)} \in \{0,1\} \qquad (2) \\
& t_{(g,m)} \geqslant t_{(g,n)}, \quad \forall m \leqslant n \qquad (3)
\end{aligned}
\tag{6.3}
$$

In the Eq.6.3, the classes (g,l) that make $t_{(g,l)} = 1$ are chosen as the subscribed layers. To solve the single generation scheduling problem, the $\sum_{k=0}^{l} s_{\hat{(g,k)}}$ is compared with $\bar{U}_g$ for any $k \in (0, L)$ in turn. The maximum k is the subscribed layer in generation g.

6.3.3 Multiple Generation Scheduling

The optimization function of the MGS is a variation of the Multiple-Choice Knapsack Problem [114], which has been proved to be an NP-complete problem [115]. However, the optimal solution can be found in a pseudo polynomial-time through the dynamic programming algorithm[116]. According to the equation proposed in Eq. 6.2, The constraint (3) in Eq. 6.2 can be seen as a multiple choice problem. In each generation g, only one quality layer can be targeted. Therefore, the Eq. 6.2 can be treated as a Multiple-Choice Knapsack Problem and solved using the dynamic programming algorithm as presented in the Algorithm.4. The time complexity of this algorithm is $\Gamma_l * \Gamma_l * \bar{U} * L$. Although

Algorithm 4: Scalable Rate Allocation Algorithm-Find the maximum value

```
$Function1 : Optimizationfunction$
$G = \Gamma_e - \Gamma_s$; $v = \bar{U} * G$; $\mathbf{R[G+1][v+1]} \leftarrow 0$
for $g \leftarrow \Gamma_s$ to $\Gamma_e$ do
    /*Loop from the start of the priority region to the end of the priority region*/
    for $v \leftarrow \bar{U} * G$ to 0 do
        $R[g][v] = R[g-1][min(\bar{U} * (g-1), v)]$ ;
        for $l \leftarrow 0$ to $L - 1$ do
            if $v > \sum_{k=0}^{l} s_{(\hat{g,k})}$ then
                /*The available bandwidth increase $\bar{U}$ for each generation*/
                $R[g][v] =$
                $max(R[g][v], \sum_{k=0}^{l} Q_{(g,k)} + R[g-1][min((g-1) * \bar{U}, v - \sum_{k=0}^{l} s_{(\hat{g,k})})]$
            end
        end
    end
end
return $\mathbf{R}$;
```

it is not a polynomial-time solution, it will not increase as the network size increase. Besides, due to the size of the priority region is limited, and the layer of video is limited. The complexity is affordable for each node.

In the Algorithm 4, we find the maximum PSNR gain we can achieve in Γ. In Algorithm 5, we find the optimal layer subscription $\mathbf{t}$, where $t[g] = -1$ means that this generation is not subscribed, and $t[g] = l$ represents the layer l is subscribed.

After the receiver calculated the optimal class subscription strategy, it sends the strategy to each sender, and then each sender allocates its bandwidth based on the layer subscription policy based on the distributed packet scheduling algorithm proposed in Section. 6.4.

Algorithm 5: Find the solution of the optimization function

```
$Function2 : Findsolution$
$remainSpace = \bar{U} * G$ , $\mathbf{t[G]} \leftarrow 0$
for $g \leftarrow \Gamma_e$ to $\Gamma_s$ do
    $t[g] = -1$ ;
    for $l \leftarrow 0$ to $L-1$ do
        if $remainSpace >= \sum_{k=0}^{l} s_{(\hat{g,k})}$ then
            if $R[g][remainSpace] - R[g-1][min(remainSpace - \sum_{k=0}^{l} s_{(\hat{g,k})}, (g-1) * \bar{U})] == \sum_{k=0}^{l} Q_{(g,k)}$ then
                $t[g] = l$ ;
                $remainSpace = min(remainSpace - \sum_{k=0}^{l} s_{(\hat{g,k})}, (g-1) * \bar{U})$ ;
            end
        end
    end
end
return $\mathbf{t}$ ;
```

Algorithm 6: Distributed Packet Scheduling

```
$\bar{U} = \sum_{k=0}^{|A_j|} U_{ij}$ , $\varsigma_g = \sum_{l=0}^{t_g} s_{(\hat{g,l})}$
for each sender $\mathcal{N}_i$ in $A_j$ do
    $d_{ij} = U_{ij}/\bar{U} * \varsigma_g$ ;
    $\bar{U} = \bar{U} - U_{ij}$ ;
    $\varsigma_g = \varsigma_g - d_{ij}$ ;
end
return $\mathbf{d}$ ;
```

6.4 Distributed Packet Scheduling

In this section, the details of the distributed bandwidth-efficient packet scheduling algorithm are clarified. In this algorithm, the DPS algorithm proposed in our previous work [113] is refined for the scalable data streaming. The scheduling method determines how senders cooperatively contribute their upload bandwidth in transmitting the subscribed layer as shown in Alg. 6.

In general, the algorithm performs the upload bandwidth allocation for each receiver based on the chronological order that the node U_i becomes the sender of the client node U_j. ς_g is defined as the number of packets that need to be transmitted to the receiver in generation g. It is equal to $\sum_{l=0}^{t_g} s_{(\hat{g,l})}$. $\bar{U}$ is the available upload bandwidth of node N_j with the unit of [packets/generation]. d_{ij} is calculated by the product of the rate of the ith allocated upload bandwidth over all available upload bandwidth. Each sender transmits packets according to the scheduling results d_{ij}.

6.5 Request model

A request model is proposed to deal with the *unrecoverable* transmission caused by unpredictable network variations. Although the scheduling compensation model considers the peer churn and loss rate in the network, the peer churn rate and the loss rate may change with time. When the receiver node has an unrecovered generation in its priority region, it will periodically broadcast its buffer map as a request signal to all neighboring nodes. One or more request signals from different receivers may arrive at the sender nodes. Then, each sender with spare upload bandwidth capacity calls the "random push algorithm" to push packets to the receivers. When a receiver successfully decodes the corresponding generation, it immediately sends its buffer-map as a stop signal to all neighboring nodes. This mechanism aims at improving the recoverability of the streaming in the system. To keep the efficiencies of the system, a local information updating procedure is called, and the packet scheduling algorithm is recomputed for this node if more than 10% of packets in the *urgent region* are requested from senders or more than 10% of packets in the *priority region* are *uninformative.*

6.6 Performance Evaluation

6.6.1 Experiment Settings

All experiments are based on the same P2P streaming network to achieve a fair comparison. The network is implemented on the event-based Network Simulator 2(NS2). All end nodes independently choose its neighbors and then form a randomly connected network. The size of the network is set to be 100 nodes. Each node randomly selects 20 nodes as its neighbors. The loss rate is 0.02. As for the download bandwidth of users, as commonly assumed in other P2P system studies, we simulate a P2P network where only the peer upload bandwidth is the bottleneck.

6.6.2 Testing Media

The testing video streaming is the Paris sequence encoded with H.264/SVC using Medium Grain Scalability. In our scenario, we have a single source node, streaming a video encoded with H.264/SVC using Medium Grain Scalability at CIF resolution. The source node is the only node that is not consuming the video. We make use of the Joint Scalable Video Model (JSVM) reference software. Three priority classes are selected, exploiting quality scalability, where the base layer has QP equal to 30, and the enhancement layer has QP equal to 22 and medium grain scalability vector partitions equal to 6 and 10, for enhancement layer 1 and 2, respectively. We use the Paris sequence with CIF spatial resolution (352*288) and 30 frames per second. The compressed video streams account for an average bit rates and PSNR of:

	bitrate [KBps]	PSNR [dB]
Base layer	41.6881	35.6259
Enhancement layer 1	78.7615	37.3439
Enhancement layer 2	113.7042	40.12

Generation size c is eight frames, which corresponds to a group of pictures covering 0.26 seconds of video. Therefore, κ is 0.26 in this experiment. The number of packets in each generation varies from 19 to 64 packets per generation based on the encoding result of the video, and the average number of packets in each generation is 32 packets on average. The size of the priority period is $8c$. It means that the *playback deadline* at each peer is 2.13 seconds and video packets need to arrive at each node in 2.13 seconds in order not to expire. The maximum upload bandwidth of streaming source is set to satisfy 15% end users. The actual size of the video block is 1024 Bytes. Coefficients of network-encoded packets are stored in the packet header and transmitted with the video packets as well, whose size depends on the number of encoding packets in the generation. In NS2 simulation, the packet header (e.g. the address of destination peer, and the sequence number of video) is 150 Bytes, and the average size of network coding coefficients is about 32 Byte per packet (1 Byte for the coding coefficient of each video packet in the generation).

6.6.3 Performance Comparison

In this section, our proposed single generation optimization scheduling(SGS) and multiple generation scheduling(MGS) is compared with the hierarchical network coding approach (HNC)[110], and a ADS algorithm[96]. In HNC, each node randomly pushes its encoded packets to its neighboring nodes according to the buffer-map of its neighboring nodes. Different from the R2 for non-scalable video streaming, in scalable video streaming, the enhancement layers are pushed to each receiver node only when the base layers are successfully decoded by this receiver node. The buffer-map which represents the data availability of its neighboring peers updates every 100ms. Furthermore, based on the HNC algorithm, to improve the informative transmission ratio over the network, two more implementations are added. First, a 15% buffer threshold for each generation is used before senders push packets. Second, the number of received packets and the number of sent packets from the receiver node and the sender node are compared to avoid useless transmission. In ADS, each sender independently chooses the optimal transmission policy from candidate transmission policies. This choice is based on the cost of each candidate policy. This transmission policy determines which generation and which receiver the packet is addressed to. The cost is defined as the product of the number of packets needed to recover the generation g and the corresponding reminding time of this generation g before the playback deadline. All candidate policies are sorted in an increasing order according to the defined cost. The optimal policy is selected from the top 30 policies with uniform probability. For each receiver, the lower layer is pushed first.

In the following experiments, we only consider a static P2P network, where peers do not churn during simulation. Our evaluations focus on the following metrics: (1) Average video quality comparison: The average received peak signal to noise ratio (PSNR) at each client node. (2) Delivery Ratio: The average fraction of the average received informative packet that could arrive at the receiver node before the playback point at each client node. (3) The uninformative ratio: The ratio of the number of uninformative video packets to the number of total received video packets. (4) Network size: The impact of network size on the average video quality. (5)Network loss: The impact of network loss on the average video quality.

We first study the performance of average video quality of each method when the network is set as simulation configurations. The average video quality is measured by the peak signal-to-noise ratio (PSNR) at the receiver nodes. As depicted in Figure 6.3, we compare the video quality of these five methods when the peer upload rate ranges from 0.1 to 1.4 times the full rate video. The results in Fig. 6.3 show that the proposed multi-generation scheduling (MGS) scheme achieves a remarkable improvement in video quality over the whole range of bandwidth values. This improvement is because the ADS and the HNC algorithm transmit packets based on the signaling message. In our experiment, the time delay between the time that the receiver receives enough packets to the time that senders receive this stop message is 100ms. Therefore, after the receiver

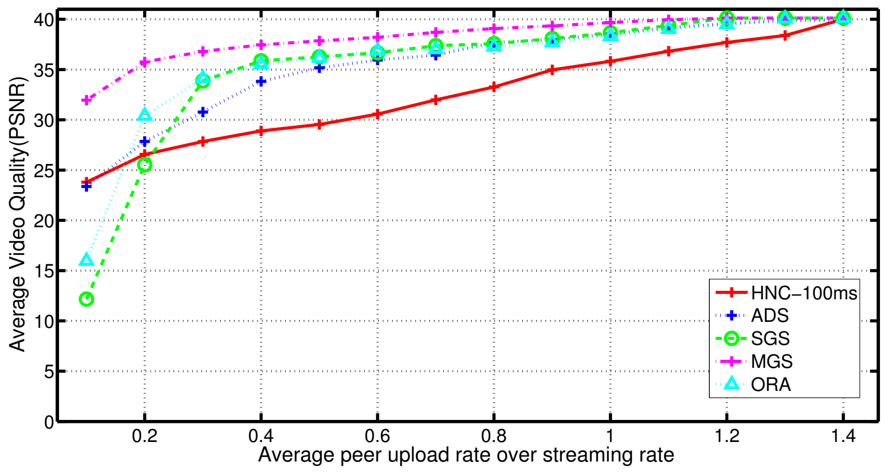


Figure 6.3: Performance comparison of the average video quality among five schemes

successfully decodes a layer in a generation, senders may still push *uninformative* packets to the receiver due to the information updating delay. Different from the HNC and the ADS algorithms, the SGS and the MGS scheduling algorithm transmit packets based on the scheduling results. Thus, our algorithms will not waste bandwidth resources in transmitting *uninformative* and *unrecoverable* packets. The received video quality of MGS is better than the HNC and ADS algorithm over the whole range of the upload bandwidth. When the average upload bandwidth is from 0.1S to 0.25S, the HNC and the ADS outperform the SGS algorithm. That is because when the upload bandwidth is small, the SGS algorithm will transmit nothing because the algorithm considers all transmission is unrecoverable. While the HNC is based on a random push policy, and the ADS is based on the rank of the each packet. Therefore, these two algorithms will have some chances to transmit decodable generations and layers, thereby bringing better PSNR than the SGS algorithm. Furthermore, the ORA algorithm outperforms the SGS algorithm when the upload bandwidth varies from 0.1 to 0.4S because the ORA algorithm uses available upload bandwidth to perform the calculate the scheduling results, while the SGS calculates the results only based on the current upload bandwidth. The SGS and the ORA achieve a similar performance when the upload bandwidth ranges from 0.4S to 1,4S. Additionally, the MGS outperforms the ORA algorithm when the upload

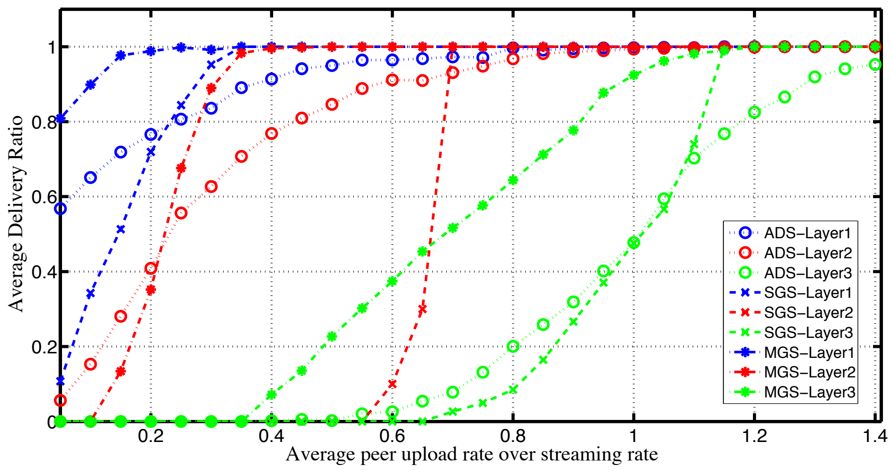


Figure 6.4: Performance comparison of the average delivery ratio among three schemes

bandwidth ranges from 0.1S to 1.4S because the MGS algorithm can make better use of the bandwidth resources.

Next in Fig.6.4, we study the performance of packet delivery ratio of each method when the peer upload rate ranges from 0 to 1.4 times the full rate video. The delivery ratio is defined as the average received informative packet rate over the streaming packet rate in each layer. Firstly, among all these scheduling algorithms, the MGS has the best delivery ratio performance. It achieves full delivery of the base layer when the peer upload bandwidth is 0.2S. In comparison, the SGS and the ADS achieve full delivery of the base layer when the peer upload bandwidth is 0.35 and 0.8 respectively. Therefore, the SGS achieves faster full delivery than ADS algorithm. However, the ADS performs better than the SGS algorithm when the upload bandwidth is from 0S to 0.2S. When applying the SGS scheduling algorithm, the system considers the available bandwidth cannot afford the base layer based on the local optimization function. It therefore determines to transmit nothing to avoid uninformative transmission. In contrast, the ADS algorithm is based on the rank of each transmission policy. Therefore, it may have some probability that some generation can be successfully decoded, and thereby bringing better delivery ratio than the SGS. When the upload bandwidth increases, the SGS will think the

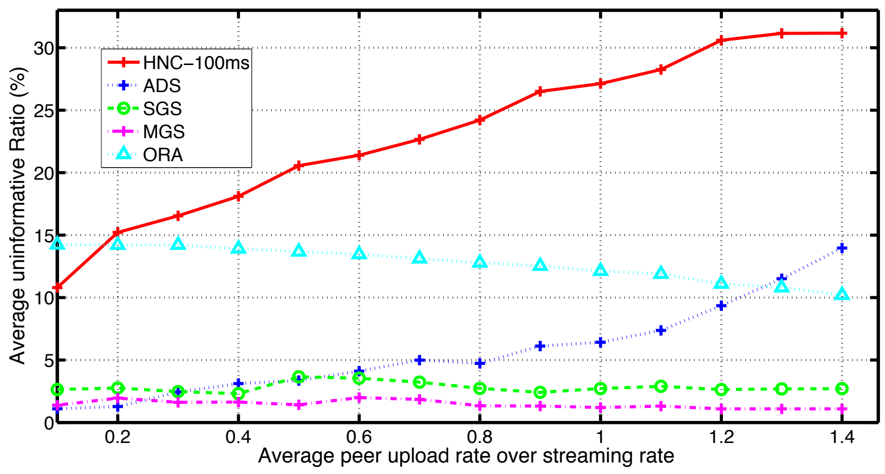


Figure 6.5: Performance comparison of the average uninformative packet ratio among five schemes

system can afford the corresponding layers and finish transmission, and bring better delivery ratio than the ADS algorithm. Comparatively, the MGS does not have this issue, because it follows the multi-generation packet scheduling algorithm. Although the average upload bandwidth cannot afford each generation, the system will automatically use this upload bandwidth to serve some selected generations, and thereby bringing better overall delivery ratio. Based on the experimental results, the multi-generation scheduling algorithms performs better than the ADS algorithm.

We further study the influence of the upload bandwidth on the performance of average uninformative packet ratio. The uninformative ratio is defined as the *uninformative* video packet rate over all transmitted video packet rate. The results in Fig. 6.5 shows that our SDP achieves about 0.2% to 0.8% *uninformative* packet ratio and the HNC has about 24.2% to 30.6% uninformative packets. By comparing the uninformative packet ratio, we find that the braking effect did generate much superfluous transmission in HNC scheme. We find that the side effect of the late buffer-map updating is more serious in the scalable video streaming than the side effect in non-scalable video streaming. This is because in scalable video streaming, the number of video packets in each layer and each generation is smaller than the number of video packets in a whole generation, and

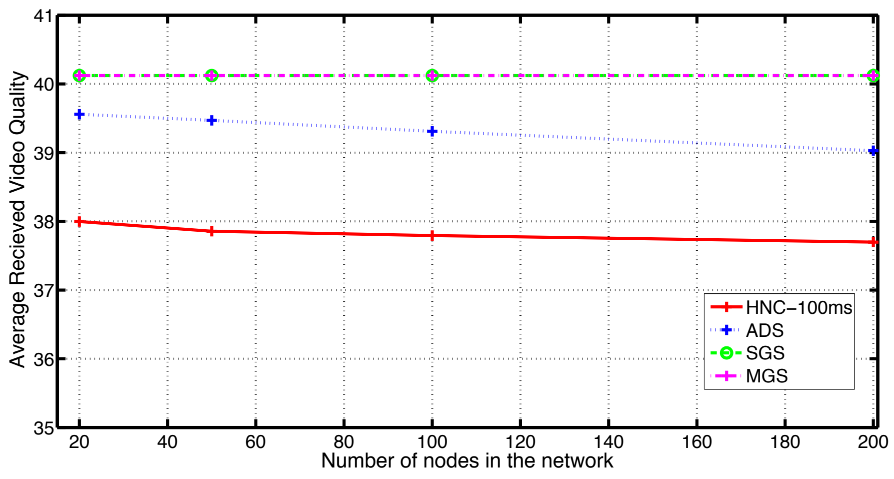


Figure 6.6: Performance comparison of the average delivery ratio as a function of network size

the signaling message updates more frequently. Therefore, there are more chances that senders transmit uninformative packets to the receivers. These uninformative transmissions waste the precious bandwidth resources and do not bring any goodness to this system. Our scheduling algorithm can effectively reduce these uninformative transmissions. The experimental results proved that our scalable content scheduling algorithm can reduce the uninformative transmission.

In the next experiment, we compare in Fig. 6.6 the average received video quality of the proposed algorithm versus the number of peers. As depicted in Figure 6.6, we compare the video quality of the two methods when the network size N is 20, 50, 100, 200 separately. The node upload bandwidth $\hat{U}$ is set to be $1.2S$. From the figure, we can see that the perceived video quality of the SGS and the MGS algorithm is almost the same at different networks. It proves that the SGS and the MGS algorithm are all scalable, and it is suitable for large-scale live streaming.

The performance of the video quality in PSNR versus the number of hops from the server is evaluated in this experiment. More specifically, this experiment demonstrates the scalability of the scheduling algorithm and the fairness among client nodes. The hop is defined as the hop distance from the client node $\mathcal{N}_j$ to the streaming server. As

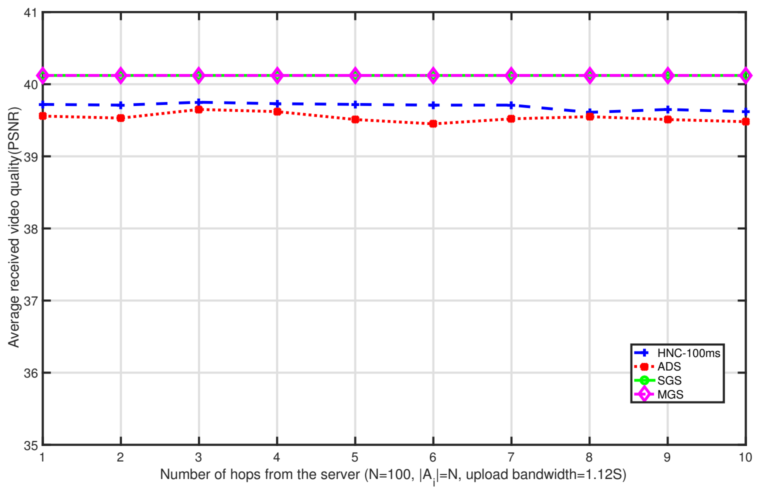


Figure 6.7: The impact of distance from server to the average received video quality in PSNR

depicted in Fig. 6.7, the average video quality remains almost the same as the number of hops increases. This shows that all algorithms are not very sensitive to the distance from the server. That is because the HNC, ADS, SGS, and MGS algorithms are built in the mesh network, where the neighbor nodes of each client node are randomly chosen. Therefore, the distance from the streaming server does not have an obvious influence on the delivery ratio.

In Fig. 6.8 the performance of the delivery ratio is analyzed in a lossy network to evaluate its resistance to network loss. We compare the achieved average received video quality among the four schemes. Through this experiment, we can see that how different algorithms perform against the network loss. ADS and the HNC use the signaling buffer-map to control the content delivery in the lossy network, and the MGS and the SGS algorithms use the scheduling compensation mode (SCM) proposed in [113] to control the content delivery in the lossy network. In all, the scheduling compensation mode is not robust enough when the network loss increases. The SCM use the average loss rate to perform scheduling compensation for each generation. However, the scheduling compensation may be inaccurate when the loss rate increase, which leads to redundant transmission, and unsuccessful transmission. Therefore, the HNC and ADS algorithms

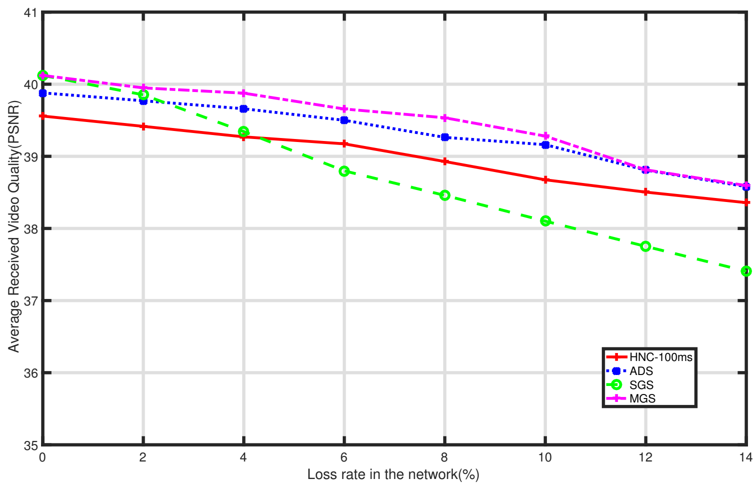


Figure 6.8: Performance comparison of the average delivery ratio as a function of network loss rates when $N = 100$ and upload bandwidth $\bar{U} = 1.2S$

are more robust than the MGS and SGS when the network loss rate is significant.

6.7 Summary

A key aspect of NC-driven live peer-to-peer streaming is the packet scheduling algorithm, which organizes the bandwidth allocation of sender nodes to improve the overall Quality-of-Service. We find that many current packet scheduling algorithms have a critical problem of inefficient bandwidth utilization. Some existing optimizations solve this problem by optimizing the bandwidth allocation for each single generation. However, their works still did not fully make use of the bandwidth resources. To better solve the problem of inefficient bandwidth utilization, a novel multiple generation scheduling(MGS) algorithm is proposed to find the optimal class subscription policy so that the upload bandwidth is reasonable utilized and the overall perceived video quality at the receiver nodes are maximized. The algorithm is formulated as a video quality maximization function under the available bandwidth resources constraints and solved through a dynamic programming algorithm. A single generation scheduling (SGS) is also proposed to provide a more simple solution to the MGS algorithm. Moreover, based on the two packet scheduling algorithms, a distributed packet scheduling(DPS) algorithm for scalable video transmission is proposed to coordinate senders' transmission of the subscribed classes. Our experimental results confirm that MGS algorithm outperforms the SGS algorithm and other scalable video scheduling algorithms, and achieves better video quality, better delivery ratio, and a very low redundant packet ratio when the peer upload bandwidth varies. Furthermore, the robustness to lossy networks and the network scalability is also proved by different experiments.

Chapter 7

Conclusions

7.1 Conclusions

This thesis focuses on the packet scheduling algorithm for video multicast in NC-based P2P streaming network.

An optimized rate allocation algorithm was proposed for scalable video streaming in the conventional NC-based mesh-push multicast overlay. This algorithm determines how senders allocate their upload bandwidth to different classes in the scalable video so that the distortion and redundancy ratio can be minimized. Compared with a random push with hierarchical network coding, the optimized rate allocation algorithm proposed in Chapter 4 has a better performance in video quality, delivery ratio, bandwidth efficiency, and resistant to network loss compared with the random-push scheme.

It is observed that the bandwidth inefficiencies in the conventional NC-based P2P streaming network are caused by uninformative and unrecoverable NC-based packet transmissions. Therefore, in Chapter 5, we improve the bandwidth efficiency, and thereby improving the OoS. First, a scalable compensation model and an adaptive push algorithm are proposed to reduce the unrecoverable transmission caused by network loss and insufficient bandwidth resources. Then a centralized packet scheduling algorithm is proposed to reduce the uninformative transmission caused by the asynchronized communication among sender nodes. Subsequently, we further propose a distributed packet scheduling algorithm, which adds a critical scalability property to the packet scheduling model. Compared with several state-of-art scheduling algorithms, the proposed algorithms bring better system performance for non-scalable video transmission.

In Chapter 6, we further present two video classes subscription algorithms to determine the quality classes that the receiver node can subscribe to so that the overall perceived video quality can be maximized. First, the video class subscription algorithm is formulated as a multiple generation scheduling algorithm, which coordinately schedule packet transmission in a given period such that the overall video quality can be maximized. The multiple generation scheduling problem is solved using two methods. One transfers the multiple generation scheduling to a single generation optimization algorithm to provide a faster and easier solution. The other solves the multiple generation scheduling directly using a dynamic programming algorithm. Experimental results confirm that the multiple generation scheduling can bring better QoE in the scalable video transmission system.

For all algorithms proposed in this thesis, great attention is given to the balance between QoE and bandwidth efficiency in the packet scheduling algorithm design. The proposed algorithms provide useful guidelines and potential solutions for packet scheduling in future video streaming network.

7.2 Future Work

7.2.1 Rate Allocation in An Unstable Streaming Network

In all proposed scheduling algorithms, upload bandwidth of peers is used as the main criterion to schedule packet transmission. However, the drawback of the proposed algorithms is that the scheduling result is based on the given upload bandwidth. When the upload bandwidth varies or peers frequently churn, the performance of our scheduling algorithm approaches to the performance of random push. Therefore, for the unstable P2P network, it is necessary to build a packet scheduling algorithm based on the historical bandwidth information or the historical buffer-map information. By observing the historical change in bandwidth and peer churn, we can predict the future available bandwidth and loss rate. In this way, the packet scheduling algorithm in a dynamic network environment can be more accurate.

Furthermore, when the peer churn rate and the peer bandwidth change frequently, it will be helpful to add several helper nodes in the network to replace the request model proposed in this system. In the current request model, if a generation is undecodable in the urgent region, the receiver node will send a request message to all its neighbors to request packets. However, in that case, many uninformative packets will be generated due to the buffer-map updating delay. Therefore, it is promising to add some helper node to the network. A helper node should have higher upload bandwidth, less churn rate, and less loss rate. A helper node will only transmit packets to a receiver when the receiver request packet specifically. In this way, the receiver can only request the needed number of packets from helpers, and avoid uninformative transmission.

7.2.2 Rate Allocation Among Senders

In Chapter 5, we propose to achieve the packet allocation based on the upload bandwidth of each sender. In this algorithm, we consider that the cost to transmit one packet from each sender to the receiver is the same. However, the RTT (Round Trip Time) or node hops from each sender node to the receiver node is different. When we want to minimize the delay or consumed network bandwidth, the proposed CPS and DPS algorithm is not useful. Recently, matching theory is becoming popular in the resource allocation algorithm in the wireless network [117]. Packet scheduling using matching theory will enable senders to achieve a better rate allocation.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update 2014-2019 white paper," Feb. 2015.

[2] J. Rosenberg, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications," IETF RFC 3550, Tech. Rep., 2003.

[3] M. Watson, "Http adaptive streaming in practice," in *Proceedings of the ACM Multimedia Systems Conference (MMSys)–Keynote, San Jose, CA, USA, Febrary*, 2011.

[4] J. Gaedtke, "Keynote speech: Optimizing qoe in large-scale video networks," Dec.2002.

[5] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 2, pp. 85–110, 1990.

[6] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *Network, IEEE*, vol. 14, no. 1, pp. 78–88, 2000.

[7] "Youtube," http://www.youtube.com.

[8] A. Cockcroft, "Netflix in the cloud," 2011.

[9] M. Zhang, L. Sun, X. Xi, and S. Yang, "igridmedia: providing delay-guaranteed peer-to-peer live streaming service on internet," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–5, 2008.

[10] "Streamroot: Smart p2p video streaming. demo available," http://www.streamroot.io/.

[11] (http://www.p2p-next.org/) P2p-next fp7.

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

[13] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," 2003.

[14] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, "Network coding meets multimedia: A review," *Multimedia, IEEE Transactions on*, vol. 15, no. 5, pp. 1195–1212, 2013.

[15] Z. Liu, C. Wu, B. Li, and S. Zhao, "Uusee: large-scale operational on-demand streaming with random network coding," in *INFOCOM, 2010 Proceedings IEEE.* IEEE, 2010, pp. 1–9.

[16] L. Yu, L. Gao, J. Zhao, and X. Wang, "Sonicvod: A vcr-supported p2p-vod system with network coding," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 2, pp. 576–582, 2009.

[17] C. Chekuri, C. Fragouli, and E. Soljanin, "On average throughput and alphabet size in network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2410–2424, 2006.

[18] D. J. MacKay, "Fountain codes," in *Communications, IEE Proceedings-*, vol. 152, no. 6. IET, 2005, pp. 1062–1068.

[19] T. Matsuda, T. Noguchi, and T. Takine, "Survey of network coding and its applications," *IEICE transactions on communications*, vol. 94, no. 3, pp. 698–717, 2011.

[20] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, 2003.

[21] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.

[22] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *Information Theory, IEEE Transactions on*, vol. 51, no. 8, pp. 2745–2759, 2005.

[23] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics, 2004, pp.

142–150.

[24] Q. Li, S. H. Ting, and C. K. Ho, “Nonlinear network code for high throughput broadcasting with retransmissions,” in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on.* IEEE, 2009, pp. 2853–2857.

[25] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, “A random linear network coding approach to multicast,” *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

[26] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” *IEEE International Symposium on Proceeding*, 2003.

[27] A. Fiandrotti, V. Bioglio, M. Grangetto, R. Gaeta, and E. Magli, “Band codes for energy-efficient network coding with application to p2p mobile streaming,” *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 521–532, 2014.

[28] T. Stockhammer, “Dynamic adaptive streaming over http: standards and design principles,” in *Proceedings of the second annual ACM conference on Multimedia systems.* ACM, 2011, pp. 133–144.

[29] B. Li, Z. Wang, J. Liu, and W. Zhu, “Two decades of internet video streaming: A retrospective view,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 9, no. 1s, p. 33, 2013.

[30] H. Schulzrinne, “Real time streaming protocol (rtsp),” 1998.

[31] R. T. S. Protocol, “Rfc 2326,” *IETF, April*, 1998.

[32] S. Mittra, “Iolus: A framework for scalable secure multicasting,” in *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4. ACM, 1997, pp. 277–288.

[33] T. Kim and M. H. Ammar, “A comparison of layering and stream replication video multicast schemes,” in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video.* ACM, 2001, pp. 63–72.

[34] S. McCanne, V. Jacobson, and M. Vetterli, “Receiver-driven layered multicast,” *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4, pp. 117–130, 1996.

[35] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," in *Multimedia Computing and Systems' 97. Proceedings., IEEE International Conference on.* IEEE, 1997, pp. 110–117.

[36] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1456–1471, 2002.

[37] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," *Technical Report*, 2001.

[38] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 58–74, 2007.

[39] F. Wang, Y. Xiong, and J. Liu, "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on.* IEEE, 2007, pp. 49–49.

[40] A. Wierzbicki, R. Szczepaniak, and M. Buszka, "Application layer multicast for efficient peer-to-peer applications," in *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on.* IEEE, 2003, pp. 126–130.

[41] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[42] "Akamai," https://www.akamai.com/.

[43] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, no. 4, pp. 383–399, 2006.

[44] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia.* ACM, 2009, pp. 25–34.

[45] "Livesky," https://theskylive.com/.

[46] N. Ramzan, H. Park, and E. Izquierdo, "Video streaming over p2p networks: Chal-

lenges and opportunities," *Signal Processing: Image Communication*, vol. 27, no. 5, pp. 401–411, 2012.

[47] N. Ramzan, E. Quacchio, T. Zgaljic, S. Asioli, L. Celetto, E. Izquierdo, and F. Rovati, "Peer-to-peer streaming of scalable video in future internet applications," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 128–135, 2011.

[48] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *p2p*. IEEE, 2001, p. 0101.

[49] Z. Shen, J. Luo, R. Zimmermann, and A. V. Vasilakos, "Peer-to-peer media streaming: Insights and new developments," *Proceedings of the IEEE*, vol. 99, no. 12, pp. 2089–2109, 2011.

[50] X. Zhang and H. Hassanein, "A survey of peer-to-peer live video streaming schemes–an algorithmic perspective," *Computer Networks*, vol. 56, no. 15, pp. 3548–3579, 2012.

[51] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video.* ACM, 2002, pp. 177–186.

[52] D. E. Pendarakis, S. Shi, D. C. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure." in *USITS*, vol. 1, 2001, pp. 5–5.

[53] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video.* ACM, 2002, pp. 177–186.

[54] D. E. Pendarakis, S. Shi, D. C. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure." vol. 1, pp. 5–5, 2001.

[55] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek *et al.*, "Overcast: reliable multicasting with on overlay network," in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4.* USENIX Association, 2000, pp. 14–14.

[56] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, *Scalable application layer*

multicast. ACM, 2002, vol. 32, no. 4.

[57] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, pp. 1283–1292, 2003.

[58] R. Tian, Y. Xiong, Q. Zhang, B. Li, B. Y. Zhao, and X. Li, "Hybrid overlay structure based on random walks," in *Peer-to-Peer Systems IV.* Springer, 2005, pp. 152–162.

[59] M. Kobayashi, H. Nakayama, N. Ansari, and N. Kato, "Robust and efficient stream delivery for application layer multicasting in heterogeneous networks," *Multimedia, IEEE Transactions on*, vol. 11, no. 1, pp. 166–176, 2009.

[60] A. Magnetto, R. Gaeta, M. Grangetto, and M. Sereno, "Turinstream: A totally push, robust, and efficient p2p video streaming architecture," *Multimedia, IEEE Transactions on*, vol. 12, no. 8, pp. 901–914, 2010.

[61] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1366–1375.

[62] P. Francis, "Yoid: Extending the internet multicast architecture," 2000.

[63] X. Jin, K.-L. Cheng, and S. G. Chan, "Island multicast: combining ip multicast with overlay data distribution," *Multimedia, IEEE Transactions on*, vol. 11, no. 5, pp. 1024–1036, 2009.

[64] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Networked Group Communication.* Springer, 2001, pp. 14–29.

[65] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video.* ACM, 2001, pp. 11–20.

[66] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh,

"Splitstream: high-bandwidth multicast in cooperative environments," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 298–313.

[67] R. Zhang and Y. C. Hu, "Borg: a hybrid protocol for scalable application-level multicast in peer-to-peer networks," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video.* ACM, 2003, pp. 172–179.

[68] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 1, pp. 41–53, 2004.

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

[70] K.-H. Chan, S.-H. Chan, and A. C. Begen, "Spanc: Optimizing scheduling delay for peer-to-peer live streaming," *Multimedia, IEEE Transactions on*, vol. 12, no. 7, pp. 743–753, 2010.

[71] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Substream trading: towards an open p2p live streaming system," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on.* IEEE, 2008, pp. 94–103.

[72] Y. Chawathe, "Scattercast: an adaptable broadcast distribution framework," *Multimedia Systems*, vol. 9, no. 1, pp. 104–118, 2003.

[73] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 1521–1531.

[74] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on.* IEEE, 2006, pp. 2–11.

[75] D. Ren, Y.-T. H. Li, and S. G. Chan, "Fast-mesh: a low-delay high-bandwidth

mesh for peer-to-peer live streaming," *Multimedia, IEEE Transactions on*, vol. 11, no. 8, pp. 1446–1456, 2009.

[76] X. Zhang and H. Hassanein, "Treeclimber: A network-driven push-pull hybrid scheme for peer-to-peer video live streaming," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 368–371.

[77] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols," *Computers, IEEE Transactions on*, vol. 52, no. 2, pp. 139–149, 2003.

[78] M. Wang and B. Li, "R2: Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655–1666, 2007.

[79] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM computer communication review*, vol. 36, no. 4. ACM, 2006, pp. 243–254.

[80] D. Katabi, S. Katti, W. Hu, H. Rahul, and M. Medard, "On practical network coding for wireless environments," in *Communications, 2006 International Zurich Seminar on*. IEEE, 2006, pp. 84–85.

[81] H. Wang, J. Liang, and C.-C. J. Kuo, "Overview of robust video streaming with network coding," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 1, pp. 36–50, 2010.

[82] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, "Lion: Layered overlay multicast with network coding," *Multimedia, IEEE Transactions on*, vol. 8, no. 5, pp. 1021–1032, 2006.

[83] X. C., X. Y., Z. C., W. R., and W. Q., "On network coding based multirate video streaming in directed networks," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*. IEEE, 2007, pp. 332–339.

[84] M. Shao, S. Dumitrescu, and X. Wu, "Layered multicast with inter-layer network coding for multimedia streaming," *Multimedia, IEEE Transactions on*, vol. 13, no. 2, pp. 353–365, 2011.

[85] J. Zou, H. Xiong, C. Li, L. Song, Z. He, and T. Chen, "Prioritized flow optimization with multi-path and network coding based routing for scalable multirate multicasting," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 3, pp. 259–273, 2011.

[86] M. Wang and B. Li, "Network coding in live peer-to-peer streaming," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1554–1567, 2007.

[87] C. Feng and B. Li, "On large-scale peer-to-peer streaming systems with network coding," in *Proceedings of the 16th ACM international conference on Multimedia.* ACM, 2008, pp. 269–278.

[88] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[89] G. Yue, N. Wei, J. Liu, X. Xiong, and L. Xie, "Survey on scheduling technologies of p2p media streaming," *Journal of Networks*, vol. 6, no. 8, pp. 1129–1136, 2011.

[90] N. Ramzan, S. Wan, and E. Izquierdo, "Joint source-channel coding for wavelet-based scalable video transmission using an adaptive turbo code," *Journal on Image and Video Processing*, vol. 2007, no. 1, pp. 16–16, 2007.

[91] S. Wan and E. Izquierdo, "Rate-distortion optimized motion-compensated prediction for packet loss resilient video coding," *Image Processing, IEEE Transactions on*, vol. 16, no. 5, pp. 1327–1338, 2007.

[92] N. Šprljan, M. Mrak, G. Abhayaratne, and E. Izquierdo, "A scalable coding framework for efficient video adaptation," in *6th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS 2005*, 2005.

[93] E. Peixoto and E. Izquierdo, "A complexity-scalable transcoder from h. 264/avc to the new hevc codec," in *Image Processing (ICIP), 2012 19th IEEE International Conference on.* IEEE, 2012, pp. 737–740.

[94] T. Zgaljic, N. Sprljan, and E. Izquierdo, "Bit-stream allocation methods for scalable video coding supporting wireless communications," *Signal Processing: Image Communication*, vol. 22, no. 3, pp. 298–316, 2007.

[95] K. Nguyen, T. Nguyen, and S.-C. Cheung, "Video streaming with network coding,"

Journal of Signal Processing Systems, vol. 59, no. 3, pp. 319–333, 2010.

[96] A. M. Sheikh, A. Fiandrotti, and E. Magli, "Distributed scheduling for low-delay and loss-resilient media streaming with network coding," *Multimedia, IEEE Transactions on*, vol. 16, no. 8, pp. 2294–2306, 2014.

[97] L. Cui, Y. Jiang, and J. Wu, "Optimizing push scheduling algorithm based on network coding for mesh peer-to-peer live streaming," in *Communications (ICC), 2012 IEEE International Conference on.* IEEE, 2012, pp. 2075–2080.

[98] M. Sanna and E. Izquierdo, "Proactive prioritized mixing of scalable video packets in push-based network coding overlays," in *Packet Video Workshop (PV), 2013 20th International.* IEEE, 2013, pp. 1–7.

[99] C. Jing, X. Zhang, F. Tang, S. Fowler, H. Cui, and X. Dong, "R2nc: Redundant and random network coding for h. 264/svc transmission," in *Network-Based Information Systems (NBiS), 2011 14th International Conference on.* IEEE, 2011, pp. 634–639.

[100] N. Thomos, J. Chakareski, and P. Frossard, "Prioritized distributed video delivery with randomized network coding," *Multimedia, IEEE Transactions on*, vol. 13, no. 4, pp. 776–787, 2011.

[101] E. Bourtsoulatze, N. Thomos, and P. Frossard, "Distributed rate allocation in inter-session network coding," *Multimedia, IEEE Transactions on*, vol. 16, no. 6, pp. 1752–1765, 2014.

[102] ——, "Decoding delay minimization in inter-session network coding," *CommunicationMultimedia, IEEE Transactions on*, vol. 62, no. 6, 2014.

[103] N. Thomos, E. Kurdoglu, P. Frossard, and M. Van der Schaar, "Adaptive prioritized random linear coding and scheduling for layered data delivery from multiple servers," *Multimedia, IEEE Transactions on*, vol. 17, no. 6, pp. 893–906, 2015.

[104] A. B. Carlson, P. Crilly, and J. Rutledge, "Communication systems: an introduction to signals and noise in electrical communication," *Guía Académica*, p. 129, 1986.

[105] J. Wang and J. Han, "Zeta: A novel network coding based p2p downloading protocol," http://code.google.com/p/zetasim/.

[106] “Gnutellasim,” sourceforge.net/projects/gnutellasim2/, [Online, accessed 22-Sep-2014].

[107] B. M. Waxman, “Routing of multipoint connections,” *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 9, pp. 1617–1622, 1988.

[108] “Waxman random network topology generator.”

[109] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, “Network coding meets multimedia: A review,” *arXiv preprint arXiv:1211.4206*, 2012.

[110] K. Nguyen, T. Nguyen, and S.-c. Cheung, “Peer-to-peer streaming with hierarchical network coding,” in *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE, 2007, pp. 396–399.

[111] Y. Huang, T. Z. Fu, D.-M. Chiu, J. Lui, and C. Huang, “Challenges, design and analysis of a large-scale p2p-vod system,” in *ACM SIGCOMM computer communication review*, vol. 38, no. 4. ACM, 2008, pp. 375–388.

[112] A. T. Nguyen, B. Li, and F. Eliassen, “Chameleon: Adaptive peer-to-peer streaming with network coding,” *Proceedings of IEEE INFOCOM*, pp. 1–9, 2010.

[113] S. Huang, E. Izquierdo, and P. Hao, “Bandwidth-efficient packet scheduling for live streaming with network coding,” *Multimedia, IEEE Transactions on*, no. 4, pp. 752–763, 2016.

[114] D. Pisinger, “A minimal algorithm for the multiple-choice knapsack problem,” *European Journal of Operational Research*, vol. 83, no. 2, pp. 394–410, 1995.

[115] H. Kellerer, U. Pferschy, and D. Pisinger, *Introduction to NP-Completeness of knapsack problems*. Springer, 2004.

[116] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[117] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, “Matching theory for future wireless networks: Fundamentals and applications,” *Communications Magazine, IEEE*, vol. 53, no. 5, pp. 52–59, 2015.

Appendix A

Appendix

A.1 Proof of Equivalence of Optimization Function

proof: The equivalence of the proposed CPS and the DPS algorithm is proved. This proof includes 2 parts. One is the proof of the equivalence of the two optimization objectives. The other is the proof of the equivalence of constraints.

Part 1: We will prove the optimization objective of the DPS is the approximation of the objective of the CPS algorithm.

$$\begin{aligned} Q(\boldsymbol{H}) &= \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}} \\ &= \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} (\tilde{U}_{ij} - 2H_{ijg} + \frac{H_{ijg}^2}{\tilde{U}_{ij}}) \end{aligned} \tag{A.1}$$

As assumed in the constraint of the DPS optimization in Eq.5.4, we have that

$$\sum_{i=1}^{N} H_{ijg} \geq \hat{U} \ \forall j, g \tag{A.2}$$

Then we have that

$$Q(\boldsymbol{H}) \leq \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} \tilde{U}_{ij} + \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} \frac{H_{ij}^2}{\tilde{U}_{ij}} - 2 \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \hat{U} \tag{A.3}$$

In the expanded function, $\tilde{U}_{ij}$ and $\hat{P}_g$ are known positive constant. The optimization objective of DPS is the p-Norm of H_{ijg} (p=2). The optimization objective of Eq.5.3 is the 1-norm of H_{ijg}. The property of norm is shown in Eq. A.4,

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{2}\|x\|_2 \tag{A.4}$$

As the vector space is a real finite-dimensional one, all norms are equivalent. So we can get the conclusion that the objective of DPS is the approximation of the objective of CPS. The two optimization function have an approximate solution.

Part 2: We will prove that the solution to the DPS optimization in Eq.5.5 satisfies the constraint (a) of CPS alggortihm in Eq.5.3. Consider

$$H_{ijg} = \frac{\tilde{U}_{ij}}{\sum\limits_{k=1}^{|A_j|} \tilde{U}_{kj}} \hat{P}_g \quad \forall i, j, g \tag{A.5}$$

As the scheduling algorithm is independent to g, we have that

$$\sum_{g=\Gamma_s}^{m} H_{ijg} = \sum_{g=\Gamma_s}^{m} \frac{\tilde{U}_{ij}}{\sum\limits_{k=1}^{|A_j|} \tilde{U}_{kj}} \hat{P}_g \quad \forall i, j, \Gamma_s \leq m \leq \Gamma_e \tag{A.6}$$

As scheduled in the handshake procedure of the DPS algorithm, we have that

$$\sum_{k=1}^{|A_j|} \tilde{U}_{kj} \geq \hat{U} \quad \forall j \tag{A.7}$$

so that

$$\sum_{g=\Gamma_s}^{m} H_{ijg} \leq \sum_{g=\Gamma_s}^{m} \frac{\tilde{U}_{ij}}{\hat{U}} \hat{P}_g \quad \forall i, j, \Gamma_s \leq m \leq \Gamma_e \tag{A.8}$$

In the adaptive packet scheduling algorithm in Section5.2.4, we have

$$\sum_{g=\Gamma_s}^{m} \hat{U} \geq \sum_{g=\Gamma_s}^{m} \hat{P}_g \quad \forall \Gamma_s \leq m \leq \Gamma_e \tag{A.9}$$

So that

$$\begin{aligned} &\sum_{g=\Gamma_s}^{m} H_{ijg} \leq \sum_{g=\Gamma_s}^{m} \tilde{U}_{ij} \quad \forall i, j, \Gamma_s \leq m \leq \Gamma_e \\ &\sum_{g=\Gamma_s}^{m} \sum_{j=2}^{N} H_{ijg} \leq \sum_{g=\Gamma_s}^{m} \sum_{j=2}^{N} \tilde{U}_{ij} \quad \forall i, \Gamma_s \leq m \leq \Gamma_e \end{aligned} \tag{A.10}$$

As supposed in the handshake procedure of DPS algorithm, we have that

$$\sum_{j=2}^{N} \tilde{U}_{ij} \leq \tilde{U}_i \quad \forall i \tag{A.11}$$

Hence,

$$\sum_{g=\Gamma_s}^{m} \sum_{j=2}^{N} H_{ijg} \leq \sum_{g=\Gamma_s}^{m} \tilde{U}_i \quad \forall i \ , \Gamma_s \leq m \leq \Gamma_e \qquad \square \tag{A.12}$$

Appendix B

Appendix

B.1 Author's Publication

Journal papers

1. **S. Huang**, E. Izquierdo and P. Hao, "Bandwidth-efficient Packet Scheduling for Live Streaming with Network Coding" *IEEE Transaction on Multimedia*, vol. 18, no. 4, pp. 752-763, 2016
2. **S. Huang**, E. Izquierdo and P. Hao, "Adaptive packet scheduling for scalable video streaming with network coding" *Journal of Visual Image Communication and Representation*, Nov, 2016

Conference papers

1. **S. Huang**, E. Izquierdo and P. Hao, "scalable video multicast system on P2P network with hierarchical network coding," *5th Latin American Conference on Networked and Electronic Media*, 2013.

2. **S. Huang**, M. Sanna, E. Izquierdo and P. Hao, "optimized scalable video transmission over P2P network with hierarchical network coding," *International Conference on Image Processing(ICIP)*, pp. 3993-3997, 2014, Paris, France

3. **S. Huang**, E. Izquierdo and P. Hao, "Optimized packet scheduling for live streaming on peer-to-peer network with network coding," *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 1515-1520, 2015

4. **S. Huang**, E. Izquierdo and P. Hao, "Distributed bandwidth-efficient packet scheduling for live streaming with network coding, *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference. ACM*, pp. 1035-1038, 2015

5. **S. Huang**, E. Izquierdo and P. Hao, "A Push Scheduling Algorithm with Network Coding for Peer-to-Peer Live Streaming, *6th Latin American Conference on Networked and Electronic Media*, 2015.

6. **S. Huang**, E. Izquierdo and P. Hao,Adaptive Packet Scheduling for Scalable Video Streaming with Network Coding, *International Conference on Communications(ICC)*, May. 2016

7. **S. Huang**, E. Izquierdo and P. Hao,Multi-generation packet scheduling for live streaming with network coding, *The IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 2016