OPTIMIZED ALGORITHMS FOR MULTIMEDIA STREAMING

LI YONGFENG

(M. Eng.), TJU

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

To my parents and my wife

ACKNOWLEDGMENT

As a graduating doctoral candidate, I would like to take an opportunity of this time to express my gratitude to the people who have been helping, encouraging, supporting and accompanying me through these years.

First and foremost, I would like to thank my supervisor, Associate Professor Ong Kok Wee, Kenneth, for motivating and guiding my research. It has always been a privilege to work with him because he has a knack for giving proper guidance, yet enough freedom to explore. His insightful feedback often provides fresh perspective. I am deeply grateful for having learned from the best. Moreover, I am very thankful for his help in my job searching.

This dissertation, along with all other accomplishments of mine, is dedicated to my parents, who have supported me for all the past years and will still sustain me for the future years. I owe them much more than I could ever express in words.

My deepest thanks to my beautiful wife: huahan for her love and moral support. She shares with me all the happiness and difficulties. With her love, I am always brave to face any frustration and challenge. She possesses my heart and my future belongs to her. I am determined to bring her and my family a better life.

I also enjoy collaborating and living with many friends. Finally, I would like to thank the National University of Singapore for funding my research.

TABLE OF CONTENT

SUMMARY

Since the introduction of video streaming a decade ago, it has been experiencing a dramatic growth and becoming an important multimedia communication method over Internet. Video streaming applications require real-time data delivery in order to provide continuous playback at clients with good perceptual quality. Nevertheless, lack of system resources and quality of service (QoS) support from Internet pose many issues that need to be resolved, such as how to deliver scalable coded video over networks with varying bandwidth, how to cater to heterogeneous requests simultaneously, and how to cache video efficiently. This research includes studies of different components of a video streaming system, and proposes several optimization models and algorithms to improve system performance in the presence of mutative environment and constrained resources.

In the first part of this dissertation, we present a compensation method for the delivery of scalable coded video. Our target is to improve the video quality in the presence of bandwidth fluctuation. In our approach, the discarded video data due to bandwidth degradation has the chance to be compensated to clients if both real-time and resource constraints can be satisfied.

In the second part, we utilize the hierarchical bit stream structure of scalable coded video and propose an optimized video adaptation scheme. By exploring the scene complexity and scene priority, we minimize the quality distortion and quality fluctuation of the adapted video stream simultaneously. In our approach, the adaptation is carried out hierarchically on both video scene level and video frame level with low computational complexity.

In the last part, we present a complete solution for proxy to cache and deliver scalable coded video in streaming system. The proposed caching scheme allows different video layers to be cached with various time spans based on user access behaviors, video characteristics, and transmission cost rates. A proxy stores video segments and coordinates video delivery, such that overall bandwidth cost and user access latency are minimized simultaneously.

The dissertation validates the proposed models, algorithms, schemes and explores their characteristics through extensive simulations.

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ACM:	adaptive compensation method
BL:	base layer
CBR:	constant bit rate
CDN:	content distribution networks
CM:	compensation model
EL:	enhancement layer
FDDC:	frame-dropping with drift compensation
FGS:	fine granular scalability
FGST:	temporal FGS
FLM:	frame level model
OAS:	optimized adaptation scheme
OCS:	optimized caching strategy
P2P:	peer-to-peer
PSNR:	peak signal-to-noise ratio
QoS:	quality of service
R-D:	rate-distortion
RQ:	re-quantization
RTCP:	real-time control protocol
RTP:	real-time transport protocol
RTSP:	real-time streaming protocol

SFD:	simple frame-dropping
SLM:	scene level model
SNR:	signal-to-noise ratio
SNR FGS:	SNR fine granular scalability
SP:	service providers
STS:	scalable transmission scheme
SVC:	scalable video coding
TFRC:	TCP-friendly rate control
VOD:	video-on-demand
WAN:	wide area network

Chapter 1: INTRODUCTION

The goal of this chapter is to provide a summary of the essential motivations and contributions of the dissertation. The dissertation outline is given at the end of this chapter as well.

1.1 Motivation

As best-effort networks (e.g. Internet) grow widespread and multimedia becomes popular, the delivery of video over Internet attracts more interest, from academic research to industrial development. Currently network-based video applications are heavily demanded in the Internet market, such as distance learning, video conferencing, Internet TV broadcasting, and video-on-demand (VOD). In these applications, the stored video or live encoded video are transmitted across networks upon clients' requests. There are two modes for clients to receive the encoded video over Internet: downloading mode and streaming mode. In downloading mode, a client requests a specific video and downloads the entire video data before it starts the video playback. Since client plays the video from its local storage, network condition does not influence the playback quality of the video. Nevertheless, client usually suffers a long and unacceptable waiting time, as well as a large consumption of storage space. In streaming mode, a client initiates the video playback after a certain part of the video data is received, and at the same time the client is constantly receiving the missing part for future display. Since streaming mode can overcome the aforementioned disadvantages of downloading mode, most of the Internet video applications employ the streaming mode in spite of the possible quality degradation incurred due to the varying network condition. Many researches are focusing on improving the client experienced quality in streaming mode as well.

Since existing well-developed Internet is best-effort in nature and lacks support for quality of service (QoS) guarantee, it poses many challenging issues for real-time video streaming, especially when many concurrent participants compete for the precious Internet resources.

- Bandwidth: Bandwidth is one of the most critical resources within Internet. Digital video streaming usually consumes a lot of bandwidth due to the large size of digital video and the real-time characteristic of streaming. Several concurrent streaming sessions can impose great pressure on the common network link and cause congestion collapse. This will degrade the streaming throughput of the whole streaming system even if the system has high bandwidth connectivity for both source server and clients. Moreover, the bandwidth fluctuation during streaming session may further deteriorate the video quality.

- Delay: Delay is an inherited issue in packet network, e.g. Internet. Data need time or delay to pass from source server to intermediate routers/switches, and finally arrive at clients. Once a client plays the video, successive video data must arrive in time. Otherwise, playback jitter will occur. Link congestion can worsen the problem by introducing excessive and unstable delay, and further degrade the user-perceived quality.
- Loss: Packet losses are unavoidable in Internet. Video packets may be discarded at any node along the path from source to destination, such as proxy, router, and switch. This discard can happen because of link congestion or excessive delay. The discarded video packets are usually not recovered in real-time video streaming due to the timeliness of video data. When packet loss ratio is high during streaming, the video presentation at client will become impossible.

Besides of the aforementioned network-related issues, a real-time video streaming application faces other challenges coming from end-systems:

- Heterogeneity: Internet incorporates a diverse of clients with distinct resource characteristics, such as bandwidth connectivity, decoding capacity, buffer size, and monitor resolution. These clients may use the same video application in Internet at the same time and request with different requirements. Video services, therefore, need to be resource scalable to accommodate and incorporate a wide range of requests simultaneously.

- Resource depletion: All the nodes in the streaming system may experience transient or continuous resource depletion. The depletion usually happens at system bottleneck, such as source server, which will degrade system performance seriously. Video applications need to focus on resolving these bottlenecks with high priority.

As outlined above, these issues constrain the large scale deployment of video streaming services in Internet. They need to be resolved with great care considering of the complex and sometimes unpredictable streaming environment. Existing researches on resolving them can be generally summarized into the following two categories:

- Network and system architecture: Adequate support from network and effective system architecture are critical for improving the quality of video streaming. Many novel system architectures, such as proxy-assisted and peer-to-peer systems, have been proposed and many content distribution networks are being deployed. Researches on the network and architectures usually need to be validated through large scale experiments and face great difficulties in practical deployment.
- System nodes: Besides of network and architecture advancement, excellent algorithms running over system nodes can improve the quality of video streaming to a large extent. These nodes include end-systems such as source server and clients, as well as intermediate network nodes such as proxies and

routers. For example, a source server may cut down the streaming rate to alleviate network congestion, or adapt the video to meet client and connection requirements. Clients may estimate bandwidth, provide prompt feedbacks, and coordinate with each other as well. Since server is normally constrained to doing only light processing on a per client basis, network nodes or clients can employ similar algorithms to relieve server's workload and hence improve the system performance.

Additional opportunities to improve streaming quality may come from the advance of video codec. In video applications, videos need to be compressed. Although many International standards have been established to target for different video applications, MPEG/ITU-T committee and many researches are constantly developing various video compression technologies for applications related to video streaming. One of the most important technologies is scalable video coding (SVC). SVC exchanges its coding efficiency to provide more flexibility for network-based video applications. With such flexibility, existing algorithms can be improved and novel algorithms can be proposed.

The main objective of the dissertation is to investigate various advanced algorithms based on well-established streaming architecture, so that the performance of video streaming system can be improved to a great extent. In this dissertation, we concentrate on the system nodes based approaches since they can be readily implemented without waiting for the large scale modification of network or system architecture. Our approaches are standards–conforming and make use of the flexibility provided by SVC. In particular, we focus on the techniques related to the streaming proxy, which is the key point for large scale video streaming system. To be more specific, we investigate the characteristics of scalable coded video and utilize the computational capacity and storage resource on streaming proxy to resolve the challenging issues, such as network fluctuation, heterogeneous requests, resource depletion, and so on, in large scale video streaming system, such that system performance can be improved in terms of streaming quality and streaming cost. By employing proposed methods and algorithms on the existing proxy servers, we can significantly reduce system bottleneck, increase system capacity, and improve system quality of service (QoS) with the same resources. It is also worth noting that most of the techniques to be investigated apply to the source server directly and effectively improve the performance of video streaming system.

1.2 Thesis Contribution

We present three topics related to video streaming, i.e. transmission, adaptation, caching, and makes the following four contributions.

- Firstly, we study the transmission problem for scalable coded video streaming over best-effort networks, and propose an adaptive compensation method (ACM) to reduce the negative effects of bandwidth fluctuation on the received video quality at clients. The proposed ACM is a simple and easy-to-implement solution that exploits the utilization ratio of client buffer and the characteristics of scalable coded video stream. This solution adaptively chooses appropriate video segments to transmit according to network conditions, and provides improved video quality to clients during bandwidth fluctuations.

- Secondly, we investigate the video adaptation problem for scalable coded video, and propose an optimized adaptation scheme for MPEG-4 FGS video streams. Proposed scheme explores the texture and motional complexity of video scenes and incorporate scene priority to guarantee user-interested scenes a better quality. We present two hierarchical adaptation algorithms on both video scene level and video frame level to minimize quality distortion and quality fluctuation simultaneously. Overall, an excellent compromise between Signal-to-Noise Ratio (SNR) and temporal resolutions for the adapted video is achieved with low computational complexity.
- Thirdly, we analyze the proxy caching problem for scalable coded video streaming and propose an optimized proxy caching scheme. This scheme tends to allocate different storage sizes to different videos according to their popularities and cache different layers with distinct lengths for each video. We choose to cache the videos on proxy in a manner that both bandwidth cost and user access latency are minimized with a certain proxy storage capacity.
- Finally, we present a real-time transmission scheme for scalable coded videos. This scheme is incorporated into proxy-assisted video streaming architecture

and works with the scalable proxy caching scheme seamlessly.

1.3 Outline

The remainder of the dissertation is organized as follows. Chapter 2 outlines the Internet standards for real-time video streaming and the development of SVC. Then a brief overview of end-to-end streaming systems and research issues is presented. Chapter 3 describes a novel compensation method for scalable coded video streaming. We begin with an analysis of the transmission problem due to bandwidth fluctuation. Then a new transmission method with optimized compensation is proposed. This method chooses to compensate proper video segments to clients at appropriate timings and we verify its effectiveness on video quality improvement through simulations. An efficient video adaptation scheme is presented in Chapter 4. We incorporate video scene complexity and priority into the adaptation models, and present efficient algorithms to obtain the optimized adaptation scheme for MPEG-4 FGS coded video stream. This scheme provides optimized user-perceived quality for the adapted video in terms of both quality distortion and quality fluctuation. Simulation results under various network conditions are presented. Chapter 5 proposes an optimized proxy caching strategy and corresponding transmission scheme for scalable coded video streaming. A multi-objective optimization model is formulated with closed-form expressions and then solved by heuristic algorithms to obtain the caching strategy. This strategy can reduce the bandwidth cost and user access latency at the same time. Simulation results demonstrate the superior characteristics of proposed caching strategy and transmission scheme. Finally, Chapter 6 completes the dissertation with concluding remarks and a discussion of major difficulties encountered and future research works.

Chapter 2: INTERNET VIDEO STREAMING

This chapter reviews the background of scalable video streaming over a packet network, e.g. Internet. More specifically, we review the existing network protocols and various scalable video coding (SVC) methods for real-time video streaming. In addition, we describe and compare two types of end-to-end video streaming systems which are widely deployed in Internet: proxy-assisted system where source server and clients are connected through dedicated proxies, and peer-to-peer (P2P) system where each participant provides the functionalities of both content provider and content consumer. Several research challenges within the proxy-assisted streaming architecture, such as adaptive transmission, video adaptation, and proxy caching are investigated briefly.

2.1 Protocols

Internet has been developed to connect heterogeneous networks and enable various services over these networks. In this section, we review some important Internet standard protocols for video streaming: Real-time Transport Protocol (RTP), Real-time Control Protocol (RTCP), and Real-time Streaming Protocol (RTSP). Moreover, TCP-Friendly rate control protocol (TFRC) is introduced to overcome the TCP/UDP disadvantages in real-time video streaming. RTP provides end-to-end data transport functions. RTCP is a companion protocol with RTP and provides QoS feedback to the participants of an RTP session. RTSP is a streaming session protocol to exchange control commands between the participants of a RTSP session. Overall, RTP/RTCP/RTSP protocol stack provides the essential functionalities for real-time video streaming.

2.1.1 RTP

RTP is a protocol which provides end-to-end delivery services that are appropriate for streaming video over best-effort networks with real-time characteristics [1][2][3][4]. These services include payload type identification, sequence numbering, timestamp, and delivery monitoring. The timestamp and sequence numbers included in RTP packets allow the receiver to reconstruct the sender's packet sequence and determine the proper location of a packet. Nevertheless, RTP itself neither addresses resource reservation nor guarantees QoS for real-time video streaming [5]. This means that RTP does not provide any congestion control mechanism for the delivery of video data. Hence, RTP usually works with other network protocols such as RTCP to provide congestion control functionalities. Typical real-time video streaming applications run RTP on top of UDP to make use of its multiplexing and checksum services. Additional application-specific protocols such as RTSP usually work with RTP in the form of application-specific profiles. Although RTP is primarily designed to meet the needs of multi-participant video conferences, it is not limited to that particular application. Various real-time video streaming applications such as interactive video streaming and video-on-demand, may find RTP applicable. For a detailed RTP header and payload format for the video coding standards H.263 and MPEG-4/H.264, we refer to [6][7][8][9].

2.1.2 RTCP

RTCP [1][3] is a companion protocol of RTP that video streaming applications generally use. RTCP and RTP packets are distinguished from each other through the distinct port numbers. Different from RTP packets, RTCP packets do not encapsulate any video data and only provide QoS feedback through the use of sender reports and receiver reports. These reports contain the statistics like packet loss rate, jitter, and packet inter-arrival time which are fully used by applications. Generally, RTCP keeps the total control packets to 5% of the total streaming session bandwidth. Among these control packets, 25% are allocated to the sender reports and 75% to the receiver reports. To prevent control packet starvation, at least 1 control packet is sent within 5 seconds at the sender or receiver.

2.1.3 RTSP

RTSP [10][11] is the de facto standard for video streaming and almost all the video players, such as Video LAN, Real Player, and Motorola set-top-box, support it. It is jointly developed by Netscape, Real-Networks, and Columbia University to tackle the problem that HTTP protocol has no session to guarantee long-lived service and no functionality to control and deliver video over Internet in real-time.

RTSP provides an extensible framework to enable controlled and efficient delivery of video streams over IP networks. It minimizes the overhead of video delivery and takes full advantage of Internet infrastructural improvements. RTSP has some overlap functionalities with HTTP, but differs fundamentally from HTTP because both RTSP server and RTSP client need to maintain state to continue to control a streaming session after streaming requests have been acknowledged.

A typical RTSP/RTP/RTCP protocol stack is shown in Figure 2-I. Session control commands are transmitted using RTSP/TCP, and video data are transported through well-established RTP/RTCP/UDP. In a RTSP session, server and client exchange the requests and response messages such as the commands of setting up a transport mechanism, starting the session, and closing the session. After RTSP session is setup, the server packs video data into RTP packets and generates reports from feedback control protocol RTCP. The resulting RTCP and RTP packets go down to the UDP/IP layer for transmission. Client receives these packets and passes them upwards through the protocol stack for processing.

RTP/RTCP/RTSP protocol stack is designed to be extended easily, so that congestion control algorithms and other network protocols such as TCP-Friendly rate control (TFRC) can be incorporated to provide advanced functionalities.

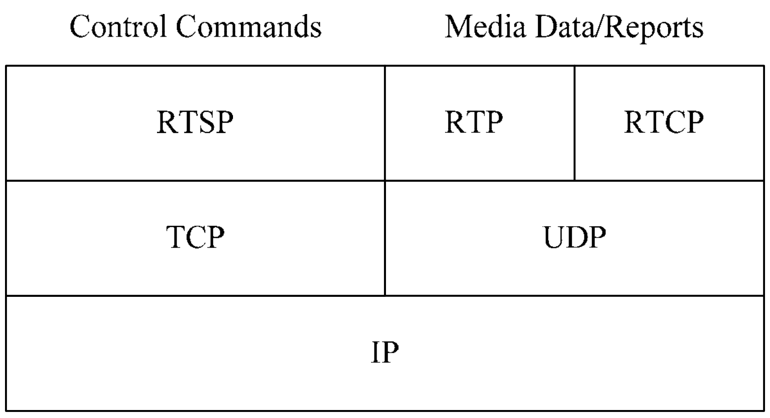


Figure 2-I: Typical RTSP/RTP/RTCP protocol stack

2.1.4 TFRC

Currently, TCP and UDP are two common transport layer protocols widely used in Internet. TCP provides certain network monitoring to avoid congestion and ensure each session receives its fair share of the available bandwidth. On the contrary, UDP does not provide a congestion control mechanism so that a UDP-based transmission consumes as much bandwidth as available, with the consequence that TCP-based transmissions do not get their fair share of the bandwidth. Therefore, UDP is not proper for the video streaming applications due to its greedy and TCP-unfair characteristic. Although video streaming sessions should be designed that behaves like TCP sessions, TCP itself is not well-suited for real-time streaming because the reliability and ordering semantics that it ensures increase end-to-end delays and delay variations. TCP-Friendly Rate Control (TFRC) protocol [12][13][14], which implement receiver-driven congestion control algorithms that are stable and interact well with TCP, are proved to fit for real-time streaming applications very well.

Recently adopted by the IETF, TFRC provides sufficient responsiveness by taking into consideration all the parameters that affect the TCP rate such as loss, round-trip time and retransmission timeout value. The key advantage of TFRC is that it has a more stable rate during the session lifetime. The TFRC sender (streaming server) estimates the throughput of a TCP session sharing the same path using (2-1) [14]:

$$X = \frac{s}{R\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \qquad (2\text{-}1)$$

where X is the transmit rate in bytes/second, s is the packet size in bytes, R is the round-trip time in seconds, p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted, t_{RTO} is the TCP retransmission timeout value in seconds, and b is the number of packets acknowledged by a single TCP acknowledgement.

2.2 Scalable Video Coding (SVC) Methods

Scalable video coding (SVC) [15][16][17] is an attractive coding method with wide application in network-based video applications. It has the capability to reconstruct low resolution or low quality images from partial video streams. This allows for easy adaptation to both network conditions and device capabilities. In this section, we review the characteristics and development history of SVC, as well as corresponding International standards.

2.2.1 SVC characteristics and development history

Currently, video delivery over network is a popular research area that receives a lot of attention. The videos with distinct lengths and bit rates are transmitted over both wired and wireless networks with varying bandwidth. They are also stored or displayed on a variety of devices with different capabilities, ranging from PDA, mobile phone to PC and HDTV systems. All these require the videos to be adapted or degraded gradually and flexibly. Nevertheless, traditional non-scalable coded videos do not meet these requirements very well. Hence, SVC was recently developed to overcome the disadvantages of non-scalable video coding methods.

SVC intends to encode the image signals once at the highest resolution in temporal, quality, or spatial dimension, and enables the decoding of partial video streams to provide scalability in each dimension. This facilitates a simple and flexible solution for network transmission and adaptation. In many network applications, lots of devices with heterogeneous capabilities exist concurrently, and their resources are not known to the video encoder. In addition, network conditions may vary frequently, and hence devices are not able to receive full bit streams sometimes. This will result in no decoding at all or pose difficulties for non-scalable codec in devices to reconstruct partial information. The compactness of non-scalable coded video not only brings difficulties for devices, but also makes video adaptation complex and inflexible. SVC, however, excels in these aspects. For highest flexibility, advanced SVC achieves scalability with both fine granularity at the bit stream level and university in terms of different dimensions.

Modern digital image and video coding research started from 1970s and matured in 1990s with the advent of block-based motion compensation/discrete cosine transform. Early video compression standards such as ITU-T H.261 [18] and ISO/IEC MPEG-1 [19] were primarily developed for conversational services or video storage, and did not provide any scalability mechanisms. These standards were released around mid of 1990s and widely used for distributing digital video at that time. ISO/IEC MPEG-2 [20] was the first general purpose video compression standard that introduces a number of tools providing scalability. MPEG-2 is back compatible with MPEG-1 where base information could be decoded by old standard, and higher quality enhancement information is processed by the new standard. MPEG-2 achieves great success around the world, especially in the field of digital TV. Scalable video coding is introduced in MPEG-2, and further developed in MPEG-4 [21] standard. Proposed in around 2000, MPEG-4 provides more flexible scalability tools, including temporal scalability within a more generic framework and Signal-to-Noise Ratio (SNR) scalability with fine granularity (FGS). MPEG-4 FGS transmits DCT coefficients in separate bit streams with high granularity and has low complexity in nature. Moreover, MPEG-4 recently incorporates advanced video coding, e.g. H.264 [22], as its part 10, and has found widespread application in various digital video consumer products. Intense research is still being carried out to provide more efficient, more flexible, and more robust video compression algorithms.

2.2.2 MPEG-4 Fine Granular Scalability (FGS)

MPEG-4 FGS has been developed with the growing need of efficient scalable video coding methods in video streaming system. The FGS framework consists of a base layer (BL) and one enhancement layer (EL). Two types of ELs are defined for hybrid temporal-SNR scalability in MPEG-4 FGS [23]: SNR FGS layer and temporal FGS (FGST) layer. Figure 2-II shows one implementation of a hybrid temporal-SNR scalability structure. The SNR FGS layer includes several discrete bit-planes and FGST layer contains a sequence of discrete B frames. Each B frame in FGST layer is predicted according to the BL frames.

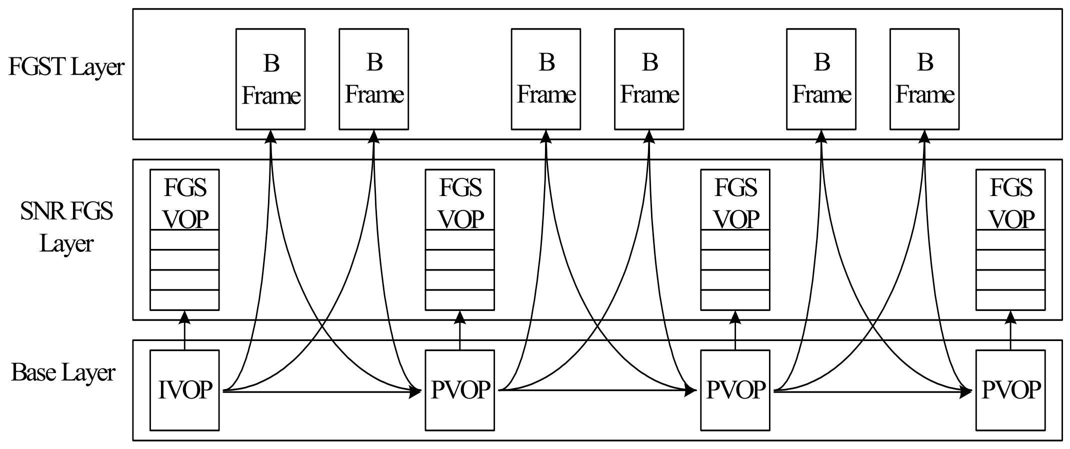


Figure 2-II: Structure of hybrid temporal-SNR scalability

2.3 Internet Video Streaming Systems

A typical video streaming system consists of a source server and several clients. The server connects to each client through networks and stores the videos, such as short video clips, full-length movies, and online lectures, which are compressed by video encoder. Upon a client's request, the server retrieves compressed video content from its storage, and adjusts the video transmission rate according to network status. The content is packed into bit streams and sent to clients through Internet.

Large scale video streaming system usually contains millions of clients sending requests simultaneously. These clients may spread around the world and consume a large amount of resources, such as storage space, computational capacity, as well as network bandwidth. These limited available resources and unpredictable Internet environment make it a challenging task to design an efficient video streaming system in a large scale. A number of technologies are being investigated to reduce the resource requirements and improve the system scalability. Generally, they are classified as the following two categories:

- Improved streaming algorithms [24][25][26][27], such as video transmission schemes, and scheduling algorithms;
- Advanced streaming architecture, such as P2P systems [28][29], and proxy-assisted systems [30][31][32].

P2P approach is to generalize the source server functionalities into each client, and these clients collaborate with one another to deliver video streams. Such a P2P communication paradigm allows economical clients to contribute their local storage and computational capacity for video streaming, and hence improves the system scalability. Specifically, the video data originally provided by a server are spread among clients, and each client stores partial or full versions of the video in its local. Then, one or more clients can coordinate to supply cached data to other clients. An increase in client demand often raises the capacity of a P2P system. Nevertheless, these clients may only wish to contribute limited resources, and sometimes leave the system unpredictably. Moreover, it is difficult to identify and authorize true clients to join in the system. These all make P2P not reliable as traditional client-server system.

Another approach is to use proxy to reduce source server workload, network traffic, and access latencies. These proxies are small servers which are strategically placed at network edges and usually close to a group clients based on the belief that different clients will retrieve many of the same video contents. Hence, these contents are cached on nearby proxies. Both synchronous and asynchronous demands for these video contents can benefit from the deployment of proxies. The system performance is, therefore, significantly improved given the static nature in video contents and highly localized access interests.

In proxy-assisted streaming systems, when a client requests for a particular video that the proxy has stored, the proxy will adapt its local copy according to the request requirement and return the video directly to the client without contacting the remote source server. If the requested video is not stored locally in proxy cache, the video will be retrieved from the server and relayed to the client. In case that many clients request for the similar video contents, the proxy can coordinate these streams to reduce resource consumption on both source server and network. Hence, the main functions that proxies provide can be summarized as: caching, adaptation, and transmission. Caching is to utilize the storage space on proxy to respond to clients' requests promptly and reduce the network traffic and server load. Adaptation means to convert the video into suitable format or bit rate to fit for the network and meet clients' requirements. Transmission is to deliver video streams efficiently considering the clients access behavior and streaming environment.

A typical proxy-assisted video streaming architecture [31] is shown in Figure 2-III. A source server delivers videos from its storage to a large number of clients across networks. Some proxies are strategically placed between the server and clients. The source server maintains a number of streaming sessions with the proxies, and each proxy in turn maintains session with each connected client. These proxies store carefully selected video segments to respond to clients' requests promptly. Each client sends requests to the connected proxy and decodes received video data for display. Overall, the proxy-assisted streaming system exploits the storage and processing capacity of proxies, and distributes video streams efficiently. We treat the streaming architecture in Figure 2-III as the basis of this dissertation and will cite this architecture in the following chapters.

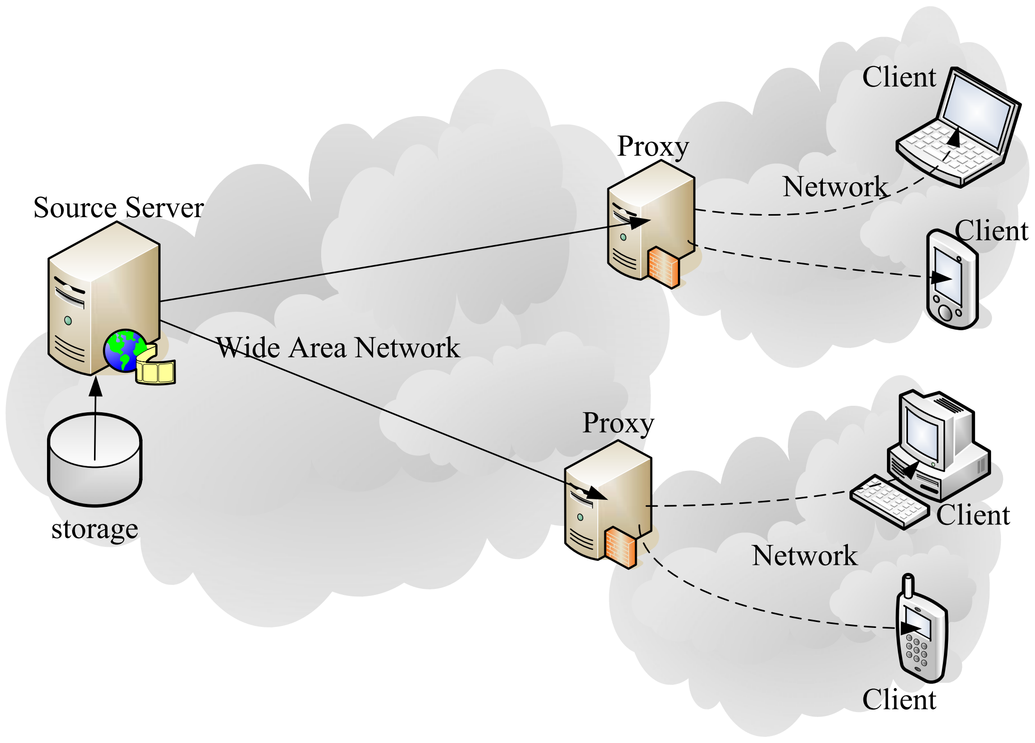


Figure 2-III: A typical proxy-assisted video streaming architecture

2.4 Research Challenges

In proxy-assisted video streaming systems, the design of proxy is more complex than that of source server. To deliver video streams on time to the clients is the main objective a source server has to achieve. Nevertheless, proxy has to determine which part of the videos should be cached and how they could be retrieved from the cache to send to the clients. The main objective of a proxy is to incorporate its own resources into the system so that the performance of the whole streaming system is improved. In this case, the management of different resources in the system, such as proxy storage capacity and network bandwidth, becomes inter-related and complicated. In the following paragraphs, we describe a number of specific research challenges related to the proxy-assisted video streaming.

In existing Internet, unpredictable bandwidth fluctuation is a prevalent problem for network-based applications including real-time video streaming. The bandwidth may vary from exceeding the QoS requirement to below the QoS requirement from time to time. For a streaming client, this will cause video quality degradation. To reduce this negative effect on received video quality, advanced transmission schemes should be employed to adjust streaming rate adaptively such that video quality can be improved for clients. We investigate the adaptive transmission problem in Chapter 3.

Video proxy usually needs to serve many clients' requests simultaneously. These clients sometimes request for the same video with different bit rate versions. If we store several of these versions on proxy and source server, or create separate streaming session between proxy and source server to respond to each request, the system resources will soon be exhausted with the increase of client reservoir. Video adaptation is the technology to deal with this problem. It enables the proxy to convert between different versions of the same videos and benefits the streaming system in at least the following two aspects:

- Several streaming sessions for the same video content can be combined into one session to save network resources, even if these sessions are created with different bit rates;
- The storage space on the source server and proxies could be saved by storing only one bit rate version of the videos.

With the advent of SVC, more efficient video adaptation methods could be designed. We investigate the adaptation problem for scalable coded video in Chapter 4.

Expensive bandwidth cost within Wide Area Network (WAN), and long response latency for clients are two important issues that inhibit the large scale deployment of video streaming systems. Proxy caching is generally regarded as one of the most effective methods to address the issues. The effectiveness of caching gains its reputation from web caching which is widely employed in web-based services. Nevertheless, proxy caching for video streaming is quite different from web caching and raises many challenges such as the large size of videos, long duration of streaming sessions, and continuous streaming constraints. We need to design efficient proxy caching scheme to solve these issues, with the new opportunities provided by the advent of SVC. We examine this proxy caching problem in Chapter 5.

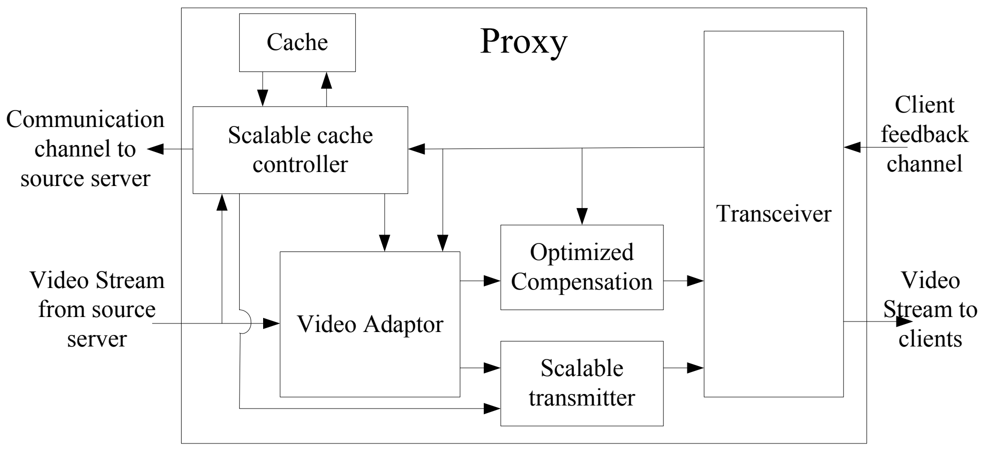


Figure 2-IV: Streaming proxy structure

Overall, a streaming proxy structure incorporating our contributions is shown in Figure 2-IV. This proxy is an intermediate system node which cooperates with central source server, and coordinates client requests and ongoing video streams to provide improved quality of streaming service with reduced resource consumption. Our four contributions, which are represented as four major component modules in Figure 2-IV, work closely to provide proxy functionalities as a whole. These modules are explained as follows:

- Scalable Cache Controller (SCC) module determines the appropriate video segments to cache and requests for the uncached portion from the source server at a proper timing.

- Video Adaptor (VA) module adapts the received video stream from source server and SCC module to the target bit rate according to network situations and request requirements.
- Optimized Compensation (OC) module receives the discarded video data from VA module and chooses the proper video segment and timing to compensate to the client by monitoring the network condition constantly.
- Scalable Transmitter (ST) module cooperates with SCC module to coordinate the video transmission to meet various requests with different requirements.

A client request is first received by the transceiver and passed to the SCC module. If SCC module can satisfy this request through its internal cache, it will fetch its local video data and pass the data to VA module for adaptation. If the request cannot be satisfied locally, SCC module will decide the proper video data and timing to request to the source server. In addition, SCC module constantly computes the statistics of the requests and determines the proper video data to cache. VA module receives the video data from both source server and SCC module, and shapes the bit rate of the video stream based on network conditions and request requirements. Adapted video is output to ST module for transmission and discarded data is sent to OC module for compensation. OC module monitors the network continuously and chooses the proper timing and video segments to compensate to the clients. ST module receives the video from VA module and coordinates the transmission of multiple ongoing streams based on SCC decisions. All the streams are finally transmitted to the clients through the transceiver. The details of each component module are described in the following chapters.

Chapter 3: ADAPTIVE VIDEO STREAMING VIA OPTIMIZED COMPENSATION

This chapter proposes an adaptive compensation method (ACM) for real-time video streaming. By exploring the characteristics of scalable coded video and monitoring client buffer constantly, ACM determines the video segment to be compensated such that the negative effect of bandwidth fluctuation on video quality is minimized. Firstly, we analyze the transmission problem for proxy-assisted video streaming system and describe a representative streaming scenario. Secondly, we formulate the compensation model with closed-form mathematical expressions. Thirdly, we obtain the optimized compensation segment by satisfying both real-time and buffer constraints. Finally, extensive simulations are performed to analyze the effects of environment parameters on the performance of ACM and illustrate the superiority of ACM. In summary, proposed method can efficiently improve the video quality during bandwidth fluctuations.

3.1 Background

Recent advances in digital video codec and network technologies have led to the rapid development of a wide range of multimedia streaming applications, such as video-on-demand, interactive/live video streaming, videoconferencing, and remote education, to meet the huge demands of Internet users for the audiovisual content.

Many of the applications involve advanced video and network technologies to enable real-time video streaming. These technologies, such as video analysis, video coding, and video retrieval, are being investigated extensively in order to provide satisfactory services [16][33][34]. Nevertheless, the efforts are not enough due to the stringent user requirements, inherent video characteristics, and existing network development.

User requirements, such as smooth video playback, good image quality, and short access latency, pose stringent transmission delay/jitter and packet loss constraints on the streaming applications and underlying networks. The large size of video data represents another difficulty when transmission constraints need to be met. In addition, although there has been an immense amount of work on quality of service (QoS), the packet networks, like Internet, employed for video streaming are predominantly best-effort in nature. This means that there is no guarantee on the deliveries of packets sent across networks, because no resource such as a guaranteed amount of bandwidth, packet loss rate, and transmission latency on the link between two nodes can be reserved. The streaming applications try to deliver video content in their best effort and usually need to estimate the most appropriate amount of available network resources constantly. Based on their estimation, these applications transmit video data with appropriate bit rate to the users for playback. The network variation is accentuated by the increasing mobility of users and their changing preferences or customization in the video streaming applications. This variation should be observed and then utilized to dynamically guide the network-adaptive control of video transmission. Besides of the time-variant network, the inherent timing constraints associated with the playback of video data rend the problem of real-time video streaming even more challenging.

Network-adaptive video streaming usually includes two fundamental functionalities: (1) on-line network monitoring which captures underlying channel condition in real-time, and (2) selective video transmission which chooses appropriate video segments to transmit at proper timings.

Most approaches for the network monitoring are receiver-driven, where the receiver measures receiving statistics such as packet loss/delay/jitter and reports them to the sender using feedback packets [35][36][37]. Another approach is send-driven monitoring, where sender estimates the channel condition by observing underlying data link layer and physical layer transmission statistics. A streaming system adopting sender-driven monitoring scheme should be established based on cross-layer design so that the application can interact with the underlying network seamlessly [38][39].

Existing selective transmission schemes usually employ adaptation method to shape video stream. A detailed survey on video adaptation is given in [40]. Most of the adaptation methods, including priority dropping [41], window-based rate control [42], and optimized adaptation [43][44], all focus on discarding data when available bandwidth is not sufficient to transmit video with negotiated QoS. Nevertheless, when bandwidth varies to exceed the QoS requirement, existing selective transmission schemes do not utilize this extra bandwidth efficiently. We hereby aim at utilizing the extra bandwidth to compensate the previously discarded video data to improve video quality. To the best of our knowledge, this type of compensation method has not been investigated systematically.

Our goal is to always deliver content at a quality level that requires a rate not exceeding the average bandwidth. This is to avoid playback interruptions due to the insufficient bandwidth in the long term. When bandwidth is not sufficient to deliver video at negotiated bit rate, we may have to discard some enhancement layer (EL) data of the scalable coded stream. This will cause quality degradation at client. On the other hand, when bandwidth resumes and exceeds the required bit rate, this extra network capacity can be utilized to transmit and compensate parts of the discarded data. This compensation segment should be carefully selected such that both client buffer constraint and real-time delivery is ensured.

In this chapter, we propose an adaptive compensation method (ACM) that can compensate optimized video segments to clients at proper timings to improve video quality. This method takes client buffer utilization information into consideration and guarantees the real-time video delivery. Client buffer, which is usually used to smooth transmission delay/jitter and receive disordered packets, will bring presentation delay for the client. This presentation delay represents the duration from the time that the server starts sending out the video data to the time that the client decodes or displays the data. Nevertheless, end-to-end transmission delay has a different meaning and denotes the duration from the time that the server starts sending out the video data to the time that the client receives the data. All the video data that has been received, yet not decoded by the client is saved in the client buffer. Generally, the transmission delay is measured in terms of milliseconds, while the presentation delay is orders higher and measured in terms of seconds. Although there are many research works on the estimation of end-to-end transmission delay such as window estimators [45], the research on the estimation of presentation delay for scalable coded video is not enough. In our research, we estimate the presentation delay by receiving the real-time client buffer utilization information through RTCP protocol stack, and adaptively adjust the streaming rate and transmission segments to make sure that the client buffer constraint is not violated. In case that client receives a bit rate lower than the content consumption rate, a so-called buffer underflow may occur. Underflow usually causes the client to stop its playback and refill its buffer before continuing playback. On the contrary, when client receives a bit rate higher than the consumption rate, buffer overflow may happen. Overflow will waste network resources and have an effect equivalent to packet loss because the packets causing overflow are discarded. Both underflow and overflow have a negative effect on video quality. We should ensure that compensated video data will not cause any overflow and underflow. In addition, to guarantee uninterrupted playback, video packets need to arrive at client on time to meet presentation deadlines. Delayed packets will introduce playback jitter and violate the real-time constraint. We should make sure that compensated video data arrive at client on time for presentation.

The remainder of this chapter is organized as follows. Section 3.2 provides an overview of the transmission problem due to bandwidth fluctuation and Section 3.3 proposes the adaptive compensation method based on a mathematical optimization model. Section 3.4 evaluates the performance of proposed method through simulations and analysis. Finally, Section 3.5 gives some concluding remarks on this chapter.

3.2 General Problem Description

In this section, we provide an overview of the video streaming problem due to bandwidth fluctuation.

We consider the interaction between one proxy and one client within the proxy-assisted video streaming architecture shown in Figure 2-III for illustration. Although we focus on solving the transmission problem for the proxy, the proposed adaptive compensation method can be directly used in various client-server based streaming systems. In Figure 2-III, a client usually sends requests to the proxy and negotiates for the QoS. After mutual agreement is reached, the proxy begins to stream requested video to the client with the negotiated bit rate. Due to network variation, available bandwidth between the proxy and the client may drop below the required bit rate. Consequently the proxy has to discard parts of the EL data. Later if the available bandwidth resumes and exceeds the required rate, this extra bandwidth can be utilized to compensate the discarded video data if the following two constraints are met:

- Real-time constraint: The compensated EL data should be transmitted before their presentation deadlines are due. In other words, the corresponding BL has not been decoded and displayed at client.
- Client buffer constraint: The compensated EL data should not cause client buffer overflow and underflow.

A general transmission scenario with bandwidth fluctuation is shown in Figure 3-I and the corresponding parameters are summarized in Table 3-I. Some parameters, such as D and C, are determined by the proxy during the negotiation process before the streaming process. Some parameters, such as $B(t)$ and $b(t)$, are provided by the underlying protocol, such as TFRC and RTCP in real-time. TFRC and RTCP are both properly designed such that we can estimate available bandwidth with TFRC and inform proxy of the client side information by attaching them to RTCP messages. Others, such as $x(t_0, t_1)$, are measured and recorded by proxy itself. In the beginning during time interval $(0, t_0)$, although available bandwidth is more than sufficient ($B_{avg}(0, t_0) >$ D), proxy still has to stream the video with negotiated bit rate D ($S(t) = D,\ 0 \le t \le t_0$). At time t_0, proxy senses that the available bandwidth drops seriously below the QoS requirement ($B_{avg}(t_0, t_1) < D$). It has to discard some EL data of the video stream (region I and II in Figure 3-I) to match its transmission rate to the bandwidth ($S(t) = B(t) < D,\ t_0 \le t \le t_1$). Later at time t_1, when available bandwidth resumes ($B(t_1) > D$), the transmission rate can be increased back to the bit rate D. Nevertheless, instead of wasting the extra bandwidth ($B(t_1)-D$) after time t_1, we can increase the transmission rate beyond rate D, such as $B(t_1)$, and start to compensate the data loss within time interval (t_0, t_1). The compensation process terminates at time t_2 after the selected compensation segment (region II in Figure 3-I) is transmitted to the client. After time t_2, we force the transmission rate back to the rate D to meet the long term negotiation if the available bandwidth remains stable. Otherwise, the discarding and compensation processes will continue to take place until the end of the streaming session.

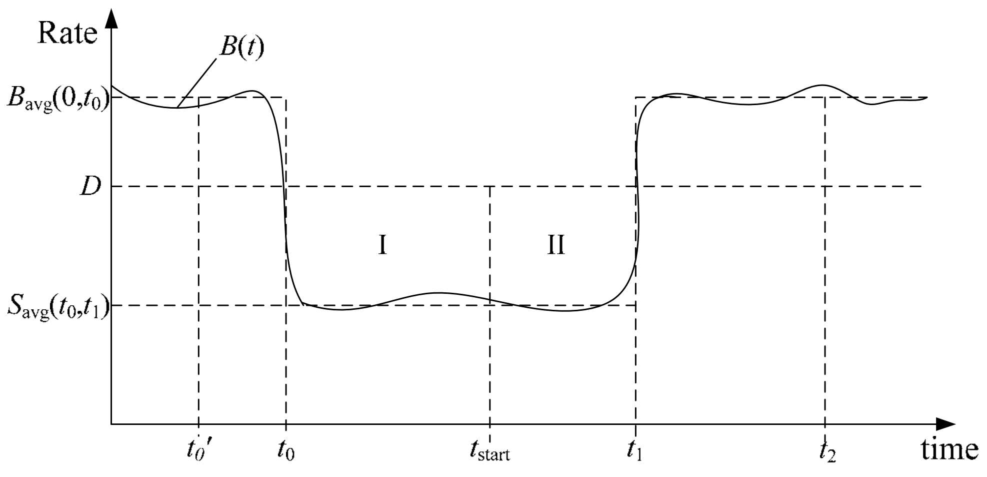


Figure 3-I: Transmission scenario with bandwidth fluctuation

Table 3-I: Parameters of the adaptive video transmission scenario and compensation model

Notation	**Description**
D	Negotiated bit rate (QoS requirement)
C	Client buffer size
$d(t)$	Bit consumption rate of client buffer at time t
$B(t)$	Available bandwidth at time t
$x(t_0, t_1)$	Size of the discarded data between time t_0 and t_1
$b(t)$	Size of the residual data within client buffer at time t
$S(t)$	Transmission rate at time t
$S_{avg}(t_a,t_b)$	Average transmission rate between time t_a and t_b
$B_{avg}(t_a,t_b)$	Average available bandwidth between time t_a and t_b

During the compensation process starting at time t_1, it may not always be desirable to transmit all the discarded data, such as those data represented by region I and region II together in Figure 3-I, due to the real-time and client buffer constraint. Generally, only part of the discarded data, so called compensation segment, can be compensated and utilized to improve video quality. This compensation segment can be characterized by the following two parameters: start point and stop point. Intuitively, the discarded data around time t_1 should be included into the compensation segment with higher priority because they have relatively loose presentation deadline at client. On the other hand, discarded data near time t_0, if selected into the compensation segment, should be transmitted to the client immediately due to their stringent presentation deadline. Therefore, we choose time t_1 as the stop point of the compensation segment and focus on determining the start point t_{start}.

Under both real-time and buffer constraints, we aim at compensating as much as possible of the discarded data within time interval (t_0, t_1) and maximizing the compensation effect C_e which is defined by (3-1):

$$C_e = \frac{C_d}{D_d + C_d} \tag{3-1}$$

where D_d is the overall discarded data size after compensation process completes and C_d is the compensated data size. They are respectively represented by region I and region II in the transmission scenario (Figure 3-I).

3.3 Adaptive Compensation Method

In this section, we consider two client buffer utilization scenarios and set up a mathematical model to derive the optimized compensation segment. For the ease of illustration we omit the network transmission latency between proxy and client. This latency is usually smaller than the time interval between two consecutive video frames ($\approx 33ms$) and has neglectable influence on the final result.

3.3.1 Video compensation model

Due to the real-time constraint described in Section 3.2, the compensation segment should be transmitted before the client starts to decode this segment together with its corresponding lower layer. We express this constraint in (3-2). The left side denotes the time to transmit the selected segment with average transmission rate $B_{avg}(t_1,t_2)-D$. The right side represents the time for the client to consume its buffer residual data before reaching the corresponding lower layer of the compensation segment.

$$\frac{\int_{t_{start}}^{t_1}(D-S(t))dt}{B_{avg}(t_1,t_2)-D} \le \frac{b(t_1)-\int_{t_{start}}^{t_1}S(t)dt}{d_{avg}(t)} \tag{3-2}$$

Similarly, we express the client buffer constraint in (3-3). The second term of the left side denotes the data increase inside client buffer during the compensation process ($t_1 \le t \le t_2$).

$$b(t_1)+\int_{t_1}^{t}(B(t)-d(t))dt \le C \quad (t_1 \le t \le t_2) \tag{3-3}$$

Figure 3-I shows that the size of compensation segment (C_d) grows with the decrease of t_{start}. Since the compensation effect C_e is proportional to C_d, we aim at minimizing t_{start} instead of maximizing C_d or C_e. The compensation model (**CM**) can be formulated as:

$$\begin{aligned} \textbf{CM}\text{:}\quad \text{Minimize:}\quad & t_{\text{start}} \\ \text{subject to:}\quad & t_0 \le t_{start} \le t_1 \\ & \frac{\int_{t_{start}}^{t_1}(D-S(t))dt}{B(t)-D} \le \frac{b(t_1)-\int_{t_{start}}^{t_1}S(t)dt}{d(t)} \\ & b(t_1)+\int_{t_1}^{t}(B(t)-d(t))dt \le C \ \text{(for each } t_1 \le t \le t_2) \end{aligned}$$

3.3.2 Optimized compensation segment

In the compensation model, we consider constant frame consumption rate at client. That is, in face of bandwidth fluctuation, the streaming system will try to preserve the frame rate, instead of frame quality, to avoid playback jitter. For example, in Figure 3-I the client consumes the frames received in time interval (0, t_0) at bit rate D, but it consumes the frames received in time interval (t_0, t_{start}) at bit rate around $S_{\text{avg}}(t_0, t_{\text{start}}) <$ D. Based on the residual data size of client buffer at the start of compensation process ($b(t_1)$), we classify the following two situations.

- Client buffer is not well utilized;
- Client buffer is well utilized.

The threshold to differentiate these two situations is determined based on the buffer consumption rate at time t_1. If the client is consuming the data received after time t_0, it belongs to the first situation and the consumption rate is $S_{avg}(t_0,t_1)$. Otherwise, it belongs to the second situation and the consumption rate is D. Hence, the threshold for the buffer residual data size ($b(t_1)$) can be set as $(t_1-t_0)\cdot S_{avg}(t_0,t_1)$. In the following paragraphs, both situations are explained in detail and the simplified expressions for above integral equations are obtained accordingly.

3.3.2.1 Client buffer is not well utilized:

In this situation, the buffer residual data size is less than the threshold which is expressed in (3-4). With the client presentation delay τ, the consumption rate at client buffer is expressed in (3-5). By utilizing the average bandwidth B_{avg} and average transmission rate S_{avg}, we rewrite the real-time and buffer constraints in (3-2) and (3-3) as (3-6) and (3-7), respectively.

$$b(t_1)\le(t_1-t_0)\cdot S_{avg}(t_0,t_1) \tag{3-4}$$

$$d(t)=S(t-\tau)<D \quad \text{(for each } t_1\le t\le t_2\text{, and } t_0\le t-\tau\le t_1\text{)} \tag{3-5}$$

$$\frac{(t_1-t_{start})\cdot(D-S_{avg}(t_0,t_1))}{(B_{avg}(t_1,t_2)-D)}\le\frac{(b(t_1)-(t_1-t_{start})\cdot S_{avg}(t_0,t_1))}{S_{avg}(t_0,t_1)} \tag{3-6}$$

$$b(t_1)+(t-t_1)\cdot(B_{avg}(t_1,t_2)-S_{avg}(t_0,t_1))\le C \quad \text{(for each } t_1\le t\le t_2\text{)} \tag{3-7}$$

To prevent buffer overflow, we choose to calculate the time t_{full} at which the buffer has the most residual data during the whole compensation process, and ensure buffer constraint is satisfied at time t_{full}. By substituting t using t_{full}, we obtain the buffer constraint in (3-8). Since client buffer always has an input bit rate larger than its consumption rate ($B(t) > d(t),\ t_1 \le t \le t_2$) during the entire compensation process, t_{full} should be the same as the compensation completion time t_2 shown in Figure 3-I ($t_{\text{full}} = t_2$). In addition, since the compensated segment between time t_{start} and t_1 is compensate between time t_2 and t_1, we obtain the equation (3-9) and then derive t_{full} and t_2 in (3-10).

$$b(t_1) + (t_{full} - t_1) \cdot (B_{avg}(t_1, t_2) - S_{avg}(t_0, t_1)) \le C \tag{3-8}$$

$$(B_{avg}(t_1, t_2) - D) \cdot (t_2 - t_1) = (t_1 - t_{start}) \cdot (D - S_{avg}(t_0, t_1)) \tag{3-9}$$

$$t_{full} = t_2 = t_1 + \frac{(t_1 - t_{start}) \cdot (D - S_{avg}(t_0, t_1))}{B_{avg}(t_1, t_2) - D} \tag{3-10}$$

By substituting t_{full} and t_2 in (3-6) and (3-8) using (3-10), we derive the real-time and buffer constraints for t_{start} in (3-11) and (3-12). Therefore, the optimal compensation point t^{o}_{start} can be expressed in (3-13), where $B_{\text{avg}}(t_1,t_2)$ and $S_{\text{avg}}(t_0,t_1)$ are abbreviated as B and S, respectively.

$$t_{start} \ge t_1 - \frac{b(t_1) \cdot (B_{avg}(t_1, t_2) - D)}{(B_{avg}(t_1, t_2) - S_{avg}(t_0, t_1)) \cdot S_{avg}(t_0, t_1)} \tag{3-11}$$

$$t_{start} \ge t_1 - \frac{(C - b(t_1)) \cdot (B_{avg}(t_1, t_2) - D)}{(D - S_{avg}(t_0, t_1)) \cdot (B_{avg}(t_1, t_2) - S_{avg}(t_0, t_1))} \tag{3-12}$$

$$t^{o}_{start} = \max[t_1 - \frac{(C - b(t_1)) \cdot (B - D)}{(D - S) \cdot (B - S)}, t_1 - \frac{b(t_1) \cdot (B - D)}{(B - S) \cdot S}] \tag{3-13}$$

3.3.2.2 Client buffer is well utilized:

In this situation, the buffer residual data size is more than the threshold which is expressed in (3-14). After compensation process starts, the buffer consumption rate may change from rate D to $S_{\mathrm{avg}}(t_0, t_1)$ as the client decodes different parts of the received video stream. The consumption rate is expressed in (3-15).

$$b(t_1) \geq (t_1 - t_0) \cdot S_{avg}(t_0, t_1) \tag{3-14}$$

$$d(t) = \begin{cases} D & , \text{if } t - \tau \leq t_0 \\ S(t-\tau) < D, & \text{if } t_0 < t - \tau \leq t_1 \end{cases} \quad \text{(for each } t_1 \leq t \leq t_2\text{)} \tag{3-15}$$

In the beginning of compensation process, client is consuming the video data received before time t_0, for example time t_0' (t_0' < t_0), with rate D. After time t_0-t_0', client begins to consume the data received in time interval (t_0, t_1) with rate around $S_{\mathrm{avg}}(t_0, t_1)$ ($S_{\mathrm{avg}}(t_0, t_1) < D$). Therefore the real-time constraint in (3-2) can be rewritten as (3-16). The right side is the duration for the client to consume its residual data before the compensated segment is decoded.

$$\frac{(t_1 - t_{start}) \cdot (D - S_{avg}(t_0, t_1))}{B_{avg}(t_1, t_2) - D} \leq t_{start} - t_0' \tag{3-16}$$

Since the buffer residual data at time t_1 can be expressed in (3-17), we obtain the expression for time t_0' in (3-18). By substituting t_0' into (3-16) using (3-18), we rewrite the real-time constraint in (3-19).

$$b(t_1) = (t_0 - t_0') \cdot D + (t_1 - t_0) \cdot S_{avg}(t_0, t_1) \tag{3-17}$$

$$t_0^{'} = t_0 - \frac{b(t_1) - (t_1 - t_0) \cdot S_{avg}(t_0, t_1)}{D} \tag{3-18}$$

$$\frac{(t_1 - t_{start}) \cdot (D - S_{avg}(t_0, t_1))}{B_{avg}(t_1, t_2) - D} \le t_{start} - t_0 + \frac{b(t_1) - (t_1 - t_0) \cdot S_{avg}(t_0, t_1)}{D} \tag{3-19}$$

Similar to the situation in Section 3.3.2.1, we only need to make sure that client buffer will not overflow at the completion time of compensation process (t_{full}) which is expressed in (3-10). Nevertheless, $b(t_{\text{full}})$ takes two mathematical expressions based on the consumption rate $d(t_{\text{full}})$. Buffer constraints should be satisfied for both expressions. For example, if $d(t_{\text{full}})=D$, it means that client is still consuming the data received before time t_0 when the compensation process completes. The size of buffer residual data at time t_{full} is expressed in (3-20). The second term of the right side represents the compensated data. If $d(t_{\text{full}}) = S_{\text{avg}}(t_0, t_1) < D$, it means that client is consuming the data received in time interval (t_0, t_1) when compensation process completes. Client has consumed not only the data received in time interval $(t_0^{'}, t_0)$ at rate D, but also the data received in time interval (t_1, t_{start}) at rate $S(t) < D$ $(t_1 \le t \le t_{start})$. So the size of buffer residual data at time t_{full} is expressed in (3-21).

$$b(t_{full}) = b(t_1) + (t_1 - t_{start}) \cdot (D - S_{avg}(t_0, t_1)) \tag{3-20}$$

$$b(t_{full}) = b(t_1) + (t_2 - t_1) \cdot B_{avg}(t_1, t_2) - (t_0 - t_0') \cdot D - (t_{start} - t_0) \cdot S_{avg}(t_0, t_1) \tag{3-21}$$

By imposing the buffer constraint ($b(t_{\text{full}}) \le C$) on (3-20) and (3-21), we obtain the constraints in (3-22) and (3-23). After further substituting t_{full} and $t_0^{'}$ into (3-23) using (3-10) and (3-18), we can solve the real-time constraint in (3-19), and buffer constraints in (3-22) and (3-23). Combining these results, we obtain the optimal compensation point t_{start}^{o} in (3-24), where $B_{\mathrm{avg}}(t_1,t_2)$ and $S_{\mathrm{avg}}(t_0,t_1)$ are abbreviated as B and S, respectively.

$$b(t_1)+(t_1-t_{start})\cdot(D-S_{avg}(t_0,t_1))\le C \tag{3-22}$$

$$b(t_1)+(t_2-t_1)\cdot B_{avg}(t_1,t_2)-(t_0-t_0')\cdot D-(t_{start}-t_0)\cdot S_{avg}(t_0,t_1)\le C \tag{3-23}$$

$$\begin{aligned} t_{start}^{o} = \max[&t_1-\frac{C(B-D)}{D(B-S)}, t_1-\frac{C-b(t_1)}{D-S}, \\ &\frac{(D^2+SB-2SD)t_1+(DB-D^2-SB+SD)t_0-(B-D)b(t_1)}{(B-S)D}] \end{aligned} \tag{3-24}$$

3.4 Performance Evaluation

In this section, we examine the performance of proposed ACM through simulations. We analyze the effects of various environment parameters on the compensation effect and investigate the video quality improvement with ACM.

3.4.1 Simulation settings

We use MPEG-4 FGS coded video as example and developed the video streaming sub-system in NS2. These videos are coded with constant bit rate (CBR). The network topology of the simulation is shown in Figure 3-II. The sub-system omits the link between proxy and source server, and contains one proxy (node 2), one client (node 7), and several other components in Internet (such as switches in node 4 and 5, FTP server in node 1). The proxy and the client communicate based on TFRC protocol.

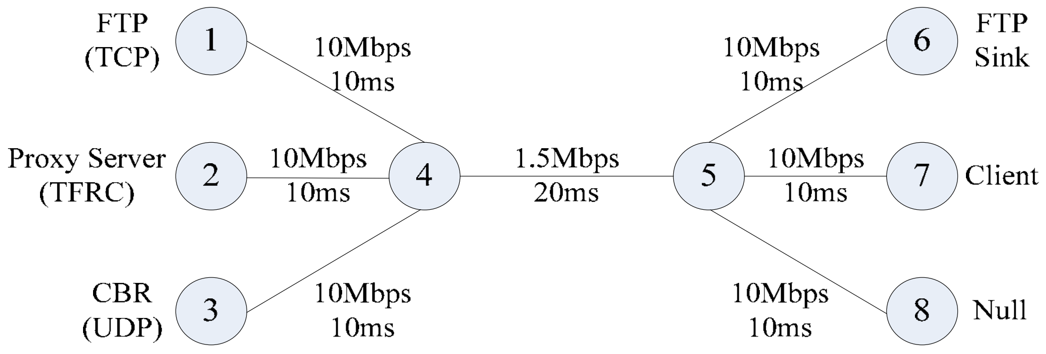


Figure 3-II: Network topology for the simulation

Table 3-II: Simulation parameters

Description	**Quantity**
FTP flow bit rate	300Kbps
CBR flow bit rate	700Kbps
Client buffer size (*C*)	3.5MBytes
Simulation length	60 seconds
CBR flow duration	[20, 40] seconds

All the queues use FIFO drop-tail scheduling discipline. The link delay is set to 20*ms* between two switches (node 4 and 5) and 10*ms* between a switch and an end system (such as node 5 and node 6). The link capacity is set to 1.5Mbps between two switches and 10Mbps between a switch and an end system. The TCP connections are modeled as FTP flow that lasts for the entire simulation. The UDP connection is modeled as CBR flow to impose extra network traffic on the link between node 4 and node 5, and simulates available bandwidth fluctuation for the proxy (node 2). Other general simulation parameters are listed in Table 3-II.

3.4.2 Adaptive video streaming scenarios

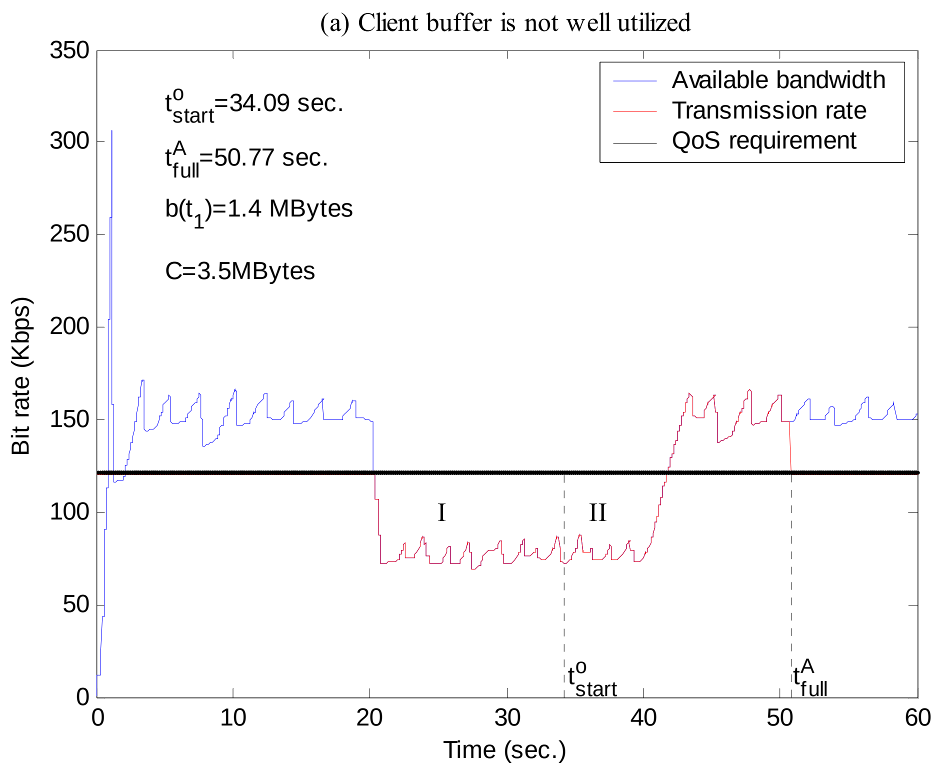
In this set of simulations, we illustrate the two video streaming scenario A and B which are respectively described in Section 3.3.2.1 and Section 3.3.2.2. The simulation parameters for both scenarios are listed in Table 3-III. In the beginning of the simulations, since negotiated bit rate (QoS requirement) is less than the available bandwidth ($D < B(t)$, $0 \leq t \leq 20\,\text{sec.}$), the proxy chooses to stream the video to the client at rate *D*. When CBR flow starts at 20 sec, the proxy senses that the available bandwidth drops below rate *D* ($S(t) < D$, $t = 20\,\text{sec.}$). Therefore, the proxy has to reduce the transmission rate and discard some of the EL data to meet the bandwidth constraint. The transmission rate drops to be the same as the available bandwidth until the end of CBR flow ($S(t) = B(t) < D$, $20 \leq t \leq 40\,\text{sec.}$). After the proxy senses the resume of available bandwidth due to the termination of CBR flow, it starts the compensation process according to the proposed ACM explained in Section 3.3. Based on the different size of buffer residual data at 40 sec., we simulate the two situations A and B which are identified in Section 3.3.2.

Table 3-III: Simulation parameters for adaptive video streaming scenarios

Description	**Quantity**
QoS requirement in bit rate (D)	120Kbps
Size of buffer residual data at 40 sec. in scenario A ($b^A(40)$)	1.4MBytes
Size of buffer residual data at 40 sec. in scenario B ($b^B(40)$)	2.5MBytes

We simulate the compensation process and plot the entire transmission scenario for both situations in Figure 3-III. In the simulations, the optimized start point of compensation segment (t^o_{start}) and the completion time of the compensation process (t_2) are calculated and shown in Figure 3-III.

In simulation scenario A and B, most of the parameters such as bandwidth constraint, buffer constraint, and simulation duration are similar except for the size of buffer residual data at the start of compensation process ($b(t_1)$). Hence, in most period of the simulation, such as time interval (0, 40), the transmission rates are similar in both scenarios. The only observable difference is that the transmission rate goes back to rate D earlier in scenario A ($t_2^A = 50.77\,\text{sec.}$) than that in scenario B ($t_2^B = 54.57\,\text{sec.}$). This is because scenario A imposes more stringent real-time constraint than scenario B does ($b^A(40) < b^B(40)$) and makes $t^A_{start} > t^B_{start}$. Region I and II in Figure 3-III together illustrate the size of the discarded data before compensation process. Region II alone denotes the compensated data. It is apparent that region II in scenario B is larger than that in scenario A, and hence scenario B has a better compensation effect.



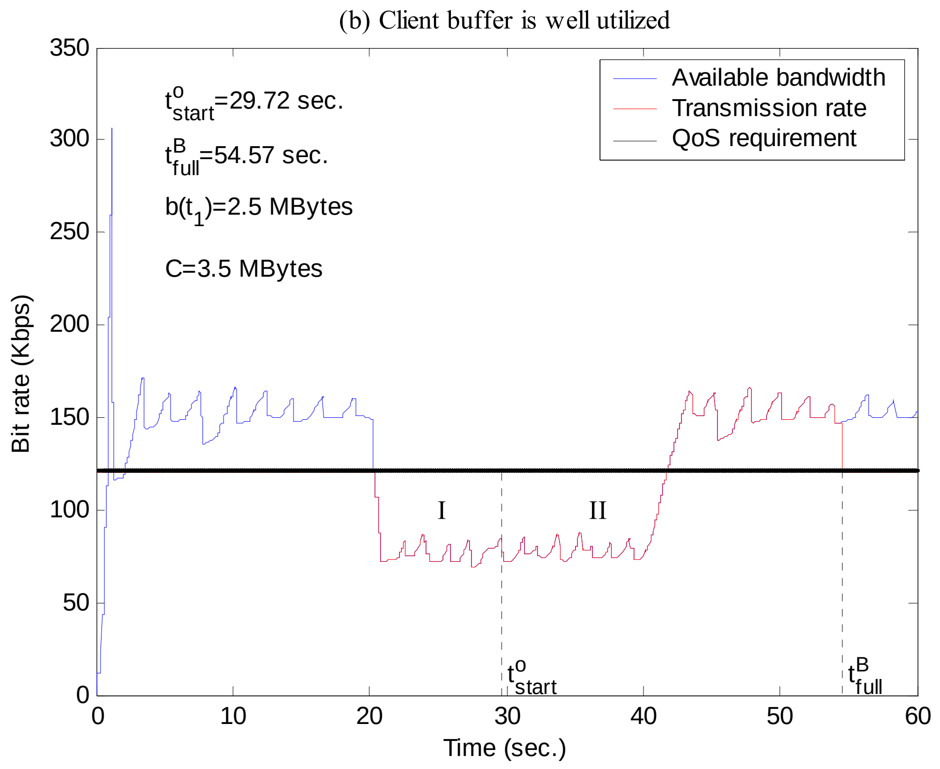


Figure 3-III: Video streaming scenarios with adaptive compensation method

3.4.3 Effect of client buffer utilization

In this part of simulations, we study the effect of buffer utilization ($b(t)/C$) in the beginning of compensation process on the final compensation effect.

We perform a set of simulations with the similar settings explained in Section 3.4.2, except that we adjust the size of client buffer residual data at time 40 second ($b(40)$). We fix the buffer size for the simulations (C=3.5MB) and gradually increase its utilization from 0 to 1. Accordingly, we calculate the compensation effect C_e and plot the results in Figure 3-IV. It shows that C_e is not linear with the client buffer utilization. In the beginning of the simulation, C_e is gradually improved with the increase of buffer utilization until reaching the point A in Figure 3-IV. This is because that the more residual data is inside the buffer, the more time is left for compensation and the better compensation effect is achieved. After point B, with the continuous increase of buffer utilization, C_e starts to degrade because less buffer space is left for compensation. In the case illustrated in Figure 3-IV, C_e reaches its maximum when the buffer utilization in the beginning of compensation process is between 74% and 86%. Therefore during the practical streaming session, it would be beneficial to keep the client buffer utilization to a certain level (such as 80% if the simulation parameters used in this chapter apply), such that compensation effect will be maximized.

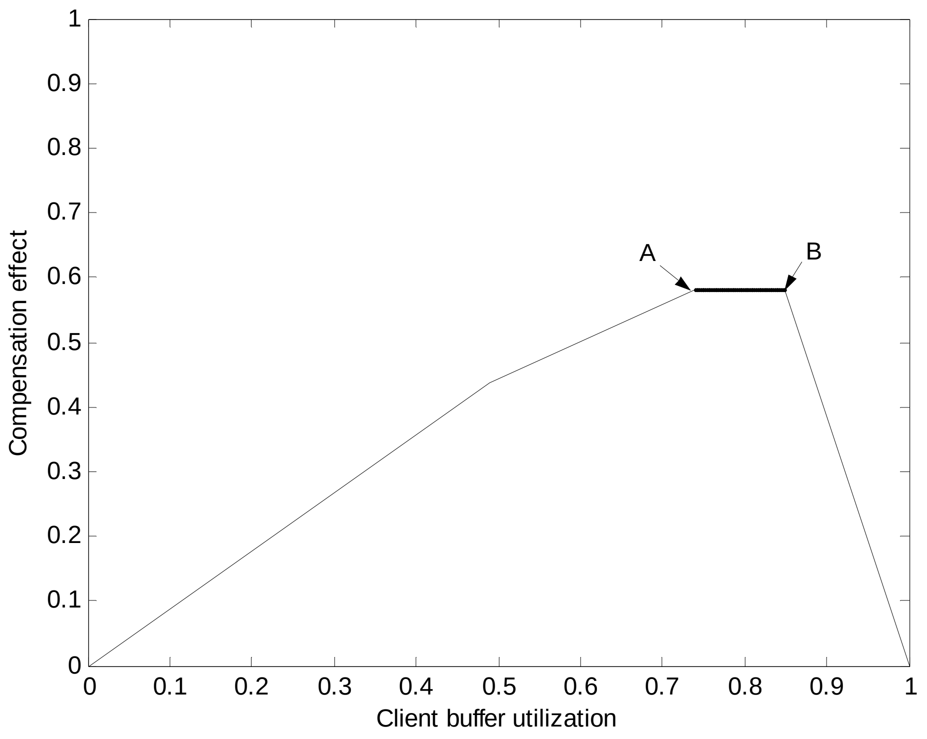


Figure 3-IV: Effect of buffer utilization ($b(t)/C$) on the compensation effect (C_e)

3.4.4 Effect of client buffer size

To investigate the effect of client buffer size on the compensation effect, we perform two sets of simulations in this section corresponding to the two scenarios described in Section 3.4.2. With the buffer residual data defined in scenario A and B, we gradually increase the buffer size (C) and repeat the simulations to calculate their compensation effects. The results are plot in Figure 3-V. It shows that in both simulation scenarios C_e grows with the increase of buffer size until reaching the threshold denoted by point A or B. After the threshold, the increase of buffer size does not bring better compensation effect. Therefore, in practical video streaming system, client buffer size does not need to be increased blindly because any size exceeding the calculated threshold will not further improve the compensation effect. It is worth noting that this threshold is determined by various system parameters such as buffer residual data size, available bandwidth, and so on.

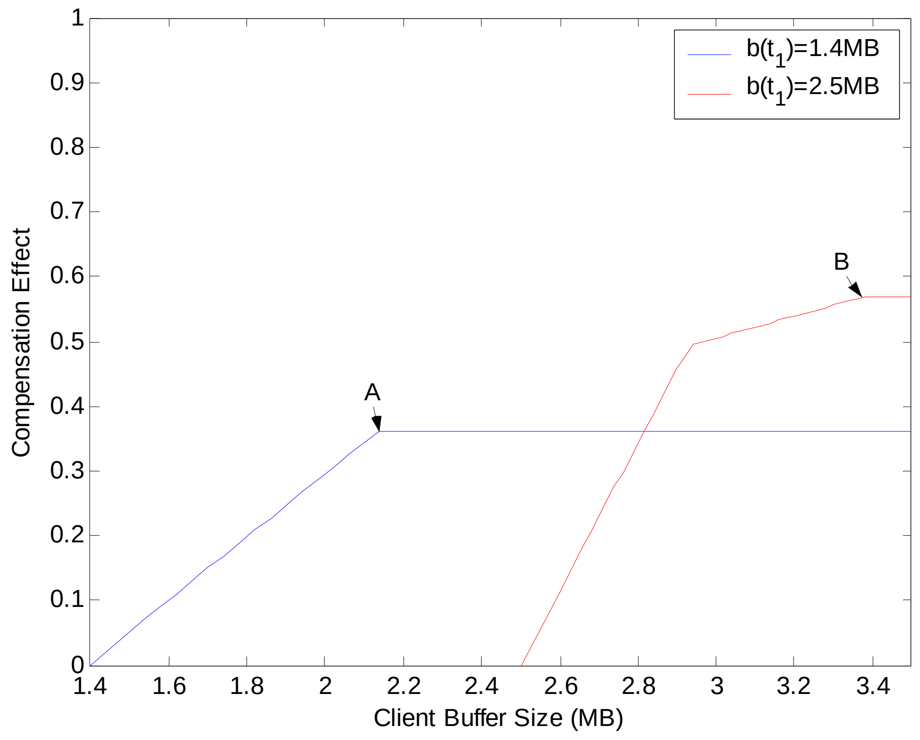


Figure 3-V: Effect of buffer size (C) on the compensation effect (C_e)

3.4.5 Video quality improvement

In this part of simulations, we are interested in examining the performance of proposed ACM in terms of video quality.

The simulations are performed with the similar settings of scenario B described in Section 3.4.2. The proxy transmits the MPEG-4 video stream using ACM and we calculate the quality of received video at client side accordingly. MPEG-4 traffic trace file presented in [46] is used in the simulations. We choose Peak Signal-to-Noise Ratio (PSNR) as the quality metric which can be expressed based on the three-color components (Y, U, and V) as in (3-25) and (3-26). The PSNR values obtained by the transmission scheme with ACM are compared with those obtained by the transmission scheme without ACM in Figure 3-VI.

$$PSNR = 20 \cdot \log \frac{255}{\sqrt{(4 \cdot MSE_Y + MSE_U + MSE_V)/6}} \tag{3-25}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} | Orignal(i,j) - Reconstruct(i,j) |^2 \tag{3-26}$$

Figure 3-VI shows that proposed method increases the video quality by 2.5 db for 100 frames, which provides around 8% quality improvement for 50% number of frames influenced by the bandwidth degradation. Although the performance of ACM, in terms of improved quality and compensation effect, depends on environment parameters, the effectiveness of ACM in real-time video streaming system is evident.

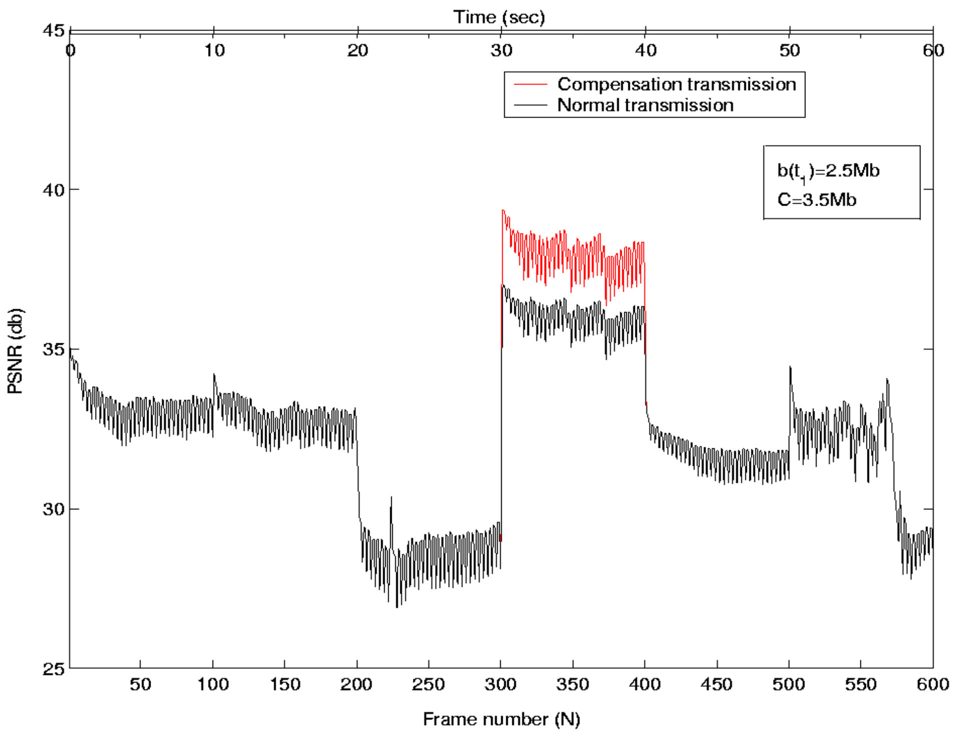


Figure 3-VI: Quality improvement achieved by the transmission scheme with ACM

3.5 Concluding Remarks

This chapter investigates the problem of streaming videos to clients under bandwidth fluctuations. By taking consideration of real-time and buffer constraints, we choose to compensate a video segment such that compensation effect is maximized. With proposed adaptive compensation method (ACM), parts of discarded video data due to bandwidth degradation are compensated to clients at a proper time and consequently video quality is improved. Our evaluations analyze the effects of various environment parameters on the ACM performance and demonstrate the superiority and effectiveness of ACM in terms of compensation effect and video quality.

Chapter 4: HIERARCHICAL VIDEO ADAPTATION

This chapter proposes an efficient video adaptation scheme for MPEG-4 fine granular scalability (FGS) streaming. By exploring the video texture and motional complexity, proposed adaptation scheme shapes video streams with minimum quality distortion and fluctuation. Firstly, we develop a mathematical model to estimate target bit budget for each video scene. The scene priority is incorporated to guarantee user-interested scenes a better quality. Secondly, we formulate a multi-objective optimization model to obtain the transmission segments within each scene. Finally, an excellent compromise between Signal-to-Noise Ratio (SNR) and temporal resolutions is achieved, such that the quality distortion and fluctuation are minimized. In summary, compared with other video adaptation methods, proposed scheme has the following superior characteristics which are demonstrated by extensive simulation results: (1) optimized perceptual quality; (2) flexible bit rate control; (3) low computational complexity.

4.1 Background

Video adaptation is an efficient technology to deal with heterogeneous streaming requests with distinct QoS requirements and resource constraints. Traditional streaming servers normally keep several versions of the same video sequence with different resolution and transmit separate streams for those heterogeneous requests. With video adaptation technology, source servers or intermediate streaming nodes, however, only need to store one copy of a specific video sequence and adapt it to satisfy different requests. Hence, video streaming systems, which are constrained by limited resources such as bandwidth and storage capacity, can accommodate more clients with reduced streaming overhead.

Flexibility is another advantage that video adaptation provides. Along the streaming path from source server to clients, any intermediate network node can employ different video adaptation schemes to meet various requirements imposed by sub-systems.

Efficient video adaptation schemes can be designed with the development of innovative video compression technology. Particularly, scalable video codec is receiving great attention as it is now regarded as a feasible approach for adaptive video communication. Many advanced video compression standards incorporate scalability profiles, such as H.264 scalability extension [47], and MPEG-4 FGS [48]. They code original video into one base layer (BL) and one or several enhancement layers (ELs). ELs gradually improve the reconstruction quality of BL. Since ELs can be adaptively dropped to meet different requirements, adaptation schemes for scalable coded video can be designed more flexibly and efficiently than those for non-scalable coded video. For example, SNR resolution and temporal resolution can be reduced separately by discarding their corresponding ELs. A tradeoff between them can also be obtained to achieve an optimized perceptual quality.

Adaptation for scalable coded video stream is usually divided into two major levels: scene level and frame level. Whereas, macroblock level and object level adaptation might be necessary for finer adjustment or video objects. In scene level, the target bit budget for each adaptation unit, for instance video scene, is estimated considering various resource constraints. In frame level, a bit allocation scheme with transcoding quantization parameters or selected stream segments is determined under the scene level bit budget constraint. With the derived bit allocation schemes, a new representation of the video stream can be generated.

Effective video adaptation schemes should adapt video scenes with different distortion scales in accordance with their distinct properties. For example, a stationary background or a less user-interested scene should be adapted with coarse resolution so that more bits can be saved for foreground or user-interested scenes. Therefore, it is necessary to analyze the stream, which consists of a sequence of video scenes, so that the bits can be estimated and distributed properly to improve the perceptual quality. In addition, to maintain the QoS, we should consistently monitor the transmitter and receiver buffer. When buffer starts to overflow or underflow, the QoS will be degraded due to data lost or data freezing. As a video streaming system may potentially support a large number of users at the same time, the efficiency of the video adaptation scheme is another important concern. In other words, the computational complexity of the adaptation should be sufficiently low in order to provide excellent scalability.

Existing video adaptation methods can be generalized, although not exclusively, as the following four categories [40]: semantic-based adaptation, transcoding adaptation, temporal condensation, and structure level adaptation. Nevertheless, various shortcomings restrict their wide deployment in practical systems. For example, extensive semantic analysis [49][50] or signal-level transcoding [51][52] is computational complex and does not scale well to accommodate a large number of concurrent users. When most of the temporal condensation methods [53][54] estimate bit budget or skip frames, they emphasize mainly on the stability of the buffer, while regardless of the content characteristics. Consequently the frequent fluctuation of bit budget between adjacent frames will lead to serious temporal visual degradation. It is also important to note that the overall distortion of the stream should include the distortion of skipped frames. This is nevertheless often overlooked in many research papers. Many structure level adaptation methods [55][56] select a number of key frames or bit-planes to meet the bandwidth or buffer constraints. They, however, either fail to consider SNR and temporal scalability together, or do not provide good SNR-temporal compromise with fine granularity rate control. Moreover, most of the existing video adaptation methods only consider a single quality concern, either quality fluctuation [57] or quality distortion [58][59]. Since human visual system [60] is sensitive to both quality criteria, they should be optimized simultaneously and an excellent tradeoff between them needs to be achieved.

Aforementioned multi-optimization process is not easy due to the intricate relations among diverse spaces such as objective, resource, and stream characteristics. The high dimensionality of each space makes the process even more complex. Therefore the representations of these relations should be chosen carefully to improve the maneuverability and the efficiency of video adaptation. Generally, a constrained optimization problem can be formulated on the multi-dimensional table consisting of those representations. Joint optimization with efficient algorithms is a challenging task due to the complex relations across dimensions. The complexity is further exacerbated by the interactions between adaptation subsystem and other subsystems within the adaptation framework.

The main motivation of this research is to investigate an effective video adaptation scheme for MPEG-4 FGS video streaming. As a suitable method for adaptive video communication, MPEG-4 FGS provides high flexibility and scalable rate adjustment capability. Its encoder generates a BL at a low bit rate and an EL with both SNR fine granular scalability (SNR FGS) and temporal fine granular scalability (FGST). Under environment constraints, we need to transmit selected segments of the EL, which constitute the transmission path, together with BL so that the user-perceptual quality of the adapted video stream is optimized.

Our work starts with proposing an adaptation concept model for the scenario shown in Figure 2-III where a streaming server connects to a number of adaptation nodes or proxies and each node caters for a reservoir of non-overlapping clients. We then obtain the target bit budget for each video scene by defining an evaluation metric "fairness index" based on scene characteristics, and maximizing it under both bandwidth and buffer constraints. These bit counts are further distributed among frames, and layers (bit-planes) of each scene to provide optimized perceptual quality. The specific properties of the adaptation scheme include: 1) buffer controller is incorporated to prevent buffer overflow and underflow; 2) scene characteristics, such as content priority, texture and motional complexity, are considered to protect important or complex video scenes without incurring extensive semantic analysis; 3) an excellent SNR-temporal compromise is selected to achieve an excellent tradeoff between two optimization objectives (minimum overall quality distortion and quality fluctuation).

The remainder of this chapter is organized as follows. Section 4.2 illustrates the video adaptation concept model and Section 4.3 provides an overview of the video adaptation problem. Section 4.4 proposes the scene level optimization model and the frame level optimization model is addressed in Section 4.5. Section 4.6 evaluates the performance of proposed video adaptation scheme through simulations and analysis. Finally, Section 4.7 gives some concluding remarks on this chapter.

4.2 Video Adaptation Concept Model

In this section, we provide an overview of proposed video adaptation concept model. This model is used to abstract the practical system to facilitate our research on its individual components. Generally, a flexible video adaptation framework consists of the following three elements: (1) source server, (2) adaptation node, and (3) client. Source server stores a reservoir of videos and transmits requested videos to its clients. Adaptation nodes reside on the path between the server and clients, and adapt received streams with different scales. These adapted streams, which are composed of selected video segments, are then forwarded to the clients to satisfy their distinct requirements. The clients are responsible to decode received video segments for presentation.

It is worth noting that adaptation operations can also be performed on the source server. In that case, adaptation node becomes an integrated module of source server. To elaborate the video adaptation process clearly, we illustrate the concept model of adaptation node in Figure 4-I and identify several key components as follows:

- Scalable coded video stream, which is encoded by MPEG-4 FGS codec into BL, SNR FGS layer, and FGST layer.
- Video adaptor, where the video adaptation problem is formulated and then solved by constantly monitoring environment update and analyzing stream content.
- Communication channels, through which control messages are exchanged and adapted video stream is transmitted.

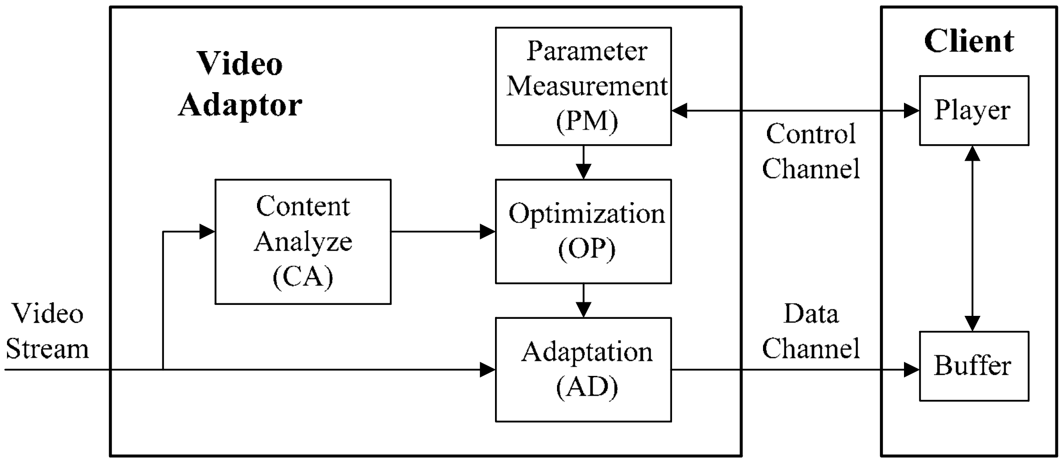


Figure 4-I: Video adaptation concept model

Video adaptor continuously predicts available bandwidth using TCP-Friendly Rate Control (TFRC) protocol and communicates control messages, such as QoS adjustment and buffer utilization information, with client through Real-time Control Protocol (RTCP). Four major component modules of video adaptor are shown in Figure 4-I and their functionalities are explained as follows:

- Content Analyzer (CA) module is responsible for directly extracting from video stream the visual and encoder-specific features such as motion vectors, frame types, and so on.
- Parameter Measurement (PM) module estimates the available bandwidth and derives the specific QoS requirement based on client feedback through control channel [61].
- Optimization (OP) module utilizes the information obtained from CA and PM modules to formulate adaptation problem and selects proper video segments to compose the transmission path.
- Adaptation (AD) module adapts the video stream according to the derived transmission path and transmits it to the client through data channel.

To be concise, the concept model illustrates the situation where the source server transmits a video stream to a single client through one adaptation node. However, this model directly fits the situation where several adaptation nodes connect to the source server and each adaptation node caters for non-overlapping client reservoirs.

4.3 General Problem Description

Within the concept model proposed in Section 4.2, we focus on describing the OP module where video adaptation problem is formulated and transmission path is obtained. In addition to the straight forward stream reconstruction method employed by AD module, many existing technologies have also been proposed for CA module and PM module, such as [61][62][63]. These technologies, which can be directly utilized in the proposed video adaptation model, provide reasonable precise and enough information for OP module with neglectable computational complexity. The video adaptation problem that OP module is responsible to solve can be stated as follows:

- Given a set of resource constraints and a video sequence encoded by MPEG-4 FGS codec, we need to decide which part of the sequence should be transmitted, so that the perceptual quality of the transmitted video is optimized.

We first identify the video stream as a sequence of video scenes and then abstract the whole problem into two sub-problems. These problems are finally solved with two hierarchical models: (1) scene level model (SLM), (2) frame level model (FLM).

Scene level model aims at providing fair bit budget to each video scene based on its content characteristic. Proper bits are assigned to easy scenes, and more bits are allocated to important scenes, or complex scenes that contain detailed texture or fast movements. Mathematical expression for the objective "fairness index" is defined and then optimized using scene level model.

Frame level model concerns selecting an optimized transmission path, which consists of a number of B frames and several bit-planes for each I, P frame for every video scene. Under the bit budget constraint imposed by scene level model, an excellent tradeoff between two optimization objectives: minimum overall quality distortion and quality fluctuation, is achieved by an efficient algorithm. The two objectives are respectively evaluated by two metrics: average quality and quality variance.

4.4 Content-Aware Scene Level Model

In this section, we obtain the target bit budgets of video scenes based on their combinational content characteristics including content priority, texture detail, and motional complexity.

4.4.1 Scene level optimization model

In practice, a video stream usually contains a sequence of video scenes with distinct content characteristics. For example, some scenes contain fast moving objects while others comprise much texture details. Some scenes are user-interested and others are less important. CA module of the adaptation framework extracts the content characteristics and feeds them into OP module as parameters. For example, scene priority, which is assigned during creation, is extracted from the MPEG-4 video stream and normalized into region [0, 1]. Parameters used in this section are listed in Table 4-I.

Table 4-I: SLM parameters

Notation	**Description**
$R_{g,i}$	Bit count of frame *i* in scene *g*
$H_{g,i}$	Bit count for non-texture information of frame *i* in scene *g*
$MV_{g,i}$	Bit count for motion vectors of frame *i* in scene *g*
I_g	Normalized scene priority (between 0 to 1)
N_g	Number of frames in scene *g*
β_g	Estimated residual data size in client buffer after scene *g* is transmitted by the proxy
R	Average available bandwidth
T_g	Target bit budget for scene *g*
$F_g(T_g)$	Fairness index of scene *g* with assigned target bit budget T_g
O_f	Output frame rate of client buffer
β_{max}	Client buffer capacity
$C_{g,i}$	Complexity of frame *i* in scene *g*
Ω	Set of video scenes

With aforementioned factors, we propose a normalized fairness index $F_g \in [0,1]$ to quantify the target bit budget allocation scheme. F_g illustrates the allocation fairness of a specific scene *g* when assigned with target bit budget T_g. Intuitively, the fairness index should be in direct proportion to T_g until reaching the upper limit 1. In this chapter, we hereby define F_g in (3-27):

$$F_g(T_g) = \min\left(\frac{T_g \cdot \sum_g (\sum_{i=1}^{N_g} C_{g,i} + I_g)}{T_\Omega \cdot (\sum_{i=1}^{N_g} C_{g,i} + I_g)},\ 1\right) \tag{3-27}$$

where frame complexity $C_{g,i}$ is taken as the texture and motional complexity. We express $C_{g,i}$ in (3-28):

$$C_{g,i} = R_{g,i} - H_{g,i} + MV_{g,i} \tag{3-28}$$

We determine the total target bit budget T_Ω for the scene set Ω considering both client buffer and bandwidth constraints. Due to the constant frame consuming rate at client side, we take account of the frame number within each video scene to prevent client buffer overflow. Hence, T_Ω can be estimated in (3-29). The third term of the right side denotes the bit count that client buffer has consumed during the entire adaptation process.

$$T_\Omega = \beta_{\max} - \beta_0 + (R / O_f) \cdot \sum_{g\in\Omega} N_g \tag{3-29}$$

Given the total bit budget for video scene set Ω (T_Ω), we aim at maximizing the overall "fairness index" by carefully distributing the budget to each scene (T_g) under client buffer constraint. The scene level model (SLM) is formally expressed as:

SLM: $\underset{T_g}{\text{Maximize}} \sum_g F_g(T_g)$

Subject to: $0 \le \sum_g T_g \le T_\Omega$, $0 \le \beta_g \le \beta_{\max}$, $g \in \Omega$

4.4.2 Adaptation algorithm for video scenes

We first derive the optimized bit budget distribution scheme, consisting of a list of T_g, for the unconstrained **SLM** problem, and then gradually adjust the scheme considering the constraints. Detailed procedures are illustrated in Figure 4-II. We initialize the parameters in step 1 and calculate the scene complexity in step 2. After obtaining the total bit budget in step 3, we derive the optimized bit budget distribution scheme in step 4 and step 5 without considering any constraint. In step 6 and step 7 we estimate the client buffer conditions after each scene is transmitted. If the buffer constraint is violated for any video scene, we will adjust its corresponding bit budget in step 8 and step 9. After all the bit budgets are verified, the optimized bit budget distribution scheme for **SLM** is obtained in step 10. The algorithm complexity is $O(N_\Omega)$ where N_Ω is the number of scenes in the set Ω.

```
Algorithm SceneBitAllocation($R,\Omega$)
1.  Initialize $\beta_0, I_g, R_{g,i}, H_{g,i}, MV_{g,i}, O_f, \Delta = 0$
2.  $C_{g,i} = R_{g,i} - H_{g,i} + MV_{g,i}$
3.  $T_\Omega = \beta_{\max} - \beta_0 + (R/O_f)\cdot\sum_{g\in\Omega} N_g$
4.  for (each scene $g$ in set $\Omega$)
5.      $T_g = \frac{T_\Omega \cdot (\sum_i C_{g,i} + I_g)}{\sum_{g\in\Omega}(\sum_i C_{g,i} + I_g)}$
6.  for (each scene $g$ in set $\Omega$)
7.      $\beta_g = \beta_{g-1} + T_g - (R/O_f)\cdot N_g + \Delta$
8.      $\Delta = \max(\beta_g - \beta_{\max}, 0) + \min(\beta_g, 0)$
9.      $T_g = T_g - \Delta$
10. $\mathbf{T} = \{T_g,\ g \in \Omega\}$ is the target bit budget
```

Figure 4-II: Video adaptation algorithm for SLM

4.5 SNR-Temporal Resolution Optimized Frame Level Model

In this section, we derive transmission paths for the video scenes such that the qualities of these adapted video scenes are optimized.

4.5.1 Frame level multi-objective optimization model

Transmission path for a specific video scene contains a number of selected B frames and several bit-planes for each I, P frame in order to provide both temporal and SNR resolutions. Since human visual system is sensitive to both quality distortion and quality fluctuation, we set up a multi-objective optimization model to achieve the tradeoff between two objectives under bit budget constraint. Table 4-II lists the parameters used in this model.

When it is necessary to reduce the bit size of an encoded video scene which includes a sequence of I, B, and P frames, we can decrease its temporal resolution (frame rate) by skipping B frames, or reduce its SNR resolution by dropping EL data of I, P frames. The BL of I, P frames should be preserved carefully because later P frames are dependent on their previous I or P frames and the lost of one particular I or P frame's BL makes the following P frames useless. Therefore when there is not enough resource to transmit all the EL data which consist of SNR FGS layer (bit-planes) and FGST layer (B frames), a compromise between them should be decided. We use the convention for frame set **S** to indicate frame skipping, i.e. $\mathbf{S}=[S_1,S_2,...,S_{N_2}]$, where $S_i = 0/1$ indicates a skipped/selected B frame among total of N_2 frames. Similarly, the number of bit-planes preserved for I, P frames are assembled as set $\mathbf{L}=[l_1,l_2,...,l_{N_1}]$, where $l_i \in [0,\ l_{max}]$ denotes the number of bit-planes selected for a particular I, P frame. When B frames are skipped, we use frame repetition as the frame rate up-conversion scheme to preserve the original frame rate at the decoder side.

Table 4-II: FLM parameters

Notation	**Description**
l_i	Number of bit-planes chosen for the *i*th I, P frame
b_j	Bit count of B frame *j*
N_1	Total number of B frames
N_2	Total number of I, P frames
MV_i	Motion vector size of frame *i*
ρ_i	Frame priority of B frame *i*
l_{max}	Maximum number of bit-plane encoded for video frames
q_{ij}	Quality of the *j*th I, P frame decoded with base layer to the *i*th bit-plane
b_{ij}	Bit count of the *i*th bit-plane for the *j*th I, P frame
B_s	Residual target bit budget for unprocessed I, P frames
Q_{pre}	Quality of previous I, P, or B frame

Given a transmission path ξ that is composed of set **S** and **L**, the average perceptual quality can be expressed in (3-30):

$$\varphi_1(\xi) = 1/(N_1 + N_2)\{\sum_{i=1}^{N_1} Q_F(l_i) + \sum_{j=1}^{N_2} Q_H(S_j)\} \tag{3-30}$$

where $Q_F(.)$ is the quality of I, P frames and $Q_H(.)$ is the quality of B frames. With each frame's quality Q_i (Q_i=Q_F or Q_i=Q_H), the quality variance is calculated in (3-31):

$$\varphi_2(\xi) = 1/(N_1 + N_2 - 1)\sum_{i=1}^{N_1+N_2} |Q_i - \varphi_1(\xi)|^2 \tag{3-31}$$

Since skipped frames do not cost any bit, the total bit count is expressed in (3-32):

$$R(\xi) = \sum_{i=1}^{N_1} R_F(l_i) + \sum_{j=1}^{N_2} R_H(S_j) \,|\, (S_j = 1) \tag{3-32}$$

where $R_F(.)$ is the bit count of I, P frames and $R_H(.)$ is the bit count of B frames. Under the bit budget constraint, we therefore aim at choosing a transmission path ξ with two optimized objectives: maximum average quality, and minimum quality variance. Formally, the frame level model (FLM) is expressed as:

$$\textbf{FLM}: \ \underset{\xi}{\text{Maximize}} \ \mathbf{\Phi}(\xi) = (\varphi_1(\xi), 1/\varphi_2(\xi))$$

$$\text{Subject to: } 0 \le R(\xi) \le T_g$$

4.5.2 Adaptation algorithm for video frames

As a rule for multi-objective optimization problems, a single global optimal solution does not exist unless the utopia point, which combines the best instances of each objective, happens to be attainable. Generally, the decision-maker needs to express an explicit preference function between the two optimization objectives and turns to optimize one combinational objective. However, the preference function is difficult to obtain in the beginning. An alternative method is to allow the decision-maker to choose from a palette of solutions, which is called Pareto optimal set.

```
Algorithm OptimizedFrameAdaptation($T_g$)
1. $\rho_i = MV_i / B_i$, $\mathbf{\Psi} = \mathbf{\Phi}^P = \varnothing$, $\mathbf{P} = \{\rho_1, \rho_2, ..., \rho_{N_2}\}$
   $l_j = 0$, $\mathbf{L}=\{l_1, l_2, ..., l_{N_1}\}$, $\mathbf{Q}=\{q_{l_1}, q_{l_2 2}, ..., q_{l_{N_1} N_1}\}$
2. while ($\sum_{i\in\mathbf{\Psi}} B_i < T_g$) do
3.     select $i$ where $\rho_i = \max\{\mathbf{P}\}$
4.     $S_i = 1$, $\mathbf{\Psi} = \mathbf{\Psi} \cup i$, $\gamma = T_g - \sum_{i\in\mathbf{\Psi}} B_i$
5.     while ($\gamma > 0$) do
6.         select $k$ where $q_{l_k k} = \min\{\mathbf{Q}\}$
7.         $l_k = \min(l_k + 1, l_{\max})$, $\gamma = \gamma - b_{l_k k}$, update $\mathbf{L}$ and $\mathbf{Q}$
8.     end
9.     for $k = 1$ to $N_1$
10.        $Q_F(k) = q_{l_k k}$
11.    $\xi = \mathbf{S} \cup \mathbf{L}$
12.    for $j = 1$ to $N_2$
13.        if $j \notin \mathbf{\Psi}$, then $Q_H(j) = Q_{pre} - \lambda \cdot \tau^{\overline{MV}}$
14.    $\varphi_1(\xi) = 1/(N_1 + N_2)\{\sum_{j=1}^{N_1} Q_F(j) + \sum_{k=1}^{N_2} Q_H(k)\}$
15.    $\varphi_2(\xi) = 1/(N_1 + N_2 - 1)\sum_{j=1}^{N_1+N_2} |Q_{H/F}(j) - \overline{Q}|^2$
16.    if current video object has slow motion
17.        then $\alpha = 1 - |\mathbf{\Psi}| / N_2$, else $\alpha = 1 + |\mathbf{\Psi}| / N_2$
18.    $\varphi_1'(\xi) = \alpha\varphi_1(\xi)$, $\varphi_2'(\xi) = \alpha / \varphi_2(\xi)$
19.    $\mathbf{\Phi}^P = \mathbf{\Phi}^P \bigcup \{\varphi_1'(\xi), \varphi_2'(\xi)\}$
20. end
21. $\varphi_1^o = \max\{\varphi_1'(\xi)\}$, $\varphi_2^o = \max\{\varphi_2'(\xi)\}$
22. select $\xi^o$ where $\{[\varphi_1'(\xi^o) - \varphi_1^o]^2 + [\varphi_2'(\xi^o) - \varphi_2^o]^2\}^{0.5}$ is minimal
```

Figure 4-III: Video adaptation algorithm for FLM

We handle **FLM** problem by identifying the Pareto optimal set which includes a list of transmission paths with high average quality and low quality variance. From the derived Pareto optimal set, a compromise solution which has the closest Euclidean distance to the utopia point is selected. Figure 4-III illustrates the detailed algorithm.

Firstly, all the B frames are sorted according to their frame size and motional complexity. The fewer bits it occupies or the faster motion it has, the higher priority the B frame is assigned.

Secondly, we gradually select high priority B frames into the transmission path until all the B frames have been selected or the total bit count of selected B frames exceeds the bit budget allocated to the current video scene. Each time a B frame is selected, we distribute the residual bits to the EL of I, P frames to obtain constant quality. The distribution is facilitated by identifying matrices **RD** and **NB** which are composed of rate-distortion (R-D) samples and bit count of I, P bit-planes, respectively. R-D samples of bit-planes and B frames are provided by CA module which extracts them from user-data part of MPEG-4 stream. In matrix **RD**, q_{0j} denotes the quality of frame j when it is decoded with BL only and b_{0j} in matrix **NB** represents its BL bit count. The bit distribution process is to select a number of bit-planes (l_i), from each frame, into the transmission path until residual bits are exhausted. These bit-planes, which constitute the set **L**, are recursively selected using dynamic programming method. In each step, the bit-plane that makes the current quality variance minimal is selected.

$$\mathbf{RD} = \begin{pmatrix} q_{01} & q_{02} & \cdots & q_{0N_1} \\ q_{11} & q_{12} & \cdots & q_{1N_1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nN_1} \end{pmatrix} \qquad \mathbf{NB} = \begin{pmatrix} b_{01} & b_{02} & \cdots & b_{0N_1} \\ b_{11} & b_{12} & \cdots & b_{1N_1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nN_1} \end{pmatrix}$$

Thirdly, after each transmission path ξ, consisting of set **S** and **L**, is formed, we obtain its average quality $\varphi_1(\xi)$ and quality variance $\varphi_2(\xi)$. Expressed in (3-30), $\varphi_1(\xi)$ comprises the quality of truncated I, P frames ($Q_F(l_i)$), selected B frames ($(Q_H(S_j)|(S_j=1)$), and unselected B frames ($(Q_H(S_j)|(S_j=0)$). $Q_F(l_i)$ and $Q_H(S_j)|(S_j=1)$ can be obtained from extracted R-D values, while $Q_H(S_j)|(S_j=0)$ can be estimated based on the quality of previous preserved frame.

Next, preference index α is multiplied to $\varphi_1(\xi)$ and $1/\varphi_2(\xi)$ according to the number of selected B frames in transmission path ξ and the motional complexity of current video scene. For fast moving scenes, α is used to assign higher priority to the transmission paths with more B frames (temporal resolution) to preserve motion smoothness. On the contrary, for stationary scenes, preference is given to the paths with more bit-planes of I, P frames (SNR resolution) to provide better image details. For each transmission path ξ, the adjusted combination of average quality $\varphi_1'(\xi)$ and the inverse of quality variance $\varphi_2'(\xi)$ is inserted into the Pareto optimal set.

Finally, from Pareto optimal set $\mathbf{\Phi}^p$, we select the optimal solution $\{\varphi_1(\xi^o), \varphi_2(\xi^o)\}$ which is as close as possible to the utopia point: $\{\varphi_1^o, \varphi_2^o\}$. The term *close* usually implies that the one minimizes the Euclidean distance **ED**, which is defined in (3-33). In order to be insensitive to the scalar of both objectives φ_1 and φ_2, we transform the original cost functions as in (3-34). This approach is consistently referred to as normalization. In this case, φ_i^{trans} generally has values between zero and one. Given the parameters shown in Table 4-II, the algorithm complexity is $O(N_1 \cdot N_2 \cdot T_g)$.

$$\mathbf{ED} = |\Phi(\Xi) - \Phi^o(\Xi)| = \{\sum_{i=1}^{k}[\varphi_i(\xi) - \varphi_i^o(\xi_i)]^2\}^{0.5} \tag{3-33}$$

$$\varphi_i^{trans} = \frac{\varphi_i - \varphi_i^o}{\varphi_i^{\max} - \varphi_i^o} \tag{3-34}$$

4.6 Performance Evaluation

Extensive simulations are exploited to evaluate the performance of proposed optimized adaptation scheme (OAS), in terms of perceptual quality, adaptation granularity, and computational complexity. The results of OAS are compared with those achieved using three other adaptation schemes: simple frame-dropping (SFD), frame-dropping with drift compensation (FDDC) [64], and re-quantization (RQ). SFD only discards B frames or P frames. Since P frames depend on their preceding I or P frames, to avoid quality drift, if a P frame is discarded, SFD simply discards all subsequent B, P frames until an I frame arrives. FDDC avoids this by performing drift compensation for the subsequent P frames. RQ adapts the stream by rescaling quantizers.

4.6.1 Simulation settings

We cascade three well-known CIF (352x288) video scenes (“Akiyo”, “Container”, “Bus”) of 100 frames each and encode them together as a single MPEG-4 FGS stream with full resolution. These video scenes have distinct characteristics which are identified in Table 4-III. Various network situations are abstracted into two categories: narrow bandwidth (such as dial-up, and wireless access) and broad bandwidth (such as LAN and ADSL). Under these two network situations, we obtain the OAS performance with the simulation settings shown in Table 4-IV. In the following sets of simulations, we employ Peak Signal-to-Noise Ratio (PSNR) as the quality measurement Q to evaluate the two quality metrics: average quality and quality variance. PSNR is calculated based on the three-color components (Y, U, and V) in (3-35) and (3-36), where $m \times n$ is the size of the reconstructed image.

$$PSNR\ (Q_i) = 20 \cdot \log \frac{255}{\sqrt{(4 \cdot MSE_Y + MSE_U + MSE_V)/6}} \tag{3-35}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} | Orignal(i,j) - Reconstruct(i,j) |^2 \tag{3-36}$$

Table 4-III: Content characteristics of CIF video scenes

Scene	Frame No.	Content Characteristic
Akiyo	0-99	Slight movements with detailed texture
Container	100-199	Medium movements only
Bus	200-299	Extensive movements with detailed texture

Table 4-IV: Simulation parameters

Free/Total client buffer size in the beginning of adaptation	2MB/4MB
Relative priority of video scenes	0.2:0.6:0.2
Base layer bit rate	360Kbps
Enhancement layer bit rate	Lossless encoding
Average narrow bandwidth	75KB/s
Average broad bandwidth	1MB/s
Frame rate	30fps

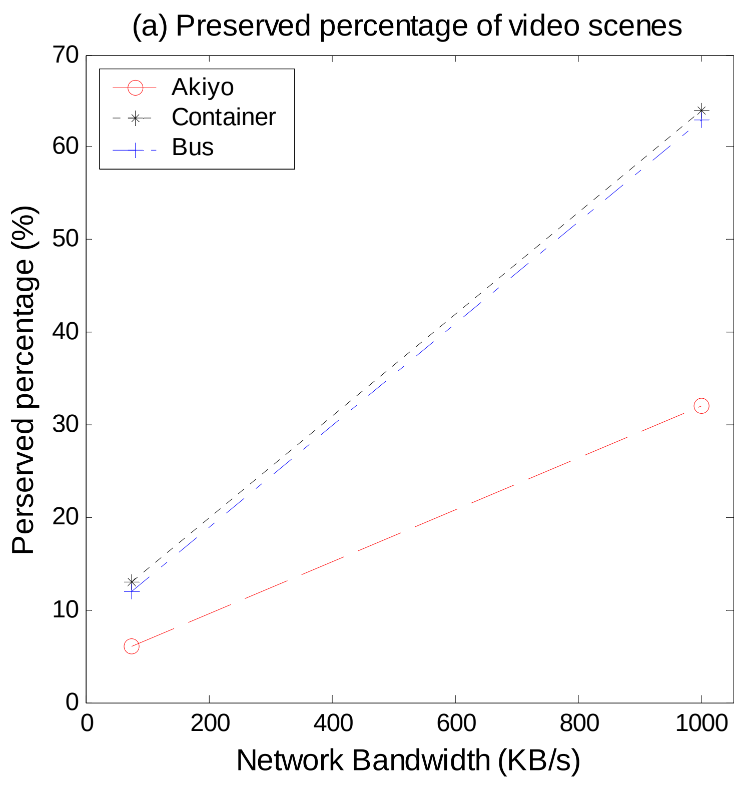
4.6.2 Fair adaptation for video scenes

In this set of simulations, we investigate the performance of scene level algorithm for the three example video scenes. The global scene complexity, including motion vector sizes and texture details, is identified in Table 4-V. Among the example scenes, “Akiyo” contains the least scene complexity, while “Bus” has the most.

With proposed bit allocation scheme, we obtain the data preservation percentage of video scenes and their corresponding fairness indices in Figure 4-IV. It is shown that the preservation percentage is in direct proportion to scene complexity, content priority, and network bandwidth. For example, important scene “Container” and complex scene “Bus” both preserve more percentage of data than “Akiyo” does. In terms of fairness index, we need to sacrifice some complex scenes such as “Bus” in order to prevent client buffer overflow and achieve maximum overall fairness index.

Table 4-V: Motion vector and texture size of video objects

Scene	**Motion Vector Size (pixels)**	**Texture Size (Bytes)**
Akiyo	1924	12481
Container	4245	16199
Bus	6026	86394



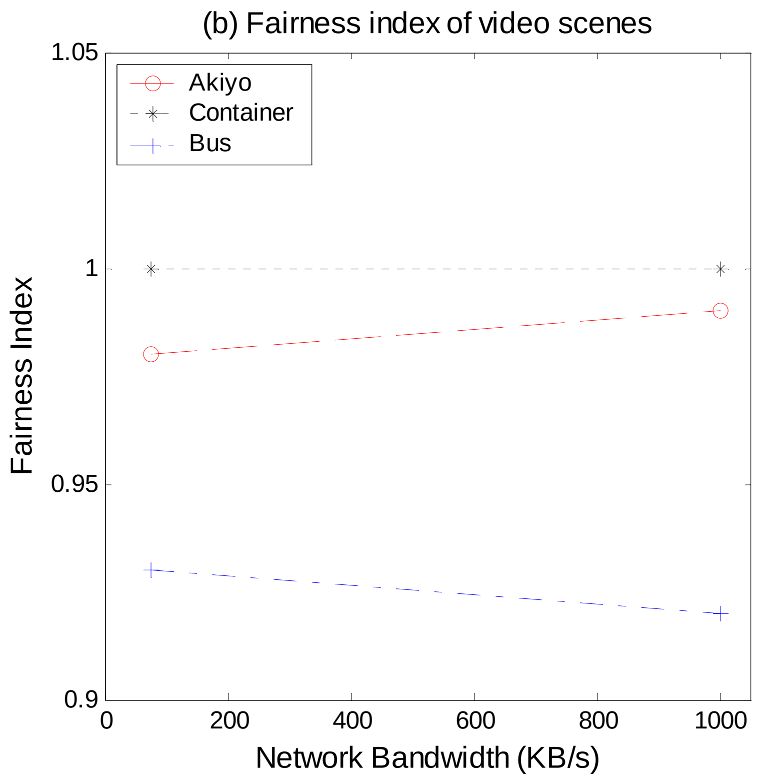
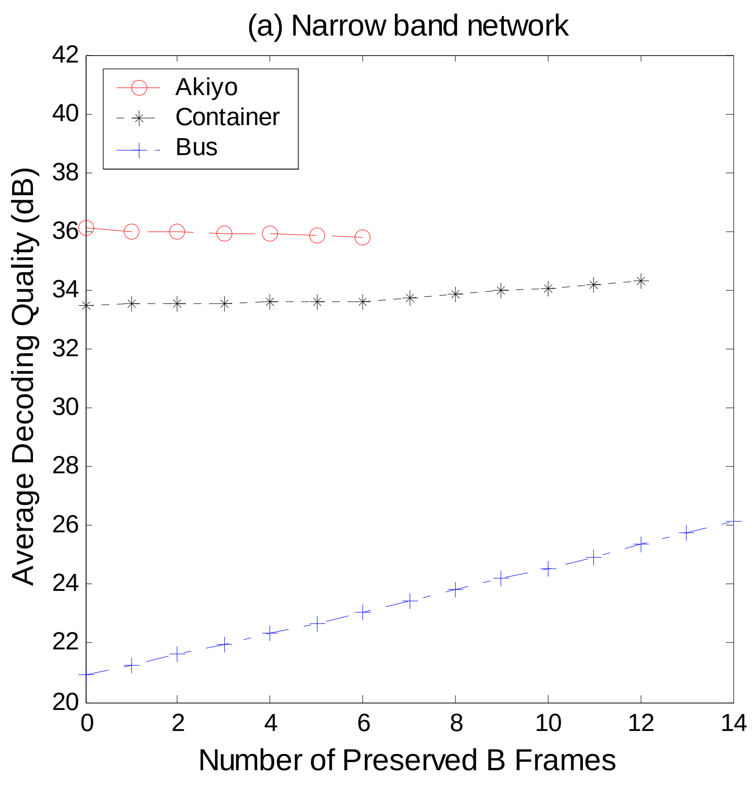


Figure 4-IV: Performance of video adaptation algorithm for scene level model

4.6.3 Effect of preserved temporal resolution

With certain buffer constraint and target bit budget derived from **SLM**, we can obtain many transmission paths for a given video scene. These transmission paths may contain different number of B frames and correspond to distinct perceptual qualities. This set of simulations is to study the effect of selected B frames on the final perceptual quality.

With the settings in Table 4-IV and the target bit budget obtained in Section 4.6.2, we gradually increase the number of selected B frames and derive the transmission paths accordingly. After transmitting the adapted video scenes, we obtain their perceptual qualities at client side. For the three example video scenes, Figure 4-V depicts the relationship between the number of selected B frames and the average perceptual quality under both narrow and broad bandwidth network conditions. Although fluctuation exists, Figure 4-V illustrates the phenomenon that video scenes with slow motions, such as “Akiyo”, tend to show better quality with fewer number of selected B frames, while fast moving scenes, such as “Bus”, has the opposite property. This observation is utilized by the proposed adaptation scheme to determine the tradeoff between two optimization objectives.



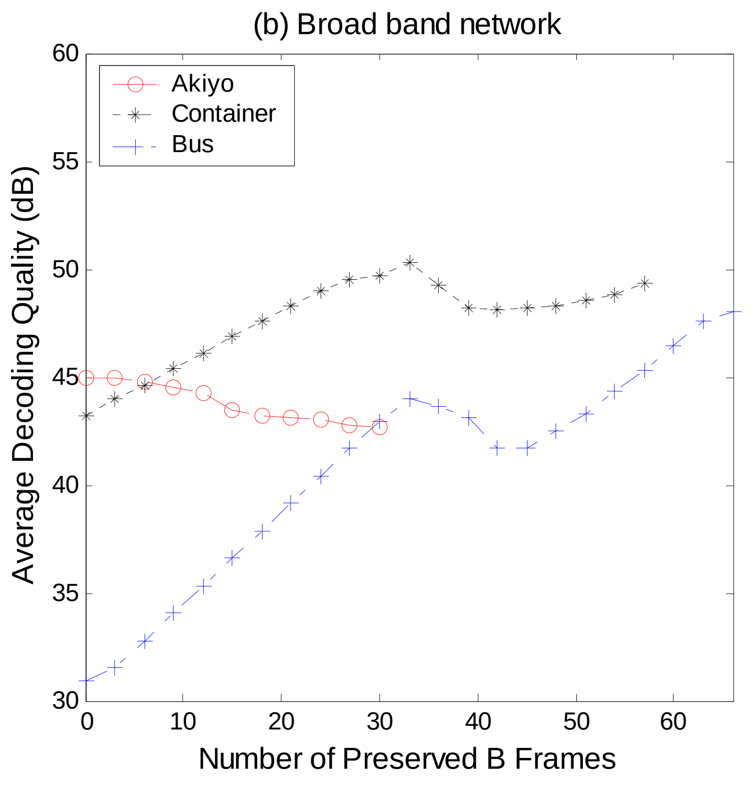
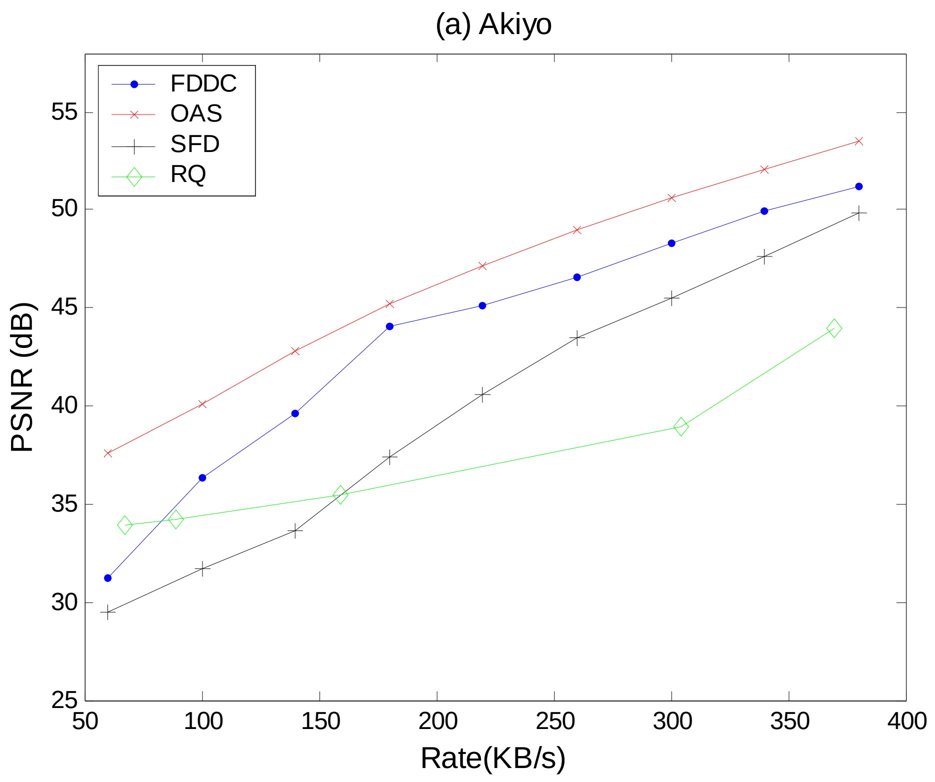


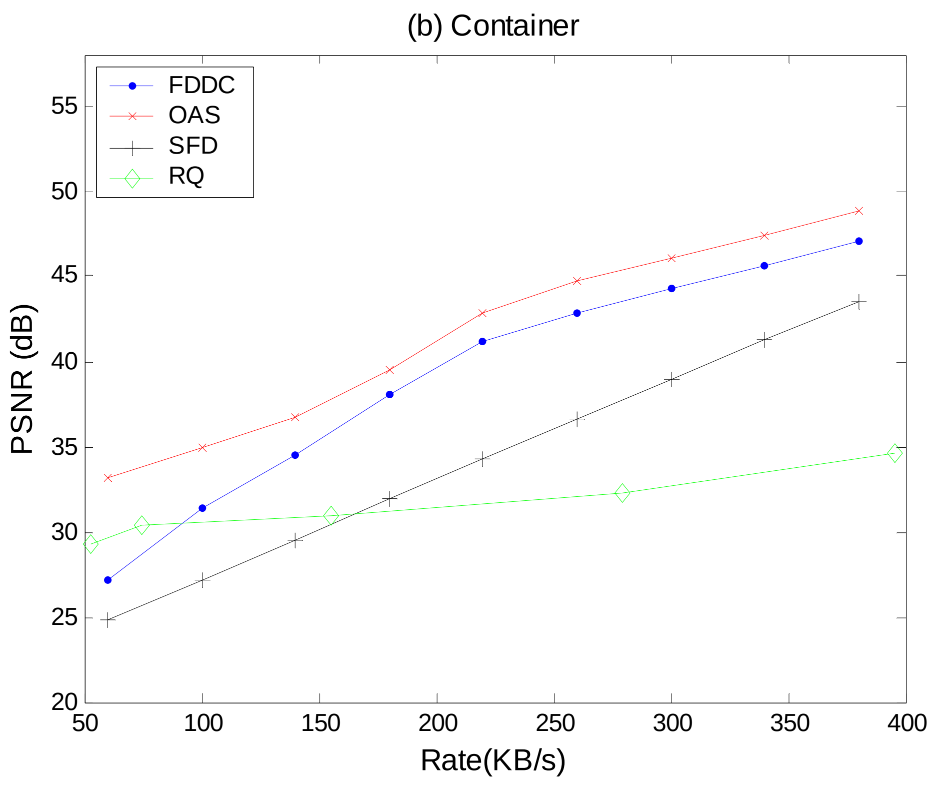
Figure 4-V: Effect of preserved B frames on the average perceptual quality

4.6.4 Perceptual quality improvement

Compared to three classic adaptation schemes: SFD, FDDC, RQ, we examine the performance of proposed OAS in terms of two evaluation metrics: average quality and quality variance.

In this section, we gradually increase the available network bandwidth for the cascaded MPEG-4 FGS stream. Under each sample bandwidth, the adapted video streams are obtained using aforementioned adaptation schemes (OAS, SFD, FDDC, and RQ) accordingly. We calculate the quality criteria at the client side and depict them in Figure 4-VI and Figure 4-VII.





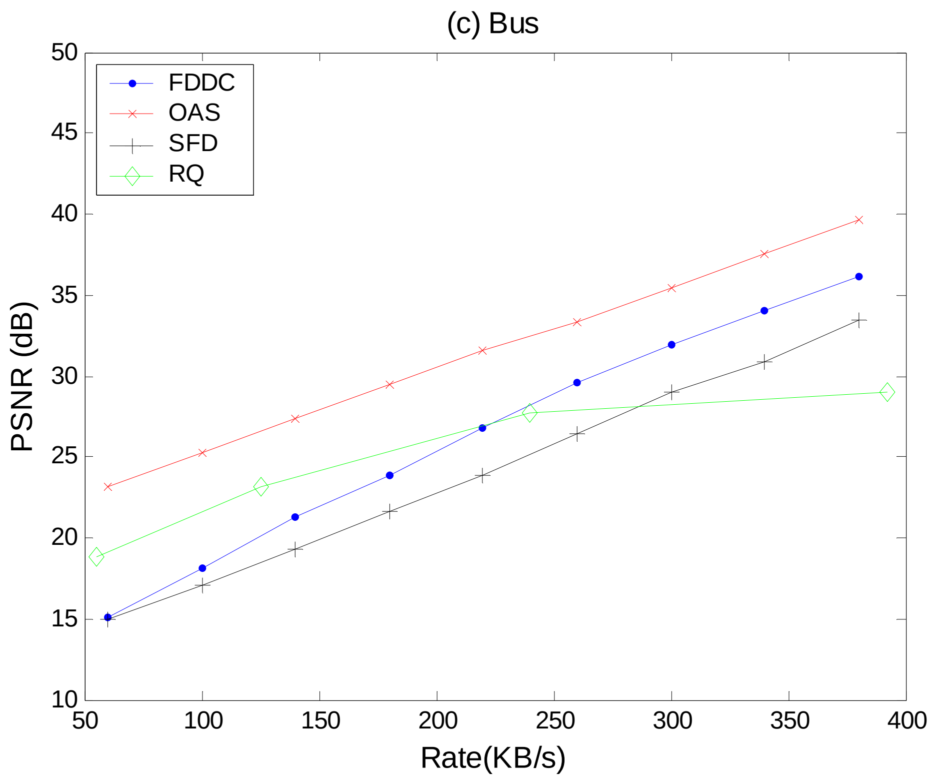
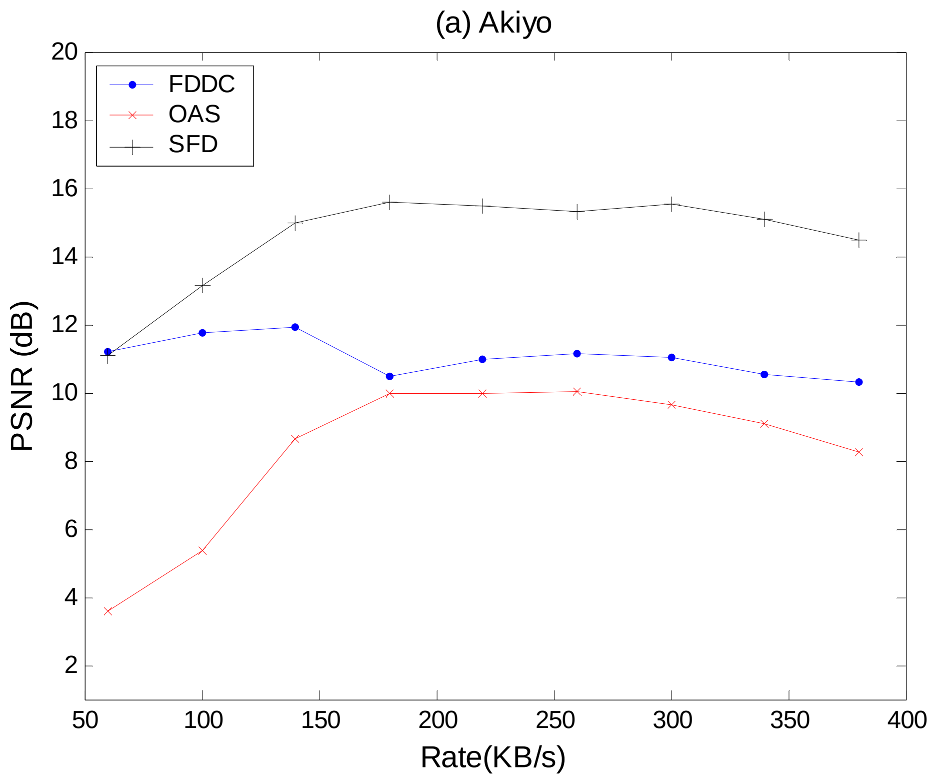


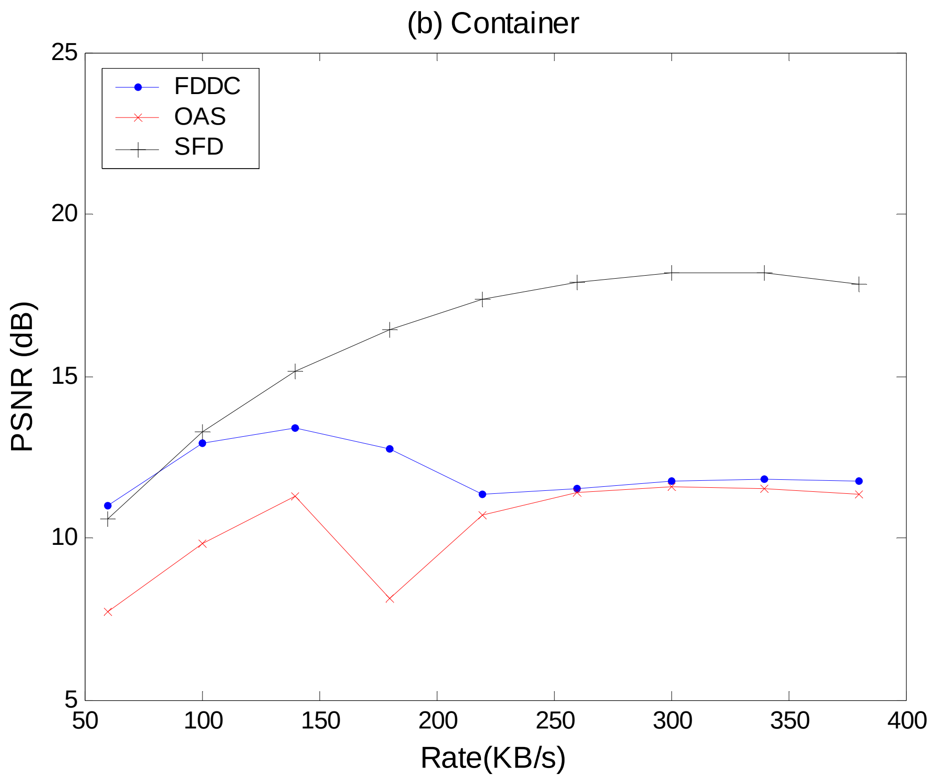
Figure 4-VI: Comparison of average quality among adaptation schemes

The following phenomena can be observed from Figure 4-VI:

- OAS achieves the highest average quality among the adaptation schemes.
- The obtainable quality margin between OAS and FDDC or SFD grows with the decrease of network bandwidth.

When faced with fewer target bit budget incurred by decreased bandwidth, FDDC and SFD both have to drop more continuous B and even P frames, which brings large quality distortion. Nevertheless, instead of dropping frames, OAS steadily drops bit-planes to avoid skipping sequential frames. With preserved reference frames nearby, skipped frame in optimized transmission path obtained by OAS can be reconstructed with good quality. By tuning the number of preserved bit-planes for each frame, OAS also avoids abrupt quality fluctuation.


(a) Akiyo
FDDC
OAS
SFD
PSNR (dB)
Rate(KB/s)
20
18
16
14
12
10
8
6
4
2
50
100
150
200
250
300
350
400


(b) Container
FDDC
OAS
SFD
PSNR (dB)
Rate(KB/s)
25
20
15
10
5
50
100
150
200
250
300
350
400

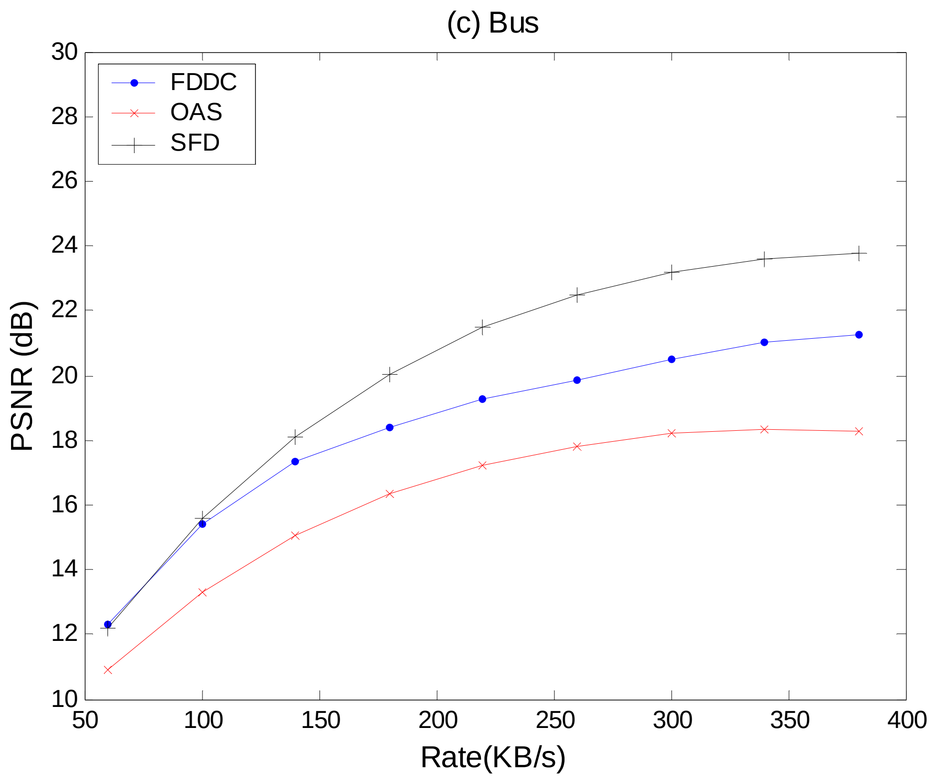


Figure 4-VII: Comparison of quality variance among adaptation schemes

Additionally, the following observations are illustrated in Figure 4-VII:

- OAS attains a lower quality variance compared to other adaptation schemes.
- With the decrease of available bandwidth, the performance improvement obtained by OAS in terms of quality variance is more significant.

The reasons behind these observations are listed as follows:

- Although quality fluctuation is crucial to the subjective quality assessment, most of the previous researches in this area, including FDDC and SFD, completely ignore it and provide relatively high quality variance.
- Under narrow network bandwidth situation, instead of dropping excessive

frames, OAS can manipulate bit-planes, instead of skipping frames, to reduce the quality variance. The large quality variance is usually brought by the imprecise reconstruction of skipped frames.

4.6.5 Adaptation granularity comparison

We further investigate the granularity of adaptation schemes which illustrates the rate match accuracy between target bit rate and adapted bit rate. The discrepancies between the above two rates for three adaptation schemes (OAS, SFD, FDDC) are compared in Figure 4-VIII. OAS achieves the most matched bit rate (more than 85% match to the target bit rate) because it operates on the bit-planes while others only control individual frames. With the reasonable rate match accuracy, OAS achieves a good tradeoff between the utilization of network resource and the toleration of imperfect environment information estimation, such as network bandwidth estimation explored in [65].

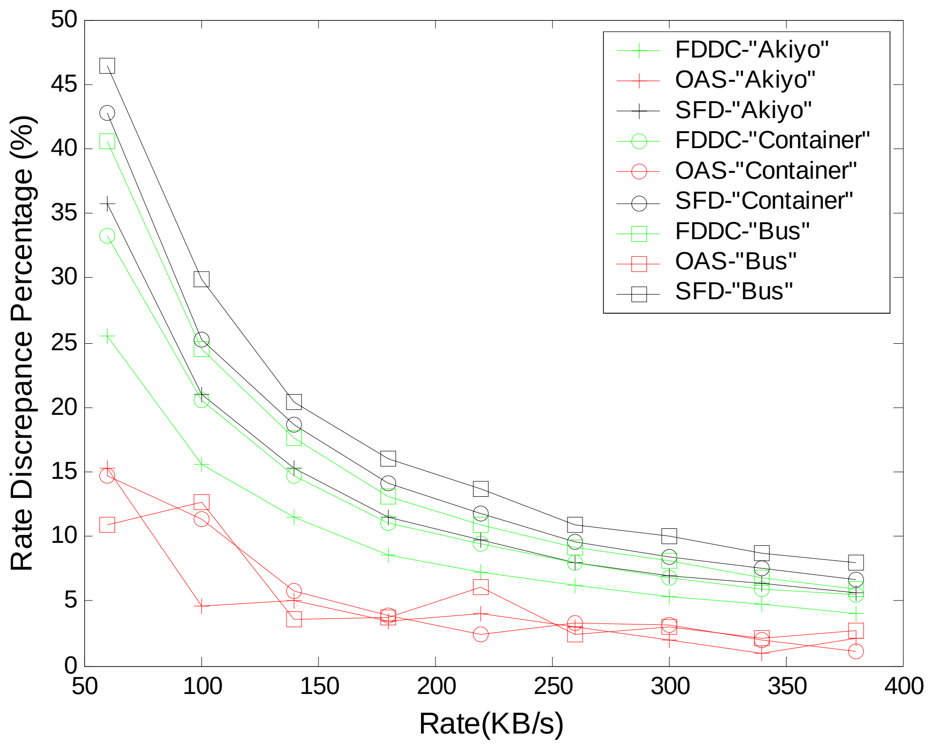


Figure 4-VIII: Comparison of bit rate match discrepancy among adaptation schemes

4.6.6 Computational efficiency comparison

Finally, another important practical concern of the video adaptation scheme is its computational efficiency. For OAS, the most computational demanding part is to choose the smallest value from the set **Q** and then update **L** and **Q** accordingly. For the example video scenes, this step involves about 34 comparison and 10 times adding. In the worst case, it should perform 34×6×33 operations (adding or comparison) for each search. By considering all the computation needed for the optimization module, the overall complexity is about $4*10^5$ operations to derive the optimized solution. For the sake of comparison, we examine three other adaptation schemes that have been discussed above: SFD, FDDC, and RQ. For each scheme, we repeat the adaptation operation for 5 times on the cascaded sequence with target rates distributed between 120KB/s and 1.2MB/s. For OAS, it takes about 3*ms* for each scene which is more than one order of magnitude lower than that of FDDC or RQ. Figure 4-IX depicts the average computation time of these schemes on our simulation server. It is shown that OAS has very low computational complexity which is comparable to SFD and much lower than FDDC and RQ.

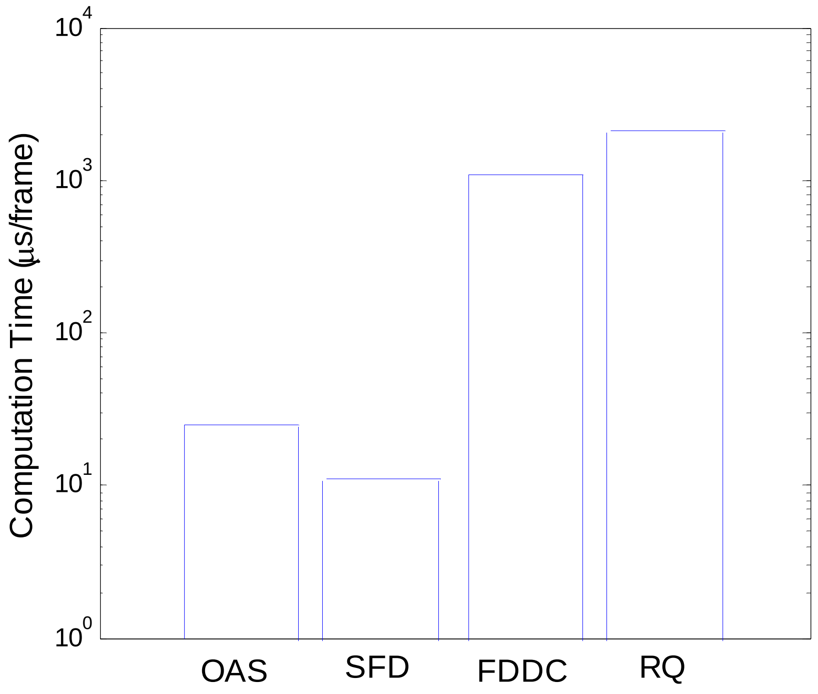


Figure 4-IX: Comparison of average computation time per frame among adaptation schemes

4.7 Concluding Remarks

This chapter investigates the problem of adapting MPEG-4 FGS encoded video stream to satisfy heterogeneous client requirements and network conditions. Within the video adaptation concept model, we propose an optimized adaptation scheme to generate a new representation of original video stream so that perceptual qualities at clients are improved. By taking account of content priority and content characteristics, we present a multi-objective optimization model and put forward heuristic algorithms to obtain the adaptation scheme with reduced computational complexity. The simulation results demonstrate the effectiveness and superiority of proposed video adaptation scheme.

Chapter 5: OPTIMIZED CACHE MANAGEMENT

This chapter proposes an optimized caching strategy (OCS) and a scalable transmission scheme (STS) for scalable coded video streaming. By exploring the characteristics of video streaming workload and system design objectives, OCS and STS work together efficiently to minimize both network bandwidth cost and user access latency. Firstly, we analyze the caching problem for proxy-assisted video streaming system and derive a maneuverable caching scenario. Secondly, we develop an efficient transmission scheme for scalable coded videos. Thirdly, we formulate a multi-objective optimization model with closed-form expressions to obtain the optimized caching strategy. Finally, with designed algorithms, an excellent compromise between two competing objectives (minimizing bandwidth cost and access latency) is achieved. In summary, compared with other video caching strategies and transmission schemes, OCS and STS achieves a significant reduction in network bandwidth cost with even a small proxy cache size.

5.1 Background

The fast development of network video streaming applications, such as live video broadcasting, video-on-demand (VOD) system, and distance education, is requesting for more and more resources. These resources, including network bandwidth, I/O speed, server computation and storage capacity, currently are not sufficient to support wide deployment of those applications due to their inherited characteristics. Some of these characteristics are identified as follows:

- The streaming sessions of these video streaming applications have rather long durations which may be up to several hours.
- Users usually request for the streams with relatively high bit rates, such as several hundred kbps, to obtain satisfied video quality.
- Huge number of users may concurrently exist in the streaming system.
- The QoS requirements and user capacities are heterogeneous.

Lack of sufficient resources will lead to degraded service quality such as long access latency, playback jitter, and low image quality. Currently, many approaches have been proposed to improve the efficiency of video streaming systems by managing the critical resources. These methods include CPU scheduling algorithms, disk management schemes, caching strategies, and data placement strategies. Among them, existing researches [31][66] on video streaming systems have shown proxy caching to be extremely important to reduce system bottlenecks and improve the scalability, from both network and source server perspectives. In a proxy-assisted video streaming system, the server normally has sufficient storage capacity. Each client connects to the server through proxies. The network bandwidth between proxy and clients is considered to be inexpensive, compared with the backbone bandwidth between the server and proxy. By utilizing the resources available on the proxies, proxy caching can significantly reduce the demand for the backbone network and server resources.

Proxy caching has already been proven to be quite effective to improve system scalability, for delivering web objects, such as hypertext and images. These web objects are entirely cached in a web proxy close to clients so that clients can retrieve them without contacting with the web server. In this way, the backbone bandwidth consumption decreases and remote web server traffic is off-loaded. Nevertheless, simply reusing the concepts and methods from traditional Internet Web caching does not suit the special characteristics of video streaming because proxy caching for video streaming presents several additional challenges which are listed as follows:

- Compared with the small size of web objects, the size of a video object is usually orders of magnitudes larger. For example, a two hours long MPEG-4 encoded video has about 1 GB size, while a web image is normally of the magnitude of 10KB. Therefore, caching an entire video object in a proxy is unrealistic considering the huge number of video objects on the server.
- Compared with traditional web page, the demand of continuous and timely delivery for video streaming is more rigorous. Therefore, a lot of resources have to be reserved to maintain the QoS during the long playback duration.

Due to the limited disk space, proxy can only store frequently accessed video data, which is a subset of video objects on the server. These partial caching methods are generally classified as two types: prefix caching [67][68] and segment caching [69][70][71]. Prefix caching stores only the first part (prefix) of the popular video objects and is very efficient in reducing the access latency perceived by clients. Segment caching saves video objects in segments and shows superiority for interactive video streaming. The challenge of both types of caching methods lies in the determination of the cached video portions, considering various environment parameters such as video popularity distribution and video bit rates.

Proxy can also choose to dynamically cache video objects in response to the changes of environments by intercepting the ongoing video streams from server to clients. In [72][73][74], various dynamic replacement methods are examined, which are robust for the evolving client workloads. Video segments are cached in and ejected from proxy according to various criteria, such as their distance from the beginning of the video objects [72], and how recently and frequently they have been requested [73]. Proper replacement methods are used to achieve a high byte-hit-ratio when video popularity changes frequently. Nevertheless, there are several problems of replacement methods which are identified as follows:

- Frequent caching and evicting videos brings a high overhead in time and I/O bandwidth.

- Ongoing streams may prevent the replacement action when the video segments need to be evicted.
- Continuous computation may exhaust proxy computational capacity and create new bottlenecks.
- Much log information, which is used to calculate replacement policy, needs to be maintained and consumes extra storage on the proxy.

In addition to the proxy caching strategy, cost-efficient video delivery is another important issue for Internet video streaming systems. The delivery process usually takes some time to complete and we call it streaming session. Sharing among overlapping sessions has been proved to be resource efficient. To this end, two main transmission approaches are proposed for sharing along the time axis: reactive/client-pull and proactive/server-push schemes. Examples of the former include batching [75][76] and patching [77][78]. In proactive schemes, there are a number of broadcasting schemes such as staggered broadcasting [79], pyramid broadcasting [80], and harmonic broadcasting [81].

Since video transmission needs to proceed at their natural rate to prevent playback jitter, in face of network congestion and client heterogeneity, transmission schemes have to be adaptive by differentiating image quality, instead of delaying the delivery. An elegant way of dealing with heterogeneity and introducing adaptive transmission is to use advanced scalable codecs [15], such as MPEG-4 Fine

Granularity Scalable (FGS) [48] and H.264 scalable extension [47]. These codecs allow dropping video segments in a more controlled way and are effective to deal with the client and network heterogeneity problem.

Nevertheless existing literature lacks detailed combinative analysis of scalable proxy caching and transmission schemes to handle heterogeneity. Moreover, only limited performance metrics and service models were considered [82]. For example, [83][84] mostly focus on optimizing a single video object and are only concerned with the reduction of backbone bandwidth consumption. [85] aims at optimizing only a single design objective: reducing the number of channels. Since in commercial video streaming systems, service providers (SP) and clients always care for different performance metrics, it is important for us to analyze the systems from both aspects. Some researches such as [86] choose to maximize byte hit ratio or quality hit ratio, which are usually used to measure how many percents of bytes requested are hit or how many percents on the quality are satisfied in the proxy. Nevertheless, it is more important for SP to care for the bandwidth cost, and for clients to care for the access latency with negotiated video quality.

Various factors including the possible competing design objectives and heterogeneous clients make the proxy caching and transmission problem complex to model and even more difficult to solve. A feasible method is to develop an approximate analytical model with closed-form expressions by analyzing the system characteristics, and then solve it with computational efficient algorithms.

In this chapter, we consider the hierarchical network architecture for the transmission of scalable coded video, where proxies connect to the server through backbone network and clients connect to proxies through heterogeneous networks. We present an efficient proxy caching with the aid of scalable transmission scheme to stream videos to heterogeneous clients. An original model for scalable proxy caching is proposed and analyzed. The objective of this model is to determine which portions of the video objects should be cached in proxy such that the design objectives are optimized simultaneously. Two design objectives from SP and client aspects are chosen: minimum overall bandwidth cost and minimum access latency. By solving the proposed model, an optimized caching strategy (OCS) is derived, according to the client access patterns to the video objects.

The remainder of this chapter is organized as follows. Section 5.2 provides an overview of the proxy caching problem and Section 5.3 illustrates the proxy caching concept model. Section 5.4 proposes a unified transmission scheme for scalable coded video. The multi-objective optimization model is established, analyzed, and then solved in Section 5.5. Section 5.6 evaluates the performance of proposed optimized caching strategy (OCS) and scalable transmission scheme (STS) through simulations and analysis. Finally, Section 5.7 gives some concluding remarks on this chapter.

5.2 General Problem Description

In this section, we provide an overview of the proxy caching problem and present several considerations and assumptions.

We take the system architecture illustrated in Figure 2-III as the video streaming scenario. A central server stores a repository of video objects which are encoded with full frequency and full size using scalable video codec. A number of proxies are placed between the central server and the clients. Some video objects are partially cached on those proxies. Each proxy responds to the requests of a set of heterogeneous clients which may vary from PC, laptop to PDA and mobile phone. A client negotiates with the connected proxy about its QoS requirement and requests for a particular part of the video object. If the requested portion is cached, the proxy will send it to the client without interacting with the source server. Otherwise, the proxy will forward the request to the source server and relays the requested stream to the clients. In this method, proxy caches save the backbone bandwidth and provide better QoS to clients. Several characteristics of the streaming system are identified as follows:

- We consider a complex network behind the proxy such as a campus network which may include both wired and wireless access clients. These clients are highly heterogeneous in terms of access bandwidths, hardware capacities, and QoS requirements. To reflect such heterogeneity, we classify several classes of clients based on their access bandwidth and QoS requirements. These classes are then mapped to several discrete coding bit rates of video layers to cater for the long-term QoS negotiations between proxy and clients.

- We consider the workload characteristic which is reported by recent analysis of media workload on large scale media streaming servers [87][88][89]: any video segment is equally likely to be accessed for frequently requested objects and large video objects; while access frequency is higher for earlier segments of infrequently requested objects.

In the following paragraphs of the chapter, we only focus on the video streaming system with a single server, a single proxy, and multiple clients. However, our proposed methods and results apply directly to the multiple-proxy Content Distribution Networks (CDN) where the server has unicast connections to multiple proxies, and each proxy serves different non-overlapping set of clients. Therefore, the proxy caching problem can be stated as follows:

- Given the limited storage space on the proxy and specific client access patterns, we need to determine which portion of the video objects should be cached, such that system design objectives: minimum bandwidth cost and minimum access latency, are achieved.

5.3 Proxy Caching Concept Model

In this section, we focus on establishing and analyzing a concept model for the proxy caching problem.

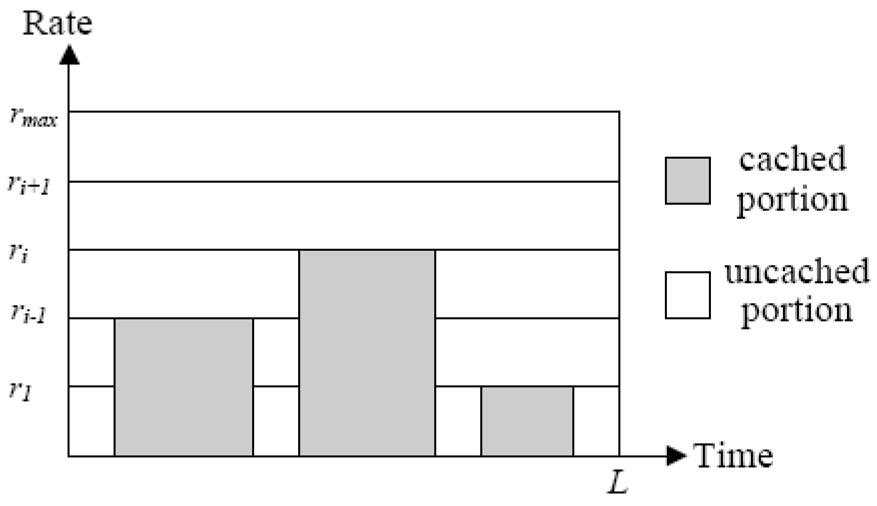


Figure 5-I: Illustration of cached segments of one video object on proxy

Based on the problem described in Section 5.2, a general illustration of cached segments for one video object can be shown in Figure 5-I. This video object is characterized by object length (L) and bit rates of coded layers (r_i). Cached segments may have different start points, lengths, and bit rates. These parameters, combined with other workload and environment variables, need to be related to the final optimization objectives through an analysis model. Nevertheless, it is potentially very difficult to establish the mathematical model and obtain the closed-form expressions for the optimization objectives considering the huge number of parameters. It is reasonable for us to inspect the characteristics of both workload and design objectives, and derive an approximate model to reduce complexity.

Given the storage space on proxy (H), let us consider the following two situations: (1) the proxy choose to cache video segment A which starts at time t_a, with length l_a and bit rate r_a ($l_a \cdot r_a = H$); (2) the proxy choose to cache segment B which starts at time t_b>t_a, with length l_a and bit rate r_a. These two segments occupy the same storage space and differ in the start time ($t_a \neq t_b$). Since existing workload has shown that segment A is generally more popular than segment B, more requests will be instantly served by the proxy without contacting the video server if we choose to cache segment A. Consequently, by caching segment A the average access latency is reduced and the bandwidth cost is saved. Similarly, since the lower layer of video objects are requested more frequently than higher layer, access latency and bandwidth cost will be reduced by caching lower layer segments. Therefore the beginning segment and low layers should be cached with high priority. This conclusion results in the caching scenario for a specific scalable coded video object ω, which is shown in Figure 5-II. Accordingly, the parameters used in proxy concept model are listed in Table 5-I and we assume the requests arrive at proxy according to Poisson process with parameter λ.

Table 5-I: Parameters of proxy caching concept model

Notation	**Description**
Ω	Video object set
H	Proxy cache size
N_Ω	Number of video objects inside set Ω
N_ω	Number of coded layers of video object ω
r_i^ω	Coded bit rate of layer i of video object ω
α_i^ω	Probability that a request's negotiated bit rate is r_i^ω ($\sum_i \alpha_i^\omega = 1$)
λ_ω	Request arrival rate for video object ω
l_i^ω	Cached prefix length for layer i of video object ω
g_i^ω	Patch length for layer i of video object ω
L_ω	Length of video object ω

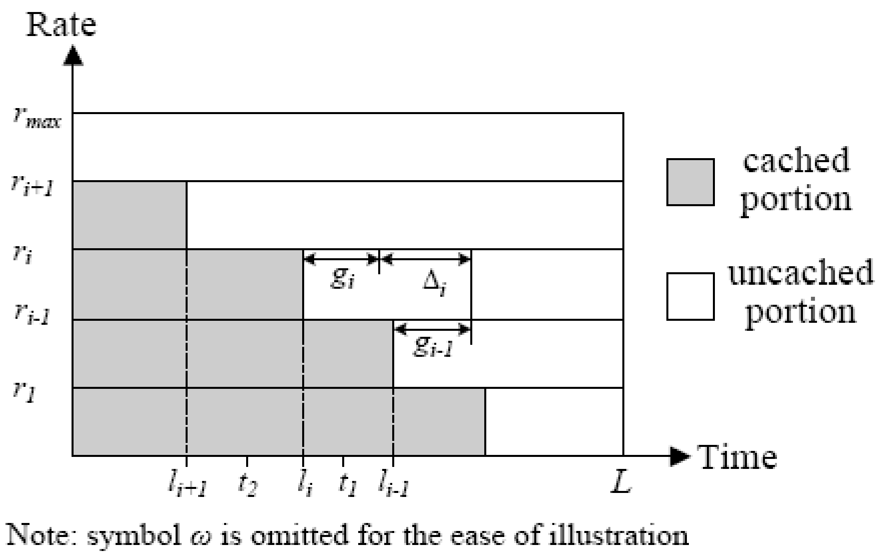

Rate
r_{max}
r_{i+1}
r_i
r_{i-1}
r_1
g_i
Δ_i
g_{i-1}
cached portion
uncached portion
Time
l_{i+1} t_2 l_i t_1 l_{i-1} L
Note: symbol ω is omitted for the ease of illustration

Figure 5-II: Simplified proxy caching scenario for one video object ω

As illustrated in Figure 5-II, for each video object ω, every layer is cached with different length (l_i^{ω}). Those lengths are constrained by: $0 \le l_i^{\omega} \le l_{i-1}^{\omega} \le L_{\omega}$ and form a vector $\mathbf{L}_{\omega}$. All the vectors $\mathbf{L}_{\omega}$ for $\omega \in \Omega$ form a 2-D matrix **L**. Finally, we attempt to solve the following caching problem:

- Given a limited cache size H to the video object set Ω, which portion of the objects, denoted by matrix **L**, should be cached such that both overall bandwidth cost and access latency are minimized.

5.4 Scalable Video Transmission Scheme

In this section, we propose a transmission scheme for scalable coded video objects so that closed-form expressions of design objectives could be obtained in the following section.

In order to propose the scalable transmission scheme (STS), we make the following two assumptions on the first client request to object ω: (1) the arrival time of this request is set to time 0; (2) this request belongs to class i which has the negotiated bit rate of r_i. With the proxy caching scenario in Figure 5-II, we describe the STS for one video object ω in the following paragraphs. It is worth noting that for the ease of description, symbol ω is omitted.

Upon receiving the first request, the proxy delays serving it by d seconds which is the maximum access latency. In the meanwhile, all the requests with similar QoS requirements are batched and served by the proxy with only one stream.

Firstly, we consider those posterior requests from class i. For the requests arriving within time $[0, d]$, the proxy schedules to get the complete suffix (with length $L-l_i$ and bit rate r_i) from the source server and forward to those clients at time l_i+d which is just in time to guarantee the continuous playback at the clients. The requests arriving within time interval $(d, l_i+d]$ will subscribe to this suffix stream while at the same time receiving the prefix sent by the proxy. No new suffix transmission is needed. For the posterior requests arriving at time t_1 within time interval $(l_i+d, l_i+d+g_i]$, the proxy will forward them the incoming suffix and schedule to send a suffix patch of duration $[l_i, t_1]$ at time t_1+l_i+d after transferring the prefix. Note that the proxy only sends one prefix stream for all the requests arriving within interval $(j\cdot d, (j+1)\cdot d]$ ($j=1,2,...,N$).

Secondly, we examine the requests from higher classes. Let's take the requests from class i+1 for example. Other requests from higher classes, such as i+2 and i+3, can be similarly served by the proxy. Suppose some requests of class i+1 arrive within time interval $(0, l_{i+1}+d]$, the proxy will transmit the prefix of class i+1 to the new clients within d seconds and schedule to transmit the complete suffix of class i+1 (with bit rate r_{i+1}) at time $l_{i+1}+d$, instead of the pre-scheduled prefix with bit rate r_i.

The proxy will shape this stream down to lower bit rate to satisfy other classes' requests. If a request of class i+1 arrive within time interval $(l_{i+1}+d, l_{i+1}+d+g_{i+1}]$, for example at t_2, the proxy will increase the bit rate of oncoming suffix streaming for class i to r_{i+1}, and forward it to the new clients of class i+1. The missing suffix part for the request of class i+1 can be patched by sending a suffix patch of duration $[l_{i+1}+d, t_2]$ after transferring the prefix.

Finally we consider the requests from lower classes. For example, if some requests of class i–1 arrive within time interval $(0, l_{i-1}+d]$, the proxy just transmits the prefix within d seconds and forwards the incoming stream to the new clients at time $l_{i-1}+d$. If a request of class i–1 arrives within time interval $(l_{i-1}+d, l_{i-1}+d+g_{i-1}]$, a suffix patch of class i–1 will be scheduled after transmitting the prefix. Those requests from lower classes, such as i–2 and i–3, are satisfied similarly. Note that each client will receive at most 2 channels at the same time. The decision to transmit a complete suffix or a suffix patch depends on the patch length g_i.

This scalable transmission scheme encompasses the following schemes: (1) scalable batching (d>0, g_i=0), (2) scalable patching (d=0, g_i>0), (3) scalable batch-patching (d>0, g_i>0); and works with the following caching strategies: (1) 0-1 caching ($l_i=L$ or $l_i=0$), (2) non-scalable caching ($l_i=l_{i+1}$), (3) partial caching ($0<l_i<L$), (4) scalable caching ($l_i \neq l_{i+1}$).

5.5 Optimized Proxy Caching Strategy

In this section, we establish the mathematical model for the proxy caching problem and derive closed-form expressions for the design objectives based on proposed scalable transmission scheme (STS) in Section 5.4. The solution for the optimization problem is finally obtained with heuristic algorithms.

5.5.1 Multi-objective optimization model

To obtain the optimized caching strategy (OCS), we develop a set of equations for the evaluation criteria: overall bandwidth cost averaged per second and average access latency.

Since the requests arrived within interval *d* are always served together, the average access latency can be expressed as: $F_1(H, d) = d/2$. The bandwidth cost composes of the following two parts: backbone bandwidth cost between proxy and source server, and intra-network bandwidth cost between proxy and clients. Considering the different network bandwidth cost per bit c_b and c_p, we express the overall bandwidth cost per second in (3-37):

$$F_2(H,\ d) = \sum_{\omega\in\Omega} f_2(H_\omega,\ d) = \sum_{\omega\in\Omega} (c_b \cdot B_b + c_p \cdot B_p) \tag{3-37}$$

where B_b is the backbone bandwidth consumption (per second), and B_p is the intra-network bandwidth consumption (per second), for a given video object ω.

B_b can be obtained by calculating the transmitted data size within the interval (I_ω) between two complete suffix transmission, and then divide this size by the interval. If the previous complete suffix transmission starts at time 0, the current one should start when some requests for layer i to layer N arrive within duration Δ_i and no request for layer j arrive within Δ_j ($j=i+1$ to N). Δ_i denotes the difference of prefix and patch length between two adjacent layers which is illustrated in Figure 5-II and can be expressed by (3-38):

$$\Delta_i = l_{i\text{-}1} + g_{i\text{-}1} - l_i - g_i,\ (\Delta_1 = \infty) \qquad (3\text{-}38)$$

Since the requests arrive according to Poisson process, the possibility of aforementioned situation where some requests for layer i to layer N arrive within duration Δ_i and no request for layer j arrives within Δ_j can be calculated by the expression in (3-39):

$$p^{ij} = \exp(-\sum_{j=i+1}^{N}\sum_{k=j}^{N}\alpha_k\lambda\Delta_j)\cdot(1-\exp(-\sum_{j=i}^{N}\alpha_j\lambda\Delta_i)) \qquad (3\text{-}39)$$

With above possibility p^{ij}, the complete suffix transmission interval I_ω^i equals to l_i+g_i+d+Δ_i', where Δ_i' is the average waiting time for the arrival of the first request for layer i to layer N within duration Δ_i. Since this waiting time only takes account of those requests arriving within Δ_i, it is usually less than the average waiting time of an unconstrained Poisson process with arrival rate $\sum_{j=i}^{N}\alpha_j\lambda$. We obtain the expression for Δ_i' in (3-40). Furthermore, considering the possibility p^{ij} for each interval (I_ω^i), we can derive the average suffix transmission interval I_ω in (3-41).

$$\Delta_i^{'} = (\sum_{j=i}^{N} \alpha_j \lambda)^{-1} - ((\sum_{j=i}^{N} \alpha_j \lambda) \cdot p_i)^{-1} - \frac{\Delta_i}{p_i}, \ (p_i = \exp(\Delta_i \sum_{j=i}^{N} \alpha_j \lambda)) \tag{3-40}$$

$$(\Delta_1^{'} = 1/\lambda)$$

$$I_\omega = \sum_{i=1}^{N} \{ \prod_{j=i+1}^{N} p_j \cdot (1-p_i) \cdot (l_i + g_i + d + \Delta_i^{'}) \}, \ (p_i = \exp(-\sum_{k=i}^{N} \alpha_k \lambda \Delta_i)) \tag{3-41}$$

The transmitted data within interval I_ω is composed of one complete suffix and several patches. In addition to the complete suffix, the source server needs to transmit the patches to the proxy so that posterior requests within interval I_ω can join the current suffix stream to save bandwidth consumption. Since we do not transmit one complete suffix with full bit rate (r_N) when there is no request for layer N arriving during the interval I_ω, we introduce the vector $\xi = [\xi_1, \xi_2, \ldots, \xi_N]$ where ξ_i denotes the complete suffix size when the maximum requested layer during I_ω is i. ξ_i is obtained by calculating the uncached portion of a complete stream with bit rate r_i and length L in (3-42):

$$\xi_i = L \cdot r_i - \sum_{j=1}^{i} (r_j - r_{j-1}) \cdot l_j \tag{3-42}$$

Since layer i will be requested with probability α_i, the average size of a complete suffix $\overline{\xi}$ can be derived in (3-43):

$$\overline{\xi} = \sum_{i=1}^{N} \{ \prod_{j=i+1}^{N} p_j \cdot [\alpha_i p_i + (1 - \sum_{j=i+1}^{N} \alpha_j) \cdot (1-p_i)] \cdot \xi_i \} \tag{3-43}$$

$$(p_i = \exp(-\alpha_i \lambda \cdot (l_i + d + g_i)))$$

Similarly, the average patch size for the posterior requests is derived in (3-44), where $G_i(g_i)$ is the maximum patch size and φ_j^i is the amendatory portion of the patch size determined by Δ_i'. They are respectively expressed in (3-45) and (3-46):

$$G_i'(g_i) = \exp(-\sum_{k=i+1}^{N}\sum_{t=k}^{N}\alpha_t\lambda\Delta_k)\cdot[1-\exp(-\sum_{t=i}^{N}\alpha_t\lambda\Delta_i)] \cdot(0.5\sum_{j=i}^{N}\alpha_j\lambda g_j G_j(g_j)+\sum_{j=1}^{i-1}\varphi_j^i) \tag{3-44}$$

$$G_i(g_i) = \sum_{j=0}^{N-i}\max(l_i+g_i-l_{i-j},\ 0)\cdot(r_{i-j}-r_{i-j-1}) \tag{3-45}$$

$$\varphi_j^i = 0.5\sum_{k=1}^{j}\{[\max(\Delta_i'+l_i+g_i-l_k,\ 0)]^2\cdot(r_k-r_{k-1})\alpha_j\lambda\} \tag{3-46}$$

Since the requests arrived within interval d are served together, the expected number of patches to be transmitted for layer i during d can be expressed as in (3-47). Therefore, the expected size of patches within interval I_ω can be calculated as in (3-48).

$$E(n_{patch}) = 1-\exp(-\alpha_i\lambda d) \tag{3-47}$$

$$E(S_{patch}) = E(n_{patch})\cdot g_i/d\cdot G'(g_i) \tag{3-48}$$

So considering all the requests for different layers and combining the overall size of transmitted patches, we obtain B_b in (3-49):

$$B_b = (1/I_\omega)\cdot[\overline{\xi}+(1/d)\cdot\sum_{i=1}^{N}(1-\exp(-\alpha_i\lambda d))\cdot g_i\cdot G_i'(g_i)] \tag{3-49}$$

B_p includes both the prefix that the proxy sends to the clients and the patches that the proxy relays to the clients from the server. Since the proxy batches the requests from similar classes within interval *d* and sends one stream for these requests, the expected number of batches to be transmitted for layer *i* during *d* can be expressed as in (3-50). Moreover, the total size of transmitted batches depends on the arrival rate of requests for all the layers and the average duration within which arrived requests are served by batches. This duration can take either l_i+g_i+d or $l_i+g_i+d+\Delta_i'$ for different layers with probability p_i. We calculate p_i in expression (3-51) and further derive the expression for B_p as in (3-52):

$$E(n_{batch}) = E(n_{patch}) = 1 - \exp(-\alpha_i \lambda d) \qquad (3\text{-}50)$$

$$p_i = [\prod_{k=i+1}^{N} \exp(-\sum_{t=k}^{N} \alpha_t \lambda \Delta_k)] \cdot [1 - \exp(-\sum_{t=i}^{N} \alpha_t \lambda \Delta_i)] \qquad (3\text{-}51)$$

$$B_p = \frac{1}{I_\omega} \sum_{i=1}^{N} \{\alpha_i \lambda r_i l_i (1 - e^{-\alpha_i \lambda d}) / d \cdot [\sum_{j=i+1}^{N} p_j (l_j + g_j + d + \Delta_j') + \sum_{j=1}^{i} p_j (l_j + g_j + d)]\}$$
$$+ B_b + \frac{1}{I_\omega} \sum_{i=1}^{N} \alpha_i r_i l_i \qquad (3\text{-}52)$$

In (3-52) the first term denotes the total bytes of the prefix forwarded to all the clients (which is sent separately to each batch of clients in interval *d*). The second and the third term of the expression represent the total bytes of the single stream sent to serve all the related requests. Therefore the overall bandwidth cost per second for one video object ω and video object set Ω are denoted in (3-53) and (3-54), respectively.

$$f_2(H_\omega, d) = c_b \cdot B_b + c_p \cdot B_p \qquad (3\text{-}53)$$

$$F_2(H, d) = \sum\nolimits_{\omega \in \Omega} f_2(H_\omega, d) \qquad (3\text{-}54)$$

The multi-objective optimization model can be established as:

$$\mathbf{MM}: \underset{l_1,l_2,\ldots,l_N}{\text{Minimize}} \ \mathbf{F}(H,d) = [F_1(H,d),\ F_2(H,d)]^T$$

$$\text{subject to: } 0 \le l_N \le l_{N-1} \le \ldots \le l_1 \le L \text{ (for each object in } \Omega)$$

$$\sum_{\omega\in\Omega}\sum_{i=1}^{N} l_i(r_i - r_{i-1}) \le H$$

where the first constraint is to ensure the feasibility of cached length of each layer and the second constraint is the cache size limitation on the proxy.

5.5.2 Heuristic optimization method

Generally a single global optimal solution to the above problem does not exist unless the utopia point, which combines the best instances of each objective, happens to be attainable. Instead of expressing an explicit preference function between the two optimization objectives, decision-makers always prefer to choose the best tradeoff from a palette of solutions (Pareto optimal set). In this section, Pareto optimal set is obtained through two hierarchy algorithms: *SingleObjSearch(H_i, d)* and *MultiObjSearch(H)*. Based on the assumed H_i and d, *SingleObjSearch* derives the cached lengths for all the layers of video object i and corresponding bandwidth cost incurred for this object. *MultiObjSearch* samples startup latency d from the region of $[d_{min}, d_{max}]$ with grain v, and optimize objective $F_2(H, d)$ to obtain a complete caching solution for all the video objects by utilizing *SingleObjSearch* iteratively. The optimization solutions for all the latency samples are obtained and put in Parte optimal set. We select from this set a compromise solution which has the closest Euclidean distance to the utopia optimal point. Figure 5-III illustrates the detailed algorithms to derive Pareto optimal set.

SingleObjSearch algorithm initially allocates the same caching length to each layer. We estimate whether increasing the caching budget, with step caching size *sstep*, for one layer while decreasing the same amount for another can reduce the bandwidth cost incurred for this video object. The two layers are identified and their caching sizes are updated accordingly such that the reduction in bandwidth cost is maximized. In the case that such two layers do not exist, we decrease *sstep* and search iteratively. If *sstep* cannot be decreased or the adjustment count exceeds predefined threshold, *SingleObjSearch* returns the caching length set $\mathbf{L}_{\omega}$ and continues to search for the patch length threshold set $\mathbf{G}_{\omega}$ for each layer if necessary. In addition to $\mathbf{L}_{\omega}$ and $\mathbf{G}_{\omega}$, the corresponding bandwidth cost for the specific video object ω is returned to *MultiObjSearch* algorithm to derive the Pareto optimal set for the **MM** optimization problem.

```
1. $\mathbf{L}_\omega = \{l_k = H_i/(N \cdot r_N), k = 1\ to\ N\}, G = \{g_k = 0, k = 1\ to\ N\}, sstep = 5\varepsilon, scnt = 0;$
2. while ($scnt < cntThreshold$ and $sstep > 0$)
3.    for (each layer $k$)
4.       $l_k^+ = l_k + sstep/(r_k - r_{k\text{-}1})$, update $\mathbf{L}_\omega$ to be $\mathbf{L}_{\omega,k}^+$;
5.       $l_k^- = l_k - sstep/(r_k - r_{k\text{-}1})$, update $\mathbf{L}_\omega$ to be $\mathbf{L}_{\omega,k}^-$;
6.       $\varphi_1(\mathbf{L}_{\omega,k}, d) = f_2(H_i \mid (\mathbf{L}_{\omega,k}, \mathbf{G}), d) - f_2(H_i \mid (\mathbf{L}_{\omega,k}^+, \mathbf{G}), d);$
7.       $\varphi_2(\mathbf{L}_{\omega,k}, d) = f_2(H_i \mid (\mathbf{L}_{\omega,k}^-, \mathbf{G}), d) - f_2(H_i \mid (\mathbf{L}_{\omega,k}, \mathbf{G}), d);$
8.    select layer $t$ with maximum $\varphi_1$, and layer $s$ with minimum $\varphi_2$, $scnt++$;
9.    if $(\varphi_1(\mathbf{L}_{\omega,t}, d) > \varphi_2(\mathbf{L}_{\omega,s}, d))$
10.      $l_t = l_t + sstep/(r_t - r_{t-1})$, $l_s = l_s - sstep/(r_s - r_{s-1})$, update $\mathbf{L}_\omega$;
11.   else $sstep = sstep - \varepsilon$;
12. for (each layer $k$)
13.   while $(g_k < \min(L - l_k, l_{k-1} + g_{k-1} - l_k))$
14.      $g_k = g_k + sstep/(r_k - r_{k\text{-}1})$, update $\mathbf{G}_\omega$ to $\mathbf{G}_\omega^+$;
15.      if $(f_2(H_i \mid (\mathbf{L}_\omega, \mathbf{G}_\omega^+), d) < f_2(H_i \mid (\mathbf{L}_\omega, \mathbf{G}_\omega), d))$
16.         $\mathbf{G}_\omega = \mathbf{G}_\omega^+$;
17. $f_2^o(H_i, d) = f_2(H_i \mid (\mathbf{L}_\omega, \mathbf{G}_\omega));$
```

(a) Algorithm *SingleObjSearch(H_i, d)*

```
1. $d = d_{\min}$, $H_i = H/N_\Omega$, $P = \varnothing$;
2. while $(d \le d_{\max})$
3.    $mstep = 5u$, $mcnt = 0$;
4.    while ($mcnt < cntThreshold$ and $mstep > 0$)
5.       for (each object $i$ in set $\Omega$)
6.          $f_+(H_i, d) = f_2^o(H_i, d) - f_2^o(H_i + mstep, d);$
7.          $f_+(H_i, d) = f_2^o(H_i + mstep, d) - f_2^o(H_i, d);$
8.       select object $t$ with maximum $f_+(H_i, d)$, and object $s$ with
         minimum $f_-(H_i, d)$, $mcnt++$;
9.       if $(f_+(H_t, d) > f_-(H_s, d))$
10.         $H_t = H_t + mstep$, $H_s = H_s - mstep$;
11.      else $mstep = mstep - u$;
12.   $F_2(H, d) = \sum_i f_2^o(H_i, d)$, $d = d + v$;
13.   $\mathbf{P} = \mathbf{P} \cup (F_1(H, d), F_2(H, d))$
```

(b) Algorithm *MultiObjSearch(H)*

Figure 5-III: Heuristic algorithms to derive Pareto optimal set

Similar to *SingleObjSearch* algorithm, *MultiObjSearch* algorithm initially allocates cache budget to each video object evenly. With each sampled access latency d, cache allocation grain u, caching budget H_i, we obtain the optimal bandwidth cost $f_2^{o}(H_i, d)$ for each object using *SingleObjSearch* algorithm. We then estimate whether increasing the caching budget, with step size *mstep*, for one object while decreasing the same amount for another can reduce the total bandwidth cost. The estimation is achieved by identifying the object t with maximum $f_+(H_i, d)$ and object s with minimum $f_-(H_i, d)$, and comparing $f_+(H_t, d)$ and $f_-(H_s, d)$ as in Figure 5-III. If an adjustment cannot be made using the current step size *mstep*, we decrease *mstep* and try to adjust again. If *mstep* cannot be decreased or the adjustment count exceeds predefined threshold, we save the combination of bandwidth cost and corresponding access latency d in Pareto optimal set **P** and start another round of search with modified access latency.

After Pareto optimal set **P** is derived, we select the optimal solution $\{F_1(H, d^{o}), F_2(H, d^{o})\}$ with the closest Euclidean distance to the utopia optimal point $\{F_1^{o}, F_2^{o}\}$. The term *close* usually implies that one minimizes the Euclidean distance $N(H)$, which is defined as in (3-55):

$$N(H) = |\mathbf{F}(H) - \mathbf{F}^{\circ}| = \{\sum_{i=1}^{k}[F_i(H_i) - F_i^{\circ}]^2\}^{0.5} \propto \{\sum_{i=1}^{k}(F_i^{trans})^2\}^{0.5} \qquad (3\text{-}55)$$

where F_i^{trans} is the transformed original cost function which is used to be insensitive to the scalar of both objectives F_1 and F_2. This approach is consistently referred to as normalization. In this case, F_i^{trans} generally has values between zero and one. It is expressed as in (3-56):

$$F_i^{trans} = \frac{F_i(H) - F_i^{\circ}}{F_i^{\max} - F_i^{\circ}} \tag{3-56}$$

This posteriori preference algorithm calculates $F_1^{\mathbf{o}}$ and $F_2^{\max}$ by setting startup latency d to $d_{\max}$, and derives $F_1^{\max}$ and $F_2^{\mathbf{o}}$ by setting startup latency d to $d_{\min}$. Then the optimized startup latency which minimizes the Euclidean distance $N(H)$ can be derived by a complete search method.

Finally, we obtain the optimized caching strategy $\mathbf{H^o}=\{H_i^{\mathbf{o}}, i=1 \textit{ to } N_\Omega\}$ for the object set Ω, cached layer set $\mathbf{L^o}$ and patch length set $\mathbf{G^o}$ for each object. Given aforementioned parameters, the computational complexity of *SingleObjSearch* algorithm is $O(N_\omega\cdot(H_\omega/u)\cdot(L_\omega/\varepsilon))$. Applying *MultiObjSearch* algorithm for a given H, we solve the problem of allocating the cache to multiple objects in time $O(N_\Omega\cdot(d_{\max}/(v\cdot u)))$. In addition, the post preference algorithm derives the global solution in time $O(d_{\max}/v)$.

5.6 Performance Evaluation

In this section, we examine the performance of proposed optimized caching strategy (OCS) and scalable transmission scheme (STS) through simulations. The results reveal their superiority compared with other caching and transmission methods, and illustrate the impact of various factors such as request arrival rate and cache capacity on the optimization objectives. The tradeoffs between the two optimization objectives are evaluated finally.

5.6.1 Simulation settings

We run the simulations for both single object case and multiple heterogeneous objects case. QoS requirements of all the requests are classified into three classes to represent three typical connection situations: narrow bandwidth, medium bandwidth, and broad bandwidth. We consider client reservoir with the following four request population distributions among those classes:

1) Narrow band dominated: (0.65, 0.25, 0.1)

2) Broad band dominated: (0.1, 0.25, 0.65)

3) Medium band dominated: (0.25, 0.65, 0.1)

4) Uniform distributed: (1/3, 1/3, 1/3)

The grain sizes in the simulations are properly selected to ensure high precision and reasonable computational complexity. For both single object and multiple objects simulation cases, we choose the cache allocation grain u as 64KB, caching length grain ε as 0.25 second, and latency grain as one second. According to the workload analysis [89], the request rates for the video objects follow the *zipf* distribution with parameter 1.6.

We represent c_p as a normalized portion of c_b, that is $c_p \in [0,1]$ and $c_b=1$. If $c_p=0$, then the bandwidth consumption between the proxy and clients does not incur any cost. So the overall bandwidth cost is the same as backbone bandwidth cost. The implication is that the proxy and clients may all reside inside the same simple LAN. If $c_p=1$, then $c_p=c_b$ which implies a complex and mixed network behind the proxy. We perform simulations to illustrate the effect of various parameters on both backbone bandwidth cost ($c_p=0$) and overall bandwidth cost ($c_p=0.1$). MPEG-4 FGS coded video objects are used in the simulations. Proper simulation parameters for single object case are shown in Table 5-II.

Table 5-II: Simulation parameters for single video object case

Notation	**Quantity**
N_ω	3
$\mathbf{r}=[r_1, r_2, r_3]$	[128, 512, 2048]Kbps
L_ω	1500 seconds
d	4 seconds

For the case of multiple objects, we use 30 MPEG-4 FGS coded video objects and randomly select request distribution for each object. Their lengths are uniformly distributed within [1000, 2000] seconds. The parameters of the simulation for multiple video objects case are listed in Table 5-III.

Table 5-III: Simulation parameters for multiple video objects case

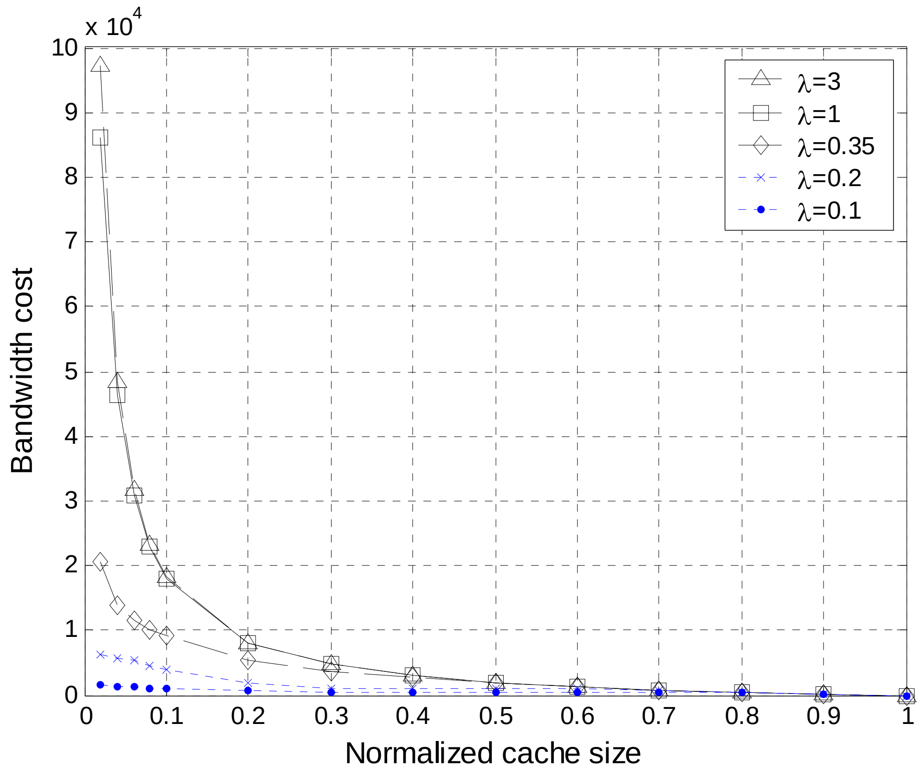
Notation	**Quantity**
N_{Ω}	30
$\mathbf{r}=[r_1 , r_2 , r_3]$	[128, 512, 2048] Kbps
$\lambda = \sum_{\omega\in\Omega} \lambda_{\omega}$	1 request/second
$\mathbf{d}=[d_{min}, d_{max}]$	[1, 11] seconds

5.6.2 Effect of user request rates and proxy cache size

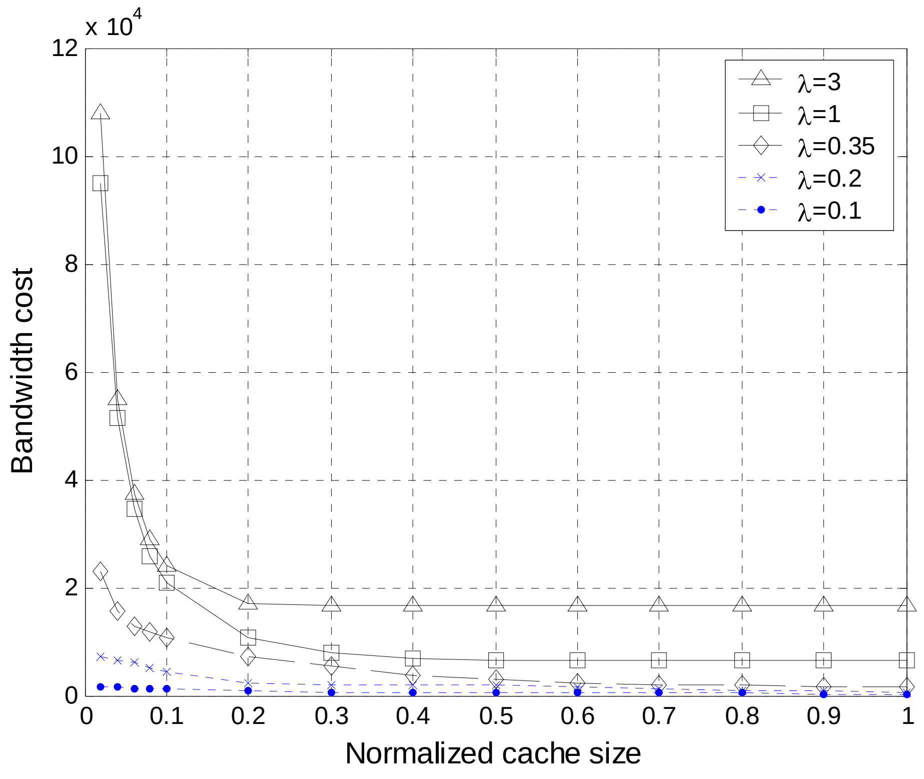
In this set of simulations, we study the effects of cache size H and request arrival rate λ on the bandwidth cost. For the ease of illustration the simulations are performed for a single video object with certain access latency (d=4) for narrow band request dominated distribution. With the parameters listed in Table 5-II, we draw the curve of bandwidth cost (F_2) against normalized cache size H under various request rates λ in Figure 5-IV. Four significant observations are identified and explained as follows:

- The bandwidth cost decreases with the rise of H and increases with the rise of λ. The two subfigures in Figure 5-IV differ in that Figure 5-IV(a) converges to 0 while Figure 5-IV(b) converges to a fixed value larger than 0. The reason is that when the cache size is large enough, nearly all the videos are cached in proxy and the backbone bandwidth cost is too small to contribute to the overall bandwidth cost in Figure 5-IV(b). Since the cache size does not affect the bandwidth consumption on the route between the proxy and clients, the curve in Figure 5-IV(b) is converged to a fixed value other than 0.

- Although the bandwidth cost increases with the rise of λ, the curves in Figure 5-IV(a) are distinct for small values of λ, but have near-complete overlap beyond λ=1 when the normalized cache size is large (such as larger than 0.04). The reason is that beyond λ=1, the requests clump into batches at the proxy and subscribe on to the same channels. Therefore they do not affect the backbone bandwidth consumption greatly.
- All the curves in both Figure 5-IV(a) and Figure 5-IV(b) have distinct costs at the start points. When the normalized cache size is small, $1/\lambda$ has more effects on the duration of the average channel interval because of the short length of prefix. Although the backbone bandwidth cost does not change, the decrease of interval duration increases its cost per second.
- The convergence values of the curves in Figure 5-IV(b) are distinct and are different from those in Figure 5-IV(a). The reason is that although λ does not affect the backbone bandwidth cost, it does affect the bandwidth cost on the route between the proxy and the clients. Since the convergence values of backbone cost are all zero which are shown in Figure 5-IV(a), the distinctness of overall bandwidth cost is apparent in Figure 5-IV(b).



(a) Effects on backbone bandwidth cost (c_p=0)



(b) Effects on overall bandwidth cost (c_p=0.1)

Figure 5-IV: Effects of proxy cache size and request rates on the bandwidth cost

5.6.3 Bandwidth cost reduction

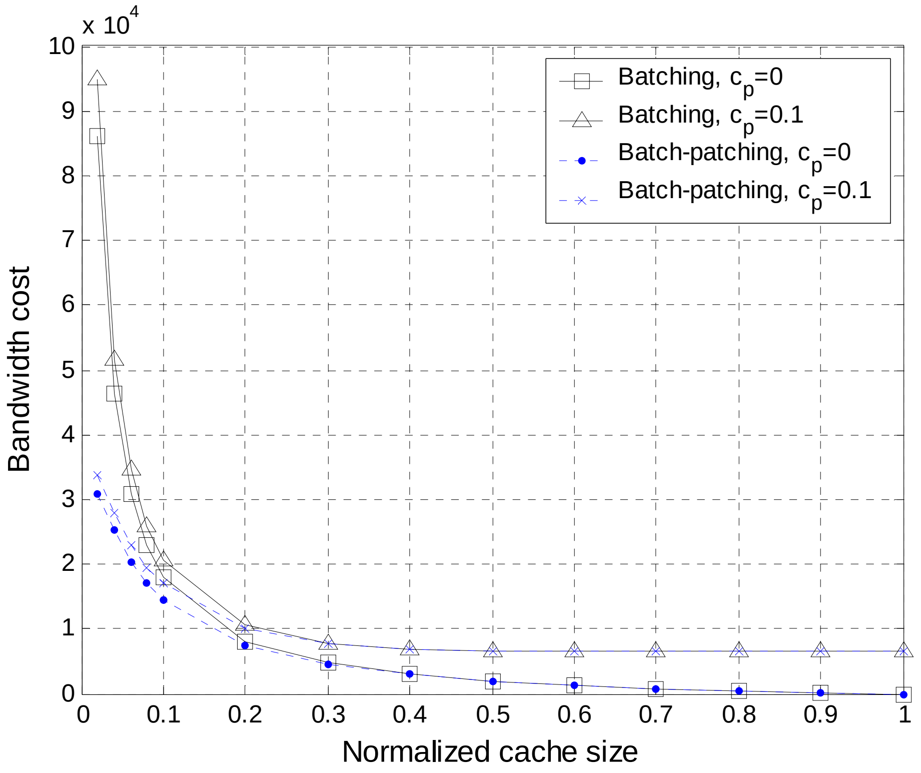
In this part of the simulations, we are interested in examining the bandwidth cost by employing the proposed OCS, compared with the following classical caching strategy:

$$MaxRate\text{:}\ l_i = l_{i+1},\ r_i = \left\{ \begin{array}{l} \dfrac{H}{l_i},\ l \in [0, l_i] \\ 0,\ l \in (l_i, L] \end{array} \right\}$$

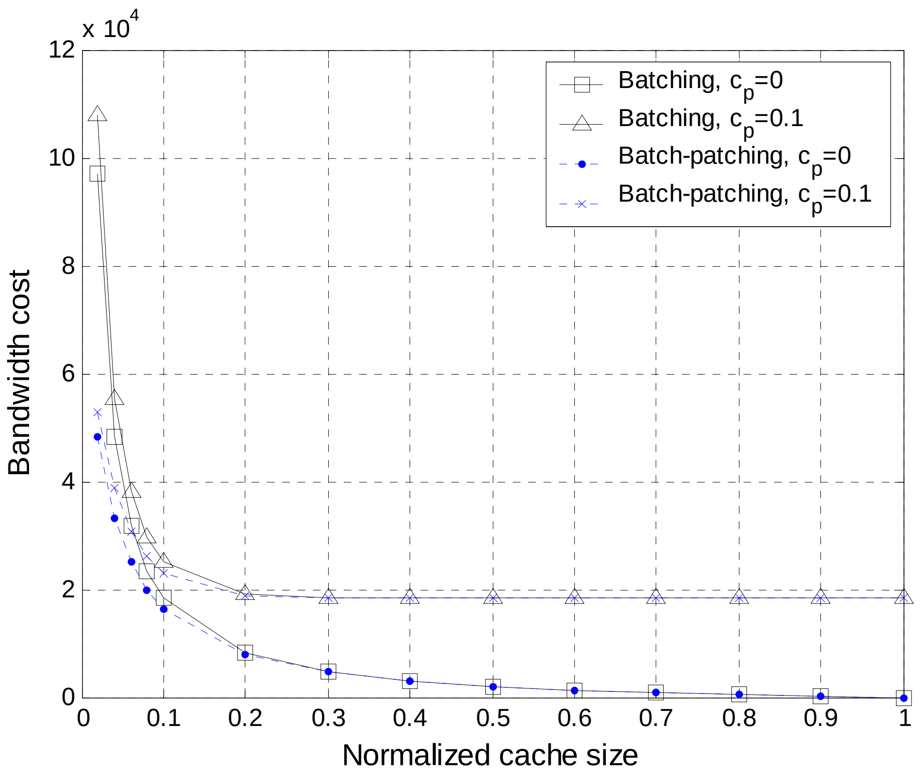
MaxRate caching strategy is not scalable because it does not consider the QoS difference among requests. With the bandwidth cost of OCS illustrated in Figure 5-V, we compare it to the bandwidth cost of *MaxRate* strategy for single video object in Figure 5-VI and Figure 5-VII. The comparison for multiple video objects is shown in Figure 5-VIII.

Figure 5-V illustrates the backbone bandwidth cost (c_p=0) and overall bandwidth cost (c_p=0.1) by employing OCS with both scalable batching and batch-patching transmission schemes for four request distributions. The results are further compared with the bandwidth cost incurred by using *MaxRate* scheme. Figure 5-VI and Figure 5-VII depict the reductions on backbone and overall bandwidth cost, respectively.

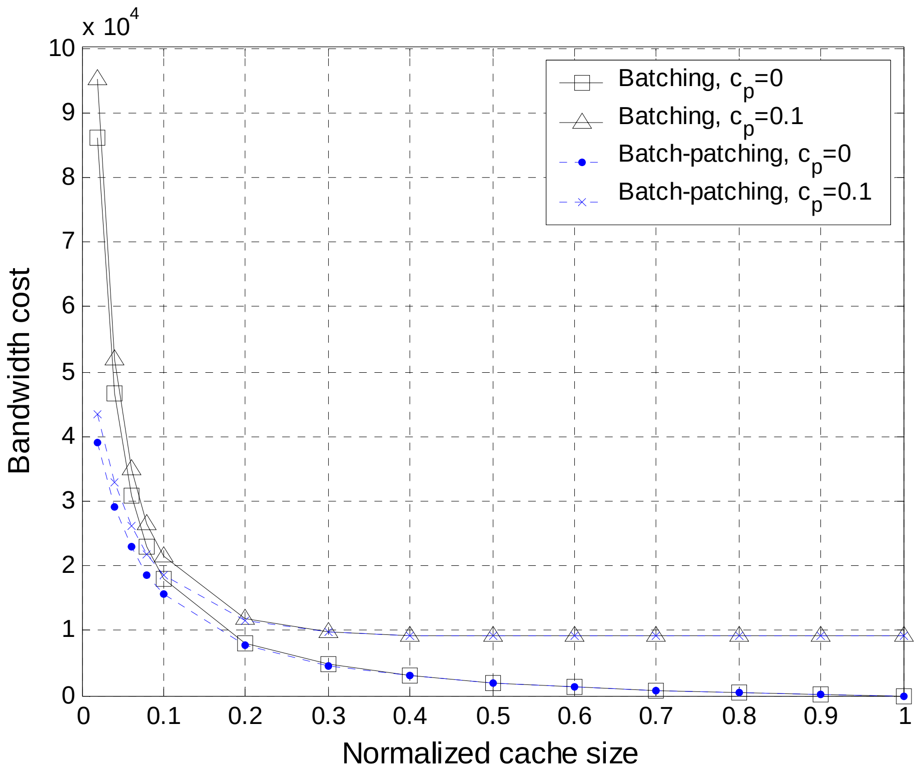
Figure 5-V reveals that OCS and STS are applicable to different request distributions. When proxy cache is small, scalable batch-patching transmission scheme has better performance than scalable batching scheme in terms of bandwidth cost. With the increase of cache size, both transmission schemes have similar convergence values. This observation can be explained as follows. When cache size increases, the length of cached prefix will increase as well. Since the sum of prefix length (l_i) and patch length (g_i) shall not exceed the video object length L, the increase of prefix will constrain the adjustment space for patch length which in turn confines the effectiveness of the patches. Therefore, when cache size is large, scalable batching and scalable batch-patching transmission schemes have the similar effects.



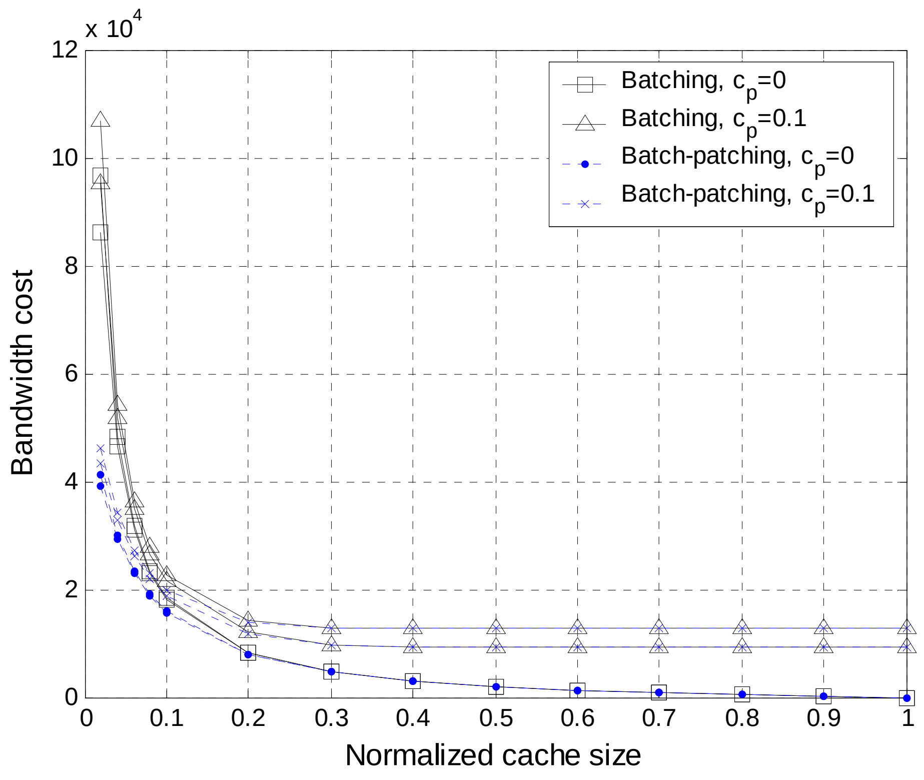
(a) narrow band dominated request distribution



(b) broad band dominated request distribution



(c) medium band dominated request distribution



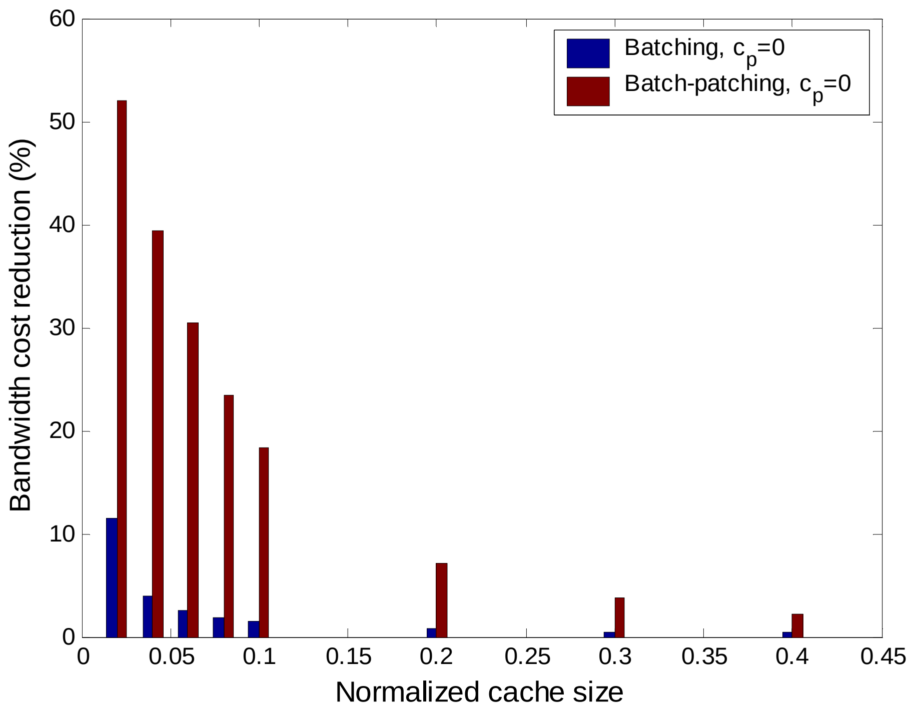
(d) uniform distribution

Figure 5-V: Bandwidth cost vs. cache size under four user request distributions

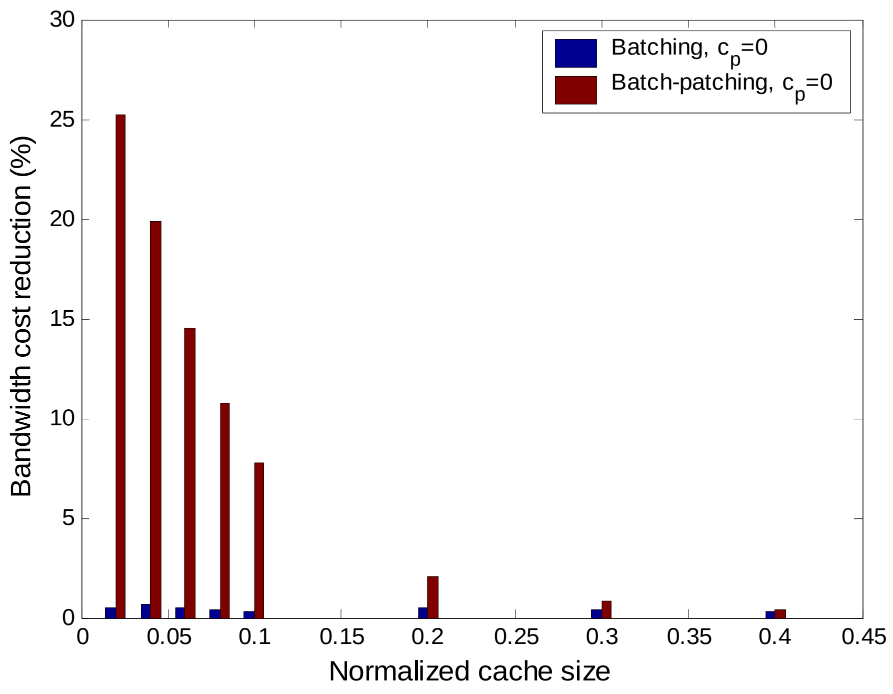
Figure 5-VI plots the backbone bandwidth cost reduction under OCS, compared with *MaxRate* caching strategy. We find that the reduction under all request distributions is large, especially when the OCS is combined with scalable batch-patching transmission scheme. In addition, two significant observations can be made from this figure:

- The reduction is more evident for the requests with narrow band dominated distribution. The reason is that *MaxRate* scheme treats all the requests equally as broad band requests while OCS differentiates them according to their QoS requirements. The more broadband requests the proxy receives, the more appropriate it is to increase the prefix lengths of all the layers altogether. Therefore under broad band dominated distribution, the prefix lengths of different layers become similar which in turn makes the performance of two caching strategies similar.

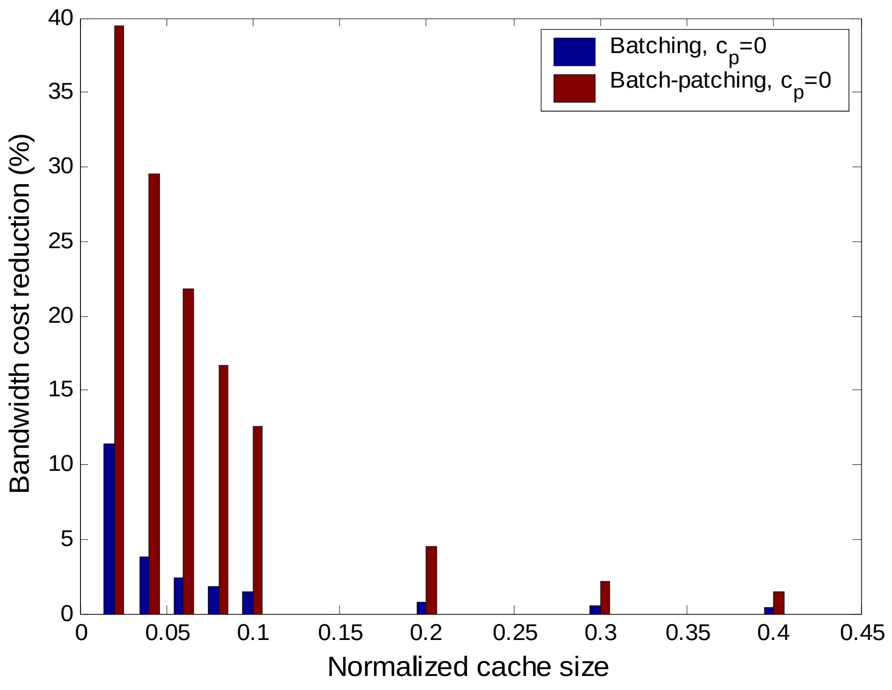
- The reduction under the scalable batch-patching transmission scheme is more significant than that under the scalable batching scheme. The reason is that even when the cached prefix lengths are similar under both caching strategies, the patch lengths can still be designed to further reduce bandwidth cost.



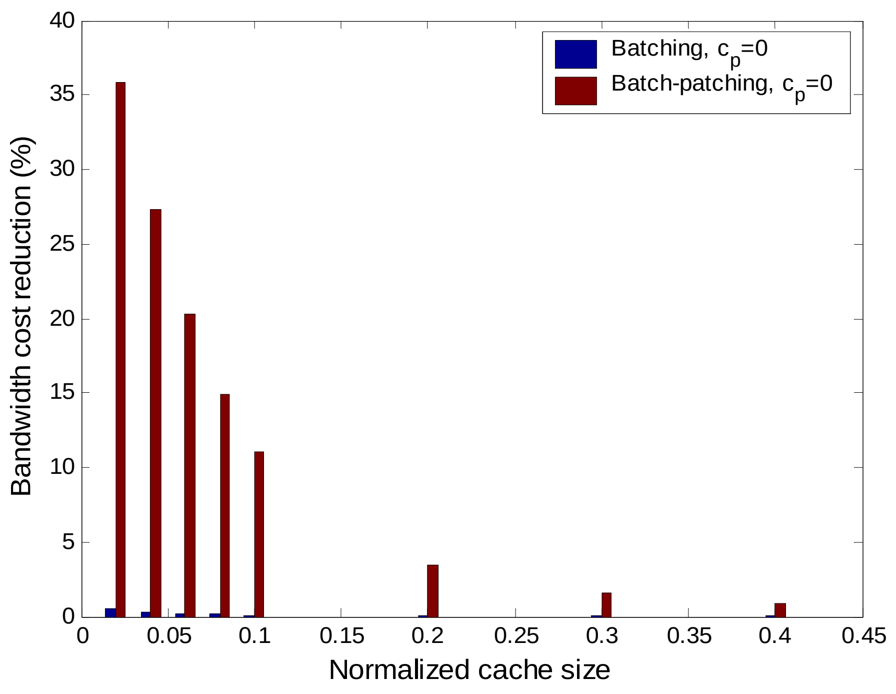
(a) narrow band clients dominated distribution



(b) broad band clients dominated distribution



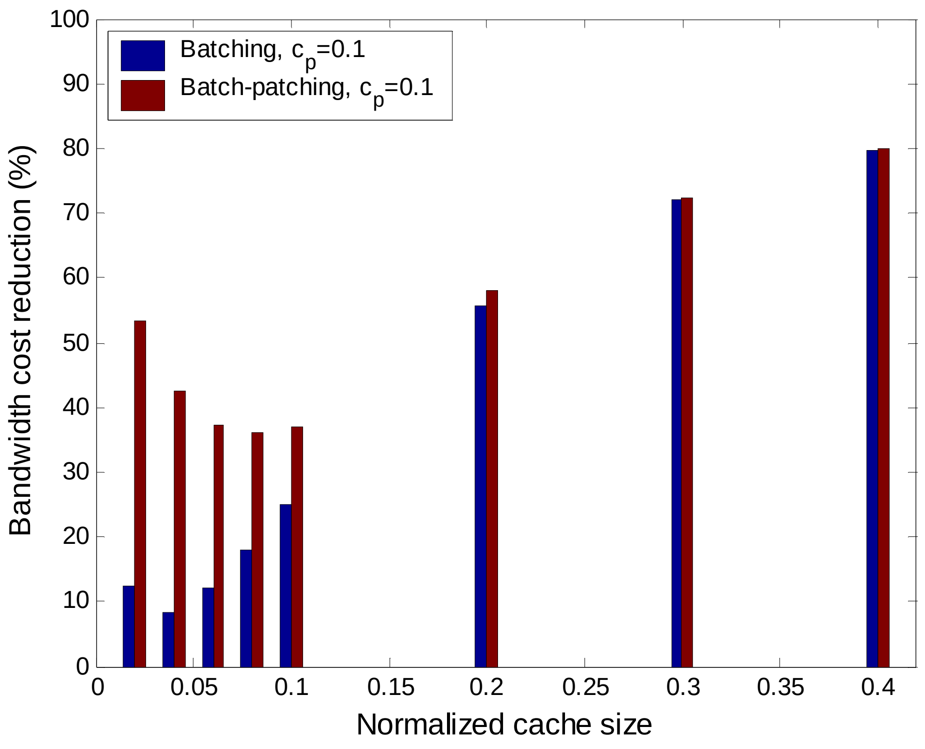
(c) medium band clients dominated distribution



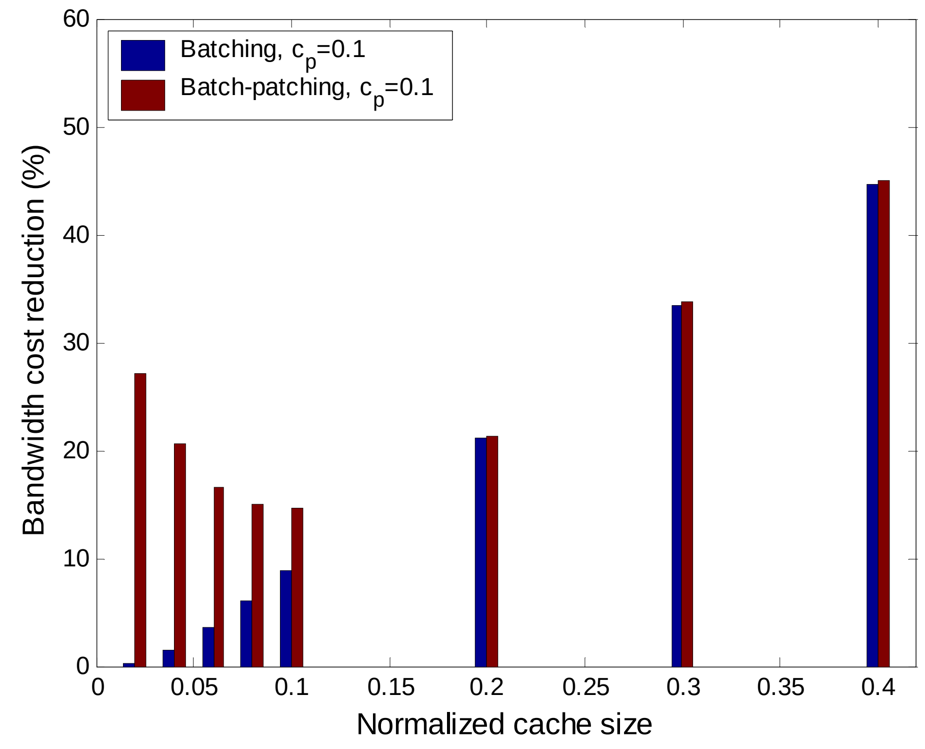
(d) uniform distribution

Figure 5-VI: Backbone bandwidth cost reduction achieved by OCS

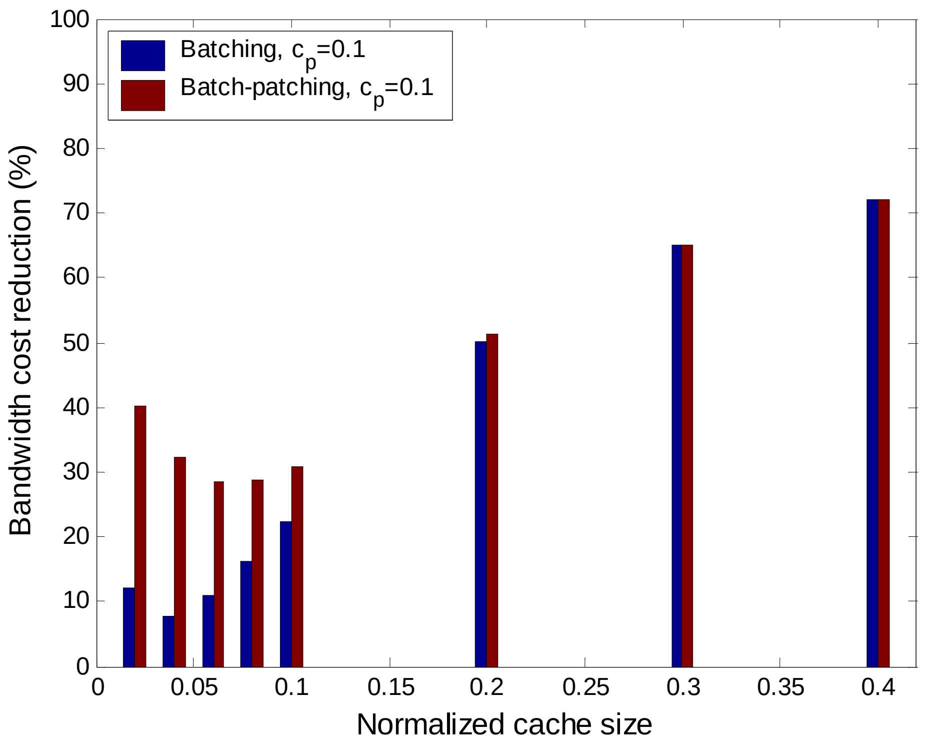
Figure 5-VII plots the overall bandwidth cost reduction under OCS, compared with *MaxRate* strategy. We find that the reduction under all request distributions is large, sometimes as large as 80%. Similar to Figure 5-VI, the reduction in Figure 5-VII(a) and Figure 5-VII(c) is more significant that in Figure 5-VII(b) and Figure 5-VII(d). In addition, the performance of OCS in terms of overall bandwidth cost is also better than that in terms of backbone bandwidth cost. The reason for this can be explained as follows. Each request distribution contains some portion of low QoS requirement requests and hence, in *MaxRate* strategy, caching the video objects with the highest rate and serving those requests with the highest rate become wasteful, especially in terms of the bandwidth cost between the proxy and the clients. In this case, OCS is very effective in reducing the bandwidth cost behind the proxy. With the increase of cache size, this part of cost occupies a larger portion in the overall bandwidth cost and hence the reduction achieved by OCS is more significant.



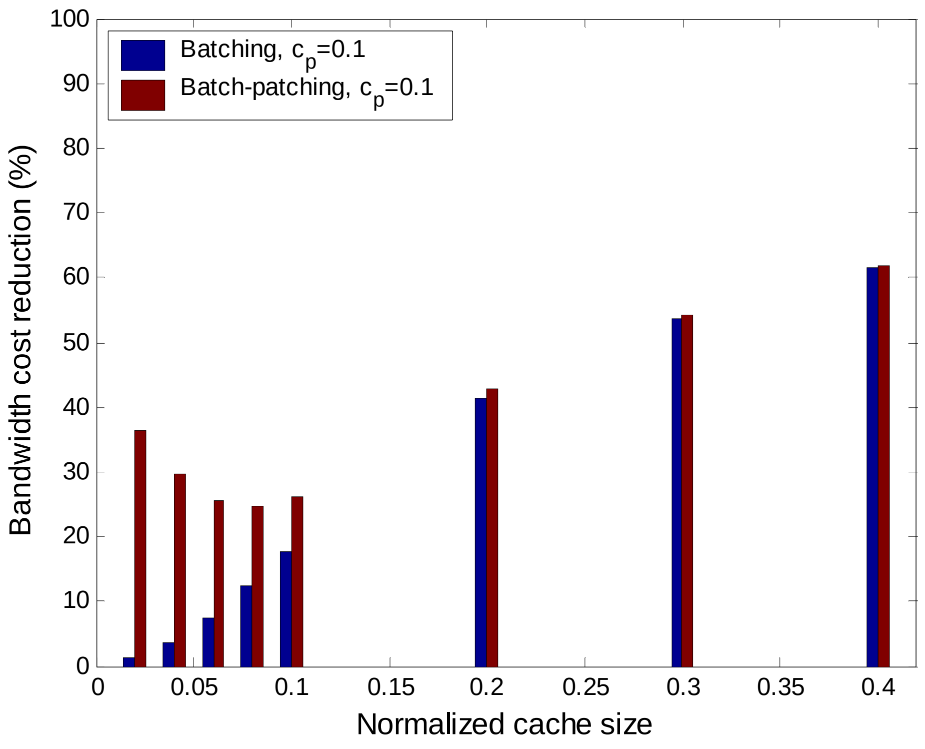
(a) narrow band client dominated distribution



(b) broad band client dominated distribution



(c) medium band client dominated distribution



(d) uniform distribution

Figure 5-VII: Overall bandwidth cost reductions achieved by OCS

The comparison for multiple objects is shown in Figure 5-VIII. Note that this simulation is performed under the batching transmission scheme and both backbone bandwidth cost and overall bandwidth cost reductions are illustrated. Similar to the simulation for a single object, the bandwidth cost reduction achieved by OCS is significant.

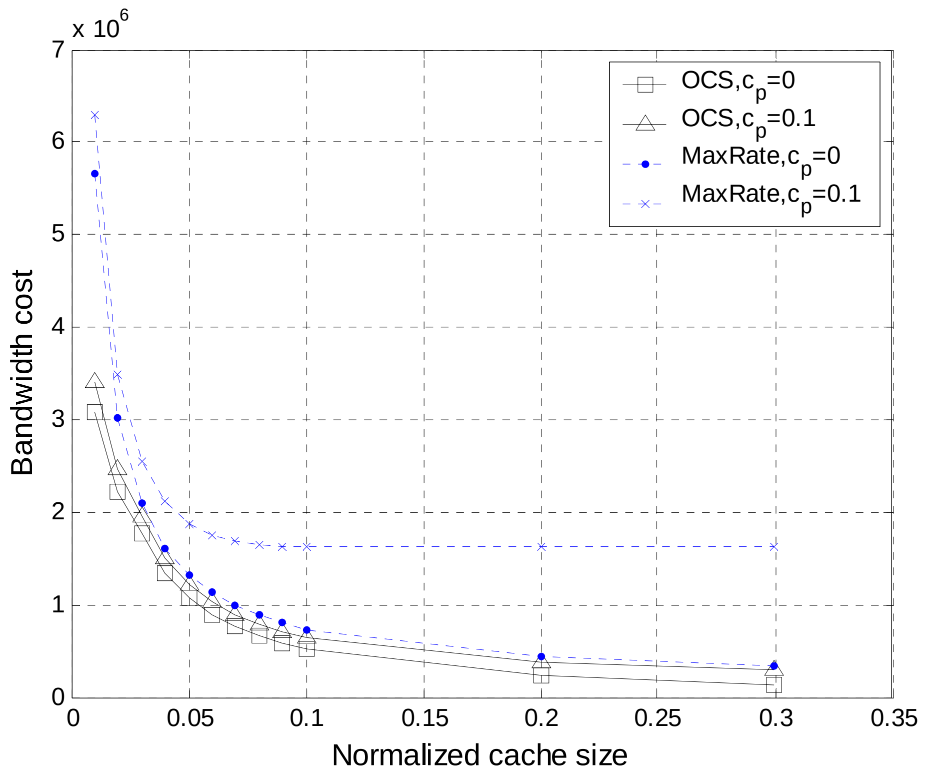


Figure 5-VIII: Bandwidth cost comparison among different schemes.

So far, we have studied the performance of the proposed OCS combined with two types of transmission schemes for MPEG-4 FGS coded video objects. We find that the best performance is given under the scalable batch-patching scheme and OCS always performs much better than *MaxRate* strategy in terms of bandwidth cost.

5.6.4 Tradeoff between bandwidth cost and access latency

In above analysis, we are concerned with one type of optimization objective (minimum bandwidth cost) only. In this section, we study the tradeoff between the two objectives (minimizing bandwidth cost and access latency). These optimization objectives are competing because there is no single design that has the lowest value for both objectives. A reliable visualization of a two objective optimization outcome is the OBJ1 vs. OBJ2 chart shown in Figure 5-IX where optimal designs spread over the Pareto frontier. The simulations are performed for the case of multiple video objects and the normalized cache size is fixed as 0.08. At this point we are able to calculate the optimal solution that better answers to the original design purposes. Figure 5-IX also gives an idea of the Pareto design boundaries, such as the point (a) and (c). Optimal designs fit the Pareto frontier in the midpoints. As a matter of fact point (b) is the best compromise between bandwidth cost and access latency.

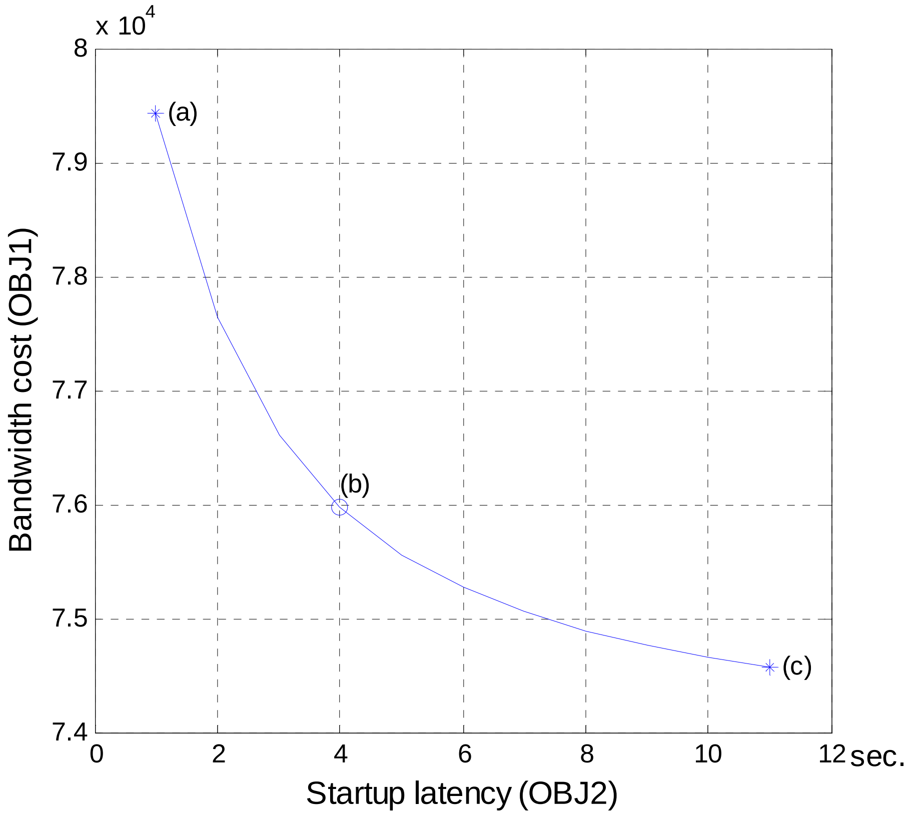


Figure 5-IX: Pareto frontier chart

5.7 Concluding Remarks

This chapter investigates the problem of caching scalable coded video objects on the proxy and transmitting them to heterogeneous clients. With proposed scalable transmission scheme, we develop a multi-objective optimization model with closed-form expressions and derive the optimized scalable caching strategy through heuristic algorithms. With obtained caching strategy, network bandwidth cost and user access latency are minimized simultaneously in favor of both service provider and clients. Our evaluation demonstrate that, compared with other caching strategy and non-scalable transmission schemes, proposed caching and transmitting methods have a better performance and provide a bandwidth and latency efficient delivery.

Chapter 6: CONCLUSIONS AND FUTURE RESEARCH

The transport of video in real-time has been one of the most important multimedia services over Internet. Due to explosive demand, we are faced with challenges to develop efficient solutions, in terms of overall system performance and user experience, to overcome the problem caused by the heterogeneous transport environment and limited system resources. Video proxy is a small server placed in the system which provides the opportunity to reduce the load of both source server and network. Moreover, scalable video coding (SVC) offers the capability to dynamically adapt to the heterogeneous requests and network conditions. MPEG-4 FGS is one of the efficient SVC schemes to provide the aforementioned scalability with fine granular quality adjustment and is utilized as the video codec in this dissertation. To improve the performance of the video streaming system with proxy assistance, we present several new techniques which efficiently utilize proxy resources based on video characteristics and user access behaviors. These techniques are practical and can be easily implemented. In this chapter, we conclude the dissertation by summarizing proposed techniques, identifying major difficulties encountered, and finally discussing the future research directions.

6.1 Conclusions

In this section, we conclude our dissertation by summarizing the contents for each chapter and giving remarks on the performance of proposed techniques.

In Chapter 1, we focus on summarizing the motivation and the main contributions of this dissertation. We start with introducing the concept and two modes of the video transport over Internet. Although the streaming model is generally regarded as being better than the download mode, it brings extra design challenges that may come from network constraints, server capacity, end-users requirements, and so on. These challenges motivate us to make use of the flexibility provided by SVC and the resources available on the proxy to deliver video streams efficiently. Several problems in the area of video transmission, adaptation, and caching are identified. At the end of this chapter, we summarize our contributions and outline the successive chapters.

We further provide a comprehensive background overview for the internet video streaming in Chapter 2. Like many other Internet services, real-time video streaming over Internet is not possible without the help of efficient protocols. We start this chapter by introducing the real-time protocol stack that our research is based on. These International standardized protocols including RTSP, RTCP, RTP, and TFRC have been proven to be effective for real-time video streaming and provides extendibility for researchers to explore. Similarly, the advance of SVC brings further possibilities for us to improve the streaming systems. Video codec has evolved from providing no or limited scalability, such as MPEG-1 and MPEG-2, to providing scalability in almost all the coding dimensions such as space, SNR, and temporal dimension. In Chapter 2, we explore the flexibility and efficiency provided by SVC and focus on describing the characteristics of MPEG-4 FGS which is taken as the example video codec in the successive chapters of this dissertation. After summarizing current research efforts on video streaming systems, we follow the direction of innovating and improving algorithms and methods in proxy-assisted video streaming architecture. This streaming architecture was depicted briefly as well. We identify three major research issues where we focus on in this dissertation as follows: adaptive video transmission in the face of bandwidth variations, video adaptation in favor of heterogeneous requests, and scalable coded video management (caching and transmission) on intermediate network proxies. In Chapter 3 to Chapter 5, we propose a set of schemes, models, and algorithms to address these issues accordingly.

In Chapter 3, we point out that existing transmission schemes are not able to efficiently compensate video quality degradation when bandwidth drops below the QoS requirement. During the streaming period with insufficient bandwidth, some stream segments have to be discarded to ensure smooth video playback at the client. To alleviate the quality degradation, we present our adaptive compensation method (ACM) where some of the discarded data are compensated after bandwidth resumed and before they were displayed. An optimized compensation segment was obtained, such that it can arrive at the client in time for display and will not cause buffer overflow. By constantly monitoring the network and adapting to bandwidth fluctuations, this method efficiently utilizes network resources to enhance video quality. The quality improvement under ACM, compared with the traditional streaming method without ACM, was illustrated through simulations. Additionally, we investigate the effects of several environment factors on the performance of ACM and gave some guidelines to select or maintain proper parameters during streaming sessions to achieve optimal compensation effect. This compensation method is lightweight and can be incorporated into various real-time streaming systems easily.

In Chapter 4, we consider the problem that existing video adaptation schemes do not focus on improving the video quality of the adapted stream and usually bring serious quality degradation during adaptation process. To solve this problem, we propose to shape scalable coded video streams hierarchically on both scene and frame levels. Video scenes are adapted (allocated proper bit budgets) fairly based on their importance, texture and motional complexity. With a derived bit budget for each scene, we further adapt its frames with different bit rates to achieve a good tradeoff between SNR and temporal resolution such that user-perceived quality, which is evaluated in terms of average quality and quality variance, is optimized. Compared with three traditional adaptation methods, the proposed optimized adaptation scheme (OAS) achieves the following three properties which were demonstrated by extensive simulations: (1) superior user-perceived quality, (2) fine-grained adaptation granularity, (3) low computational complexity. OAS efficiently utilizes the resources on the adaptation nodes and is easy to be employed in real-time video streaming systems, such as small systems deployed in hotels and residual apartments containing hundreds of users, and large scale systems deployed across countries consisting of millions of users.

In Chapter 5, we investigate the issue to utilize the proxy resources to reduce network and server workload and to improve the quality of service experienced by users. We propose to statically cache video frames and video objects with different bit rates and coordinate the transmission to serve correlated requests. A maneuverable multi-objective optimization model is established and then solved to provide an optimized proxy caching strategy such that performance metrics concerning both service provider and user are optimized. A good tradeoff between two metrics: bandwidth cost and user access latency is obtained with heuristic algorithms. Through extensive simulations, we investigate the effects of several system parameters on the performance of proposed scalable cache management solution and compare it with traditional non-scalable caching methods under various video transmission schemes. Evaluation results illustrate the effectiveness of the proposed solution. Since the computational complexity of the algorithms grows much slower with the number of video objects than with the number of video layers, the proposed caching solution is appropriate for those streaming systems containing lots of videos coded with limited number of layers.

In summary, we present several new techniques for video streaming applications. Specifically, for MPEG-4 FGS video stream, we provide adaptive compensation method, video adaptation scheme, cooperative transmission scheme and proxy caching strategy, that can help to improve the streaming system in the aspect of throughput, cost, and user experienced quality. These proposed techniques are practical and ready to be deployed. The simulation results obtained are promising.

6.2 Major Difficulties Encountered

In this dissertation, we follow a sequence of procedures to realize the challenges that we have identified. Various difficulties exist in these procedures:

- Select significant and maneuverable objectives to handle
- Design proper schemes to facilitate the modeling process
- Establish models incorporating environment parameters and objectives with closed-form expressions
- Derive solutions for the models
- Evaluate the performance of proposed solution comprehensively and credibly

We manage to deal with most of the difficulties smoothly. In this section, we list several major ones that we encounter during this research.

Choosing proper design objectives is regarded as a crucial factor to the success of research. The objectives need to be significant to the research community and at the same time maneuverable to be achieved with reasonable computational complexity. Nevertheless, significance and maneuverability are usually competing and difficulties come in determining the proper tradeoff between them. Our approach was to forecast the difficulty of the modeling and resolving processes based on our specific research scenario and extensive reference of related sources. Furthermore, primary derivations were performed to validate our tentative decision.

For example, in Chapter 3 we choose to maximize the compensation effect because it can be achieved with instant calculation which illustrates the effectiveness of proposed compensation method in a lightweight manner. Since the compensation decisions need to be made for each client instantly and frequently in our research scenario, we choose to optimize the compensation effect to avoid the possible complex computation which is inevitable if we optimize other objectives such as user-perceived quality. On the other hand, we choose to achieve a more straight forward objective: bandwidth cost in Chapter 5. This objective will incur some complexity in both formula derivation and solution computation. Nevertheless, we regard it as appropriate in the research scenario because proposed the static caching scheme does not require frequent calculation. In Chapter 5, a direct and meaningful objective, instead of calculation simplicity, was preferred.

In the process of solving proposed models, we face the difficulties to obtain the optimal solution between two competing objectives, such as average quality and quality variance in Chapter 4, and bandwidth cost and startup latency in Chapter 5. The general method employing genetic algorithm is computational complex and does not fit our real-time video streaming scenarios where the solution space is relatively small. After considering the pros and cons, we decide to sample one objective dimension with certain grain and derive the Pareto optimal set, from which the final optimal solution could be selected according to different criteria. A recommended solution was obtained with the criterion of Euclidean distance. In this manner, multi-objective optimization was transformed with reduced computational complexity, and the proposed solution can be adjusted easily to fit different scenarios.

In the final stage of the research, we need to validate the proposed solution credibly and comprehensively. Nevertheless, sometimes it is difficult to perform the comparative evaluation because there may not exist a scheme that can be directly compared with to give illustrative results. Our approaches to resolve this problem generally contain the following steps:

- Identify classic and comparable schemes investigated thoroughly
- Utilize video clips/streams well-known to the research community
- Simulate under same environment and with same parameters
- Wrap or combine with similar auxiliary schemes to facilitate evaluation

For example, in Chapter 4 we use three well-known video clips (“Akiyo”, “Container”, “Bus”) as the input streams and compared the proposed scheme with three classic adaptation methods (SFD, FDDC, RQ). In Chapter 5, we compare the proposed caching method with the well-known *MaxRate* strategy by combining them with similar transmission schemes to obtain the bandwidth cost. All the contrasting simulations were performed under environments that were as similar as possible.

In summary, during this research we encounter many difficulties similar to aforementioned ones and we manage to solve most of them with smart methods. Tradeoffs were decided by carefully considering all the pros and cons. Nevertheless, further improvements on this research can still be performed. We give some suggestions on future research in the following section.

6.3 Suggestions for Future Research

In this section, we list some further improvements that can still be performed on this research.

In our adaptive compensation method presented in Chapter 3, we choose a video segment to compensate after network bandwidth resumes. This segment should be fully transmitted or compensated before the client can start to decode it. One aggressive approach is to select a longer segment to compensate such that the client can start to decode the beginning part of this segment before it fully receives the segment. In this way, the video quality will be further improved. In addition, dynamic cache adjustment during the compensation process, which enables clients to adjust its internal buffer usage for several concurrent streaming sessions, has not been addressed in Chapter 3. However, aggressive compensation and dynamic cache adjustment require us to take both average bandwidth and bandwidth variance into account in order to avoid possible playback jitter at client. Furthermore, a more complex optimization model needs to be established and solved.

Our video adaptation scheme is discussed in Chapter 4, where the EL data of each frame is adapted with different bit rates according to frame complexity and scene priority. Given the target bit rate for a sequence of video scenes, we limit the adaptation operation to the frame level. That is, we only preserve individual bit-planes for each frame. We can further investigate how to adapt within each bit-plane such that the adapted stream has a better quality and a closer rate match to the target bit rate. Moreover, we can consider employing more advanced methods to measure video complexity and assess user-perceived quality. Following the same process described in Chapter 4, we can, hence, obtain a new adaptation scheme.

With the proxy caching strategy and scalable transmission scheme proposed in Chapter 5, we aim to optimize bandwidth cost and user access latency. However, in practical video streaming systems, there may be many different objectives to optimize, such as number of channels used by server, peak bandwidth usage within backbone network, and video jitter experienced by user. We can use the process described in Chapter 5 to obtain a similar multi-objective optimization model and attempt to balance the tradeoff among these objectives.

Another topic we can try in the future is to extend these techniques to the H.264 coded videos. The H.264 video coding standard provides outstanding coding performance and receives much attention from academic and industrial area. We can improve the proposed techniques according to the new characteristics introduced by scalability extension of H.264. Moreover, it is worth noting that the proposed algorithms to solve optimization models in Chapter 4 and Chapter 5 can be further refined in terms of complexity and optimality. These topics can be studied in the future.

Bibliography

[1] Ahmad, S., Gohar, N. D., Kamal, A., “A Dynamic Congestion Control Mechanism for Real Time Streams over RTP,” in *Proceedings of the 9th International Conference on Advanced Communication Technology*, vol. 2, pp. 961-966, Feb. 2007.

[2] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., “RTP: A Transport Protocol for Real-Time Applications,” RFC-3550, July 2003.

[3] Dapeng Wu, Hou, Y. T., Wenwu Zhu, Hung-Ju Lee, Tihao Chiang, Ya-Qin Zhang, Chao, H.J., “On End-to-end Architecture for Transporting MPEG-4 Video over the Internet,” *IEEE Trans. on Circuits System and Video Technology.*, vol. 10, no. 6, pp. 923-941, Sep. 2000.

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-time Applications,” Internet Engineering Task Force, RFC 1889, Jan. 1996.

[5] Lihong Xu, Shuhua Ai, “A New Feedback Control Strategy of Video Transmission Based on RTP,” in *Proceedings of IEEE Conference on Industrial Electronics and Applications*, pp. 1-4, May 2006.

[6] M. Degermark, “Requirements for Robust IP/UDP/RTP Header Compression,” Internet Engineering Task Force, RFC 3096, July 2001.

[7] J. Ott, C. Bormann, G. Sullivan, S. Wenger, R. Even Ed., “RTP Payload Format for ITU-T Rec. H.263 Video,” Internet Engineering Task Force, RFC 4629, Jan. 2007.

[8] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, P. Gentric, “RTP Payload Format for Transport of MPEG-4 Elementary Streams,” Internet Engineering Task Force, RFC 3640, Nov. 2003.

[9] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer, “RTP Payload Format for H.264 Video,” Internet Engineering Task Force, RFC 3984, Feb. 2005.

[10] Changwoo Jee, Shin, K.G., “A DAVIC Video-on-Demand System Based on the RTSP,” in *Proceedings of Symposium on Applications and the Internet*, pp. 231-238, Jan. 2001.

[11] H. Schulzrinne, A. Rao, R. Lanphier, “Real Time Streaming Protocol (RTSP),” Internet Engineering Task Force, RFC 2326, Apr. 1998.

[12] Jinyao Yan, Katrinis K., May M., Plattner B., “Media and TCP-friendly Congestion Control for Scalable Video Streams,” *IEEE Trans. on Multimedia*, vol. 8, no. 2, pp. 196-206, April 2006.

[13] Vieron J., Guillemot C., “Real-time Constrained TCP-compatible Rate Control for Video over the Internet,” *IEEE Trans. on Multimedia*, vol. 6, no. 4, pp. 634-646, Aug. 2004.

[14] M. Handley, S. Floyed, J. Padhye, J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol specification,” Internet Request for Comments 3448, Jan. 2003.

[15] Ohm, J.-R., “Advances in Scalable Video Coding,” in *Proceedings of the IEEE*, no.1 vol. 93, pp. 42-56, Jan. 2005.

[16] Sikora, T., “Trends and Perspectives in Image and Video Coding,” in *Proceedings of the IEEE*, no.1 vol. 93, pp. 6-17, Jan. 2005.

[17] Sullivan G. J., Wiegand T., “Video Compression – From Concepts to the H.264/AVC Standard,” in *Proceedings of the IEEE*, no.1 vol. 93, pp. 18-31, Jan. 2005.

[18] “Video Codec for Audiovisual Services at p*64 kbit/s,” Int. Telecomm. Union-Telecomm. (ITU-T), Geneva, Switzerland, Recommendation H.261.

[19] “Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s, Part 2: Video,” Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC), ISO/IEC 11 172-2.

[20] “Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video,” Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC), ISO/IEC 13 818-2.

[21] “Coding of Audiovisual Objects—Part 2: Visual,” Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC), 14496-2.

[22] “Coding of Audiovisual Objects—Part 10: Advanced Video Coding,” Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC), ISO/IEC 14 496-10.

[23] T. Kim, M. Ammar, “Optimal Quality Adaptation for MPEG-4 Fine-grained Scalable Video,” in *Proceedings of IEEE Infocom*, vol. 1, pp. 641-651, Apr. 2003.

[24] Chatterje, M., Sengupta, S., Ganguly, S., “Feedback-based Real-time Streaming over WiMax,” *IEEE Wireless Communications*, vol. 14, no. 1, pp. 64-71, Feb. 2007.

[25] Kyungtae Kang, Yongwoo Cho, Jinsung Cho, Shin, H., “Scheduling Scalable Multimedia Streams for 3G Cellular Broadcast and Multicast Services,” *IEEE Trans. on Vehicular Technology*, vol. 56, no. 5, pp. 2655-2672, Sep. 2007.

[26] Ying Wai Wong, Lee, J.Y.B., Li, V.O.K., Chan, G.S.H., “Supporting Interactive Video-on-Demand With Adaptive Multicast Streaming,” *IEEE Trans. on Circuits System and Video Technology*, vol. 17, no. 2, pp. 129-142, Feb. 2007.

[27] Kusmierek, E., Du, D.H.C.,“Proxy-assisted Periodic Broadcast with Multiple Servers,” in *Proceedings of International Conference on Distributed Computing systems workshops*, pp. 91-96, Mar. 2004.

[28] Jurca, D., Chakareski, J., Wagner, J.-P., Frossard, P., “Enabling Adaptive Video Streaming in P2P Systems,” *IEEE Communications Magazine*, vol. 45, no. 6, pp. 108-114, June 2007.

[29] Tsang, D.H.K., Ross, K.W., Rodriguez, P., Li, J., Karlsson, G., “Advances in Peer-to-Peer Streaming Systems,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1609-1611, Dec. 2007.

[30] Wai-Kong Cheuk, Lun, D.P.-K., Tai-Chiu Hsung, Tsun-Ping To, “Admission Control in Video Streaming Proxy Server,” in *Proceedings of IEEE International Workshop on VLSI Design and Video Technology*, pp. 322-325, May 2005.

[31] Lixin Gao, Zhi-Li Zhang, Towsley, D, “Proxy-assisted Techniques for Delivering Continuous Multimedia Streams,” *IEEE Trans. on Networking*, vol. 11, no. 6, pp. 884-894, Dec. 2003.

[32] Songqing Chen, Bo Shen, Wee, S., Xiaodong Zhang, “Designs of High Quality Streaming Proxy Systems,” in *Proceedings of IEEE Infocom*, vol. 3, pp. 1512-1521, Mar. 2004.

[33] Katsaggelos, A.K., Eisenberg, Y., Zhai, F., Berry, R., Pappas, T.N., “Advances in Efficient Resource Allocation for Packet-based Real-time Video Transmission,” in *Proceedings of the IEEE*, vol. 93, no. 1, pp. 135-147, Jan. 2005.

[34] Ganjam, A., Zhang, H., “Internet Multicast Video Delivery,” in *Proceedings of the IEEE*, vol. 93, no. 1, pp. 159-170, Jan. 2005.

[35] Q. Zhang, W. Zhu, and Y. Q. Zhang, "End-to-end QoS for Video Delivery over Wireless Internet," in *Proceedings of the IEEE*, vol. 93, no. 1, pp. 123-134, Jan. 2005.

[36] T. Ahmed, A. Mehaoua, R. Boutaba, Y. Iraqi, “Adaptive Packet Video Streaming over IP Networks: A Cross-layer Approach,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 385-401, Feb. 2005.

[37] T. K. Chiew, P. Ferre, D. Agrafiotis, A. Molina, A.R. Nix, D.R. Bull, “Cross-layer WLAN Measurement and Link Analysis for Low Latency Error Resilient Wireless Video Transmission,” in *Proceedings of IEEE ICCE'05*, pp. 177-178, Jan. 2005.

[38] G. Convertino, D. Melpignano, E. Piccinelli, F. Rovati, F. Sigona, “Wireless Adaptive Video Streaming by Real-time Channel Estimation and Video Transcoding,” in *Proceedings of IEEE ICCE'05*, pp. 179-180, Jan. 2005.

[39] M. U. Demircin, P. van Beek, "Bandwidth Estimation and Robust Video Streaming over 802.11E Wireless LANs," in *Proceedings of IEEE Conference on Multimedia and Expo*, pp. 1250-1253, July 2005.

[40] Shih-Fu Chang, Vetro, A, “Video Adaptation: Concepts, Technologies, and Open Issues,” in *Proceedings of the IEEE*, vol. 93, no. 1, pp. 148-158, Jan. 2005.

[41] Jie Huang, Krasic, C., Walpole, J., Wu-chi Feng, “Adaptive Live Video Streaming by Priority Drop,” in *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 342-347, July 2003.

[42] Carta, M. T., Onali, T., Atzori, L., “Window-based Rate Control Approach for Video Streaming Over Wireless Networks,” in *Proceedings of IEEE International Conference on Communications*, pp. 1772-1777, June 2007.

[43] Yongfeng Li, Kenneth Ong, “Optimized Media Filtering Scheme for Layered Video Streaming,” in *Proceedings of IEEE Conference on Multimedia and Expo*, pp. 711-714, July 2007.

[44] Taehyun Kim, Ammar M.H., “Optimal Quality Adaptation for Scalable Encoded Video,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 344-356, Feb. 2005.

[45] Gibbon, J.F.; Little, T.D.C., “The Use of Network Delay Estimation for Multimedia Data Retrieval,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1376-1387, Sep. 1996.

[46] P. de Cuetos, M. Reisslein, K.W. Ross, “Evaluating the Streaming of FGS–Encoded Video with Rate-Distortion Traces” (http://www.eurecom.fr/decuetos), June 2003.

[47] Schwarz H., Marpe D., Wiegand T., “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Trans. on Circuits System and Video Technology*, vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[48] Jun Sun, Wen Gao, Debin Zhao, Qingming Huang, “Statistical Model, Analysis and Approximation of Rate-distortion Function in MPEG-4 FGS Videos,” *IEEE Trans. on Circuits System and Video Technology*, vol. 16, no. 4, pp. 535-539, Apr. 2006.

[49] B. Li, J. Errico, H. Pan, I. Sezan, “Bridging the Semantic Gap in Sports Video Retrieval and Summarization,” *Journal of Visual Communication and Image Representation*, vol. 15, no. 3, pp. 393-424, Sep. 2004.

[50] D. Zhang, S.-F. Chang, “Event Detection in Baseball Video Using Superimposed Caption Recognition,” in *Proceedings of ACM International Conference on Multimedia*, pp. 315-318, Dec. 2002.

[51] Y. Wang, S.-F. Chang, A. C. Loui, “Subjective Preference of Spatio-temporal Rate in Video Adaptation using Multi-dimensional Scalable Coding,” In *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 3, pp. 1719-1722, June 2004.

[52] J. Xin, C.-W. Lin, M.-T. Sun, “Digital Video Transcoding,” in *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84-97, Jan. 2005.

[53] Feng Pan, Z. Li, K. Lim, and G. Feng, “A Study of MPEG-4 Rate Control Scheme and its Improvements”, *IEEE Trans. on Circuits System and Video Technology*, vol. 13, no. 5, pp. 440-446, May 2003.

[54] Wen-Nung Lie, Cheng-Hsiung Tseng, Lin, T.C.-I., “Constant-Quality Rate Allocation for Spectral Fine Granular Scalable (SFGS) Video Coding by Using Dynamic Programming Approach,” in *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 1, pp. 655-658, Jun. 2004.

[55] T. Kim, M. Ammar, “Optimal Quality Adaptation for MPEG-4 Fine-grained Scalable Video,” in *Proceedings of IEEE Infocom*, pp. 641-651, Apr. 2003.

[56] L. Zhao, J. Kim, C.-C. J. Kuo, “MPEG-4 FGS Video Streaming with Constant-quality Rate Control and Differentiated Forwarding,” in *Proceedings of SPIE Conference on Visual Communications and Image Processing*, pp.

230–241, Jan. 2002.

[57] Y. Sun, J. Luo, I. Ahmad, "Rate Control for Multi-object Video Transmission over Wireless Systems," in *Proceedings of IEEE International Conference on ICC*, vol. 12, pp. 5463-5467, June 2006.

[58] Ahmed, T., Mehaoua, A., Lecuire, V., "Streaming MPEG-4 Audiovisual Objects Using TCP-friendly Rate Control and Unequal Error. Protection," in *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 2, pp. 317-320, July 2003.

[59] Iain E.G. Richardson, "H.264 and MPEG-4 Video Compression," Wiley and Sons, Ltd. ISBN: 0-470-84837-5, 2003.

[60] W.P. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," *IEEE Trans. on Circuits System and Video Technology*, vol. 11, no. 3, pp.301-317, March 2001.

[61] Yongfeng Li, Kenneth Ong, "On Compensation Technique in Multimedia Streaming System," in *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 1, pp. 1524-1527, July 2005.

[62] Weiming Hu, Xie. D., Zhouyu Fu, Wenrong Zeng, Maybank, S., "Semantic-Based Surveillance Video Retrieval," *IEEE Trans. on Image Processing*, vol. 16, no. 4, pp. 1168-1181, April 2007.

[63] Seung Yeob Nam, Sunggon Kim, Junsu Kim, Sung, D. K., "Probing-based Estimation of End-to-end Available Bandwidth," *IEEE Communications Letters*, vol. 8, no. 6, pp. 400-402, June 2004.

[64] J. Liu, B. Li, and Y.-Q. Zhang, "Adaptive Video Multicast over the Internet," *IEEE Multimedia*, vol. 10, no. 1, pp. 22-33, Jan. 2003.

[65] Jain, M., Dovrolis, C., "End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP throughput," *IEEE Trans. on Networking*, vol. 11, no. 4, pp. 537-549, Aug. 2003.

[66] Zhonghang Xia, I-Ling Yen, "Proxy Assistant for Streaming Media Delivery," in *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 2, pp. 1331-1334, June 2004.

[67] S. Prabhakar, R. Chari, "Minimizing Latency and Jitter for Large-scale Multimedia Repositories Through Prefix Caching," *International Journal of Image and Graphics*, vol. 3, no. 1, pp. 95-117, Jan. 2003.

[68] B. Wang, S. Sen, M. Adler, D. Towsley, “Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution”, *IEEE Trans. on Multimedia*, vol. 6, no. 2, pp. 366-374, Apr. 2004.

[69] S. Chen, B. Shen, S. Wee, X. Zhang, “Adaptive and Lazy Segmentation Based Proxy Caching for Streaming Media Delivery”, in *Proceedings of NOSSDAV03*, pp. 22-31, June 2003.

[70] Hyung Rai Oh, Hwangjun Song, “Scalable Proxy Caching Algorithm Minimizing Client’s Buffer Size and Channel Bandwidth,” *Journal of Visual Communication and Image Representation*, vol. 17, no. 1, pp. 57–71, Feb. 2006.

[71] L. Shen, W. Tu, E. Steinbach, “A Flexible Starting Point Based Partial Caching Algorithm for Video on Demand”, in *Proceedings of IEEE Conference on Multimedia and Expo*, pp. 76-79, July 2007.

[72] K. L. Wu, P. S. Yu, J. L. Wolf, “Segmentation of Multimedia Streams for Proxy Caching”, *IEEE Trans. on Multimedia*, vol. 6, no 5, pp. 770-780, Oct. 2004.

[73] A. Satsiou, M. Paterakis, “Impact of Frequency-based Cache Management Polices on the Performance of Segment Based Video Caching Proxies,” in *Proceedings of the 3rd IFIP Networking Conference*, pp. 1120-1131, May 2004.

[74] A. Satsiou, M. Paterakis, “Efficient Caching of Video Content to an Architecture of Proxies According to a Frequency-based Cache Management policy”, in *Proceedings of the 2nd International Workshop on Advanced architectures and algorithms for Internet Delivery and Applications,* Oct. 2006.

[75] N. Fonseca, R. eFaCanha, “The Look-ahead-maximize-batch Batching Policy,” *IEEE Trans. on Multimedia*, vol. 4, no. 1, pp. 114–120, Mar. 2002.

[76] Boggia, G., Camarda, P., Mazzeo, L., Mongiello, M, “Performance of Batching Schemes for Multimedia-on-demand Services,” *IEEE Trans. on Multimedia*, vol. 7, no. 5, pp. 920-931, Oct. 2005.

[77] Guo M., Ammar, M.H., “Scalable Live Video Streaming to Cooperative Clients Using Time Shifting and Video Patching,” in *Proceedings of IEEE Infocom*, vol. 3, pp. 1501-1511, Mar. 2004.

[78] Ha Sook-Jeong, Oh Sun-Jin, Bae Ihn-Han, “Practical Patching for Efficient Bandwidth Sharing in VOD Systems,” in *Proceedings of ICNC*, vol. 5, pp. 351-355, Aug. 2007.

[79] Azad S.A., Murshed M., Dooley L.S, "A Novel Scalable Interactive Multiple-rate Staggered Broadcasting Video-on-demand System", in *Proceedings of INCC*, pp. 146-151, Jun. 2004.

[80] Michael K. Bradshaw, Bing Wang, Subhabrata Sen, Lixin Gao, Jim Kurose, Prashant Shenoy, Don Towsley, "Periodic Broadcast and Patching Services - Implementation, Measurement, and Analysis in an Internet Streaming Video Testbed," *ACM Multimedia Systems Journal, Special Issue on Multimedia Distribution*, vol. 9, no. 1, pp. 78-93, July, 2003

[81] Hung-Chang Yang, Hsiang-Fu Yu, Li-Ming Tseng, Yi-Ming Chen, "Interleaving Harmonic Broadcasting and Receiving Scheme with Loss-anticipation Delivery", in *Proceedings of ISCC*, vol. 2, pp. 600-605, July 2004.

[82] Huifang Sun, Anthony Vetro, Jun Xin, "An Overview of Scalable Video Streaming", *Wireless Communications and Mobile Computing*, vol. 7, no. 2, pp. 159-172, Jan. 2007.

[83] J. Liu, X. Chu, J. Xu, "Proxy Cache Management for Fine-grained Scalable Video Streaming," in *Proceedings of IEEE Infocom,* vol. 3, pp. 1490-1500, Mar. 2004.

[84] M. Zink, J. Schmitt and R. Steinmetz, "Layer-encoded Video in Scalable adaptive streaming," *IEEE transactions on Multimedia*, vol. 7, no. 1, pp. 75-84, Feb. 2005.

[85] Yiu-Wing Leung, Tony K. C. Chan, "Design of an Interactive Video-on-demand System," *IEEE Transactions on Multimedia,* vol 5, no. 1, pp130-140, March 2003.

[86] Xu Du, Tai Wang, Zongkai Yang, Jiang Yu, Wei Liu, "Variable Rate Caching for Video Delivery in Heterogeneous Environment", in *Proceedings of IEEE ICC*, vol. 3, pp. 1046-1051, June 2006.

[87] C. P. Costa, I. S. Cunha, Alex Borges, *et al.*, "Analyzing Client Interactivity in Streaming Media," in *Proc. of WWW 2004*, May 2004.

[88] E. Veloso, V. Almeida, Jr. W. Meira, A. Bestavros, Shudong Jin, "A Hierarchical Characterization of a Live Streaming Media Workload," *IEEE/ACM Trans. on Networking,* vol. 14, no. 1, pp. 133-146, Feb. 2006.

[89] L. Cherkasova, M. Gupta, "Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Dynamics, and Rate of Change," *IEEE/ACM Trans. on Networking,* vol. 12, pp. 781-794, Oct. 2004.

Publication List

Yongfeng Li, Kenneth Ong, "Optimized scalable cache management for video streaming system," in print, Journal of Multimedia Tools and Applications, 2009.

Yongfeng Li, Kenneth Ong, "Optimized cache management for scalable video streaming," in *Proceedings of ACM Conference on Multimedia*, pp. 799-702, Sep. 2007.

Yongfeng Li, Kenneth Ong, "Optimized Media Filtering Scheme for Layered Video Streaming," in *Proceedings of IEEE Conference on Multimedia and Expo*, pp. 711-714, July 2007.

Yongfeng Li, Kenneth Ong, "On Compensation Technique in Multimedia Streaming System," in *Proceedings of IEEE Conference on Multimedia and Expo*, vol. 1, pp. 1524-1527, July 2005.