Application Layer System Design for Layered Video Streaming

D I S S E R T A T I O N

for the Degree of

Doctor of Philosophy (Electrical Engineering)

Hao Hu

January 2012

Application Layer System Design for Layered Video Streaming

D I S S E R T A T I O N

Submitted in Partial Fulfillment
of the REQUIREMENTS for the

Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

POLYTECHNIC INSTITUTE OF NEW YORK UNIVERSITY

by

Hao Hu

January 2012

Approved:

_____
Department Head

_____
Date

Copy No. ____

Approved by the Guidance Committee:

Major: Electrical and Computer Engineering

Yao Wang
Professor of
Electrical and Computer Engineering

Yong Liu
Associate Professor of
Electrical and Computer Engineering

Xiaoqing Zhu
Ph.D. Member of
Advanced Architecture & Research
Cisco Systems Inc.

Minor: Computer Science

Keith W. Ross
Leonard J. Shustek Professor of
Computer Science and Engineering

VITA

Hao Hu was born in China on November 16, 1983. He received the B.E. degree in Electrical Engineering from Nankai University and M.E. degree in Electrical Engineering from Tianjin University in 2005 and 2007, respectively. Since August 2007, he has been a Ph.D. candidate at Electrical Engineering Department in Polytechnic Institute of New York University, Brooklyn, NY 11201 under the supervision of Prof. Yao Wang and Prof. Yong Liu. From June 2008 to September 2008, he interned at the Corporate Research, Thomson Inc. (now Technicolor). He visited Cisco Systems Inc. during June 2010 to September 2010 as a student consultant and from May 2011 to August 2011, he was a cisco choice intern. During his thesis study, he published 5 technical papers and had 2 patents pending. He has conducted the research in the fields of P2P networking, scalable video complexity modeling, quality-rate modeling and wireless video adaptation.

To my wife Ningning and our parents
for their love and support.

ACKNOWLEDGEMENT

First, I would like to gratefully and sincerely thank my thesis advisors Professor Yao Wang and Professor Yong Liu for their invaluable support, inspiration, patience and guidance throughout the my research at Polytechnic. I have learned a lot from their insightful comments and constructive criticisms which discovered my faltered steps and would continue to benefit my future career. I am also indebted to other committee members of this dissertation. Professor Keith W. Ross and Dr. Xiaoqing Zhu have closely followed my research. Their ways of tackling problems and thinking in big pictures also influenced me a lot.

I would also like to thank Dr. Zhengye Liu, Dr. Yang Guo and Dr. Zhan Ma, who worked with me closely on many projects through my research. The discussions with them are always inspiring and helpful. I am also very grateful to my colleagues at Thomson Corporation Research and Cisco Systems, Inc.: Dr. Yunfei Zheng, Dr. Rong Pan, Dr. Jiang Zhu, Dr. Flavio Bonomi for their warm-hearted help and insightful technical discussion. Thanks to the members of the Video Lab at Polytechnic: Xuan Zhao, Zhili Guo, Yen-Fu Ou, Dr. Xiaozhong Xu, Dr. Xinggong Zhang, Dr. Ozgu Alay, Cagdas Bilen, for their collaboration, discussion and activities which brightened my graduate life. I also want to thank other members and friends at Polytechnic: Dr. Yanming Shen, Dr. Pei Liu and Dr. Chao Liang, Dr. Chao Zhang, Sha Hua, Xiwang Yang and Yang Xu. Because of all of them, my life at Poly is much more enjoyable.

In particular, I want to thank my dear wife, Ningning Wang, for her continuous caring and love. Without her, the achievements of my thesis are impossible. Finally, I would like to thank our parents for their persistent support and encouragement.

This work is supported by National Science Foundation under Grant No. 0933985, and partially supported by GIFT award from Cisco Inc. and the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic Institute of New York University, Brooklyn, New York, USA.

AN ABSTRACT

Application Layer System Design for Layered Video Streaming

by

Hao Hu

Advisors: Yao Wang and Yong Liu

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy (Electrical Engineering)

January 2012

Given the ever-increasing Internet video demands, how to handle the large video traffic over the current network is an active research area. The main objective of this thesis is to improve end users' video experience by considering application layer system optimization, scalable video coding and Peer-to-Peer technology.

Scalable Video Coding (SVC) is well positioned in the low-cost Peer-to-Peer (P2P) video streaming network. We model such system in utility maximization framework to understand the interplay between system efficiency, user fairness and incentive. We propose to use taxation mechanisms to strike the right balance between them. The proposed scheme guarantees the streaming rate of a node is at least proportional to its uploading contribution. The left-over uploading bandwidth (taxed portion) from all nodes is used to help "slow" peers, therefore improves the social welfare. The tax rate works as a knob controlling the tradeoff between the three aspects. We then propose practical taxation-based P2P layered streaming designs and validate its performance with extensive trace-driven simulations.

To facilitate SVC-based system design, we propose a rate-quality model for scalable video with temporal and amplitude scalability that can be easily incorporated into a net-

work maximization framework. The associated optimal layer extraction is done by layer ordering within each Group of Pictures (GOP) such that each additional layer provides the best possible quality gain over the bitrate increment. The proposed rate-quality model and layer ordering scheme are evaluated with various sequences ranging from slow to intensive motion.

Based on the proposed rate-quality model, we investigate a proxy-based solution for adapting the SVC streams at the edge of a wireless network. The proxy iteratively allocates rates of different video streams to maximize a weighted sum of video qualities. Given the rate allocation, appropriate video layers are transmitted to end users. Simulation studies show that our scheme consistently outperforms TCP-friendly rate control (TFRC) in terms of agility to track link qualities and overall subjective quality. In addition, the proposed scheme supports differential services and competes fairly with TCP flows.

Recent industrial effort has been on streaming video using HTTP protocol. Typically, HTTP streaming relies on clients to request appropriate video chunks based on local playback status and bandwidth estimation. To facilitate the chunk request strategy design, we propose a cost function for each chunk that jointly considers its contribution to the average playback quality, the variation of playback quality and the playback buffer level. We then show that finding the optimal chunk request strategy is a dynamic programming problem. Due to the difficulty of predicting the underlying network dynamics and the complexity of solving the exact problem, we propose a simple heuristic algorithm to solve it. We implement a testbed to evaluate its performance. Furthermore, we compare the HTTP live streaming system with TCP-based SVC live streaming to show the advantage of employing SVC into the system.

List of Publications

1. Hao Hu, Zhan Ma and Yao Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation," *submitted to IEEE International Conference on Image Processing (ICIP)*, 2012.

2. Hao Hu, Xiaoqing Zhu, Yao Wang, Rong Pan, Jiang Zhu and Flavio Bonomi, "Proxy-based Multi-stream Scalable Video Adaptation over Wireless Networks using Subjective Quality and Rate Models," *submitted to IEEE Trans. on Multimedia*, Nov. 2011.

3. Hao Hu, Xiaoqing Zhu, Yao Wang, Rong Pan, Jiang Zhu and Flavio Bonomi, "QoE-based Multi-Stream Scalable Video Adaptation over Wireless Networks with Proxy," *submitted to IEEE International Conference on Communications (ICC) workshop on Realizing Advanced Video Optimized Wireless Networks*, 2012.

4. Xinggong Zhang, Hao Hu, Zongming Guo, Yong Liu and Yao Wang, "Video Streaming on Multi-path Networks with Cross-path FEC," *prepared for IEEE Trans. on Circuits and Systems for Video Technology*, Dec. 2011, to be submitted.

5. Xinggong Zhang, Yang Xu, Hao Hu, Yong Liu, Zongming Guo and Yao Wang, "Profiling Skype Video Calls: Rate Control and Video Quality," *Proc. of IEEE INFOCOM*, 2012.

6. Zhan Ma, Hao Hu and Yao Wang, "On Complexity Modeling of H.264/AVC Video Decoding and Its Application for Energy Efficient Decoding," *IEEE Trans. on Multimedia*, vol 12, no. 6, Dec. 2011, pp.1240-1255.

7. Hao Hu, Yang Guo and Yong Liu, "Peer-to-Peer Streaming of Layered Video: efficiency, fairness and incentive," *IEEE Trans. on Circuits and Systems for Video Technology*, vol 21, no. 8, Aug. 2011, pp. 1013-1026.

8. Zhengye Liu, Hao Hu, Yong Liu, Keith Ross, Yao Wang and Markus Mobius, "P2P Trading in Online Social Networks: the Value of Staying Connected," *Proc. of IEEE INFOCOM*, 2010.

9. Hao Hu, Yang Guo and Yong Liu, "Mesh-based Peer-to-Peer Layered Video Streaming with Taxation," *Proc. of ACM NOSSDAV*, 2010.

10. Hao Hu, Yang Guo and Yong Liu, "Mesh-based Peer-to-Peer Layered Video Streaming with Taxation," *IEEE ComSoc MMTC e-letter*, pp. 24-29, Nov. 2010.

11. Zhan Ma, Hao Hu and Yao Wang, “DVFS-enabled energy efficient video decoding,” *IEEE ComSoc MMTC e-letter*, pp. 20-24, July 2010.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Challenge, Motivation and Our Approach

Nowadays, video traffic has already accounted for over 40 percent of consumer Internet traffic, and the percentage is growing exponentially for the upcoming years according to Cisco Visual Networking Index [1]. How to handle the rapid increment of video demand over the current network infrastructure has been an active research area for years. However, in order to keep up with the Internet Multimedia era, more work needs to be done.

1.1.1 Peer-to-Peer Video Streaming

Conventionally, Internet video is served from dedicated content servers to end-users, e.g., Content Delivery Networks (CDN). Yet, server infrastructure is costly to meet the video demand in terms of bandwidth costs and server hardware costs. Peer-to-Peer (P2P) technology is attractive in the sense that it is a low cost alternative to stream video to a large number of subscribers who are, both video consumers and suppliers. As each peer contributes its resources, e.g., bandwidth, computational power, storage, the capacity of the system scales up at the same pace as the number of users increases. Adoptions of the P2P technology can be found in several commercial video streaming systems: PPlive [2], PPstream [3], UUsee [4], just to name a few. It is also widely employed to offload the traditional server-client based video systems [5, 6].

Despite the success of P2P in video streaming, there are still limitations on current solutions in terms of their robustness, scalability and performance [7]. Current systems, e.g., PPlive, PPstream, typically employ single layered video coding structure, therefore all peers will receive the same quality regardless of their resource contributions and bandwidth constraints. In addition, any packet loss can lead to severe video quality degradation. Scalable video coding (SVC) [8] is well positioned to overcome these limitations of the current solutions. SVC encodes video into progressive layers. The base layer can be independently decoded, while higher layers are decodable only if the layers beneath have been decoded [1]. The video quality perceived by a peer increases as the number of decodable layers increases. While multiple-layer coding generally incurs coding overhead, recent advance in SVC coding has brought down the overhead to 10% [9]. It is now practical to adopt SVC into P2P video streaming to extend its design space.

The adoption of SVC into P2P streaming faces two key design challenges:

- *layer subscription: how many layers each peer should receive?*
- *layer scheduling: how to deliver to peers the layers they subscribed?*

From the system point of view, the most *efficient* solution is to maximize the aggregate video quality perceived by all peers, i.e, to optimize the *social welfare*. From individual peer point of view, the solution should be *fair*. However, in P2P streaming, due to the dual server-consumer role of peers, the notion of fairness is much more subtle than that in the traditional server-client systems, where clients are only considered as resource consumers. A solution allocating the same video quality to all peers regardless of their contributions would not be considered as fair, and therefore would not provide *incentives* for peers to contribute. A good layered P2P streaming solution has to strike the right balance between the efficiency, fairness and incentive.

We develop analytical models and practical streaming designs to understand and control the interplay between the efficiency, fairness and incentive in layered P2P streaming.

[1] For simplicity, we assume the one-dimension layer structure, i.e., layer $i+1$ depends on layer $i, i-1, \ldots, 1$. Any SVC stream can be ordered into this layer structure as will be discussed in Chapter 3.

Specifically, we develop network-coding based utility maximization models to obtain the most efficient layered streaming solution. The choice of peer utility function reflects the target fairness among peers when they are considered only as video consumers. To incorporate contribution-awareness, we adopt *taxation* as a peer-incentive mechanism and augment the utility maximization models to make the solution incentive-compatible. The proposed taxation-based scheme guarantees the streaming rate of a node is at least proportional to its uploading contribution. The left-over uploading bandwidth (taxed portion) from each node is used to help "slow" peers, therefore improves the social welfare [2]. The tax rate works as a knob controlling the tradeoff between the system efficiency, fairness and incentives. Then, we develop practical taxation-based P2P layered streaming designs, including layer subscription strategy, chunk scheduling policy, and mesh topology adaptation. With extensive trace driven simulations, we show that the proposed layered streaming designs can effectively drive P2P streaming systems to the desired operating points in a distributed fashion.

1.1.2 SVC Rate-Quality Modeling and Layer Ordering

The latest SVC standard [10] defines three types of scalabilities, i.e., temporal scalability, amplitude scalability and spatial scalability, among which the first two are commonly used. Temporal scalability allows the SVC bitstream to be extracted at different frame rate (FR) while amplitude scalability allows different quantization stepsize (QS). By combined usage of temporal scalability and amplitude scalability, a wide bitrate range (with a factor of more than 10) is provided.

Due to its low complexity and flexibility, SVC has been advocated for video adaptation to network capabilities [11]. However, two fundamental and challenging problem are yet answered:

- *What is the proper utility function for SVC streams?*

[2] Usually, the utility function for each peer is concave and as a results, the total system-wide utility increases.

- *Given a target bit rate, how to determine at which temporal resolution (i.e., frame rate) and amplitude resolution (i.e., quantization stepsize (QS)) the SVC (sub)stream should be sent?*

The first problem is critical, because if we had a good utility function for SVC streams, then the network can be optimized in a network utility maximization (NUM) framework [12] and the optimal SVC rate can be determined. Traditional video quality metric, e.g. peak signal-to-noise ratio (PSNR), does not accurately characterize the true QoE of a SVC coded video with temporal scalability. Therefore, we can not directly use rate-distortion (RD) model as a utility function. Instead, we try to derive a rate-quality (RQ) model that directly relates the video rate with its perceptual quality.

Given a sending rate for SVC stream, one may send the video at a high frame rate, but high QS, yielding noticeable coding artifacts in each frame. Or one may use a low frame rate, but small QS, producing high quality frames with possible jittering. These and other combinations can lead to very different perceptual quality. Ideally, there is optimal combination of FR and QS that leads to the best perceptual quality, while satisfying the target bit rate. Formally, the problem is formulated as

$$\max \quad Q \text{ subject to } R \leq R_0, \tag{1.1}$$

where Q and R stand for video quality and video rate, R_0 is the rate constraint.

By leveraging the parametric models from the prior work [13] which explicitly account for the impact of FR and QS on rate and subjective quality of the scalably encoded stream, we are able to determine the optimal combination of FR and QS solving (1.1). A set of equations are derived to relate the optimal FR f^*, the optimal QS q^*, the optimal perceptual quality Q^* and the rate constraint R_0. Then, we numerically find the optimal rate-quality tradeoff curve $Q^*(R)$ and propose a simple rate-quality model that correlates very well with the curve and can be easily incorporated into network utility maximization framework.

To solve the second problem, we propose a quality optimized layer ordering strategy. The ordering is based on the rate and quality models in [13]. The resulting ordered stream has the property that each additional layer offers maximum quality improvement for the rate

increment. At the same time, layer decoding dependency is satisfied. The layer ordering can be easily implemented in the network (at proxy/CDN node) or at the original server. With such a ordered SVC stream, the proxy/server can simply keep sending additional layers, until the rate target is reached.

1.1.3 Proxy-Based SVC Streaming over Wireless Networks

Recent years have seen a proliferation of smart phones and constant bandwidth upgrades in broadband mobile networks. These two factors combined have fueled the rapid growth of mobile media traffic. The study in [1] predicts that by 2015, two-thirds of world's mobile data will be video. On the other hand, mobile media streaming remains a daunting task, especially for users in a highly dynamic environment. The presence of heterogeneous access networks and high user mobility contribute to the wide fluctuations of wireless link qualities in terms of their throughputs and latencies. As multiple video streaming sessions share the same access node (e.g., a cellular base station or a WiFi access point), the system also needs to allocate wireless channel resources wisely among competing traffic flows. It is therefore of crucial importance to have an effective video rate adaptation scheme to strive for the best possible viewing experience of individual users in face of wide link quality fluctuations and dynamic network traffic patterns.

The challenges are multifold. First, rate adaptation for streaming video needs to closely track fluctuations in the available wireless link bandwidth. Conventional techniques such as TCP-friendly rate control (TFRC) [14], however, typically rely on end-to-end packet statistics and fall behind abrupt changes in the underlying network conditions. Second, existing approaches achieve fairness by allocating equal rates to all competing flows, whereas video streams naturally differ in their utilities of rate depending on their contents. For instance, it would be desirable for an action movie sequence to be streamed at a higher rate than a head-and-shoulder news clip competing over the same bottleneck wireless link. Such *content-aware* allocation is missing in today's systems. Thirdly, clients connecting to the same access node may experience different throughputs over their respective wireless links, due to factors such as distance and channel fading characteristics.

Without proper in-network information, rate adaptation decisions made at the senders can easily lead to inefficient resource sharing. More specifically, packet transmissions over a low-quality wireless link can block the access node from adequately serving other streams over higher-quality links, a problem commonly known as head-of-line blocking [15].

We address the above issues in a novel rate adaptation scheme for streaming video over a highly dynamic environment. Our design introduces a proxy at the edge of the network, right where congestion over the wireless links occurs. This allows the rate adaptation module to constantly monitor the bottleneck buffer level, which, in turn, reflects variations in the throughput and delay of wireless links for all receivers. To strike a balance between computational complexity and efficiency, we adopt the latest SVC standard [10] for lightweight in-network rate adaptation.

The goal of the video adaptation module at a proxy node is to maximize the overall viewing experience of all traversing streams. We show that the problem can be decomposed into two steps: i) to allocate the video rate for each stream based on their respective rate-quality relations and wireless link throughputs and the common bottleneck buffer level; and ii) to extract video packets belonging to the appropriate temporal and amplitude layers from each scalable video stream based on the allocated rate. Given the rate-quality model we have derived, the first subproblem of multi-stream rate allocation is solved by maximizing the weighted sum of user qualities under a total network utilization constraint. We propose an iterative solution, whereby the per-stream rate is calculated based on periodic updates of bottleneck buffer level and relative link throughputs. The second subproblem can be solved using the SVC layer ordering strategy, so that each additional layer offers maximum quality improvement for the rate increment. The proxy can simply keep sending additional layers, until the rate target is reached.

Extensive simulation studies confirm that the proposed scheme consistently outperforms conventional TFRC-based rate adaptation used in the Datagram Congestion Control Protocol (DCCP) [16]. More specifically, our scheme adapts more swiftly in the presence of abrupt changes in wireless link throughputs and delays. For the case of multiple streams with heterogeneous contents and link conditions sharing the same wireless access node, the

proposed scheme leads to higher overall subjective quality experience by all viewers than TFRC. Furthermore, it is flexible enough to support differentiated service for video streams with different user-specified importance levels.

1.1.4 Adaptive HTTP Video Streaming

Recent industrial trend is streaming video content over HTTP. Progressive downloading, a special HTTP video streaming scheme, has been very attractive for Internet multimedia delivery due to its simplicity. Clients download the media chunks progressively from the standard HTTP Web servers and can seek to desired positions in the media file by performing byte-range requests. The play-out starts as soon as enough necessary data is retrieved and buffered. YouTube [17], among others, now uses this approach to deliver its huge pool of video contents. However, there are some disadvantages of progressive downloading:

- It is not bitrate adaptive. The user will have to buffer a long time before the playback starts or experience frequent freezes and re-buffering in case of insufficient access bandwidth.

- It can hardly support live streaming as the content needs to be preloaded and it is not possible to combat the bandwidth mismatch between content and network.

Adaptive HTTP streaming is a hybrid of progressive download and streaming. Like the progressive downloading approach, it relies on clients sending pull signals to the HTTP web server to request the media segments, thus relieving the server load. Unlike the progressive downloading approach, clients are allowed to switch between different versions of the content encoded with different bitrates with sophisticated segments generation design. The client player strives to always retrieve the next best segment after examining a variety of parameters related to available network resources, such as available bandwidth and the state of the TCP connections; device capabilities, such as display resolution and available CPU; and current streaming conditions, such as playback buffer size. The goal

is to provide the best quality of experience by displaying the highest achievable quality, starting up fast and reducing skips, freezes, and stutters. Seamless switching is achieved by time-aligning different versions of the content. This approach addresses the weaknesses of progressive downloading, and thus is advocated by several companies and organizations, e.g. Microsoft, Apple, Adobe, MPEG, 3GPP, etc.

The Quality of Experience of this sequential requests and downloads process directly hinges on the client's requesting strategy. If the client requests chunks too aggressively, e.g., with high bitrate, then he/she may experience video freeze occasionally; if the client requests chunks too conservatively, then the available bandwidth is wasted and the video quality is low. To facilitate the chunk request strategy design, we first define a cost function for each chunk by jointly considering its contribution to the average playback quality, the variation of playback quality and the playback buffer level. Assuming that the cost is additive over all video chunks, the problem of finding the optimal chunk request strategy is de facto a Dynamic Programming problem. Due to the difficulty of predicting the underlying network dynamics and the complexity of solving the exact problem, we resort to a simple heuristic algorithm to approach it. The next chunk request is determined at the finishing time of the previous chunk request, with play buffer fullness, throughput estimation, chunk size as inputs. We implement a testbed with Apache HTTP server, DummyNet and VLC to evaluate the algorithm's performance in terms of average playback quality and quality variations. In addition, we compare the adaptive HTTP streaming system with a TCP-based SVC adaptive streaming scheme to show the potential advantage of employing SVC into the system. In TCP-based SVC adaptive streaming, the server transmits as many video packets as possible within each chunk in their order of importance in the allowed time for this chunk; the allowed time is set to be a fraction of the buffered video length at the client (the server can easily estimate it), so that the buffer underflow can be avoided. As this scheme allows partial transmission of the video chunk, it is more flexible than current HTTP adaptive streaming which offers no such freedom. It also alleviates the need for the client to request appropriate chunks. Simulation results show that TCP-based SVC adaptive streaming offers a higher average playback rate and faster convergence speed than

adaptive HTTP streaming.

1.2 Dissertation Outline

This thesis is organized as follows. In Chapter 2, we introduce utility maximization models to study the interplay between efficiency, fairness and incentive in layered P2P streaming. Then we propose a taxation mechanism to adjust the balance between the social welfare and individual peer welfare. We develop practical taxation-based P2P layered streaming designs, including layer subscription strategy, chunk scheduling policy, and mesh topology adaptation. We then evaluate the designs with extensive trace driven simulations.

In Chapter 3, we first review the rate and subjective quality models for SVC with temporal and amplitude scalability. Then we analytically derive the relations between optimal subjective quality, optimal frame rate, optimal quantization stepsize given a bitrate. We obtain the rate-quality model numerically and discuss its application to utility maximization problems. A rate-quality optimized layer ordering scheme for SVC layers is presented and evaluated, utilizing the rate and quality models.

In Chapter 4, we develop a quality maximization framework for rate allocation among multiple wireless receivers under the same wireless access node. An iterative algorithm is proposed to approach the optimal solution. Then, we discuss practical system design and evaluate our system design with extensive simulations.

In Chapter 5, we first review existing adaptive HTTP streaming system. Then, we propose a cost function for each chunk that jointly considers its contribution to the average playback quality, the variation of playback quality and the playback buffer level. Next, we convert the problem of finding the optimal chunk request strategy for all chunks of a video into a Dynamic Programming problem and propose a heuristic algorithm to solve it. We develop a testbed based on which various system settings are evaluated. We also compare the adaptive HTTP streaming of single layer video, with a proposed TCP-based SVC streaming scheme with simulation results.

Chapter 6 concludes the thesis and discusses some future research directions.

Chapter 2

Peer-to-Peer Streaming of Layered Video Considering Efficiency, Fairness and Incentives Through Taxation

In this chapter, we develop utility maximization models to understand the interplay between the efficiency, fairness and incentive in layered P2P streaming. We show that taxation mechanisms can be devised to strike the right balance between social welfare and individual peer welfare. We then develop practical taxation-based P2P layered streaming designs, including layer subscription strategy, chunk scheduling policy, and mesh topology adaptation. Finally, we evaluate the proposed designs with extensive trace-driven simulations.

2.1 Background

2.1.1 Social Welfare vs. Individual Welfare

The majority of existing P2P live streaming systems can be classified into two groups: one group targeting at maximizing the aggregate received video quality, namely social welfare; the other striving for the fairness among peers, or individual welfare. Fig. 2.1 depicts the relationship between these two groups of strategies, which have contradicting goals that are not attainable simultaneously.

Taxation based incentive mechanism [18, 19] offers a flexible framework that allows the tradeoff between individual users' fairness/welfare and the system-wide social welfare

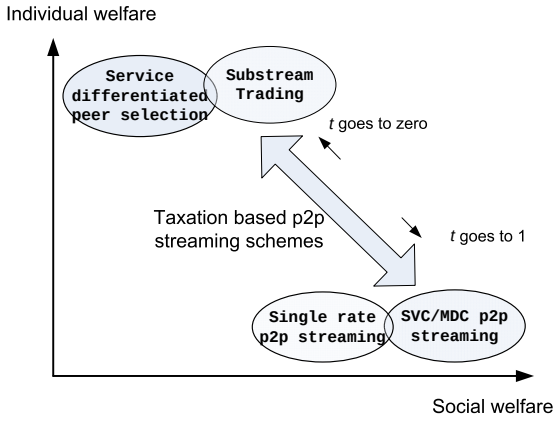


Figure 2.1: Social welfare vs. personal welfare. The tax rate t determins the balance between social welfare and personal welfare.

(see Fig. 2.1). Let u_d be the upload bandwidth contributed by user d. Under a tax rate $0 \leq t \leq 1$, the target received video rate of user d is $r_d = (1-t)u_d + \frac{t}{N}\sum_{i=1}^{N} u_i$, where N is the total number of peers in the system. The received video rate on a peer consists of two parts: a fraction of its own contribution, and a fair share from the pool of taxed bandwidth. The tax rate t adjusts the balance between individual peers' welfare and the social welfare. As t approaches zero, the received video rate approaches the contributed rate, mimicking the 'tit-for-tat' strategy [20, 21]. As t approaches one, the received video rate is the same for all peers, thus achieve the social optimum.

2.1.2 Layered Coding in P2P

Multi-stream coding, such as SVC [8] or MDC [22], allows users to receive the same video with different qualities. A user's perceived video quality is proportional to the number of received video representations. SVC encodes a video into multiple layers with nested dependency: an upper layer becomes decodable only if all layers beneath it have been received and decoded successfully. In contrast, MDC encodes a video into descriptions that are independently decodable. Hence MDC is more flexible than SVC. Our goal, however, is to design a layered P2P protocol that can be implemented and deployed in the field. The current designs of MDC still incur much higher bandwidth overhead than that of SVC. We

thus choose SVC over MDC. We address the challenges of supporting SVC type of inter-correlated layered video in P2P streaming. Note here, throughout the chapter, "layered" implicitly means that the streams have a nested dependency.

Layered coding has been applied to P2P streaming to improve the social welfare over the traditional single layer P2P streaming. For instance, [23] studies how to use layered coding to fully utilize the available peer upload bandwidth in a tree-based P2P overlay multicast. [24] proposes a three-stage chunk scheduling algorithm for mesh-based layered video streaming to achieve high throughput and low video quality jitter. In terms of achieving individual fairness/welfare, Cohen advocated a tit-for-tat algorithm in the seminar paper [21]. [25] proposes a score-based incentive mechanism for P2P live streaming. [26] proposes a service differentiated peer selection algorithm that gives peers with higher contributions more flexibility in choosing neighbors, thus obtain better viewing quality. [27] and [20] utilize layered coding to achieve fairness. Substreams/layers are traded among peers, and peers contributing more are able to receive more in reciprocity. In contrast, we develop taxation mechanisms to strike the right balance between social welfare and individual peers' welfare. The practical taxation-based P2P streaming protocol is designed to drive the system to the desired operating point. In general, a mesh-based P2P streaming topology is more resilient to peer churn than tree-based topology [28, 29]. Therefore, we study mesh-based SVC P2P streaming with taxation.

2.2 Modeling Layered P2P Streaming

To gain insights into layered P2P streaming, we first develop analytical models for layered P2P streaming systems with arbitrary topologies. The models not only allow us to analytically study the interplay between the efficiency, fairness and incentive, but also offer us guidelines in designing the practical system.

2.2.1 Maximizing Efficiency

converge When peers are cooperative, they are willing to contribute their upload bandwidth without any incentive mechanism. The design objective of the system is to maximize the aggregate video quality on all peers. With layered coding, the perceived video quality on a peer is an increasing function of the number of video layers received. PSNR (Peak Signal-to-Noise Ratio) is the standard objective metric to evaluate the quality of a compressed video and thus can be adopted as the utility function in layered video streaming. PSNR of a video coded at rate r_c can be approximated by a logarithmic function $\beta \log(r_c)$ [30], where β is a constant related to the video feature. This approximation is also valid in the SVC case when frame rate is fixed [8]. Let r_d be the total rate of received video layers on peer d. If the aggregate uploading capacity of the server and all peers is U, the aggregate receiving rate on all peers is naturally bounded by $\sum_{d=1}^{N} r_d \leq U$. Since $\log(\cdot)$ is a concave function, the aggregate utility can be maximized when all peers receive video at the same rate, i.e., $r_d = \frac{U}{N}$[1]. For single-layer video streaming, it was shown in [31] that, if peers are fully connected, a two-hop relay streaming can achieve this optimal rate. The solution for layered video streaming naturally follows if we let all peers subscribe to the same number of video layers allowed by the rate $\frac{U}{N}$ and deliver each layer to all peers using the two-hop relay scheme. However, it is unrealistic to have fully connected mesh in a large-scale streaming system. For arbitrary streaming topology, the utility maximization in layered streaming deserves more study.

Network Coding Model

We consider a SVC system where the source server encodes a video stream into L layers with nested dependency. Layer l can be decoded if all the layers below l are received. A peer can subscribe up to k, $k \leq L$, layers. The server multicasts each layer to all peers subscribed to it. There are L simultaneous multicast sessions, one for each layer, in the P2P overlay network.

[1] Throughout this Chapter, we assume peer's downlink bandwidth is always higher than the video steaming rate, thus not the bottleneck.

It is difficult to accurately model a mesh-based P2P streaming system with arbitrary overlay topology due to the content bottleneck problem [32]. Fortunately, by assuming network coding, we can model the system as a closed-form optimization problem for arbitrary topologies with peers' uplink capacity as the constraint. We allow the server and peers apply *network coding* to video blocks. Network coding has been shown to achieve the maximum multicast rate for single multicast session in general network topology [33]. For multiple multicast sessions, *inter-session network coding* might be needed to achieve the maximal multicast rates. However, the complexity of inter-session network coding is generally too high to be justified by its additional performance gain on top of intra-session network coding. In this paper, we only focus on intra-session network coding. The server and peers apply network coding to video blocks in the same layer.

Let a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the overlay topology of the P2P streaming system under study. Let S be the video source server, and $R = V \backslash S$ be the set of peers interested in receiving the video. Let $\vec{x_d} = (x_d^1, x_d^2, \cdots, x_d^L)$ be the binary vector of layers received by peer d: x_d^l equals to 1 if peer d received layer l, 0 otherwise. The video rate for layer l is r^l. To model network coding, we introduce $g_{ij}^{l,d}$ to denote the *information flow* of layer l on link $\langle i, j\rangle \in E$ to destination peer d. For a given peer d and layer l, $\{g_{ij}^{l,d}, \langle i, j\rangle \in E\}$ form a legitimate flow with rate r^l from the source S to d and satisfy the flow conservation on all nodes in the network. Denote by $f_{ij}^l \triangleq \max_d g_{ij}^{l,d}$ the maximum information flow on $\langle i, j\rangle$ for all receivers of layer l. According to the theory of intra-session network coding [33,34], the multicast session for layer l is supportable if and only if a bandwidth of f_{ij}^l is allocated to layer l on link $\langle i, j\rangle$.

We are interested in seeking the optimal P2P streaming solution to maximize the aggregate video experience of all peers. By adopting the PSNR-Rate model, we quantify a user's video experience by a utility function: $F_d(\vec{x_d}) = \beta \log(\sum_{l=1}^{L} x_d^l r^l)$. With notations summarized in Table 2.1, the optimal streaming solution can be found by solving the utility maximization problem **P1**.

Constraint (2.2a) of **P1** guarantees the information flow conservation on each peer. In (2.2b), f_{ij}^l corresponds to the maximum information flow on $\langle i, j\rangle$ for all receivers of layer

Table 2.1: Notations

Notation	Description
V	set of nodes in the system
E	set of overlay links
S	video source server
$R = V \backslash S$	receiving peers
L	number of layers
r^l	rate of layer l
$\vec{x_d} = \{x_d^l\}$	layers received by peer d
$g_{ij}^{l,d}$	information flow of layer l on link $\langle i, j \rangle$ to peer d
f_{ij}^l	bandwidth needed for layer l on link $\langle i, j \rangle$
U_d	peer d's uplink capacity
$F_d(\vec{x_d})$	utility function of peer d

l. In a SVC bitstream, higher layers depend on lower layers, and so peer d may request $l + 1$ layer only if it has received all layers up to l. (2.2c) captures this dependency among layers. (2.2d) is the uplink capacity constraint for all layers on all peers and the server .

2.2.2 Achieving Fairness

In the traditional resource allocation problems, utility maximization achieves different notions of fairness between competing resource consumers. In P2P video streaming, each peer plays a dual-role of server and consumer. We ignore peer's server role in the contribution-oblivious utility maximization. The obtained optimal solution can also be interpreted as fairness among peers without considering their contributions. Within the fairness context, it is straightforward to show that the solution of the utility maximization problem **P1** achieves the *proportional fairness* [35] among peers under the given overlay topology $\mathcal{G}$ and node upload capacity profile $\mathcal{U}$.

Another commonly used fairness measure is the *weighted fairness*. In the context of layered streaming, because of the layer dependency in SVC encoding, peers have to first retrieve lower layers. A solution achieving weighted fairness should give priority

P1: Utility Maximization

Variables:

$g_{ij}^{l,d}$ continuous non-negative variable
f_{ij} continuous non-negative variable
x_d^l binary variable

Objective:

$$\mathbf{max} \sum_{d \in R} \log(\sum_{l=1}^{L} x_d^l r^l) \tag{2.1}$$

Constraints:

$$\sum_{\langle i,j \rangle \in E} g_{ij}^{l,d} - \sum_{\langle j,i \rangle \in E} g_{ji}^{l,d} = \begin{cases} x_d^l r^l, & i = S \\ -x_d^l r^l, & i = d \\ 0, & otherwise \end{cases}$$

$$\forall d \in R, \forall l \leq L \tag{2.2a}$$

$$g_{ij}^{l,d} \leq f_{ij}^l, \quad \forall l \leq L, \forall d \in R, \forall \langle i,j \rangle \in E \tag{2.2b}$$

$$x_d^{l+1} \leq x_d^l, \quad \forall l \leq L, d \in R \tag{2.2c}$$

$$\sum_l \sum_{\langle i,j \rangle \in E} f_{ij}^l \leq U_i, \quad \forall i \in V \tag{2.2d}$$

to streaming lower layers to peers. Weighted fair streaming solution can be obtained by replacing the PSNR-rate utility function in **P1** with a weighted-sum function. We assign weights to layers in a decreasing order. Let w^l be the weight assigned to layer l. We have $w^i > w^j$, if $i < j$. Instead of using PSNR-rate model, the video experience of a peer is characterized by the summation of the weights of all the received layers: $F_d(\vec{x_d}) = \sum_{l=1}^{L} x_d^l w^l$. The marginal gain of receiving a lower layer outweighs that of receiving higher layers. As a result, the optimal solution with the weighted-sum utility function will easily satisfy the constraint (2.2c). If we further relax the binary variables x_d^l in **P1** to continuous variables within $[0, 1]$, the optimal solution will naturally have the property that $x_d^l > 0$ only if $x_d^k = 1, \forall k < l$. Formally, the original non-linear mixed integer programming problem

is relaxed into the following linear programming problem.

P2: Linear Approximation

Variables:

$g_{ij}^{l,d}$ continuous non-negative variable
f_{ij}^{l} continuous non-negative variable
x_d^l $\in [0,1]$, continuous variable

Objective:

$$\max \sum_{d \in R} \sum_{l=1}^{L} x_d^l w^l \tag{2.3}$$

Constraints: (2.2a), (2.2b), (2.2d)

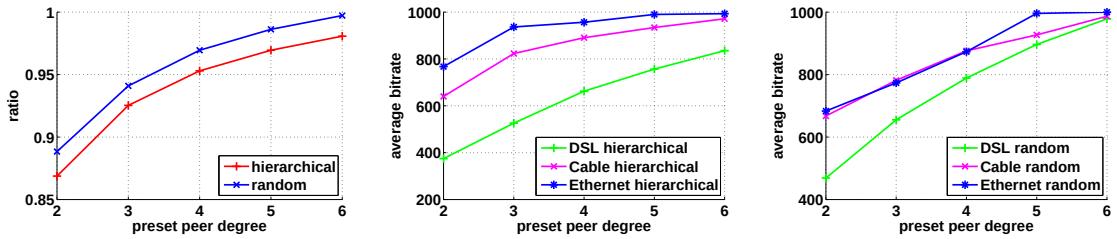
The solution of the linear programming problem **P2** gives *weighted priority* for peers to receive lower layer video. Another commonly employed fairness criterion is the *max-min fairness*. Similar to the max-min network flow allocation [36], the max-min fairness in layered streaming can be achieved using a "onion-peeling" solution. We refer interested readers to [37] for details.

2.2.3 Providing Incentives

In the previous efficiency and fairness study, we do not consider any incentive issues. This could cause serious problem in reality. For example, if an Ethernet user with uplink capacity of 2,000 Kbps and a DSL user with uplink capacity of 200 Kbps both receive video at rate of 500 Kbps, why would the Ethernet user contribute more than 200 Kbps? If we assume all peers are strategic, then the bandwidth contributed by peers will decrease and everyone will get poor video quality. On the other hand, if the DSL user uploads video at its full capacity, he may deserve some "help" from Ethernet users to download video at a rate higher than 200 Kbps.

Taxation

It is well-known that, in social welfare theory, taxation can help to improve the total utility of the whole society while maintaining a certain level of fairness. An optimal tax rate



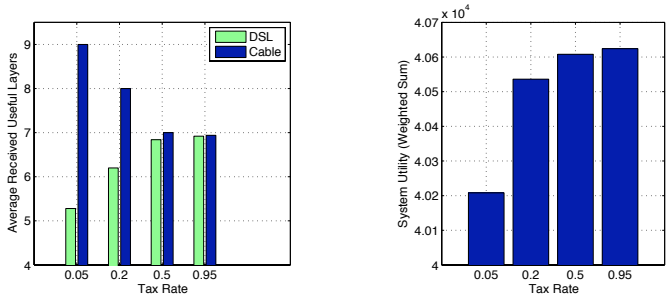
(a) (b) (c)

Figure 2.2: Utility maximization and fairness in hierarchical and random topologies. (a) System-wide utility under hierarchical and random topologies with different peer connectivity; (b) Averaged received bitrate for heterogeneous peers under hierarchical topology with different peer connectivity; (c) Averaged received bitrate for heterogeneous peers under random topology with different peer connectivity.

is usually non-linear and is complicated to determine given that taxes distort user behaviors. Here, we adopt the simple linear taxation method in [18, 19], that is

$$r_d = (1-t)U_d + \frac{t}{N}\sum_i U_i, \tag{2.4}$$

where t is the tax rate, N is the total number of peers. Unlike the definition in [19], we define $(1-t)U_d$ as peer d's entitled rate, and then map this rate to layers. All layers other than the entitled layers are denoted as excess layers. The trade-off between efficiency, fairness and incentive can be balanced by adjusting the tax rate t. Higher tax rate introduces higher system utility; smaller tax rate moves closer to tit-for-tat type of fairness. When the tax rate equals 0, the taxation degrades to the "tit-for-tat" or "bit-for-bit" strategy. In such a system, the system utility is obviously the lowest. Some poor peers can only receive a small portion of the video and thus obtain a rather degraded quality even though they contribute all of their uplink bandwidth. On the opposite side, when tax rate is 1, all peers retrieve the same video rate regardless of their contributions. Clearly, both scenarios are not desirable.



(a) (b)

Figure 2.3: Impact of taxation on fairness and system utility. (a) Averaged received layers for heterogeneous peers under different tax rate; (b) Achieved system-wide utility under different tax rate.

P2P Layered Streaming with Taxation

Two kinds of P2P layered streaming designs can be considered under taxation: *Equal share* and *Biased share*.

Equal share: In this case, the taxation pool is equally shared by all tax payers, i.e., participating peers in the system, which is exactly following (2.4). To perfectly implement the taxation scheme, one has to fully utilize the upload bandwidth available on all peers in the system. A sophisticated scheduling design is needed to meet this requirement by avoiding wasting bandwidth as much as possible.

Biased share: We only require a peer to receive all its entitled layers, i.e., $r_d \geq (1-t)U_d$. The bandwidth in the common taxation pool is distributed to maximize the system-wide utility. Towards this goal, we augment the utility maximization models studied in the previous sections by imposing an additional constraint on peer's receiving rate. At a given tax rate $0 < t \leq 1$, the utility maximization problem $\mathbf{P2}$ can be reformulated as follows.

Constraint (2.6d) guarantees that every peer should at least receive video at a rate proportional to its uploading contribution.

P3: Utility Maximization under Taxation

Variables:

$g_{ij}^{l,d}$ continuous non-negative variable
f_{ij}^{l} continuous non-negative variable
x_d^l $\in [0,1]$, continuous variable

Objective:

$$\max \sum_{d \in R} \sum_{l=1}^{L} x_d^l w^l \tag{2.5}$$

Constraints:

$$\sum_{\langle i,j \rangle \in E} g_{ij}^{l,d} - \sum_{\langle j,i \rangle \in E} g_{ji}^{l,d} = \begin{cases} x_d^l r^l, & i = S \\ -x_d^l r^l, & i = d \\ 0, & otherwise \end{cases}$$

$$\forall d \in R, \forall l \leq L \tag{2.6a}$$

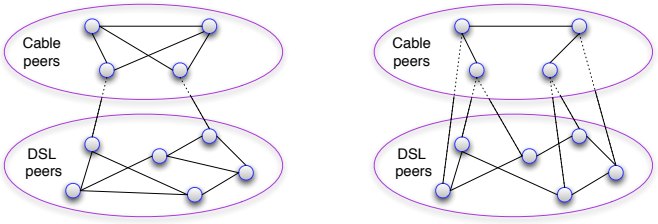
$$g_{ij}^{l,d} \leq f_{ij}^{l}, \quad \forall l \leq L, \forall d \in R, \forall \langle i,j \rangle \in E \tag{2.6b}$$

$$\sum_{l} \sum_{\langle i,j \rangle \in E} f_{ij}^{l} \leq U_i, \quad \forall i \in V \tag{2.6c}$$

$$\sum_{l} x_d^l r^l \geq (1-t) \sum_{l} \sum_{\langle d,j \rangle \in E} f_{dj}^{l} \quad \forall d \in R \tag{2.6d}$$

2.2.4 Numerical Studies

To gain insights on the interplay between efficiency, fairness and incentive, we conducted numerical studies on example systems with different topologies. For each system, we solve **P2** and **P3** using AMPL [38]. The obtained numerical results allow us to study the impact of streaming topology on the system performance. We first solve **P2** on a streaming overlay topology with 40 peers. There is one server and three types of peers with different upload bandwidth as summarized in Table 2.2. The server only streams video to at most eight Ethernet or Cable peers. We vary the peering connection degree of peers to investigate the impact of peer connectivity. The maximum peer degree is 10. Peers can operate



(a) Hierarchical topology (b) Random topology

Figure 2.4: Examples of hierarchical and random topologies.

at two different modes to select neighbors. One is the hierarchical mode (cf. Fig. 2.4(a)) in which a peer prefers to connect to peers with the same type (with 70% probability). The other is the random mode (cf. Fig. 2.4(b)) in which peers randomly connect to other peers, regardless of their types. The video stream is coded in 10 layers, each layer is encoded at rate of 100 Kbps.

Fig. 2.2 shows the results of solving **P2** under different preset peering degree, averaged over 10 runs. Fig. 2.2(a) compares the aggregated utility in hierarchical and random cases. The Y-axe value is obtained by dividing the aggregate utility by the maximum possible utility (here it is the case when every receiver gets $\frac{U}{N}$). The system utility increases as peering degree increases. The random mode gives better system wide utility. Fig. 2.2(b) and 2.2(c) compare the average receiving rate for each type of peers under the two peering modes. The hierarchical mode achieves higher service differentiation. In the protocol design described in Section 2.3.3, we introduce mesh topology adaptation to ensure the topology suits the taxation strategy.

To study the impact of taxation, we study a numerical example as shown in Fig. 2.3. The system has 15 cable peers with uploading capacity of 1000 Kbps and 25 DSL peers with 400 Kbps. They are hierarchically connected with degree 6. The video source is coded in 10 layers, each with 100 Kbps. The layer weight w^l is set as $2^{(10-l)}$. We vary the tax rate from 0.05 to 0.95. As can be seen from Fig. 2.3(a), when the tax rate is small, Cable peers with higher upload capacity obtain more layers. The service differentiation provides good

Table 2.2: Nodes' Settings

Type	Uplink Capacity	Number
Server	8000 Kbps	1
Ethernet peer	4000 Kbps	3
Cable peer	1000 Kbps	12
DSL peer	400 Kbps	25

incentives for them to participate in P2P sharing. As the tax rate increases, the differences between Cable peers and DSL peers decrease. On the other hand, system-wide utility increases with tax rate.

From the numerical results, we obtained the following guidelines for taxation-based P2P layered streaming design. 1) When the tax rate is small and thus the system is geared towards high service differentiation, hierarchical topology is preferred. 2) When the tax rate increases and the resulting system operates on a highly efficient point (high total utility), a more random topology is preferred. 3) Receiving entitled layers should be guaranteed for all peers and the uplink bandwidth should not be devoted to excess layers unless there is no more request for entitled layers.

2.3 Layered P2P Streaming Protocol Design

While the analytical models allow us to understand the the trade-offs in taxation-based layered video streaming, our ultimate goal is to design distributed mesh-based streaming protocols to dynamically balance the needs of fairness, incentive and system efficiency. In our design, peers form a mesh over which the video is distributed. A tracker serves as the bootstrapping node for the system. The key design issues for such a layered P2P streaming protocol are *layer subscription*, *chunk scheduling*, and *mesh topology adaptation*.

Multiple virtual streaming overlays, one for each SVC video layer, are formed among participating peers. Due to the dependency among video layers, upper streaming overlays must have fewer peers than lower overlays. A peer uses layer subscription scheme to deter-

mine how many layers to subscribe to. The chunk scheduling algorithms on peers allocate bandwidth among different overlays to balance the streaming needs of different layers. Finally, the mesh topologies need to be dynamically adjusted to adapt to the changing layer subscription due to peer churn and/or other network dynamics.

In the theoretic framework developed in Section 2.2, network coding is adopted to achieve the optimum multicast efficiency in general overlay topology. The gain of adopting network coding in real P2P systems is still an open question [34]. In layered P2P streaming systems, applying network coding to individual layers incurs extra coding/decoding overhead, increases video playback delays, and makes the protocol design more complex. Recent study [39] showed that when peers are fully connected and peer uplinks are the only bottlenecks, network coding is not needed. Even though we don't assume peers are fully connected, our distributed design does not employ network coding. We will show through simulations that the performance of the proposed mesh-based P2P streaming design is very close to the performance bound allowed by network coding.

2.3.1 Dynamic Layer Subscription

Under linear tax rate t, the target video download rate of peer d is $r_d = (1-t)U_d + \frac{t}{N}\sum_i U_i$, where $(1-t)U_d$ is peer d's entitled rate and $\frac{t}{N}\sum_i U_i$ is peer d's excess rate. Entitled and excess rates are then mapped to the number of entitled and excess layers. Tax rate t is a global configuration parameter and is known to all peers. Therefore peers can compute the number of entitled layers locally. However, the calculation of the number of excess layers needs global information - all active peers' uplink bandwidth. The number of excess layers on a peer also varies as other peers join and leave the system. We develop a distributed algorithm that probes the numbers of excess layers on peers and dynamically adjusts peers' layer subscriptions. The algorithm allows the system to approach the utility maximization under taxation, and adapt to peer churn and network dynamics nicely.

Let L_i denote peer i's entitled layers, and l_i denote the highest layer it is subscribed to. Motivated by the distributed utility maximization achieved by TCP in congestion control [40], we propose a distributed layer subscription algorithm with *Additive Increase Ad-*

Algorithm 1 The subscription increase process

```
if !inATrialProcess then
  t_i ← t_i − 1
  if t_i == 0 then
    l_i ← l_i + 1
    inATrialProcess ← TRUE
    trialProcessTimer ← 0
  end if
else
  trialProcessTimer ← trialProcessTimer + 1
  if trialProcessTimer == T′ then
    check success or not
    if !success then
      k ← k + 1
      t_i ← rand(2^{k−1}T, 2^k T)
    else
      k ← 0; inATrialProcess ← FALSE;
      t_i ← rand(1, T);
    end if
  end if
end if
```

ditive Decrease (AIAD) and exponential backoff. Upon joining the streaming session, peer i sets its initial layer subscription, l_i, to be L_i, the number of its entitled layers. It also starts a *retry timer*, $t_i = rand(1, T)$, where T is the retry time period. Upon the expiration of the retry timer, if all currently subscribed layers can be received and at least one neighbor peer possesses chunks of layer $l_i + 1$, peer i increases its subscribed layer by one, $l_i = l_i + 1$, and enters a *trial period* of T'. Peer i sends out requests for chunks in the newly added layer. If peer i is able to successfully obtain most of requested chunks of the new layer at the end of the trial period, it passes the test and the new layer subscription is accepted. Otherwise, peer i reverts back to the original subscription, and enters an exponential back-off stage. The retry timers is set to be $t_i = rand(1, 2^k T)$, where k is the number of consecutive failures. Algorithm 1 presents the pseudo-code of this process. Meanwhile, peer i runs a parallel *subscription decrease process* to ensure that it can receive all subscribed layers. Subscription decrease process periodically monitors the status of received layers.

If the top subscribed layer, l_i, becomes undecodable, and peer is not in the aforemen-

tioned trial period, peer i reduces the number of subscribed layers to $l_i = \max(l_i - 1, L_i)$. If a lower layer becomes undecodable, all higher subscribed layers are undecodable too. The peer will drop those layers gradually one by one.

2.3.2 Chunk Scheduling

Each peer maintains a downloading window that moves forward periodically. Peers periodically exchange chunk availability with their neighbors using buffer-maps. Neighbors help each other retrieve missing chunks. Chunk scheduling decides how to issue chunk requests to neighbor peers, and how to serve the chunk requests from neighbor peers. The goal is to properly utilize peers' uplink bandwidths so that peers always receive the entitled layers and receive the subscribed excess layers with high probability. In the following, we present the peer chunk requesting and chunk serving algorithms.

Chunk requesting

In a SVC coded video, lower layer bit-streams are more important than higher layer bit-streams. Hence in principle, lower layer chunks should be requested before higher layer chunks. In order to increase the data chunk diversity and improve the chance that two peers always have chunks to exchange, we further assume that data chunks belonging to the entitled layers are equally important. This is reasonable because the aggregated upload bandwidth in the system is sufficient to deliver the entitled layers to all peers. There is no need to distinguish different entitled layers.

The chunks are requested in the order of their importance: from entitled layer chunks to excess layer chunks. A peer selects one neighbor peer that owns a missing chunk to request for the chunk. The probability of choosing a specific peer is proportional to its serving rate to that peer. For example, if requester R serves neighbors A, B and C with 20Kbps, 50Kbps and 30Kbps respectively, it then sends the chunk request to A, B and C with probability 0.2, 0.5, and 0.3 respectively. Algorithm 2 presents the pseudo-code of requesting a missing chunk. C_{tot} and $C_{thisNeighbor}$ represent the total contribution and

Algorithm 2 Chunk requesting

```
for all neighbors do
  if this neighbor possesses the chunk then
    put this neighbor into candidate poll
    C_tot ← C_tot + C_thisNeighbor
  end if
end for
for i ∈ neighbors in the poll do
  prop_i ← C_thisNeighbor / C_tot
end for
choose one neighbor according to prop_i
```

contribution to the selected neighbor, respectively.

Chunk serving

Chunk serving is more sophisticated. Individual peers maintain two FIFO queues for each neighbor (see Fig. 2.5). One queue is called entitled queue and the other is called excess queue. Entitled queue holds chunk requests for entitled layers, while excess queue holds chunk requests for excess layers. The chunk requests in excess queues are sorted in ascending order of video layers, with the lowest layer chunk requests at the head. The entitled queues have strict priority over the excess queues. Excess queues would not be served unless all entitled queues become empty. If entitled queues become empty, the leftover bandwidth serves the requests in excess queues in a round robin fashion. Fig. 2.5 shows the idea of this serving process. The requests for video chunks that have passed their playback deadlines are cleared out of the queues and won't be served.

2.3.3 Mesh Topology Adaptation

As discussed in Section 2.2, hierarchical mesh topology is more favorable than random topology in providing differentiated services while random mesh is better for maximizing the system-wide utility. In this section, we consider how to efficiently adapt the mesh topology to achieve different design goals. Mesh topology adaptation is achieved through neighbor adaptation. A peer periodically contacts the tracker to retrieve a list of candidate

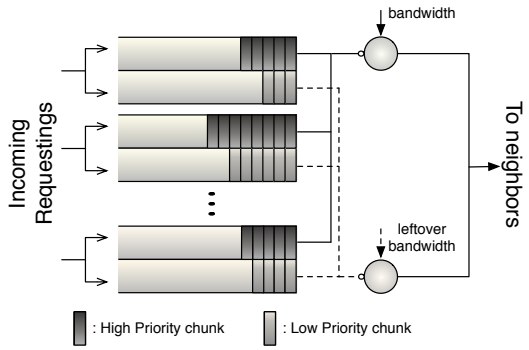
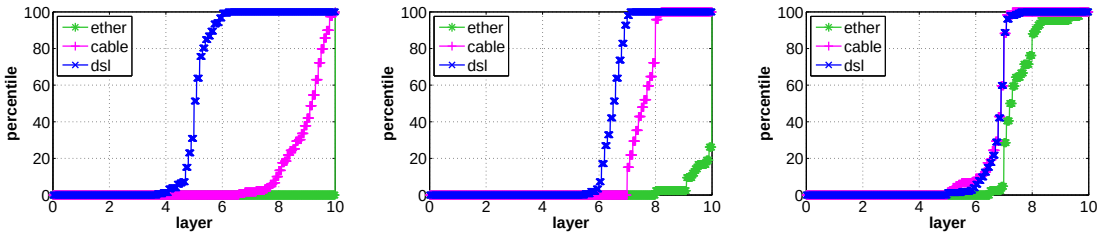


Figure 2.5: Peer serves neighbors. Low priority chunks will be served only if high priority queues are empty.

neighbors. It then applies the adaptation strategy described below to ensure the overlay topology converge to the desired topology.

Every peer has a preset peer out-degree. If the number of neighbors falls below the preset out-degree, a peer increases the number of neighbors by adding neighbors randomly selected from the candidate list. If the current number of neighbors is no less than the preset out-degree, a peer will terminate connections with some peers until its degree is one less than the preset out-degree and then replace them with a new peer from the candidate list. Specifically, a peer uses a *replacement index* (RI) to determine which peer to be replaced. Suppose peer i needs to adapt its neighbors. Let c_j^l be the number of retrieved chunks of layer l from peer j, and w_l be the weight associate with layer l. The replacement index for peer j is defined to be $\sum_{l \in i's\ entitled\ layers} c_j^l w_l$. In addition, the layer weights w are set such that $w_l > w_k$ if $l > k$ for the reasons explained below. The neighbor with the smallest replacement index is selected and swapped out. The length of the adaptation period is chosen as ten seconds in our design. The philosophy behind this design is two-fold.

- *Layer Level:* a neighbor offering high layers up to the entitled layers should stay. There are fewer peers in the higher virtual overlay. Peers who can offer high layer chunks are more precious and are more likely of the same class (with the same enti-



(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.6: Cumulative distribution of received video layers for different types of peers under various tax rates.

Algorithm 3 Neighbor adaptation

```
if number of neighbors >= Preset number then
    compute neigbors' replacement index
    while number of neighbors >= Preset number do
        drop the neighbor with smallest RI
        number of neighbors ← number of neighbors − 1
    end while
end if
while number of neighbors < Preset number do
    randomly pick one new peer and connect
    number of neighbors ← number of neighbors + 1
end while
```

tled layers).

- *Chunk Level:* among all neighbors offering chunks at the same layer, those uploading more chunks should stay.

Algorithm 3 presents the pseudo-code of neighbor adaptation. Simulation results in Section 2.4 indicate that the mesh topology will converge to the desired structures. At a low tax rate, which emphasizing differentiated services, hierarchical topology is achieved. At a large tax rate, which optimizing the system-wide utility, more random topology is realized.

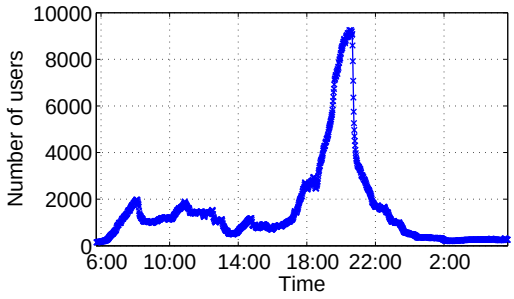


Figure 2.7: PPlive Online Users

2.4 Performance Evaluation

We conduct extensive trace-driven simulations to evaluate the performance of the proposed taxation-based P2P layered streaming design. Specifically, we investigate the following issues: (1) the effectiveness of taxation-based incentive mechanism; (2) peer uplink bandwidth utilization; (3) the mesh topology adaptation; (4) user/peer perceived video quality; and (5) the convergence and optimality of AIAD layer subscription.

A flow-level event-driven simulator is developed in C++. Unless stated otherwise, the simulations are driven by a trace collected from the measurement study of PPlive [41], a real-world P2P live streaming system. The trace was collected from Nov 22nd 17:43, 2006 to Nov 23rd 17:43, 2006. All peer arrivals and departures are recorded during this time. More than 100,000 participants are observed and the number of concurrent peers varies from 100 to more than 9,000. Fig. 4.5 shows the evolution of the number of concurrent peers.

The video is encoded into ten layers with layer rate of 100 Kbps. A ten layers SVC coded video can be created by combining hierarchical B structure [42] coding and coarse-grain scalability (CGS) [8] coding with acceptable overhead. More layers give the framework larger operational region and more room to adjust the balance between social welfare and individual welfare. More layers also impose more challenges on the underlying P2P system to properly handle the layer subscription, chunk scheduling, and overlay adaption.

There are three types of peers: DSL peers (400 Kbps), Cable peers (800 Kbps) and Ethernet peers (1500 Kbps). The fraction of individual peer types and their respective uplink bandwidths are summarized in Table 2.3. In our simulation, there is one video server with upload capacity of 10 Mbps.

The peer download window is set to 30 seconds. Peers exchange buffer-maps every second to calculate the missing chunk downloading schedule. Mesh topology adaptation is conducted every ten seconds. The values of T and T' in AIAD layer subscription algorithm are set to be 5 seconds and 10 seconds, respectively.

2.4.1 Overall System Performance

Effectiveness of Taxation Based Incentive Mechanism

To reduce the randomness introduced by short-lived peers, only peers with life time greater than one minute are included in this experiment. In taxation based P2P streaming, a peer's received video quality, or the number of layers, reflects its bandwidth contribution and the system-wide tax rate. In addition, the peers with similar bandwidth contributions receive similar video quality. Both are true as shown in Fig. 2.6, which depicts the Cumulative Distribution Functions (CDFs) of the numbers of received layers for different types of peers at different tax rates. The peers from the same class consistently receive a similar number of layers, while the numbers of video layers received by different peer classes are close to the optimum values–(5,9,10) under tax rate 0; (6,8,10) under tax rate 0.5 and (7,7,8) under tax rate 0.95.

Bandwidth Utilization Efficiency

Peers' uplink bandwidth utilization is a key performance metric for any P2P streaming system design. If the system is not well designed, the so-called "content bottleneck" lowers down the uplink bandwidth utilization, and degrades the average peers' received video quality. Content bottleneck refers to the situation that a peer receives fewer chunk requests than it can serve and consequently the peer's bandwidth is not fully utilized. We use uplink

Table 2.3: Peer upload bandwidth distribution

Peer Type	Uplink Bandwidth	Percentage
DSL	400 Kbps	45%
Cable	800 Kbps	40%
Ethernet	1500 Kbps	15%

Table 2.4: System Bandwidth Utilization

Tax rate	UBU	WBR
0	99.3%	0.4%
0.5	98.0%	0.1%
0.95	93.1%	0.7%

bandwidth utilization (UBU) and wasted bandwidth ratio (WBR) as metrics for evaluating the overall system utilization. A high BU means there is almost no content bottleneck. Meanwhile, a chunk may become undecodable at a receiver if its arrival time is later than its playback deadline, or its corresponding chunks in lower layers are not received. We call such chunks as wasted chunks. To quantify the effectiveness of bandwidth utilization, we define wasted bandwidth ratio (WBR) as the ratio of the bandwidth used to transmit wasted chunks to the total system bandwidth. The lower the WBR the higher, the bandwidth effectiveness.

Table 2.4 lists the UBU and the WBR for various tax rate settings. Overall, the uplink bandwidth utilization is consistently over 90%, indicating that the protocol can efficiently alleviate content bottleneck even without network coding. Interestingly, UBU is worse at larger tax rates. As tax rate increases, the peers become more altruistic, which requires more bandwidth sharing among different types of peers. Due to the peer churn and mesh topology constraint, the bandwidth sharing may not be always possible, thus lower the utilization. On the other hand, the WBR is pretty small for all tax rate settings, pointing to an efficient protocol design.

Table 2.5: Topology Statistics For tax rate 0

Neighbor Type	Ethernet	Cable	DSL
Ethernet	28.4%	19.0%	9.0%
Cable	47.3%	58.1%	21.1%
DSL	24.3%	22.9%	69.9%

Mesh Overlay Adaptation

Mesh overlay topology plays a key role in service differentiation. Table 2.5 and 2.6 list the peer neighborhood statistics with the tax rate of 0 and 0.95, respectively. With tax rate 0, the optimal number of video layers for Ethernet users, Cable users, and DSL users are 10, 9, and 5, respectively. Around 76% of Ethernet peers' neighbors are either Ethernet or Cable users, and around 77% of Cable peers' neighbors are either Ethernet or Cable users. In contrast, DSL users mainly connect with other DSL users (70%). The strong bias towards connecting with similar type of peers leads to a hierarchical mesh topology, which allows Ethernet and Cable users to exchange higher video layers (from layer 6 to layer 10) that are not available at DSL users.

With tax rate 0.95, all peers are supposed to receive a similar number of video layers regardless of their individual bandwidth contributions. Numerical results in Section 2.2.4 suggest that random mesh topology is better at achieving high social welfare. Our mesh topology adaption scheme is able to reflect this requirement. For DSL users, the fraction of DSL neighbors is reduced from 70% (with tax rate 0) to 44%. For Ethernet users, the fraction of DSL neighbors is increased from 24% to 40%. Compared with the mesh topology constructed at tax rate 0, this is a more randomized topology for the peer distribution in Table 2.3.

2.4.2 Video Quality Evaluation

To demonstrate the video quality of our system, we generate a bitstream with JSVM 9.12 [43] and use the real video trace to drive the simulations. We encode the sequence

Table 2.6: Topology Statistics For tax rate 0.95

Neighbor Type	Ethernet	Cable	DSL
Ethernet	21.1%	14.9%	14.5%
Cable	38.5%	39.9%	41.5%
DSL	40.4%	45.2%	44.0%

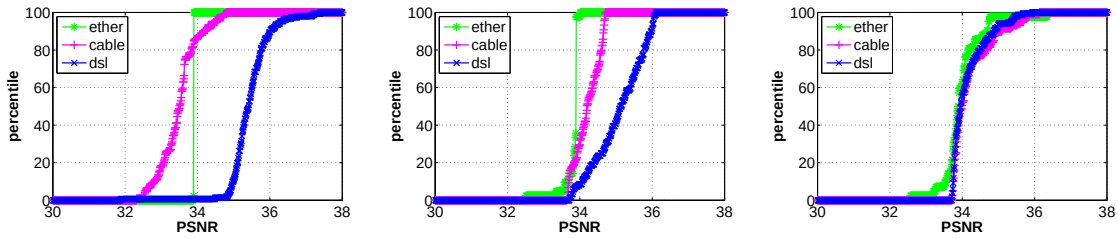
Table 2.7: Bitstream statistics (sequence FOOTBALL)

Layer	Bitrate (Kbps)	PSNR (dB)	Framerate (fps)
1	303.8	36.97	3.75
2	420.1	38.32	3.75
3	503.0	34.89	7.5
4	595.9	36.36	7.5
5	705.3	33.65	15
6	800.4	34.73	15
7	905.8	32.32	30
8	992.4	33.91	30

FOOTBALL into a bitstream with 4 temporal layers and 2 SNR layers. We use the FixedQPEncoderStatic tool to do the rate control with rate mismatch setting [-3%, 1%]. The bitstream statistics are summarized in Table 2.7. We loop the bitstream repeatedly until the simulation ends.

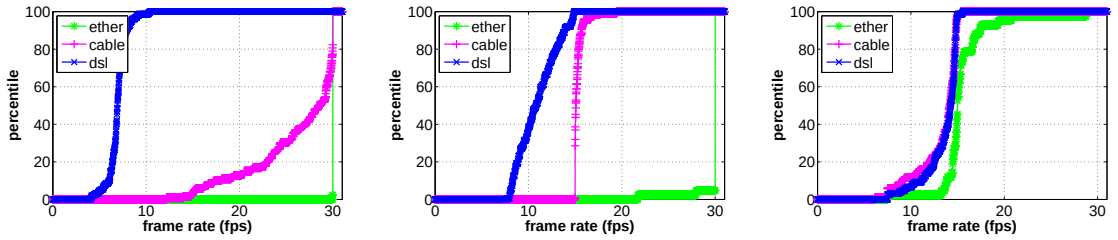
Received Video Quality

Fig. 2.8 and Fig. 2.9 show the cumulative distributions of peers' average PSNR and framerate. Note that PSNR of higher layers might be less than lower layers. But the video quality not only depends on PSNR but also depends on the framerate. Therefore, we present the subjective video quality obtained by using a subjective quality model [44]. Fig. 2.10 shows the cumulative distributions of peers' subjective quality. We can see from Fig. 2.10 that the predicted subjective quality experienced by peers matches the differential service criterion. In addition, at a smaller tax rate, the quality gap between weak and strong peers



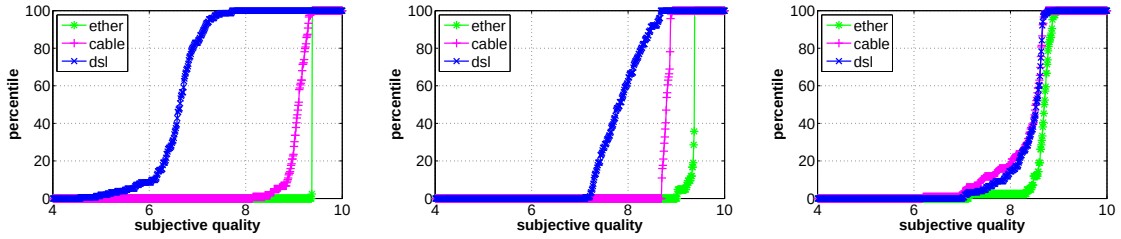
(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.8: Cumulative distribution of received PSNR for different types of peers under various tax rates.



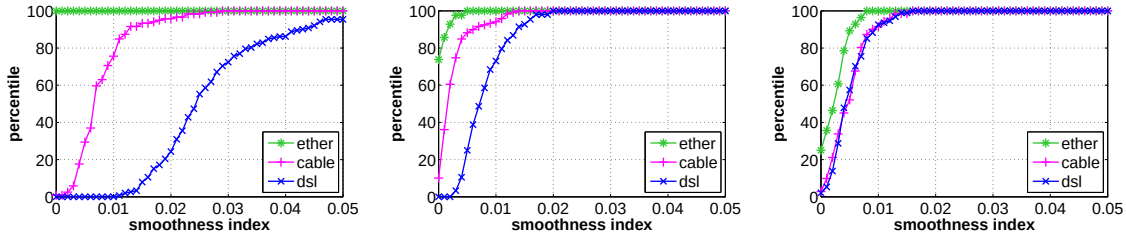
(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.9: Cumulative distribution of received framerate for different types of peers under various tax rates.



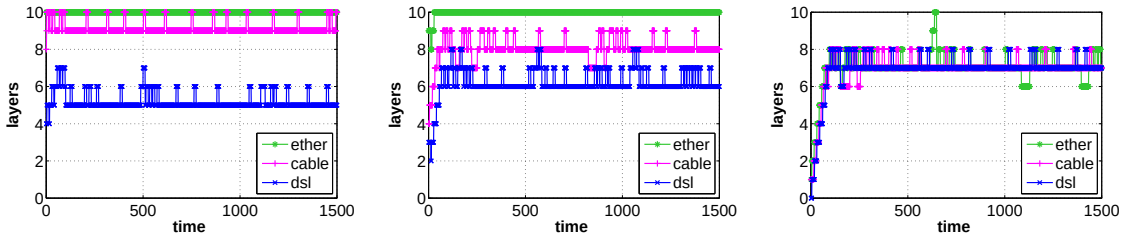
(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.10: Cumulative distribution of received subjective quality for different types of peers under various tax rates.



(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.11: Cumulative distribution of smoothness index for different types of peers under various tax rates.



(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.12: Layer subscription evolution for different types of peers under various tax rates.

is larger than that at a larger tax rate, which pushes the system to equal sharing of system resources. Therefore, by tuning the tax rate, our system can operate at any desired point, not only in terms of the rate allocation but also in terms of the video quality of experience (QoE).

Smoothness of Received Quality

In addition to the PSNR and framerate, viewing quality could be affected by the variations of the received video layers over time as well. QoE is degraded if the number of received video layers changes frequently. We define the following *smoothness index* to quantify the playback smoothness of the received video.

$$SI = \frac{1}{K}\sum_{k=0}^{K}\frac{|v(k)-v(k-1)|}{v(k)}, \tag{2.7}$$

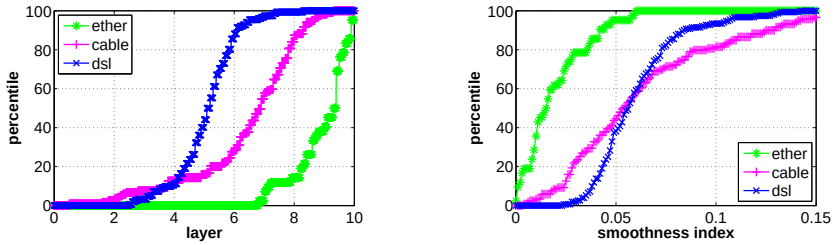
where $v(k)$ is the received decodable layers at time period k, and K is peer's total number of online time period. The time period is one second. Large smoothness index indicates bad viewing quality caused by frequent layer increasing/droping. Fig. 2.11 shows the CDFs of smoothness index under different tax rates. Under all scenarios, peers contributing more enjoy smoother video playback. We also observed that as tax rate increases, the smoothness indexes for bandwidth-rich peers increase, while the smoothness indexes for bandwidth-

poor peers decrease. We suspect this is caused by the peering topology at different tax rates.

As discussed in Section 2.4.1, a hierarchical topology is formed at tax rate 0. DSL peers are mainly connected with other DSL peers and have to actively look for bandwidth resources, which causes more layer changes. In contrast, Ethernet and Cable users are mainly connected with each other, and have abundant bandwidth within the cluster. Thus fewer layer changes. As tax rate increases, the overlay topology becomes more randomized. Different peers have equal/similar access to bandwidth, leads to similar smoothness index.

2.4.3 Layer Subscription Convergence

In order to examine the behavior of the layer subscription algorithm without the impact of peer churn, a static topology with 500 peers is used in this experiment. The peers' uplink bandwidth follows the distribution as stated in Table 2.3. We randomly pick one peer from each bandwidth category and plot the evolution of its layer subscription. We also vary the tax rate to examine its impact. Fig. 2.12 shows the layer subscription process over time with tax rate of 0, 0.5, and 0.95, respectively. With tax rate of zero, peers are entirely selfish. The Ethernet peers with bandwidth of 1500 Kbps receive all ten layers. The leftover bandwidth subsidizes other peers. As a result, the optimal layer subscription for Cable and DSL peers are 9 layers and 5 layers, respectively. With tax rate of 0.5, the optimal layer subscription for DSL, Cable, and Ethernet peers are 6 layers, 8 layers, and 10 layers, respectively. Finally, with tax rate of 0.95, peers are altruistic and every peer should receive 700 Kbps except for the Ethernet peers (800 Kbps). Since video is encoded at 100 Kbps per layer, there are more "free" bandwidth in this case, introducing minor oscillations in layer subscription. In all cases, AIAD algorithm is able to quickly converge to the target subscription layer and peers stay in their optimal layers for most of the time.



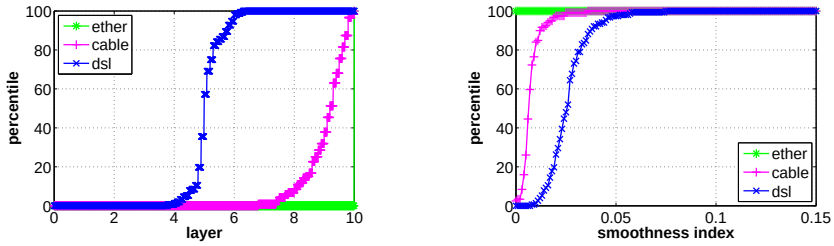
(a) CDF of receiving layers (b) CDF of smoothness index

Figure 2.13: System performance of LayerP2P protocol.

2.4.4 Performance Comparison with Existing Protocol

In this section, we present the performance comparison between LayerP2P [45] and our proposed design. The design objective for LayerP2P is to provide incentive for peers to upload. It allows peers to trade layers with each other. This corresponds to a special case of our design with zero tax rate. We compare LayerP2P with our system at tax rate 0 Fig. 2.13 and Fig. 2.14 show the cumulative distributions of peers receiving layers and smoothness index of the two systems. As can be seen in Fig. 2.13(a), LayerP2P is able to provide differential services to different peers, but there exists larger variations of receiving layers for all peers compared with Fig. 2.14(a). LayerP2P also performs worse in terms of the video playback smoothness. Comparing Fig. 2.13(b) with Fig. 2.14(b), one can infer that peers would experience more jitters in LayerP2P system.

Table 2.8 summarizes the UBU, WBR and overhead of the two systems. LayerP2P is able to achieve a high UBU as the proposed design. However, the WBR is significantly larger for LayerP2P which results in more jitters. In addition, LayerP2P has a much higher overhead (5X) than the proposed design. In other words, the proposed design has a much better chunk scheduling efficiency.



(a) CDF of receiving layers (b) CDF of smoothness index

Figure 2.14: System performance of the proposed protocol.

Table 2.8: Comparison of UBU, WBR and overhead between LayerP2P and the proposed design

Protocol	UBU	WBR	Overhead
LayerP2P	98.1%	5.9%	211.3826
Proposed	99.3%	0.4%	41.0242

Table 2.9: Performance comparision: noDLS vs DLS

Scenario	$\bar{L}_D$	$\bar{L}_C$	$\bar{L}_E$	UBU	WBR	Overhead
noDLS	4.67	9.01	10	99.6%	3.3%	89.20
DLS	5.21	8.94	10	99.3%	0.4%	41.02

2.4.5 Impact of Different Modules

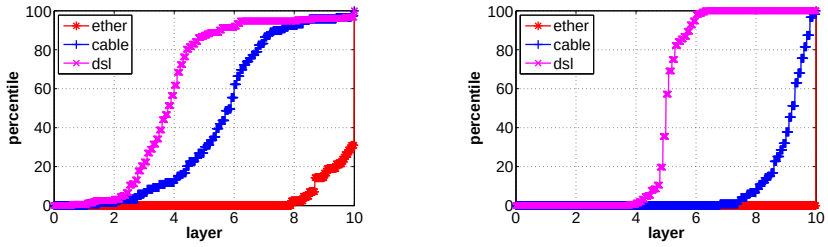
To evaluate the impact of different algorithm modules, we conducted comparison simulation studies by disabling/enabling different modules. We conducted experiments for tax rate of 0, 0.5 and 0.95 respectively. The trends are essentially the same. In this section, we only present the results for tax rate of 0.

Impact of Dynamic Layer Subscription

Table 2.9 summarizes the system performance of two scenarios: noDLS and DLS. In noDLS case, all peers subscribe to all layers while in DLS case, peers use dynamic layer subscription as described in Sec. 2.4.5. $\bar{L}_D$, $\bar{L}_C$ and $\bar{L}_E$ denote the average numbers of received layers on DSL peers, Cable peers and Ethernet peers respectively. The overhead is measured by the average number of chunk requests sent out by a peer in one slot. It can be seen that, without DLS, peers are sending out more chunk requests on average than with DLS. In addition, the WBR is much higher without DLS, and consequently, a much lower request success ratio. DLS is more important for "weak" peers because they are supposed to receive fewer layers. For example, a DSL peer is allowed to receive 5 layers, but it sends out chunk requests of all layers without DLS. This does not only increase the overhead but also reduce the chance of getting lower layer requests served.

Impact of Chunk Scheduling

We substitute the proposed chunk scheduling algorithm with a purely random chunk scheduling algorithm. With random scheduling, for a missing chunk, a peer randomly pick a neighbor possessing the chunk to download the chunk. Fig. 2.15 shows peers' receiving profile with the two scheduling algorithms. It can be seen in Fig. 2.15(a) that random scheduling is inadequate to provide good differential service. Even for peers of the same type, the receiving rates vary over a wide range.



(a) Random scheduling (b) Proposed scheduling

Figure 2.15: Cumulative distribution of received layers for different peers of two scheduling algorithms.

Impact of Mesh Adaptation

Intuitively, peers with higher bandwidth benefit more from mesh adaptation. They can increase their chunk request fulfillments by probing for "strong" peers. Fig. 2.16 shows the cumulative distributions of the received video layers for Cable and Ethernet peers under two settings: enabling mesh adaptation and disabling mesh adaptation. In the latter case, the connection between two peers will never terminate until one is offline. It can be seen that, with mesh adaptation, peers can receive more layers on average than without mesh adaptation. For example, with mesh adaptation, more than 70% peers can receive more than 9 layers, while the number is only 40% without mesh adaptation.

2.4.6 Impact of Severe Peer Churns

In this section, we evaluate the performance of our system when there is severe peer churn. We extract the "spike" of the PPlive trace between time 18:00 to 22:00 and use it to drive the simulator. The results are shown in Fig. 2.17 and Fig. 2.18.

From Fig. 2.17, we can see that peers' receiving layers during flash crowd period are still quite stable in all three tax rate settings. Peers of the same type receive similar numbers of layers and the system can effectively provide service differentiation among different types of peers. On the other hand, severe peer churn indeed degrades peers'

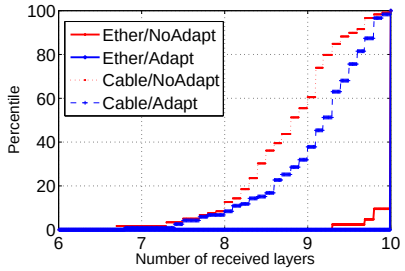
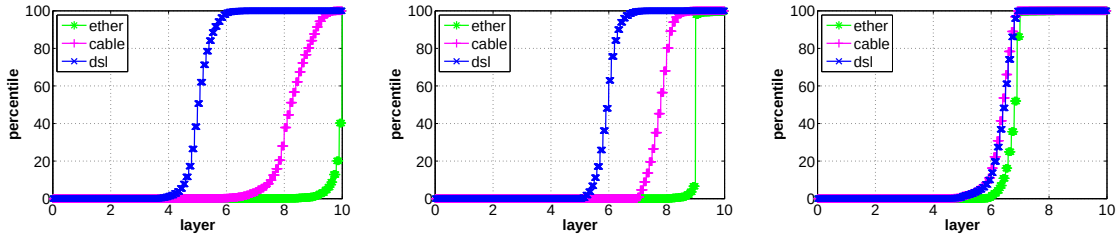


Figure 2.16: Cumulative distribution of received layers for Cable and Ethernet peers with mesh adaptation and without mesh adaptation.

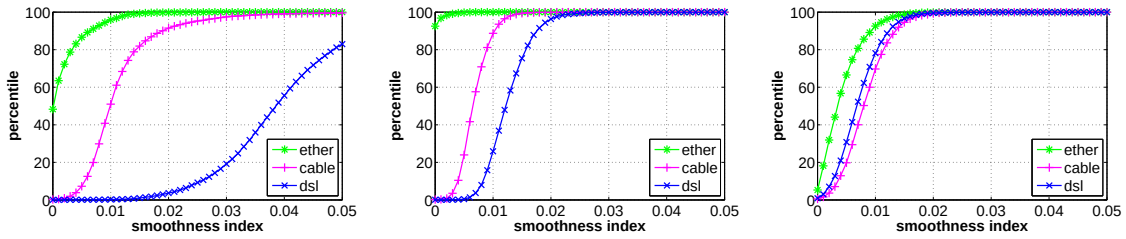


(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.17: Cumulative distribution of received layers for different types of peers during severe peer churn.

receiving quality. For example, at tax rate 0, all Ethernet peers received 10 layers during normal peer churn whereas only about 50% Ethernet peers received 10 layers. Fig. 2.18 shows the smoothness index of peers under severe peer churn. Compared with Fig. 2.11, it can be noticed that the system performs slightly worse during severe peer churn but the performance is still acceptable.

Table 2.10 summarizes the uplink bandwidth utilization (UBU) and wasted bandwidth ratio (WBR) during severe peer churn. At tax rate 0 and 0.5, severe peer churn does not impact too much as UBU and WBR are very close to those under normal peer churn. However, the UBU drops at tax rate 0.95, which means that the system efficiency is adversely



(a) Tax rate 0 (b) Tax rate 0.5 (c) Tax rate 0.95

Figure 2.18: Cumulative distribution of smoothness index for different types of peers during severe peer churn.

Table 2.10: System Bandwidth Utilization During Severe Peers Churn

Tax rate	UBU	WBR
0	99.2%	1.0%
0.5	98.1%	0.1%
0.95	87.5%	0.2%

impacted by severe peer churn at high tax rate.

From the above results we see that the system is running more efficiently during slow to moderate peer churn period than during severe peer churn period. Nevertheless, our system is still capable of handling severe peer churn and provides an acceptable performance. Flash crowd is a well-known challenge for P2P streaming. We will further address this challenging issue in our future work.

2.5 Summary

In summary, we develop utility maximization models to understand the interplay between the efficiency, fairness and incentive in layered P2P streaming. The models enable us to numerically investigate the impact of peering strategies and chunk scheduling policies on the fundamental trade-offs between the above three factors. We further integrate

taxation-based incentive mechanism into P2P layered streaming, and develop a practical streaming system. Taxation-based P2P streaming allows us to freely adjust the balance between the social welfare and individual peer welfare. Extensive trace-driven simulations demonstrate that the proposed designs can effectively drive layered P2P streaming systems to operating points with the desired balance between efficiency, fairness and incentive.

Chapter 3

SVC Rate-Quality Modeling and Layer Ordering

This chapter presents the rate-quality modeling for scalable video with focus on the joint temporal and amplitude scalability. In addition, a SVC layer ordering strategy is proposed such that each additional layer provides best quality gain given the bitrate increment. As will be shown later, the rate-quality model and layer ordering strategy can be easily used for video adaptation within network utility maximization framework.

3.1 Background

3.1.1 SVC Coding Scheme

Scalable video coding schemes have been advocated for video adaptation to network and terminal capabilities due to its low complexity and flexibility [11]. This approach eliminates computationally demanding transcoding processes at video servers or intermediate proxies by simply extracting appropriate bitstreams according to network or terminal constraint. It is shown that the latest SVC standard [10] can achieve comparable coding efficiency as the state-of-the-art H.264/AVC non-scalable coding [46].

In SVC, the motion-compensated transform coding architecture is extended to achieve a wide range of spatio-temporal and amplitude scalabilities. Fig. 3.1 illustrates the typical structure of a group of pictures (GOP) that implements only temporal and amplitude scalabilities. Each frame in the video sequence is encoded into multiple amplitude layers

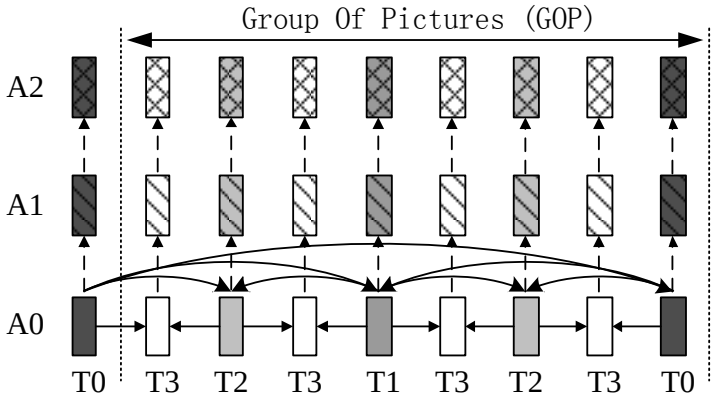


Figure 3.1: Structure of a group of pictures (GOP) in an H.264/SVC stream encoded with the coarse granularity scalablity (CGS) approach. The GOP length is 8 frames in this example. The stream supports 3 amplitude layers and 4 temporal layers.

(labeled as $A0$, $A1$ and $A2$) with decreasing QS q. Inter-frame prediction among the pictures in each amplitude layer follows a dyadic pattern, leading to several temporal layers (labeled as $T0$, $T1$, etc.) with increasing FR f. This approach, also known as coarse granularity scalability (CGS), allows the stream to be flexibly decoded at various combinations of amplitude and temporal levels without introducing any mismatch error in the decoding process. In the example of Fig. 3.1, for instance, the stream supports 3 amplitude layers and 4 temporal layers, thereby allowing 12 rate-quality tradeoff points. If at certain point, the server or proxy decides to send $(A2, T2)$, then all chunks with label $(A_i \leq A2, T_i \leq T2)$ will be extracted for sending, e.g. those shaded chunks in the GOP. In the bitstream, each chunk represents a network abstract layer (NAL) unit, containing bits at that layer from a single video frame.

With CGS, video adaptation is allowed at GOP boundaries. In the same example, suppose the highest framerate is 30 frames per second (fps), then the video rate can be switched every $8/30$ seconds. There is a tradeoff between granularity of the adaptation interval and coding efficiency since smaller GOP size leads to lower compression ratio. The typical choice of GOP size ranges between 8 to 32 for video streaming.

3.1.2 SVC Rate Model and Quality Model

There have been quite extensive research exploring the impact of frame rate and quantization step size, individually and jointly, on the perceptual quality [13, 44, 47–52]. However, there is not a widely adopted quality model that considers explicitly the effect of both FR and QS.

In [13, 44], the authors studied the impact of FR and QS on the subjective quality and bitrate of scalable video. Based on the mean opinion score (MOS) obtained from subjective quality tests, it is observed that the impact of FR and that of QS on the MOS is separable. Therefore, the subjective quality of SVC encoded video can be modeled as the product of two functions of the QS q and FR f, respectively. The overall subjective quality model is:

$$Q(q,f) \quad = \quad Q^{max}\frac{e^{-c\frac{q}{q^{min}}}}{e^{-c}}\frac{1-e^{-d\frac{f}{f^{max}}}}{1-e^{-d}} \quad , \tag{3.1}$$

where c, d are content-dependent model parameters; q^{min} denotes the minimum QS; f^{max} denotes the maximum FR. Here Q^{max} denotes the subjective quality achievable at (q^{min}, f^{max}).

For the same set of encoded sequences, their bitrates are recorded and the influence of q and f on the bitrate is analyzed. It is found that the bitrate can also be modeled as the product of two functions of q and f, respectively. The overall rate model is:

$$R(q,f) \quad = \quad R^{max}(\frac{q}{q^{min}})^{-a}(\frac{f}{f^{max}})^{b} \quad , \tag{3.2}$$

where a, b are content-dependent model parameters and R^{max} corresponds to the bitrate at (q^{min}, f^{max}). The parameter values for seven test sequences are given in Table 3.1. How to estimate the values of a, b, c, d, R_{max} based on the video characteristics can be found in [53]. We note that for most applications, SVC streams should have similar subjective quality scores at the highest rate, and thus the Q^{max} values. Therefore, it is the normalized quality $\tilde{Q} := Q(q,f)/Q^{max}$ that is of interest. In the sequel, we focus on how to model the relation between the normalized quality and the rate, and how to maximize the normalized quality.

3.2 Rate-Quality Modeling

3.2.1 Deriving the Rate-Quality Model

As decreasing FR or increasing QS will lead to decreasing bitrate and vice versa, there might be multiple combinations of FR and QS that satisfy a given bitrate constraint R. But the associated quality is different. It is then desirable to find the combination that gives the best quality while satisfying R. By finding the optimal (q, f) and the corresponding maximum achievable normalized quality $\tilde{Q}$ for each possible R, we arrive at the rate-quality model, to be denoted as $\tilde{Q}(R)$.

First, we define some notations that will be used later: normalized bitrate $\tilde{R} = R/R^{max}$, normalized FR $\tilde{f} = f/f^{max}$ and relative QS $\tilde{q} = q/q^{min}$. Given a rate constraint $R \leq R^{max}$, it can be derived from (3.2) that $\tilde{R} = \tilde{q}^{-a}\tilde{f}^{b}$, which is equivalent to $\tilde{q} = \tilde{R}^{-\frac{1}{a}}\tilde{f}^{\frac{b}{a}}$. Then, we can rewrite (3.1) in terms of $(\tilde{f}, \tilde{R})$ as

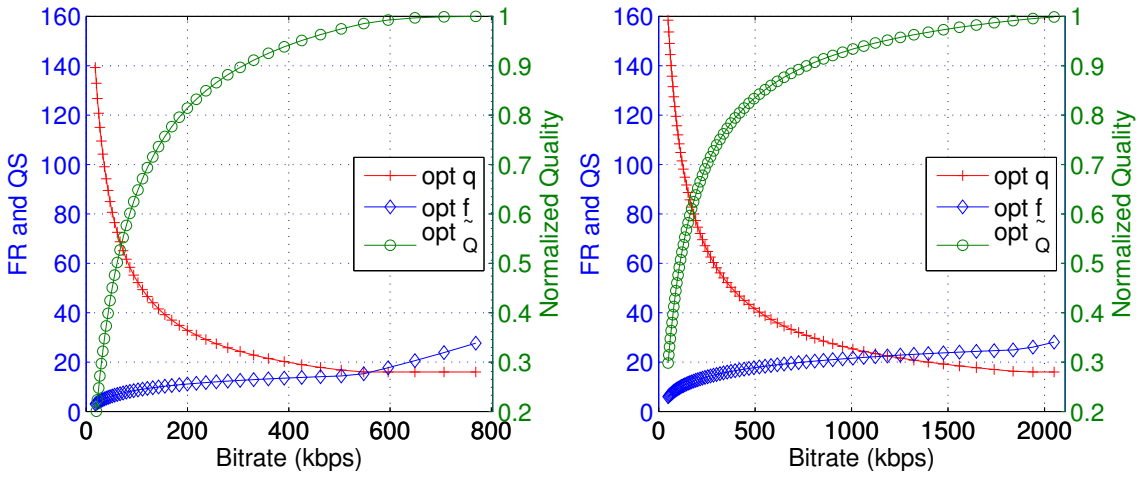
$$\tilde{Q}(\tilde{f}, \tilde{R}) = \frac{e^{-c\tilde{R}^{-\frac{1}{a}}\tilde{f}^{\frac{b}{a}}}(1 - e^{-d\tilde{f}})}{e^{-c}(1 - e^{-d})} \tag{3.3}$$

To find the optimal FR $\tilde{f}^*$ which maximizes $\tilde{Q}(\tilde{f}, \tilde{R})$ for a given $\tilde{R}$, we solve for $\partial\tilde{Q}(\tilde{f}, \tilde{R})/\partial\tilde{f} = 0$. Together with (3.2), we have the following relation between the optimal $(\tilde{q}^*, \tilde{f}^*)$ and the rate contraint $\tilde{R}$,

$$\tilde{R} = \left(\frac{bc}{ad}\right)^{a} \tilde{f^*}^{b} \left(\frac{e^{d\tilde{f}^*} - 1}{\tilde{f}^*}\right)^{a} \tag{3.4}$$

$$\tilde{R} = \tilde{q^*}^{-a}\tilde{f^*}^{b} \tag{3.5}$$

Note here, as $\tilde{q} >= 1$ and $\tilde{f} \in (0, 1]$, if the resulting $\tilde{q}^*$ is less than 1, the q_{min} constraint is active and then $\tilde{q}^*$ is clipped to 1; a new $\tilde{f}^*$ is calculated using (3.5). The above equations give a criteria for choosing the FR and QS under any rate constraint. It is easy to verify that $\partial\tilde{f}^*/\partial\tilde{R} > 0$ and $\partial\tilde{q}^*/\partial\tilde{R} < 0$ so that (q^*, f^*) is unique and f^* is monotonically increasing while q^* is monotonically decreasing as $\tilde{R}$ increases. Then, the best normalized quality $\tilde{Q}^*$ can be derived from (q^*, f^*) by using (3.1). Although it is hard to derive closed-form relations between the rate constraint $\tilde{R}$ and the optimal $q^*(\tilde{R})$, $f^*(\tilde{R})$, and $\tilde{Q}^*(\tilde{R})$,



(a) FOREMAN (b) FOOTBALL

Figure 3.2: Optimal FR f and QS q (left axis) and the corresponding optimal normalized quality Q (right axis) versus bitrate. These results assume that both FR and QS can take on any value in their respective ranges, i.e., $f \in (0, 30]$ and $q \in [16, 160]$.

we can numerically compute f^*, q^*, and $\tilde{Q}^*$ for any given $\tilde{R}$. For notational simplicity, in the sequel, we will ignore the superscript $*$ in $\tilde{Q}^*(\tilde{R})$, and use $\tilde{Q}(\tilde{R})$ to denote the optimal rate-quality tradeoff. Fig. 3.2 shows the (q^*, f^*) and the corresponding $\tilde{Q}^*$ versus bitrate for two sequences, FOREMAN and FOOTBALL.

Fig. 3.3 shows the numerically computed optimal rate-quality tradeoff curves in terms of normalized rate for seven sequences with various video characteristics. We found that these curves can be closely approximated with the following exponential function

$$\tilde{Q}(\tilde{R}) = e^{-\alpha \tilde{R}^{-\beta} + \alpha} \tag{3.6}$$

where α, β are model parameters. The model parameters for each sequence can be found by least squares fitting into the numerically calculated $(\tilde{R}, \tilde{Q})$ data for that sequence. Table 3.1 summarizes the model parameters for each sequence and the model accuracy in terms of the root-mean-square error (RMSE). The fitting curve for individual sequence matches with its data very accurately and hence are not shown separately in Fig. 3.3.

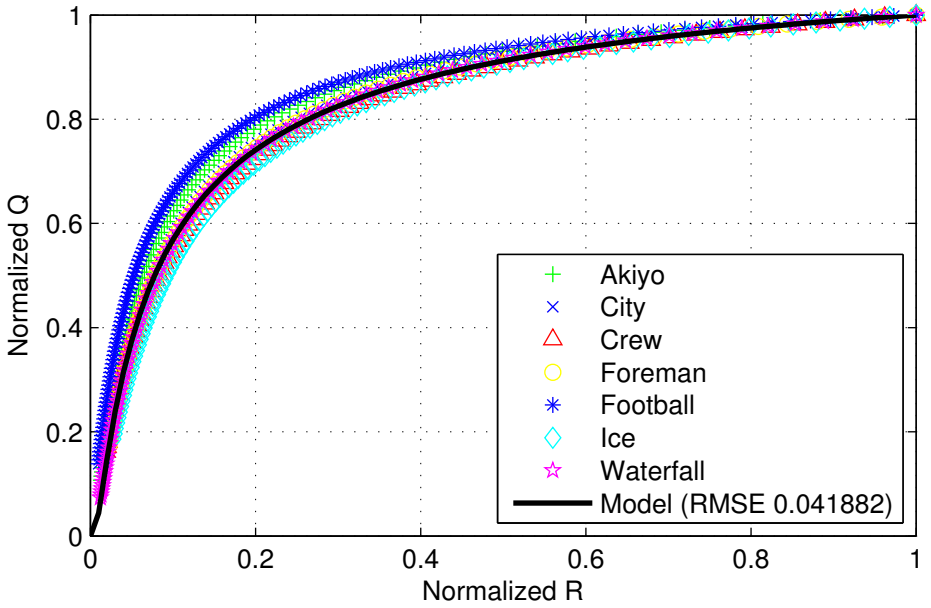


Figure 3.3: Normalized optimal Rate-Quality tradeoff curves for seven sequences: AKIYO, CITY, CREW, FOOTBALL, FOREMAN, ICE and WATERFALL. The solid line gives a unified rate-quality model.

Because the optimal $\tilde{Q}(\tilde{R})$ curves for different sequences in Fig. 3.3 are very close to each other, we further propose to use a unified model with the same parameter set for all sequences. Using least squares fitting to the $(\tilde{R}, \tilde{Q})$ data from all sequences, we found $\alpha = 0.16$ and $\beta = 0.66$. This unified model is also shown in Fig. 3.3 and summarized in Table 3.1. Note that although we propose to use the same model for the normalized rate-quality relations for different sequences, the maximum rate R_{max} is still video-sequence dependent. We can rewrite the $\tilde{Q}(\tilde{R})$ as a function of the absolute rate as

$$\tilde{Q}(R) = e^{-\alpha(\frac{R}{R_{max}})^{-\beta}+\alpha} \tag{3.7}$$

3.2.2 Model Discussions

The proposed model (3.7) can be easily incorporated to solve network utility maximization problem of the form $\sum_{i \in I} U_i(R_i)$ subjects to system constraints, where I denotes

Table 3.1: Parameters for Rate Model, Quality model, and Rate-Quality Model and the RMSE for Rate-Quality Model

Seq. Name	a	b	c	d	α	β	R_{max} (kbps)	RMSE
AKIYO	1.213	0.470	0.114	7.696	0.12	0.69	163.2	0.003
CITY	1.194	0.484	0.130	7.512	0.14	0.69	658.3	0.003
CREW	1.243	0.671	0.184	6.902	0.20	0.59	1381.4	0.003
FOREMAN	1.149	0.570	0.149	8.236	0.15	0.68	805.5	0.003
FOOTBALL	1.128	0.739	0.088	5.197	0.13	0.63	2154.1	0.005
ICE	1.039	0.670	0.150	6.674	0.17	0.67	693.0	0.004
WATERFALL	1.294	0.430	0.145	7.060	0.14	0.68	462.9	0.006
UNIFIED	N/A	N/A	N/A	N/A	0.16	0.66	N/A	0.042

receiver set. In the following, we consider two special cases with different objectives.

Case I

if the objective is to provide proportional fairness [35] while maintaining sufficient high system utilization, the utility function is defined as

$$U_i(R_i) = lg\tilde{Q}_i(R_i) = -\alpha(\frac{R_i}{R^i_{max}})^{-\beta} + \alpha \tag{3.8}$$

It can be easily verified that (3.8) is a concave function, therefore, the NUM problem can be solved efficiently. In addition, there also exists distributed algorithms to solve this problem [54].

Case II

if the objective is to maximize system utilization, then the utility function can be defined as

$$U_i(R_i) = \tilde{Q}_i(R_i) = e^{-\alpha(\frac{R_i}{R^i_{max}})^{-\beta}+\alpha} \tag{3.9}$$

(3.8) is not concave over the region $(0, R^i_{max}]$. However, we claim that in practical video rate regions, (3.8) is a concave.

Claim 1. *In practical video rate regions,* (3.8) *is a concave.*

To justify this claim, we show the concave region for $\tilde{Q}(R)$, or equivalently, $\tilde{Q}(\tilde{R})$.

$$\begin{aligned}\tilde{Q}''(\tilde{R}) &= (e^{-\alpha\tilde{R}^{-\beta}+\alpha})'' \\ &= \alpha\beta e^{(-\alpha\tilde{R}^{-\beta}+\alpha)}\tilde{R}^{-\beta-2}\left(\frac{\alpha\beta}{\tilde{R}^{\beta}}-\beta-1\right)\end{aligned}$$

As $\alpha\beta e^{(-\alpha\tilde{R}^{-\beta}+\alpha)}\tilde{R}^{-\beta-2} > 0$, then $\frac{\alpha\beta}{\tilde{R}^{\beta}} - \beta - 1 < 0$ gives the concave region of $\tilde{Q}(\tilde{R})$. And the concave region is $\tilde{R} > (\frac{\alpha\beta}{\beta+1})^{1/\beta}$. At the lower boundary $\tilde{R}_l = (\frac{\alpha\beta}{\beta+1})^{1/\beta}$, the corresponding normalized quality $\tilde{Q}_l(\tilde{R}_l) = e^{-1-\frac{1}{\beta}+\alpha}$. With the parameters shown in Table 3.1, we have the normalized quality alway less than 0.1 and $\tilde{R}_l \approx 0.015$. As a normalized quality of 0.1 (e.g., 1 on a 10 rating scale) is considered very annoying and unacceptable, the video stream should never be extracted at a normalized rate lower than this bound. With a SVC video, the lowest rate is the base layer rate. In out test videos, the base layers all have normalized rate above 0.016 with normalized quality above 0.13, as indicated in Table 3.2. Therefore, we claim that for the practically meaningful range of rate, $\tilde{Q}(\tilde{R})$ or equivalently $\tilde{Q}(R)$, is concave. We can also observe the concavity from Fig. 3.3. So efficient and distributed algorithm also applies.

3.3 Quality Optimized Layer Ordering

To efficiently stream a pre-coded scalable video where the target bit rate is changing dynamically, it is desirable to pre-order the SVC layers in a rate-quality optimized manner, so that each additional layer yields the maximum possible quality improvement. With such a pre-ordered SVC stream, the proxy can simply keep sending additional layers, until the rate target is reached. Noting that each SVC layer (together with its previous layers) corresponds to a feasible (q, f) pair, the problem is equivalent to ordering the feasible (q, f) pairs, subject to decoding dependency constraint. In this section, we discuss how to employ the rate and quality models given in Sec. 3.2 to optimize the ordering of (q, f) pairs.

3.3.1 Ordering Strategy

We have shown that (3.4) and (3.5) can be employed to determine the optimal (q^*, f^*) which gives the best quality under a rate constraint. However, the resulting (q^*, f^*) might not always be feasible. For example, if a SVC stream is encoded with a dyadic temporal prediction structure, the feasible FR is doubled every time from the lowest to the highest FR. Similarly, there are only a small number of amplitude layers in a typical SVC stream, corresponding to a few discrete levels of QS q. In the following, we discuss how to take into account such practical limitations.

Suppose there are M temporal layers and N amplitude layers, the corresponding feasible choices of FR and QS are $\{f_1, f_2, ..., f_M\}$ and $\{q_1, q_2, ..., q_N\}$, respectively. We can construct a table which gives all possible combinations, each indicated by a quadruplet $(R_i, f_i, q_i, \tilde{Q}_i)$. If some combination in this table has higher R but lower $\tilde{Q}$ than at least one other combination, then it is clearly not rate-quality optimal. We can eliminate these points and order the remaining points in increasing rates with two steps. The first step is to sort all points in terms of their rates from low to high. Then starting from the point with the second lowest rate to the end (the first point corresponds to the base layer), we compare the quality of the current point to that of the previous kept point; we remove the current point if the quality is less or equal, otherwise keep the current point. We denote the table that contains the remaining entries as $\mathcal{T}_{init}$, which is applicable to real-time encoding decision. The complexity of generating this table is $O(MN)$.

The points in $\mathcal{T}_{init}$ has the property that as the rate increases, the quality also increases. Some of the points in $\mathcal{T}_{init}$ may not be optimal in the sense that they do not provide the maximum possible quality improvement for the incurred rate increment. Furthermore, some points in $\mathcal{T}_{init}$ may not satisfy the SVC decoding dependency. For example, a current point may have a FR that is lower than the previous point, or a QS that is higher than the previous point. There are multiple ways to remove the non-feasible points in $\mathcal{T}_{init}$ based on the rationale that the next feasible point can be either increasing in FR or decreasing in QS or both. We use the following Algorithm 1 to remove the non-optimal and non-feasible points in $\mathcal{T}_{init}$, and create the table $\mathcal{T}_{opt}$. The associated complexity is also $O(MN)$.

Algorithm 1 Generate rate-quality optimized table $\mathcal{T}_{opt}$

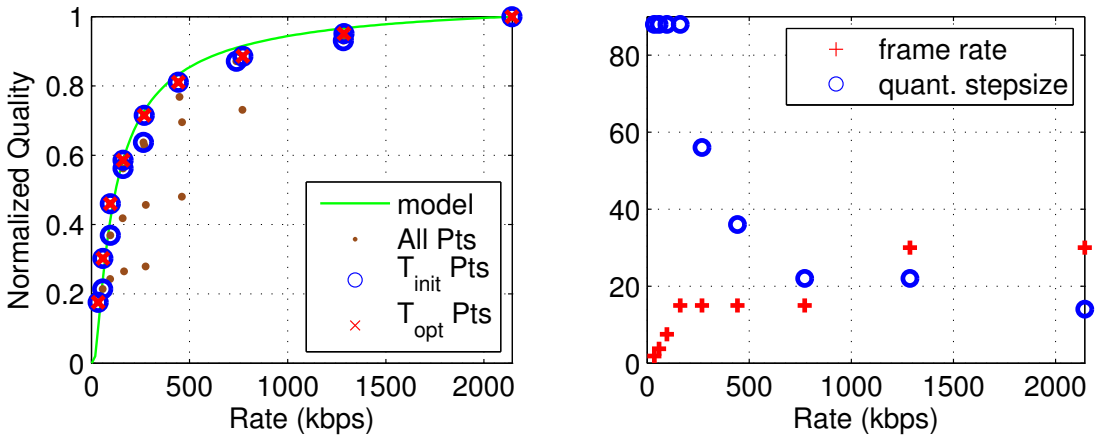
put $\mathcal{T}_{init}(1)$ into table $\mathcal{T}_{opt}$; $idx \leftarrow 1$.
while $idx < len(\mathcal{T}_{init})$ **do**
  for $cand\_idx \in \{idx+1, ..., min(idx+3, len(\mathcal{T}_{init}))\}$ **do**
    if $\mathcal{T}_{init}(cand\_idx)$ has lower FR or higher QS than last point in $\mathcal{T}_{opt}$ **then**
      continue;
    end if
    calculate $\delta Q/\delta R$ from points $\mathcal{T}_{init}(cand\_idx)$ and $\mathcal{T}_{init}(idx)$;
  end for
  find $cand\_idx$ which gives the highest $\delta Q/\delta R$ value;
  put $\mathcal{T}_{init}(cand\_idx)$ into table $\mathcal{T}_{opt}$;
  $idx \leftarrow cand\_idx$;
end while

3.3.2 Performance Evaluation

We apply the proposed layer ordering strategy to various sequences. Figure 3.4 shows the rate-quality relations of points contained in table $\mathcal{T}_{init}$ and $\mathcal{T}_{opt}$ and the corresponding (q, f) for each $\mathcal{T}_{opt}$ point obtained using Algorithm 1 for sequence FOOTBALL. Clearly, many feasible points are not optimal and are removed to get $\mathcal{T}_{init}$. A few points in $\mathcal{T}_{init}$ are still not optimal or do not satisfy the decoding dependency, which are removed to get $\mathcal{T}_{opt}$. Note that the removed points in $\mathcal{T}_{init}$ tend to have a much smaller quality improvement compared to a neighbor point with slightly larger rate. As the target rate increases, the FR monotonically increases while the QS monotonically decreases, thereby satisfying dependency across the layers. The points in $\mathcal{T}_{opt}$ follow the concave shape of the rate-quality curves in Fig. 3.3 very closely, indicating that ordering SVC layers based on these points yields near-optimal rate-quality tradeoff. This is also true for other sequences as shown in Fig. 3.5. Table 3.2 summarizes the entries in $\mathcal{T}_{opt}$ for all seven sequences.

Table 3.2: Entries of $\mathcal{T}_{opt}$ for seven sequences. Each entry is a quadruplet of $(R, f, q, \tilde{Q})$.

Index	1	2	3
AKIYO	(4.8, 1.875, 88, 0.21)	(6.6, 3.75, 88, 0.34)	(9.1, 7.5, 88, 0.47)
CITY	(19.1, 1.875, 88, 0.19)	(26.7, 3.75, 88, 0.31)	(37.3, 7.5, 88, 0.43)
CREW	(21.9, 1.875, 88, 0.13)	(34.8, 3.75, 88, 0.22)	(55.4, 7.5, 88, 0.31)
FOOTBALL	(34.7, 1.875, 88, 0.18)	(58.0, 3.75, 88, 0.30)	(96.7, 7.5, 88, 0.46)
FOREMAN	(19.9, 1.875, 88, 0.18)	(29.5, 3.75, 88, 0.29)	(43.9, 7.5, 88, 0.40)
ICE	(16.0, 1.875, 88, 0.15)	(25.4, 3.75, 88, 0.26)	(40.4, 7.5, 88, 0.37)
WATERFALL	(13.0, 1.875, 88, 0.17)	(17.5, 3.75, 88, 0.27)	(23.5, 7.5, 88, 0.39)
	4	5	6
AKIYO	(15.8, 7.5, 56, 0.61)	(21.9, 15, 56, 0.70)	(37.4, 15, 36, 0.82)
CITY	(64.0, 7.5, 56, 0.57)	(89.5, 15, 56, 0.66)	(151.6, 15, 36, 0.80)
CREW	(97.2, 7.5, 56, 0.47)	(168.4, 7.5, 36, 0.73)	(268.1, 15, 36, 0.73)
FOOTBALL	(161.4, 15, 88, 0.59)	(268.8, 15, 56, 0.72)	(442.5, 15, 36, 0.81)
FOREMAN	(73.7, 7.5, 56, 0.56)	(122.5, 7.5, 36, 0.69)	(181.8, 15, 36, 0.78)
ICE	(64.6, 7.5, 56, 0.52)	(102.3, 7.5, 36, 0.64)	(162.8, 15, 36, 0.76)
WATERFALL	(42.2, 7.5, 56, 0.54)	(56.9, 15, 56, 0.63)	(100.8, 15, 36, 0.77)
	7	8	9
AKIYO	(68.0, 15, 22, 0.92)	(117.7, 15, 14, 0.98)	(163.2, 30, 14, 1.0)
CITY	(273.0, 15, 22, 0.91)	(468.3, 15, 14, 0.97)	(658.3, 30, 14, 1.0)
CREW	(494.5, 15, 22, 0.87)	(867.4, 15, 14, 0.97)	(1381.4, 30, 14, 1.0)
FOOTBALL	(771.2, 15, 22, 0.87)	(1287.1, 30, 22, 0.95)	(2154.1, 30, 14, 1.0)
FOREMAN	(320.2, 15, 22, 0.90)	(538.2, 15, 14, 0.98)	(805.5, 30, 14, 1.0)
ICE	(271.5, 15, 22, 0.89)	(434.3, 15, 14, 0.96)	(693.0, 30, 14, 1.0)
WATERFALL	(190.7, 15, 22, 0.89)	(342.2, 15, 14, 0.97)	(462.9, 30, 14, 1.0)

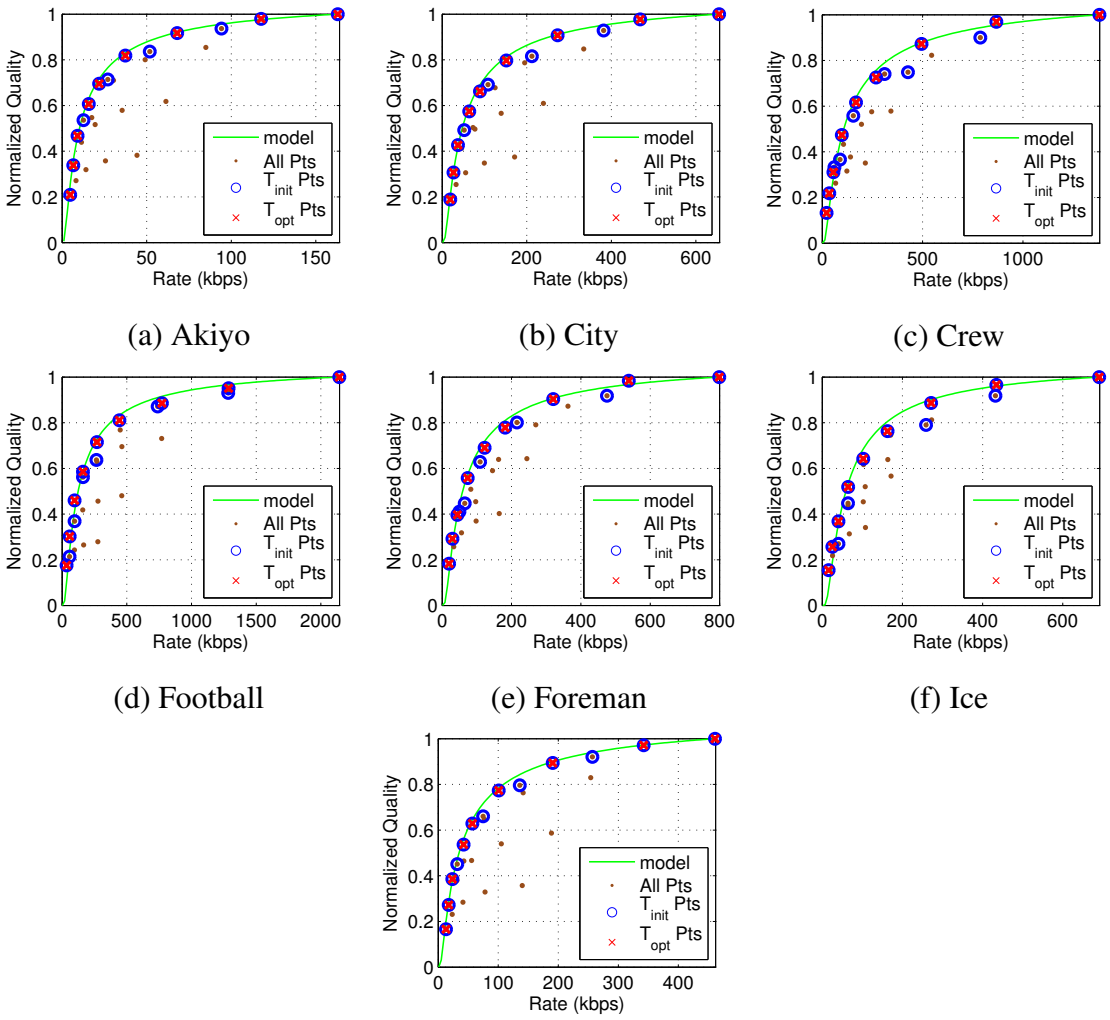


(a) (b)

Figure 3.4: Quality optimized table for the sequence FOOTBALL. (a) rate-quality tradeoff of points in $\mathcal{T}_{init}$ and $\mathcal{T}_{opt}$; (b) corresponding (q, f) values for points in $\mathcal{T}_{opt}$.

3.4 Summary

In this chapter, we propose a rate-quality model for SVC coding considering temporal scalability and amplitude scalability. The model has a simple form such that it can be easily used to optimize video system with network utility maximization framework. We also propose a quality optimized SVC layer ordering strategy so that each additional layer provides best possible quality gain given the bitrate increment. The layer decoding dependency is also satisfied for all ordered rate points.



(a) Akiyo (b) City (c) Crew

(d) Football (e) Foreman (f) Ice

(g) Waterfall

Figure 3.5: Quality optimized table for all sequences.

Chapter 4

Proxy-based Multi-Stream Adaptation over Wireless

Despite growing maturity in broadband mobile networks, wireless video streaming remains a challenging task, especially in highly dynamic environments. Rapidly changing wireless link qualities, highly variable round trip delays, and unpredictable traffic contention patterns often hamper the performance of conventional end-to-end rate adaptation techniques. This chapter presents how to improve wireless video streaming service using proxy and SVC quality optimized rate adaptation.

4.1 Background

4.1.1 Previous Work

It has been long agreed that the video streaming rate needs some form of adaptation to match the time-varying wireless channel capacity [55], to provide a better user experience. At the encoder end, techniques such as adaptive encoder rate control [56, 57], transcoding [58], and bitstream switching [59] are proposed to dynamically adjust video rate. A viable alternative to this is scalable video coding, whereby a stream only needs to be encoded once yet can be flexibly decoded at several different target rates [11]. Such a design greatly facilitates on-the-fly adaptation of the spatial resolution, temporal rate, and frame quality (controlled by QS) of the transmitted video stream. The recently standardized SVC extension in H.264, in particular, has succeeded in achieving comparable coding efficiency

as the non-scalable encoding in H.264/AVC [60]. Hence it is especially appealing for video streaming in a mobile environment [61]. However, given a target rate, there are many possible combinations of SVC spatial, temporal and amplitude layers that can lead to different perceptual quality. Therefore, adaptation of SVC streams subject to a rate constraint is not a trivial problem. Using the rate and quality models in [13], we can pre-order the temporal and amplitude layers in a SVC stream to reach rate-quality optimality, greatly simplifying the SVC adaptation problem.

In terms of underlying rate control protocol/algorithm, many conventional schemes rely on equation-based TCP-friendly rate control (TFRC) [14] for regulating the rate of each stream [62, 63]. However, these end-to-end schemes often suffer from slow convergence when the bottleneck link bandwidth changes rapidly, and lead to allocation results oblivious of video content characteristics. [64] proposed a TCP-friendly video transport protocol targeting for wireless enviorment, but it is still content-agnostic. [65–68] rely on using a video rate-distortion model to solve network resource allocation while providing video content-awareness. The model is however only applicable for videos at a fixed frame rate. Our work stands apart from existing approaches by combining the rate and quality adaptation capability of H.264/SVC with a rate-quality tradeoff model that considers effect of both frame rate and quantization stepsize on the rate and quality. In addition, link bandwidth heterogeneity is considered in the rate allocation. The proposed algorithm enables a fast converging, quality optimized rate allocation at the proxy node. The proposed system is capable of both closely following dynamics in the wireless link bandwidth and tailoring the rate allocation for each stream based on its own rate-quality tradeoff and the effective link bandwidth, and choosing the optimal combination of frame rate and quantization stepsize that maximizes the quality for a given rate.

4.1.2 Proposed System: Overview

In a wireless video streaming system, the video server maintains the original video bistreams. Upon requests for certain video contents, the server will send the (sub)stream through IP network to the wireless access node via which the (sub)stream is served to end-

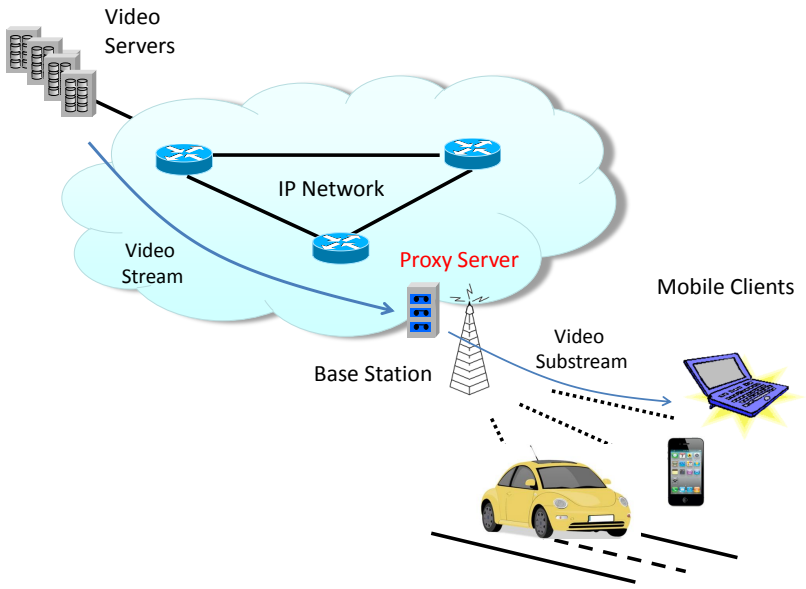


Figure 4.1: Architecture overview of the adaptive video streaming system. A proxy node at the edge of the network performs video adaptation before relaying video streams to mobile clients.

users. The wireless access node is usually the bottleneck where the congestion is likely to occur as it is shared by many users and has relatively low bandwidth capacity compared to the IP network. In order to track the wireless link status, an end-to-end feedback mechanism would be expected to inform the video adapter if it is at the server. However, the end-to-end delay is typically large in a wireless environment.

To agilely trace the link status, we envision a proxy node colocated with the access node. It is in charge of tracking the time-varying status of the wireless access link, while dynamically adapting the traversing scalable video streams. Fig. 4.1 provides an architectural overview of the proxy-based video streaming system. The benefit of such a design is multifold. First, it requires no additional modifications at either the video servers or the mobile clients; video adaptation is performed at the proxy and is agnostic to both ends. Second, since the proxy node is located right at the bottleneck wireless node, it can react much more agilely than end-to-end adaptation schemes in face of abrupt changes over the wireless hop. Furthermore, the proxy node has knowledge and control of *all* traffic travers-

ing the bottleneck wireless link, therefore is well-positioned to optimize the allocated rate *across* the competing streams in a more holistic manner.

The main drawback of this architecture is the potentially large wasted bandwidth on the IP network. By the time adaptation happens at the proxy, the required video (sub)streams are expected to be arrived. Therefore, the server needs to send the whole bitstream to the proxy or send over-provisioned (sub)streams based on some prediction algorithms. Considering that the bandwidth is abundant in the core network and the prevalent deployment of caching servers in the network, this is acceptable.

4.2 System Model and Algorithms For Rate Allocation

In this section, we develop a subjective quality maximization framework for rate allocation among multiple wireless receivers under the same wireless access node. The framework is based on the rate-quality model in Chapter 3 and can be easily adapted to accommodate other models.

4.2.1 Problem Formulation

Consider a set of video receivers sharing a common access node. For each receiver $i \in \{1, 2, ..., I\}$ experiencing a wireless link throughput of C_i, the interested video can be adapted to have an equivalent coding parameter setting indicated by (q_i, f_i), which results in subjective quality of $Q_i(q_i, f_i)$ and video bitrate of $R_i(q_i, f_i)$. In addition, suppose there are a set of background flows, each generates rate $R_j, j \in \{1, 2, ..., J\}$. From system-wide of view, the optimal network utilization is achieved by solving the following utility

maximization problem.

$$\max_{\vec{q},\vec{f}} \sum_{i=1}^{I} w_i \tilde{Q}_i(q_i, f_i) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{J} \frac{R_j}{C_j} + \sum_{i=1}^{I} \frac{R_i(q_i, f_i)}{C_i} \leq 1 \tag{4.2}$$

$$q_i \in \Omega_i, \quad f_i \in \Gamma_i \quad \forall i = 1, \ldots, I \tag{4.3}$$

Here Ω_i and Γ_i denote the possible choice of QS and FR for the video requested by receiver i, respectively. The objective function (4.1) is the weighted sum of subjective quality over all video receivers where important receivers will be assigned with a larger weight. The constraint (4.2) ensures that the aggregated channel utilization time is below the maximal system utilization ratio (which we assume to be 1 here, but can in general be less than 1). (4.3) specifies the coding parameter constraints.

The computational complexity involved in solving the optimization problem increases with the dimension of the coding parameter set and the number of video streams. Besides, these coding parameters are usually integers. Thus, the problem becomes combinatorial hence computationally expensive. Suppose there are N possible choices of q, M possible choices of f and I video streams, the number of possible combinations would be $(MN)^I$.

We, instead, propose to solve a relaxed form of this problem by allowing continuous choices of video rates, and by leveraging the optimal rate-quality tradeoff developed in Chapter 3. After obtaining candidate sending rates from the first step, we can then simply pump pre-ordered video packets into the network as described in Chapter 3.

To solve for the candidate sending rate for each video receiver, we reformulate the problem as follows.

$$\max_{\vec{R}} \sum_{i=1}^{I} w_i \tilde{Q}_i(R_i) \tag{4.4}$$

$$\text{s.t.} \quad \sum_{j=1}^{J} \frac{R_j}{C_j} + \sum_{i=1}^{I} \frac{R_i}{C_i} \leq 1 \tag{4.5}$$

$$R_i^{min} \leq R_i \leq R_i^{max} \quad \forall i = 1, \ldots, I \tag{4.6}$$

where R_i^{min} denotes the base layer bitrate for video i.

4.2.2 Iterative Solution

Lemma 1. *In practical video rate regions, the problem* (4.4)-(4.6) *is a concave maximization problem.*

Proof. Notice the Claim 1 in Chapter 3 and summation preserves the concavity. □

Concavity of the objective function (4.4) ensures that the local maximum is also the global maximum. We propose to use an iterative algorithm to approach the optimal point. The reason behind this is trifold: 1) if the bottleneck link is highly dynamic with time-varying bandwidth and delay, the results by directly solving problem (4.4)-(4.6) given past observation (or estimation) of R_j, C_j and C_i could be highly skewed; 2) directly solving problem (4.4)-(4.6) incurs much larger complexity and overhead; 3) directly solving problem (4.4)-(4.6) requires an accurate observation (or estimation) of R_j, C_j and C_i, otherwise, the results can be invalid even if computed at a high frequency. Iterative solutions, on the other hand, are generally more robust against variations of the network conditions and measurement errors [69].

Denote $x := \sum_{j=1}^{J} R_j + \sum_{i=1}^{I} R_i$ and $y := \sum_{j=1}^{J} R_j/C_j + \sum_{i=1}^{I} R_i/C_i$ the instantaneous incoming rate and the required serving time at the access point, respectively. Then $\tilde{C} := x/y$ is the instantaneous effective link outgoing rate. Following the same idea of the primal-dual algorithm in network rate control [35, 54], we propose the following two iterative steps:

$$\dot{R}_i = \theta R_i \left(w_i \tilde{Q}_i'(R_i) - \frac{p}{C_i} \right) \tag{4.7}$$

$$\dot{p} = \phi \left(y - 1 \right) \frac{x}{y} = \phi(x - \tilde{C}) \tag{4.8}$$

Here θ, ϕ are two scaling factors; $\tilde{Q}_i'(R_i)$ is the first derivative of $\tilde{Q}_i(R_i)$ w.r.t R_i. p denotes the price of using the link and $\tilde{C} = x/y$.

Intuitively, the value of p increases when the network is temporarily over-congested, leading to a negative or slower increment of R_i, whereas temporarily underutilization of the network results in decreased p and consequently higher R_i from all contributing streams.

Theorem 1. *The iterative algorithm of* (4.7)-(4.8) *will converge to an equilibrium point which solve the probem* (4.4)-(4.6) *in a time-sharing wireless network under static channel conditions.*

Proof. The constrained optimization problem of (4.4)-(4.6) can be converted to maximizing the following objective function

$$\begin{aligned} L = & \sum_{i=1}^{I} w_i \tilde{Q}_i(R_i) - q(y-1) \\ & - \sum_{i=1}^{I} h_i(R_i^{min} - R_i) - \sum_{i=1}^{I} k_i(R_i - R_i^{max}) \end{aligned}$$

where $q, \vec{h}, \vec{k}$ are Lagrange multipliers. The Karush-Kuhn-Tucker (KKT) conditions are as follows:

$$\frac{\partial L}{\partial R_i} = w_i \tilde{Q}_i'(R_i) - \frac{q}{C_i} + h_i - k_i = 0 \quad \forall i \tag{4.9}$$

$$q(y-1) = 0 \tag{4.10}$$

$$h_i(R_i^{min} - R_i) = 0 \quad \forall i \tag{4.11}$$

$$k_i(R_i - R_i^{max}) = 0 \quad \forall i \tag{4.12}$$

Note that the algorithm of (4.7)-(4.8) is essentially gradient descending algorithm, which stabilizes at certain point $(\vec{R}^*, p^*)$, i.e. when $\dot{R} = 0$ and $\dot{p} = 0$. Next, we show that the equilibrium point $(\vec{R}^*, p^*)$ achieved by the algorithm of (4.7)-(4.8) solves the problem (4.4)-(4.6) by examining the satisfaction of its KKT conditions.

Case 1: if $p^* \neq 0$ and $\vec{R}^{min} < \vec{R}^* < \vec{R}^{max}$, the point $(\vec{R}^*, p^*)$ satisfies

$$w_i \tilde{Q}_i'(R_i^*) = \frac{p^*}{C_i} \quad \forall i$$

$$\sum_{j=1}^{J} R_j + \sum_{i=1}^{I} R_i^* = \tilde{C}^* = \frac{\sum_{j=1}^{J} R_j + \sum_{i=1}^{I} R_i^*}{\sum_{j=1}^{J} \frac{R_j}{C_j} + \sum_{i=1}^{I} \frac{R_i^*}{C_i}}$$

Then, we have the corresponding $\vec{h} = 0, \vec{k} = 0, q = p^*$ for the KKT condition.

Case 2: if $p^* \neq 0$ and $R_i^* = R_i^{min}$ for some i, we have $h_j = 0 \; \forall j \neq i, \vec{k} = 0, q = p^*$ and $h_i = -w_i \tilde{Q}_i'(R_i^{min}) + \frac{p^*}{C_i}$ for the KKT condition.

Case 3: if $p^* \neq 0$ and $R_i^* = R_i^{max}$ for some i, we have $\vec{h} = 0, k_j = 0 \; \forall j \neq i, q = p^*$ and $k_i = w_i \tilde{Q}_i'(R_i^{max}) - \frac{p^*}{C_i}$ for the KKT condition.

Case 4: if $p^* = 0$, then $R_i^* = R_i^{max}$ according to (4.7). So, we have $\vec{h} = 0, q = 0$ and $k_i = w_i \tilde{Q}_i'(R_i^{max}) \; \forall i$ for the KKT condition. □

4.3 Practical System Design

Previously, we have developed the mathematical framework for solving the system-wide quality maximization problem. In this section, we focus on how to implement the proposed iterative solution in a practical wireless video streaming system.

4.3.1 Implementing the Algorithm at the Proxy

The iterative algorithm consists of two processes. The first process, which follows (4.7), updates the streaming rate given the observation of p and $\vec{C}$, which is the vector containing the observed link throughputs of all video receivers. The second process, with (4.8), determines a new link price given the observation of the sending rates in the last update interval from all users $\vec{R}$ and the effective serving rate $\tilde{C}$. Note here, $\tilde{C}$ depends on both $\vec{C}$ and $\vec{R}$. Suppose the proxy adapts video stream every τ seconds, we write (4.8) in a discretized form,

$$p(n+1) = max\left(p(n) + \phi\left(x(n) - \tilde{C}(n)\right)\tau, 0\right) \tag{4.13}$$

The middle term in (4.13), i.e. $x(n) - \tilde{C}(n)$ is in fact the evolution of queue length at the access node. At time index $n+1$, the new stream rate for video i is calculated as

$$R_i(n+1) = min\left(R_i^{max}, max\left(R_i^{min}, R_i(n) + \theta R_i(n)\left(w_i \tilde{Q}_i'(R_i(n)) - \frac{p(n)}{C_i(n)}\right)\tau\right)\right) \tag{4.14}$$

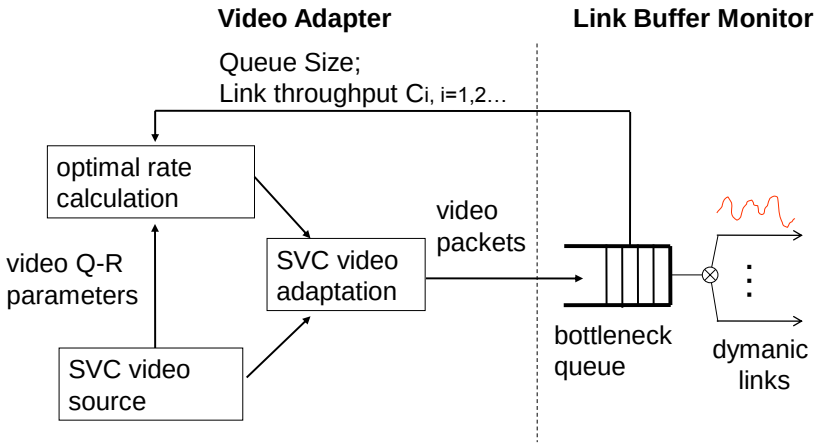


Figure 4.2: Main components in proxy-based adaptation architecture.

and according to the rate-quality model in (3.7), $\tilde{Q}_i'(R_i)$ is given by

$$\tilde{Q}_i'(R_i) = \alpha\beta \left(\frac{R_i}{R_i^{max}}\right)^{-\beta} R_i^{-1} e^{-\alpha\left(\frac{R_i}{R_i^{max}}\right)^{-\beta}+\alpha}$$

We implement (4.13) and (4.14) in two separate modules: link buffer monitor and video adapter both at the proxy. Fig. 4.2 shows the diagram of the two modules and the signaling in between. The link buffer monitor checks the bottleneck queue length once every τ seconds. It is also responsible for estimating the link throughput C_i for each receiver i. In our system, the packets' inter-departure time at the interface queue is inspected and it is used to derive the instantaneous throughput of the link that transports the packet under consideration (via dividing the packet length by the inter-departure time for that packet). Then, the link throughput C_i can be estimated by averaging over a number of packets.

The optimal rate allocation module will calculate the new stream rate based on the feedback from the link buffer monitor and the video rate-quality parameters embedded in the SVC stream. Then, the SVC stream is adapted to the new rate by simply sending video packets up to the target rate assuming stream is pre-ordered in the quality-optimized manner. Note that the pre-ordering should ideally be done at video encoder so that the video streams arriving at the proxy are already in optimal orders. However, it is possible to have the proxy to order the SVC layers if the video servers do not have pre-ordered streams and are agnostic of the rate-quality model adopted by the proposed proxy-based adaptation

system.

4.3.2 Discussions

Feedback Interval

For CGS video streams, the rate adaptation can only be carried out at the boundary of GOPs. For example, with a CGS encoded video stream with GOP size of 16, the rate switching can only be done every $16/30 = 0.533$ seconds assuming 30 frames per second. On the other hand, for the proxy to accurately track the link status, it should run the iterative algorithm of (4.13)-(4.14) at a much faster update frequency. To coop with this situation, we run the iterative algorithm with a short update interval, but only adapt the video rate at the beginning of each new GOP. A filter may be applied to obtain a smoother sending rate. In our simulation, the smoothed sending rate at the begining of each GOP is calculated as a weighted sum of the current sending rate determined from (4.14) and the smoothed sending rate for the previous GOP with coefficients 0.8 and 0.2 respectively. Then, the video rate is adapted according to the most recent smoothed sending rate.

Iterative Algorithm vs. Exhaustive Search

We carried out some numerical case studies and found that the iterative algorithm incurs at most 5% efficiency loss (in terms of the total utility at the same rate) compared with the brute-force exhaustive search approach as shown in Fig. 4.3. The loss is largely due to the discrete nature of the feasible rate points. An allocated rate (which is determined assuming any rate is achievable) is not always fully utilized. The complexity for exhaustive search is on the order of $O(25^3)$, while the iterative algorithm is $O(1)$ at each iteration. The corresponding running time for the MATLAB scripts is 500ms and 0.17ms, respectively. Through our extensive simulations, we found that the discrete nature of the feasible rate points does not impact the algorithm convergence speed much.

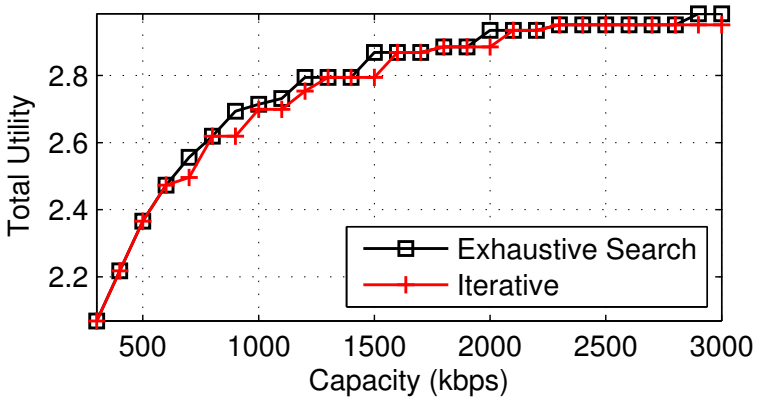


Figure 4.3: Performance comparison of the iterative algorithm with the exhaustive search. Three sequences (AKIYO, FOREMAN and FOOTBALL) are requested by three receivers and all the receivers have the same link throughput, ranging from 300kbps to 3Mbps. Five amplitude layers and five temporal layers are generated, thereby allowing 25 discrete quality-rate points. But only one of those points given in Table 3.2 is chosen for each allocated rate for a sequence.

Limitation of the Effective Link Bandwidth Estimation Method

As we assume there is no cross-layer information, i.e., the information from MAC layer and below, exposed to the proxy, the proxy is agnostic to the packet loss below the link layer. This limitation will result in inaccurate bandwidth estimation, especially when there are severe packet losses. For example, in WiFi system, a packet may undergo several retransmissions and finally be dropped at the MAC layer, therefore the instantaneous link throughput is 0. However, the proxy had only observed the packet sojourn time at the MAC layer, and calculated the instantaneous link throughput as the packet size divided by the sojourn time (which is equivalent to the packet inter-departure time at the interface queue if other processing overheads are negligible). Thus, the resulting estimation does not consider potential packet loss after the maximum retransmission limit is reached.

Equalizing Receivers' Subjective Qualities

In the formulation (4.4)-(4.6), the objective is to maximize the aggregated weighted system utility. When all the weights are equal, the quality at all receivers are generally not equal. By appropriately choosing the weights, we can equalize the quality of all receivers, or making some receivers enjoying higher quality. Without detailed derivations, we can show that setting $w_i \propto R_i^{max}/C_i$ for each receiver i leads to equalized subjective qualities at the receivers.

4.4 Performance Evaluation

In this section, we evaluate our system design with extensive simulations based on the `ns-2` [70] simulator. We implemented video adaptation agents and a video player emulator which generates video playback trace. We conducted two sets of simulations. The first set, targeting for evaluating the effectiveness of the proxy-based design in adapting the sending rate based on the time-varying channel condition, is driven by a real-world wireless measurement trace. The impact on background traffic (e.g. TCP) is also evaluated. The second set of simulations investigate the effectiveness of the proposed approach in rate allocation among multiple receivers based on both channel conditions and video characteristics. We assume the channel conditions of all receivers are static so as to isolate the effect of channel dynamics.

4.4.1 Common Simulation Settings

We use real video packets traces to drive the simulations. Two typical video sequences are used in the simulations: FOREMAN and FOOTBALL, representing a slow-to-medium motion clip and a highly intense motion clip, respectively. Both sequences have a spatial resolution of 352×288 pixels (CIF) and temporal rate of 30 fps. We encode the sequences with JSVM version 9.12 [43] to generate SVC streams with 5 CGS layers and 5 temporal layers, and then pre-order the packets in a quality-optimized manner. The feasible

rate points are as shown previously in Table 3.2. Note that the maximum rates needed to achieve the highest quality are very different for the two videos, 0.8Mbps and 2.1Mbps, respectively. This means that to achieve similar qulity, FOOTBALL should be allocated much higher rate.

We choose TFRC as a comparison rate control mechanism, as it is considered suitable for media streaming. We applied the Datagram Congestion Control Protocol (DCCP) patch for `ns-2`[1] which contains a TFRC implementation. DCCP does not require reliable in-order delivery of packets and the stream rate will be determined by the TFRC solely.

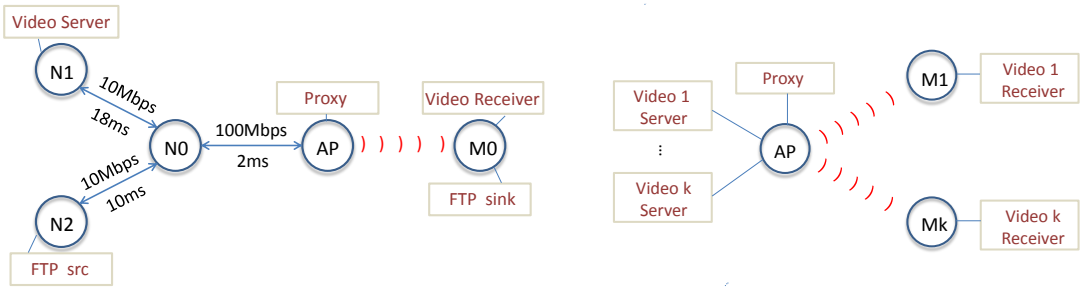
In our simulations, we choose $\theta = 1$, $\phi = 5I$(1/Mbps) where I denotes total number of receivers, and $w_i = 1$ for every receiver i unless otherwise stated. The proxy senses the interface queue every 50ms and estimates the effective bandwidth for each link by averaging over 16 most recent packets. The original candidate sending rate for each video is updated using (4.14). A smoothed candidate sending rate is calculated as discussed in Sec. 4.3.2 and the actual video rate is only changed at the beginning of each GOP. The rate point among those in Table 3.2 that is closest to the smoothed candidate rate from below is chosen. The packet size is set to 500 bytes and if a video NAL unit size is larger than that, it will be segmented into several packets. The access point queue size is set to 75 packets as suggested in [71].

We set the video playback delay to be 2 seconds which means the player will start 2 seconds later after receiving the first video packet. If any packet belonging to a NAL unit is lost during transmission, the entire NAL unit is discarded as well as all other NAL units that depend on it. Remaining NAL units that meet the playback deadline are used to determine the highest decodable temporal layer and amplitude layer, which correspond to a certain FR f and QS q. This (q, f) pair is used to derive the normalized quality using (3.1).

4.4.2 One Receiver with a Dynamic Wireless Environment

In this set of simulation, we are interested in the responsiveness of our system design to the dynamic behavior of the wireless link. The most important aspect is how fast

[1] http://eugen.dedu.free.fr/ns2/dccp-ns2.34.patch



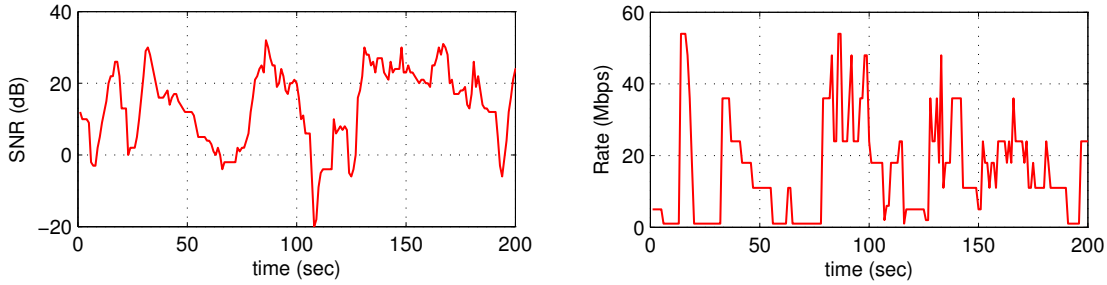
(a) (b)

Figure 4.4: Simulation topology setup. (a) topology for the first set of simulations; (b) topology for the second set of simulations.

the proposed design can tract and react to the wireless link quality changes. Besides, it is also important to study how video traffic interacts with background TCP flows. Only FOOTBALL sequence is used in this part.

Setup

Figure 4.4(a) illustrates the simulation topology setup, where the video server is attached to node N1 and the proxy is located at the AP. A video client agent and a video player agent are attached to node M0. There is also a background TCP traffic generated by a FTP session from node N2 to node M0, traversing the same AP. The wired link between node N0 and AP is of 100Mbps and 2ms delay; the wired link between N1 and N0 has 10Mbps and 18ms delay while the wired link between N2 and N0 has 10Mbps and 10ms delay; the wireless link between AP and M0 is driven by a real-world measurement trace obtained when driving around Mountainview, CA. The average speed of the vehicle was around 20mph. Fig. 4.5 shows the signal to noise ratio (SNR) (in dB) and PHY rate (in Mbps) over time. These two traces are used to generate PER rate and calculate transmission time. Note here, the effective link rate is always less than the PHY rate due to retransmission at the PHY layer up to a present retransmission limit. The video server starts streaming at time 0 while FTP session starts at time 80. Both flows end at time 200.

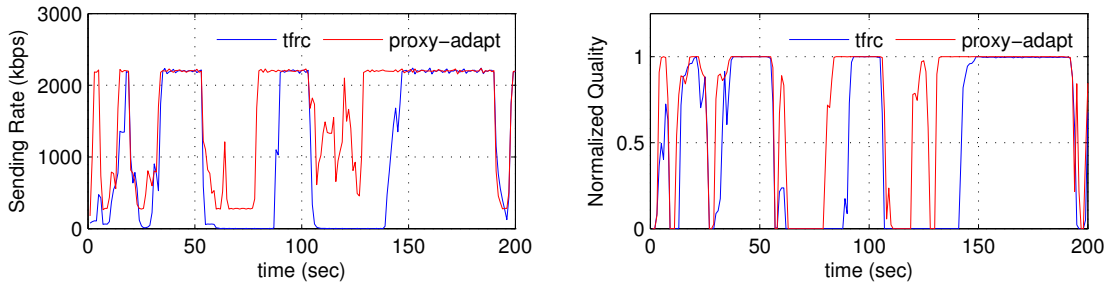


(a) Signal-to-Noise Ratio (b) PHY rate

Figure 4.5: Wireless SNR and PHY rate traces from a real-world measurement. The trace was collected while driving around Mountainview, CA and the average speed of the vehicle was around 20mph.

Proxy Adaptation vs TFRC

Figure 4.6 shows the performance comparison for proxy based adaptation and TFRC scheme. From Fig. 4.6(a), it can be seen that when the channel condition is good and stable, e.g., around time 50, 90 and 150, TFRC achieves good performance. However, when the channel quality changes dynamically, TFRC can not react agilely experiencing a slow convergence speed, for example during time 75-85 and 125 to 140. On the other hand, proxy-based adaptation can adapt the sending rate quickly, and the resulting video playback quality is significantly improved over the TFRC scheme. When the channel condition changes, the proxy-based adaptation can agilely follow up and stabilize at a high streaming rate. In terms of playback quality, the proxy-based adaptation provides higher quality than TFRC in most time, and the average quality over the entire duration is much higher, as shown in Fig. 4.6(c). It is worth noting here that since the proxy does not have information from the MAC layer, it does not know whether a packet is dropped or not and the link throughput estimation may not be accurate at severe packet loss. This can lead the proxy-based adaptation react improperly when continuous packet losses happen. For example, around time 110-120, the proxy is still sending out video packets which are all dropped (which can be inferred from the zero quality during that time period in Fig. 4.6(c)).



(a) sending rate

(b) normalized quality

Figure 4.6: Performance comparison of proxy-based adaptation and TFRC. The average sending rates for proxy-based adaptation and TFRC are 1601kbps and 1072kbps respectively. The average playback rates over time for proxy-based adaptation and TFRC are 1333kbps and 919kbps respectively (not shown here). The average normalized qualities over time for proxy-based adaptation and TFRC are 0.66 and 0.47 respectively.

When there are other flows destined to good wireless channels, the system will unnecessarily lower the rate assigned to those flows. To alleviate this problem, either cross-layer information exchange or user feedback can be considered so that the proxy can estimate the link throughput more accurately. We defer this to our future study. Figure 4.7 shows the throughput of the background TCP traffic over time. TCP throughputs are very similar under both types of competing video streams. This indicates that the proxy-based adaptation is also TCP-friendly.

4.4.3 Multiple Receivers with Static Behavior

In this second part, we isolate the randomness of the wireless network and use fixed wireless links to drive the simulator. The topology setup is given in Fig. 4.4(b) where two or more servers stream videos via individual wireless links to the receivers depends on the specific scenario. In each scenario, we set different link conditions for different receivers. In this set of simulations, there is no background traffic injected.

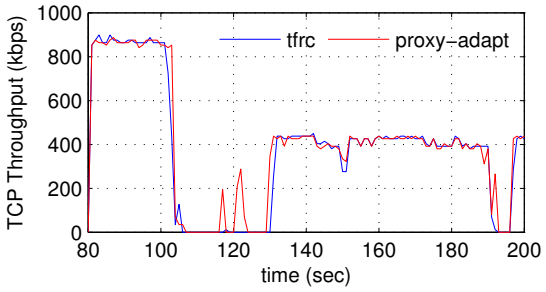
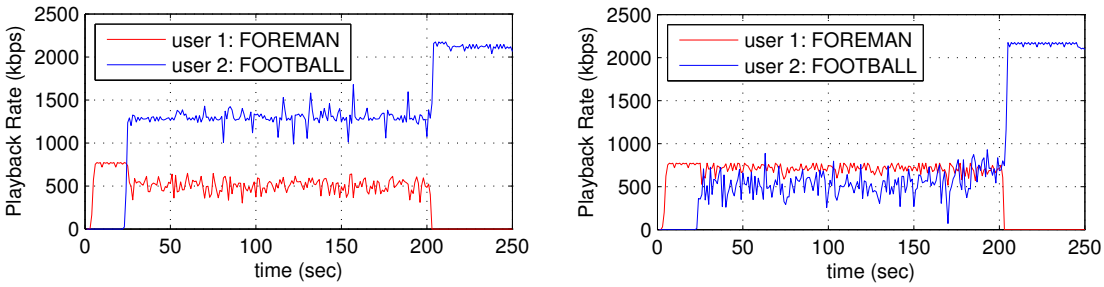


Figure 4.7: TCP throughput over time when competing with TFRC video stream and proxy-based adaptive video stream, respectively.



(a) proxy-based adaptation (b) TFRC

Figure 4.8: Comparison of video playback rate achieved by TFRC, and the proposed proxy-based scheme. In this simulation, the FOREMAN sequence and the FOOTBAL sequence share a base station with PHY rate of 1Mbps and 11Mbps respectively. FOREMAN lasts between 0-200 sec, FOOTBALL between 20-250 sec. Same weights are used for both receivers.

Content-Aware Rate Allocation among Two Users

We consider two nodes M1 and M2 share the same AP node. The link between AP and M1 has a PHY rate of 1Mbps while the link between AP and M2 has a PHY rate of 11Mbps. M1 streams the FOREMAN sequence starting from time 0 and lasts for 200 seconds, whereas M2 starts to stream FOOTBALL at time 20 seconds and finishes at time 250 seconds. Figure 4.8 shows the sending rate for the two competing video streams. As can be seen in Fig. 4.8 (a), with the proxy-adapt approach (with equal weights for the two videos) the rate allocation of two streams converges in a short time period with FOOTBALL being allocated more rate. After session 1 ends, session 2 quickly jumps to the full video rate. On the other hand, TFRC converges to a equilibrium point which gives similar rates to both sessions. Some rate variations are observed for the second stream (FOOTBALL) due to the limited rate choices. TFRC also requires more time to converge, e.g. when the first stream ends, TFRC takes 3 more seconds to converge to the full video rate for the second stream.

Performance with Varying Number of Users

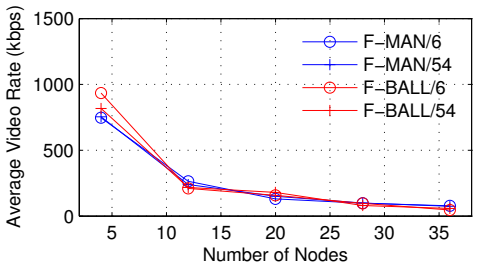
In this scenario, an AP is shared by multiple users, each receives one video stream. The number of concurrent users ranges from 4 to 36 with half of them receving FOREMAN and the others receiving FOOTBALL. In both groups of receivers, half of them have a good link condition of 54Mbps PHY rate and the others have a fair link condition of 6Mbps PHY rate. Equal weights are used for all receivers and they all start at time 0. The total simulation time is 100 seconds. Results are averaged over a duration of 60 seconds after the system has reached convergence.

Figure 4.9 shows the average video rate and the resulting normalized quality for each category of receivers. Comparing Fig. 4.9(a) and Fig. 4.9(b), we can easily notice the effectiveness the content-aware and link throughput-aware rate adaptation of the proxy-based approach. Fig. 4.9(a) shows that, with TFRC, all users are receiving similar video rate regardless of their video characteristics and link status. This also leads to the head-

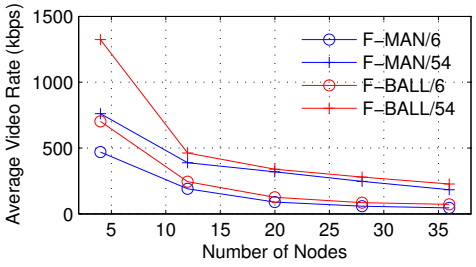
of-line blocking especially when the system is overloaded with large number of receivers. As can be seen in Fig. 4.9(a), the video rate for all receivers are similar to the receiver with 6Mbps PHY rate. Figure 4.9(b) confirms that with proxy-based scheme, FOOTBALL users will receive higher rate than FOREMAN users if their channel conditions are the same, whereas users with poor channel quality will receive lower rates than those with good channel quality. The resulting average normalized qualities for both scenarios are shown in Fig. 4.9(c) and Fig. 4.9(d). With TFRC, more complex video (FOOTBALL) is delivered with a lower quality. Also, same video content receivers have similar quality even though their link throughputs are different. On the other hand, with proxy-based scheme, received quality not only depends on the video content but also depends on the link throughput. It is also clear that proxy-based scheme gives larger aggregated quality than TFRC.

Differential Services

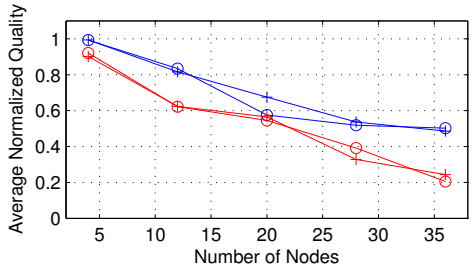
Previous results for the proxy-based approach are all obtained with equal weights for all users. Although the sending rates are content-aware, with higher rates allocated to more complex video, the received qualities are not equalized. By assigning different weights to different users, the proxy-based scheme provides the capability of differential services or equalizing receivers' qualities. We consider an AP being shared by 9 users divided into three groups. The first group, denoted as G1, receives FOREMAN; the second group, G2, receives FOOTBALL and the third group G3, also receives FOOTBALL. Two sets of weightings are investigated as shown in Table 4.1. The first one (denoted as Proxy-1), targeting for equal subjective quality, has weights 1 and 2.6 and 2.6 respectively for the three groups. These weights are chosen as discussed in Sec. 4.3.2. The second set (denoted as Proxy-2) has weights 1, 2.6 and 10 respectively for the three groups, so that G1 and G2 receive similar quality and G3 receives a higher quality. In practice, the second set may represent the case where one group of users paid premium price for better video services. All users have link PHY rate 12Mbps. As shown in Fig. 4.10, with proxy-based scheme and the first set of weights (Proxy-1), all FOOTBALL receivers obtain higher rates than FOREMAN receivers and all receivers receive very similar quality. With the second set of



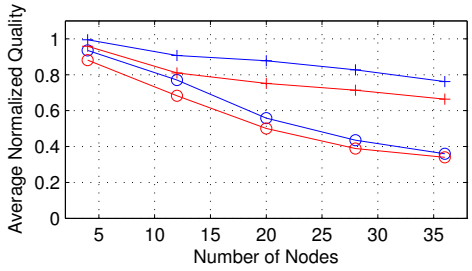
(a) Average video rate for each category of users: TFRC case



(b) Average video rate for each category of users: Proxy-adapt case



(c) Average normalized quality for each category of users: TFRC case

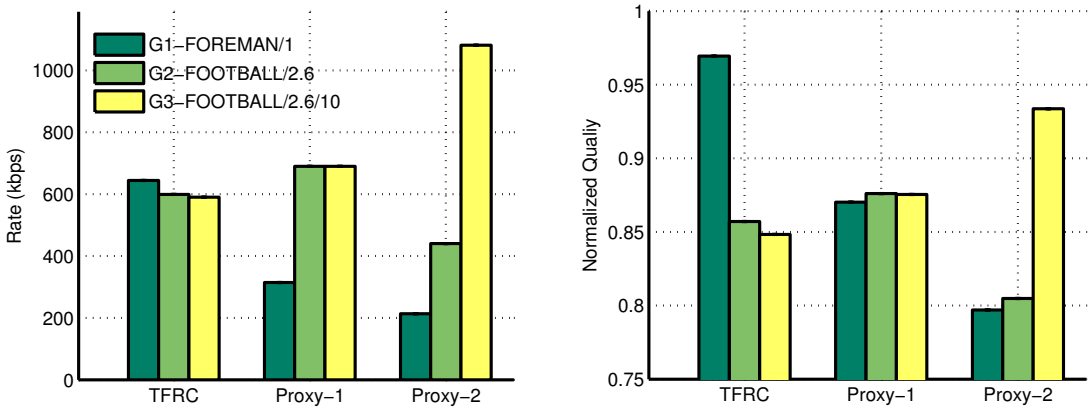


(d) Average normalized quality for each category of users: Proxy-adapt case

Figure 4.9: Normalized quality and video rate seen by users. The number of users ranges from 4 to 36 where half of them receive FOREMAN and the others receive FOOTBALL. In addition, half of the members within each group have 6Mbps PHY rate and the others have 54Mbps PHY rate.

Table 4.1: Weight settings for differential services

GROUP	Sequence	Proxy-1	Proxy-2
G1	FOREMAN	1	1
G2	FOOTBALL	2.6	2.6
G3	FOOTBALL	2.6	10



(a) Rate (b) Quality

Figure 4.10: Users' receiving rate and quality under the Proxy-based and TFRC schemes with heterogeneous weight assignment.

weights (Proxy-2), the premium users are able to receive a much higher quality than the other users. TFRC can not provide either equal or differentiated quality as it attempts to allocate same rates to all users. As a result, while FOREMAN users enjoy a higher quality, all FOOTBALL users experience much lower quality.

Chapter 5

Adaptive HTTP Video Streaming

Despite the increasing popularity gained in the industry, the research on HTTP adaptive streaming is still very preliminary. Typically, HTTP adaptive streaming hinges on client requesting video chunks sequentially therefore the client requesting strategy directly affects the video viewing quality. We start this chapter with an overview of adaptive HTTP video streaming system and current state-of-the-art. Then we propose a cost function for each chunk by jointly considering its contribution to the average playback level, the variation of playback level and the playback buffer level, to facilitate the chunk request strategy design. We show that the problem of finding the optimal chunk request strategy is de facto a Dynamic Programming problem. Due to the difficulty of directly solving the DP problem, we propose a heuristic algorithm to solve it. Finally, we demonstrate the potentials of employing scalable video coding by comparing the performance of adaptive HTTP streaming of single-layer video vs TCP-based SVC adaptive streaming.

5.1 Background

5.1.1 HTTP Progressive Download

The methods for delivering video are traditionally broken down into two categories: *streaming* and *download*. For streaming, Real Time Streaming Protocol (RTSP) and Real

Time Messaging Protocol (RTMP) are two most popular protocols, while download is typically associated with the hyper-text transfer protocol (HTTP). A recent trend in industry is to divide a whole video into small video segments or video chunks and rely on HTTP protocol to transport those chunks to client. While client is downloading new video chunks, he/she may start playback previously downloaded video chunks. By decoupling the video download and the video playback, client can more or less continue to play downloaded video smoothly until the end. This scheme, a.k.a., progressive download, has a few advantages. (1) Reuse the network infrastructure which is optimized for HTTP traffic. (2) Easy and effortless streaming services by traversing NAT and firewall. (3) Cheaper to move HTTP data to the edge of the network using standard HTTP servers and caches. (4) Video can be encoded with VBR data rate control to improve coding efficiency and save bandwidth.

Although progressive download has become an increasingly popular alternative to multimedia streaming, (e.g., YouTube, ESPN and CNN solely base on progressive download to deliver the videos), it still suffers from several drawbacks. (1) It is not bitrate adaptive. The user will have to buffer a long time before the playback starts or experience frequent freezes and rebuffering in case of insufficient access bandwidth. (2) Potentially, the bandwidth can be wasted in case of user churn. The buffered data is useless if the user had switched to another program. (3) It can hardly support live streaming as the content needs to be preloaded and it is not possible to combat the bandwidth mismatch between content and network.

5.1.2 Adaptive HTTP Video Streaming Architecture

To overcome the disadvantages in progressive download, adaptive HTTP video streaming is proposed recently. Adaptive HTTP streaming is a hybrid of progressive download and streaming. Like the progressive download approach, it relies on clients sending pull signals to the HTTP web server to request the media segments, thus relieving the server load. Unlike the progressive download approach, clients are allowed to switch between different versions of the content encoded with different bitrate with sophisticated segments

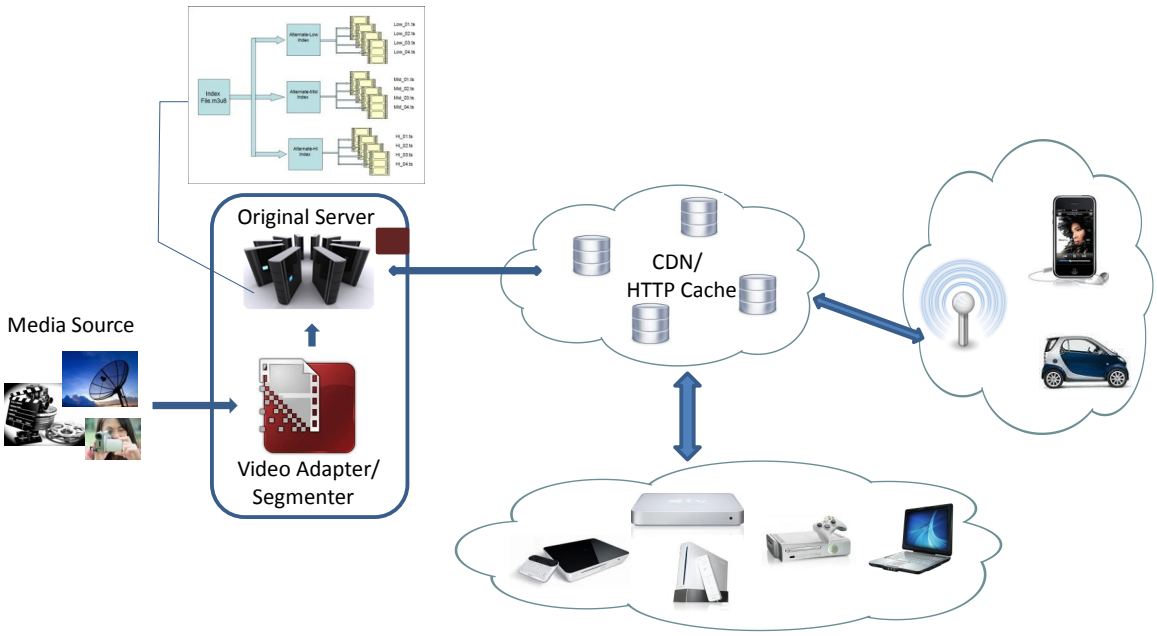


Figure 5.1: Typical System Architecture

generation design. The client player strives to always retrieve the next best segment after examining a variety of parameters related to available network resources, such as available bandwidth and the state of the TCP connections; device capabilities, such as display resolution and available CPU; and current streaming conditions, such as playback buffer size. The goal is to provide the best quality of experience by displaying the highest achievable quality, starting up faster, and reducing skips, freezes, and stutters. Seamless switching is achieved by time-aligning the different versions of the content. This approach addresses the weaknesses of traditional streaming approaches and progressive download, and thus is advocated by several companies and organizations, e.g. Microsoft [72], Apple [73], Adobe [74], MPEG [75], etc.

Figure. 5.1 shows a typical adaptive HTTP streaming architecture. The media contents are processed through video adaptor and/or video segmenter and then fed into original HTTP server. CDNs and/or HTTP caches assist the content dissemination to end-users. There will be multiple versions of the same video encoded with different bitrate

– all are available for clients to request. Usually, there are two separate groups: wireless and wireline. Wireline devices are relatively stable and with continuous connections. Wireless devices, on the other hand, experience variable network conditions and somehow ephemeral. Because of the inherent discrepant nature of these two groups, we might treat them separately. In other words, we need to design *adaptive* client request algorithm with *fast convergence speed* to guarantee video viewing quality.

5.1.3 Previous Work

Many efforts have been devoted to identify issues and evaluate system performance of adaptive HTTP streaming. [76] investigates the HTTP adaptive streaming performance under vehicular mobility. The authors conducted several experiments with a proprietary implementation of HTTP adaptive streaming system driven by HSDPA wireless network bandwidth mobility trace. The authors concluded that HTTP adaptive streaming is effective in responding to the widespread fluctuations that are inherent in vehicular mobility and ultimately achieving an improved viewer experience. [77] considers establishing multiple HTTP-based request-response streams for video streaming. Request-response streams are able to scale with the available bandwidth by increasing the chunk size or the number of concurrent streams. It is shown that by establishing multiple HTTP connections the streaming quality does not deteriorate rapidly as would do single connection. [78] presents an in-depth experimental evaluation of rate adaptation algorithms of two commercial HTTP streaming player (Smooth Streaming, Netflix) and one open source player (OSMF). It is found that Smooth Streaming operates conservatively in presence of bandwidth variations and avoids large transitions in the requested bitrate. It is inefficient in handling large bandwidth spikes. Like the Smooth Streaming, Netflix also maintains two states. However, larger video buffer occupancy at startup and larger variation of fragment request interval are observed. Adobe OSMF player has small playback buffer, therefore, it usually oscillates between the lowest and highest bitrates in presence of bandwidth variations. [79] presents some interesting results of how Akamai Adaptive Video Service performs with changing bandwidth and competes with TCP flow. The video adaptation controller is inefficient in

that it takes about 150 seconds to settle down at correct quality level when increase of bandwidth occurs. Besides, when a sudden drop in the available bandwidth occurs, short interruptions of the video playback can occur due to the a large actuation delay. [80] tries to address the inefficiency issue of the default Akamai rate control algorithm. The authors propose to shift the controller from client to the server and utilize sending queue buffer length to adjust sending rate. This design also inspired us to consider scalable video coding and server-side control. [81] proposes a heuristic algorithm to control video switch-up and switch-down for HTTP adaptive streaming. The decision is made based on the ratio of HTTP segment duration and its fetch time. If the ratio is larger than a threshold determined by representations bitrate, the video will be ramped up; if the ratio is less than a fixed threshold, the video will be ramped down.

5.2 Client Chunk Request Strategy Design

5.2.1 Problem Description

First, we introduce some notations used in the following discussion. Denote L_i the quality level of video chunk i. T_i is the download finish time of chunk i. So $T_i - T_{i-1}$ is the downloading time of chunk i, to be denoted as ΔT_i. R_i is network throughput when downloading chunk i. δ denotes target playback buffer level, measured in second. P_i is the playback deadline of chunk i. $D_i = P_i - \delta$ is the deadline of fetching chunk i so that the target playback buffer level is maintained. The notations are summarized in Table 5.1.

When streaming starts, the client sends HTTP request (*GET signal*) of certain chunk to the server. The server, upon receiving the HTTP request, deliver the requested chunk to the client. Fig. 5.2 shows a sequence of requests from client. Note that, the downloading time for different chunk is varying due to the network dynamics and various chunk size at different quality level. Then, the problem is *given the sequence of chunk requests and the finish time of each chunk up to chunk k, at which quality level should chunk $k+1$ be requested?* If the quality level for chunk $k+1$ is too high, the network can not deliver the

Table 5.1: Notations

Notation	Description
L_i	quality level of video chunk i
T_i	download finish time of chunk i
ΔT_i	downloading time of chunk i
R_i	network throughput when downloading chunk i
D_i	deadline of fetching chunk i
P_i	playback deadline of fetching chunk i
δ	target playback buffer level

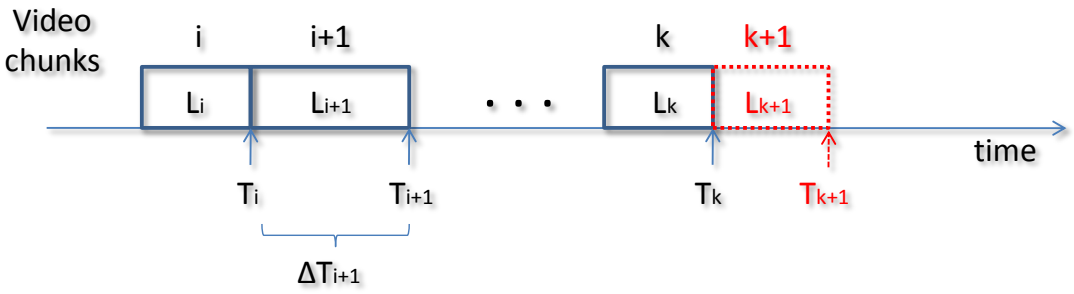


Figure 5.2: Client's Chunk Requesting Process.

chunk on time so that the video may freeze. On the other hand, if the quality level is too low, then the available bandwidth is not fully utilized. Jumping between different quality levels also introduces quality deterioration and thus the level switch should be additionally taken into account at decision making for next chunk. Additionally, the playback buffer should be maintained at a certain fullness level in order to combat the temporally unpredictable network QoS degradation to keep the video playback continuous.

5.2.2 Problem Formulation

For simplicity, we assume each chunk has unit duration. Since the objective of the chunk request strategy is to maximize the overall quality of the delivered video and to minimize the number of quality fluctuations while maintaining certain playback buffer fullness,

we define a cost function G_i for chunk i as follows.

$$G_i = -\alpha L_i + \beta(L_i - L_{i-1})^2 + \gamma I_{T_i > P_i - \delta} \tag{5.1}$$

Here, α, β, γ are three non-negative weighting coefficients determining the importance of the three factors (quality, quality fluctuation and playback buffer level) at decision making. δ is the preset target playback buffer level. $I_{T_i > P_i - \delta}$ is an indicator function taking value of either 0 or 1. If $T_i > P_i - \delta$ is true, i.e., downloading chunk i takes too long so that the target playback buffer level is not maintained, $I_{T_i > P_i - \delta} = 1$. Otherwise, $I_{T_i > P_i - \delta} = 0$. Intuitively, $\gamma I_{T_i > P_i - \delta}$ is the penalty of not maintaining target playback buffer level. T_i is a random variable depending on the dynamic underlying network throughput.

Given an initial state, i.e., the first chunk L_0, and an admissible chunk requesting sequence $\vec{L}$, the overall objective function can be defined as

$$J_{\vec{L}}(L_0) = E_{\vec{T}}(\sum_{i=0}^{N-1} G_i + G_N) \tag{5.2}$$

if we assume the cost for all chunks are additive. The set of decisions that can be taken by the client corresponds to the set of quality levels available at the server. An optimal chunk requesting sequence $\vec{L}^*$ is one that minimizes this cost; that is

$$J_{\vec{L}^*}(L_0) = min J_{\vec{L}}(L_0) \tag{5.3}$$

This problem is a dynamic programming problem [82] that satisfies two principal features: (1) a cost function that is additive over time; and (2) an underlying dynamic system (the network throughput).

5.2.3 Heuristic Algorithm

If the underlying network throughput can be perfectly measured or predicted, then standard algorithm can be applied to find the optimal chunk request sequence $\vec{L}^*$. In reality, however, it is hard to obtain the perfect network information and the complexity to find the optimal request strategy for all chunks is formidable. Therefore, we resort to design

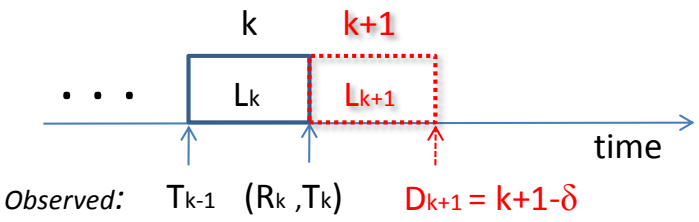


Figure 5.3: Quality Level Decision for Chunk $k+1$.

a heuristic algorithm which determines the quality level for next chunk on-the-fly. Our heuristic algorithm is based on the "principle of optimality" [82] and consists of two stages: the first one is estimation of sustainable quality level and the second one is decision on quality level for next chunk.

For simplicity of presentation, we assume uniform rate distribution of each quality level, i.e., quality level l produces bitrate lR. We also assume γ is much larger than α and β, so that the penalty of playback buffer running low is not affordable. Suppose chunk k has been downloaded, and the decision is going to be made for chunk $k+1$, as shown in Fig. 5.3. We have observed T_{k-1}, T_k and R_k: the finish time for chunk $k-1$ and chunk k and the network throughput during downloading chunk k. The deadline for downloading chunk $k+1$ is $k+1-\delta$ without causing penalty. We predict the network throughput during downloading chunk $k+1$ will be the same as during downloading chunk k. Therefore, we can estimate the sustainable quality level during time T_{k-1} to D_{k+1} as

$$L_k^{\epsilon} = \frac{(D_{k+1} - T_{k-1})R_k}{2 * R} \tag{5.4}$$

The numerator in (5.4) represents the maximum bits can be delivered within this duration. The denominator converts the bits into quality level over the two chunks.

After estimating the sustainable quality level, we are now ready to determine the level for chunk $k+1$ using the rule:

$$L_{k+1} = \begin{cases} L_k + (N-k)\frac{\alpha}{2\beta} & \text{if } L_k < L_k^{\epsilon} - (N-k)\frac{\alpha}{2\beta}, \\ 2L_k^{\epsilon} - L_k & \text{otherwise} \end{cases} \tag{5.5}$$

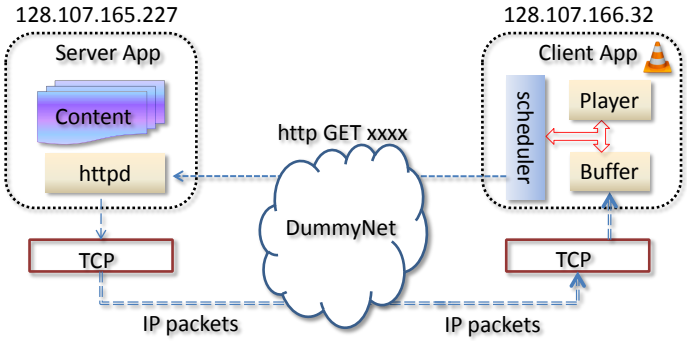


Figure 5.4: Diagram for the Testbed.

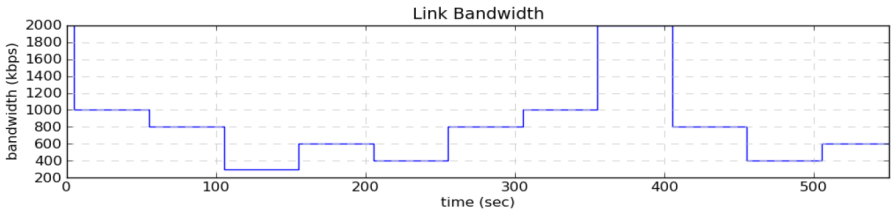
The intuition behind this is that if the quality level for chunk k is too low, then try *rate probing* by increasing quality level for chunk $k+1$; otherwise, keep *rate smoothing* by setting the quality level for chunk $k+1$ to sustainable level. We consider every 6 chunks as a decision unit so that $N = 6$, and k is reset to 0 every 6 chunks.

5.3 Performance Evaluation

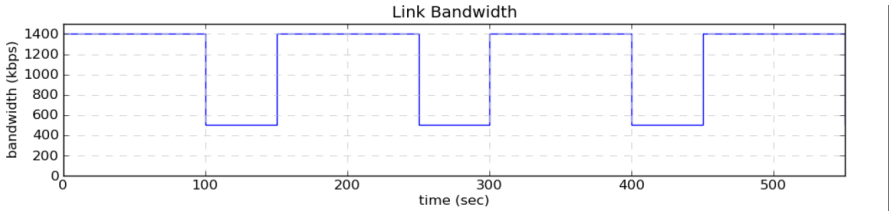
5.3.1 Testbed Implementation

To evaluate the proposed strategy, we implement a testbed based on Apache HTTP server [83] and VLC [84] video player. We implemented a HTTP request module and use the proposed strategy to determine quality level for each chunk. A greedy algorithm which matches requested quality level to estimated network throughput is also implemented for comparison (Method 1). We use DummyNet [85] to generate network dynamics so that the bandwidth varies on the link connecting server and client. Fig. 5.4 shows the diagram for the testbed.

A 10-min video content is generated using ffmpeg [86] and an open-source video segmenter [87]. There are 5 quality levels being 200kbps, 400kbps, 600kbps, 800kbps and 1Mbps. Each segment contains 5-second video. δ is set to be 3 which is equivalent to 15-second video. We evaluated two parameter settings: $\alpha = 1$, $\beta = 6$, which represent switching sensitive video (Method 2) and $\alpha = 6$, $\beta = 1$, representing switching non-



(a) Gradual Bandwidth Changes



(b) Hi-Lo Bandwidth Changes

Figure 5.5: Bandwidth Variation Trace for Experiment.

sensitive video (Method 3). Two bandwidth dynamics scenarios are considered as shown in Fig. 5.5. We consider the average playback quality level and playback quality level variation as the performance metrics.

5.3.2 Experimental Results

Fig. 5.6 shows the performance comparison of the three methods under gradual bandwidth change as in Fig. 5.5(a). The playback quality level over time for Method 1 follows the bandwidth changes, but the resulting quality level variation is relatively high. Method 2, as the parameter setting suggests switching sensitive, has a very low quality level variation. No quality level jumping larger than two is observed. In the mean time, it also provides a higher average quality level than Method 1. Method 3 has the highest average quality level among the three, but it also incurs highest quality level variation due to the aggressive parameter setting emphasizing on average playback quality.

For the Hi-Lo bandwidth change in Fig. 5.5(b), we only show the results for Method 1 and Method 2. Again, Method 2 not only provides a higher average quality level but also a much smaller quality level variation.

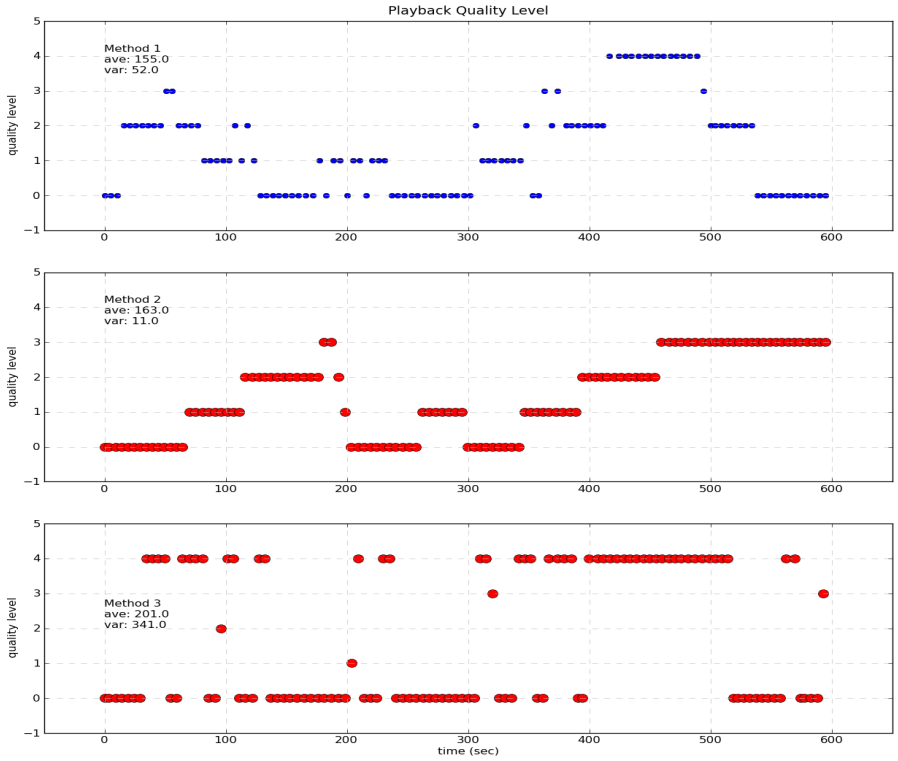


Figure 5.6: Playback quality level over time for the three methods under gradual bandwidth changes.

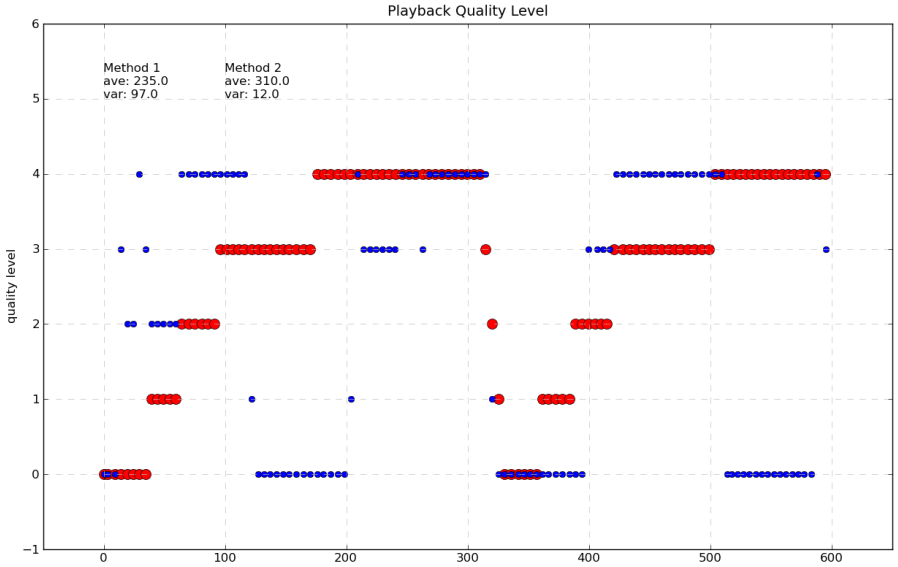


Figure 5.7: Playback quality level over time for Method 1 and Method 2 under Hi-Lo bandwidth changes.

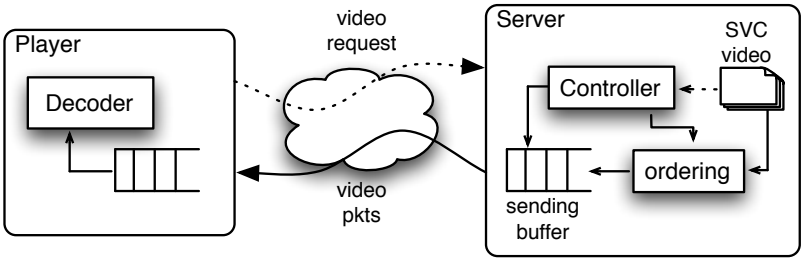


Figure 5.8: System Diagram for SVC Based Streaming.

5.4 SVC as Enhancement

In [80] the authors propose to shift the controller from client to the server and utilize sending queue buffer length to adjust sending rate for Akamai video streaming service. This design also inspired us to consider scalable video coding with preordering and server-assisted control.

5.4.1 System Designs

Fig. 5.8 shows the system diagram for the proposed design. This system does not rely on client to send sequential HTTP requests for video chunks. Instead, the client sends request for the interested video at the beginning only. Once the playback starts, the server will control how to adapt the rate for each video chunk.

The scalable coded video is divided into Group of Pictures (GOP). One or more GOPs can be grouped into one video chunk. Every video chunk will pass through the ordering module which generates pre-ordered packets based on the quality optimized layer ordering strategy (c.f., Chapter 3). The controller will inform the ordering module the video parameters so that the ordering is done properly. After the ordering, all video packets within the chunk will be fed into the sending buffer.

Packets queued in the sending buffer are intended be sent using TCP protocol. However, not all packets within the video chunk can always be sent. Depending on the network condition, the TCP throughput may vary over time, therefore, the controller sets up a sending deadline for each video chunk in order to guarantee continuous playback at the client

side. After sending the video chunk up to its sending deadline, the residual packets will be cleared out and the sending buffer is reloaded with next video chunk. If the TCP throughput is higher than the full video rate, all video packets are sent within the deadline. In this case, the controller immediately starts the next video chunk upon finishing previous video chunk.

We use a simple algorithm to determine the sending deadline for each video chunk. The controller can easily estimate the playback buffer level T_p if the client reports the target playback delay T_d. Suppose the already-sent video length is T_s, the consumed video length is T_c (which is estimated as current time minus streaming start time), then the estimated playback buffer level is $T_p = T_s - T_c + T_d$. We set the sending deadline for each video chunk as σT_p, where σ is a control parameter.

5.4.2 Performance Comparison

We evaluate the proposed design on the `ns-2` [70] simulator. We implemented a TCP-based SVC adaptive streaming system and a HTTP adaptive streaming system for comparison. We use the algorithm proposed in [81] for HTTP adaptive streaming.

Comparison System

The client monitors the ratio of chunk duration and its downloading time, denoted as μ. ϵ is defined as the maximum of $\frac{r_{l+1}-r_l}{r_l} \ \forall l \in [1, 2, ..., L]$. If $\mu > 1 + \epsilon$ and the playback buffer at lease contains 10-second content, then increase the quality level of next chunk. If $\mu < 0.67$, then ramp down the quality level to the highest possible satisfying the observed downloading bandwidth.

Simulation Settings

We encoded a SVC video with GOP size 32, frame rate 30Hz. To compare fairly with the HTTP adaptive streaming, only temporal scalability is enabled so that there is no coding efficiency loss compared with single layered coding. We obtained 6 rate points in total, i.e.,

158.6kbps, 268.8kbps, 471.9kbps, 753.1kbps, 1096.7kbps and 1537.5kbps. The generated video packet trace is used to drive the simulator. In HTTP adaptive streaming case, each video chunk contains 5-second video while for TCP-based SVC adaptive streaming, each chunk has 4 GOPs. σ is set to 0.7.

We use a dumbbell topology to simulate the network where a single bottleneck link is shared by various flows. We simulate two scenarios. The first scenario (TCP background traffic) has 20 background FTP flows each starts every 8 seconds. All FTP flows stop at 200. The bottleneck link has 20Mbps capacity and the round-trip delay is 20ms. The second scenario (UDP background traffic) has 2 background UDP flows with 800kbps each. One flow starts at 50; the other starts at 130; both end at 250. The bottleneck link has 2Mbps and round-trip delay is 20ms. In each scenario, we consider two cases: the first case is Video-on-Demand (VoD) service where all video chunks are available to client at any time; the second case is Live streaming service (Live) where video chunks are made available to client in a moving window fashion. We set the window to be 20 second ahead of streaming start time, (e.g. at time 0, client can get at most 20 second, and so forth). In both scenarios, the playback delay is 10 second, and streaming starts at 10 (so the player starts playback at 20). For clarity of presentation, we shift all results (playback bitrate over time) along x-axle to the left by 20.

TCP background traffic

Fig. 5.9 shows the playback rate comparison of TCP-based SVC streaming (noted as TCPsvc) and HTTP adaptive streaming (noted as HTTP) with TCP background traffic under VoD case. We can see that TCPsvc adapts to available bandwidth much quicker than HTTP. When the traffic load from background FTP flows increases, as can be seen around 75, TCPsvc adapts the rate to a lower level and maintains a relatively stable playback rate thereafter. On the other hand, playback of HTTP stays at high rate point even if the network is overloaded. This is due to its pre-buffered high quality chunks. Starts from 100, it backs off to lower level. We can see that occasionally the backoff is too conservative for HTTP causing a very low playback quality. On average, TCPsvc achieves 1152kbps which is

better than HTTP's 972kbps.

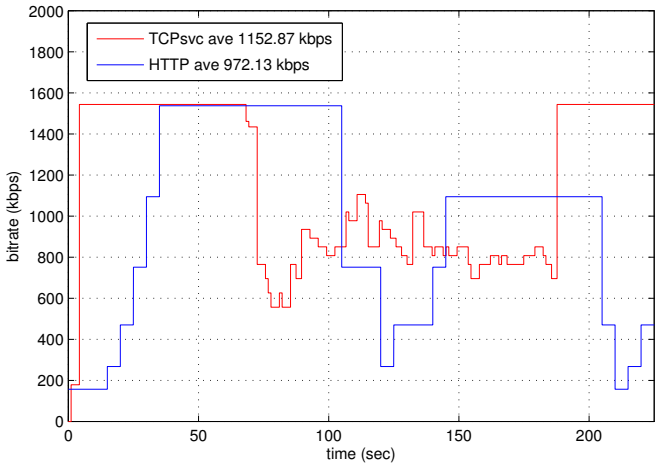


Figure 5.9: Playback Rate Comparison of SVC streaming and HTTP adaptive streaming with TCP background traffic: VOD case.

In case of Live streaming (shown in Fig. 5.10), the performance for HTTP adaptive streaming is severely degraded, resulting average playback rate 748kbps. Interestingly, the playback rate starts to drop around 40. The reason is that as the link is not over loaded, HTTP downloads the video chunk faster than its duration. As the video content at the server is only 20-second ahead, HTTP client stops requesting due to the lack of content at the server until it sees new content available. Steaming stalls but playback continues for HTTP client. During this "waiting" period, other FTP flows see no video traffic and the TCP rate control increases FTP sending rate. This causes HTTP back off to lower level at next available chunk as the throughput for next chunk is dropping. It takes about 30 second for HTTP to recover the full transmission rate (the time different between the two spikes). After that, HTTP backs off again as more FTP flows join the system. On the other hand, the performance of TCPsvc is slightly affected, achieving average playback rate 1024kbps. This result indicates that TCP-based SVC adaptive streaming is a good candidate for live video streaming.

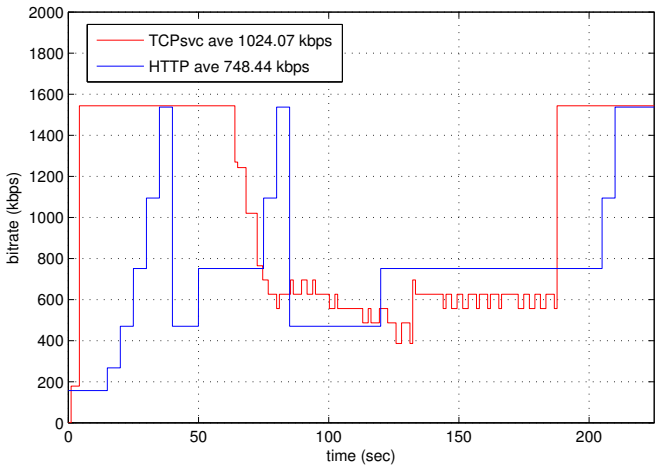


Figure 5.10: Playback Rate Comparison of SVC streaming and HTTP adaptive streaming with TCP background traffic: Live case.

UDP background traffic

Fig. 5.11 and Fig. 5.12 show the playback rate for the two schemes under VoD case and Live case respectively. We can see that TCPsvc performs the same under both cases indicating that the available bandwidth is efficiently utilized. On the other hand, as HTTP adaptive streaming tries to ramp up the quality level gradually, the available bandwidth is higher than the request video rate, so the video playback buffer is built up. This allows HTTP to maintain at a high playback rate even though the available bandwidth is dropping. The difference between VoD and Live case is the buffered video length. The average playback rate for VoD and Live cases are 785kbps and 733kbps respectively. TCPsvc achieves 1002kbps under both cases which is much higher than HTTP scheme.

5.5 Summary

In this chapter, we deal with adaptive HTTP video streaming, say the client chunk request strategy design. A good playback quality results from high average playback quality level, low quality switching and no playback freeze. To this end, we propose a cost

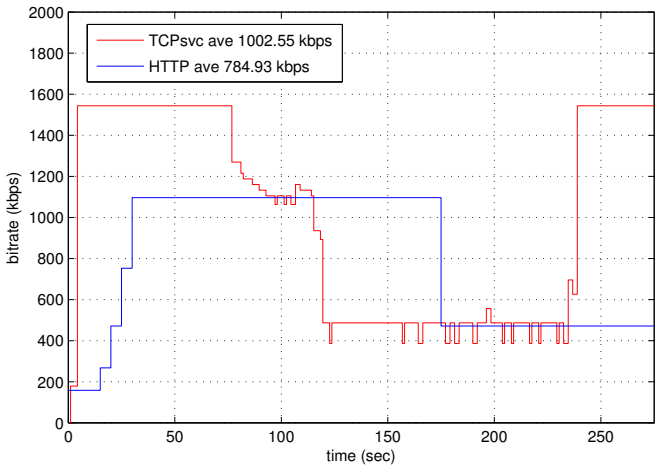


Figure 5.11: Playback Rate Comparison of SVC streaming and HTTP adaptive streaming with UDP background traffic: VOD case.

function associate with each video chunk capturing the impacts from three factors: its quality level, its contribution to quality variation and its contribution to playback buffer level. Assuming the cost function is additive, we can convert the client chunk request strategy into a Dynamic Programming problem. Due to the difficulty of estimating the underlying network dynamics and high-complexity of directly solving the problem, we design a heuristic algorithm to solve it. We evaluate the algorithm with a testbed developed. To further improve the streaming performance, we consider scalable video coding combined with server-assisted TCP transmission. A simple controller for the server is proposed by setting up the sending deadline for each per-ordered video chunk. The simulation results demonstrate advantages of such design comparing with existing HTTP adaptive streaming design.

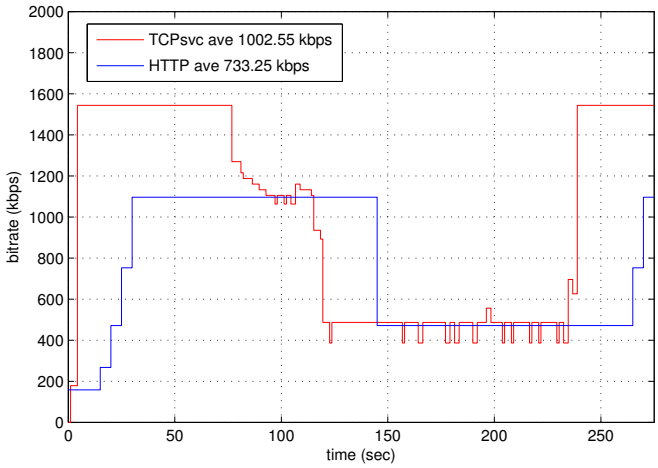


Figure 5.12: Playback Rate Comparison of SVC streaming and HTTP adaptive streaming with UDP background traffic: Live case.

Chapter 6

Conclusion and Future Work

6.1 Summary of Major Contributions

In this thesis, we consider application layer system designs for video streaming. Specifically, we apply advanced video coding scheme, i.e., scalable video coding, to provide low-complexity in-network rate adaptation. Three particular video streaming architectures are investigated: P2P-based video streaming, proxy-assisted wireless video streaming and HTTP based adaptive streaming. Within each architecture, we propose novel utility functions and algorithms for maximizing the utility to help the system design and improve the performance.

First, we model the mesh-based layered P2P streaming system into a optimization problem with the objective being maximizing the aggregate video quality of all peers constrained by the network capacity. To incorporate contribution-awareness, we propose *taxation* as a peer-incentive mechanism and augment the utility optimization problem to make the solution incentive-compatible. The tax rate controls the tradeoff between the system efficiency, fairness and incentives. The proposed taxation-based scheme guarantees the streaming rate of a node is at least proportional to its uploading contribution. The left-over uploading bandwidth (taxed portion) from each node is used to help "slow" peers, therefore improves the social welfare due to the concavity of utility function (video quality). Using the taxation-based utility optimization problem, we solve for the streaming rate for

each node with various network settings. From the results, we obtain several guidelines for taxation-based P2P layered streaming design including which topology is preferred and how to use the uplink bandwidth efficiently. Then, we propose practical system designs including layer subscription, chunk scheduling, and mesh topology adaptation. For layer subscription, Additive Increase Additive Decrease (AIAD) and exponential backoff are used to help peers probing the network resources. In the meantime, peers run a parallel subscription decrease process to ensure all subscribed layers can be received. Chunk scheduling consists of two parts: requesting and serving. The chunks are requested in the order of their importance. The probability of choosing a specific peer to serve the missing chunk is proportional to its serving rate to that peer. Peers use a replacement index (weighted sum of chunks) to determine which neighbor to be replaced by other peer. Extensive trace-driven simulations demonstrate that the proposed designs can effectively drive layered P2P streaming systems to the desired operating point given a tax ratio. The taxation-based P2P streaming designs allows us to freely adjust the balance between the social welfare and individual peer welfare.

Second, to better incorporate SVC into streaming system design, we study its rate and subjective quality tradeoff. By leveraging the parametric models from prior work [13], we give a criteria for choosing the optimal frame rate and quantization stepsize under a rate constraint, which is used to obtain the optimal subjective quality. We further model the optimal rate-quality tradeoff with a exponential function where the model parameter is content dependent. The model is highly accurate for all test sequences. We also show the concavity of this model so that it can be easily incorporated into network utility maximization framework. To facilitate SVC rate adaptation, we propose a quality optimized layer ordering strategy. The resulting ordered stream has the property that each additional layer offers maximum quality improvement for the rate increment. At the same time, layer decoding dependency is satisfied. The layer ordering can be easily implemented in the network (at proxy/CDN node) or at the original server. With such a ordered SVC stream, the proxy/server can simply keep sending additional layers, until the rate target is reached.

Third, with the rate-quality model and layer ordering strategy developed, we optimize

streaming of multiple scalable video streams through a common wireless access node. To combat the highly dynamic wireless link condition, we envision a proxy node co-located with the access node. Such a design requires no additional modifications at either the video servers or the mobile clients as video adaptation is performed at the proxy and is agnostic to both ends. The proxy is in charge of tracking the time-varying status of the wireless access links connecting to the access node, while dynamically adapting the traversing scalable video streams. It iteratively allocates rates of different video streams to maximize a weighted sum of video qualities associated with different streams, based on the periodically observed link throughputs and the sending buffer status at the access node. After determining the sending rate for each stream, the proxy sends preordered video packets in each stream sequentially until the rate budget is used up. Simulation studies show that our scheme consistently outperforms TFRC in terms of agility to track link qualities and overall subjective quality of all streams. In addition, the proposed scheme supports differential services for different streams, and competes fairly with TCP flows.

Forth, as the adaptive HTTP streaming gains increasing interest as a video delivery solution, we develop novel client chunk request strategy to improve its video viewing quality. Each chunk request will impact the average playback quality, the variation of playback quality and the playback buffer level. We propose a cost function for each chunk by jointly considering the three factors. By assuming the additive property of cost function over all chunks, we convert the problem of finding the optimal chunk request sequence minimizing the total cost, into a dynamic programming problem. Due to the difficulty of predicting the underlying network dynamics and high complexity involved in solving the problem exactly, we propose a heuristic algorithm to solve it. The algorithm will either enter rate probing state or rate smoothing state depending on the estimated sustainable throughput for the next chunk. To evaluate the performance of the proposed algorithm, we develop a testbed with Apache HTTP server [83], DummyNet [85] and VLC [84] video player. We implement a HTTP request scheduler module for VLC based on the proposed algorithm. The experimental results show that the proposed chunk requesting algorithm has a smoother playback quality than greedy algorithm. To further improve the performance, we propose a TCP-

based SVC adaptive streaming scheme. The server transmits as many video packets as possible within each chunk in their order of importance in the allowed time for this chunk; the allowed time is set to be a fraction of the buffered video length at the client side (the server can easily estimate it), so that the buffer underflow can be avoided. As this scheme allows partial transmission of the video chunk, it is more flexible than current HTTP adaptive streaming which offers no such freedom. Simulation results show that TCP-based SVC adaptive streaming offers a higher average playback rate and faster convergence speed than adaptive HTTP streaming.

6.2 Future Work

6.2.1 Extension of Proposed Rate-Quality by Considering the Spatial Scalability

Our proposed rate-quality model has considered the joint impact of temporal and amplitude scalability provided by the SVC. The additional dimension provided by spatial scalability is not addressed. Practically, it is desirable to investigate the quality and rate variation introduced by spatial scalability, and to incorporate the spatial resolution impact into current metrics in addition to the temporal and amplitude resolutions. The quality optimized layer ordering needs to be revised also. Our preliminary results show that there exists a simple backward search algorithm (from highest layer to lowest layer) that has acceptable results [88].

6.2.2 Implementation of Server-Assisted Adaptive Streaming

The simulation results in Chapter 5 confirm the advantages of server-assisted adaptive streaming of layered video over the client-based adaptive streaming of single layer video. The next step would be implementing the real system. There are many open-source platforms for video streaming development, e.g., Darwin Streaming Server [89], OSMF [74]. By reusing the code base from these platforms, the developing process can be simplified.

It is also desirable to have real-time scalable video encoder/decoder. Fortunately, we can also start with public domain solutions, e.g., opensvc [90].

6.2.3 Multi-Party Video Conference

Another active research area in application layer video optimization is multi-party video conference. In our preliminary studies, we measured the two most popular multi-party video conference solutions: Skype [91] and Google+ [92]. From the measurement results, we understand the system architecture of both systems and we have identified several issues therein.

As both systems rely on dedicated servers to assist video transmission, the operation cost is inevitably high. To relieve this problem, we will consider P2P architecture as a candidate solution. Based on the rate-quality model developed, we can formulate the system in a utility maximization framework to determine the receiving video rate for each participant. Then, we will design sophisticated data delivery route to fulfill the video rate. Scalable video coding can guarantee graceful degradation in face of network condition changes.

We will further investigate the data transmission strategy to improve the overall video throughput. From our preliminary simulation results, we noticed that UDP with prioritized retransmission is superior to TCP or UDP with FEC schemes. This inspires us to consider UDP with prioritized retransmission as a candidate for multi-party video conference. We will evaluate the true performance of this scheme with real network experiments.

Bibliography

[1] Cisco, "Visual networking index: global mobile data traffic forecast update, 2010-2015," Febrary 2011.

[2] "PPLive, http://www.pplive.com/." [Online]. Available: http://www.pplive.com/

[3] "PPStream, http://www.ppstream.com/." [Online]. Available: http://www.ppstream.com/

[4] "UUsee, http://www.uusee.com/." [Online]. Available: http://www.uusee.com/

[5] "VideoSpeedy, http://www.videospeedy.com/." [Online]. Available: http://www.videospeedy.com/

[6] "PPacc, http://www.ppacc.com/." [Online]. Available: http://www.ppacc.com/

[7] J. Liu, S. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, Jan. 2008.

[8] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. Circuit and System for Video Technology*, vol. 17, pp. 1103–1120, Sep. 2007.

[9] M. Wien, H. Schwarz, and T. Oelbaum, "Performance Analysis of SVC," *IEEE Trans. Circuit and System for Video Technology*, vol. 17, pp. 1194–1203, Sep. 2007.

[10] *ITU-T recommendation H.264-ISO/IEC 14496-10(AVC), advanced video coding for generic audiovisual services, amendment 3: scalable video coding*, ITU-T and ISO/IEC JTC 1, 2005.

[11] J. R. Ohm, "Advances in Scalable Video Voding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 42–56, January 2005.

[12] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, January 2007.

[13] Y. Wang, Z. Ma, and Y.-F. Ou, "Modeling Rate and Perceptual Quality of Scalable Video as Functions of Quantization and Frame Rate and Its Application in Scalable Video Adaptation," in *Proceedings of Packet Video*, May 2009, pp. 1–9.

[14] S. Floyd, M. Handley, J. Pahdye, and J. Widmer, "Tcp Friendly Rate Control (TFRC): protocol specification," RFC 5348 (Proposed Standard), September 2008.

[15] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 473–489, Augest 1999.

[16] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, March 2006.

[17] "Youtube." [Online]. Available: http://www.youtube.com/

[18] Y. hua Chu, J. Chuang, and H. Zhang, "A Case for Taxation in Peer-to-Peer Streaming Broadcast," in *ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004.

[19] Y.-W. Sung, M. Bishop, and S. Rao, "Enabling Contribution Awareness in an Overlay Broadcasting System," in *SIGCOMM*, 2006.

[20] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Substream Trading: Towards an Open P2P Live Streaming System," in *ICNP*, 2008.

[21] B. Cohen, "Incentives Build Robustness in Bittorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer systems*, 2003.

[22] Y. Wang, A. Reibman, and S. Lin, "Multiple Description Coding for Video Delivery," *Proceedings of the IEEE*, vol. 93, Jan. 2005.

[23] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, "LION: Layered Overlay Multicast with Network Coding," *IEEE Trans. Multimedia*, vol. 8, pp. 1021–1032, Oct. 2006.

[24] X. Xiao, Y. Shi, Y. Gao, and Q. Zhang, "LayerP2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks," in *IEEE INFOCOM*, 2009.

[25] P. K. Hoong and H. Matsuo, "Palms: A Reliable and Incentive-based P2P Live Media Streaming System," *Lecture Notes in Electrical Engineering*, pp. 51–66, 2008.

[26] A. Habib and J. Chuang, "Service Differentiated Peer Selection: An Incentive Mechanism for Peer-to-Peer Media Streaming," *IEEE Trans. on Multimedia*, vol. 8, pp. 610–621, 2006.

[27] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using Layered Video to Provide Incentives in P2P Live Streaming," in *Sigcomm 2007 workshop on peer-to-peer streaming and IP-TV*, 2007.

[28] X. Zhang, J. Liu, B. Li, and P. Yum, "Coolstreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming," in *IEEE INFOCOM*, 2005.

[29] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-tree: A Comparative Study of Live P2P Streaming Approaches," in *Proc. IEEE INFOCOM*, May 2007.

[30] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility Maximization in Peer-to-peer Systems," in *Proceeding of ACM SIGMETRICS*, June 2008.

[31] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. IEEE INFOCOM*, May 2007.

[32] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming," in *Proc. IEEE INFOCOM*, May 2007.

[33] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. on Information Theory*, vol. 46, pp. 1204–1216, 2000.

[34] D. M. Chiu, R. W.-H. Yeung, J. Huang, and B. Fan, "Can Network Coding Help in P2P Networks?" in *Second Workshop of Network Coding*, Boston, USA, April 2006.

[35] F. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices Proportional Fairness and Stability," *Journal of the Operational Research Society*, 1998.

[36] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice Hall Inc., 1992.

[37] H. Hu, Y. Guo, and Y. Liu, "Peer-to-Peer Streaming of Layered Video: Efficiency, Fairness and Incentive," in *Technical Report, Polytechnic Institute of NYU*, 2009. [Online]. Available: http://vision.poly.edu/~hao/papers/2009_TR_P2Ptax.pdf

[38] "Ampl, http://www.ampl.com/." [Online]. Available: http://www.ampl.com/

[39] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: An Efficient Mechanism for Content Distribution in a P2P Network," in *Sigcomm Asia Workshop*, 2005.

[40] S. woo Cho and A. Goel, "Pricing for Fairness: Distributed Resource Allocation for Multiple Objectives," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006.

[41] X. Hei, C. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A Measurement Study of a Large-scale P2P IPTV system," in *Workshop on Internet Protocol TV (IPTV) services over World Wide Web*, Edinburgh, Scotland, May 2006.

[42] H. Schwarz, D. Marpe, and T. Wiegand, "Hierarchical B pictures," *Joint Video Team Doc. JVT-P014*, July 2005.

[43] H. Schwarz, D. Marpe, and T. Wiegand, "Joint scalable video model (jsvm) 2," *Joint Video Team, Doc. JVT-O202*, April 2005.

[44] Y.-F. Ou, Z. Ma, T. Liu, and Y. Wang, "Perceptual Quality Assessment of Video Considering both Frame Rate and Quantization Artifacts," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 286–298, March 2011.

[45] Z. Liu, Y. Shen, K. Ross, S. Panwar, and Y. Wang, "LayerP2P: Using Layered Video Chunks in P2P Live Streaming," *IEEE Trans. on Multimedia*, pp. 1340–1352, Nov., 2009.

[46] M. Wien, H. Schwarz, and T. Oelbaum, "Performance Analysis of SVC," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1194–1203, September 2007.

[47] H. Chen and J. E. Thropp, "Review of Low Frame Rate Effects on Human Performance," *IEEE trans. on systems, man and cybernetics–part A: systems and humans*, vol. 37, no. 6, pp. 1063–1076, November 2007.

[48] Y. Wang, S.-F. Chang, and A. C. Loui, "Subjective Preference of Spatio-Temporal Rate in Video Adaptation using Multi-Dimensional Scalable Coding," in *Proceedings of IEEE International Conference on Multimedia and Expo*, June 2004, pp. 1917–1722.

[49] K.-C. Yang, C. C. Guest, K. El-Maleh, and P. K. Das, "Perceptual Temporal Quality Metric for Compressed Video," *IEEE trans. on Multimedia*, vol. 9, no. 7, pp. 1526–1535, November 2007.

[50] Q. Huynh-Thu and M. Ghanbari, "Temporal Aspect of Perceived Quality in Mobile Video Broadcasting," *IEEE trans. on Broadcasting*, vol. 54, no. 3, pp. 641–651, September 2008.

[51] S. H. Jin, C. S. Kim, D. J. Seo, and Y. M. Ro, "Quality Measurement Modeling on Scalable Video Applications," in *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, October 2007, pp. 131–134.

[52] K. Yamagishi and T. Hayashi, "QRP08-1: Opinion Model for Estimating Video Quality of Videophone Services," in *Proceedings of IEEE Clobal Communications Conference*, November 2006, pp. 1–5.

[53] Z. Ma, M. Xu, Y.-F. Ou, and Y. Wang, "Modeling Rate and Perceptual Quality of Video as Functions of Quantization and Frame Rate and Its Applications," *submitted to IEEE Trans. on Circuits and Systems for Video Technology*, 2011.

[54] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhauser, 2003.

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

[56] J. Cabrera and A. Ortega, "Stochastic Rate Control of Video Coders for Wireless Channels," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 496–510, June 2002.

[57] L. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips, "Optimized Video Streaming Over 802.11 by Cross-Layer Signaling," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 115–121, January 2006.

[58] P. van Beek and M. U. Demircin, "Delay-Constrained Rate Adaptation for Robust Video Transmission Over Home Networks," in *Proceedings of IEEE International Conference on Image Processing*, September 2005, pp. 173–176.

[59] T. Stockhammer, M. Walter, and G. Liebl, "Optimized H.264-Based Bitstream Switching for Wireless Video Streaming," in *Proceedings of IEEE International Conference on Multimedia and Expo*, July 2005, pp. 1396–1399.

[60] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of H.264/AVC," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[61] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile Video Transmission Using Scalable Video Coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1204–1217, September 2007.

[62] M. Chen and A. Zakhor, "Rate Control for Streaming Video Over Wireless," in *Proceedings of IEEE International Conference on Computer Communications*, March 2004, pp. 1181–1190.

[63] H. Luo, D. Wu, S. Ci, H. Sharif, and H. Tang, "TFRC-based Rate Control for Real-Time Video Streaming over Wireless Multi-Hop Mesh Networks," in *Proceedings of IEEE International Conference on Communications*, June 2009, pp. 1–5.

[64] G. Yang, T. Sun, M. Gerla, M. Y. Sanadidi, and L.-J. Chen, "Smooth and Efficient Real-Time Video Transport in the Presence of Wireless Errors," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 2, no. 2, pp. 109–126, May 2006.

[65] J. Chakareski and P. Frossard, "Rate-Distortion Optimized Distributed Packet Scheduling of Multiple Video Streams Over Shared Communication Resources," *IEEE trans. on Multimedia*, vol. 8, no. 0, pp. 207–218, April 2006.

[66] L. Zhou, X. Wang, W. Tu, G. Muntean, and B. Geller, "Distributed Scheduling Scheme for Video Streaming Over Multi-Channel Multi-Radio Multi-Hop Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 409–419, April 2010.

[67] X. Zhu, R. Pan, N. Dukkipati, V. Subramanian, and F. Bonomi, "Layered Internet Video Engineering (LIVE): Network-Assisted Bandwidth Sharing and Transient Loss Protection for Scalable Video Streaming," in *Proceedings of IEEE International Conference on Computer Communications Mini-Conference*, March 2010.

[68] X. Zhu and B. Girod, "Distributed Media-Aware Rate Allocation for Wireless Video Streaming," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1462–1474, January 2010.

[69] S. Ulukus and R. D. Yates, "Stochastic Power Control for Cellular Radio Systems," *IEEE Trans. on Communications*, vol. 46, no. 6, pp. 784–798, June 1998.

[70] Ns-2 network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/

[71] Cisco. QoS on wireless LAN controllers and lightweight APs configuration example. [Online]. Available: http://www.cisco.com/en/US/tech/tk722/tk809/technologies_configuration_example09186a00807e9717.shtml

[72] Alex
Zambelli, "IIS Smooth Streaming Technical Overview," http://www.microsoft.com/downloads/en/details.aspx?FamilyID=03d22583-3ed6-44da-8464-b1b4b5ca7520.

[73] Apple Inc., "HTTP Live Streaming Overview," http://developer.apple.com/library/ios/#documentation/networkinginternet/conceptual/streamingmediaguide.

[74] Adobe Inc., "Open source media framework," http://http://osmf.org/.

[75] MPEG, "Iso/iec dis 23001-6: MPEG Systems Technologies – part 6: Dynamic Adaptive Streaming over HTTP (DASH)," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623.

[76] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan, "Empirical Evaluation of HTTP Adaptive Streaming under Vehicular Mobility," in *Proceedings of IFIP Networking*, May 2011, pp. 92–105.

[77] R. Kuschnig, I. Kofler, and H. Hellwagner, "Evaluation of HTTP-based Request-Response Streams for Internet Video Streaming," in *Proceedings of ACM MMsys*, Feb. 2011, pp. 245–256.

[78] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *Proceedings of ACM MMsys*, Feb. 2011, pp. 157–168.

[79] L. D. Cicco and S. Mascolo, "An Experimental Investigation of the Akamai Adaptive Video Streaming," in *Proceedings of USAB*, 2010, pp. 447–464.

[80] L. D. Cicco, S. Mascolo, and V. Palmisano, "Feedback Control for Adaptive Live Video Streaming," in *Proceedings of ACM MMsys*, Feb. 2011, pp. 145–156.

[81] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptiv HTTP streaming," in *Proceedings of ACM MMsys*, Feb. 2011, pp. 169–174.

[82] D. P. Bertsekas, *Dynamic Programming and Optimal Control Volume I*. Athena Scientific, 2005.

[83] A. S. Foundation, "Apache HTTP server project," http://httpd.apache.org/.

[84] VideoLAN., "VLC media player," http://www.videolan.org/vlc/.

[85] L. Rizzo, "DummyNet," http://info.iet.unipi.it/~luigi/dummynet/.

[86] FFMPEG group., "FFMPEG project," http://www.ffmpeg.org/.

[87] C. Douglas, "HTTP live streaming segmenter," http://svn.assembla.com/svn/legend/segmenter/.

[88] H. Hu, Z. Ma, and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation," http://vision.poly.edu/~hao/papers/2012_TR_RQSTAR.pdf, Polytechnic Institute of New York University, Tech. Rep., Jan. 2012.

[89] Apple Inc., "Darwin Streaming Server," http://dss.macosforge.org/.

[90] OpenSVC, http://www.opensvc.com/.

[91] Skype Inc., "Skype features," http://www.skype.com.

[92] Google Inc., "Google+," http://plus.google.com.