Offering High-Definition Peer-Assisted Video on-Demand Systems: Modeling, Optimization and Evaluation

by

Le Chang
B. Eng., Central South University, 2004
M. Eng., Central South University, 2007

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Offering High-Definition Peer-Assisted Video on-Demand Systems: Modeling, Optimization and Evaluation

by

Le Chang
B. Eng., Central South University, 2004
M. Eng., Central South University, 2007

Supervisory Committee

Dr. Jianping Pan, Supervisor
(Department of Computer Science)

Dr. Sudhakar Ganti, Departmental Member
(Department of Computer Science)

Dr. Wusheng Lu, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Jianping Pan, Supervisor
(Department of Computer Science)

Dr. Sudhakar Ganti, Departmental Member
(Department of Computer Science)

Dr. Wusheng Lu, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The past decade has witnessed the fast development of peer-assisted video on-demand (PA-VoD) systems, which have attracted millions of online users. The efforts on improving the quality of video programs have never ceased since the beginning, and nowadays offering high-definition (HD) channels has become a common practice. However, compared with standard-definition (SD) channels, HD channels have to sustain a higher streaming rate to peers, which is a challenging task. In real systems, HD channels often suffer from poor streaming quality, or impose a heavy burden on the servers.

This thesis conducts an in-depth study on peer cache and upload bandwidth management at the same time for multi-channel PA-VoD systems, where HD and SD channels coexist with different bandwidth and cache requirements. The objective is to minimize the server bandwidth consumption, and thus the maintenance cost of VoD service providers. The solution is cross-channel allocation (or view-upload decoupling), i.e., making SD channels help HD viewers with the surplus peer-contributed resources. The management of these resources includes bandwidth allocation and caching strategies.

We first propose a generic modeling framework to capture the essential characteristics of PA-VoD systems: the demand and supply of bandwidth from peers. Our

modeling framework can be customized or extended to model a variety of caching strategies, including FIFO, passive caching, and active caching with different user behaviors. We then apply the modeling framework to two representative scenarios: stationary scenarios, where the channels have fixed popularity; and non-stationary scenarios, in which a new movie is released, and peers enter the channel in a flash-crowd manner. We prove using our models that passive caching is efficient for stationary user behaviors, and derive the optimal caching solutions when the channels in the system demonstrate different popularity evolutions, i.e., with non-stationary behaviors.

With the insights gained from our modeling work, we design effective centralized heuristic algorithms and practical distributed strategies for peer cache replacement and upload bandwidth allocation, with a near-optimal utilization of these resources. We propose centralized and distributed cross-channel allocation, and also extend the substreaming technique from live streaming to VoD systems, where it demonstrates its extreme feasibility. Our extensive simulation results verify the efficacy of these heuristic and practical strategies.

Keywords Peer-assisted video on-demand (PA-VoD) systems, resource balancing, bandwidth allocation, caching strategies, modeling, optimization

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor, Dr. Jianping Pan, for guiding me and supporting me throughout these four and half years. You have set an example of excellence, not only as a researcher, supervisor, and instructor, but also a friend who is always ready to offer help in my research career and daily life.

my committee members, Dr. Sudhakar Ganti and Dr. Wusheng Lu, for their valuable suggestions on my research topic, and their efforts on reviewing the thesis. Without their help, I would have to spend a longer time finishing my PhD study.

the professors from the Faculty of Engineering, Dr. Lin Cai and Dr. Kui Wu, for their help and instructions on my course work, and their enlightening guidance on my research projects.

my mother, father, and my wife, for their endless support, no matter in good times or bad, on silly days or sad. Without your love, this thesis would not have been made possible.

my friends and group members at University of Victoria, for accompanying me in my daily life in Victoria. The happiness you have brought to me has made these four years a truly memorable experience.

China Scholarship Council, for funding me with a scholarship, which helped provide a good research environment in these four years.

DEDICATION

Dedicated to my family, and my motherland, China.

Chapter 1

Introduction

1.1 Problem and Motivation

In the past decade, online video on-demand (VoD) systems have become increasingly popular. Different from live video programs, VoD offers users the freedom of watching their favorite video programs at their own convenience, i.e., watching any channel at any time. These videos attract a huge number of users. For instance, YouTube [4], one of the most famous VoD systems, has reported 4 billion daily views as of January 2012 [5].

However, YouTube follows a strategy similar to the *Server/Client* (C/S) model, in which all the users fetch video content from their servers (or distributed servers). Such a model requires great processing power and bandwidth capacity of the server(s). Other than the maintenance cost of the YouTube servers, ISPs also charge YouTube for the Internet usage based on the consumed volume of bandwidth. As estimated, YouTube has consumed 30 million megabits-per-second on average as of 2009, resulting in 300 million US dollars for the year on such server bandwidth consumption [6].

In contrast, in P2P or peer-assisted systems, users act as "peers", and "peers" help each other. They contribute their processing and bandwidth capacity to the system, which helps relieve the burden and reduce the consumed volume of bandwidth on the servers. The P2P technology has been applied to P2P file sharing applications, e.g., BitTorrent [7], Voice-over-IP (VoIP) systems, e.g., Skype [8], and VoD systems, the main topic of this thesis. The potential of such user contributed bandwidth in VoD systems has been evaluated, and the conclusion is quite encouraging: 97% of the server bandwidth consumption can be potentially saved by using the P2P technology [9].

This boosts the deployment of many commercialized Peer-Assisted VoD (PA-VoD) systems, such as PPLive [10], PPStream [11], UUSee [12], etc.

Nevertheless, measurement results have shown that such a conclusion is over-optimistic due to the resource imbalance problem [1, 2]. The bandwidth demand and supply from peers vary dramatically between video channels, making some of the channels to be well-provisioned and others poorly-provisioned, especially when high-definition (HD) videos are offered. HD videos have higher streaming rates and require much more user-contributed bandwidth and cache space, while the supply from peers in these channels hardly meets the demand. As a result, without the extra support from powerful server(s), or other channels, these HD channels are not able to offer fluent viewing experience to users.

To solve this problem, a natural way is to allow resource allocation across channels, i.e., making bandwidth-surplus channels, usually standard-definition (SD) channels, to help HD channels. Such help involves two kinds of resources: peer bandwidth and cached content. Therefore, in this thesis, we focus on two sets of strategies: *bandwidth allocation*, which allocates the upload bandwidth of peers across different channels; and *caching*, which manages the local cache of each peer, with a variety of user behaviors. The objective is to reduce the server bandwidth consumption as much as possible. As such bandwidth consumption is charged by ISPs, less consumption simply means lower cost, and more profit for VoD service providers.

1.2 Contributions

In this thesis, we conduct extensive studies on designing, modeling, optimizing and evaluating the bandwidth allocation and caching strategies for PA-VoD systems with HD channels. The contributions of this thesis are summarized as follows.

- We propose a modeling framework to capture the essential characteristics of PA-VoD systems offering SD and HD channels, e.g., the bandwidth demand and supply from peers. Our framework can be extended to model a variety of caching strategies, including FIFO, passive caching, and active caching, under different scenarios, such as stationary and non-stationary scenarios.

- Under stationary scenarios, i.e., the popularity of movies never changes, we extend our modeling framework to derive the statistical performance bounds, i.e., the server bandwidth consumption, with FIFO and passive caching, and

prove that passive caching is sufficiently efficient for such stationary user behaviors. We also formulate bandwidth allocation as a linear programming problem to calculate the tight lower bound at any time instant, with global information available and system-wide coordination possible (e.g., through a tracker). Moreover, we design heuristic algorithms for peer upload bandwidth allocation to fit the caching strategies. Their performance are compared with the performance bounds through extensive simulation, which shows the efficacy of the proposed algorithms.

- For non-stationary scenarios, we consider the case that a new HD movie is released into the system, and peers enter the channel in a flash-crowd manner. We use our modeling framework to investigate the cost of releasing the new HD movie. Both our model and simulation results show that passive caching is inefficient in such a scenario. Instead, the system needs to actively push some video chunks of the HD movie to peers with available bandwidth, even if they are not watching it. Aiming at minimizing the server bandwidth consumption during a monitoring period when releasing the new movie, we use mixed integer linear programming (MILP) to find the optimal active caching solutions, and their efficacy is verified through simulation.

- We also design practical techniques that will be useful in the real-world implementation of our strategies. These practical techniques cover two major components: the overlay maintenance through peer gossiping; and the stream multiplexing with different uploaders. These practical techniques bridge the gap between our modeling work and real-world applications.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 introduces the background of the P2P technology and the challenges of offering HD channels in PA-VoD systems. The related work is also reviewed.

Chapter 3 describes our modeling framework and how it applies to stationary scenarios. We mainly focus on two caching strategies: FIFO and passive caching. Heuristic algorithms and their performance evaluation are also included. This chapter is based on our published work [13, 14].

Chapter 4 focuses on a typical case of non-stationary scenarios, i.e., a new HD movie is released into the system. Again, we customize our modeling framework accordingly, to find the best active-caching strategies and evaluate their efficacy through simulation. Practical distributed techniques useful in the implementation of real systems are also discussed with simulation results. This chapter is extended from our published work [15].

Chapter 5 concludes the thesis with a restatement of the claims and results of this thesis. The future directions and further development are also discussed.

Chapter 2

Background and Related Work

Nowadays, peer-assisted video-on-demand (PA-VoD) systems have demonstrated a great potential to harness the vast amount of peer-contributed resources, such as peer upload bandwidth and cached content, to lower the server bandwidth consumption, and thus the barrier to offering such services. However, due to the heterogeneity of the channel playback rate and popularity, as well as dynamic user behaviors, the bandwidth supply from peers varies greatly between channels, which poses the grand "resource imbalance" challenge to the research community, especially when HD movies are offered. HD channels usually require more bandwidth supply and cache space, which exceeds the capacity of the participating peers watching these channels. As a result, they either suffer from poor streaming quality, or impose a huge bandwidth consumption at the server.

In this chapter, we first present an overview of the P2P technology and PA-VoD services. We discuss the design objective as well as important components and principles. Then the resource imbalance problem in offering HD videos is explained in detail. We reason how the current network infrastructure impairs the capability of PA-VoD systems on utilizing the peer-contributed resources. At last, a brief summary of existing approaches to the problem in the literature is also presented.

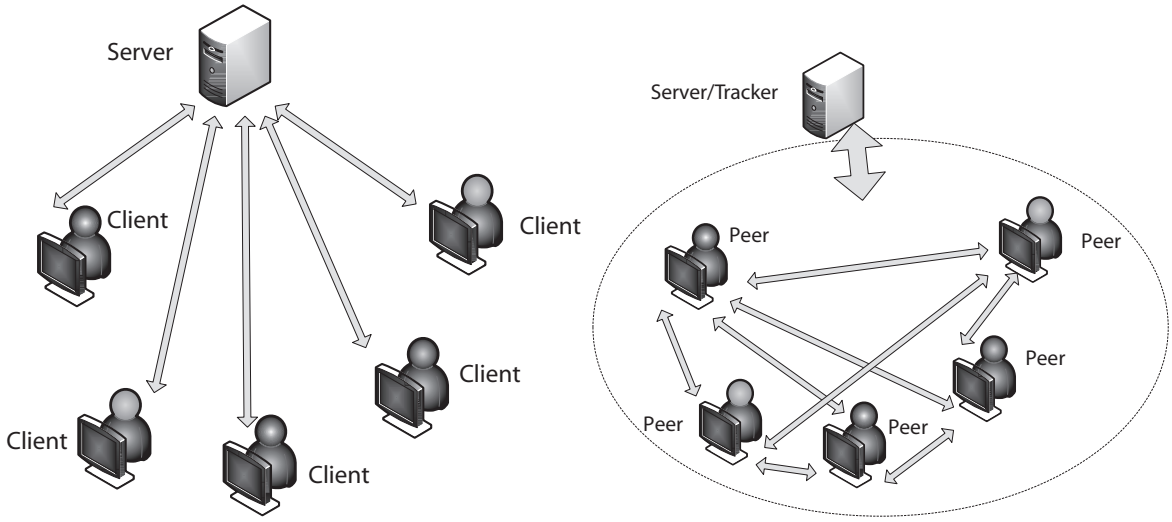
2.1 Peer-Assisted Video-on-Demand Systems: an Overview

The Peer-to-Peer Networking Technology

Traditionally, file distribution or video streaming systems on the Internet follow the *Client/Server* (C/S) model, in which a server or a group of clustered servers with greater processing power and bandwidth capacity support many capacity-limited end users, i.e., clients. This model inherently suffers from many problems. First, the bandwidth capacity of servers is not infinite, which results in the *scalability* problem. If the demand of clients exceeds the service capacity of the server(s), the system has to either degrade the service quality to all clients or repel some clients to keep others satisfied. Moreover, the extreme centralization of the system makes it vulnerable to *a single point of failure*. Under *Denial-of-Service* (DoS) or *Distributed Denial-of-Service* (DDoS) attacks, where malicious users attempt to saturate the server to make them unavailable to the intended users, C/S systems are easy to collapse.

One improved variant of such a model is *Content Delivery Network* (CDN), which has been later proposed in order to provide a better service quality as well as scalability by adding multiple content delivery servers distributed at the edges of networks. In such a kind of systems, file or video content is distributed to these servers in advance or on demand, and a client is usually served by the closest server. However, it is required that the aggregate capacity of all the content delivery servers should increase in proportion to the population of online users, which imposes a great cost on the servers, similar to C/S-based systems.

In contrast, in a *Peer-to-Peer* (P2P) system, end-users, i.e., peers, not only download content, but also contribute their own resources to the system when they upload to other peers. This will significantly reduce the burden placed at the server end and thus provide better scalability. According to a measurement study of the MSN video service in December 2006, the aggregate peer upload capacity accounts for 97% of their bandwidth demand. This means the server bandwidth consumption can be potentially reduced by 97%, if proper peer assistance is applied [16]. Figure 2.1 shows the different structures of the C/S and P2P model.



(a) The C/S model (b) The P2P model

Figure 2.1: The C/S model vs the P2P model.

Multi-Channel Peer-Assisted Video On-demand Systems

Regarding the viewing options offered to users, video streaming services can be classified into two categories: *live streaming* and *Video on-Demand* (VoD). In a live streaming system, the live video content is streamed to users in real time. In contrast, VoD allows users to watch any video programs, e.g., movies, TV dramas, or recorded matches etc., at any time, offering a better viewing flexibility to users, which makes these *multi-channel* systems extremely popular nowadays. As reported, YouTube [4] has 3 billion online videos ready for users to watch in 2011 [17], and 4 billion daily views as of January 2012 [5].

As a direct descendant of the application-layer video multicast [18, 19], modern online video streaming applications have adopted the P2P networking technology, for both live streaming and VoD. The past years have witnessed the fast development of P2P-structured video streaming systems since CoolStreaming [20] was first released in May, 2004 [21]. Many practically deployed commercial systems have attracted a huge number of online users, e.g., PPLive [10], PPStream [11], UUSee [12], AnySee [22], Xunlei Kankan [22], GridCast [23], etc. These systems provide thousands of channels, attracting millions of users. Take a glance at PPLive, a popular peer-assisted live and VoD streaming platform in China. PPLive is currently holding 100 million video clips [24]. As of June 2012, it recorded 36 million views per day, during the 2012 UEFA European Football Championship [25].

In P2P file sharing systems, users can only access a file after it is completely downloaded to the local hard drive, regardless of the actual downloading order of each small chunk of the file. In contrast, video chunks transmitted in video streaming systems are associated with strict playback order and deadlines. In order to meet the playback deadlines, each user has to seek enough bandwidth as well as available content, either from the server(s) or other peers. As a reaction to this, most P2P video streaming systems enforce more rigorous centralized coordination than file sharing systems. Moreover, the server(s) acts as the origin of all video programs, an indispensable functional part of video streaming systems. Therefore, in the rest of this thesis, we refer to P2P-structured VoD systems as *Peer-Assisted* VoD systems (PA-VoD) instead of P2P-VoD systems. Also, we refer to the different video programs in these systems as video *channels*.

The Design Objective

A good design of P2P systems always utilizes peer-contributed resources as much as possible, as to reduce the resource consumption at the server side. An important kind of such resources is the upload bandwidth. C/S or CDN-based VoD systems usually suffer from a huge *server bandwidth consumption*, due to the tremendous demand from users. This has imposed a heavy burden on maintaining these servers, as ISPs charge VoD service providers based on the 95% or total consumed volume of bandwidth. It has been estimated that YouTube, one of the most famous video-sharing service provider adopting CDN, has spent approximately 1 million US dollars per day, for the 30 million megabits-per-second bandwidth consumption, in the year 2009 [6].

In P2P systems, given the same volume of demanded bandwidth from peers, if more is supplied by peers themselves, less will be requested from the server. Therefore, in this thesis, the design objective of a PA-VoD system is defined as to minimize the server bandwidth consumption (SBC), which is also widely accepted in the existing literature [1, 9, 26, 27].

Design Component 1: Bandwidth Allocation

Bandwidth allocation in PA-VoD systems determines which peer should be allocated with how much bandwidth. With a carefully designed strategy, the peer upload bandwidth can be utilized as much as possible. A bad allocation strategy, in contrast,

can result in great waste of such peer contributed bandwidth.

To guarantee a real-time and fluent playback experience, the download rate of each peer should catch up with the *playback rate* of the video program currently being watched. The playback rate of a video program is its inherent property related to the playback quality. Nowadays, the playback rate of standard-definition (SD) video programs in peer-assisted video streaming systems varies between 300 and 500 Kbps, and that of high-definition (HD) channels can reach up to 1,000 Kbps (PPLive, 2007 [28]).

The current download rate of a peer is determined by many factors. Counter-intuitively, the downlink bandwidth capacity is usually not the determinant factor, as demonstrated by many studies [29, 30]. This is because the downlink bandwidth capacity of peers far more exceeds the playback rate of the video programs. For instance, ISPs in Canada usually provide residential users with a downlink bandwidth capacity between 5 to 9 Mbps, and cap the uplink bandwidth capacity to around 870 Kbps on average [31]. In fact, the download rate of a peer is determined by the aggregate upload bandwidth supporting it, either from the server(s) or other peers. Considering the above example, the maximum average download rate of all peers cannot exceed 870 Kbps without the supply of the server, as 870 Kbps is the maximum bandwidth a peer can contribute on average. From the perspective of a channel or the entire system, a common rule is the *bandwidth conservation law*, stated as follows. In many studies, the inequality is taken off for simplicity, which indicates that peers can fully utilize their upload bandwidth [9, 32–36].

Theorem 1 (Bandwidth Conservation Law)**.** *The aggregate download rate of all peers is no greater than the total upload capacity of all peers plus that of the server.*

The bandwidth conservation law has many practical values. First, it can be used to classify channels into *bandwidth-surplus* and *deficit* channels. In most cases, if the average peer upload capacity is greater than the playback rate of a channel, the peers can survive themselves with an appropriate bandwidth allocation strategy, as the bandwidth demand can be satisfied by the peer upload bandwidth. This kind of channels are referred to as *surplus channels*. In contrast, the channels with playback rates greater than the average peer upload capacity are *deficit channels*. For example, given `Channel A` with a playback rate of 600 Kbps, `Channel B` of 1,000 Kbps, and the average peer upload bandwidth of 870 Kbps, we can easily see that `Channel A` is a surplus channel while `Channel B` is a deficit channel. The server thus only

needs to offer little or a small amount of bandwidth to support surplus channels, e.g., `Channel A`, and use most of its bandwidth to support deficit channels, e.g., `Channel B`. In more advanced approaches, the extra bandwidth in surplus channels can be even utilized to help deficit channels. Moreover, the bandwidth conservation law can also be used to calculate the minimum amount of server-provided bandwidth to support a deficit channel, or the entire system. Consider the above example again. The server needs to offer an extra 130 Kbps per peer to `Channel B` on average, as to compensate the bandwidth deficit.

Design Component 2: Caching Strategies

In addition to the upload bandwidth, PA-VoD systems also require peers of a small cache space from their local drives. Such a space is used to store the video content that the peer is watching or has watched before. Video content stored in the local cache can be either fed to the video player or uploaded to other peers interested in it.

Due to the limit of the peer cache space, the caching strategy plays an important role in PA-VoD systems. Cache replacement occurs when the local cache of a peer is full. In order to accommodate new video chunks, the peer has to remove some existing ones from its local cache. A simple approach is *First-In-First-Out* (FIFO) [26,37], in which the earliest chunks are removed when necessary. In a more complex strategy, *passive caching*, the importance of the chunks in the local cache is evaluated and the least important one will be removed first. Some examples include *Least-Recently Used* (LRU) or *Least-Frequently Used* (LFU) [2,26,38]. Finally, in *active caching* [2,37,39], a peer may even actively fetch some video chunks that are not watched by itself. This may happen when this peer has extra bandwidth available, and is willing to help peers watching other channels using the fetched content. The caching strategies in P2P systems are similar to the replication strategies in CDN. However, in CDN the replication is performed at the edge-servers, as to satisfy the requests of users geographically close to these servers. In contrast, caching happens at each individual peer in PA-VoD systems, with the assistance of the tracker(s).

The size of the local cache varies in real systems. The local cache was originally fixed at 1 GB in PPLive [16] and PPStream. Nowadays many PA-VoD applications allow users to customize the size of the local cache, ranging from 512 MB to 10 GB. However, many users leave it as the default value as 1 GB, or even adopt the minimum setting, 512 MB. However, in this thesis we will show that a good caching strategy

will neutralize the limitation of the small peer cache and a 1-GB cache space can bring desirable performance in most cases.

The Relationship between Bandwidth Allocation and Caching strategies

In PA-VoD systems, balancing imbalanced resources calls for a perfect coordination of bandwidth allocation and caching strategies. Through bandwidth allocation a peer selects upstream or downstream peers, establishes transmission links, and sets the amount of the upload bandwidth allocated to each upload link. This is based on how well the corresponding content is replicated in the system, i.e., the *content availability*. Caching strategies decide which video chunks to remove when the peer local cache is full, and which to actively fetch in order to make more replicas. This will affect the content availability at peers. The mis-alignment of the two components will result in "*content bottleneck*". For instance, if a peer has available bandwidth to support another peer, but does not have the interested video chunks, no transmission will happen between the two peers. In this case the bandwidth of the peer is wasted, or *underutilized*.

Underlying Techniques

Bandwidth allocation and caching are two high-level design components of PA-VoD systems. To implement these components, system designers also employ a variety of underlying techniques.

The first thing is to decide the streaming overlay. In early days, structured tree-based PA-VoD systems were proposed using application level multicast, e.g., Overcast [18] and ESM [19]. Video programs are streamed from parent nodes to child nodes. Due to the complexity and rigidity of the tree-based structure, popular commercial PA-VoD systems such as PPLive, CoolStreaming and UUSee nowadays adopt a mesh-based overlay [1,28,40]. Each peer maintains a neighbor set with the assistance of a centralized tracker, and retrieves video stream from its neighbors. If one neighbor is saturated or aborts the connection, the peer can resort to the next available neighbor. Such an overlay is inherently more robust against peer churning, through maintaining backup upstream neighbors.

After peers obtain a set of neighbor information from the tracker, they will connect to these neighbors and start to advertise the video content they have to their neighbors. The exchange of such availability is achieved through *buffer map exchange*,

which produces transmission overhead. To minimize such overhead, buffer maps are encoded in the format of *bitmap*. Each bit is used to represent the availability of a file unit, depending on the segmentation granularity. Through bitmap exchange, the percentage of such overhead of popular systems can be lowered, varying from 1.5% to around 10% [1, 41].

Finally, the video content transmission is conducted in either the push or [42–45] pull-based [32–34, 46–48] manner. Push-based approaches are originally proposed to work with the tree-based structure, where an upstream (parent) peer pushes the video chunks to its downstream (child) peers, without receiving the download request from them. The transmission overhead associated with the download request is thus avoided, but the coordination between multiple upstream peers is a real challenge, as the streamed content between upstream peers has to be distinct to each other. In pull-based video transmission, a peer sends the download requests to its neighbors, specifying the targetted video chunks. The neighbors then respond with the requested video chunks if available. A peer decides locally which chunk to be downloaded from which neighbors. Therefore, the coordination between upstream peers becomes a trivial task, but the download request for each video chunk will introduce overhead. Recently, to achieve a balance, a hybrid approach with both push and pull-based approaches are adopted by commercial systems [1, 40]. Pull-based approaches are usually adopted at a higher granularity, e.g., a large chunk or segment of a video file. Then a segment or chunk is further divided into blocks, equivalent to a UDP packet, and a group of these blocks are pushed from the upstream to downstream peers, following the push-based manner. Network coding, which enables the perfect coordination of upstream peers, has made great contribution to the rise of the hybrid push and pull-based strategies [1].

2.2 The Challenges of Offering HD Channels

Online HD Videos

Today online VoD systems offer video programs with a variety of playback quality, including HD channels. However, the definition of online "HD" videos is still under debate and varies between different service providers. YouTube allows users to select the video quality in some video programs from 240p to 1080p. PPLive offers "Fluency" (SD), "high-definition" (HD), "super high-definition" (super-HD) and

"Blu-ray" options, with the last option available to paid users only. The categorizations of video qualities and typical playback rates of online PA-VoD systems are listed in Table 2.1 [1, 3, 28, 49, 50]. Here "PA" represents a peer-assisted system.

We can see from the table that most PA-VoD service providers set the playback rate of fluency or SD videos to around 500 Kbps. This is because the average peer upload capacity today is greater than 500 Kbps, and thus these channels can receive sufficient bandwidth from peers, i.e., surplus channels. However, HD channels usually have a playback rate higher than the critical value, the average peer upload capacity, and are deficit channels as a result. To support these bandwidth-consuming HD channels, the VoD providers adopt a variety of strategies, either by restricting the HD service to paid users only, or holding a smaller number of HD channels.

Table 2.1: The typical playback rates of online video programs.

VoD service provider	Playback qualities available		
Youtube (CDN)	360p: 800 Kbps	480p: 1,200 Kbps	720p (HD): $> 2,000$ Kbps
PPLive (PA)	Fluency: 400 Kbps	HD: 700 Kbps	Super-HD: 1,000 Kbps
PPStream (PA)	Standard: 500 Kbps	HD: 700 Kbps	
UUsee (PA)	NQ: 500 Kbps	HQ: 800 Kbps	
Xunlei Kankan (PA)	SD 480p: 600 Kbps	HD 720p: 1,000 Kbps	HD 1080p: 1,600 Kbps

The Resource Imbalance Problem

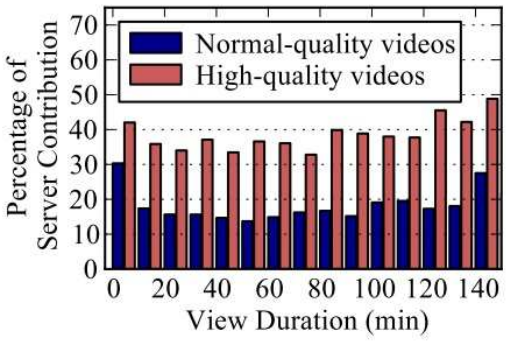
The resource imbalance problem in PA-VoD systems refers to the mismatch between the resource demand and supply in different channels or user groups. As explained in the previous section, in terms of bandwidth supply, video channels can be categorized into surplus channels and deficit channels, according to a critical metric, the average peer upload capacity. We observe that in many real systems, the average peer upload capacity often sits between the playback rates of SD and HD channels. In China, the typical upload capacity of residential users is 512 Kbps [1], between the playback rates of SD and HD channels of PPLive and PPStream listed in Table 2.1. In Canada, the average peer upload capacity can reach 870 Kbps, still residing between the Fluency channels and Super-HD channels of PPLive, or the SD 480p and HD 1080p of Xunlei Kankan. As a result, HD or super-HD channels are deficit channels and request extra support from the server. Compared with PA-VoD systems with SD channels only, these deficit HD channels require more system resources and centralized coordination, which bring a great challenge to system designers of PA-VoD systems.

In the future, the upload capacity of users may be greatly improved. For instance, in Canada, the ISPs begin to provide cable or fiber users with upload capacity over 10 Mbps [51]. Such user upload capacity will be sufficient to sustain the channels with the best quality in Table 2.1. However, as stated in [2], VoD service providers also tend to improve the playback quality of video programs in order to attract users, by leveraging the fast development of the user upload bandwidth. Online HD videos will be offered at significantly improved quality, e.g., Blu-ray, 3D and 4KHD, and the playback rate can be far beyond 40 Mbps. Therefore, to VoD service providers, there will always exist the channel heterogeneity. Some channels are "SD" channels with guaranteed streaming quality, which they limit the playback rate to be below the average peer upload capacity. Others are advanced resources which they offer to attract more users with extra cost, e.g., "HD" channels with much higher playback rates. Therefore, we can expect that in the next 10 years, the resource imbalance problem will still exist, with the coexistence of new "SD" and "HD" channels.

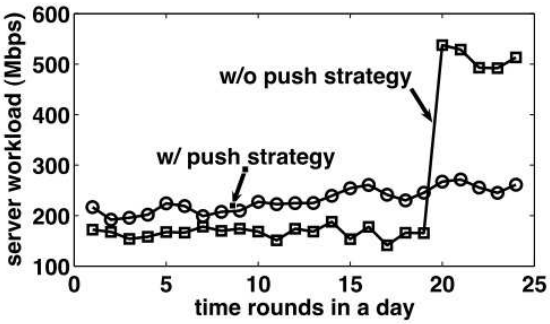
Dynamic User Behaviors: Stationary vs Non-stationary

When HD movies are offered in PA-VoD systems, peer-contributed resources vary dramatically between channels. First, even if the system is stable, i.e., the popularity of channels never changes, HD channels usually require more bandwidth supply and cache space, which exceeds the capacity of the participating peers, while peers in SD channels have surplus bandwidth [1, 3]. These scenarios are referred to as *stationary scenarios*. Figure 2.2a demonstrates the percentage of bandwidth supply from the server to normal-quality and high-quality videos respectively in UUSee, 2008 [1]. UUSee already employs peers who watched these HQ movies before to help those currently watching, however the server bandwidth consumption still acounts for around 40% of the total bandwidth demand. The same problem has also been reported by PPLive shown in Fig. 2.2c, where HD and Blu-ray channels suffer from a poor fluency rate, e.g., with many interruptions during the viewing process [3].

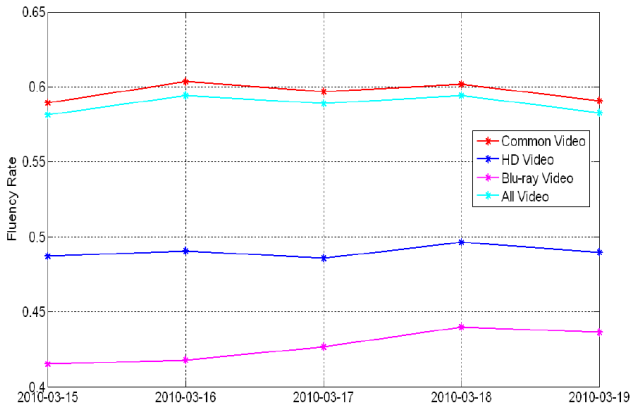
Moreover, a greater challenge is presented concerning the newly released movies [2]. In addition to improving the playback quality of movies, PA-VoD service providers periodically update their channel list, offering latest movies or TV programs to retain online users. Once a new movie is released, it becomes extremely popular and many users enter the new channel in a flash-crowd manner. On the other side, there will be few peers who carry the video content as none has watched the new movie



(a) SBC, UUSee 2010 [1]



(b) SBC, new movie release [2]



(c) The fluency rate, PPLive 2010 [3]

Figure 2.2: The performance of HD channels under stationary and non-stationary scenarios.

before, and thus few peer contributors. Due to a huge number of peers watching and few contributing, it imposes a heavy bandwidth consumption at the server after the movie is released. In these scenarios, the popularity of the movies has changed, and we refer to such scenarios as *non-stationary scenarios*. Figure 2.2b shows the server bandwidth consumption when a new movie is released (at round 20 in the figure) in [2], where a sudden increase at round 20 can be easily observed (the "w/o push strategy" curve). It is worth mentioning that the figure is for an SD channel case. For HD channels, the server bandwidth consumption is expected to be much higher.

Cross-channel Allocation: the Road to Solutions?

Cross-channel allocation was originated from BitTorrent-like file sharing systems, where BitTorrent has a similar terminology, *inter-torrent cooperation* [52]. It has been proved in theory that the inter-torrent cooperation can significantly improve the download performance of peers. In practical BitTorrent systems, peers may participate in different torrents while they download files, which makes such an inter-torrent cooperation possible. In many PA-VoD systems, peers also have been helping other channels by making use of the local cache and upload bandwidth [1, 28], i.e., *cross-channel allocation* or *view-upload decoupling* (VUD) [35,36]. The peers uploading to channels other than the one it is watching are referred to as *bandwidth helpers*. A usual approach is to make peers upload what they have watched and cached before, if many other peers are interested in the cached video content. This proves to be an effective approach to resource balancing when a system is offering SD-only service, as each channel mainly relies on the viewer-contributed bandwidth within the same channel and some small-scale help from outside the channel can be highly efficient.

When SD channels with surplus bandwidth and HD channels with bandwidth deficit co-exist in PA-VoD systems, it is natural to make SD channels help HD channels with the extra bandwidth. However, this is nontrivial for PA-VoD systems with HD channels. First, to help HD channels, the selected peers should have extra bandwidth and the cached HD video content at the same time, which makes it difficult to locate bandwidth helpers. Such bandwidth and content availability are highly dynamic, and can be easily affected by user behaviors. For instance, if a peer has watched HD channel `HD 1` and cached the video content, and then starts to watch another HD channel `HD 2` immediately, the `HD 1` content can be hardly used to support the viewers of `HD 1`, because `HD 2` also requires the peer of bandwidth contribution,

and the channel already suffers from bandwidth deficit. Only if this peer starts to watch an SD channel after watching `HD 2` can we safely retain its extra bandwidth and the cached `HD 2` content to help other `HD 2` viewers. Second, the HD video takes a larger space in the peer cache. For a 1,000-Kbps online HD movie, the typical byte length is around 1 GB. Considering that the peer cache limit is 1 to 2 GB, and some of the space has to be reserved for the movie being watched, in many cases a peer can only cache one complete or part of an HD movie. To utilize such partially cached HD movies to feed the requests of HD viewers on different chunks of the video, a match between the requests and the availability of the cached content needs to be achieved, which is a challenging task. At last, in the case that a new HD movie is released, no content has been cached by peers. The system thus needs to actively inject the HD content to some SD peers, i.e., through active caching. This will also introduce extra server bandwidth consumption, as the server(s) has to transmit the content to these active helpers. As HD videos have a larger file size, they also consume a greater server bandwidth when performing such an active caching.

2.3 Related Work

The potential of utilizing peer-contributed resources in VoD systems has been studied by Huang et al. [9], with the estimation that 97% of the server bandwidth consumption can be saved through a P2P solution. This boosts a variety of studies on large-scale PA-VoD systems [1, 26, 28, 53–59].

Research efforts in the literature mainly focus on reducing server bandwidth consumption through appropriate bandwidth allocation and caching strategies, including modeling, algorithm design, simulation-based experimentation and implementation. Although PA-VoD systems are usually multi-channel, researchers start from analyzing one single channel. Based on the results of a single channel, cross-channel allocation is also proposed and analyzed in multi-channel systems. However, most modeling work limits itself to PA-VoD systems with SD channels only, due to the great challenge of provisioning HD channels. In real systems, cross-channel allocation is implemented, but the strategies are lack of theoretical support, and thus the performance is far from optimal.

Single-Channel VoD

For a single-channel PA-VoD system, Annapureddy et al. investigated a variety of chunk scheduling strategies, with a focus on the network-coding approach [55, 60]. Wu et al. modeled and compared the performance of the passive-caching strategies, LFU and LRU, with a simple FIFO [26]. Parvez et al. built fluid models to compare different bandwidth allocation strategies and conjectured that a chain-based allocation is able to overcome the imbalance of the peer bandwidth supply [57]. Yang et al. proposed practical queuing techniques to overcome the resource imbalance problem through simulation [61]. Ciullo et al. modeled the chain-based bandwidth allocation, under both stationary [62] and non-stationary scenarios [63]. Zhao et al. characterized the impact of VCR operations for a single VoD channel [27]. The bandwidth allocation for BitTorrent-like systems under a flash-crowd scenario has been studied by D'Acuntoa et al., where they characterized the trade-off between injecting new chunks into the system and replicating existing ones [64]. These studies focused on single-channel systems, and thus did not consider cross-channel allocation.

Multi-Channel Live Streaming and VoD

For multi-channel live streaming systems, view-upload decoupling (VUD) aims at allocating the peer upload bandwidth across channels by decoupling the uploading content from the movie that a peer is watching [35, 36]. Linear programming models have also been built for the bandwidth allocation with several user-customized viewing behaviors [65]. These studies all applied cross-channel allocation to live streaming systems.

Concerning multi-channel VoD systems, [2, 38, 66, 67] built different models to study the server bandwidth consumption with a limited cache size at each peer and homogeneous or heterogeneous peer upload capacity. However, these studies all assumed that channels are of homogeneous playback rates, and conducted movie-level analysis without considering that HD movies are partially cached. Moreover, they mainly focus on stationary systems with strong centralized coordination. Similar to these studies, Amoza et al. modeled cloud-assisted P2P VoD systems in the steady state under stationary scenarios, where they focused on optimizing the caching strategies on the super nodes in the cloud [68].

VUD may also apply to a fine-grained sub-movie level, i.e., the chunk or segment (a group of chunks) level. Working at the chunk/segment level, He et al. built linear

programming models by assuming the helpers are known [69], and Wang et al. developed heuristic algorithms to locate them [70]. Supporting HD channels is still out of the scope of these two studies.

Two recent studies focused on the similar topic to our work in this thesis. Ciullo et al. proposed an analytical framework to characterize the scaling laws for the server bandwidth consumption with passive and active caching under stationary scenarios and centralized control [37]. Different from existing studies relying on centralized optimization, Zhao et al. proposed the first analytical work to achieve close-to-optimal streaming capacity through the management of neighbors at each individual peer, i.e., under decentralized control [39]. Passive and active caching were also studied, under stationary scenarios. These two studies both modeled cross-channel allocation, provided consistent results to our work, and also best complemented with this thesis.

2.4 Summary

In this chapter we have introduced the *resource imbalance* problem in offering HD channels in multi-channel PA-VoD systems. The problem is caused by the imbalanced distribution of the demand and supply of resources between channels. Some channels are well-provisioned, e.g., SD channels, as the demand on bandwidth can be satisfied mostly by the viewing peers themselves. On the other hand, HD channels usually require much more bandwidth and cache space, and thus will suffer from poor performance without extra help from other channels. To solve this problem, *cross-channel allocation* is desired. It includes two components: *bandwidth allocation* and *caching* strategies, and a perfect coordination in between is required as the two components affect each other. Existing research efforts mainly focus on PA-VoD systems with SD channels only, despite the fact that many systems are deployed with both SD and HD channels. The limited understanding of supporting HD channels leads to the inappropriate design of the bandwidth allocation and caching strategies, which causes the poor streaming quality of these HD channels in real systems.

Chapter 3

The Generic Modeling Framework and Stationary Scenarios

In stationary scenarios, a PA-VoD system is assumed to be stable enough. There is a fixed number of users in the system, and users transition between different channels following fixed transition probabilities, e.g., the closed Jackson model. Although such stationary scenarios do not reflect the evolution of movie popularity and peer churning in real systems, it resembles the scenarios of a short duration (usually several hours) when the population of online peers is stable, and the insights gained can facilitate understanding more dynamic scenarios. In this chapter, we focus on optimal caching with corresponding bandwidth allocation strategies for PA-VoD systems under stationary scenarios. We first propose a generic modeling framework to characterize PA-VoD systems with cross-channel allocation. The modeling framework is then applied to stationary scenarios, leading to statistical performance bounds in terms of server bandwidth consumption for two popular caching strategies, FIFO and passive caching. In addition, we formulate bandwidth allocation as a linear programming problem to calculate the tight instantaneous lower bound, and design heuristic algorithms for peer cache replacement and bandwidth allocation.

The chapter is organized as follows. Section 3.1 briefly introduces the background and our contributions. We explain our generic modeling framework in Section 3.2. In Section 3.3 the generic modeling framework is customized as our detailed mathematical models to capture the steady-state and transient behaviors of PA-VoD systems. Heuristic algorithms are proposed in Section 3.4, which are then evaluated in Section 3.5. Section 3.6 concludes the chapter.

3.1 Introduction

As explained in Chapter 2, due to the heterogeneity of the channel playback rate and popularity, as well as dynamic user behaviors, the bandwidth supply from peers varies greatly between channels, i.e., the grand "resource imbalance" challenge.

To overcome the bandwidth imbalance problem in PA-VoD systems, we may first investigate the solutions for peer-assisted live streaming systems. The view-upload decoupling (VUD) strategy is proposed, which decouples what a peer is uploading from what it is watching [35,36]. A peer may be assigned to upload to another channel than the one it is currently watching, i.e., enabling cross-channel optimization, and the objective is to equalize the bandwidth demand-to-supply ratio for every channel through a *water-leveling* approach. If a channel is observed to be well-provisioned, peers will stop contributing to it, and the resources will be allocated to other deficit channels, with the coordination of a centralized server/tracker.

"Water-leveling" approaches for PA-VoD systems have been studied at the movie level with homogeneous, moderate playback rate [2, 38, 66, 67]. However, HD movies need much more bandwidth supply and cache space, and the content may not be completely cached at peers, which makes the existing approaches inapplicable. "Water-leveling" may also apply to a fine-grained model, e.g., at the chunk or segment level [69, 70], but obtaining the demand-to-supply ratio for each segment in real time is a major challenge.

Therefore, we apply a mixed strategy at different granularities. We model the system at the movie level. Even if a movie is partially cached at a peer, we consider it able to help other peers using such partially-cached video content. Later, we show that this can be achieved through heuristic algorithms and practical methods. Throughout this thesis, two constraints are taken into account: 1) each peer has a limited cache size; 2) the playback rate of some channels in the system exceeds the average peer upload capacity, e.g., HD channels. We allocate and balance such two kinds of resources, peer cache and upload bandwidth, under different user viewing behaviors. Our contributions in this chapter are highlighted as follows.

- We propose a generic modeling framework to capture the essential characteristic of a PA-VoD system, i.e., the demand-vs-supply relationship. Our modeling framework can be easily modified or extended to characterize a variety of user behaviors and caching strategies.

- We apply our modeling framework to stationary scenarios where users transition between channels following fixed probabilities. This allows deriving statistical performance bounds for PA-VoD systems with heterogeneous playback rates and dynamic user behaviors. The model indicates that user viewing behaviors can largely affect the bandwidth provisioning for HD channels, e.g., switching to SD channels after watching HD movies is highly desirable to reduce the server bandwidth consumption.

- In addition, a tractable linear programming optimization problem is formulated at the segment level to minimize the server bandwidth consumption for any given instance of the system at any time, which can be solved in polynomial time in a centralized manner. It provides tight instantaneous performance bounds, and thus serves as the perfect benchmark when evaluating bandwidth allocation algorithms.

- We also develop heuristic bandwidth allocation and cache management algorithms without calculating the dynamic segment-level demand-to-supply ratio. The performance is evaluated with comparison to the water-leveling strategies and our lower bounds from the model through extensive simulation, which verifies the efficacy of our algorithms in stationary scenarios.

- Other than the closed Jackson network, a representative stationary model, we investigate more dynamic user behaviors through simulation, e.g., the diurnal arrival pattern. It is verified that our heuristic approaches achieve a desirable performance under these scenarios as well, furthering the insights gained from our modeling work and heuristic algorithms.

3.2 Multi-Channel PA-VoD Modeling Framework

3.2.1 System Description

In peer-assisted video streaming systems, if the playback rate of a channel is less than the average peer upload bandwidth $\bar{u}$, e.g., SD channels, the viewers/peers of the channel can adopt a proper bandwidth allocation strategy within the channel (e.g., chain-based algorithms) to achieve the download rate no less than the playback rate [2, 57, 62, 63]. This means that SD viewers can watch the video program without

interruption. Meanwhile, the bandwidth support from the server can be minimized, regardless of any kind of user dynamics, i.e., "surplus channels". On the other hand, HD channels usually have a playback rate up to $1,000$ Kbps, while the typical uplink bandwidth of residential Internet access links varies from 384 Kbps to 900 Kbps, depending on the network infrastructure and the services provided by ISPs. Therefore, an HD channel needs extra "help" from SD channels or the server(s). We call peers that are watching SD channels, but uploading to HD channels *bandwidth helpers*. To serve as such a "helper", a peer needs to satisfy two conditions. First, it has unused upload bandwidth available (usually SD viewers). Second, the content of the corresponding HD movie has to be stored ("cached") in its local cache.

Due to the limit of the peer cache space, cache replacement occurs when the local cache of a peer is full. In order to accommodate new video chunk/segments, the peer has to remove some existing ones from its local cache, referred to as *caching* strategies. Three popular caching strategies are considered and modeled in this thesis.

FIFO

A simple but naive approach is FIFO, in which the earliest segments of the earliest movie is removed when the buffer is full. This strategy adopts no deliberate adjustment on the number of replicas of video segments in the system.

Passive Caching

A peer selects a movie or segment to remove following the water-leveling criterion rather than FIFO, as to balance the resource provisioning to match the dynamic demand at either the movie or segment level. Some examples include *Least-Recently Used* (LRU) or *Least-Frequently Used* (LFU). We can also simply let a peer remove early SD segments first but keep HD ones it has watched before, as SD movies are most likely to be well-provisioned. This strategy is effective in some scenarios, but still passive. If an HD movie is never watched before, there is no chance for a peer to cache it. We refer to the helpers using passively-cached content as *passive cachers/helpers*.

Active Caching

In many cases, the server can actively introduce more helpers for a particular HD channel, if the channel is observed as poorly provisioned and passive caching alone is considered ineffective. Peers watching SD channels with available upload bandwidth

are selected as the helpers, and these helpers actively fetch the content of the HD channel from the server and use it to serve the viewers watching that HD movie later. These helpers are referred to as *active cachers/helpers*.

A Short Summary on Caching Strategies

With a limited cache space, passive caching retains what is considered useful, and thus its capability of balancing replicas depends on a longer viewing history. In contrast, FIFO only allows peers to cache the most recent segments it has watched, so its capability of balancing the video replicas is determined by its most recent viewing behaviors, e.g., within a few hours. Given a larger peer cache, FIFO will perform better, as the number of "recent" segments increases along with the cache space.

Intuitively, with a passive-caching strategy, users can update their cached content slowly through a natural viewing process to meet the demand of HD viewers. However, if a new movie is released in the system, and users form a flash crowd to watch it, existing passive cachers may be insufficient for helping all the viewers, and there will be a considerably heavy workload on the server during the first several hours or days. In this case, the server can actively push the video content to some peers with extra upload bandwidth before or right after the movie is released. Pushing such content to peers also consumes server bandwidth, but these active cachers can serve as the helpers thereafter [2].

3.2.2 Modeling Framework

To model a PA-VoD system, we start from a single HD channel. The bandwidth provisioning of HD channels is critical because they are deficit channels, contributing to a major portion of the server bandwidth consumption. Figure 3.1 shows our modeling framework. There are two queues considered in an HD channel i, the viewer queue and the helper queue, with the number of residing peers as x_i and y_i, respectively. The arrival and the departure rates are denoted as λ_i and θ_i for the viewers, and η_i and γ_i for the helpers, respectively. Note that such arrival and departure rates are fixed for stationary scenarios and can be time-dependent under non-stationary ones. The residence time of viewers is t_i^{view}. After t_i^{view}, a peer leaves the viewer queue, and may either become a helper, or leave the current channel. We denote the probability that a viewer becomes a helper as $p_i^{\text{view}\to\text{help}}$, and that of leaving the channel as $p_i^{\text{view}\to 0}$. Once a peer becomes a helper, it stays there for time

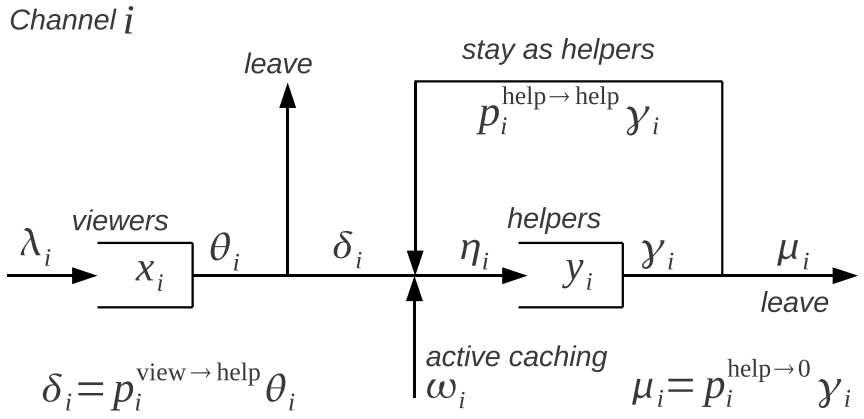


Figure 3.1: The modeling framework of PA-VoD systems.

Table 3.1: Notations of the modeling framework.

Symbol	Definition
$\bar{u}$	The average upload capacity of all peers
x_i, y_i	The number of viewers, helpers of channel i
λ_i, θ_i	The arrival, departure rate of the viewers of channel i
η_i, γ_i	The arrival, departure rate of the helpers of channel i
ω_i	The number of introduced active cachers of channel i
t_i^{a}	The residence time of queue a of channel i
$p_i^{\text{a}\to\text{b}}$	The transition probability from queue a to b of channel i
$p_i^{\text{a}\to 0}$	The probability of leaving channel i from queue a
$\bar{d}_i^{\text{view}}$	The average viewer bandwidth deficit of channel i
$\bar{s}_i^{\text{help}}$	The average helper bandwidth contribution of channel i

t_i^{help}, and then makes a decision of whether to stay or leave. Such probabilities are denoted as $p_i^{\text{help}\to\text{help}}$ and $p_i^{\text{help}\to 0}$, respectively. For active caching, denote the arrival rate of such active cachers as ω_i. At last, denote the average viewer bandwidth deficit as $\bar{d}_i^{\text{view}}$, and the average bandwidth supply contributed by helpers as $\bar{s}_i^{\text{help}}$. Other parameters are also marked in the figure, and the notations used in our modeling framework are listed in Table 3.1.

The framework captures the major characteristics of a PA-VoD system, such as the bandwidth deficit of viewers as $x_i\bar{d}_i^{\text{view}}$ and the extra bandwidth supply of helpers as $y_i\bar{s}_i^{\text{help}}$ for each channel i. The parameters $p_i^{\text{view}\to\text{help}}$, $p_i^{\text{view}\to 0}$, $p_i^{\text{help}\to\text{help}}$, $p_i^{\text{help}\to 0}$, t_i^{view}, t_i^{help}, and ω_i can be easily customized to model different caching strategies under a variety of scenarios, and x_i, y_i, $\bar{s}_i^{\text{help}}$ and $\bar{d}_i^{\text{view}}$ are derivable after these parameters are specified. For instance, $\omega_i = 0$ indicates FIFO or passive caching, and

$\omega_i > 0$ represents active caching. Moreover, for stationary scenarios, we can derive the steady-state metrics by assuming the arrival rate of each queue is equal to its departure rate. If λ_i is dependent on time, i.e., non-stationary scenarios, the framework is able to capture the effect of the evolution of movie popularity or user dynamics, and thus the cost of accommodating such dynamics at the server. At last, the framework is not limited to a single HD channel. If the transition matrix on channels is given, the probability $p_i^{\text{view}\to\text{help}}$ for any channel i can be derived, and thus the total server bandwidth consumption of all the channels.

This modeling framework is the basis of our models and will be used throughout this thesis. We use it to model FIFO and passive caching in stationary scenarios in Chapter 3, and active caching in non-stationary scenarios in Chapter 4.

3.3 The Detailed Model for Stationary Scenarios

In this section, we apply the modeling framework to stationary scenarios. We first describe the settings and assumptions of our detailed models, and then present theoretical bounds for the server bandwidth consumption in the steady state and at any time instance. The terms and notations used in this section are listed and briefly explained in Table 3.2.

3.3.1 Model Customization and Assumptions

We assume all video programs (movies) are released from a centralized server (or servers), and the server is able to support any peers at any time. However, the amount of such server-offered bandwidth will add up to the server bandwidth consumption, SBC. There are two categories of video programs, standard-definition (SD) and high-definition (HD) movies, and peers watching SD (HD) movies are referred to as SD (HD) viewers. The playback rate of HD movies r_H is much higher than the average peer upload capacity $\bar{u}$ and that of SD movies r_S lower than $\bar{u}$, which are quite representative in today's multi-channel VoD systems [1, 3]. We adopt the well-accepted assumption that there is no downlink bottleneck, since the peer download capacity is usually large enough to accommodate HD videos [2, 27, 37, 39]. Another common assumption is that each peer contributes a finite cache space which can only store a small number of movies [2, 38, 66]. Taking into account the fact that HD movies usually occupy more cache space, similar to [27, 37, 39, 71], we assume that

Table 3.2: Notations of the detailed model, stationary scenarios.

Symbol	Definition
SBC	Total server bandwidth consumption
D	Total bandwidth demand of peers
B	Total bandwidth supply of peers
η	Efficiency of utilizing the peer upload bandwidth
r_S	Playback rate of an SD movie
r_H	Playback rate of an HD movie
r_i	Playback rate of movie i
r_i^{seg}	Playback rate of segment i
T	Time duration of each movie
S	Number of distinct segments in the system
$\boldsymbol{P}^c$	Transition probability matrix between all available video categories
$\boldsymbol{P}^m$	Transition probability matrix between video channels
p_{ij}^c	Probability of transition from category i to j
p_{ij}^m	Probability of transition from movie i to j
$\bar{u}$	Average upload capacity of all peers
u_j	Upload capacity of peer j
N	Number of peers in the system
M	Number of channels in the system
N_S	Expected number of viewers in SD channels in the steady state
N_H	Expected number of viewers in HD channels in the steady state
N_i	Expected number of viewers of channel i in the steady state
H_i	Expected number of helpers of channel i in the steady state
$\boldsymbol{W}$	Watching matrix that indicates which peer is watching which segment
$\boldsymbol{C}$	Cached matrix that indicates peer-cached segments
$\boldsymbol{A}$	Allocation matrix that indicates a bandwidth allocation

the peer cache can store either two complete SD movies or one single HD movie, and all movies are assumed to have the same time duration, T. This is reflected by the typical settings of real systems. For example, the size of the local cache is usually fixed at 1 GB in PPLive [16, 26] and PPStream. For an SD (HD) movie of playback rate 500 (1,000) Kbps, the cache can store about two SD movies or one HD movie, each of 2.3 hours, a representative time duration of full-length movies. We divide an SD (HD) movie into 10 (20) segments to facilitate the segment-level optimization. Here the segment is the unit for content scheduling and cache management. Each segment may be further divided into smaller sub-segments, e.g., chunks or blocks, serving different purposes. Chunk can be the unit of playback, and block is usually the transmission unit, i.e., a single UDP packet. All segments have the same size, i.e., 1/20 GB for a 1-GB movie, which is comparable to the settings in real systems [1].

To ensure that a user watches a movie smoothly, the user download rate must catch up with the video playback rate. We assume that peers stream exactly at the video playback rate, which is also a common assumption in [2, 37, 62–64, 70]. Such an exact-rate streaming can be easily implemented using a sliding window of a fixed size: all the segments within the window can be requested for downloading, while the priority is given to the segments closest to the current playback point; the window is moving forward as the segments are consumed by the video player. Peers first attempt to seek bandwidth supplies from each other, and resort to the server at last if the desired streaming rate still cannot be achieved. Peers watching the same channel are referred to as "concurrent peers/neighbors" to each other [2], and those uploading to other channels are "helpers" [69,70]. Therefore, a peer can receive bandwidth support from its concurrent peers/neighbors, helpers, or the server(s).

There are N peers in the system. When watching a movie, a peer starts from the beginning of the movie and watches each segment sequentially until finishing the last one. After that, the peer will transition to another channel. Such user behaviors follow the closed Jackson network model adopted in [2, 35, 36, 38, 67], which is useful to analyze systems in the steady-state and provides insights for more dynamic scenarios. When transitioning to another channel, we assume that a peer first determines whether it will stay in the same movie category (i.e., SD or HD). A transition matrix $\boldsymbol{P}^c$ at the category level is used to capture this behavior, with elements p^c_{ij} defined as the probability of transition from category i to j. For instance, p^c_{HS} denotes the transition probability from HD to SD movies. After determining the next category, a peer selects a movie in the chosen video category to watch. The transition between

any two movies is determined by a transition matrix $\boldsymbol{P}^m$ at the movie level, which conforms to $\boldsymbol{P}^c$. The transition probability from channel i to j is defined as p_{ij}^m. Considering that a user usually does not return to the movie it has just watched, we set $p_{ij}^m = 0$, if $i = j$. Therefore, such user transition behaviors at the category and movie level can be simply modeled as Markov Chains, and the transition probabilities $p_i^{\text{view}\to\text{help}}$, $p_i^{\text{view}\to 0}$, $p_i^{\text{help}\to\text{help}}$, $p_i^{\text{help}\to 0}$ in the modeling framework in Section 3.2 can be easily calculated using the two-level transition matrices.

3.3.2 Movie-Level Steady-State Analysis

In this section, we present the steady-state analysis at the movie level, when the transition matrix $\boldsymbol{P}^c$ and $\boldsymbol{P}^m$ are given, i.e., the user transition behavior is known.

The Universal Optimal Performance Bound

The server bandwidth consumption (SBC) can be simply computed as the total bandwidth deficit of the system, i.e., $SBC = (D - B\eta)^+$, where D is the total bandwidth demand, $B = N\bar{u}$ is the total upload capacity of all peers, $a^+ := \max\{a, 0\}$, and $0 \leq \eta \leq 1$ is the bandwidth efficiency factor, which indicates the efficiency of utilizing the peer upload bandwidth. From the transition probabilities in $\boldsymbol{P}^c$ and $\boldsymbol{P}^m$, we can derive the expected performance in the steady state, such as N_S (N_H), the expected number of peers watching SD (HD) movies, and N_i as that of each movie i, etc. Therefore, the expected total bandwidth demand is $D = N_S r_S + N_H r_H = N(\frac{p_{HS}^c r_S}{p_{HS}^c + p_{SH}^c} + \frac{p_{SH}^c r_H}{p_{HS}^c + p_{SH}^c})$. Some SD peers may not consume bandwidth if it is watching an SD movie that has been watched before and still resides in its local cache. Denote the probability of peers becoming such "no demand" SD viewers as P_{null}, the sever bandwidth consumption can be calculated as

$$SBC = (D - Nr_S P_{\text{null}} - B\eta)^+. \tag{3.1}$$

Setting $\eta = 1$, which means that all peers are able to fully utilize their upload bandwidth to support others, we have the first performance bound SBC_{opt}, **Bound I**, for the server bandwidth consumption. Bound I is the best possible performance for any bandwidth allocation or caching strategies, as it is calculated on the total demand and supply of the entire system, assuming the resources are perfectly balanced among channels.

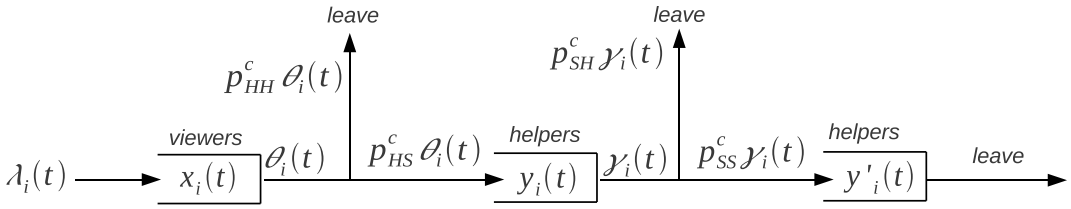


Figure 3.2: An HD channel i with the FIFO caching strategy.

Performance Bounds with Content Bottleneck

1. FIFO

Due to the limited cache space of each peer, a peer cannot always find the needed video content from other peers, especially when a naive FIFO approach is adopted. In this case, the peer upload capacity is underutilized, which is referred to as "content bottleneck". This will lead to $\eta < 1$, so the goal of the system design is to maximize η. In our system, SD viewers have more upload bandwidth than their demand ($\bar{u} > r_S$), which can be allocated to help HD viewers. However, the utilization of such bandwidth is determined by the availability of the HD content cached by these SD viewers. We define a group of bandwidth allocation strategies following the *Bandwidth Helping Principles*: viewers in SD channels first try to satisfy their concurrent neighbors within the same channel, and then use the extra bandwidth to upload the cached content, if needed, to HD viewers. HD viewers will never serve as "helpers" for other channels as they have already suffered from their own bandwidth deficit.

To derive the maximum bandwidth efficiency η for bandwidth allocation algorithms following the *Bandwidth Helping Principles* and FIFO caching, we extend our modeling framework in Section 3.2 to build a simple queuing network to obtain the expected number of viewers and helpers, shown in Fig. 3.2. Let x_i denote the number of viewers in an HD channel i in the steady state. After a peer finishes watching the HD movie, it has the HD content in its cache, which will remain for a while. When the peer transitions to another HD channel after watching channel i, it cannot serve as a helper and will leave the queuing system immediately, with probability p^c_{HH} or p^c_{SH}. As the peer cache can only store two SD movies, if it has transitioned more

than twice to SD channels, the cached HD content will be completely removed, and thus this peer is not able to serve as a helper. Therefore, we let y_i and y_i' denote the number of helpers for these two cases: peers that are watching the first and the second SD movie after finishing the HD movie i. These viewers and helpers form a queuing network including three G/D/∞ queues with service time T. In the steady state, the arrival rate of each viewer and helper to any queue equals the corresponding departure rate. Therefore, $\lambda_i = x_i/T$, $x_i/T * p_{HS}^c = y_i/T$, $y_i/T * p_{SS}^c = y_i'/T$, and $x_i = N_i$. Solving the equation set above, the expected number of helpers for channel i is $H_i = y_i + y_i' = N_i(p_{HS}^c + p_{HS}^c p_{SS}^c)$.

Remarks: From the equation above we can see that in the steady state, the number of helpers for each HD channel is in proportion to the viewers of that channel. If there are more viewers watching an HD channel, there will be more helpers helping it. Moreover, if the expected numbers of HD and SD viewers, N_H and N_S, are fixed (i.e., the aggregate demand and supply of the entire system is fixed), we can derive $p_{SS}^c = 1 - p_{SH}^c = 1 - p_{HS}^c N_H/N_S$, so $H_i = N_i p_{HS}^c(2 - p_{HS}^c N_H/N_S)$, which is monotonously increasing on $p_{HS}^c \in [0,1]$, if $N_S > N_H$. That means, if there are more SD viewers than HD ones, which is usually the case in real systems [1], transitioning to SD channels after watching an HD movie (a larger p_{HS}^c) will bring more helpers for the HD channel (a larger H_i). With H_i derived, the supply that can be offered by such helpers is $(\bar{u} - r_S)H_i$ and the total amount of such helping bandwidth in the system is $\sum_{i\in\{HD\}} H_i(\bar{u} - r_S) = N_H(p_{HS}^c + p_{HS}^c p_{SS}^c)(\bar{u} - r_S)$.

According to our model, there exists the case that a peer happens to watch an SD movie i already stored in its local cache, when the movie it is currently watching happens to be the one that it watched $2T$ ago. The probability can be calculated as $P_{\text{null}} = \sum_{i\in\{SD\}} (p_i^m \sum_{j\in\{SD\}} \frac{p_j^m p_{ji}^m}{\sum_{k\in\{SD\}} p_k^m p_{ki}^m} \frac{p_i^m p_{ij}^m}{\sum_{k\in\{SD\}} p_k^m p_{kj}^m})$, where p_i^m is the steady-state popularity of movie i, which can be computed from the transition matrix $\boldsymbol{P}^m$ at the movie level. Let the bandwidth of such "no demand" SD viewers be equally utilized by other SD viewers, and such helping bandwidth is increased to $N_H(p_{HS}^c + p_{HS}^c p_{SS}^c)(\bar{u}N_S/(N_S - NP_{\text{null}}) - r_S)$. Therefore, the maximum bandwidth that can be utilized from peers is $B_{\max} = (N_S - NP_{\text{null}})r_S + N_H\bar{u} + N_H(p_{HS}^c + p_{HS}^c p_{SS}^c)(\bar{u}N_S/(N_S - NP_{\text{null}}) - r_S)$, and the maximum bandwidth efficiency is thus $\eta_{\max} = B_{\max}/N\bar{u}$. Taking $\eta_{\max}$ into Eqn. (3.1), we have **Bound II** for FIFO, where the content bottleneck with FIFO is taken into account.

2. Passive Caching

Now we derive the server bandwidth consumption of passive caching, and prove

that a simple passive-caching strategy achieves the best (or minimum) SBC for the steady state of the system, i.e., $\omega_i = 0,\ \forall i$.

In our passive-caching strategy, we let SD viewers use half of its local cache (0.5 GB) to store the most recently watched HD movie, and the rest of the space to store the SD movie it is watching. Therefore, the SD movie can be cached entirely while the HD movie is cached in its 50% in file size. In order to derived the best performance of passive caching, we assume such 50% HD video content can still satisfy the requests for any parts of it, and show that it is achievable through a smart bandwidth allocation in following sections. Moreover, according to our assumption, no peers will watch the movie it just finishes. As only 0.5 GB is used for storing SD movies, the last watched SD movie will be removed completely after the peer finishes watching another SD or HD movie. Therefore, if SD viewers decide to watch a previously watched SD movie, there is no chance that they can find the video content in their local cache, i.e., $P_{\text{null}} = 0$. Passive caching can be illustrated in the original modeling framework, i.e., Fig. 3.1.

Theorem 2. *Under stationary scenarios, assuming the server has unlimited capacity, passive caching achieves the optimal expected server bandwidth consumption* $SBC_{\text{opt}} = (D+Nr_SP_{\text{null}}-B)^+$, *where* $D = \sum_{i=1}^{M} \overline{x}_i r_i$ *and* $B = N\bar{u}$ *are the total bandwidth demand and supply of all peers, respectively,* r_i *is the playback rate of channel* i, *and* $a^+ := \max\{a, 0\}$.

Proof. The optimal SBC in the steady state is calculated as $SBC_{\text{opt}} = (D - B + Nr_SP_{\text{null}})^+ = (N_Sr_S + N_Hr_H - N\bar{u})^+$, where $N_S = Np^c_{HS}/(p^c_{HS} + p^c_{SH})$ and $N_H = Np^c_{SH}/(p^c_{HS} + p^c_{SH})$ are the expected number of SD and HD viewers in all channels, respectively. In the steady state, the arrival rate of each queue in Fig. 3.1 is equal to the corresponding departure rate. Thus $\lambda_i = \overline{x}_i/T = \theta_i$ and $p^c_{HS}\theta_i + p^c_{SS}\gamma_i = H_i/T = \gamma_i$, which leads to $\overline{y}_i = (p^c_{HS}\overline{x}_i)/(1 - p^c_{SS})$. Therefore, the expected sever bandwidth consumption for this HD channel i is $SBC_i = (D_i - B_i)^+ = (\overline{x}_i(r_H - \bar{u}) - \overline{y}_i(\bar{u} - r_S))^+ = (\overline{x}_i(r_H - \bar{u}) - (p^c_{HS}\overline{x}_i(\bar{u} - r_S))/p^c_{SH})^+ = (N_H(r_H - \bar{u}) - N_S(\bar{u} - r_S))^+(\overline{x}_i/N_H)$. With the expected server bandwidth consumption SBC_i for each channel i, the total expected SBC of all the HD channels can be calculated as $SBC_H = \sum_{i\in HD} SBC_i = \sum_{i\in HD}(N_Hr_H + N_Sr_S - (N_H + N_S)\bar{u})^+(\overline{x}_i/N_H) = (N_Hr_H + N_Sr_S - N\bar{u})^+$. As SD viewers require no extra bandwidth from the server, the total server bandwidth consumption is $SBC_{\text{passive}} = SBC_H = (N_Hr_H + N_Sr_S - N\bar{u})^+ = SBC_{\text{opt}}$. □

Remarks: Theorem 2 indicates that if a PA-VoD system is sufficiently stable, no complicated caching strategy is required. Each peer can simply keep at least half of the HD content it watched before until it starts to watch another HD movie, without any global information or coordination. This property best fits the distributed nature of a P2P system. The reason is quite similar to the FIFO approach. After peers finish an HD movie, they serve as helpers for it. Thus, if there are more viewers in an HD channel, there will be more helpers that watched it before in the steady state.

3.3.3 Segment-Level Transient-State Analysis

Although the steady-state analysis offers statistical bounds, a PA-VoD system can be highly dynamic, which makes the system performance deviate from these expected bounds at a given time. To understand the transient-state behaviors of PA-VoD systems, we continue to study the optimal bandwidth allocation at the segment level at a given time instant, by assuming that all needed global knowledge is known.

We define three $S \times N$ matrices, where N is the number of peers and S the total number of distinct movie segments in the system at a given time. The S movie segments are grouped by different movies, and listed in the sequential order for each movie.

The *watching matrix* $\boldsymbol{W}$ represents which peer is watching which movie segment, with the element defined as a binary indicator

$$w_{ij} = \begin{cases} 1 & \text{if peer } j \text{ is watching video segment } i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The *cached matrix* $\boldsymbol{C}$ characterizes the cached content of each peer, with the element defined as

$$c_{ij} = \begin{cases} 1 & \text{if peer } j \text{ has video segment } i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

An *allocation matrix* $\boldsymbol{A}$ contains the solutions, with the element $0 \leq a_{ij} \leq 1$. a_{ij} represents the percentage of the upload bandwidth that peer j allocates to movie segment i, and $a_{ij}u_j$ is the amount of such bandwidth allocated, where u_j is the upload capacity of peer j. We assume all such global information ($\boldsymbol{W}$, $\boldsymbol{C}$) is known for now in this section.

The bandwidth demand for a video segment i is composed of the required stream-

ing rate of all peers watching it, if the peer does not have the segment in its local cache, which can be computed as $D_i = r_i^{\text{seg}} \sum_{j=1}^{N} (w_{ij} - c_{ij})^+$, where r_i^{seg} is the playback rate of segment i. The supply B_i is computed as $B_i = \sum_{j=1}^{N} a_{ij} u_j$. Similar to Eqn. (3.1), we have

$$SBC = \sum_{i=1}^{S} (D_i - B_i)^+. \tag{3.4}$$

An important constraint is that a peer cannot be assigned to upload any video segments it does not have in its local cache, i.e., $\forall i, j, a_{ij} = 0$, if $c_{ij} = 0$. Moreover, for each peer, the sum of such bandwidth allocation percentages should be no greater than 1, i.e., $\forall j, \sum_i^S a_{ij} \leq 1$. In summary, the optimization problem is formulated as

$$\begin{aligned} \textbf{Min} \quad & \sum_{i=1}^{S} (r_i^{\text{seg}} \sum_{j=1}^{N} (w_{ij} - c_{ij})^+ - \sum_{j=1}^{N} a_{ij} u_j)^+ \\ \textbf{s.t.} \quad & \forall i,\ j,\ a_{ij} = 0,\ \text{if } c_{ij} = 0; \\ & \forall i,\ j,\ 0 \leq a_{ij} \leq 1; \\ & \forall j,\ \sum_{i}^{S} a_{ij} \leq 1. \end{aligned} \tag{3.5}$$

However, such an objective function for SBC exhibits a nonlinear programming problem, which is very hard to solve. Therefore, we convert the formulation into a tractable problem by introducing a new $S \times 1$ vector $\boldsymbol{U}^s$, with the element u_i^s defined as the amount of bandwidth allocated to movie segment i from the server, and the objective function is converted to minimizing the total server bandwidth consumption $\sum_{i=1}^{S} u_i^s$, with the constraint that the demand of each segment is no greater than the supply from the server and other peers, i.e., $\forall i,\ r_i^{\text{seg}} \sum_{j=1}^{N} (w_{ij} - c_{ij})^+ \leq \sum_{j=1}^{N} u_j a_{ij} +$

u_i^s. Moreover, u_i^s is a non-zero real number. The problem can then be formulated as

$$\begin{aligned}
\textbf{Min} \quad & \sum_{i=1}^{S} u_i^s \\
\textbf{s.t.} \quad & \forall i,j,\ a_{ij} = 0,\ \text{if}\ \ c_{ij} = 0; \\
& \forall i,j,\ 0 \le a_{ij} \le 1\ ; \\
& \forall i,\ r_i^{\text{seg}} \sum_{j=1}^{N} (w_{ij} - c_{ij})^+ \le \sum_{j=1}^{N} u_j a_{ij} + u_i^s; \\
& \forall i,\ u_i^s \ge 0\ ; \\
& \forall j,\ \sum_{i}^{S} a_{ij} \le 1,
\end{aligned} \tag{3.6}$$

with all a_{ij} and u_i^s as the unknown variables. Now the formulation exhibits a linear programming problem and thus is solvable in polynomial time in a centralized manner. Although the optimization solutions are difficult to obtain for large-scale systems in real time, they do provide guaranteed performance bounds for different scenarios, which we refer to as **Instance Bound** or **Bound III**. Moreover, as the optimization is independent from any specific bandwidth allocation algorithms (not following the *Bandwidth Helping Principles*), we expect that such a solution will lead to a server bandwidth consumption even lower than Bound II, which will be verified in Section 3.5.

3.3.4 A Summary of the Performance Bounds

Finally, we summarize the three bounds we have derived in this section. Bound I gives the best expected server bandwidth consumption by assuming that all upload bandwidth from peers is fully utilized, and thus there is absolutely no content bottleneck. Bound II takes the content bottleneck into account, where the bandwidth allocation follows the *Bandwidth Helping Principles* and the cache replacement follows FIFO or passive caching, and produces statistical performance bounds as well. As passive caching removes the content bottleneck, Bound II with passive caching overlaps with Bound I in most cases. At last, Bound III captures the instantaneous minimum server bandwidth consumption for the best bandwidth allocation at any given time, and thus provides tight lower bounds for the system along the time line.

3.4 Heuristic Algorithms

To achieve the analytical bounds derived in the previous section, we design heuristic caching and bandwidth allocation strategies. These two strategies have to coordinate with each other seamlessly, as cache replacement also affects segment availability (i.e., the *cached matrix* $\boldsymbol{C}$) and thus serves as the basis for bandwidth allocation.

In our system, HD movies are often partially stored in the local cache of some peers, so movie is not the suitable level for bandwidth allocation. An intuitive method is to migrate the "water-leveling" strategy from the movie level to the segment level for cache replacement [69, 70]. When removing a segment from the local cache of a peer, one can select the segment that is best globally provisioned, i.e., with the smallest global demand-to-supply ratio in terms of bandwidth. We refer to such a strategy as *Best-Globally-Provisioned-First* (BGPF), which is representative among a variety of LRU and LFU strategies. For the corresponding bandwidth allocation, *Perfect-Fair-Sharing* (PFS) is usually adopted by P2P systems due to its simplicity, where all the upload connections equally share the upload bandwidth of a peer [38, 39, 72, 73]. At the segment level, we can let each segment in the local cache of a peer have an upload connection. As PFS-BGPF takes the global demand and supply into consideration, it should demonstrate the best performance. The drawbacks of PFS-BGPF are similar to our optimization solution. It is challenging to collect and calculate the demand-to-supply ratio at the segment level in real time, as it can be highly dynamic. However, we still use it as the "semi-practical" benchmark in our simulation.

A different approach is to take advantage of the temporal relationship of viewers in the same channel, which is referred to as "stratification" or "chain-based" approaches [57, 62, 74, 75]. Viewers at later playback points are actually earlier/older viewers as they arrived earlier into a channel. They are likely to have watched all the segments before the segments they are currently watching, and thus are able to serve any viewers following them. In contrast, later/newer viewers can never help preceding viewers as they have no interested segments. In a chain-based allocation, each viewer only uploads to its closest followers in terms of the playback point within the channel, as to balance the upload workload of all the viewers in the channel. Figure 3.3 shows a simple example of the strict in-order chain-based allocation for a single channel with playback rate 500 Kbps, when the overlay is perfectly maintained and known to all the viewers. There are 4 viewers in the channel. `Viewer` 1 arrives early in the channel, and now is watching the last segment of the movie, `Seg` 20, followed by

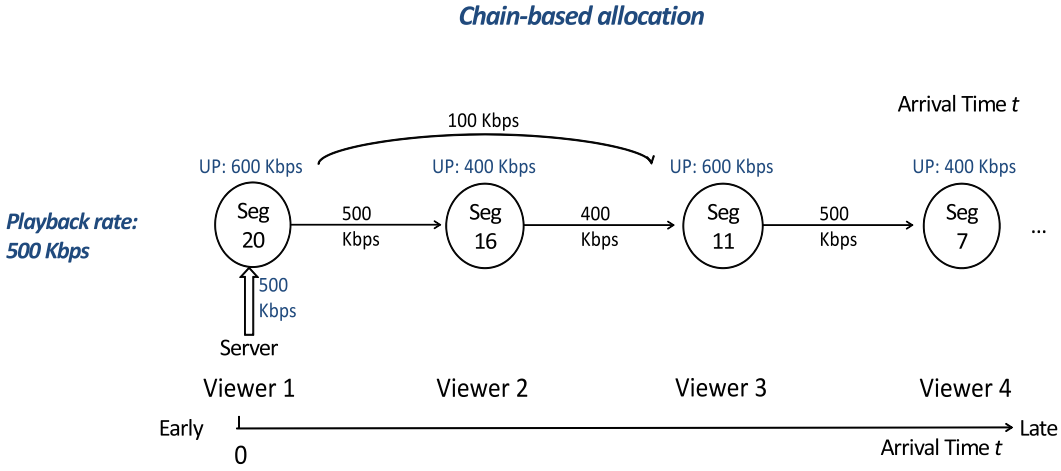


Figure 3.3: The strict in-order chain-based allocation for a channel with playback rate = 500 Kbps.

`Viewer` 2, `Viewer` 3 and `Viewer` 4. The upload capacity of each viewer is marked as "UP: xxx Kbps". The allocation starts from the earliest viewer, i.e., `Viewer` 1. It will allocate 500 Kbps towards `Viewer` 2, as to satisfy its 500-Kbps streaming demand. There will be no content bottleneck as `Viewer` 1 has cached all the segments before `Seg` 20, in which `Viewer` 2 is interested. As `Viewer` 1 has 100-Kbps upload bandwidth left, it allocates such an amount of bandwidth towards `Viewer` 3. All the viewers perform the similar allocation until all the streaming demands are satisfied or no viewers have available upload bandwidth. At last, the server compensates the bandwidth deficit of each viewer to ensure that all the viewers are satisfied, i.e., 500 Kbps for `Viewer` 1 in this case. As to multi-channel allocation [2, 69], the peer bandwidth is usually allocated first within the channel they are watching similar to such a chain-based algorithm. Help is offered to other channels only if the viewers still have extra upload bandwidth after the single chain allocation on their own channel, e.g., `Viewer` 3 and `Viewer` 4 in the example. Through these approaches, it is demonstrated that peer upload bandwidth can be effectively utilized for a single movie [57], and multi-channel systems with homogeneous and moderate playback quality [2].

The principle of our strategy is simple. As derived in Bound II, the number of helpers is statistically in proportion to that of viewers for HD channels, so we do not

explicitly balance the demand and supply at the movie level. At the segment level, instead of using water-leveling approaches, we adopt a chain-based structure, and try to make the peer upload bandwidth as *transferable* as possible. For instance, as early viewers are always able to help late ones in the same channel, when allocating the bandwidth of helpers, we let them upload to the early HD viewers first. Later, the bandwidth of these early viewers can be allocated to following viewers, which *transfers* the bandwidth from the helpers to all the viewers.

3.4.1 Caching Strategies

Considering that early HD viewers are requesting late segments, to meet the requests of such early HD viewers, it is desirable for HD helpers to store the last few segments rather than the beginning part of a movie. Therefore, at the segment level, a simple and natural FIFO replacement strategy is adopted, which pushes out early segments first. At the movie level, besides the simple FIFO cache replacement, we can also let SD viewers keep the HD segments that have been watched before in order to make sufficient HD replicas in the system, i.e., passive caching. Recall that the peer cache can only store 1 HD movie or 2 SD movies at most, and the movie being watched must be stored completely to facilitate bandwidth allocation within the viewing channel. As a result, each HD helper will always have the second-half segments of HD movies. If the helper starts to watch another HD movie, it will allocate its bandwidth within the HD channel first and thus may no longer serve as a helper, rendering such pre-watched HD segments useless. In this case, all the segments of the previous movies will be removed to make space for the new HD segments, where SD segments are still pushed out first. The details of such a passive-caching strategy at the movie level is explained in Algorithm 1. Note that such a peer cache management is easy to implement and requires no centralized coordination. In the future, if the peer cache is greater than 1 GB, i.e., many movies can be accommodated in the local cache, a peer may decide which movie to remove based on the demand-to-supply ratio at the movie level, following the water-leveling approaches in [2, 35, 36]. This will require a light-weight coordination of centralized tracker/server(s), as claimed in these studies.

Figure 3.4 illustrates an example of our passive caching. (a) A peer is caching two SD movies, `SD 1` and `SD 2`, at the beginning. (b) Then it starts to watch an HD movie, `HD 10`. As the HD movie will occupy all the cache space, the two SD movies will be marked to remove (with red color). (c, d) After the cache is filled with `HD 10`

Algorithm 1 Movie-level passive caching at a peer.

Require: A peer starts to watch a movie mov, and denote all the movies previously stored in the local cache as $\mathbb{M} = \{mov_1, mov_2, ..., mov_m\}$.

Ensure: A movie list $\mathbb{M}_{\mathtt{rv}}$, in which all the movies will be marked to remove from $\mathbb{M}$, and enough space will allow the new movie mov to be added to the local cache $\mathbb{M}$.

set the watch time of movie mov to the current time
sort all the movies in $\mathbb{M}$ based on the watch time wt, in the ascending order
if mov is an SD movie **then**
  if $mov \in \mathbb{M}$ **then**
    return $\mathbb{M}_{\mathtt{rv}} = \emptyset$
  else
    find all SD movies from $\mathbb{M}$, denote this SD movie list as $\mathbb{M}_S$
    if $\mathbb{M}_S \neq \emptyset$ **then**
      add the SD movie from $\mathbb{M}_S$ with the earliest watch time wt into $\mathbb{M}_{\mathtt{rv}}$
    else
      add any movie from $\mathbb{M}$ to $\mathbb{M}_{\mathtt{rv}}$
    end if
    return $\mathbb{M}_{\mathtt{rv}}$
  end if
else
  {mov is an HD movie}
  add all the movies from $\mathbb{M}$ to $\mathbb{M}_{\mathtt{rv}}$
  return $\mathbb{M}_{\mathtt{rv}}$
end if

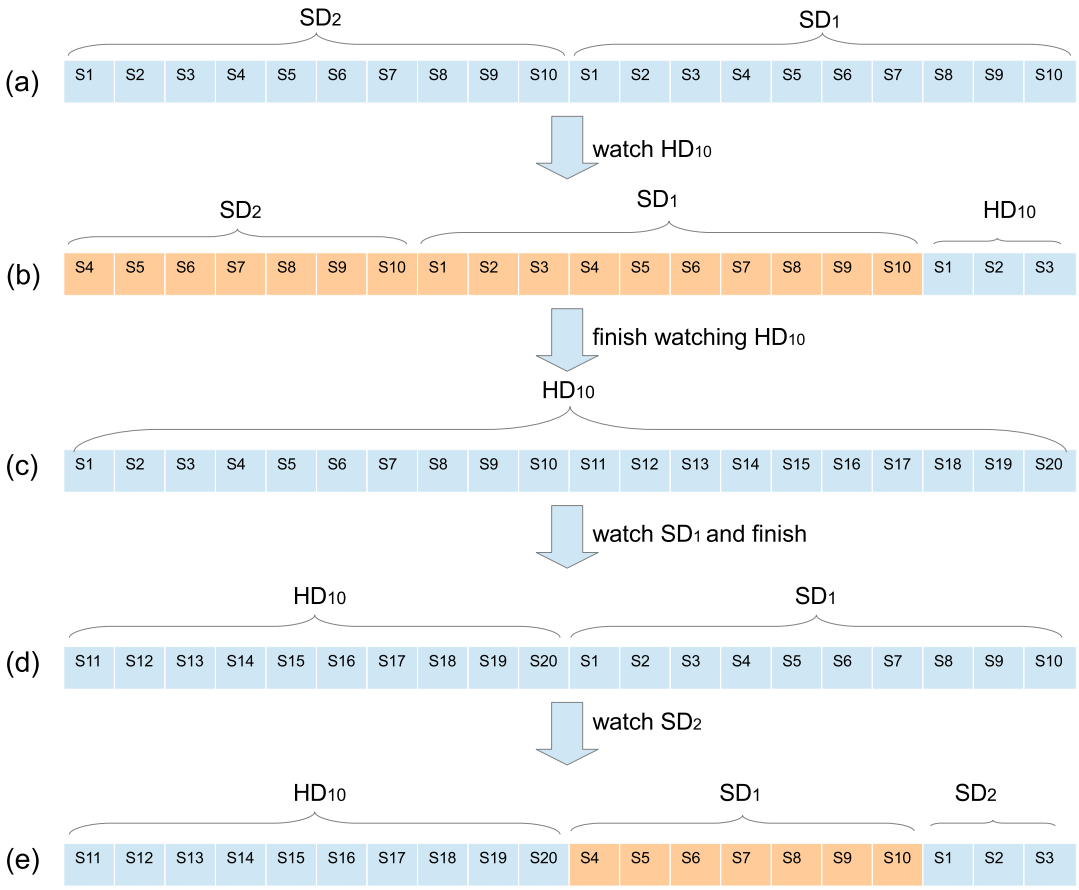


Figure 3.4: An example of passive caching.

segments, it starts to watch SD 1 again. As SD 1 is the movie being watched, all the segments will be cached, and the first 10 segments of HD 10 will be pushed out. (e) At last, the peer transitions to watch an SD movie, SD 2. This time, HD segments will be kept, and the SD movie that is not being watched, SD 1, will be replaced by the current movie, SD 2.

Remarks: Such a passive-caching strategy firstly guarantees that any movie being watched will be cached entirely, allowing the chain-based bandwidth allocation within a single channel. Next, HD segments will be kept as many as possible, bringing more candidate HD helpers. At last, keeping the second half HD segments will benefit allocating the bandwidth of HD helpers to viewers, which will be explained in the following section.

3.4.2 Bandwidth Allocation

Given the late HD segments cached by helpers, we need to find a way to allocate the bandwidth from helpers to viewers to transfer a portion of the available segments, and thus the upload bandwidth at the same time. Inappropriate allocation will result in *nontransferable*, i.e., wasted bandwidth. For instance, if the bandwidth of these helpers is allocated to late viewers in an HD channel, early viewers may suffer from insufficient supply, and the bandwidth of those late viewers will never be used by early viewers, as late viewers have no needed content. In this case, the content bottleneck occurs and the bandwidth is wasted.

We propose an inter-chain allocation algorithm to allocate the bandwidth of the helpers towards the viewers of an HD channel, such that the bandwidth wastage is reduced as much as possible. After SD viewers perform the inner-chain bandwidth allocation following the strategy shown in Fig. 3.3, the inter-chain allocation is invoked. First, we find all the helpers that satisfy these conditions: 1) it is watching an SD movie and has extra upload bandwidth available; 2) it has cached the needed HD segments already. After that, these helpers are sorted in the ascending order based on the portion of HD segments they have. Helpers with a small portion of HD segments only have the last few segments, and thus can only support a small range of early viewers. To take this factor into account, such helpers will make decisions before others on selecting the HD viewers. The viewers are also sorted based on their arrival time in the channel chain, where early viewers will be served first until the bandwidth supply meets the streaming demand. The advantage is that the bandwidth allocated

to early viewers can be then *transferred* to following ones. The details are listed in Algorithm 2.

Algorithm 2 Inter-chain bandwidth allocation.

Require: There are m viewers as $\mathbb{V} = \{v_1, v_2, ..., v_m\}$ in an HD channel chain Ch, and set the current assigned streaming bandwidth d_i^v to 0 for each viewer v_i.

Ensure: An inter-chain allocation from all helpers to viewers.

find all helpers $\mathbb{H} = \{h_1, h_2, ..., h_n\}$ who have cached part of the HD movie already, and sort them in the ascending order based on the number of the cached HD video segments they have
set the current available upload bandwidth u_i^h of all helpers to its available upload bandwidth after it has contributed to its own channel chain
$i \leftarrow 1$ and $j \leftarrow 1$
while $i \leq n$ **do**
  while $j \leq m$ **do**
    if helper h_i has cached the segment that v_j is watching and $d_j^v < r_H$ **then**
      $B_{needed} \leftarrow r_H - d_j^v$
      if $u_i^h \leq B_{needed}$ **then**
        assign all h_i's upload bandwidth u_i^h to v_j
        $d_j^v \leftarrow d_j^v + u_i^h$
        $u_i^h \leftarrow 0$
      else
        assign part of h_i's upload bandwidth to v_j
        $d_v^j \leftarrow r_H$
        $u_i^h \leftarrow u_i^h - B_{needed}$
        $j \leftarrow j + 1$
      end if
    end if
  end while
  $i \leftarrow i + 1$
end while

After the inter-chain allocation, the traditional inner-chain bandwidth allocation algorithm will be invoked for this HD channel following Lemma 3 in [2]. Early viewers will use the upload bandwidth to support late viewers, where the priority is still given to the closer viewers in terms of the playback point. The server will compensate the bandwidth shortage, if no bandwidth or content is available from peers.

The inner-chain allocation algorithm for SD channels is slightly different from that for HD chains. First, it will be invoked before SD viewers start to help HD channels. Moreover, an SD viewer may watch a movie already cached, such that there is no need to download again. We let these peers support the earliest viewers

in that SD channel. The benefits are twofold: 1) the server bandwidth consumption for the earliest viewers in the original inner-chain allocation can be saved; 2) such bandwidth supporting the earliest viewers can be transferred to late viewers, who have a better chance of caching more HD segments. Through such an allocation strategy, the bandwidth of peers will eventually be *transferred* to where it is needed most, and thus the server bandwidth consumption are possibly minimized.

Figure 3.5 illustrates an example of the inter-chain and inner-chain bandwidth allocation process. Assume that the playback rates of the SD and HD movies are 500 Kbps and 1,000 Kbps, respectively, and the upload capacity of each peer varies from 400 Kbps to 800 Kbps. `Helper 1` and `2` from channel `SD 2` have cached several HD video segments, and are able to offer 100 Kbps. `Helper 3` and `4` from channel `SD 1` have cached a half of the HD video segments, and each can provide 600 Kbps. Since `Viewer 1` is watching `Segment 20` of the HD movie, all helpers can provide this segment. So `Helper 1`, `2` and `3` will assign all of their upload bandwidth to `Viewer 1`, and `Helper 4` only needs to provide 200 Kbps to meet `Viewer 1`'s remaining bandwidth demand. As `Helper 4` still has 400-Kbps extra bandwidth, it allocates all the remaining upload bandwidth to `Viewer 2`. After all the helpers have utilized their bandwidth, the earliest viewer (`Viewer 1`) locates the closest viewer that needs bandwidth (`Viewer 3`) and allocates all of its 600 Kbps. Following viewers will repeat the same process until they use up their bandwidth.

Further Discussions

If centralized, the time complexities of the inner and inter-chain allocation are $O(n)$ and $O(mn)$, respectively, where m is the number of helpers and n the viewers of a channel. For a more distributed implementation, the helpers and viewers of a particular channel can maintain the chaining overlay through the coordination of a tracker, as well as periodically exchanging gossip messages containing their viewing and caching information, as any typical P2P streaming systems. Peers make decisions locally based on the local knowledge of the inter-chain overlay. We will discuss these implementation issues with more details and simulation results in Chapter 4, where a more distributed version of our heuristic algorithms demonstrates comparable performance on server bandwidth consumption, while does not bring much control cost and overhead.

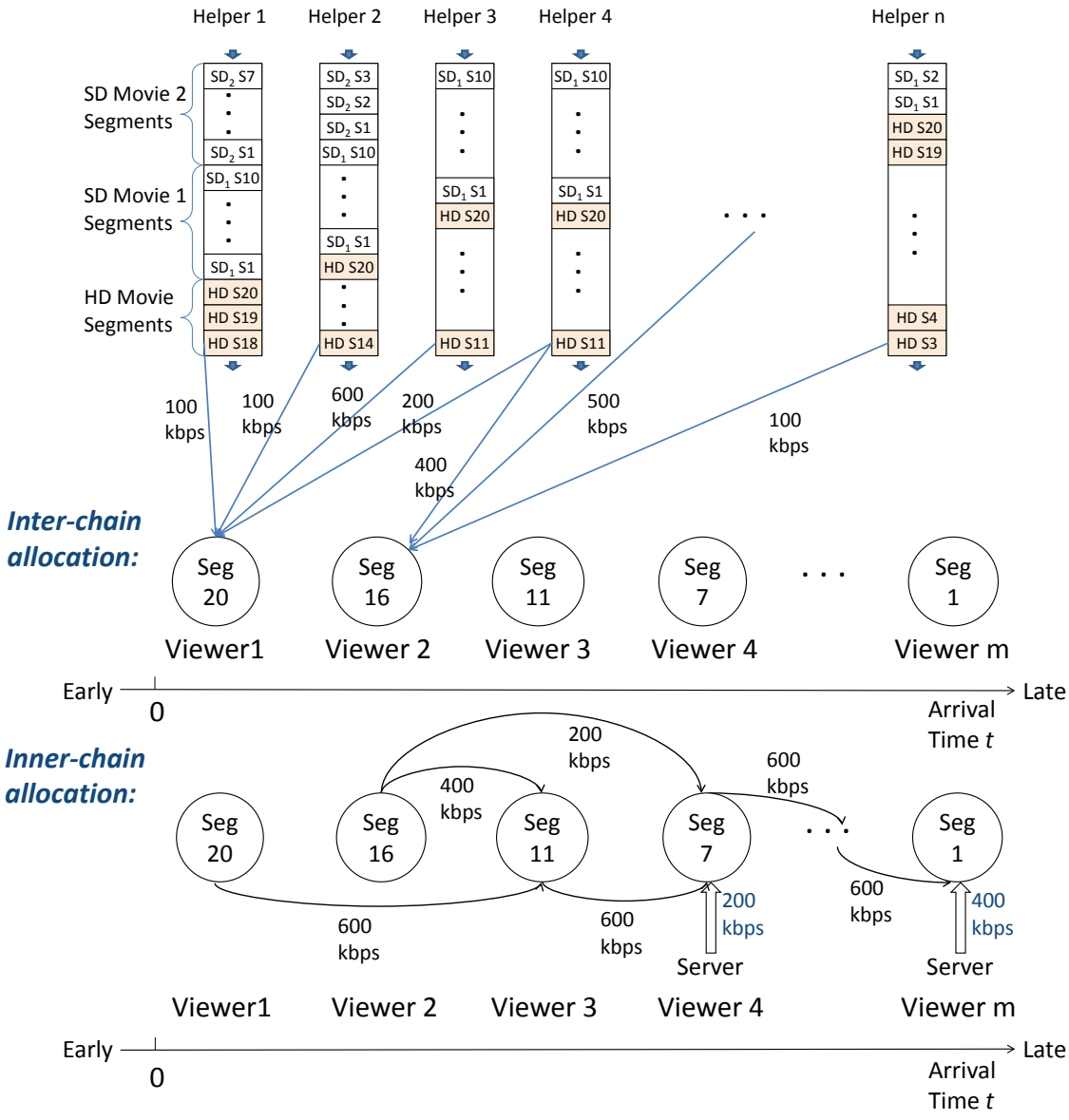


Figure 3.5: The inter and inner-chain bandwidth allocation.

Table 3.3: The distribution of peer upload capacity.

Upload Capacity	800 Kbps	500 Kbps	400 Kbps
Percentage	50%	30%	20%

3.5 Performance Evaluation

3.5.1 System Setup

We develop a `Java`-based event-driven simulator to emulate a multi-channel PA-VoD system, which is extended from the widely-adopted BitTorrent simulator [76, 77]. In the simulator, a series of discrete events are generated and processed, which include peer behavior, cache management, bandwidth allocation, and measurement events. Peer behavior events define the time points and the function of the join, movie transition, and departure behaviors of peers. Cache management events implement our caching strategies at each individual peer, and thus these events involve the arrival of a new segment and the removal of existing ones based on different caching strategies. Bandwidth allocation events are invoked at fixed time intervals or in real time, following our heuristic algorithms and the benchmark, `PFS-BGPF`. At last, in the measurement events we measure the system periodically to retrieve useful statistical information.

The playback rates of the SD and HD movies are 500 Kbps and $1,000$ Kbps, respectively. The peer upload capacity follows the distribution listed in Table 3.3, with an average of 630 Kbps. This setting is higher than the usual 530 Kbps in China [9], as the upload capacity of residential Internet accesses using DSL or Cable modem in North America nowadays is approaching 800 Kbps. There are 10 SD movies and 10 HD movies. Within each category, the popularity of the 10 movies follows a Zipf distribution with parameter 1, as such a kind of distribution is widely observed in multi-channel online video systems [1, 78, 79]. At the beginning of each simulation, we let peers (from $N = 1,000$ to $100,000$) join the system at the same time and never leave. Such a scale of peer population is set according to the statistics in real systems [41]. An HD movie is divided into 20 segments, and an SD movie into 10 segments. These numbers are adjustable, and we adopt this setting to achieve a balance between the simulation accuracy and the speed. With more segments (or smaller segments), the bandwidth allocation and caching is performed at a more fine-grained granularity, and thus will be more accurate. However, as we will show later,

dividing an HD movie into 20 segments can already result in a performance close to the heretical bounds. To investigate how fast the server bandwidth consumption converges, we set the initial state of the system to be different from its steady state. Each peer picks a random segment of a random movie to watch, i.e., a uniform distribution among both movies and segments initially. All segments ahead of the picked segment are assumed to be stored in advance in a peer's local cache to its limit.

We study three user viewing behaviors in the PA-VoD system. The transition matrices $\boldsymbol{P}^c$ between SD and HD categories are:

Case 1: $P^c = \begin{bmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \end{bmatrix}$; Case 2: $P^c = \begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix}$; and Case 3: $P^c = \begin{bmatrix} 0.5 & 0.5 \\ 0.8 & 0.2 \end{bmatrix}$.

The channel popularity in the steady state of the three cases is illustrated in Fig. 3.6. In Case 1 and Case 2, the percentage of peers watching SD (HD) channels in the steady state is 0.8 (0.2), and the average bandwidth demand per peer is thus 600 Kbps. In Case 3, the population percentage of SD (HD) channels becomes 0.615 (0.385), and the average bandwidth demand is 692.5 Kbps. Case 1 and Case 2 both imply the surplus mode, where the average peer upload capacity and the average bandwidth demand are 630 Kbps and 600 Kbps, respectively. Thus Bound I is calculated as zero. The two curves in the figure overlap with each other, which means the popularity of SD (`Channel` 1 to 10) and HD (`Channel` 11 to 20) channels in Case 1 and Case 2 are almost the same. However, different user behaviors are exhibited by these two cases. Compared with Case 2, peers in Case 1 are more active in transitioning to a different category. Case 3 reflects the system in the deficit mode, i.e., not able to survive by the peer-contributed bandwidth even in the steady state, as a large portion of peers are watching HD movies.

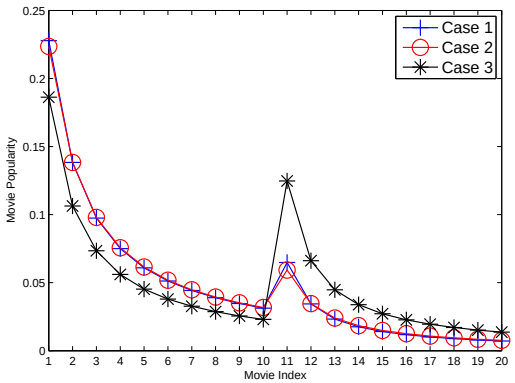


Figure 3.6: Movie popularity in the steady state of the three cases.

For each setting of these user behaviors, we evaluate a set of cache replacement and bandwidth allocation strategies, including our chain-based heuristics and PFS, with and without the corresponding passive-caching strategies, denoted by `Chain-Passive`, `Chain-FIFO`, `PFS-BGPF`, and `PFS-FIFO`, respectively. For small-scale system instances, we also use `lp_solve` for Java to obtain the optimal linear programming solution, which is corresponding to `Bound III-FIFO` or `Bound III-Passive`.

3.5.2 Helper Peer Statistics

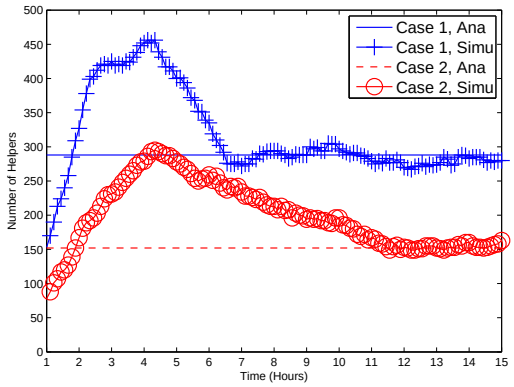
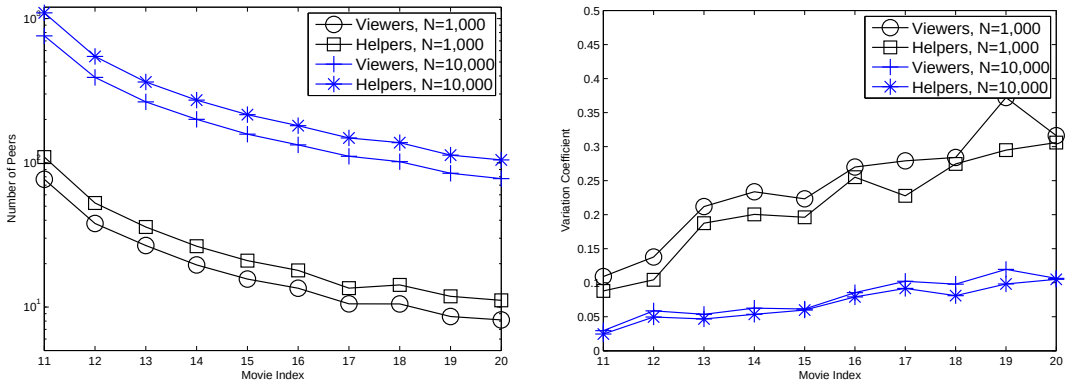


Figure 3.7: Number of helpers with two user viewing behaviors, FIFO.

In Case 1 (p^c_{HS}=0.8), peers tend to transition between HD and SD channels more frequently than Case 2 (p^c_{HS}=0.4), and thus bring more helpers for HD channels, as demonstrated by the analytical and simulation results in Fig. 3.7. Here the cache replacement is a simple FIFO. We thus gain the insights that it is desirable to let peers transition to SD channels immediately after they finish watching an HD movie. First, the overall bandwidth requirement will be reduced. Moreover, after watching an HD movie and transitioning to an SD movie, the peer will have the HD content in its local cache, and thus is able to act as a helper to support HD viewers using its extra bandwidth. As it is shown that the popularity and the transition probability of movies are controllable through a recommendation mechanism [78, 80], we can adopt a similar strategy to control p^c_{HS}. For example, the system can recommend related SD programs after a peer finishes watching an HD movie, or offer some form of rewards for transitioning to SD movies or staying online. Figure 3.7 also shows a slower ramp-up process for Case 2, as peers in Case 2 are reluctant to transition between categories.



(a) Distribution of HD viewers and helpers, Case 1, FIFO

(b) Variation coefficient of HD viewers and helpers, Case 1, FIFO

Figure 3.8: Helper peer distribution and variation coefficient.

Figure 3.8a plots the distribution of these helpers for $N = 1,000$ and $N = 10,000$ where we retrieve the statistics and calculate the mean values of HD channels in the steady state in Case 1. The simulation result shows that the number of helpers of an HD channel is always in proportion to the numbers of viewers, confirming the analytical results in Bound II. We refer to such a phenomenon as the *self-adaptivity* of the system. Intuitively, if there are more viewers in an HD channel, after they finish the movie, many will transition to SD channels with the cached HD content, and thus bring more helpers for the HD channel. The similar phenomenon is also observed in Fig. 3.7, where the number of HD helpers increases dramatically in the first four hours, as half of the peers are viewing HD movies at the beginning, and later become HD helpers. However, these channels demonstrate different levels of *self-adaptivity*, as shown by the coefficients of variation (the ratio of the standard deviation to the mean) in Fig. 3.8b. Either a channel or the entire system with a smaller population suffers from a higher coefficient of variation, which indicates that the viewer and helper population is more unstable. We will detail its influence on the system performance in Section 3.5.4.

3.5.3 System Evolution

The instantaneous server bandwidth consumption under the stationary scenarios is plotted in Fig. 3.9, with the peer population set to $N = 1,000$, as to avoid the long processing time of `lp_solve`. All of the strategies are assumed to be able to perform in real time in these scenarios.

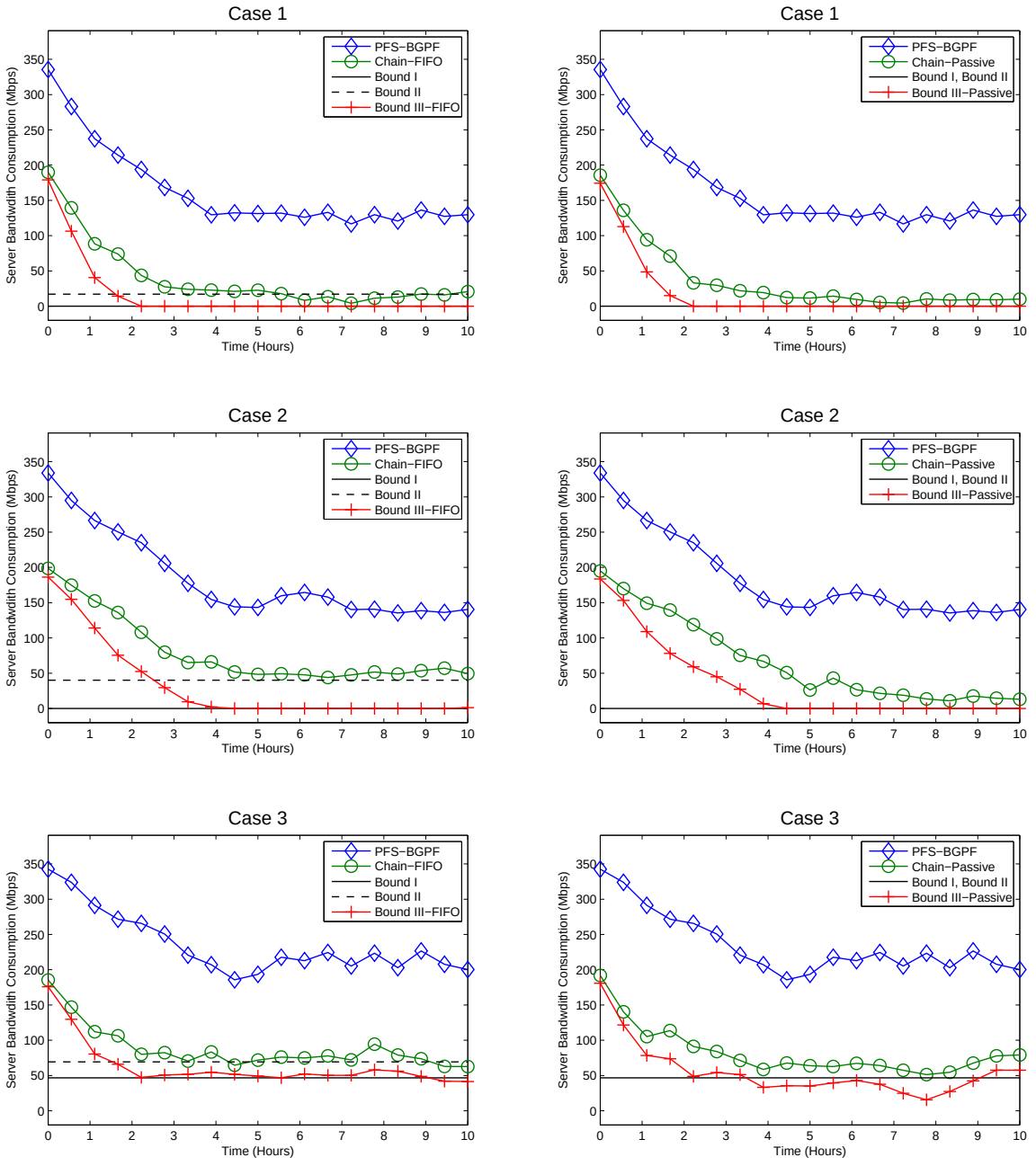


Figure 3.9: Server bandwidth consumption of the three cases in stationary scenarios.

Convergence Time

At the beginning of each simulation, HD viewers account for half of the overall peer population in the system due to the random pick of movies, which makes the system temporally staying in the deficit mode. After a short time (2 to 4 hours), the server bandwidth consumption decreases significantly. Recall that in Fig. 3.7, the number of helpers increases to a peak value in around 2 hours for Case 1, and 4 hours for Case 2, which is consistent to the convergence time here. Peers transition between HD and SD channels more frequently in Case 1, and thus distribute more HD content faster.

Server Bandwidth Consumption

In Case 1 and Case 2, the peer-contributed bandwidth exceeds the total bandwidth demand, so Bound I reaches zero. However, Bound II is different for these two cases, and a more active transition behavior between HD and SD channels (i.e., Case 1) leads to a lower Bound II. In Case 3, due to the large number of HD viewers, the system has bandwidth deficit even in the steady state, shown by the non-zero Bound I. Note that the chain-FIFO approach is statistically bounded by Bound II, and Bound I and Bound II overlap with each other in all cases when passive caching is adopted, as it removes the content bottleneck.

We find that the proposed chain-based heuristic algorithms perform generally well, and achieve lower server bandwidth consumption with passive caching as more helpers are present. Note that the total upload bandwidth of all peers is about 615 Mbps. The performance of the linear programming solution (Bound III) in these three cases demonstrates its optimality unsurprisingly, where the server bandwidth consumption can be even reduced to Bound I. The reason is that it attempts to find the best match between helpers and viewers, which does not necessarily follow the *Bandwidth Helping Principles*. Counter-intuitively, PFS-BGPF consumes much more server bandwidth than our chain-based heuristic algorithms in all cases. This is due to the small population of peers. Given $N = 1,000$ peers in the system, on average there will be 200 HD viewers, and the least popular HD channel only has a few viewers. The population of such channel is extremely unstable and it becomes even worse at the segment level, as each HD movie has 20 segments. Consider the case that peers have just removed the 10th segment of an HD movie as no peer is watching it. However, it is possible that those viewers are watching the 9th segment,

and will request for the following one very soon. As the 10th segment is removed, there is no chance for it to return to the peers in a short time without active caching. Consequently, it becomes difficult for peers to catch up with the dramatic demand change at the segment level. We believe this is an important insight when designing bandwidth allocation algorithms for PA-VoD systems.

3.5.4 System Scalability

We also investigate the scalability of the system with the user behavior following Case 2. We vary the number of peers, and measure the bandwidth efficiency η in the steady state of each algorithm, which is visualized in Fig. 3.11. The efficiency is calculated as the average value based on 350 system instances we retrieve during the simulation process. We find that such a system not only scales well, but also exhibits a better performance under larger peer populations, which conforms to the "smoothing effect" of the large population in real systems [81]. Such a large population can better demonstrate the expected performance of each strategy, while in contrast a small population of peers examines the robustness of the algorithms against the peer churning between different channels and segments, as the system is more likely to deviate from its steady state.

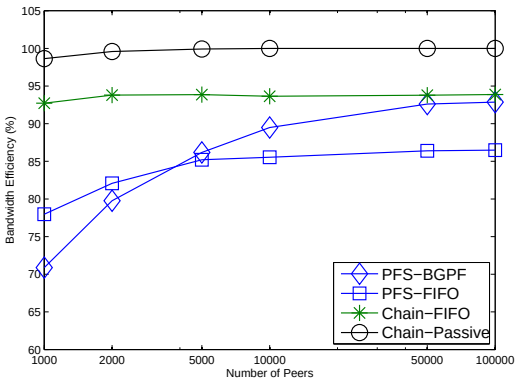


Figure 3.10: Bandwidth efficiency under different peer population, Case 2.

Once again our chain-based heuristic algorithms demonstrate the best performance, independent of the peer population, and passive caching helps further to reduce the server bandwidth consumption. The performance of PFS-BGPF is not comparable with our heuristic algorithms, and even worse than PFS-FIFO, in the small peer population cases. However, when the peer population increases, the system will become more stable, and PFS-BGPF will perform increasingly better. Given

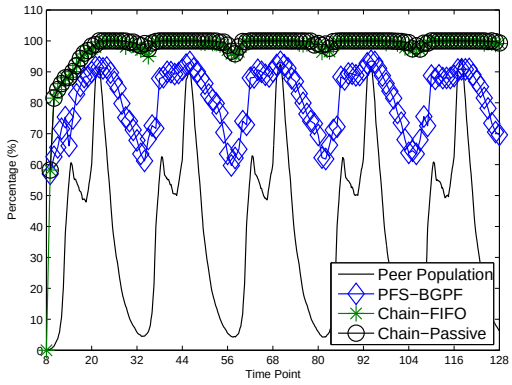


Figure 3.11: Peer population and bandwidth efficiency in diurnal arrival scenarios.

a sufficient number of peers in the system, PFS-BGPF will eventually demonstrate its capability of balancing the global demand and supply of each segment with the provision of global and local information in real time, which is close to the performance of our heuristic algorithms, as illustrated when $N = 100,000$.

3.5.5 Diurnal Arrival Scenarios

At last, we evaluate the performance of the algorithms in a more dynamic scenario. We still assume that the movie popularity is fixed. Measurement studies have observed that VoD users usually follow the diurnal arrival pattern, with 12:00 PM to 2:00 PM and 7:00 PM to 11:00 PM as peak hours [1, 16, 28, 79, 82]. To emulate such a phenomena, we generate $N = 10,000$ peers at the beginning of our simulation and set them as offline users. Then we randomly pick some of these offline users and let them join the system starting from 8:00 AM, following a non-homogeneous Poisson arrival process with time-dependent arrival rates. We carefully interpret the statistics from these measurement studies to set the arrival rate for each half an hour to regenerate a similar diurnal process, where the highest population of online users is reached at around 9:00 PM, and the second highest population at 2:00 PM, which is visualized as the percentage of online peers in Fig. 3.11. Once a peer finishes watching a movie, it will either leave the system with probability 50%, or stay online and pick another movie to watch. If the peer leaves the system, we assume it will keep the cached content and bring it back when returning to the system, which is a common practice in current PA-VoD systems. The transition behavior of peers follows Case 1 in this section.

Prolonged Convergence Time

We first observe that the convergence time is prolonged to around 24 hours, which is due to two reasons. First, we let peers start with empty local cache, which takes time to replenish, especially for HD content. Second, some helpers may leave the system. However, such helping bandwidth is only temporarily lost as these helpers will return later with the previously-cached video content. After all peers have cached some HD video content, the system converges to its steady state.

Effect of Online Population

PFS-BGPF clearly shows its vulnerability to peer churning, as its bandwidth efficiency varies significantly between valleys and peaks of the peer population. When there is a small population of online peers (i.e., the valleys), low bandwidth efficiency is observed as the system is unstable and beyond the control of PFS-BGPF. This conforms to the insights we have from Section 3.5.4. For the proposed algorithms, both Chain-Passive and Chain-FIFO strategies demonstrate a better robustness. In Case 1, peers are active enough to transition between different movie categories, so there is little difference between the performance of Chain-Passive and Chain-FIFO.

3.6 Conclusions

In this chapter, we study the resource imbalance problem in PA-VoD systems in stationary scenarios with heterogeneous channel playback rates, where peers from SD channels help HD channels in terms of both upload bandwidth and cached content. A generic modeling framework is proposed and explained, which is able to model a variety of caching strategies under different scenarios. Extending the modeling framework, our detailed models for stationary scenarios provide statistical performance bounds for FIFO and passive-caching strategies in terms of server bandwidth consumption with given transition behaviors in the steady state, and instance bounds with any snapshot of a system at any given time. Moreover, simple heuristic algorithms are proposed to efficiently allocate the peer upload bandwidth within and across channels, by making it as "transferable" as possible. Simulation results verify that our heuristic algorithms reduced the server bandwidth consumption to the preferable theoretical lower bounds.

Chapter 4

Non-stationary Scenarios and Practical Methods

In this chapter, we model non-stationary scenarios using our modeling framework and discuss practical methods of implementing our heuristic algorithms. Consider a new HD movie released into the system, and peers enter the channel in a flash-crowd manner. Different from stationary scenarios where the expected server bandwidth consumption at the steady state is evaluated, in non-stationary scenarios, we aim at minimizing the aggregate server bandwidth consumption for a time period around the release time of the new movie, i.e, the releasing cost of this new movie. We investigate the optimal caching strategies during such a "monitoring" period, and use mixed integer linear programming (MILP) to find the optimal or suboptimal solutions. From our model and simulation results, passive caching is demonstrated inefficient in such scenarios. To introduce sufficient helpers in time, the server(s) needs to actively inject the HD content to peers with available bandwidth, i.e., active caching. Although pushing such content to peers consumes server bandwidth as well in the short term, once the content is fetched by these peers, they can stay in the system and act as long-term helpers. Moreover, we propose and discuss practical methods that are useful in real systems to facilitate the cross-channel cooperation between peers, in a simple, fast, and distributed manner.

This chapter is organized as follows. Section 4.1 briefly reviews the background and preliminaries. Section 4.2 presents the detailed model for non-stationary scenarios, where the optimal caching strategies are derived. We discuss the corresponding heuristic bandwidth allocation strategies in Section 4.3. These strategies are then

extensively evaluated in Section 4.4. The practical methods for implementation and incentive issues are discussed in Section 4.5. Section 4.6 concludes the chapter.

4.1 Introduction

The resource imbalance problem in releasing new HD movies is due to two factors. First, HD channels usually require more bandwidth supply and cache space, which exceeds the capacity of the participating peers; and second, due to the sudden increase of the viewer population of newly released movies, a huge number of peers are watching and few contributing, which imposes a heavy bandwidth consumption at the server after the movie is released.

Again, to reduce the server bandwidth consumption under such non-stationary scenarios, it is desirable to utilize the surplus bandwidth of well-provisioned channels to help compensate the bandwidth deficit of other channels, i.e., view-upload decoupling (VUD). Concerning VUD in PA-VoD systems, existing work relies on relaxed assumptions to simplify the analysis. The stationary model described in the previous chapter is widely used to describe user behaviors, in which users form a closed Jackson queuing network and transition between channels (queues) with fixed probabilities, i.e., the popularity of each channel will never change [2, 36, 38]. Although these models are applicable to a sufficiently stable system within a short period (e.g., several hours), they miss the dynamic nature of PA-VoD systems, and fail to reflect the influence of the evolution of movie popularity, especially when a new movie is released.

We now consider a PA-VoD system with the coexistence of SD and HD channels under non-stationary scenarios, i.e., the arrival rate of users to a channel may change over time (a *non-homogeneous arrival process*). We design caching strategies to determine what content to remove or fetch at a peer, and the corresponding bandwidth allocation to select peers to upload to or download from. For such non-stationary scenarios, we aim at answering the following questions.

- What is the demand-vs-supply relationship of an HD channel with non-homogeneous arrival processes? Can we mathematically characterize it?
- How to find the best caching strategy when a new movie is released and causing flash crowds?

- How to design a bandwidth allocation strategy which seamlessly coordinates with the best caching strategy above, so that the cost of introducing new helpers is acceptable without compromising the download performance of peers?

The main contributions of our work in this chapter are summarized as follows.

- Based on the modeling framework, once again we build a detailed model to derive the server bandwidth consumption with passive or active-caching strategies under non-stationary scenarios, where pure passive caching is not sufficient to handle such user dynamics.

- We formulate finding the best caching strategies in non-stationary scenarios as a mixed integer linear programming (MILP) problem. Despite the intractability of general MILP problems, many instances of our optimization problems are actually solvable and generate optimal solutions.

- We adopt a combination of bandwidth allocation strategies to fit the optimal caching solution obtained from the optimization: chain-based allocation plus substream multiplexing. These strategies guarantee a lower cost of active caching without compromising the bandwidth utilization, when many HD movies are partially cached by helpers due to the space limit of the peer local cache.

- In our further discussions, we describe how our allocation algorithms can be implemented in real systems through *distributed approximate chaining* and *passive substreaming*, with acceptable control cost and overhead. Besides, we explain how existing approaches from real systems are efficient to encourage SD viewers to help HD viewers, with simple modifications. These practical methods bridge the gap between our theoretical analysis in this thesis and real-world implementations.

4.2 The Detailed Model for Non-stationary Scenarios

In this section, we customize our modeling framework to model a multi-channel PA-VoD system with the coexistence of SD and HD channels under non-stationary scenarios. The system settings and assumptions are similar to stationary scenarios, so

Table 4.1: Notations of the detailed model, non-stationary scenarios.

Symbol	Definition
$\bar{u}$	The average upload capacity of all peers
$x_i(k), y_i(k)$	The number of viewers, helpers of channel i in round k
$\lambda_i(k), \theta_i(k)$	The arrival, departure rate of the viewers of channel i in round k
$\eta_i(k), \gamma_i(k)$	The arrival, departure rate of the helpers of channel i in round k
$\omega_i(k)$	The number of introduced active cachers of channel i in round k
t_i^{a}	The residence time of queue a of channel i
$p_i^{\mathrm{a}\to\mathrm{b}}$	The transition probability from queue a to b of channel i
$p_i^{\mathrm{a}\to 0}$	The probability of leaving channel i from queue a
$\bar{d}_i^{\mathrm{view}}$	The average viewer bandwidth deficit of channel i
$\bar{s}_i^{\mathrm{help}}$	The average helper bandwidth contribution of channel i
r_S, r_H	The playback rate of an SD, HD movie
T	The time duration of each movie
F	The size of the peer cache
p_i^{quit}	The probability of leaving the VoD system from channel i
p_{ab}^c	The transition probability from category a to b
$u^{\max}$	The capacity of the server(s)
$SBC_i(k)$	The server bandwidth consumption for channel i in round k
$\overline{SBC}_{i,t_s\to t_e}$	The average server bandwidth consumption for channel i over the period t_s to t_e
D, S	The total bandwidth demand, supply of peers
R	The total number of segments of an HD movie
f	The byte length of the video fetched by an active cacher
u_i^{c}	The bandwidth allocated to channel i by the server(s)
N_{sub}	The number of substreams for an HD movie

we briefly review them here. We also list and explain the terms and notations used in this Chapter in Table 4.1.

4.2.1 Model Customization and Assumptions

We assume two categories of the video channels in the system, SD channels with playback rate r_S, and HD channels with playback rate r_H. For simplicity, we set $r_H = 2r_S$, a typical case with $r_H = 1,000$ Kbps and $r_S = 500$ Kbps. The average peer upload bandwidth is assumed to be $r_S < \bar{u} < r_H$ as observed in real systems [1,3]. We let HD viewers allocate their bandwidth within their viewing channels only, as these channels already suffer from bandwidth deficit. SD viewers will perform the

bandwidth allocation within their viewing channels first, and then use the surplus bandwidth to help HD channels, i.e., following the *Bandwidth Helping Principles*.

Each peer contributes some local cache space of a finite size 1 GB (e.g., in PPLive [28]). The peer local cache can accommodate either one complete HD movie or two SD ones. The time duration of each movie is T. The movie file is also equally divided into small segments, and each segment has the same byte length (e.g., 20 segments for an HD movie and 10 for SD). Each SD viewer will use part of the cache space to preserve partially the HD movie previously watched. The other half is reserved to store the SD movie that is being watched.

When watching a movie, each peer starts from the beginning of the movie and watches in its entirety for time T, i.e., VCR operations are not considered. After that, the peer can select another movie to watch, or leave the VoD system with probability p_i^{quit}. We also assume that peers stream exactly at the playback rate of a movie, following [2, 62–64].

To model the user behaviors across channels, we assume two transition matrices as in Chapter 3: $\boldsymbol{P}^c$ at the category level and $\boldsymbol{P}^m$ at the movie level. $\boldsymbol{P}^c$ is a two-by-two matrix containing the transition probabilities between SD and HD categories, i.e., p_{SS}^c, p_{SH}^c, p_{HH}^c and p_{HS}^c, where S represents SD movies and H represents HD movies.

There is a server with capacity $u^{\max}$, which is able to help any peer at any time if the bandwidth consumption does not exceed $u^{\max}$. The instantaneous server bandwidth consumption (SBC) at time t_k is denoted as $SBC(t_k)$. In order to evaluate different caching strategies in non-stationary scenarios, we define the average server bandwidth consumption over a time duration of t_s to t_e as $\overline{SBC}_{t_s \to t_e}$, which is the aggregate data volume that the server injects into the system over the duration of the period $t_e - t_s$. $\overline{SBC}_{t_s \to t_e}$ quantifies the cost to run the server from time t_s to t_e. For instance, if a new movie is released at a time close to t_s and the system becomes stable at time t_e, $\overline{SBC}_{t_s \to t_e}$ can be interpreted as the cost of releasing the new movie.

With the assumptions above, we now customize all the critical parameters in our modeling framework in Chapter 3 to build the detailed model. An SD viewer is considered as a helper if it transitions from an HD channel to an SD channel, as it has both extra bandwidth and the content of the previously watched HD movie, so $p_i^{\text{view}\to\text{help}} = (1 - p_i^{\text{quit}})p_{HS}^c$. Since a helper will preserve HD segments when it is watching SD movies due to passive caching, $p_i^{\text{help}\to\text{help}} = (1 - p_i^{\text{quit}})p_{SS}^c$. Each movie lasts for time T, and each peer watches a movie in its entirety, after which it leaves the viewer/helper queue, i.e., $t_i^{\text{view}} = t_i^{\text{help}} = T$. We assume that $\lambda_i(t_k)$ is known for

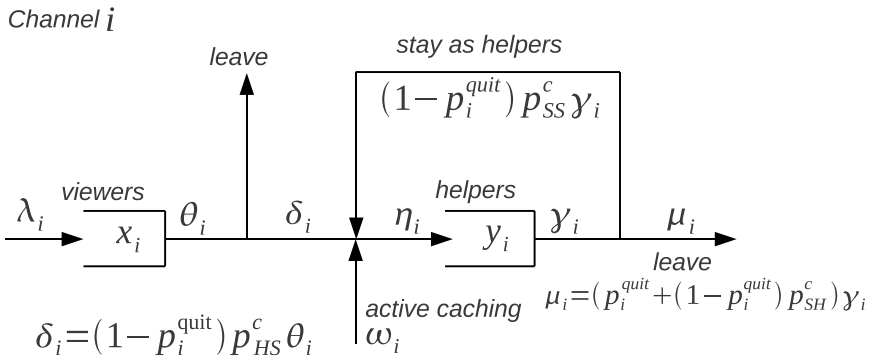


Figure 4.1: The detailed model under non-stationary scenarios.

any t_k in non-stationary scenarios. ω_i, specifically for the active-caching strategy, can either be given to evaluate the system performance, or be the unknown variables in an optimization problem minimizing certain server cost, e.g., bandwidth. The detailed model after such a customization is shown in Fig. 4.1.

An example of the system following the detailed model for an HD channel (`HD 2`) with passive and active caching is shown in Fig. 4.2. The arrows indicate the possible transitions of peers. If `Viewer 1` decides to watch an SD channel, `SD 5`, after it finishes `HD 2`, it becomes a helper of `HD 2`, i.e., through the passive caching process. However, a viewer may select another HD movie, or leave the VoD system, as illustrated by the two choices of `Viewer 2`. A helper (e.g., `Helper 1`) may also choose to stay if it always watches SD movies, leave the channel if transitioning to an HD channel, or leave the VoD system. Moreover, the server can actively introduce helpers through active caching, e.g., `Helper 2`. Note that HD movies are not necessarily cached in their entirety, due to the limited size of the peer cache.

4.2.2 Optimal Caching Strategies under Non-Stationary Scenarios

In a non-stationary scenario, the arrival rate of viewers will change over time. Therefore we adopt the average server bandwidth consumption $\overline{SBC}_{t_s \to t_e}$ over a *monitoring period* from t_s to t_e. t_s and t_e can be defined by VoD providers for different purposes. If the long-term performance is of the most concern, a longer monitoring period $t_e - t_s$ can be selected, otherwise a shorter duration is preferred. Moreover, in the case a new movie is released, t_s can be set to a time prior to the release time of the movie

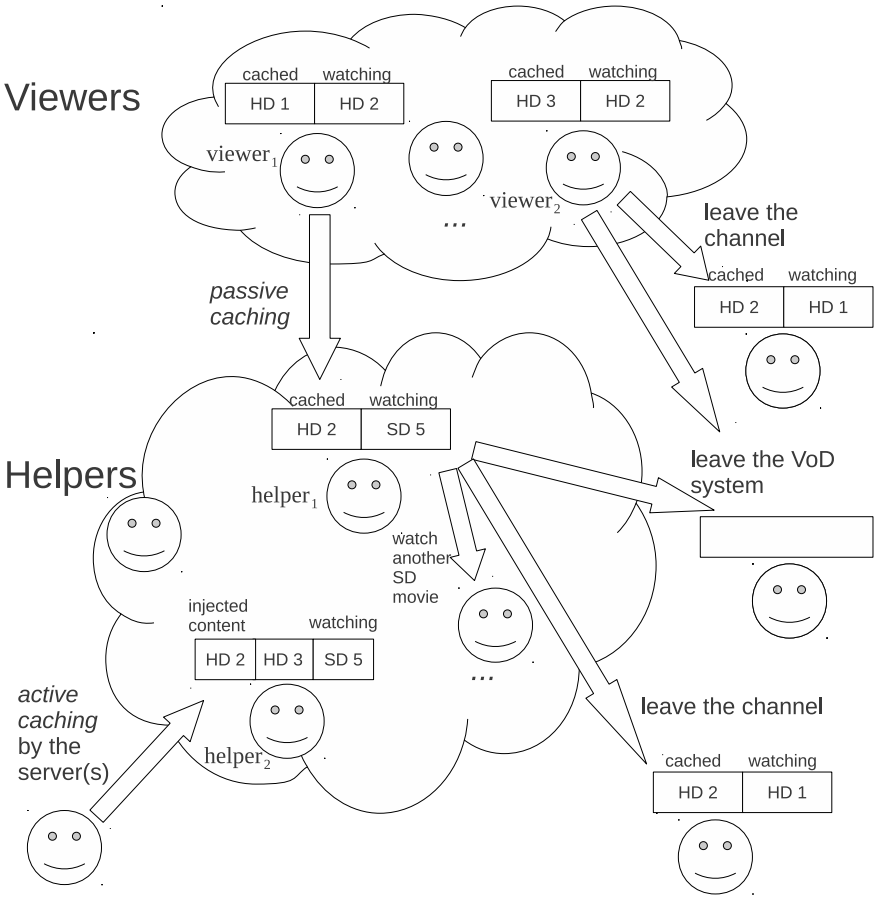


Figure 4.2: A PA-VoD system with passive/active caching.

while the movie content is time locked [2]. Such a "pre-release" active caching can reduce the tremendous server bandwidth consumption during the first several hours after the movie is released, when users start to watch the movie in flash crowds. This is especially useful when the server capacity is limited, i.e., with a small $u^{\max}$.

We discretize time into small time slots or rounds. The duration of each round is $\Delta t = T/R$, where R is the total number of segments of an HD movie. We set $t_e - t_s = KR\Delta t$, i.e., $K \times R$ rounds for the monitoring period. The numbers of the viewers and helpers during the k-th round after the start of the monitoring period are denoted as $x_i(k)$ and $y_i(k)$, respectively. Let $\lambda_i(k)$ denote the number of user arrivals, and $\theta_i(k)$ the departures at the end of (or during) round k. Also let $\delta_i(k)$ denote the number of viewers that turn into helpers, $\omega_i(k)$ the number of new helpers introduced by the server through active caching, $\eta_i(k)$ the number of all arrivals, and $\gamma_i(k)$ the number of departures of the helper queue, respectively. For the simplicity of analysis, we assume the arrival rate $\lambda_i(k)$ at any round k is known during the monitoring period, and aim at evaluating the system performance with a series of $\omega_i(k)$. In real systems, only the arrival rate in the past is known. To predict the arrival rate in the near future, prediction techniques can be used [83–85]. However, they are out of the scope of this thesis.

According to *Little's Law* for non-homogeneous arrivals,

$$\begin{aligned} x_i(k) &= \sum_{j=1}^{R} \lambda_i(k-j), \\ y_i(k) &= \sum_{j=1}^{R} \eta_i(k-j). \end{aligned} \tag{4.1}$$

As we assume the local cache of each peer can accommodate one complete copy of an HD movie (R segments), $k - R$ represents the earliest round in which the arrived peers still reside in the view/helper queue in round k. As a viewer/helper will stay in the viewer/helper queue for time T (or R rounds) and then leave the queue (a helper may return to the help queue with probability $p_i^{\text{help}\to\text{help}}$ after "leaving"), the departure rate in round k is equal to the arrival rate R rounds before for both queues,

so the following equations hold:

$$\begin{aligned} \delta_i(k) &= p_0\theta_i(k) = p_0\lambda_i(k-R), && (4.2)\\ \gamma_i(k) &= \eta_i(k-R) \\ &= \delta_i(k-R) + \omega_i(k-R) + p_1\gamma_i(k-R), && (4.3) \end{aligned}$$

where $p_0 = (1-p^{\text{quit}})p_{HS}^c$, and $p_1 = (1-p^{\text{quit}})p_{SS}^c$. At the start time t_s, i.e., $k = 1$, there are already viewers and helpers residing in the queuing network. They arrived at the system before the monitoring time, and the corresponding arrival rates also affect the number of viewers and helpers in the future. Therefore, we allow round number $k \leq 0$, and assume that $\lambda_i(k)$, $\delta_i(k)$ and $\gamma_i(k)$ for any $k \leq 0$ are known. The arrival rate of the helper queue at round k can be derived as

$$\begin{aligned} \eta_i(k) =& \delta_i(k) + \omega_i(k) + p_1\gamma_i(k) \\ =& p_1^{l+1}\gamma_i(k-lR) + p_1^l\delta_i(k-lR) \\ &+ \sum_{j=1}^{l} p_1^{l-j}\omega_i(k-(l-j)R) \\ &+ p_0\sum_{j=1}^{l} p_1^{l-j}\lambda_i(k-(l-j+1)R), \end{aligned} \qquad (4.4)$$

where $l = \lfloor (k-1)/R \rfloor + 1$ represents the batch number of time rounds, if we treat every R rounds from the start of the monitoring period as one batch.

The server bandwidth consumption at round k contains two components. First, if the total bandwidth deficit of the viewers exceeds the extra bandwidth of the helpers, the server has to compensate for the difference, which we refer to as the *compensation cost* for these viewers. Moreover, in order to introduce new active cachers, the server has to upload specific content to the cachers in advance, which also consumes its upload bandwidth. We refer to the latter as the *caching cost* for the active caching. Assume that each active cacher downloads the HD content of size f, the caching cost is calculated as $\omega_i(k)f/\Delta t$ during round k. Therefore, the server bandwidth consumption for channel i at round k is $SBC_i(k) = (\bar{d}_i^{\text{view}}x_i(k) - \bar{s}_i^{\text{help}}y_i(k))^+ + \omega_i(k)f/\Delta t$, and thus the average SBC for channel i over the monitoring

period $t_s \to t_e$ can be calculated as follows:

$$\begin{aligned}
&\overline{SBC}_{i,t_s \to t_e} \\
&= \frac{1}{KR} \sum_{k=1}^{KR} ((\bar{d}_i^{\text{view}} x_i(k) - \bar{s}_i^{\text{help}} y_i(k))^+ + \omega_i(k) f/\Delta t) \\
&= \frac{1}{KR} \sum_{k=1}^{KR} (\bar{d}_i^{\text{view}} \sum_{j=1}^{R} \lambda_i(k-j) - \bar{s}_i^{\text{help}} \sum_{j=1}^{R} \eta_i(k-j))^+ \\
&\quad + \sum_{k=1}^{KR} \frac{\omega_i(k) f}{KR\Delta t},
\end{aligned} \tag{4.5}$$

where $\bar{d}_i^{\text{view}} = r_H - \bar{u}$, $\bar{s}_i^{\text{help}} = \bar{u} - r_S$, and $\eta_i(k-j)$ can be obtained from Eqn. (4.4). Now we propose Theorem 3.

Theorem 3. *Under non-stationary scenarios, during a period from t_s to t_e, given the arrival rate of viewers $\lambda_i(k)$ and the number of active cachers $\omega_i(k)$ injected for an HD channel i at each round k, the average server bandwidth consumption for channel i over this period is bounded by Eqn (4.5).*

Remarks: Equation (4.5) represents the lowest cost to run the server from time t_s to t_e, if the caching strategy, i.e., $\omega_i(k)$ for any k, is given. When $\omega_i = 0$, it represents a special case as with passive caching only. Moreover, the equation quantitatively characterizes the tradeoff between injecting new content to active cachers and compensating the bandwidth deficit of the current viewers. A larger ω_i at a given time will lead to more helpers (a larger y_i) in the future and thus will reduce the compensation cost. However, it also results in a larger caching cost in terms of $\omega_i(k) f/\Delta t$. Thus we resort to optimization techniques to obtain the desired balance.

The objective of the optimization for a single HD channel i is to minimize the average SBC from time t_s to t_e. One constraint we consider is that the instantaneous server bandwidth consumption $SBC_i(k)$ at round k must be bounded by the maximum bandwidth $u_i^{\max}$ allocated to this channel by the server, where $\sum_{i \in HD} u_i^{\max} = u^{\max}$. $\omega_i(k)$ are the unknowns. Thus, the optimization problem can be formulated as

$$\begin{aligned}
\textbf{min}: &\sum_{k=1}^{KR} ((\bar{d}_i^{\text{view}} x_i(k) - \bar{s}_i^{\text{help}} y_i(k))^+ + \omega_i(k) f/\Delta t) \\
\textbf{st}: &\ \forall k,\ (\bar{d}_i^{\text{view}} x_i(k) - \bar{s}_i^{\text{help}} y_i(k))^+ + \omega_i(k) f/\Delta t \le u_i^{\max}; \\
&\ \forall k,\ \omega_i(k) \in \mathcal{Z}_0^+
\end{aligned}$$

However, the objective function above exhibits nonlinearity again, which is difficult to solve. To make it tractable, we introduce a variable vector u_i^{c} containing the compensation cost of the server for channel i at each round, and convert the optimization problem to a mixed integer linear programming (MILP) problem. To achieve it, we add a constraint that the aggregated bandwidth supply from helpers and the server is no less than the total bandwidth deficit of these viewers at any time, i.e., $u_i^{\mathrm{c}}(k)+\bar{s}_i^{\mathrm{help}}y_i(k)-\bar{d}_i^{\mathrm{view}}x_i(k)\geq 0$ for $1\leq k\leq KR$. It is a necessary condition for all the viewers to watch the movie fluently. The optimization problem then becomes

$$\begin{aligned}
\mathbf{min}: & \sum_{k=1}^{KR}(u_i^{\mathrm{c}}(k)+\omega_i(k)f/\Delta t) \\
\mathbf{st}: & \forall k,\ u_i^{\mathrm{c}}(k)+\bar{s}_i^{\mathrm{help}}y_i(k)-\bar{d}_i^{\mathrm{view}}x_i(k)\geq 0 \\
& \forall k,\ u_i^{\mathrm{c}}(k)+\omega_i(k)f/\Delta t\leq u_i^{\max}; \\
& \forall k,\omega_i(k)\in\mathcal{Z}_0^+,\ u_i^{\mathrm{c}}(k)\geq 0,
\end{aligned} \tag{4.6}$$

where both $u_i^{\mathrm{c}}(k)$ and $\omega_i(k)$ are unknown variables. The optimization becomes an MILP problem; although still NP-hard, many methods in the literature can generate "suboptimal" solutions in polynomial time. These suboptimal solutions can also provide insights and guidelines for the design of an active-caching strategy.

4.3 Bandwidth Allocation Strategies

The optimal caching strategies we obtained in the previous section assume that a helper is always able to help viewers. Similar to stationary scenarios, this is not true as such an HD movie cached on helpers is often incomplete due to the space limit. Therefore, we adopt a combination of the following two bandwidth allocation strategies: chain-based allocation and substream multiplexing. Our objective is to reduce the chance of content bottleneck as much as possible, which leads to a better utilization of the peer upload bandwidth.

4.3.1 Chain-based Allocation

Our chain-based allocation has been explained in details under stationary scenarios in Chapter 3. The allocation for non-stationary scenarios is slightly different, which we explain briefly as follows.

The chain-based allocation is composed of two parts: the inner-chain and inter-chain allocation, where the former allocates the upload bandwidth of concurrent peers watching the same channel following the traditional chain-based allocation, and the latter manages the allocation between the viewers and their helpers similar to Algorithm 2. The allocation process includes the following three steps.

Step 1: SD channels perform the inner-chain allocation, in which each peer reserves a certain amount of bandwidth, $\bar{u} - r_{\mathrm{S}}$, for helping HD channels later. The allocation starts from earlier viewers in the channel and stops if the bandwidth demands of peers are all satisfied, or the remaining bandwidth of each peer is less than or equal to $\bar{u} - r_{\mathrm{S}}$.

Step 2: The inter-chain allocation is invoked between an HD channel and its helpers following Algorithm 2 in Chapter 3.

Step 3: At last, the HD viewers perform an inner-chain allocation within the HD channel without reserving any upload bandwidth. After that, if there are still unsatisfied viewers, the server will compensate for the bandwidth deficit.

Remarks: The inter and inner-chain allocation aim at making the bandwidth "transferable" between peers, from SD to HD channels through the cooperation of viewers and helpers. It utilizes the cached content of the passive cachers.

4.3.2 Substream Multiplexing

Although the chain-based allocation is demonstrated to be efficient in stationary scenarios with passive caching in Chapter 3, it does not apply to non-stationary scenarios with active caching. For active cachers, a portion of the HD content needs to be determined to fetch. On one hand, to fetch a small size f of the HD content is preferred, as to reduce the caching cost $f/\Delta t$. On the other hand, f indicates the availability of the parts of the HD movie, where a larger f enables the helper to help a greater range of HD viewers. To address this concern, we adopt *substream multiplexing*, to achieve the balance between these two kinds of cost, which originates from VUD for peer-assisted live streaming systems [35, 40].

When introducing active cachers, we divide them into N_{sub} substreams. Each HD movie is divided into *sub-segments*, a smaller unit than segments, which we refer to as *chunks*. The i-th substream is responsible for the distribution of the i-th chunk of every N_{sub} chunks in the HD movie. Figure 4.3 illustrates an example of substream multiplexing of active HD helpers. There are 20 substreams. `Helper 1`, `Helper 2`,

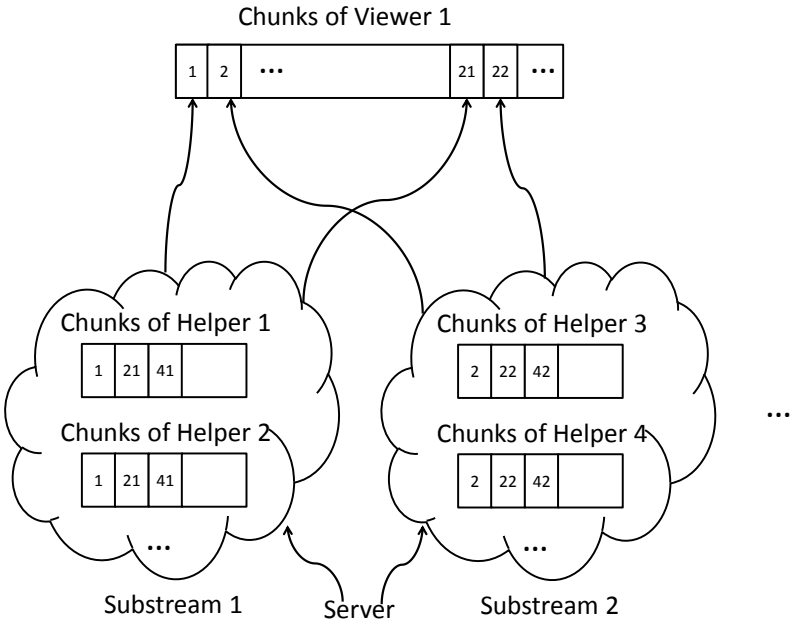


Figure 4.3: Substream multiplexing of active HD helpers.

etc. form `Substream 1`, in which they only fetch and distribute the first chunk in every 20 chunks. `Substream 2` is composed of some other helpers and distributes different chunks. Consequently, the caching cost $f/\Delta t$ is reduced dramatically as each helper only needs to cache a small portion of the HD chunks, i.e., of size $f = F/N_{\text{sub}}$, where F is the size of an HD movie. For the above case, $N_{\text{sub}} = 20$, an active helper only downloads 1/20 of the entire movie. The bandwidth cost to achieve that in one round is $F/(20\Delta t) = 1,000$ Kbps, which does not exceed the typical downlink bandwidth of broadband Internet access links. After the content is cached, the helper stays in the system, and continuously contributes the actively-cached content with its available bandwidth until leaving the substream. Therefore, the initial short-term cost during the caching round brings a higher accumulating benefit in the near future. This underlies the efficacy of our active-caching strategies.

Remarks: Substreaming aims at utilizing the active cachers' upload bandwidth with a small size of the cached HD content. Given more substreams of an HD channel, each single substream will take a smaller space on the peer cache. This not only allows a helper to contribute continuously to satisfy a variety of requests from the beginning to the end of the HD movie, but also reduces the caching cost of injecting the substream content into active cachers.

Table 4.2: The distribution of peer upload capacity.

Upload Capacity	800 Kbps	500 Kbps	400 Kbps
Percentage	40%	40%	20%

4.4 Performance Evaluation

In this section, we evaluate our caching strategies obtained from the optimization problem. We first follow a methodology to find the desired active-caching strategy (i.e., ω_i) using the `TomLab` optimization toolbox in `MatLab`, and then use a `Java`-based event-driven simulator to verify its efficacy.

4.4.1 System Setup

The peer cache size or the byte length of an HD movie is $F = 1$ GB. An SD movie takes 0.5 GB. They both last for 20 rounds in time. The upload bandwidth of peers follows the distribution listed in Table 4.2, with an average of 600 Kbps. The playback rate of the SD (HD) movies are 500 $(1,000)$ Kbps. Thus, on average one HD viewer has a bandwidth deficit of 400 Kbps, and four helpers (each has a 100-Kbps bandwidth surplus) are needed to help one viewer. One new HD movie i is released, and peers enter the channel in a flash-crowd manner.

We set the monitoring period to include three stages: the *pre-release stage*, the *flash-crowd stage* and the *steady stage*. The pre-release stage refers to the time rounds before the new movie is released, when the arrival rate of viewers is 0. Viewers start to watch the movie in the flash-crowd stage following different arrival patterns. After the flash-crowd stage, we assume that the arrival rate becomes a fixed constant, which is equal to the last round of the flash-crowd stage, and the system becomes stable afterward, i.e., the steady stage. The length of the first and third stages can be customized by VoD service providers, but the flash-crowd stage is determined by user behaviors. Similar to [64], three arrival patterns during the flash-crowd stage are as follows.

- Low intensity: the arrival rates follow an exponentially decreasing function, i.e., $\lambda_i(k) = \lambda_i(1)e^{-\frac{k-1}{\tau}}$, where $\tau = 20$ and $\lambda_i(1) = 20$. The number of arrivals $\lambda_i(k)$ are rounded up to the smallest integers at each round k.

- Medium intensity: the arrival rate at round k is $\lambda_i(k) = k$, i.e., a linearly

increasing function.

- High intensity: the arrival rate at any round during the flash-crowd and steady stage is 20, and zero otherwise.

We set the transition matrix $\boldsymbol{P}^c = \begin{bmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \end{bmatrix}$, i.e., $p^c_{SS} = p^c_{HS} = 0.8$, which indicates that SD viewers account for 80% of the total population, i.e., four helpers helping one HD viewer on average, close to the observed ratio in [1]. This can be achieved through recommendation mechanisms. For example, after a user watched an HD movie, the system can recommend some related SD programs, e.g., the discussions, interviews, and talk shows about the movie, which have lower requirements in terms of the playback rate. The effect of such recommendation is verified in [80].

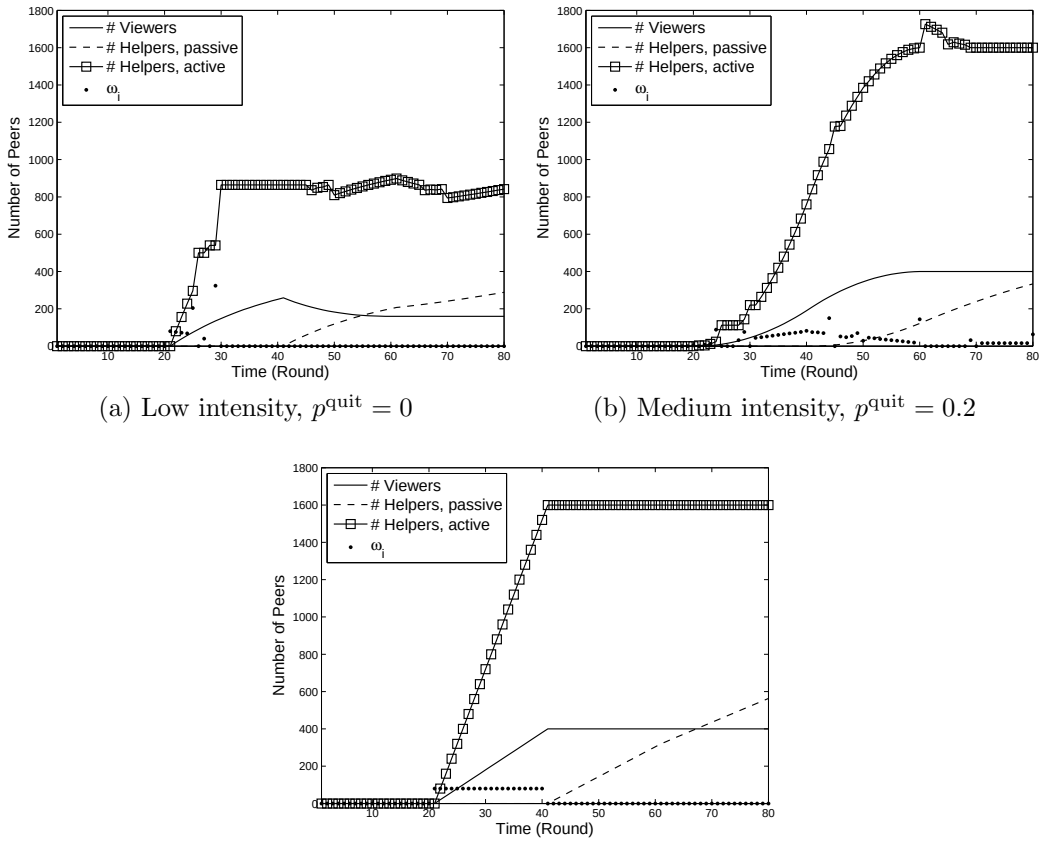
4.4.2 Obtaining the Optimal Caching Strategies

Under the system settings and the scenarios described above, we now present our methodology of obtaining the optimal caching strategies. Here we use the terminology "optimal" (instead of suboptimal) as many solutions we obtained are reported optimal by `TomLab`. We first investigate the system evolution under the three arrival patterns to gain insights, and then decide proper configuration parameters, such as the length of the monitoring period, and the minimum server capacity that guarantees our optimal solution.

System Evolution

We set $f = F/20$, and relax $u^{\max}$ to ∞, which means the caching cost of an active cacher is quite small, and the server has unlimited capacity. Let $p^{\text{quit}} = 0.2$ for the medium-intensity case, and $p^{\text{quit}} = 0$ for the other two cases (i.e., no peer leaves). We also set the pre-release stage to 20 rounds, and the overall monitoring period to 80 rounds, i.e., $KR = 80$. We solve the optimization using `TomLab` and plot in Fig. 4.4 the evolution of the system, showing the expected number of viewers, helpers in active and passive cases and ω_i (i.e., the number of new active cachers introduced) in each round in the channel under consideration.

We first observe the "mis-alignment" of the viewers and helpers with passive caching (denoted by `# Viewers` and `# Helpers, passive`), with a duration gap greater than $T = 20$ rounds. This result is consistent with [63]. The number of



(a) Low intensity, $p^{\text{quit}} = 0$

(b) Medium intensity, $p^{\text{quit}} = 0.2$

(c) High intensity, $p^{\text{quit}} = 0$

Figure 4.4: The optimal caching solutions for the three flash-crowd patterns.

helpers increases slowly after the new movie is released, which fails to provide sufficient bandwidth support for the viewers. Note that one viewer needs four helpers in terms of bandwidth supply. This is because the helper population starts to increase only after the first viewer finishes watching the HD movie and transitions to SD channels (from round 41). In contrast, through active caching, the helpers are actively introduced by the server, which quickly catch up with the fast bandwidth demand increase of the viewers. Second, active caching occurs during a short period from round 21 to round 40 or 60 in the cases $p^{\text{quit}} = 0$ for the low or high intensity. This indicates: (1) the monitoring period does not need to be very long; (2) with unlimited server capacity, no pre-release caching is needed, as we can see $\omega_i = 0$ before the flash-crowd stage. Active caching starts right after the new movie is released. However, for the case $p^{\text{quit}} = 0.2$, peers may leave the system, resulting in an insufficient number of helpers throughout the monitoring period. Thus a periodic active caching

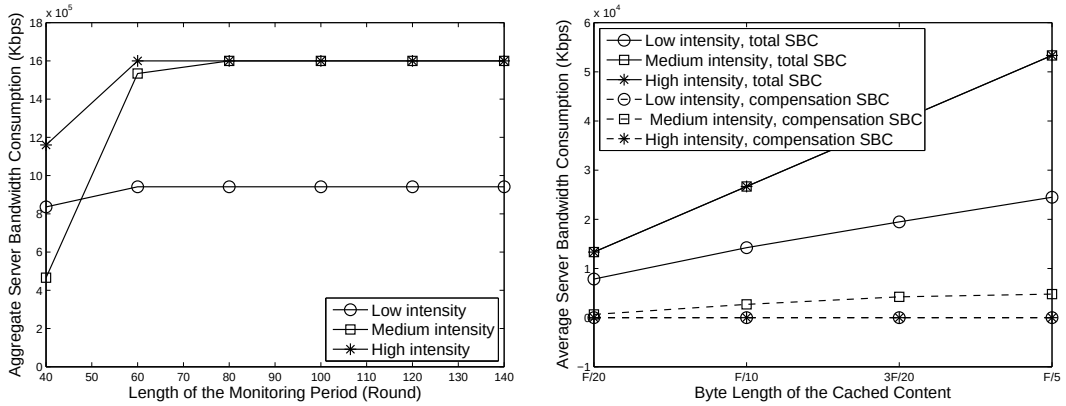
should be invoked, as shown in Fig. 4.4b. At last, from Fig. 4.4b and Fig. 4.4c, we may conjecture that for non-decreasing arrivals, a simple active caching strategy is to measure the number of helpers N^{help} and viewers N^{view} at a time instant, and inject the difference as $4N^{\text{view}} - N^{\text{help}}$ active cachers. This does not require the prediction of future arrivals.

Monitoring Period

We then investigate the desired length of the monitoring period when $p^{\text{quit}} = 0$. To reduce the cost of the server for such monitoring, a shorter period is better. However, the length of the monitoring period cannot be too short, either. Otherwise the optimization places more emphasis on the short-term benefits. For example, if the viewers are lack of 100-Kbps bandwidth in a particular round, we can either let the server compensate the bandwidth deficit at a cost of 100 Kbps, or introduce a new active cacher who has 100-Kbps extra bandwidth. The caching cost here is $1,000$ Kbps. Obviously, for this round only, active caching is not optimal, as the caching cost, $1,000$ Kbps, is far more than the compensation cost, 100 Kbps. However, once a new active cacher is introduced, it may keep helping the viewers for around 20 rounds, with an aggregate bandwidth contribution of 100×20 Kbps, which is greater than the initial caching cost. As a result, the monitoring period needs to cover at least 20 rounds in order to reflect the long-term contribution of this active cacher. To this end, we vary the length of the monitoring period during the steady stage and plot the aggregate server bandwidth consumption, i.e., $\sum_{k=1}^{KR} SBC_i(k)$. In addition, we keep 20 pre-release rounds. Figure 4.5a shows that such aggregate costs are increasing functions of the monitoring period until reaching their maximum. The cost below the maximum value (at length 40 or 60) is the *short-term cost*, as the monitoring period is too short to capture the long-term benefit, similar to the example we explained above. To conclude, the best choice in these cases is to monitor 60 more rounds after the movie is released, equivalent to 7 hours.

Cached Content

To explore the tradeoff between the compensation and caching cost, we vary the size of the actively cached content from $f = F/20$ to $f = F/5$, which we consider to be the maximum possible caching cost in real systems. The monitoring period is set to 80 rounds as suggested above (with 20 pre-release rounds). We plot in Fig. 4.5b the



(a) Length of the monitoring period (b) Byte length of the cached content

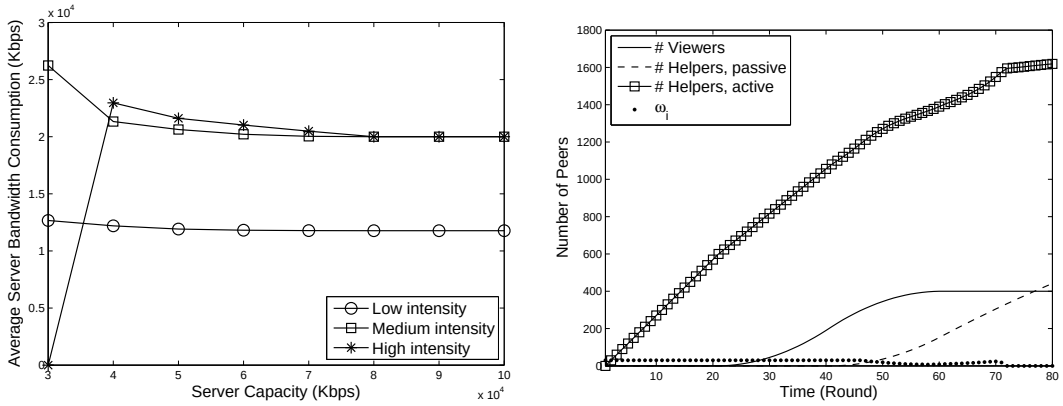
Figure 4.5: The cost to run the server with different parameters.

average server bandwidth $\overline{SBC}_{i,t_s \to t_e}$, and the average compensation cost, mean(u_i^c), allocated by the server through solving the optimization problem. With the increasing cost f of injecting active cachers, the compensation by the server rises only with the low-intensity arrival pattern. This is because from round 41, the number of viewers starts to decrease, which reduces the long-term bandwidth demand in the future. As there is no much demand in the future, it is better to compensate the current deficit rather than introducing active cachers with a greater caching cost. With the other two arrival patterns, the number of viewers is always increasing, and so is the bandwidth demand. Therefore, the long-term benefit in the future always overwhelms the caching cost at present, and the server will always introduce new active cachers, i.e., no compensation at all.

In summary, compensation fixes the current deficit without considering the long-term benefits, while active caching benefits the future with a cost at present. Therefore, the choice is based on the active-caching cost, f, as well as the trend of future demands of users.

Server Capacity

We find that the maximum instantaneous server bandwidth consumption per round from the optimization is 8×10^4 Kbps for the three arrival patterns. Then we place constraints on the maximum server bandwidth $u_i^{\max}$ and vary it from 3×10^4 to 10×10^4 Kbps. The average server bandwidth consumption is shown in Fig. 4.6a, which characterizes the trade-off between the short-term and long-term cost at the server. The first star for the high-intensity pattern reaches zero because the optimization



(a) Average SBC with different server capacity

(b) With limited server capacity, medium intensity, $u_i^{\max} = 3 \times 10^4$ Kbps

Figure 4.6: The cost to run the server with different server capacity.

problem is MILP infeasible in this case, i.e., with $u_i^{\max} = 3 \times 10^4$ Kbps, there is no way to satisfy all the viewers during the monitoring period. From the figure, a greater instantaneous server bandwidth consumption actually leads to a lower overall cost to run the server during the monitoring period. The reason is, as the active caching during the flash-crowd stage with a small $u_i^{\max}$ can only inject a limited number of active cachers, the optimization also makes peers "pre-cache" the HD content before the movie is released, as illustrated in Fig. 4.6b. However, some active cachers introduced during the pre-release stage may leave the channel system during the flash-crowd stage. The loss of the bandwidth of such active cachers results in a higher aggregate cost to run the server. In the future, to prevent the loss of such active cachers, VoD providers may also consider using cloud servers as "super" active cachers. The cost becomes a constant if $u_i^{\max} \geq 8 \times 10^4$ Kbps, which means that such a $u_i^{\max}$ is the minimum instantaneous server capacity that leads to the optimal active caching solution of this HD channel.

Moreover, although pre-release caching results in a higher average server bandwidth consumption over the period, on the other hand, it mitigates the high instantaneous bandwidth consumption at the server side. Thus the optimal choice depends on the server capacity of VoD service providers. In many cases, the server capacity is limited, and not able to support all the viewers through passive and active caching after the movie is released. To solve the problem, the server can prepare in advance, i.e., choose a pre-release stage of a proper length, and inject the HD content to SD viewers with available bandwidth. Although some of these viewers may leave before

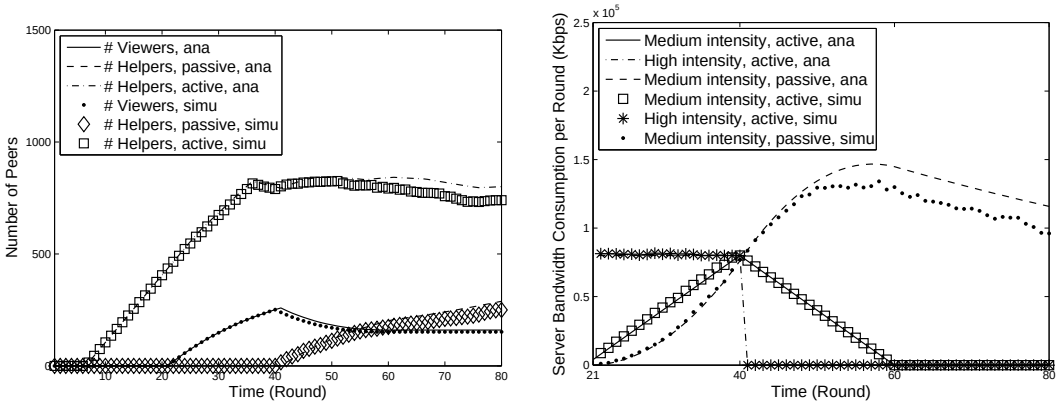
they start helping, those staying as helpers will offer bandwidth support to HD viewers after the release time. Such pre-release caching strategies are also studied through simulations in [2], while in this thesis our models provide mathematically guaranteed optimal solutions.

4.4.3 Model Validation

To further validate the performance of our active-caching strategies, we extend our `Java`-based event-driven simulator to emulate a multi-channel PA-VoD system in non-stationary scenarios. There are 10 HD channels (`Channel` 0 to 9) and 10 SD channels (`Channel` 10 to 19) at the beginning. Viewers start to join these channels following the popularity of movies, a Zipf distribution with parameter 1 for each category. The transition probabilities between categories are $p^c_{SS} = 0.8$ and $p^c_{HS} = 0.8$. Peers make decisions independently and locally when transitioning to other channels. They first decide which category to transition to according to p^c_{SS} and p^c_{HS}, and then select channels in the targeted category following the movie popularity. We set a 24-hour ramp-up stage to allow peers to replenish their local cache through passive caching. After 24 hours, we release a new HD movie in the system, i.e., `Channel` 20. A number of peers in the system are selected in each round to start watching `Channel` 20, forming the flash-crowd patterns aforementioned.

We measure the bandwidth consumption of each channel and select peers from the helpers of those over-provisioned HD channels. We then assign these peers to actively fetch $f = F/20$ of the new HD movie and help that channel. The number of such helpers follows the optimal ω_i for each round we obtained from the optimization problem. After that we first perform the inner and inter-chain allocation, and then use a group of active cachers to support the unsatisfied viewers through substream multiplexing. When adding helpers to substreams, we select the substream with the least bandwidth supply. When helpers are selected by viewers, we choose helpers from the best-provisioned substreams, i.e., a simple water-leveling approach.

We validate two things through our simulation: first, the evolution of the number of viewers $x_i(k)$ and helpers $y_i(k)$ derived using Eqn. (4.1), (4.2), (4.3) and (4.4); second, the efficiency of our bandwidth allocation approaches, i.e., the inner and inter-chain allocation for passive cachers, and substream multiplexing for active cachers. Fig. 4.7a shows the simulation and analytical results under the low-intensity pattern with $u_i^{\max} = 3 \times 10^4$ Kbps, while Fig. 4.7b demonstrates the server bandwidth con-



(a) Low intensity, $u_i^{\max} = 3 \times 10^4$ Kbps (b) Medium and high intensity, $u_i^{\max} = \infty$

Figure 4.7: Model validation.

sumption in the medium and high intensity patterns with unlimited server capacity. We also plot the server bandwidth consumption with passive caching for the medium intensity pattern, which increases dramatically and lasts for a long time, after the new movie is released at Round 21. With our optimal active-caching strategies, the server bandwidth consumption quickly decreases to zero in a short period, due to the sufficient bandwidth supply of the active helpers. These two figures illustrate that the simulation results agree with our analysis in all cases.

4.5 Practical Methods and Further Discussions

In this section, we move to the practical side of PA-VoD systems supporting HD channels. We first introduce the typical system structure of real PA-VoD systems, and then discuss practical techniques based on such a structure that help HD viewers receive the bandwidth and cache support from SD viewers, with desirable performance and acceptable overhead. These practical techniques include: *distributed approximate chaining*, which maintains the viewer chain of SD/HD channels with little centralized coordination, computational cost and transmission overhead; *passive substreaming and multiplexing*, which distributively enables the desirable coordination of viewers and helpers, with minimum overhead and cost as well; at last, we discuss the incentive and locality issues.

4.5.1 System Structure

A PA-VoD system is usually composed of *server*(s), *tracker*(s), and *peers*. The *log server* is also deployed in most cases, which collects performance data. However the first three components are most critical, and we will use such a structure to develop practical methods.

Video Server(s)

The video server, or a cluster of servers with coordination (nowadays, many PA-VoD systems use cloud servers as the video servers [68, 85, 86]), are the origins of video programs. New video content will be released from the server periodically and injected into the system. Another important role it plays is to provide bandwidth support to peers that cannot receive enough bandwidth from other peers.

Tracker(s)

The tracker serves as the meta-info monitor and the coordinator of the system. It collects and updates the information of each channel in real or near-real time, which is realized by receiving periodic *heartbeat* messages from peers. Moreover, once new peers join the system, it helps them bootstrap into the system by sending them the initial peer information list in which the peers will become their neighbors, including *concurrent neighbors* watching the same channel, and *helpers* from other channels.

Peers

The peers are the actual users of the VoD system. They may watch a channel during a specific period, or do nothing but just stay in the system. In our design, a peer may upload what resides in the local cache to other peers, so they act as downloaders and uploaders at the same time.

4.5.2 Distributed Chain Maintenance

In Chapter 3, we adopt in our heuristic algorithms the inner-chain allocation, which allocates the bandwidth of peers watching the same channel. Although the inner-chain allocation achieves a near-optimal performance, i.e., minimum server bandwidth consumption, it has two major drawbacks when implementing in real systems. First,

sorting peers based on their exact playback points is difficult in real time. The playback points may change quickly, and thus the entire chain needs to be re-sorted from time to time. Second, sorting all the peers watching all the channels on the centralized tracker, as proposed by [87], may introduce great cost and overhead, considering that each channel may have hundreds of viewers and a PA-VoD system may have hundreds of popular channels. To solve these two practical problems, approximate chaining, or "stratification" are proposed, where peers are sorted in approximate order or batches [39, 57, 62, 63, 74, 75]. Inspired by these proposed algorithms, we design a two-level distributed chain maintenance strategy, where the sorting process is accomplished by both the centralized tracker and distributed peers.

The Role of the Tracker

At the tracker, the peer list is maintained at a coarse-grained granularity, i.e., segments. Recall that a movie is divided into 10 to 20 segments. Each peer sends out a *heartbeat* message including the movie ID and segment ID that they are currently watching every 10 seconds to the tracker, and the tracker updates the peer list according to such movie and segment information. Such an interval of sending the periodic *heartbeat* messages varies between 0.5 to 30 seconds in real systems [54, 88–90]. The peer also reports the movie ID which it is helping if the content and bandwidth is available. The format of a *heartbeat* message (a UDP packet from a peer) to the tracker is explained in Table 4.3.

Table 4.3: The *heartbeat* message to the tracker.

Field	Meaning
IP	The IP address of the peer
port	The port number of the peer
movieID	The ID of the movie that the peer is watching
segID	The ID of the segment that the peer is watching
hmovieID	The ID of the movie that the peer is helping
timeStamp	The time when this *heartbeat* message is received
hasExtraBW	An binary indicator, indicating if the peer has extra bandwidth to help HD channels

We now explain in detail our design of the peer list maintenance at the tracker. To accelerate the querying and updating process, hash tables are used to store the information of peers watching each segment of each channel. Similar to [91], we design the data structure of the peer list at the tracker as illustrated in Fig. 4.8. The hash

table *allinfo* stores the reference to each entry of the peer information, and is used for the fast lookup of an entry. All the entries in all the hash tables are indexed by a tuple (*IP*, *port*). The updating process of a peer is as follows. When a *heartbeat* message is received, the tracker retrieves the *IP*, *port*, *movieID*, *segID*, *hmovieID* and *hasExtraBW* from the packet. Then it attempts to find the peer in the *allinfo* hash table, and retrieves the original *movieID*, *segID* and *hmovieID* of the existing record. If it successfully locates the record, it updates the record and the viewer and helper lists, accordingly, or a new record is created and inserted to the viewer and helper lists, otherwise. Specifically, a movie in the help list only has two hash tables, one containing those helpers with extra bandwidth, and one with no available bandwidth. If *hasExtraBW* is False, we put it into the unavailable helper hash table, or the available helper hash table otherwise. The space complexity of using a hash table is usually $O(n)$. Moreover, through such hash-table operations we avoid traversing the viewer and helper lists to locate a record and the time complexity of adding/removing records is always $O(1)$. In real systems, to further reduce the burden of the tracker, *Distributed Hash Table* (DHT) can also be implemented. With these existing mature techniques, one single PC in our experiments is able to handle the *heartbeat* messages from thousands of online peers.

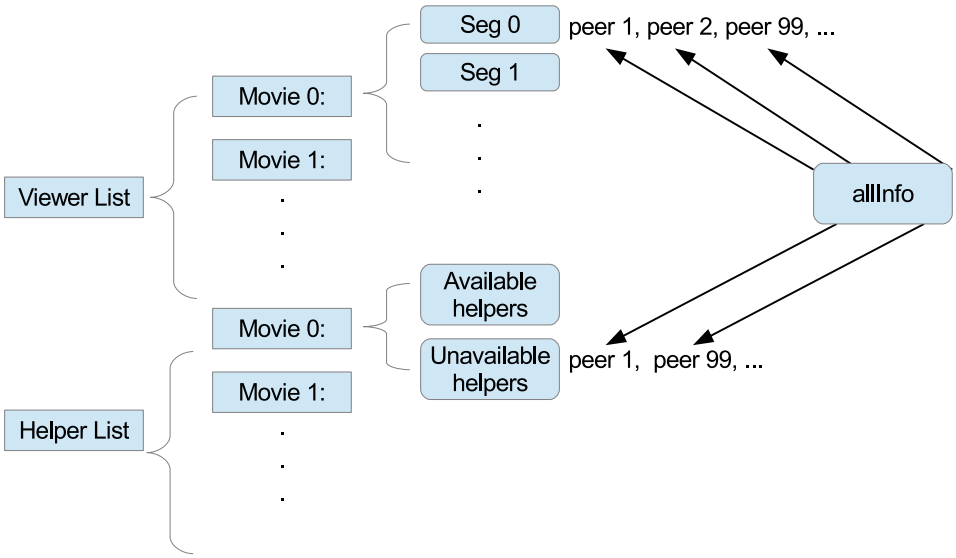


Figure 4.8: The data structure of the peer list on the tracker.

Once a peer requests for the peer list from the tracker by sending the *request list* message, the tracker will create a partial peer list, including the peers in the same

segment and following ones, adding up to a certain number of peers. Such a partial list includes the IP addresses and port numbers of online peers and is encapsulated in a single UDP packet (*peer list* packet) and sent to the requesting peer. In some cases, a peer may leave the system without notifying the tracker. An LUT (*Last-Update-Time*) field is used by the tracker to detect such departing peers without notification, and we adopt a *lazy update* strategy to handle such departures. Once a peer list is requested, the tracker needs to traverse the viewer and helper lists. For each entry that is visited, the tracker checks the LUT field. If the current time minus LUT is greater than a threshold, e.g., $3 \times LUT$, the entry will be removed from all lists, i.e., the peer is considered as an offline peer.

Gossiping between Peers

Peers maintain a smaller but fine-grained concurrent neighbor set compared with that of the tracker. After a peer receives the *peer list* packet from the tracker, it will extract from the packet the IP addresses and port numbers, and send them a *hello* message, including the fields listed in Table 4.4. Here *chunkID* represents the playback point, ranging from 1 to 10 seconds. Once received the *hello* message, these peers will reply with a *heartbeat* message including similar fields and add the peer to their neighbor sets. Once received a *heartbeat* message, a peer will update its neighbor set accordingly. Peers periodically send out the *heartbeat* messages to their neighbors in the neighbor set. The difference between *heartbeat* and *hello* messages is that the *hello* messages require an immediate *heartbeat* reply from the receivers.

Table 4.4: The *heartbeat*/*hello* message between peers.

Field	Meaning
IP	The IP address of the peer
port	The port number of the peer
movieID	The ID of the movie that the peer is watching
chunkID	The playback point of the peer
hmovieID	The ID of the movie that the peer is helping
timeStamp	The time when this *heartbeat* message is received
extraCon	The available bandwidth to allocate to concurrent neighbors
extraHelp	The available bandwidth to allocate to other channels

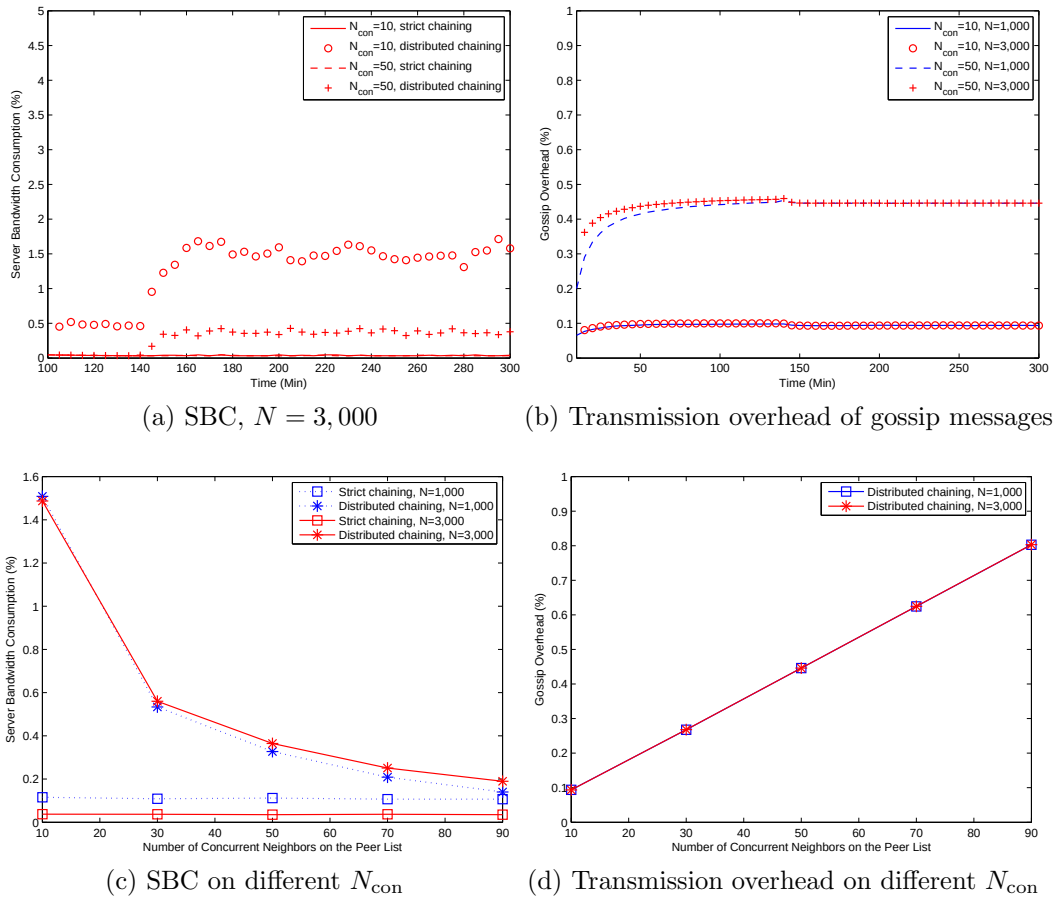
Similar to that of the tracker, the neighbors are also updated using a lazy-removal strategy, i.e., whenever an entry in the neighbor set is visited locally, the LUT field is checked to determine if the *heartbeat* message of this peer has not been received for a

long time. If such a time duration is greater than a threshold, the neighbor is removed from the neighbor set. To reduce the computational cost, the neighbors of a peer will be sorted when necessary rather than in a periodic manner. For instance, if the peer wants to seek bandwidth support from neighbors, it firstly sorts its concurrent neighbors according to the playback points stored in its neighbor set, and then tries to connect to those closest neighbors and asks for bandwidth support. As the neighbor set usually contains less than 100 peers, sorting such a list with 100 elements is trivial.

Performance Discussion

We extend our `Java`-based event-driven simulator to implement such a two-level distributed approximate chaining, referred to as `distributed chaining`. We let users join an SD channel with playback rate 500 Kbps. The arrival rates are set to allow $N = 1,000$, and $N = 3,000$ concurrent peers staying online in the steady state, respectively. Peers start watching from the beginning of the movie, and leave the system after finishing the movie. The interval of generating and sending out *heartbeat* messages is 10 seconds, and we let the number of concurrent neighbors on the list returned by the tracker, N_{con}, vary from 10 to 90. Each chunk is set to 1 second. For every 60 seconds, a peer checks its download status. If it does not receive enough downloads from other peers, it will contact the tracker again to retrieve more potential supporters. The system is sampled every 5 minutes, and the performance is illustrated in Fig. 4.9.

In Fig. 4.9a, the server bandwidth consumption is interpreted as the percentage of server-contributed bandwidth over the total bandwidth demand of peers. At the beginning of the simulation, there is a small number of peers in the system, introducing a small aggregate bandwidth demand. However, the server always has to support the first viewer as no other viewers have interested content, leading to a server bandwidth consumption at least 500 Kbps. After around 160 minutes, the system becomes stable, and we can see that our distributed-chaining approach performs quite close to the strict chain-based allocation with complete global information (with a performance gap of only 2%). We also observe that more concurrent neighbors will further reduce the server bandwidth consumption. This is because with more concurrent neighbors, a local peer is able to build a more precise chaining overlay. If the concurrent neighbor set is infinitely larger, each peer will have a complete knowledge of the channel, so that the distributed chaining will achieve the same performance with the strict



(a) SBC, $N = 3,000$ (b) Transmission overhead of gossip messages

(c) SBC on different N_{con} (d) Transmission overhead on different N_{con}

Figure 4.9: The performance of the distributed-chaining strategy of an SD channel.

in-order chaining based on the global information .

We adjust the number of returned concurrent neighbors by the tracker, N_{con}, and plot in Fig. 4.9c the server bandwidth consumption on different N_{con}. With a greater N_{con}, the server bandwidth consumption can be decreased from 1.5% to 0.2%. However, the cost is that peers will send out more gossip messages periodically, as they have more neighbors to notify, which will be explained in details later. Moreover, we observe that with a greater peer population, the performance of distributed chaining will become slightly worse, but the degradation is not very significant.

We can calculate the transmission overhead of such a two-level chain maintenance in real systems by obtaining the number of *gossip* messages sent out by peers in the simulation. First, we estimate the byte-length of a single UDP packet of such messages, ignoring the UDP headers. In a *heartbeat* message, each IP address takes 4 bytes, and a port number takes 2 bytes. *movieID*, *segID*, *chunkID*, *hmovieID*,

timeStamp, *hasExtraBW*, *extraCon*, *extraHelp* are all integer numbers, which take 2 bytes each. Allowing some more space for other information in the packets, 30 bytes will be enough to carry all the information for these packets, similar to typical P2P streaming systems [1]. We thus assume that each of such packets is of 30 bytes, and record the total number of such messages sent out to either the tracker or other peers within every 5 minutes in the simulation. The transmission overhead is calculated as the bandwidth consumption on these gossip messages over the aggregate bandwidth demand of all peers. From Fig. 4.9b we can see that such an overhead is less than 0.1% for all the arrivals when $N_{\text{con}} = 10$, and around 0.5% if $N_{\text{con}} = 50$, which is hardly noticeable. Shown in Fig. 4.9d, the maximum overhead is achieved when $N_{\text{con}} = 100$, i.e., with most concurrent neighbors, yet is still less than 1%.

A Short Summary

Through such a two-level overlay-maintenance strategy, we avoid sorting the entire peer list for each channel chain on the tracker. Instead, the task is broken down into small pieces and performed by the tracker and peers together. The tracker sends a peer the neighbors that are the best candidates to provide bandwidth support, and each peer maintains the chaining overlay of these candidates at a much smaller scale. Consequently, the computational cost at both the tracker and peers is acceptable. Moreover, according to our estimation based on simulation statistics, such maintenance overhead on the Internet is less than 1%, which demonstrates the feasibility of our distributed-chaining strategy for real-world PA-VoD systems. At last, the server bandwidth consumption is not compromised with such a distributed-chaining strategy.

4.5.3 Passive Substreaming and Multiplexing

In our heuristic algorithms, we also propose *inter-chain allocation* to allocate the bandwidth of passive cachers, and use *substream multiplexing* to minimize the active caching cost. In the real-world implementations, inter-chain allocation requires extra cost and overhead to maintain the helper chain of each HD channel. Although we can use the same distributed strategy of maintaining the viewer chains explained in Section 4.5.2, we can actually reduce such cost greatly through substreaming for both passive and active cachers.

Passive Substreaming

We design *passive substreaming*, in which passive cachers remove HD content in the unit of substreams rather than sequential segments or chunks, when the local cache is full. Recall that the ith substream of a movie contains every ith chunks of the movie. To keep a desirable diversity of the substreams as well as chunks in the system, when some HD content has to be removed at a peer, we select a random substream to remove. Figure 4.10 illustrates the process of removing `Substream 3` when an HD movie, `HD 3`, is divided into 10 substreams. The peer is watching an SD movie, `SD 1`, and it has cached an HD movie, `HD 10`, as it has watched the HD movie before. The peer cache is full, and the movie that is being watched, `SD 1`, must be stored completely before the peer finishes watching. Therefore, we randomly pick a substream, `Substream 3` in this case, and remove all the video chunks belonging to this substream from the cache.

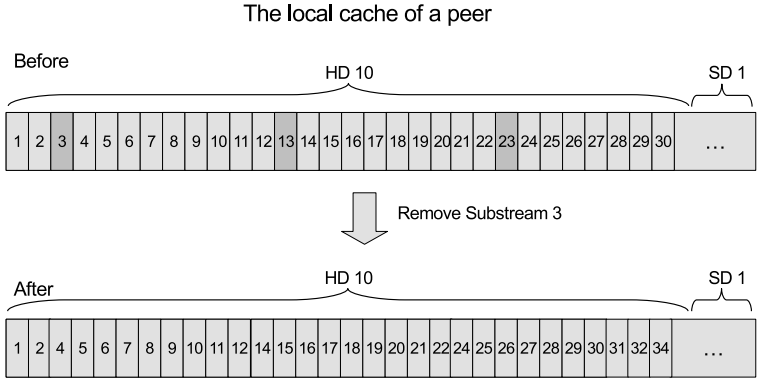


Figure 4.10: Removing a substream of an HD helper.

We now explain the benefits of such a passive substreaming. First, consider the following example illustrated in Fig. 4.11. An HD viewer is watching `Chunk 10` of an HD movie, `HD 10`, and three helpers have available bandwidth to upload to this viewer, of which one adopts the original passive-caching strategy in previous chapters, and the other two with passive substreaming. We can easily see that `Helper 2` and `Helper 3` can serve the viewer throughout the entire watching process, as the cached chunks range from the very beginning to the end of this movie. In contrast, `Helper 1` is not capable of contributing due to content bottleneck, at least before `Chunk 1001`, and thus the bandwidth of `Helper 1` will be wasted on the viewer.

Moreover, passive substreaming helps reduce the cost and overhead of pairing

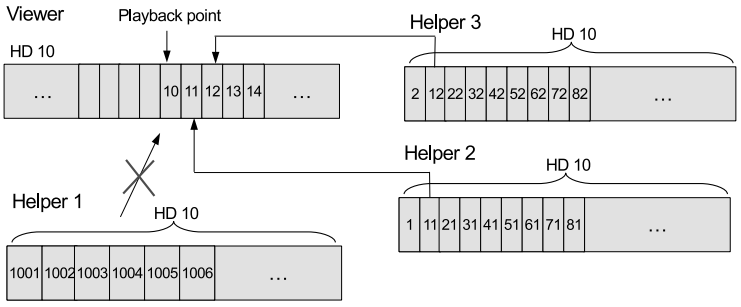


Figure 4.11: Helpers with different passive-caching strategies.

viewers and helpers. There is no need to maintain the helper chain according to the portion of the cached HD content in the inter-chain allocation in Chapter 3. In addition to the concurrent neighbor set, each HD viewer maintains a helper set, which contains the helpers to the HD movie they are watching. These helpers are also included in the peer list returned by the tracker, when HD viewers start to watch the HD movie. We simply label an HD helper as "available" if there is bandwidth available, or "unavailable" otherwise. When sending out *heartbeat* messages to the tracker, helpers include in the UDP packets an integer field $hasExtraBW$, where $hasExtraBW$=1 represents "available" and 0 represents "unavailable". The tracker will introduce available helpers in priority to the viewers of an HD movie when sending out *peer list* packets. Meanwhile, when a peer becomes a helper, i.e., transitioning from HD to SD channels, it will contact the tracker to retrieve a list of viewers watching the channel, and send *hello* messages to notify these viewers that a new helper is available.

A natural concern is on the efficiency of such random substream removal. If a viewer selects a random helper, how likely will the cached substream be helpful? Assume that an HD movie with $1,000$ Kbps is divided into 20 substreams. A viewer has to receive different substreams in order to form the complete video stream. Assume that an HD viewer is watching an HD movie of $1,000$ Kbps, receiving 600 Kbps from concurrent neighbors, and needing 8 helpers of 50 Kbps each, i.e., a typical case in this chapter. A passive cacher caches half of the HD movie, due to the cache limit explained before. For an HD movie divided into 20 substreams, 10 random substreams will be cached by each passive cacher. An active helper may have fewer substreams, e.g., only 1 substream, as to reduce the cost of active caching. In the

worst case, the helper list returned from the tracker contains only active cachers, and we also assume that a viewer will retrieve one substream from a helper due to its availability of bandwidth. Selecting such 8 distinct substreams from active cachers is a *coupon collector's problem* [92], where there are 20 distinct coupons, and 8 coupons need to be collected. Therefore, the expected number of trials can be calculated as $20 \times (\frac{1}{13} + \frac{1}{14} + \frac{1}{15} + ... + \frac{1}{20}) < 10$. This indicates that a peer needs to contact 10 different helpers on average, to retrieve enough distinct substreams. Therefore, we set the number of helpers returned by the server to $30 > 10$, as to offer enough number of helpers to HD viewers.

Algorithm 3 explains the detailed random pairing process of a viewer and a helper. The viewer and helper first decide the maximum possible streaming rate $b_{\max}^{h \to v}$, and the corresponding maximum number of substreams, $N_{\max}$, from the bandwidth point of view. Then, considering the content availability, the helper locates candidate substreams for the viewer, $\mathbb{S}_{\text{can}}$, which contains the substreams needed by the viewer and available at the helper. If $N_{\max}$ is greater than the number of candidate substreams in $\mathbb{S}_{\text{can}}$, all the substreams will be selected. Otherwise, the helper randomly picks $N_{\max}$ substreams to support the viewer. There are two main advantages of such a random algorithm. First, the pairing process takes place between two peers with no centralized information or coordination, i.e., completely distributed. Moreover, the random pick inherently achieves a balanced demand for all substreams throughout the system, i.e., with a similar effect to the water-leveling approaches.

Stream Multiplexing

Stream multiplexing is to select different video chunks from concurrent neighbors and helpers, to avoid the underutilization of the bandwidth allocated on the corresponding connections. For instance, if a concurrent neighbor and a helper are helping an HD viewer. The concurrent neighbor has all the chunks that the viewer is interested in, while the helper only has `Substream` 1. In this case, the ideal strategy is to download the chunks belonging to one of the substreams from the helper, and all other ones from concurrent neighbors. If a chunk belonging to `Substream` 1 is requested from concurrent neighbors, the bandwidth of the helper will be wasted, as no other chunks can be requested from the helper.

In real systems, stream multiplexing is more complex than the above example. First, the bandwidth from peers might be inadequate, so the server(s) needs to par-

Algorithm 3 Determining the bandwidth and substreams between viewers and helpers.

Require: A viewer v with bandwidth deficit d^v and the needed substream set $\mathbb{S}_{\text{need}}$; a helper h with available help bandwidth u^h and the cached substream set $\mathbb{S}_{\text{ava}}$; the streaming rate of a single substream of the corresponding movie, r_{sub}.

Ensure: A selected substream set $\mathbb{S}_{\text{sel}} \subseteq (\mathbb{S}_{\text{ava}} \cap \mathbb{S}_{\text{need}})$, such that the allocated bandwidth from helper h to viewer v, $b^{h \to v}$, is maximized.

1: $b^{h \to v}_{\max} := \min\{d^v, u^h\}$, $N_{\max} := \lfloor b^{h \to v}_{\max} / r_{\text{sub}} \rfloor$
2: $\mathbb{S}_{\text{can}} := \mathbb{S}_{\text{need}} \cap \mathbb{S}_{\text{ava}}$
3: **if** $N_{\max} > 0$ and $\mathbb{S}_{\text{can}} \neq \emptyset$ **then**
4: **if** $N_{\max} > |\mathbb{S}_{\text{can}}|$ **then**
5: $\mathbb{S}_{\text{sel}} := \mathbb{S}_{\text{can}}$, $b^{h \to v} := |\mathbb{S}_{\text{can}}| \times r_{\text{sub}}$
6: $\mathbb{S}_{\text{need}} := \mathbb{S}_{\text{need}} - \mathbb{S}_{\text{sel}}$
7: **else**
8: $\mathbb{S}_{\text{sel}} :=$ randomly select $N_{\max}$ substreams from $\mathbb{S}_{\text{can}}$
9: $b^{h \to v} := |\mathbb{S}_{\text{sel}}| \times r_{\text{sub}}$
10: $\mathbb{S}_{\text{need}} := \mathbb{S}_{\text{need}} - \mathbb{S}_{\text{sel}}$
11: **end if**
12: **else**
13: $\mathbb{S}_{\text{sel}} := \emptyset$, $b^{h \to v} := 0$
14: **end if**

ticipate in such a process as well. Second, each video chunk is associated with a playback deadline. For the chunks with a closer deadline, it is better to send the download request to a more reliable supporter, i.e., the server(s). Therefore, we need a stream multiplexing strategy considering both the content availability on supporters and the playback deadlines of video chunks.

The *sliding window* buffer management is widely used to schedule chunk requests to multiple supporters [48, 88]. Here we briefly discuss how it applies to receiving chunks from multiple supporters with different roles. The viewer maintains a buffer with a fixed size, containing all the video chunks of its recent interests. The buffer size varies from 20 seconds to 2 minutes for popular P2P streaming systems [90,93,94]. At the left end, the video chunks are continuously fed to the video player as the viewer watches the movie, and a new chunk is then ready for scheduling at the right end.

Figure 4.12 illustrates the sliding window buffer management of an HD viewer. The cells with green color represent already-arrived chunks, and other cells represent incomplete chunks that need to request for, or are currently downloading. The *server deadline* indicates how urgently a video chunk is needed. If a video chunk is on the left to the server deadline and still incomplete, it will be requested from the server.

On the right of the server deadline, all the requests will be sent to peers, including concurrent neighbors and substreaming helpers. The priority is given to helpers, i.e., if a chunk belongs to a substream provided by a helper, the requests will be sent to the helpers, and the remaining chunks to concurrent neighbors.

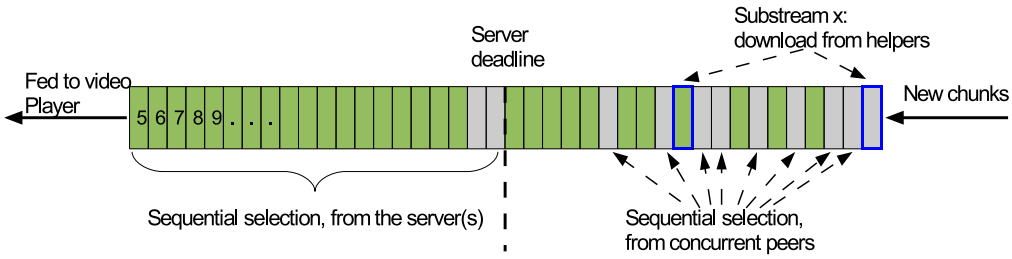
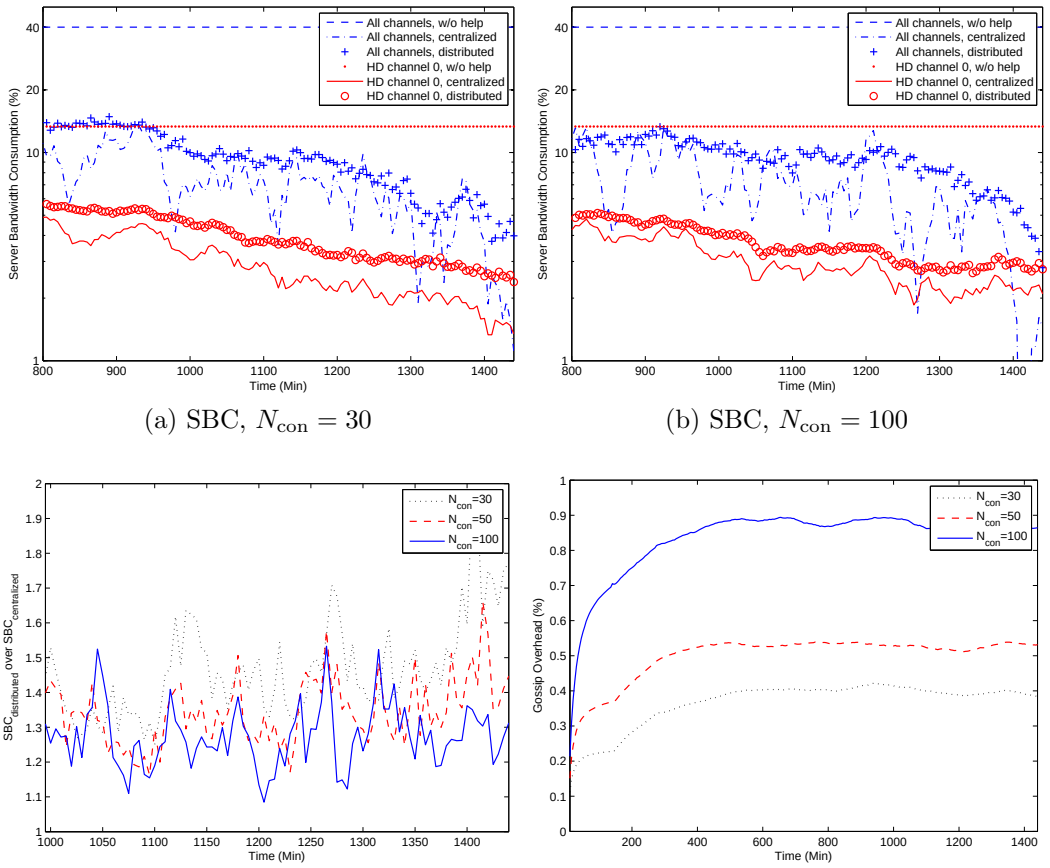


Figure 4.12: The sliding window buffer management.

Performance Discussion

To evaluate the performance of passive substreaming and random pairing between HD viewers and helpers, we also implement these strategies along with the distributed chaining in our `Java`-based event-driven simulator. We let $N = 10,000$ peers join the system with no pre-cached content within the duration of a single movie. Viewers select a movie to watch following the popularity similar to previous sections, i.e., 80% viewers in SD channels and 20% in HD in the steady state. There are 10 HD and 10 SD movies in the system. The distribution of the peer upload capacity follows Table 4.2, with an average of 600 Kbps. The playback rates of SD and HD movies are 500 and 1,000 Kbps, respectively. Therefore, the best server bandwidth consumption percentage without cross-channel allocation is 13.4% for the entire system, and 40% for a single HD channel. If an HD helper has a 800-Kbps bandwidth capacity, we make it reserve 250 Kbps to help the HD channel, and use the rest 550 Kbps to support its own viewing channel. Therefore, the average bandwidth allocated to HD channels by the helpers is 100 Kbps per peer. Each HD movie is divided into $N_{\text{sub}} = 20$ substreams. Peers transition between channels following the movie popularity as well, but will not return to the movie it has stored in its local cache. The performance of such a combination of distributed strategies is plotted in Fig 4.13.

Figure 4.13a and Fig. 4.13b illustrate the percentage of the server bandwidth consumption over the total bandwidth demand of the entire system and a single



(a) SBC, $N_{\text{con}} = 30$

(b) SBC, $N_{\text{con}} = 100$

(c) Distributed vs Centralized

(d) Transmission overhead of gossip messages

Figure 4.13: The performance of distributed and centralized strategies.

HD channel, `HD 0`, when $N_{\text{con}} = 30$ and $N_{\text{con}} = 50$, respectively. The performance with our distributed chaining and substreaming is referred to as `distributed` and the inner and inter-chain heuristic allocation as `centralized` in all figures. We also observe that the distributed strategies are able to bring similar performance to the centralized inner and inter-chain allocation, in all cases. The best server bandwidth consumption without cross-channel allocation for the entire system and the channel `HD 0` are also plotted for reference, denoted as `All channels, w/o help`, and `HD channel 0, w/o help`. Again, with a larger concurrent neighbor set, the performance gap between the distributed and centralized cross-channel allocation becomes smaller. In Fig. 4.13c we quantify this gap by calculating the ratio of the server bandwidth consumption of the distributed allocation to the centralized version. The blue solid curve, representing the largest concurrent neighbor set, outperforms smaller ones in most cases.

As *heartbeat* messages are transmitted between different channels, we record the aggregate number of gossip messages sent by all the peers from all channels during each 5-minute interval, and calculate the bandwidth spent on transmitting these messages using the same method for the single-channel case in the previous section. Different from the single-chain maintenance, between helpers and viewers, the availability of substreams also needs to be included in the *heartbeat* messages. This can be implemented by using the *bitmap*, where each bit indicates the availability of a substream. For 20 substreams, 4 bytes (32 bits) will be enough to store such information. As a result, we still assume that each *gossip* packet takes 30 bytes. Once again, illustrated by Fig. 4.13d, the overhead is still below 1% in all cases, which is negligible in real systems. With a larger concurrent neighbor set, there will be more gossip messages transmitted, therefore a greater overhead approaching 1%.

A Short Summary

The passive substream caching proposed in this section has demonstrated high efficiency in our simulations. This is due to two facts. First, many HD viewers are able to receive bandwidth support from their concurrent neighbors. Such an amount of bandwidth, although not sufficient to catch up with the playback rate of the HD movie, can reduce the number of needed substreams at the viewers. Second, each passive cacher caches half of all the substreams, which provide a large choice space of substreams for HD viewers. Therefore, HD viewers always select a small number of substreams from a large number of distinct candidates. In most cases, we can safely claim that any helper is able to help any viewer. At the same time, the overlay maintenance of these helpers is simplified greatly, as they form a mesh-based overlay which requires little centralized control and signaling overhead.

4.5.4 Incentive Mechanisms

At last, we discuss the incentive mechanisms in PA-VoD systems with HD and SD channels, which encourage peers to contribute upload bandwidth. Although PA-VoD applications employ many methods to utilize more bandwidth from users, such as hiding the option of adjusting the upload rate, or creating background processes called “accelerators” that upload the cached video content without explicitly notifying users, users can still throttle the upload bandwidth rate of a PA-VoD application by using a variety of OS and network management software.

The *tit-for-tat* mechanism from the BitTorrent protocol is not suitable for PA-VoD systems, due to the asymmetric data transmission between peers. Therefore, the contribution of peers in PA-VoD systems are usually evaluated on a long-term basis, rather than the short-term contribution rate in BitTorrent. Liang et al. defined such contribution as the total number of uploaded video chunks since a peer joins the system, and supported peers with a higher contribution to prefetch at a higher speed [87]. Wu et al. defined the rewards to peers as an increasing function of their dedicated maximum upload bandwidth [95]. Wang et al. proposed a light-weight currency-based incentive mechanism, and granted helper peers 10 points with every KB that they have uploaded [96]. Wu et al., on the other hand, designed an auction-based scheme, in which viewers bid for each video chunk, and let the price of each chunk be determined by the market, based on the corresponding demand and supply. In real systems such as PPLive, PPStream and Xunlei Kankan, users can pay real money for some special membership. Many HD movies are only open to these special members, even without upload contribution.

In fact, an existing solution from private BitTorrent communities is able to offer sufficient incentives to viewers in PA-VoD systems [97]. In private BitTorrent communities, users are forced to keep the share ratio above a threshold, i.e., the aggregate upload volume to the download volume. Users start with free resources, download them and upload to other peers later. After they increase the share ratio to a certain level, they will be allowed to approach more advanced resources. If the share ratio is below a threshold, the user is required to increase it to above that within a certain period, e.g., two weeks. Otherwise the user will be suspended. Therefore, users will automatically attempt to download popular files and act as seeds, i.e., uploading only, for a longer time thereafter. This is quite similar to the behaviors of the active cachers in our PA-VoD systems. To help users accumulate the upload volume, the real-time statistics of all the resources currently distributing in the community are advertised on specific websites, with the number of seeds and downloaders accessible to users. The users will browse these websites to locate the files that require most uploads. Such incentives have demonstrated a significant positive effect on encouraging peers to contribute, as many users even rent seedboxes (a private server rented by users to upload user-specified files) to continuously upload to other peers [98].

The incentives for PA-VoD systems may follow the same strategy. Users are forced to maintain a certain share ratio. The service provider lists the statistics of all movie channels on the graphical interface of the VoD application. After a user finishes an

HD channel and observes that many users are watching it but few uploading, it will stay in the system and upload the content of this movie, as to quickly accumulate the upload volume. The application may also allow a user to download one or two substreams of an HD channel for free, to upload to other peers with these substreams, i.e., performing as active helpers. Through such an approach, we believe there will be a great number of viewers willing to contribute to HD channels, as to accumulate the upload volume to facilitate the access to more advanced resources.

4.5.5 Locality Issues

In practical systems, it is also desirable to make P2P systems ISP-friendly, i.e., locality-aware, as ISPs start to throttle P2P traffic due to its huge volume [99]. According to measurement studies on BitTorrent systems, 50%–90% of local file chunks are downloaded from external peers [100]. Such a huge amount of inter-ISP traffic leads to the great cost on ISPs, discouraging them from tolerating P2P applications.

Extensive research work has been conducted concerning ISP-friendly design of P2P systems, through either peer-driven biased neighbor selection [101–103], ISP-driven biased neighbor selection [104, 105], or ISP caching [106–108]. The strategies in the first two categories try to guide peers to select nearby neighbors, either using proximity information provided by ISPs, or through end-to-end distance estimation by end-users themselves. For instance, in PPLive [28] and CoolStreaming [40], peers adopt a latency-based neighbor selection, which implicitly favors neighbors within the same ISPs and thus help reduce the inter-ISP traffic.

In this thesis, we assume a mesh-based topology where any peers can connect to each other, and have not considered locality-related issues. Nevertheless, the biased neighbor selection mentioned above can be easily applied to our practical algorithms in this thesis, where peers connect to close neighbors in terms of geographical location in priority. Again, such a biased peer selection can be achieved by both the tracker and the peers. When the tracker is requested a neighbor list by a peer, it also looks up the ISP information according to the IP address of that peer, and replies with the neighbors at closer playback and geographical positions. Similar to PPLive and UUSee, the peer can perform a latency test, and connect to those neighbors with closer playback points and shorter transmission latency. The VoD service providers can also deploy edge-servers or trackers closer to the peers at different locations, e.g., continents, as to better assist them on the locality-ware neighbor selection and overlay

maintenance. In this case, the original system in this thesis is therefore divided into several sub-systems according to their geographical locations, where our models and practical algorithms still apply to each of these sub-systems.

4.6 Conclusions

This chapter focuses on the caching strategies in multi-channel PA-VoD systems with HD channels under non-stationary scenarios, as well as practical methods for implementation. We use our modeling framework to model different user behaviors and evaluate a variety of caching strategies. The server bandwidth consumption is derived, and the optimal caching strategies are investigated using mixed integer linear programming (MILP). We verify that passive caching is usually inefficient when a new HD movie is released into the system, as few peers have cached the content, i.e., no helpers. Active caching, on the other hand, through paying a short-term cost to inject HD content to peers with available bandwidth, is able to bring long-term benefits with the everlasting contribution of active cachers. We also discuss practical techniques that help make cross-channel bandwidth allocation feasible for real PA-VoD systems. These techniques include distributed chain maintenance, substreaming, multiplexing, incentive mechanisms and locality aware neighbor selection. Through these techniques, we try to provide implementation guidelines based on the insights gained from our modeling work, which bridge the gap between our theoretical analysis and real-world implementations.

Chapter 5

Conclusions

Peer-assisted video on-demand systems have evolved into its maturity stage. High-definition channels are offered prevalently on the Internet, offering a great watching experience to users. However, this also introduces great challenge to system designers, as HD channels consume more system resources and the peer contribution from these channels can hardly meet such a requirement. In this thesis, we have studied a variety of caching and bandwidth allocation strategies under different scenarios, to enable the cross-channel resource allocation, i.e., making SD channels help HD channels. We have built models to mathematically characterize the effectiveness of these strategies and also designed heuristic algorithms and practical techniques that are feasible for real-world PA-VoD systems.

In this chapter, we first summarize this thesis, and then discuss the future directions.

5.1 Summary of the Thesis

In this thesis, we aim at solving the *resource imbalance* problem in PA-VoD systems offering HD channels. These channels, with a better playback quality (i.e., higher playback rate), have introduced a much higher bandwidth demand than SD channels. Such demand can easily exceed the bandwidth capacity of HD viewers, and thus they have to seek extra help from SD viewers, or the server at the final resort. Without a proper cross-channel resource allocation strategy between SD and HD channels, the latter will either suffer from poor streaming performance, or consume a huge amount of the processing power and bandwidth of the server(s). This not only increases the

maintenance cost of the servers, but also brings a large bill on the bandwidth consumption to the VoD service providers, as ISPs charge them based on the consumed volume of bandwidth.

We study two major design components of PA-VoD systems in this thesis: *bandwidth allocation* and *caching* strategies. Bandwidth allocation is to determine how much bandwidth to allocate from which peer towards which peer, and thus determines the streaming overlay of peers. Typical candidates include *perfect-fair-sharing*, *inner chain*, and *inter-chain allocation*, etc. Caching strategies, on the other hand, select which video chunk to remove or fetch, and thus affects the availability of these chunks. We mainly focus on three strategies: *FIFO*, *passive* and *active caching*. This thesis also investigates two kinds of representative scenarios: *stationary* and *non-stationary* scenarios. The stationary scenarios indicate a system that is stable enough, i.e., the expected viewer population of each channel remains fixed, while in non-stationary scenarios, we release a new HD movie into the system, and let the population of this new HD movie increase from zero.

To provide a solid understanding of PA-VoD systems with channel heterogeneity, we first build a generic modeling framework to capture the essential characteristics of these systems, e.g., the bandwidth supply and demand relationship of each channel. This modeling framework is flexible and extensible, with which we can model a variety of caching strategies, under either stationary or non-stationary scenarios.

Applying the modeling framework to stationary scenarios, it can be easily proved that passive caching achieves the optimal performance, i.e., with a minimum server bandwidth consumption. This phenomenon is supported by the simulation results of our event-driven `JAVA`-based simulator, which is referred to as "self-adaptivity". As to non-stationary scenarios, our modeling framework allows locating the optimal active-caching strategies, in which the VoD service providers are provided with the detailed solutions: the time and number of active cachers to introduce, and the cost and benefits of such active-caching strategies.

Other than the modeling work, we also propose heuristic algorithms which perform closely to the optimal solutions we obtained from the model. We extend the existing inner-chain allocation in the literature to inter-chain allocation, where the bandwidth of SD viewers can be effectively utilized to help HD viewers, by making such bandwidth as "transferable" as possible. Moreover, our passive and active caching algorithms are simple enough to implement. The passive caching requires no global coordination. Each peer only removes SD movies in the cache that are not

being watched. As to active caching, we divide a movie into multiple substreams, and let an active cacher be responsible for distributing one substream. This significantly reduces the server cost on pushing content to potential helpers. Our heuristic algorithms, although not able to implement directly, have provided us further insights into the implementation of PA-VoD systems.

At last, we go one step further to extend our heuristic bandwidth allocation strategies to follow a more distributed manner, i.e., the distributed practical techniques that will be useful for real-system implementation. As the strict in-order chaining is difficult to achieve in real time, we design a two-level distributed chaining strategy with less cost and overhead. We propose passive substreaming and multiplexing, and bandwidth negotiation between viewers and helpers, through which the desirable coordination between multiple uploaders can be achieved in a distributed manner. The efficacy of these practical methods is verified through simulation.

5.2 Other Accomplished Work that Contributes to this Thesis

Besides the work presented in Chapter 3 and 4, there is other published work that contributes to the completion of this thesis.

5.2.1 On the System Parameters of Peer-to-Peer Video Streaming with Network Coding

In this work we studied two chunk scheduling algorithms, rarest-first and random linear network coding, in P2P video streaming systems under an extreme scenario, i.e., all the peers watch the same channel, beginning at the same time. We propose a simple analytical model to characterize and verify the efficiency of network coding. Several system parameters such as chunk size, server capacity, and peer aggressiveness are investigated with their influence on the system performance under such flash-crowd scenarios. Both our theoretical analysis and simulation results demonstrate that network coding can perform very close to the idealized scheduling algorithm for peer-to-peer video streaming [109].

This work has provided insights into the chunk-level optimization of P2P streaming systems, which facilitates our further analysis on the bandwidth level. Through

this work, and results from real systems [1], we learn that there are always effective strategies, such as network coding, that can balance the content availability at the chunk level, even under extreme flash-crowd scenarios. Therefore, in this thesis, we ignore the chunk scheduling details and focus on the high-level bandwidth allocation design.

5.2.2 Optimizing BitTorrent-like Peer-to-Peer Systems in the Presence of Network Address Translation Devices

In this work, we focus on the resouce imbalance problem caused by the accessibility of peers in BitTorrent systems, one of the most important P2P file-sharing applications on the Internet. Network Address Translation (NAT) has become pervasive in almost all networking scenarios, from residential Internet access to enterprise networks. Despite the effort of NAT traversal, it is still very likely that P2P applications cannot receive incoming connection requests properly if they are behind NAT. We propose and model the biased optimistic unchoke strategy, a simple but effective strategy to mitigate the negative impact on NAT peers. Furthermore, we optimize the system performance in terms of both average peer download time and system finish time [77].

This work studies the detailed transmitting and signaling protocols on distributed peers in BitTorrent swarms, which has provided insights into the design of our distributed chaining and substreaming strategies in Chapter 4. The bootstrap process of peers, function of the tracker, and the periodic gossiping between peers are all based on similar techniques in the BitTorrent protocol. Moreover, we also create a fast `Java`-based BitTorrent simulator during this work. This simulator is further extended to PA-VoD systems, which we use to validate the models in this thesis.

5.3 Future Directions

Although many commercialized PA-VoD systems have been deployed, these following directions are still worth exploring, in order to further improve the system performance.

Measurement or trace study on user behaviors. Our models clearly indicate that user behaviors will affect the bandwidth provisioning of channels, especially HD ones. Therefore, a measurement or trace study of real PA-VoD systems is

preferred, with the focus on the user transition behaviors. How do users transition between HD and SD channels? How are such behaviors influenced by the the streaming quality and the popularity of channels? How likely will a peer perform VCR operations while watching different movies? Measurement studies have been conducted on a variety of video streaming systems [1,16,28,110,111], but the aforementioned aspects are still not well-studied.

Supporting VCR operations. Supporting VCR operations in PA-VoD systems has been studied through modeling, simulation, and real-system experiments [39, 54,58,59,112,113]. In this thesis we do not explicitly consider VCR operations, as evidence from real systems reports that for online movies, in most cases, such operations do not happen frequently. For example, the average number of jumps per movie is around 2 in PPLive and UUSee [1,28]. We believe our chaining overlay should be resilient to these VCR operations. If a viewer jumps to another playback point, we can simply remove it from the original position in the chain and insert it to the new position, like the removals and insertions in a linked list. To better support VCR operations, we also need to learn from the study on user behaviors, and then develop an improved chaining strategy accordingly.

Cloud and peer-assisted VoD. In peer-assisted VoD systems, the server(s) plays a crucial role. Evidently, existing systems employ dedicated servers with a fixed capacity and cost [1,114], and it is a desirable alternative to use the cloud infrastructure [68,85,115,116]. As the cloud service is charged based on the actual network, storage and CPU usage in real time, it inherently adapts to the user dynamics and helps avoid the unnecessary cost of dedicated servers. The superiority of cloud over P2P has been claimed in [85,116,117], while others suggest a hybrid approach [68,115]. As stated in this thesis, P2P video streaming systems nowadays require intensive coordination between the centralized server(s) and peers, where a purely decentralized P2P approach is infeasible. On the other hand, the assistance from the huge number of peers can further reduce the cost on the cloud usage. This will motivate us to study the design for a hybrid VoD systems, i.e., with the support from both the P2P and cloud technologies.

Migrating the incentive mechanism from private BitTorrent communities. Although such a solution from private BitTorrent communities has demon-

strated its extreme feasibility for encouraging peers to contribute, it is still interesting to investigate its efficiency in real-world PA-VoD applications. Moreover, recently, many PA-VoD companies have launched membership systems for users. The evaluation and modeling of these incentive mechanisms through membership systems is also of great value.

Bibliography

[1] Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. UUSee: large-scale operational on-demand streaming with random network coding. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE, 2010.

[2] Weijie Wu and John CS Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1206–1214. IEEE, 2011.

[3] Guangqing Deng, Ting Wei, Changjia Chen, Wei Zhu, Bin Wang, and Dengrong Wu. Measurement and modeling for QoS of VoD system. In *Proceedings of IEEE International Conference on Internet Computing & Information Services (ICICIS)*, pages 169–172. IEEE, 2011.

[4] YouTube homepage. `http://www.youtube.com`.

[5] YouTube Blog. Holy Nyans! 60 Hours Per Minute and 4 Billion Views a Day on YouTube. `http://youtube-global.blogspot.com/2012/01/holy-nyans-60-hours-per-minute-and-4.html`, January 2012.

[6] YouTube's Bandwidth Bill Estimated At $300M For 2009. `http://www.multichannel.com/internet-video/youtubes-bandwidth-bill-estimated-300m-2009/130930`, March 2009.

[7] BitTorrent Homepage. `http://www.bittorrent.com`.

[8] Skype homepage. `http://www.skype.com/en/`.

[9] Cheng Huang, Jin Li, and Keith W Ross. Can internet video-on-demand be profitable? In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 133–144. ACM, 2007.

[10] PPLive Homepage. `http://www.pplive.com/en/index.html`.

[11] PPStream Homepage. `http://www.ppstream.com`.

[12] UUSee Homepage. `http://www.uusee.com`.

[13] Le Chang and Jianping Pan. Reducing the Overhead of View-Upload Decoupling in Peer-to-Peer Video On-Demand Systems. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

[14] Le Chang, Jianping Pan, and Min Xing. Effective Utilization of User Resources in PA-VoD Systems with Channel Heterogeneity. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Emerging Technologies in Communications*, accepted.

[15] Le Chang and Jianping Pan. Towards the optimal caching strategies of peer-assisted VoD systems with HD channels. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2012.

[16] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, 2007.

[17] YouTube Blog. Thanks, YouTube Community, for Two BIG Gifts on Our Sixth Birthday! `http://youtube-global.blogspot.com/2011/05/thanks-youtube-community-for-two-big.html`, May 2011.

[18] John Jannotti, David K Gifford, Kirk L Johnson, M Frans Kaashoek, et al. Overcast: reliable multicasting with on overlay network. In *Proceedings of the conference on Symposium on Operating System Design & Implementation*, volume 4, pages 14–14. USENIX Association, 2000.

[19] Yang-hua Chu, Sanjay G Rao, Srinivasan Seshan, and Hui Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1456–1471, 2002.

[20] CoolStreaming Homepage. http://www.coolstreaming.us/hp.php?lang=en.

[21] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y-SP Yum. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, volume 3, pages 2102–2111. IEEE, 2005.

[22] Anysee Homepage. http://www.anysee.com.

[23] Gridnet Homepage. http://www.gridnetworks.com.

[24] PPTV, the most popular net TV in the world. http://download.pptv.com/en/.

[25] http://www.pptv.com/aboutus.

[26] Jiahua Wu and Baochun Li. Keep cache replacement simple in peer-assisted vod systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM), mini-conference*, pages 2591–2595. IEEE, 2009.

[27] Zhen Wei Zhao, Sameer Samarth, and Wei Tsang Ooi. Modeling the effect of user interactions on mesh-based P2P VoD streaming systems. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(2):13, 2013.

[28] Yan Huang, Tom ZJ Fu, Dah-Ming Chiu, John Lui, and Cheng Huang. Challenges, design and analysis of a large-scale p2p-vod system. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 375–388. ACM, 2008.

[29] Dongyu Qiu and Rayadurgam Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*, 34(4):367–378, 2004.

[30] Bin Fan, Dah-ming Chiu, and John CS Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 239–248. IEEE, 2006.

[31] Broadband Report. http://www.crtc.gc.ca/eng/publications/reports/broadband/bbreport1111.htm, November 2011.

[32] Rakesh Kumar, Yong Liu, and Keith Ross. Stochastic fluid theory for P2P streaming systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 919–927. IEEE, 2007.

[33] Chen Feng and Baochun Li. On large-scale peer-to-peer streaming systems with network coding. In *Proceedings of ACM international conference on multimedia*, pages 269–278. ACM, 2008.

[34] Chen Feng, Baochun Li, and Bo Li. Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 891–899. IEEE, 2009.

[35] Di Wu, Chao Liang, Yong Liu, and Keith Ross. View-upload decoupling: A redesign of multi-channel p2p video systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM), mini-conference*, pages 2726–2730. IEEE, 2009.

[36] Di Wu, Yong Liu, and Keith W Ross. Queuing network models for multi-channel P2P live streaming systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 73–81. IEEE, 2009.

[37] Delia Ciullo, Valentina Martina, Michele Garetto, Emilio Leonardi, et al. How much can large-scale Video-On-Demand benefit from users cooperation? In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[38] Yipeng Zhou, Tom ZJ Fu, and Dah Ming Chiu. Statistical modeling and analysis of p2p replication to support vod service. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 945–953. IEEE, 2011.

[39] Can Zhao, Jian Zhao, Xiaojun Lin, and Chuan Wu. Capacity of p2p on-demand streaming with simple, robust and decentralized control. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[40] Bo Li, Susu Xie, Yang Qu, Gabriel Y Keung, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. Inside the new coolstreaming: Principles, measurements and

performance implications. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1031–1039. IEEE, 2008.

[41] Yishuai Chen, Changjia Chen, and Chunxi Li. Measure and model p2p streaming system by buffer bitmap. In *Proceedings of IEEE International Conference on High Performance Computing and Communications*, pages 242–249. IEEE, 2008.

[42] Laurent Massoulie, Andy Twigg, Christos Gkantsidis, and Pablo Rodriguez. Randomized decentralized broadcasting algorithms. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1073–1081. IEEE, 2007.

[43] Yong Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In *Proceedings of ACM international conference on Multimedia*, pages 127–136. ACM, 2007.

[44] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 325–336. ACM, 2008.

[45] Shao Liu, Rui Zhang-Shen, Wenjie Jiang, Jennifer Rexford, and Mung Chiang. Performance bounds for peer-assisted live streaming. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 313–324. ACM, 2008.

[46] Yipeng Zhou, Dah Ming Chiu, and John CS Lui. A simple model for analyzing P2P streaming protocols. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 226–235. IEEE, 2007.

[47] Fangming Liu, Bo Li, Lili Zhong, Baochun Li, and Di Niu. How P2P streaming systems scale over time under a flash crowd? In *Proceedings of the International workshop on Peer-to-Peer Systems (IPSTS)*, volume 9, 2009.

[48] Bridge Q Zhao, J Lui, and Dah-Ming Chiu. Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 271–280. IEEE, 2009.

[49] Comparing four popular online video services. `http://soft.zol.com.cn/208/2089169.html`.

[50] A comparison of the playback rates of five IPTV systems. `http://it.21cn.com/software/wlyy/2008/11/17/5481039_1.shtml`.

[51] Internet in Canada. `https://en.wikipedia.org/wiki/Internet_in_Canada`.

[52] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proceedings of ACM SIGCOMM conference on Internet Measurement*, pages 4–4. USENIX Association, 2005.

[53] Yung Ryn Choe, Derek L Schuff, Jagadeesh M Dyaberi, and Vijay S Pai. Improving VoD server efficiency with bittorrent. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 117–126. ACM, 2007.

[54] Bin Cheng, Hai Jin, and Xiaofei Liao. Supporting VCR functions in P2P VoD services using ring-assisted overlays. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 1698–1703. IEEE, 2007.

[55] Siddhartha Annapureddy, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez Rodriguez. Is high-quality VoD feasible using P2P swarming? In *Proceedings of the 16th international conference on World Wide Web*, pages 903–912. ACM, 2007.

[56] Xiaojun Hei, Yong Liu, and Keith W Ross. Inferring network-wide quality in P2P live streaming systems. *IEEE Journal on Selected Areas in Communications (JSAC)*, 25(9):1640–1654, 2007.

[57] Nadim Parvez, Carey Williamson, Anirban Mahanti, and Niklas Carlsson. Analysis of bittorrent-like protocols for on-demand stored media streaming. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 301–312. ACM, 2008.

[58] Xiaoyuan Yang, Minas Gjoka, Parminder Chhabra, Athina Markopoulou, and Pablo Rodriguez. Kangaroo: Video seeking in P2P systems. In *Proceedings of the 2009 International workshop on Peer-To-Peer Systems (IPTPS)*, volume 1, page 23, 2009.

[59] Yuan He and Yunhao Liu. VOVO: VCR-oriented video-on-demand in large-scale peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(4):528–539, 2009.

[60] Siddhartha Annapureddy, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez. Exploring VoD in P2P swarming systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 2571–2575. IEEE, 2007.

[61] Yan Yang, A Chow, Leana Golubchik, and Danielle Bragg. Improving QoS in bittorrent-like vod systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE, 2010.

[62] Delia Ciullo, Valentina Martina, Michele Garetto, Emilio Leonardi, and Giovanni Luca Torrisi. Stochastic analysis of self-sustainability in peer-assisted VoD systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1539–1547. IEEE, 2012.

[63] Delia Ciullo, Valentina Martina, Michele Garetto, Emilio Leonardi, and Giovanni Luca Torrisi. Performance analysis of non-stationary peer-assisted vod systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM), mini-conference*, pages 3001–3005. IEEE, 2012.

[64] Lucia D'Acunto, Tamás Vinkó, and Henk Sips. Bandwidth allocation in bittorrent-like vod systems under flashcrowds. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 192–201. IEEE, 2011.

[65] Miao Wang, Lisong Xu, and Byrav Ramamurthy. Linear programming models for multi-channel P2P streaming systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–5. IEEE, 2010.

[66] Bo Tan and Laurent Massoulié. Optimal content placement for peer-to-peer video-on-demand systems. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 694–702. IEEE, 2011.

[67] Yipeng Zhou, Tom ZJ Fu, and Dah Ming Chiu. A unifying model and analysis of p2p vod replication and scheduling. In *Proceedings of IEEE International Con-*

ference on Computer Communications (INFOCOM), pages 1530–1538. IEEE, 2012.

[68] Franco Robledo Amoza, Pablo Rodriguez-Bocca, Pablo Romero, and Claudia Rostagnol. A new caching policy for cloud assisted Peer-to-Peer video-on-demand services. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 43–49. IEEE, 2012.

[69] Yifeng He and Ling Guan. Solving streaming capacity problems in P2P VoD systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(11):1638–1642, 2010.

[70] Zhi Wang, Chuan Wu, Lifeng Sun, and Shiqiang Yang. Strategies of collaboration in multi-channel P2P VoD streaming. In *Proceedings of IEEE International Conference on Global Telecommunications Conference (GLOBECOM)*, pages 1–5. IEEE, 2010.

[71] Weijie Wu and John CS Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1492–1503, 2012.

[72] Wei-Cherng Liao, Fragkiskos Papadopoulos, and Konstantinos Psounis. Performance analysis of bittorrent-like systems with heterogeneous users. *Performance Evaluation*, 64(9):876–891, 2007.

[73] Alix LH Chow, Leana Golubchik, and Vishal Misra. BitTorrent: An extensible heterogeneous model. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 585–593. IEEE, 2009.

[74] Tai T Do, Kien A Hua, and Mounir A Tantaoui. P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In *Proceedings of IEEE International Conference on Communications (INFOCOM)*, volume 3, pages 1467–1472. IEEE, 2004.

[75] Anh-Tuan Gai, Fabien Mathieu, Julien Reynier, and Fabien De Montgolfier. Stratification in p2p networks-application to bittorrent. *arXiv preprint cs/0612130*, 2006.

[76] Ashwin R Bharambe, Cormac Herley, and Venkata N Padmanabhan. Analyzing and improving bittorrent performance. *Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA*, 98052:2005–03, 2005.

[77] Le Chang, Yangyang Liu, Zhonghua Wei, and Jianping Pan. Optimizing BitTorrent-like peer-to-peer systems in the presence of network address translation devices. *Peer-to-Peer Networking and Applications*, 4(3):274–288, 2011.

[78] Hongliang Yu, Dongdong Zheng, Ben Y Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 333–344. ACM, 2006.

[79] Matthew S Allen, Ben Y Zhao, and Rich Wolski. Deploying video-on-demand services on cable networks. In *Proceeding of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 63–63. IEEE, 2007.

[80] Xu Cheng and Jiangchuan Liu. Exploring interest correlation for peer-to-peer socialized video sharing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 8(1):5, 2012.

[81] Carsten Griwodz, Michael Bär, and Lars C Wolf. Long-term movie popularity models in video-on-demand systems: or the life of an on-demand movie. In *Proceedings of ACM international conference on Multimedia*, pages 349–357. ACM, 1997.

[82] Yi Zheng, Jin Peng, Qing Yu, Dan Huang, Yishuai Chen, and Changjia Chen. A measurement study on user behavior of p2p vod system. In *Proceedings of International Asia Conference on Informatics in Control, Automation and Robotics (CAR)*, volume 3, pages 373–376. IEEE, 2010.

[83] Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. Risk management for video-on-demand servers leveraging demand forecast. In *Proceedings of ACM international conference on Multimedia*, pages 1229–1232. ACM, 2011.

[84] Di Niu, Baochun Li, and Shuqiao Zhao. Understanding demand volatility in large VoD systems. In *Proceedings of ACM international workshop on Network and operating systems support for digital audio and video*, pages 39–44. ACM, 2011.

[85] Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 460–468. IEEE, 2012.

[86] Prithula Dhungel, Keith W Ross, Moritz Steiner, Ye Tian, and Xiaojun Hei. Xunlei: Peer-assisted download acceleration on a massive scale. In *Passive and Active Measurement*, pages 231–241. Springer, 2012.

[87] Chao Liang, Zhenghua Fu, Yong Liu, and Chai Wah Wu. ipass: Incentivized peer-assisted system for asynchronous streaming. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 2741–2745. IEEE, 2009.

[88] Riccardo Petrocco, Johan Pouwelse, and Dick HJ Epema. Performance analysis of the Libswift P2P streaming protocol. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 103–114. IEEE, 2012.

[89] Stefano Traverso, Luca Abeni, Robert Birke, Csaba Kiraly, Emilio Leonardi, R Lo Cigno, and Marco Mellia. Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 13–24. IEEE, 2012.

[90] Guangqing Deng and Changjia Chen. Simulation study of buffering mechanism in P2P live streaming system. In *Proceedings of IET International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN)*. IET, 2011.

[91] Linchen Yu, Xiaofei Liao, Hai Jin, and Wenbin Jiang. Integrated buffering schemes for P2P VoD services. *Peer-to-Peer Networking and Applications*, 4(1):63–74, 2011.

[92] Coupon collector's problem. `http://en.wikipedia.org/wiki/Coupon_collector's_problem`.

[93] Guangqing Deng, Chunxi Li, Changjia Chen, and Yunfei Zhang. A bitmap coding method for P2P streaming protocols. In *Proceedings of International Asia Conference on Informatics in Control, Automation and Robotics (CAR)*, volume 3, pages 368–372. IEEE, 2010.

[94] Chunxi Li, Yishuai Chen, Baoxian Zhang, Cheng Li, and Changjia Chen. A Study on Peer Startup Process and Initial Offset Placement in P2P Live Streaming Systems. In *Proceedings of IEEE International Conference on Global Telecommunications Conference (GLOBECOM)*, pages 1975–1980. IEEE, 2012.

[95] Weijie Wu, John CS Lui, and Richard TB Ma. Incentivizing upload capacity in P2P-VoD systems: a game theoretic analysis. In *Game Theory for Networks*, pages 337–352. Springer, 2012.

[96] Chi Wang, Hongbo Wang, Yu Lin, and Shanzhi Chen. A lightweight currency-based p2p vod incentive mechanism. In *Proceedings of IEEE International Joint Conference on Computational Science and Optimization (CSO)*, volume 1, pages 272–276. IEEE, 2010.

[97] Zhengye Liu, Prithula Dhungel, Di Wu, Chao Zhang, and Keith W Ross. Understanding and improving incentives in private p2p communities. In *Proceeding of IEEE International Conference on Distributed Computing Systems (ICDCS)*, volume 4, page 6. IEEE, 2010.

[98] Seedbox, Wikipedia. `http://en.wikipedia.org/wiki/Seedbox`.

[99] List of ISPs throttling P2P traffic. `http://www.p2pon.com/guides/list-of-internet-service-providers-that-throttle-p2p-traffic/`.

[100] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proceedings of ACM SIGCOMM conference on Internet Measurement*, pages 6–6. USENIX Association, 2005.

[101] Ruchir Bindal, Pei Cao, William Chan, Jan Medved, George Suwala, Tony Bates, and Amy Zhang. Improving traffic locality in BitTorrent via biased neighbor selection. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 66–66. IEEE, 2006.

[102] Yao Liu, Lei Guo, Fei Li, and Songqing Chen. A case study of traffic locality in Internet P2P live streaming systems. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 423–432. IEEE, 2009.

[103] Shansi Ren, Enhua Tan, Tian Luo, Songqing Chen, Lei Guo, and Xiaodong Zhang. TopBT: a topology-aware and infrastructure-independent bittorrent client. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE, 2010.

[104] Haiyong Xie, Y Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4P: provider portal for applications. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 351–362. ACM, 2008.

[105] Michael Piatek, Harsha V Madhyastha, John P John, Arvind Krishnamurthy, and Thomas E Anderson. Pitfalls for ISP-friendly P2P design. In *Proceedings of ACM HotNets*, 2009.

[106] Mohamed Hefeeda and Osama Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Transactions on Networking*, 16(6):1447–1460, 2008.

[107] György Dán. Cooperative caching and relaying strategies for peer-to-peer content delivery. In *Proceedings of the 2009 International workshop on Peer-To-Peer Systems (IPTPS)*, page 16, 2008.

[108] Jie Dai, Bo Li, Fangming Liu, Baochun Li, and Hai Jin. On the efficiency of collaborative caching in isp-aware p2p networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1224–1232. IEEE, 2011.

[109] Le Chang and Jianping Pan. On the system parameters of peer-to-peer video streaming with network coding. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2010.

[110] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 350–361. ACM, 2011.

[111] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, and Hui Zhang. Understanding the impact of video qual-

ity on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362, 2011.

[112] Xin Wang, Changyi Zheng, Zhenyuan Zhang, Hong Lu, and Xiangyang Xue. The design of video segmentation-aided VCR support for P2P VoD systems. *IEEE Transactions on Consumer Electronics*, 54(2):531–537, 2008.

[113] Xuanjia Qiu, Chuan Wu, Xiaola Lin, and Francis Lau. InstantLeap: fast neighbor discovery in P2P VoD streaming. In *Proceedings of ACM international workshop on Network and operating systems support for digital audio and video*, pages 19–24. ACM, 2009.

[114] Bin Cheng, Xiuzheng Liu, Zheng Zhang, Hai Jin, Lex Stein, and Xiaofei Liao. Evaluation and optimization of a peer-to-peer video-on-demand system. *Journal of Systems Architecture*, 54(7):651–663, 2008.

[115] Yu Wu, Chuan Wu, Bo Li, Xuanjia Qiu, and Francis CM Lau. Cloudmedia: When cloud on demand meets video on demand. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 268–277. IEEE, 2011.

[116] Yuan Feng, Baochun Li, and Bo Li. Airlift: Video conferencing as a cloud service using inter-datacenter networks. In *IEEE International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2012.

[117] Baochun Li, Yuan Feng, and Bo Li. Rise and fall of the peer-to-peer empire. *Tsinghua Science and Technology*, 17(1):1–16, 2012.