Spring 4-18-2011

MULTI-CHANNEL PEER-TO-PEER STREAMING SYSTEMS AS RESOURCE ALLOCATION PROBLEMS

Miao Wang
University of Nebraska-Lincoln, mwangcse@gmail.com

MULTI-CHANNEL PEER-TO-PEER STREAMING SYSTEMS AS RESOURCE ALLOCATION PROBLEMS

by

Miao Wang

A DISSERTATION

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professors Lisong Xu and Byrav Ramamurthy

Lincoln, Nebraska

May, 2011

MULTI-CHANNEL PEER-TO-PEER STREAMING SYSTEMS AS RESOURCE ALLOCATION PROBLEMS

Miao Wang, Ph.D

University of Nebraska, 2011

Advisors: Lisong Xu and Byrav Ramamurthy

In the past few years, the Internet has witnessed the success of Peer-to-Peer (P2P) streaming technology, which has attracted millions of users. More recently, commercial P2P streaming systems have begun to support multiple channels and a user in such systems is allowed to watch more than one channel at a time. We refer to such systems as multi-channel P2P streaming systems. In this dissertation, we focus on designing multi-channel P2P streaming systems with the goal of providing optimal streaming quality for all channels, termed as system-wide optimal streaming quality. Specifically, we design the systems from the perspective of how to optimally allocate resources in the whole system (e.g., bandwidth contributed by peers).

To achieve system-wide optimal streaming quality, we need to solve two fundamental problems in multi-channel P2P streaming systems, namely bandwidth allocation and block scheduling. According to measurement studies, bandwidth availability across different channels is not uniform, which means that some channels suffer from bandwidth deficit, while some others have surplus bandwidth. The bandwidth allocation problem can be defined as optimally allocating bandwidth to different channels to improve the overall streaming quality. In contrast, the block scheduling problem can be defined as optimally utilizing the allocated bandwidth for delivering useful video streams to peers before their corresponding playback deadlines. We study both problems in this dissertation.

Since there already exist many efficient block scheduling protocols, bandwidth allocation protocols for cross-channel bandwidth sharing should be flexible to adopt any of the existing block scheduling protocols. We propose an optimal bandwidth allocation protocol based on Divide-and-Conquer strategy (DAC) and a utility-based optimization model, which is flexible enough to incorporate existing block scheduling protocols and is scalable to support a large number of channels and peers. To provide guidelines for choosing the proper protocol for a specific application scenario, we compare existing and potential designs. Our results show the trade-off between bandwidth utilization efficiency and implementation complexity. When the overall system has insufficient bandwidth to support all peers, we should use admission control algorithms to reject some users. We study a class of admission control algorithms, based on the processor-sharing queueing model, which statistically guarantees that a P2P streaming system has sufficient bandwidth. The bandwidth allocation problem and the block scheduling problem are solved separately in existing works, where each problem has its own optimization objective. Therefore, from the system perspective, the optimal solution to bandwidth allocation is not necessarily the optimal solution to block scheduling and vice versa. We jointly study the two problems to improve the system-wide streaming quality. We establish general nonlinear optimization models for solving the two problems under various scenarios and apply a two-player game theoretic model to analyze the interaction between the two problems. Our analysis results establish the performance loss bounds for special applications and our packet-level simulations show the performance loss in general cases. In future, this work can be extended to other time sensitive systems.

ACKNOWLEDGEMENTS

I have been very lucky to study computer networking at the University of Nebraska-Lincoln, where I have opportunity to work with a number of outstanding professors. I am deeply indebted to my Ph.D advisors Dr. Lisong Xu and Dr. Byrav Ramamurthy. Their guidance and insights over the years have been invaluable to me. I would like to thank Dr. Sharad Seth, Dr. Witty Srisa-an and Dr. Srikanth Iyengar for serving as my Ph.D committee members and reading my dissertation. I also thank Dr. Jitender Deogun for introducing me to advanced algorithms, which was very helpful for my research and job interviews. I thank Dr. Stephen Hartke, who taught me nonlinear optimization. His interesting discussions on primal-dual optimization inspired me to conduct research on resource allocation. I also owe thanks to Zhipeng Ouyang and Peng Yang, who helped me with various problems (technical and otherwise).

Before coming to the University of Nebraska-Lincoln, I spent seven years in Xi'an Jiaotong University, Xi'an, China, where I decided to work on computer networking. I thank Dr. Xiaohong Guan, Dr. Wei Li and my former colleague Guodong Li for both introducing computer networking to me and continuous support after my graduation from Xi'an Jiaotong University.

I must thank my wife Jie Feng and my family members in China, who support me a lot both emotionally and financially. Without their endless love and encouragements I would have never completed this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Peer-to-Peer (P2P) Streaming Systems

Internet users are watching more and more videos online through video websites (e.g., Youtube [1]). Comscore [2], the leading digital media measurement company, recently released the data of users watching online videos in the U.S. during March, 2010. According to this data, over 180 millions Internet users in the U.S. watched 31.2 billion videos in March 2010 and Youtube (the world's largest online video website) accounted for about 41% of these videos. Therefore, providing smooth video streaming service to Internet users is a very important problem. Several solutions have been proposed to provide high-quality video streaming service, which fall into two categories: 1) Delivering video streams over private IP networks based on a combination of multicast and unicast (e.g., U-verse service provided by AT&T [3]); and 2) Deploying a large number of dedicated streaming servers (e.g., Youtube uses Content Delivery Network to serve millions of users). Both solutions have limitations. The private IP networks can only be used for delivering video streams within a single Internet Service Provider (ISP) and require capital investments on building

these networks. Content Delivery Network (CDN) based solutions mainly suffer the scalability problem, which means that the number of content delivery servers has to proportionally increase with the number of users to maintain continuous playback.

Nowadays, users usually access the Internet via high-speed connections (e.g., DSL, Cable), which makes it possible to deliver video streams by efficiently utilizing the user's upload bandwidth[1]. The basic design rationale of Peer-to-Peer (P2P) streaming systems is to encourage users[2] (peers) to upload video streams in their buffers to neighboring peers, while downloading fresh video streams. P2P streaming systems leverage the public Internet for video distribution and reduce server bandwidth consumption by using participating peer's upload bandwidth. Note that P2P streaming systems are different from traditional P2P file sharing systems (e.g., Bittorrent [20]) in various aspects, which lead to more challenging tasks of designing streaming systems. For example, video streams are divided into blocks and these blocks have critical time constraints (i.e., a requested block should arrive at a peer before its corresponding playback deadline).

P2P streaming systems have attracted research and development efforts, where the academia mainly focus on seeking optimal designs to achieve the theoretical performance bounds (e.g., minimizing the latency for packet delivery [50] [101], maximizing the peer's bandwidth utilization [88] [89], etc.) and the industry mainly focuses on maximizing the user perceived streaming quality with seemingly simple designs. More recently, commercial P2P streaming systems (PPLive [66], PPStream [67], UUSee [77]) have begun to support multiple channels (e.g., with UUSee client, a peer is able to choose programs from nearly 10,000 channels) and a peer

[1]In P2P literature, it is widely accepted that bandwidth bottlenecks occur at network edges instead of the network core and download bandwidth is much higher than upload bandwidth at network edges [18].

[2]We hereinafter use the terminologies *user* and *peer* interchangeably to refer to an Internet user in a P2P streaming system.

is allowed to watch more than one channel at a time. We refer to such systems as *multi-channel* P2P streaming systems. In this dissertation, we study fundamental problems in designing multi-channel P2P streaming systems.

For a clear understanding of our dissertation research, in Section 1.2, we first introduce some necessary background material on single-channel P2P streaming systems. Then, we introduce the fundamental problems in multi-channel P2P streaming systems, solved in this dissertation, in Section 1.3. Finally, we briefly highlight our contributions in Section 1.4.

1.2 Design of Single-Channel P2P Streaming Systems

In a single-channel P2P streaming systems, peers are organized into a virtual topology, called an *overlay*, where each peer maintains a set of virtual links with other peers watching the same channel. Similar to IP multicast [25], early P2P streaming designs adopt a tree-based overlay, where the streaming server serves as the root of the tree. Peers are located at different levels of the tree and peers from lower levels receive video contents from either the root (i.e., streaming server) or from peers in upper levels. The tree-based topology has two disadvantages: 1) the upload bandwidth of leaf peers cannot be utilized by others, since they do not have child peers; and 2) the tree-based overlay is not resilient to peer churn (e.g., a peer leaves the overlay or a peer suffers a connection failure). Overcast [37] and End System Multicast [19] are examples of tree-based P2P streaming systems.

To overcome the first drawback of tree-based systems, an improvement, referred to as multi-tree based systems, was proposed in [55], where the video stream is divided into substreams and each substream is delivered over its corresponding subtree. A leaf

peer of one subtree could be an internal node of another subtree. With a multi-tree overlay, the leaf peer's upload bandwidth can be used by other peers. The multi-tree based P2P streaming systems have higher utilization of peers' upload bandwidth, which, however, are not widely used in commercial deployments, mainly due to the following two reasons. First, it introduces control overhead and implementation complexity to P2P streaming systems, since this design has to maintain multiple trees overlays. Second, multi-tree systems are not resilient to peer churn either.

Due to the disadvantages of tree-based topologies, mesh-based overlays are widely used by both academia [101] [104] and industry [66] [67] [77]. In a mesh-based overlay, each peer maintains several neighbors in the system and there is no strict parent-child relationship between any two connected peers, which indicates that the two peers can download data from each other. Since a peer can download data from multiple neighbors, the mesh-based systems are resilient to peer churn. In practical implementations, when joining a mesh-based overlay, a peer first contacts with the tracker server, which maintains the information of online peers. Then, the tracker server returns a list of peer IDs (e.g., a peer's IP address) to the newly joined peer and the peer selects some of the returned peers as its neighbors. Since mesh-based overlays are resilient to peer churn and have low implementation complexity, in this dissertation, we adopt the mesh-based overlays. Note that our work can be extended to tree-based systems with minor revisions, because our established analytical models are independent of overlay structures and our simulation implementations can be changed to tree-based overlays.

1.3 Problems in Designing Multi-Channel P2P Streaming Systems

As introduced in Section 1.2, peers watching the same channel construct an overlay network, where peers utilize their upload bandwidth to deliver useful video contents to their neighbors. Therefore, the total bandwidth contributed by peers in a specific overlay is critical to the streaming quality of that channel [40] [102]. In multi-channel P2P streaming systems, each channel maintains its own overlay and a peer might join multiple overlays, in that it can watch multiple channels at a time (e.g., using the Picture-in-Picture function provided by PPStream [67]). Figure 1.1 shows an example of three channels. The overlays for the three channels are overlapped, since peers U_2, U_3, U_5 and U_6 join multiple overlays. Based on measurement studies in large-scale P2P streaming networks [32] [33], the bandwidth available across different channels is not uniform. It means that some channels have surplus bandwidth, while some others suffer bandwidth deficit. The *bandwidth allocation problem* in multi-channel P2P streaming systems is to optimally allocate bandwidth among all channels, so that the channels with surplus bandwidth help other channels suffering from bandwidth deficit. It is a fundamental problem in designing multi-channel P2P streaming systems. For example, suppose that channel A in Figure 1.1 has surplus bandwidth and channel B does not have sufficient bandwidth to support all peers watching channel B. In this case, a good design is to allocate more bandwidth of the peers joining both channels (i.e., U_2 and U_3) to channel B than to channel A.

Besides the bandwidth allocation problem, there is another equally important problem in P2P streaming systems, namely *block scheduling*, which determines how to optimally utilize the allocated bandwidth to deliver useful video streams to peers before their corresponding playback deadlines. In P2P streaming systems, the video

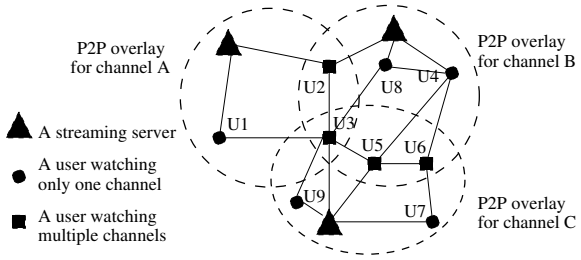

Figure 1.1: Example of three overlays for three channels.

stream is usually divided into fixed-size blocks [66] [67] [77] and peers in a channel request useful blocks (i.e., the missing blocks in their playback buffers) from their neighbor peers, which are also watching that channel. We continue to use the 3-channel case shown in Figure 1.1 to explain block scheduling. Let us take channel B as an example. Figure 1.2 shows the overlay of channel B, which is exactly the same as channel B's overlay in Figure 1.1. We omit channels A and C, in that after peers watching multiple channels have determined how much bandwidth should be allocated to channels A, B and C, the three overlays can be considered as three separated overlays. In Figure 1.2, U_4 holds blocks with sequence numbers 3 and 4 and U_5 holds block 4. If U_6 pulls blocks from U_4 and U_5, block scheduling algorithms help U_6 determine to pull which block from which neighbor, considering the bandwidth constraints on overlay links (U_4, U_6) and (U_5, U_6).

In this dissertation, we first study bandwidth allocation in multi-channel P2P streaming systems and then we provide insight into the interaction between bandwidth allocation and block scheduling. In Section 1.4, we briefly highlight our contributions in the following chapters.

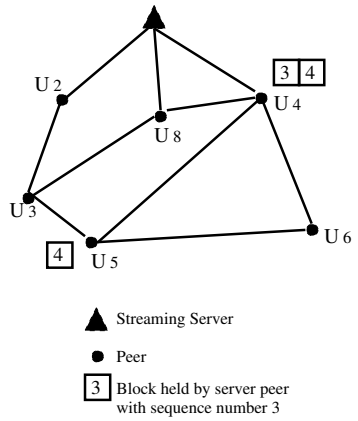

Figure 1.2: Overlay of channel B with 1 streaming server and 6 peers.

1.4 Our Contributions

Figure 1.3 shows the relationship among different chapters, where the root problem solved in this dissertation is resource allocation. Chapters 2 and 3 can be read together, in that Chapter 2 proposes a protocol corresponding to a specific design studied in Chapter 3. Chapters 4 and 5 can be read independently. Background knowledge and comparisons with existing works are covered in each chapter, when necessary. We provide an overview of each chapter below.

In Chapter 2, we study the emerging multi-view P2P streaming systems, where a user can simultaneously watch multiple channels. Previous work on multi-view P2P streaming solves the fundamental inter-channel bandwidth allocation problem at the individual peer level, which requires specific intra-channel block scheduling protocols (e.g., network coding based protocols). To provide the flexibility of adopting any existing block scheduling protocols, we propose a new protocol for multi-view P2P streaming, called Divide-and-Conquer (DAC), which efficiently solves the inter-channel bandwidth allocation problem at the channel level. Our DAC protocol is more suitable for upgrading current single-view P2P streaming systems (i.e., a peer is allowed to watch only one channel at a time) to multi-view P2P streaming systems.

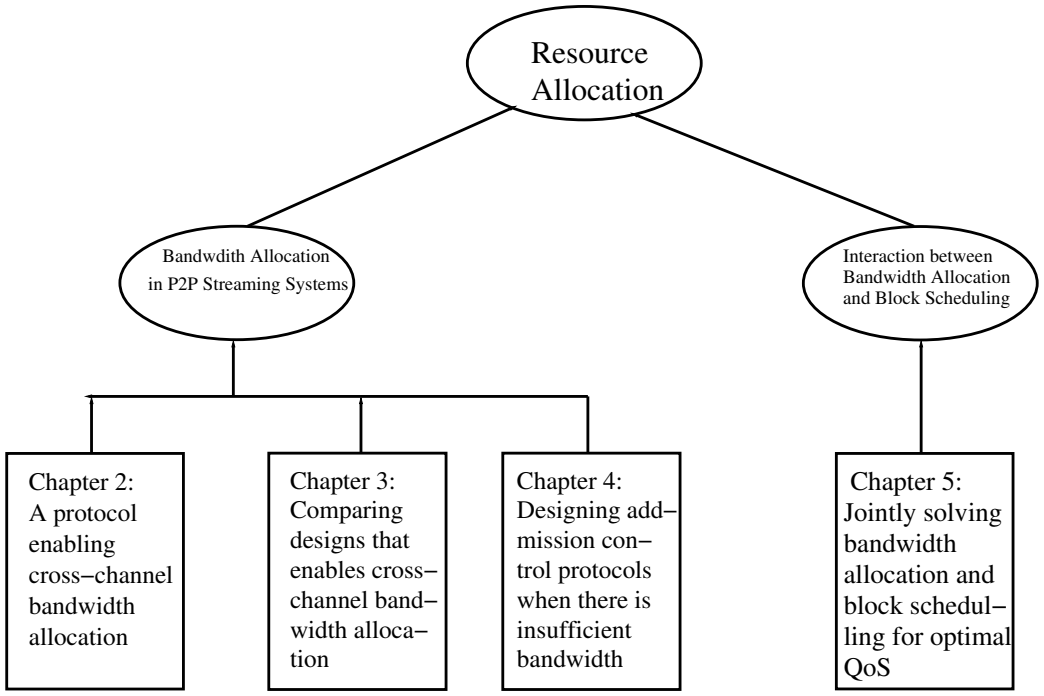

Figure 1.3: Dissertation Organization and Chapter Relationship.

Our packet-level simulations show that DAC is efficient in allocating the overall system bandwidth among different channels, is flexible in working with various block scheduling protocols, and is scalable in supporting a large number of users and channels.

In Chapter 3, we extend our work in Chapter 2 to compare existing and potential designs in multi-channel P2P streaming systems. We focus on the following fundamental problems: 1) what are the general characteristics of existing and potential designs? and 2) under what circumstances, should a particular design be used to achieve the desired streaming quality with the lowest implementation complexity? To answer the first question, we propose simple models based on network flow graphs for three general designs, namely Naive Bandwidth allocation Approach (NBA), Passive Channel-aware bandwidth allocation Approach (PCA) and Active Channel-aware bandwidth allocation Approach (ACA) respectively, which shed insight into under-

standing the key characteristics of cross-channel bandwidth sharing. For the second question, we first develop closed-form results for two-channel systems. Then, we use extensive numerical simulations to compare the three designs for various peer population distributions, upload bandwidth distributions and channel structures. Our analytical and simulation results show that: 1) the NBA design can rarely achieve the desired streaming quality in general cases; 2) the PCA design can achieve the same performance as the ACA design in general cases; and 3) the ACA design should be used for special applications.

Most of the literature on P2P streaming focuses on how to provide best-effort streaming quality by efficiently using the system bandwidth; however, there is no guarantee about the provided streaming quality. In Chapter 4, we consider how to provide statistically guaranteed streaming quality to a P2P streaming system and study a class of admission control algorithms. Our results show that there is a trade-off between the user blocking rate and user-behavior insensitivity (i.e., whether the system performance is insensitive to the fine statistics of user behaviors). We also find that the system performance is more sensitive to the distribution of user inter-arrival times than to that of user lifetimes.

The quality of service in P2P streaming systems highly depends on the protocols used for solving bandwidth allocation and block scheduling problems. Existing algorithms solve the two problems separately. However, directly combining optimal solutions to the two separate problems does not necessarily lead to system-wide optimal solutions. In Chapter 5, we seek methods of designing protocols to provide system-wide optimal quality of service in P2P streaming systems and propose design guidelines. We first establish two generic nonlinear optimization models for designing distributed protocols, which are used to solve the two problems. We also provide a detailed analysis on when and how system-wide optimal streaming quality can

be achieved via a two-player game theoretic model. Briefly, the system-wide sub-optimal streaming quality is due to the misaligned objectives of bandwidth allocation and block scheduling. Moreover, if the objectives are misaligned, the system-wide streaming quality could be arbitrarily low, though each individual problem is solved optimally. To validate our analysis, we design and implement three groups of bandwidth allocation and block scheduling algorithms using a packet-level simulator. Both our analytical models and implementations can be applied directly to design protocols for specific applications.

1.5 Our Publications

Our earlier work on neighbor selection [80] in multi-channel P2P streaming systems was presented at the IEEE International Conference on Computer Communications and Networks, IPMC Workshop 2008. Our work on multi-view P2P streaming systems [81] was presented at IEEE Peer-to-Peer Computing 2009 and a longer version has been submitted to the leading journal, Computer Networks [85]. Our work on admission control [82] was presented at ACM Network and Operating System Support for Digital Audio and Video 2009. The work of comparing multi-channel P2P streaming systems was presented at IEEE International Conference on Computer Communications 2010 [86] and IEEE International Workshop on Local and Metropolitan Area Networks 2010 [83]. A longer version of multi-channel P2P streaming comparison has been submitted to the top journal IEEE Transactions on Networking [84]. Our work on jointly solving bandwidth allocation and block scheduling has been submitted to a leading conference [87].

Chapter 2

Improving Multi-View P2P Streaming With Divide-And-Conquer

2.1 Motivation and Introduction to Multi-View P2P Streaming

Peer-to-peer (P2P) live streaming systems have been extensively studied [51], and most of the earlier works focus on single-view P2P live streaming, where a user can subscribe to and watch only one channel at a time. Multi-view[1] P2P live streaming has recently emerged, where a user can simultaneously subscribe to and watch multiple channels. For example, PPStream [67] supports a limited multi-view capability with the picture-in-picture feature.

In general, multi-view P2P streaming can be used in two types of applications: 1)

[1]We use multi-view to refer to the case where a user can simultaneously join multiple different channels. These channels are not necessarily correlated like in a multi-view video system by the video coding community.

Multi-View Internet TV Applications: Fig 2.1 illustrates a possible case where a user can enjoy a high-quality movie channel shown in a large window, while still being able to monitor the weather information on another channel displayed in a small window; 2) Multi-Camera Live Streaming Applications: Fig 2.2 illustrates a possible case where a user can watch a stock-car race from several selected cameras, such as a pit-box camera, a corner camera, and a driver's point-of-view camera.

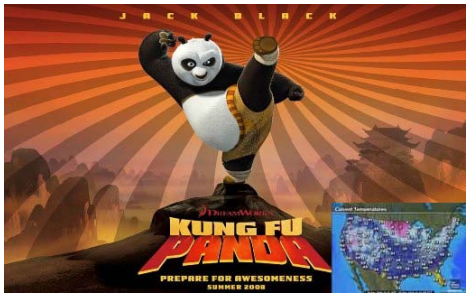

Figure 2.1: Multi-View Internet TV application.

Figure 2.2: Multi-camera live streaming of stock-car racing.

Since a peer might simultaneously watch multiple channels, its downstream neighbors from different channels will compete for its upload bandwidth, which is referred to as the inter-channel bandwidth competition problem and is unique in multi-view systems. Based on measurement studies [33] [42], the upload bandwidth of different channels in P2P streaming systems are unbalanced. Therefore, when designing multi-view systems, the inter-channel bandwidth competition problem should be optimally solved taking into account bandwidth imbalance among different channels,

which makes it fundamentally different from designing single-view systems.

While the inter-channel bandwidth competition problem is important for multi-view P2P streaming, another equally important problem is the choice of the streaming protocol used within a channel (referred to as the intra-channel streaming problem). A streaming protocol includes both an overlay construction method and a block scheduling algorithm, and it greatly influences the system streaming quality. Various streaming protocols [51] have been well studied and tested in single-view P2P streaming systems. Since most commercial systems construct a mesh-based topology which is resilient to peer churns, in this chapter, we also use a mesh overlay topology and focus on various block scheduling algorithms when studying streaming protocols.

Wu *et al.* [88] [89] first proposed a protocol for multi-view systems by solving the inter-channel bandwidth allocation problem based on game theory. Specifically, all users in a multi-view P2P streaming system participate in a decentralized collection of bandwidth auctions with the goal to optimally allocate the system bandwidth among different channels and then the peers utilize the allocated bandwidth with some intra-channel streaming protocol. However, since the inter-channel bandwidth competition problem is solved at the individual peer level, it limits choices of streaming protocols to those that can efficiently use the bandwidth allocated for each pair of users (i.e., network-coding-based streaming protocols). Their proposed approach solves both inter-channel bandwidth competition and intra-channel streaming problems all at once, referred to as AAO. In addition, decentralized auctions require message exchanges among all peers in the system, which introduces high control overhead. Even though network coding has been proven to be feasible for P2P live streaming [79], especially with relatively cheap GPU [73], few commercial P2P streaming systems actually use it due to various reasons (e.g., the implementation cost, new hardware requirements at end users, etc.). Based on the current report [77], only UUSee imple-

ments network coding. More importantly, since there is no single streaming protocol that is better than all other streaming protocols in every aspect, commercial P2P streaming systems actually use different streaming protocols for different purposes. Therefore, in this chapter, we are interested in the following problem: *Can we design a protocol for multi-view P2P live streaming that can efficiently solve the inter-channel bandwidth competition problem and is flexible enough to incorporate any streaming protocol?*

In order to design the new protocol for multi-view systems, we will solve three challenging problems, which correspond to three design goals of the new protocol.

- Flexibility problem: The system should be able to incorporate various intra-channel streaming protocols. Therefore, the new protocol should solve the inter-channel bandwidth competition problem with minimum changes at each individual peer. The AAO does not have the flexibility, since it requires each peer to participate in auctions and requires network coding to fully utilize the allocated bandwidth.

- Efficiency problem: The system should achieve a good overall streaming quality (e.g., packet delivery ratios, packet delays) for all users across all channels. Since the upload bandwidth is very important for streaming quality [40] and the bandwidth is unbalanced among different channels in P2P streaming systems [33], it requires a method that optimally solves the inter-channel bandwidth competition problem.

- Scalability problem: The new protocol should be able to maintain a good overall streaming quality in a large-scale system with a large number of channels and users.

Inspired by the *divide and conquer* strategy, DAC first divides the overall system

problem into several small channel problems, and then solves each channel problem separately. Specifically, DAC first solves the inter-channel competition problem to optimally allocate the bandwidth to different channels, and then solves the intra-channel streaming problem individually to achieve a good streaming quality for each channel.

Flexibility and scalability are achieved by the divide and conquer strategy, in that it solves the inter-channel bandwidth competition problem and intra-channel streaming problem separately and divides the large problem into several smaller problems. Since DAC solves the inter-channel bandwidth competition at the channel level, one challenge of using divide and conquer strategy is how to effectively measure the information of each channel (e.g., the total upload bandwidth demand and supply), so as to achieve a reasonably good accuracy with affordable measurement overheads. DAC uses the statistical sampling method based on continuous-time random walk, which satisfies the accuracy and overhead requirements.

For the efficiency goal, DAC allocates the upload bandwidth to different channels according to their demands via our proposed utility-based optimal resource allocation model. This model is aware of the inter-channel bandwidth competition and has a larger feasible region than [89] (refer to Section 2.3.3). We evaluate DAC with extensive and carefully designed packet-level simulations and the results show that DAC meets the three design goals well.

The rest of this chapter is organized as follows. Section 2.2 briefly summarizes the related work. Section 2.3 describes implementation details of the DAC protocol. Section 2.4 evaluates the performance of DAC with extensive packet-level simulations. Section 2.5 summarize this chapter.

2.2 Comparison With Existing Systems

2.2.1 Related work on multi-view P2P streaming

There is very little work on multi-view P2P streaming. Liang *et al.* [45] present a general framework for future IPTV based on multi-view P2P streaming, which supports content-based channel selection, multi-channel view customization, and semantics-aware bandwidth allocation. However, they only consider how to allocate the download capacity of a user to different channels. In contrast, our work focuses on how to allocate the upload capacity of a user to different channels, since the upload capacity is a more precious resource than the download capacity in the current Internet. Wang *et al.* [80] study the neighbor selection problem in multi-view P2P streaming systems and propose a simple neighbor selection algorithm based on peers' subscribed channels and upload bandwidth. However, they do not consider how to optimally allocate the upload bandwidth of peers watching multiple channels.

The most related works are [88, 89] by Wu *et al.*, which tackle the inter-channel competition and intra-channel streaming simultaneously via organizing decentralized collections of bandwidth auctions at the peer level (AAO). Even though the proposed protocol has been proved to achieve Nash Equilibrium and optimal allocation with *tight* constraints, it requires network coding for intra-channel streaming, which limits its flexibility of using the existing intra-channel streaming protocols (e.g., random block scheduling). Compared with [88, 89], we solve the two problems separately with the goal of providing a *flexible* framework for existing protocols as well as achieving a good overall performance for all channels. In Table 2.1, we compare the three protocols DAC, AAO [2] and ISO. ISO is a direct extension from single-view systems and different channels are always isolated from one another (refer to Section 2.4). Dynam-

[2]Note that AAO refers to the protocol designed by Wu *et al.* [88] hereinafter, unless explicitly explained

ics refer to the peer churn. AAO converges quickly when the total bandwidth supply is greater than the total bandwidth demand; while DAC always converges due to the relaxed constraints. There is also extensive research work on related applications, (e.g., multi-party and multi-stream systems [95, 96]), which include multi-camera video conferencing and 3D tele-immersion. They usually consider a small number of relatively stable users due to the real-time interactivity constraint, whereas our multi-view P2P streaming applications consider a large number of dynamic users.

Table 2.1: The Comparison of Three Protocols

Design Concerns	DAC	AAO	ISO
Rationale	Solving the inter-channel bandwidth competition problem and the intra-channel streaming problem separately	Solving the inter-channel bandwidth competition problem and the intra-channel streaming problem simultaneously	Solving the intra-channel streaming problem only
Flexibility	No restriction on intra-channel streaming protocol	Requiring network coding based intra-channel streaming protocol	No restriction on intra-channel streaming protocol
Efficiency	Modeling the bandwidth competition problem with *Relaxed* Constraints for a larger feasible region	Modeling the bandwidth competition problem with *Tight* Constraints for a smaller feasible region	Cannot efficiently use peers' bandwidth of all channels
Scalability	Supports a large number of peers and channels	Supports a large number of peers and channels	Supports a large number of peers and channels
Dynamics	Pause DAC in high dynamics	Use old values and loosely synchronized	-
Convergence	Yes, < 10 sec for 32 channels 20,000 peer	Conditional, < 10 sec for 4 channels 20,000 peers	-

2.2.2 Related work on intra-channel block scheduling

In general, intra-channel streaming in a P2P streaming system consists of overlay construction and intra-channel block scheduling. Since most of the commercial P2P streaming systems deployed over the Internet are mesh-based [67], which is resilient to peer churns, in this subsection, we only briefly summarize various block scheduling algorithms. Random scheduling is proposed, due to its simplicity and high performance with proper configuration [102]; Adaptive queue based chunk scheduling [30], min-cost scheduling [101], randomized decentralized broadcasting [58] are examples of optimal scheduling algorithms to fully utilize the resources and achieve the maximum streaming rate. Other scheduling algorithms include rarest-first scheduling (DONet/Coolstreaming [42]), Chainsaw [62], PRIME [56], etc. Network-coding-based streaming protocols, based on information theory, enhance the traditional block scheduling algorithms mentioned above, since they allow information mixture in peers, which simplifies the block scheduling and increases the data diversity. Wang *et al.* [79] perform a reality check for network coding and [103] proposed a market model for applying network coding. However, few (if any) commercial P2P streaming systems actually use network coding due to various reasons (e.g., it requires extra computation at end users for coding/decoding, etc.). The coexistence of different block scheduling algorithms implies that there is no single intra-channel streaming protocol that is better than all other streaming protocols in every aspect. Therefore, the flexibility to incorporate various streaming protocols is one of our primary design goals for DAC.

Besides the specific analytical models associated with intra-channel block scheduling algorithms [30] [101] [56], etc., there are some studies on modeling and evaluating different intra-channel streaming protocols [105] [58] [13]. Zhou *et al.* [105] compare the performance of two kinds of intra-channel block scheduling algorithms with a simple probability model, which is helpful for intra-channel streaming protocol design.

Massoulie *et al.* [58] propose efficient decentralized broadcasting algorithms based on the network flow model and linear programming. Bonald *et al.* [13] prove that their random peer, latest useful chunk algorithm can distribute data chunks at an optimal rate with bounded delay.

2.2.3 Related work on inter-channel cooperation and P2P streaming theory

Liao *et al.* [46] use inter-channel cooperation in their AnySee P2P streaming system to balance the resources among different channels and optimize the streaming path. Wu *et al.* [92] propose the View-Upload Decoupling approach to build multi-channel P2P streaming systems, which improves the streaming quality and reduces the channel churn. Although the problems studied by these works also exist in multi-view systems, the results of these works are mainly about single-view systems, which cannot be used directly to multi-view systems. Kumar *et al.* [40] propose a stochastic fluid model to study the fundamental performance characteristics of single-view P2P systems, which sheds insights on the relationship between system performance and the channel resource. We use their work to establish our optimal bandwidth allocation model. Wu *et al.* [93] establish queueing network models to analyze the performance of their VUD design and provide guidelines for building single-view systems with inter-channel cooperation.

2.3 The Divide And Conquer Protocol (DAC)

In this section, we introduce the proposed DAC protocol from the perspective of how DAC meets the three design goals. Section 2.3.1 first describes the inter-channel bandwidth allocation problem using a simple example and then defines two categories of

solutions, which correspond to two different designs of multi-view P2P streaming systems. We introduce the divide and conquer strategy with examples in Section 2.3.2, highlighting the design rationale of DAC to achieve flexibility and scalability. Section 2.3.3 describes the utility-based optimal bandwidth allocation model and algorithms, which mainly contribute to the goal of efficiency. In Section 2.3.4 and 2.3.5, we discuss the statistical sampling scheme for channel information measurement and the distributed method for disseminating bandwidth allocation results to users, respectively, which makes DAC scale well. Finally, Section 2.3.6 describes how DAC deals with network *dynamics* (e.g., peer joining/leaving, etc.), which is a critical issue in all P2P systems.

2.3.1 Motivating Example

In P2P streaming systems, each channel maintains its own overlay. Therefore, peers watching multiple channels (views) simultaneously join more than one overlay, which makes the overlays for these channels overlap with each other, as shown in Fig 2.3. Peers joining multiple overlays can contribute their upload bandwidths to several overlays, which implies that the peers should determine how to allocate their upload bandwidth to different overlays (the *bandwidth allocation problem*). For example, U_3 joins three overlays and therefore it can contribute its upload bandwidth to three channels. Based on the measurement studies [32] [92] [33], the channels in a P2P streaming system have imbalanced upload bandwidth (i.e., some channels have surplus upload bandwidth; some other channels suffer bandwidth deficit). Thus, an optimal bandwidth allocation strategy should be aware of the bandwidth imbalance among different channels, which can be described by a utility-based optimization model and will be introduced in Section 2.3.3.

Generally, the designs for multi-view P2P streaming systems fall into two cate-

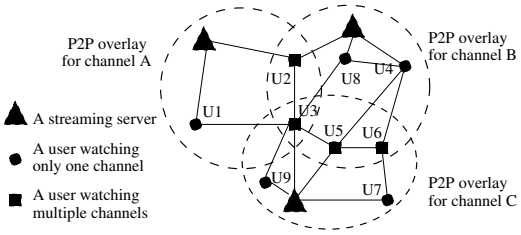

Figure 2.3: The overlapping overlays for a multi-view system with three channels.

gories according to the method used for solving the bandwidth allocation problem. Peer-level design (e.g., AAO [88]), where each peer individually determines how to allocate its upload bandwidth to its neighboring peers. *Peer-level* design requires specific intra-channel streaming protocols (i.e., network coding based protocols) to control the utilization of allocated bandwidth between the peer and its neighbors, which limits the flexibility of the design. Channel-level design (e.g., our DAC protocol), where the group of peers watching the same set of channels makes the same bandwidth allocation decision based on the bandwidth demand and supply relationship of subscribed channels. It means that every peer in the same group contributes the same faction of its bandwidth instead of the same amount. For example, U_5 and U_6 watch channels B and C. If peer-level design is used, U_5 and U_6 individually determine how to allocate their upload bandwidths. By contrast, with channel-level design, they make the bandwidth allocation decision together. Our DAC protocol is a channel-level design and *does not* require the accurate bandwidth utilization control between the peer and its neighbors, which provides flexibility of incorporating existing intra-channel streaming protocols. Note that *accurate bandwidth utilization* means that for a given set of neighbors, the peer controls the bandwidth utilization of each individual neighbor, which is a drawback of the existing AAO protocol [89]. In Section 2.3.2, we will introduce the key rationale of our DAC protocol.

2.3.2 Divide and Conquer Strategy

We explain the basic idea of DAC with the example shown in Fig 2.4. There are a total of three channels: A, B, and C. Some users watch only a single channel, and some users watch multiple channels. All users watching the same channel form a single P2P overlay for the channel, and there are some overlaps between different P2P overlays as shown on the left side of Fig 2.4. We do not show the overlay topology for each channel (i.e., how the users in a P2P overlay are connected to each other), in order to emphasize that DAC has no specific requirement on the topology of a P2P overlay.

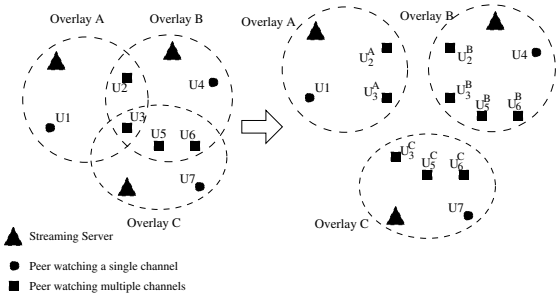

Figure 2.4: DAC splits three *physically overlapping* P2P overlays into three *logically disjoint* P2P overlays.

To achieve flexibility and scalability, DAC follows the divide-and-conquer strategy to divide the overlapped overlays into different independent overlays (corresponding to different channels). Then, it solves the inter-channel competition at the channel level, which is different from [88, 89], which solve the problem at the peer level. Therefore, DAC does not have any specific requirement for intra-channel streaming protocols. For example, DAC splits three *physically overlapping* P2P overlays into three *logically disjoint* P2P overlays as shown in Fig 2.4. User U_2 is split into two logical users U_2^A and U_2^B, each of which has its own upload capacity and does not interfere with one another. Note that the upload capacity of physical user U_2 is the sum of the upload capacities of logical users U_2^A and U_2^B.

2.3.3 Optimal Bandwidth Allocation

To achieve the efficiency design goal, DAC properly allocates the peers' upload bandwidth to their subscribed channels by considering competitions for upload bandwidth among these channels due to their upload bandwidth imbalance [88]. As previously mentioned, DAC solves the bandwidth allocation problem at the channel level based on the divide and conquer strategy. Therefore, we first describe how DAC efficiently splits multiple physically overlapping P2P overlays into multiple logically disjoint P2P overlays by efficiently splitting each physical user into multiple logical users, one for each subscribed channel.

In order to use the divide and conquer strategy and achieve better scalability, DAC makes the splitting decision for a group of users who watch the same set of channels, instead of considering the splitting decision for each individual user. For example, in Fig 2.4, DAC considers the splitting decision for both U_5 and U_6 together, since both of them watch channels B and C. Let Θ denote the set of all channels. For example, $\Theta = \{A, B, C\}$ for Fig 2.4. For a subset of channels $\theta \subseteq \Theta$, let S_θ denote the set of users, who are watching *just* the channels in channel set θ. As an example, if $\theta = \{B, C\}$, then user set S_θ (also written S_{BC}) denotes the set of users watching just channels B and C, and in Fig 2.4, $S_{BC} = \{U_5, U_6\}$. A streaming server is considered as a special user who only contributes its upload capacity and belongs to the corresponding user set.

For each user set S_θ, DAC considers how to optimally allocate the total bandwidth of all users in S_θ to all channels $c \in \theta$. Intuitively, a user set S_θ provides its upload bandwidth to some channels and a channel c requests bandwidth from some user sets, therefore, we call a user set *a bandwidth supplier* and a channel *a bandwidth consumer*. The relationship between suppliers and consumers can be described by a bipartite resource allocation graph $G = (S, D, E)$, where vertex set S is the set

of all suppliers (i.e., S contains S_θ for any $\theta \subseteq \Theta$), vertex set D is the set of all consumers (i.e., D contains c for any $c \in \Theta$), and edge set E represents the supplier-consumer relationship (i.e., $e = (S_\theta, c) \in E$ iff $c \in \theta$). Fig 3.1 illustrates the bipartite graph with 7 suppliers and 3 consumers for a multi-view P2P streaming system with M=3 channels: A, B, and C. For example, supplier S_{BC} allocates its bandwidth to consumers B and C.

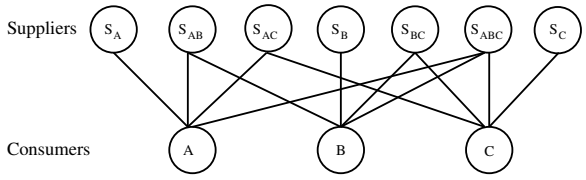

Figure 2.5: A resource allocation graph for a multi-view P2P streaming system with three channels A, B and C.

2.3.3.1 Optimal Bandwidth Allocation Model

With the resource allocation graph $G = (S, D, E)$, we can model the upload bandwidth allocation problem as solving the global optimization problem below

$$\max_{a \geq 0} \sum_{(\theta,c) \in E} U_c^\theta(a_c^\theta) \tag{2.1}$$

subject to

$$\sum_{c \in \theta} a_c^\theta \;\leq\; B^\theta \qquad \forall \theta \tag{2.2}$$

where B^θ is the total upload bandwidth of all users in S_θ, and a_c^θ is the bandwidth to be allocated from supplier S_θ to consumer c. $U_c^\theta(\cdot)$ is the utility function[3] of c associated with bandwidth obtained from S_θ. The constraint means that the total allocated bandwidth from supplier S_θ cannot exceed its total upload bandwidth.

[3]A utility function of channel c maps the allocated bandwidth into the streaming quality of that channel, which is a non-decreasing function of allocated bandwidth.

Compared with [88], we relax the constraint that the total allocated bandwidth should be greater than or equal to the desired bandwidth by each consumer, in order to guarantee the convergence of the algorithm in the case of bandwidth fluctuations. The measurement of P2P networks [71] shows that the upload bandwidth fluctuates frequently due to congestion, jitter, etc. of the underlying physical network and peer dynamics (e.g., joining/leaving the overlay). Therefore, the model for bandwidth allocation should consider the bandwidth fluctuation. Otherwise, the convergence of the allocation algorithm corresponding to the model will be affected when the fluctuation causes violations of the constraints.

We determine the utility function as follows: the utility obtained by each channel should be non-decreasing with respect to the allocated bandwidth. To achieve the efficiency goal, the solution to problem 2.1 allocates bandwidth based on the demand of each consumer (i.e., to solve the competitions among different consumers). In order to make 2.1 a computationally solvable problem, we follow [72] to assume the utility function be an increasing and twice differentiable concave function, due to two reasons: 1) The utility function of multimedia application is an increasing and concave function of bandwidth [33] [72]; and 2) A twice differentiable function simplifies analysis of our nonlinear optimization model [54]. The utility function used here is formulated as

$$U_c^{\theta}(a_c^{\theta}) = R_c \log(1 + a_c^{\theta})$$

where R_c represents the c's bandwidth demand and the utility function is always non-negative. Moreover, due to the strict concavity of the logarithmic function used in the above utility function, the optimal bandwidth allocation strategy to convex program 2.1 is proportionally fair [39], which means that the solution to 2.1 allocates bandwidth based on each channel's demand. In addition, since the above model allocates bandwidth based on R_c for channel c, we can use R_c to determine the priority

of that channel by multiplying R_c by a priority constant. For example, if channel A has a higher priority than channel B, we multiply R_A by a priority constant 1.2 and multiply R_B by 0.9. Our proposed protocol DAC uses the sampling method to determine R_c and B^θ, which is scalable and will be described in following sections. In the next subsection, we propose a distributed algorithm for the global optimization problem 2.1 for a large-scale system.

2.3.3.2 Algorithms for Solving Convex Problem 2.1

The distributed solution to problem 2.1 is based on the standard dual decomposition [64], referred to as *Dual-Algorithm* in the remaining of the chapter. Before developing the Dual-Algorithm, we first establish the Lagrangian of 2.1

$$\begin{aligned} L(\mathbf{a}, \boldsymbol{\lambda}) &= \sum_{e \in E} U_c^\theta(a_c^\theta) + \sum_\theta \lambda^\theta (B^\theta - \sum_{c \in \theta} a_c^\theta) \\ &= \sum_{e \in E} [U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta] + \sum_\theta B^\theta \lambda^\theta \\ &= \sum_{e \in E} L_{c,\theta}(a_c^\theta, \lambda^\theta) + \sum_\theta B^\theta \lambda^\theta \end{aligned} \tag{2.3}$$

where $e = (S_\theta, c)$ (we use (θ, c) to represent the edge $e = (S_\theta, c)$ hereinafter) is an edge in the resource allocation graph indicating the bandwidth that c obtains from S_θ, $\lambda^\theta \geq 0$ is the Lagrange multiplier (bandwidth price of multi-view user set S_θ) associated with the linear capacity constraint (2.2) of S_θ, and $L_{c,\theta}(a_c^\theta, \lambda^\theta) = U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta$ is the Lagrangian associated with the edge (θ, c) to be maximized on that edge by the consumer c.

Based on the dual decomposition, each edge $e \in E$ whose starting vertex is c, for the given λ^θ, solves

$$a_c^{\star\theta}(\lambda^\theta) = \arg\max_{a \geq 0} [U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta] \qquad \forall c \tag{2.4}$$

which is unique due to the strict concavity of $U_c^\theta(\cdot)$. The master dual problem which determines the bandwidth price of S_θ, is

$$\min_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}) = \sum_c g_c(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \mathbf{B} \tag{2.5}$$

subject to

$$\boldsymbol{\lambda} \geq 0 \tag{2.6}$$

where $g_c(\boldsymbol{\lambda}) = L_{c,\theta}(a_c^{\star\theta}(\lambda^\theta), \lambda^\theta)$. The unique solution to 2.4 indicates that the dual function $g(\boldsymbol{\lambda})$ is differentiable and therefore there exists a gradient method that updates the λ^θ at each iteration

$$\lambda^\theta(t+1) = [\lambda^\theta(t) - \alpha(B^\theta - \sum_{(\theta,c)\in E} a_c^{\star\theta}(\lambda^\theta(t)))]^+ \ \forall\theta \tag{2.7}$$

where t is the iteration index, $\alpha > 0$ is the step size, $[\cdot]^+$ represents the nonnegative orthant projection.

Theorem 1. *The Dual-Algorithm solves the problem 2.1 in a distributed manner.*

Proof Sketch: Due to the concavity of the utility function, the duality gap for problem 2.1 is zero. Therefore, the dual variable $\boldsymbol{\lambda}(t)$ converges to $\boldsymbol{\lambda}^\star$ as $t \to \infty$. The solution to 2.4 has a unique solution and the primal variable $a_c^{\star\theta}(\boldsymbol{\lambda}(t))$ will converge to the primal optimal variable $\boldsymbol{a}^\star$. Detailed proofs for the convergence of concave maximizations are available in [54]. Problem 2.4 can be independently solved at the edges at each consumer (channel) and the gradient based update 2.7 can be independently carried out at each supplier (user set). □

We summarize the algorithms carried out at the consumers and suppliers at round t in Algorithms 1 and 2.

/*The consumer determines how much bandwidth should be obtained from S_θ based on λ^θ. Since $a_c^{\star\theta}$ can be updated independently by solving problem 2.4, the algorithm does not need synchronization mechanism to receive all λ^θ simultaneously.*/
INPUT: λ^θ
OUTPUT: updated $a_c^{\star\theta}$
On receiving update messages of λ^θ;
foreach θ, *such that edge* $(\theta, c) \in E$ **do**
    update $a_c^{\star\theta}$ by solving the problem 2.4;
    submit the updated $a_c^{\star\theta}$ to corresponding supplier S_θ;
end

Algorithm 1: Consumer c at round t

/*The supplier updates the $\lambda^\theta(t)$ based on each consumer's new bandwidth demand $a_c^{\star\theta}$ and $\lambda^\theta(t-1)$. Therefore, $\lambda^\theta(t)$ can be considered as the bandwidth price at S_θ. To simplify implementation, S_θ waits for a period T to receive updates from each consumer. If update from a specific consumer is not received, the old valued is used.*/
INPUT: $a_c^{\star\theta}, \quad \forall(\theta, c) \in E$
OUTPUT: updated $\lambda^\theta(t)$
Waiting for a period T to receive update messages of $a_c^{\star\theta}, \quad \forall(\theta, c) \in E$;
Independently update $\lambda^\theta(t)$ with the $\lambda^\theta(t-1)$ and the updated $a_c^{\star\theta}$;
foreach c, *such that edge* $(\theta, c) \in E$ **do**
    Send the new $\lambda^\theta(t)$ to consumer c;
end

Algorithm 2: Supplier S_θ at round t

2.3.3.3 Discussion

What if there are a large number (i.e., M) of channels and then a large number (i.e., 2^M) of suppliers? Notice that each supplier S_θ with $|\theta| = 1$ has only one consumer, so it does not need to run the allocation algorithm 2 and it can directly allocate all of its bandwidth to the consumer. Furthermore, in order to achieve better scalability, only if a supplier has a large enough number of users, does it run the bandwidth allocation algorithm (i.e., the set of users has sufficiently large impact on the system performance). Specifically, supplier S_θ runs allocation algorithm 2, only if $N_\theta/N > \alpha$, where N_θ is the number of users in S_θ (i.e., $N_\theta = |S_\theta|$), N is the total number of users across all channels (i.e., $N = \sum_{\theta \subseteq \Theta} N_\theta$), and α is a system parameter. Note that this implies that there are at most $1/\alpha$ suppliers running the allocation algorithm. For example, if $\alpha = 0.001$, then there are at most $1/\alpha = 1000$ concurrent allocations in the system. If supplier S_θ does not run an allocation, it directly allocates its bandwidth to its consumers in proportion to their corresponding streaming rates (i.e., r^c for consumer c). For a system with a large number of channels, this method can significantly reduce the total number of concurrent allocations while not greatly affecting the system efficiency (based on our simulation, for 32 channels with 20,000 peers, it converges within several seconds). The value of α is determined by the required accuracy of the bandwidth allocation. Smaller α provides better accuracy due to better approximation of the bandwidth allocation in the system.

What if there is insufficient bandwidth for the system? In case of insufficient bandwidth, the system either suffers a degraded quality of service, if all the channels are considered equally important, or provides differentiated quality of service depending on the priorities of different channels. The proposed bandwidth allocation program 2.1 has the potential to provide differentiated QoS, in that we can change the order of utility functions based on the priority of each channel. Therefore, channels

with higher priorities have the privileges to obtain more bandwidth to sustain their service quality than those with lower priorities.

How to implement the concurrent allocations? We require that there will be a small group of dedicated allocation servers in the system, each handling multiple suppliers. The total number of allocation servers is proportional to the value of $1/\alpha$. We expect that very few allocation servers will be necessary for a small value of $1/\alpha$, such as 1000. Nevertheless, additional allocation servers can provide better fault tolerance. The tracker server (bootstrap server) of each channel can act as the consumer for the channel. Since it only needs to communicate with a small group of allocation servers for at most $1/\alpha$ allocations during each allocation round, we do not expect that this would overload the tracker server.

2.3.4 Measuring System Information Required by the Allocations

In this section, we describe our design choices and implementation details on how to measure the system information required by the bandwidth allocation.

2.3.4.1 Information to measure

To allocate its upload bandwidth, supplier S_θ must know B^θ which is the total upload bandwidth of all users watching just the channels in θ. To determine whether to allocate it bandwidth, supplier S_θ must know N_θ and N (i.e., whether $N_\theta/N > \alpha$), where N_θ is the total number of users watching just the channels in θ and N is the total number of users in the system. The consumer c uses formula $R_c = N_c \times r_c \times \gamma$ to determine the bandwidth demand of channel c, which is used in the utility function, where r_c is the streaming rate of channel c and γ is a scalar for considering the control overhead required by intra-channel streaming protocols (e.g., random block

scheduling achieves near optimal performance with $\gamma \geq 1.1$ [102]). Since r_c and γ are known for channel c, consumer c only needs to measure N_c which is the total number of users watching channel c.

2.3.4.2 Design Choices

One straightforward method to measure the above required information is to use a distributed information management system, such as DHT-based RandPeer [44], to keep track of the information of all users. This method can accurately measure the required information. However, in order to keep track of the information of dynamic users (joining/leaving/failure), this method generates a significant amount of traffic overhead between users and the information management system. It is not scalable to systems with a large number of users and channels.

Instead of directly keeping track of the information of all users, DAC adopts a sampling method that statistically measures the information of all users with a reasonably good accuracy and an affordable traffic overhead. Sampling methods [27, 75, 57] have been studied recently for selecting peers uniformly at random from a P2P overlay. The difficulty lies in how to select peers uniformly at random in a dynamic and heterogeneous P2P overlay, where peers may join and leave the overlay and have different numbers of neighbors. There are two types of unbiased sampling methods: Metropolized Random Walk with Backtracking (MRWB) [75] method based on Metropolis-Hastings method for Markov Chains, and Sample and Collide (S&C) [57] method based on Continuous Time Random Walk. While both the MRWB method and the S&C method can be used to uniformly sample the information of peers, the S&C method can also be used to estimate the total number of peers in a group. Therefore, DAC chooses the S&C method to measure the required information.

2.3.4.3 Implementation Details

Considering that $B^\theta = N_\theta \times \overline{b^\theta}$ where $\overline{b^\theta}$ is the average upload capacity of every user in S_θ, DAC first measures N_θ and $\overline{b^\theta}$, and then calculates B^θ as $N_\theta \times \overline{b^\theta}$. Overall, DAC needs to measure four types of information: N, N_c for any $c \in \Theta$, N_θ for any $\theta \subseteq \Theta$, and $\overline{b^\theta}$ for any $\theta \subseteq \Theta$.

There are a few sampling servers in the system (for fault-tolerant reasons), which are responsible for statistically measuring all the required information, and periodically reporting them to each consumer and each supplier. Below we first explain how the sampling server measures N, and then explain how it measures N_c, N_θ and $\overline{b^\theta}$.

Every Δt time interval, the sampling server uses the S&C method [57] to measure N as described below.

- The sampling server randomly identifies a series of users in the system as initiators.
- Each initiator initiates a continuous-time random walk, which may cross different channels if a visited user watches multiple channels. The random walk will finally stop at a uniformly selected user.
- Each selected user reports its information such as its unique user ID, its upload capacity, and its subscribed channels to the sampling server.
- The sampling server keeps identifying new initiators until it obtains the information of n selected users such that there are exactly β pairs of equal user IDs among these n selected users (called β collisions in [57]).
- Finally, N can be estimated by solving the following equation with the standard bisection search.

$$\sum_{i=0}^{n-\beta-1} \frac{i}{N-i} - \beta = 0 \tag{2.8}$$

According to the theory of the S&C method, any user in the system can be identified as an initiator, and it does not need to be uniformly selected. Therefore, any standard P2P neighbor selection method can be used to identify an initiator. However, in practice for better accuracy, we try to identify users in different channels and in different locations. Note that, the sampling server can identify a series of initiators back to back, so that multiple continuous-time random walks can be performed by different initiators simultaneously. Parameter β is a system parameter determining the accuracy of the estimation and the overhead of sampling traffic. The larger the size of a P2P system, the bigger the value of β to maintain a certain degree of accuracy. For example, based on our simulation results, for a P2P system with 20,000 users, $\beta = 50$ can achieve a good estimate with less than $\pm 10\%$ error and a light-weight sampling traffic.

The information N_c, N_θ, and $\overline{b^\theta}$ can be measured simultaneously while the sampling server is measuring N. Recall that the sampling server selects n users uniformly at random in the system, and it knows all the information of these n users. Therefore, N_c can be estimated by the product of N times the percentage of n peers watching channel c, N_θ can be estimated by the product of N times the percentage of n users watching just the channels in θ, and $\overline{b^\theta}$ can be estimated by the average upload capacity of all users (among these n users) watching just the channels in θ. We can see that it does not require any extra sampling traffic to measure N_c, N_θ, and $\overline{b^\theta}$.

2.3.5 Distributing Allocation Results to Users

Because there are at most $1/\alpha$ concurrent allocations at the suppliers, the proposed allocations converge very quickly. Every Δt time interval, DAC distributes only the final allocation results (i.e., the results at the optimal point) but not the intermediate results to users, in order to reduce the overhead of control traffic. DAC encapsulates

the final results into a single packet, which is then distributed to all users in the system by using any standard gossip-style protocol [24]. Note that the allocation servers do not have to directly distribute the allocation results to all users; instead, they send the results to some randomly selected peers and the selected peers spread the results with the epidemic style update, which guarantees that all peers receive the results in $O(\log(N))$ (N is the total number of peers in the system) rounds with high probability [24]. Even though this packet contains the result of each allocation, its size is not very large since there are at most $1/\alpha$ concurrent allocations. When a user in S_θ receives the packet, it checks whether the packet contains the result of allocation for S_θ. If so, it allocates its upload capacity among its subscribed channels according to the received allocation result; otherwise, it allocates its upload capacity among its subscribed channels proportional to their corresponding streaming rates (i.e., r^c for channel c). When a new user joins the system or when an existing user changes its subscribed channels, it also allocates its upload bandwidth to its subscribed channels proportional to their streaming rates until it receives a packet containing the allocation results.

2.3.6 DAC Dynamics

An important feature of a real P2P system is user dynamics. A user may randomly join or leave the system, and change its subscribed channels. To deal with user dynamics, DAC *periodically* performs the divide-and-conquer strategy to divide the system into different sets of logically disjoint P2P overlays at every Δt time interval as illustrated in Fig 2.6. The response time period Δt is a system parameter, which depends on how long DAC takes to perform the divide-and-conquer strategy, how much control overhead DAC generates, and how dynamic the system is. DAC sets Δt on the order of minutes, for example 2 minutes in our simulations, for the following

reasons: 1) It takes only a short time on the order of seconds for DAC to perform the divide-and-conquer strategy, and then it introduces only a light-weight control overhead for performing DAC once every Δt time interval which is on the order of minutes. 2) Recent P2P measurement studies [31, 78, 42] show that a P2P system is relatively stable over an interval of minutes, because most users have a lifetime longer than a minute, and at any time instant a significant percentage of users (e.g., $> 70\%$ on average reported in [78]) even have a lifetime on the order of hours. On average, the percentage change of a P2P system population in a minute [31, 42] is usually less than 1%. Therefore, we believe that a Δt on the order of minutes is fast enough for DAC to respond to user dynamics.

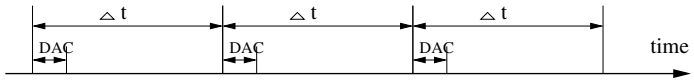

Figure 2.6: DAC periodically performs the divide-and-conquer strategy every Δt time interval in response to user dynamics.

When a new user joins the system or when an existing user changes its subscribed channels, it allocates its bandwidth to its subscribed channels in proportion to their streaming rates until it receives a divide-and-conquer result from DAC. In the special cases when a large number of users simultaneously join or leave a P2P overlay (e.g., at the beginning and the end of a program), DAC can detect the sharp population change and then temporarily pause the divide-and-conquer strategy until the system population becomes relatively stable. Therefore, even in these special cases, a system with DAC should perform at least as good as a system without DAC.

2.4 Simulation Results

In this section, we use packet-level simulations to evaluate the performance of DAC protocol.

2.4.1 Simulation Setup

We develop a packet-level, event-driven multi-view P2P streaming simulator based on the event-driven architecture of P2PStrmSim, which is a single-view P2P streaming simulator originally developed by Zhang [99]. Each channel organizes a mesh overlay [56] and uses the pull-based data request strategy [102], where peers periodically exchange buffermaps with each other and request missing data chunks based on the received buffermaps. In order to evaluate the streaming quality, the buffer at each peer is carefully simulated and two block scheduling algorithms are implemented, which are the random block scheduling and the mincost block scheduling [100]. The default buffer size is 20 seconds of video chunks. By default, a peer can have a maximum of 15 neighbors for each watched channel (e.g., suppose that a peer watches channels A and B, it can have 15 neighbors in channel A and 15 neighbors in channel B). For end-to-end latency setup, we use a real-world latency matrix (2500×2500) [4] obtained by measuring a group of DNS servers. Since the pairs of peers are more than 2,500, we randomly select a latency value from the matrix for each pair of peers. The average end-to-end latency is 75 ms. By default, the buffermap exchange interval is set to 1 second and the data chunk request interval is set to 0.5 second, which are suggested by [102] by considering the streaming quality and control overhead trade-off. For a typical simulation with 20,000 peers with 4 channels, the running time is about 1 day on a Linux server with 8 2.2GHz CPUs and 8 GB RAM. We also implement the S&C sampling algorithm used for the information estimation. To evaluate the scalability of our DAC protocol, we enhance the simulator to support up to 32 channels and 100,000 peers. Moreover, we provide interfaces to load the real trace collected by GridMedia [99], a real implementation of push-pull mesh-based P2P streaming system, which can support up to 224,453 concurrent users [102] (reported in 2006).

We simulate three protocols for the inter-channel competition problem in multi-view P2P streaming: 1) Our DAC protocol based on the divide-and-conquer strategy; 2) The protocol [88] by Wu *et al.* (referred to as AAO) based on the all-at-once strategy; 3) A reference protocol in which each user always allocates its upload capacity to its subscribed channels in proportion to their streaming rates (referred to as ISO) so that different channels are always isolated from one another. For all three protocols, we construct an overlay with a mesh topology for each individual channel. Since we are interested in the capability of DAC and AAO in supporting various block scheduling algorithms such as simple random and optimization-based algorithms, we simulate two blocking scheduling algorithms: random scheduling representing simple scheduling algorithms, and min-cost scheduling [101] representing optimization-based scheduling algorithms.

In order to evaluate our proposed DAC protocol, we have to configure three groups of parameters, which are different from simulations in single-view systems [102]). The three groups of parameters are: 1) the DAC protocol parameters; 2) the system bandwidth information parameters; 3) the channel and peer information parameters.

The DAC protocol parameters include the time interval Δt for running DAC (default value is 2 min; the influence of Δt is studied in Section 2.4.4.3), the system parameter α for the maximal number of concurrent allocations (default value is 0.001), the scalar γ for intra-channel streaming control overhead (default value is 1.1 [102]), and the collision number β for S&C sampling (default value is 50, selected based on our Group I simulations).

The system bandwidth information parameters include the streaming rate r_c for each channel c (default value $300Kbps$) and the *resource index* for each channel (the total upload bandwidth over the total required bandwidth [102] in that channel. The resource index varies in each group of simulations and will be provided separately).

Note that the resource index for each channel is calculated based on the ISO protocol, which allocates multi-view peers' upload bandwidth based on the streaming rate of each subscribed channel. To achieve the desired resource index, we change the fraction of peers with upload bandwidth of 3 Mbps, 1Mbps, 784Kbps, 300Kbps, and 200Kbps. As in [89] [102], we assume that the peer's download bandwidth is enough for sustaining the channel's streaming rate. In order to calculate the bandwidth of each channel, we implement the sampling method proposed in Section 2.3.4.3 to estimate the bandwidth demand and supply of each channel (i.e., we estimate the number of peers and the average upload bandwidth of each channel, with which we can calculate the bandwidth demand and supply of that channel). In addition, similar to [89] [102], during the simulation the peer's upload bandwidth does not change, but it can dynamically join/leave that channel, which influences the bandwidth demand and supply of that channel. Please refer to Section 2.4.4.3 for discussions on peer dynamics.

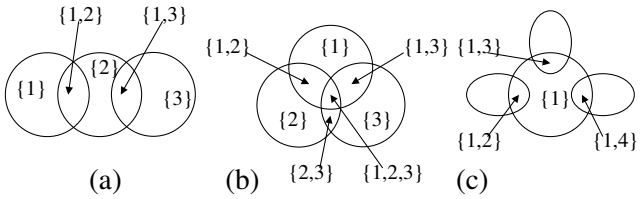

Figure 2.7: Three types of channel structures: a) chain, b) mesh, and c) star.

The channel and peer information parameters include the number of channels M, the total number of peers N, the channel structure, and beta distribution parameters y, z. Beta distribution is a general type of statistical distribution, with the probability function $P(x) = \frac{(1-x)^{z-1}x^{y-1}}{B(y,z)}$, where $B(y,z)$ is the beta function defined as $B(y,z) = \frac{(y-1)!(z-1)!}{(y+z-1)!}$ [9]. We simulate a multi-view P2P system with one of the following three types of channel structures illustrated in Fig 2.7: a) a chain structure where a user can view only the streams from either a single camera or two consecutive cameras in

a row of cameras. b) a mesh structure where every user watches a random number of channels, and c) a star structure where there is one popular channel that every user watches. The population of each channel set is determined as follows: we first arrange the channel sets in lexicographical order and then assign a channel set a fraction f of the total number of peers N, which means the number of peers watching that channel set is $f * N$. We use the beta distribution to determine the fraction. Fig 2.8, 2.9, 2.10 illustrate the shapes of population distributions for chain, mesh, and star channel structures simulated in this chapter with a small number of channels as an example. For example, as shown in Fig 2.8, we first generate the channel sets with 3 channels and a chain channel structure (i.e., channels sets $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}$) and arrange them in lexicographical order in the x-axis. Then we generate the beta distribution with parameters $(2, 2)$ and assign the each channel set a value based on the distribution. Finally, we calculate the fraction of peers watching a specific channel set according to the rule above. Since the shape of the population distribution is determined by beta distribution, we will give the channel structure with parameters for the beta distribution in each group of simulations below.

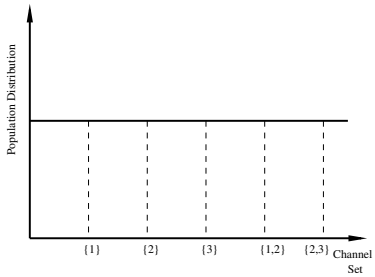

Figure 2.8: Population distribution of chain structure with 3 channels, beta distribution with parameters (1,1).

Our simulation results fall into four categories based on different evaluation motivations for DAC. Group I: we evaluate the accuracy and the overhead of sampling and then the impact of different sampling parameters on DAC. This group of sim-

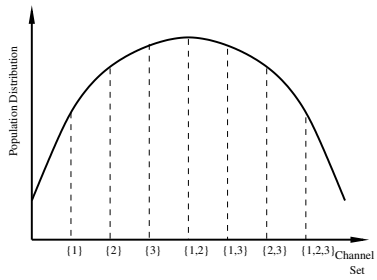

Figure 2.9: Population distribution of mesh structure with 3 channels, beta distribution with parameters (2,2).

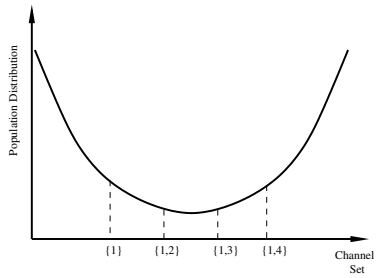

Figure 2.10: Population distribution of star structure with 4 channels, beta distribution with parameters (0.8, 0.8).

ulation results is also for selecting proper sampling parameter for Group II and III. Group II: we evaluate the flexibility of DAC compared with AAO using the two block scheduling algorithms. Group III: we conduct the comprehensive performance evaluation of DAC compared with ISO. Group IV: we evaluate the intra-channel streaming quality of DAC with various metrics defined in [102], which aims to show that DAC can provide good streaming quality with low control overhead for each channel.

To compare the performance of DAC with ISO and AAO, we measure the packet delivery ratio of the system. The *packet delivery ratio* of a user for channel c is defined as the ratio of the total number of packets of channel c received by the user before the playback deadline to the total number of packets sent by the streaming server of channel c. The packet delivery ratio of channel c is defined as the average delivery ratio of all users watching channel c, which means that we use the worst channel's performance to represent the system performance. Finally, the packet delivery ratio of

the system is defined as the lowest delivery ratio among all channels. Intuitively, this is because the satisfaction of a user watching multiple channels is usually determined by the channel with the worst quality.

2.4.2 Group I: Impact of the Sampling Method on DAC

2.4.2.1 Sampling accuracy

Fig 2.11 shows the impact of collision number β on the sampling accuracy. We simulate a static system with a total of 16,800 users, and with a chain channel structure of 4 channels. We use beta function with parameters (1,1), whose shape is shown in Fig 2.8. We can see that when β is larger than 20, the estimated number of users is very close to the actual result. DAC sets β to 50 by default, which can achieve good sampling accuracy for systems with up to 100,000 users based on our simulation results. Fig 2.12 shows the estimated number of users in a very dynamic system where the total number of users first increases quickly from 10,000 to 60,000, and then drops to 20,000. Even in this case, the sampling method still achieves good accuracy.

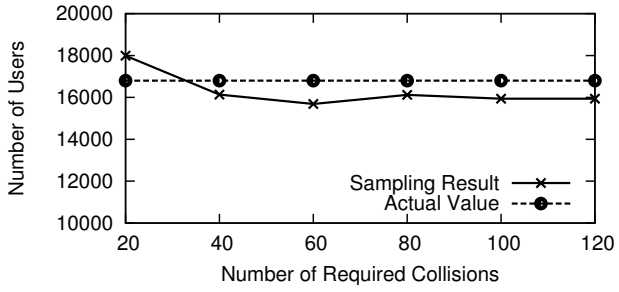

Figure 2.11: Impact of collision number β on sampling accuracy in a static system.

2.4.2.2 Sampling overhead

We use this group of simulations to evaluate the control overhead of the S&C sampling method. The accurate system information can always be obtained by requiring peers

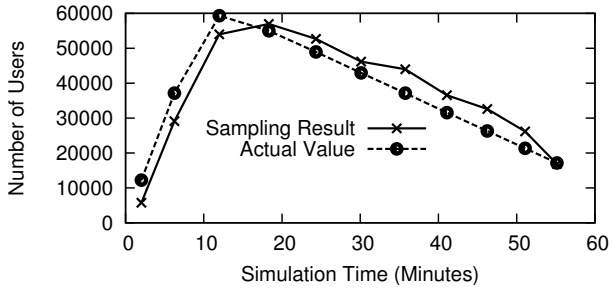

Figure 2.12: Sampling a dynamic system.

to report their information with $O(N)$ messages (N is the number of peers in the system). Therefore, we define the sampling overhead as the number of messages, MSG, used in sampling divided by the total number of peers, N. We simulate both static and dynamic cases, where the sampling parameter β is 50. For the static case, there are 200,000 peers and 32 channels. For the dynamic case, peers arrive at the system at the rate of 10 peers per second and the sojourn time of each peer is determined by the real trace (please refer to Section 2.4.4.4). Fig 2.13 shows that the sampling overhead for static case is about 4% and is about 10% for dynamic case, which indicates that our sampling method can achieve good estimation accuracy with affordable overhead. The theoretical overhead bound of $S\&C$ is $O(\sqrt{N})$ [57].

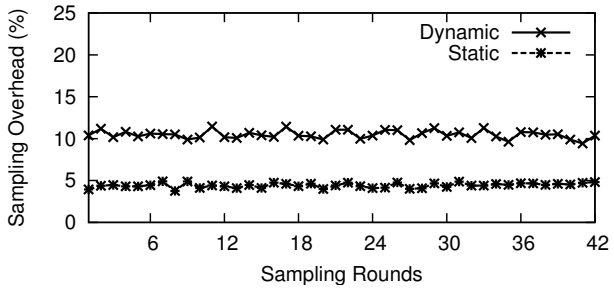

Figure 2.13: Sampling overhead of a static system with 200,000 peers and a dynamic system with user arrival rate 10 users/second.

2.4.2.3 Impact on DAC

To study the impact of the sampling method on the performance of DAC, we simulate two protocols: DAC and Oracle (as a reference protocol). Note that the major difference between DAC and Oracle is that Oracle uses the exact information collected by a centralized monitoring server; while DAC uses the estimated information based on sampling. The channel structure and the beta distribution is same as the above simulations. Oracle is very similar to DAC, except that it has the accurate information of the system and thus does not use the sampling method as DAC. We simulate the same system as the one simulated for Fig 2.11, where the total bandwidth of the system is enough to support all channels, but different channels have different resource indices (See previous page) with ISO. This is likely to happen in a P2P system with multiple channels as shown by a recent measurement study of PPLive [31]. The resource index with ISO for four channels are: 0.9, 1.3, 1.0 and 1.3. Fig 2.14 shows the bandwidth satisfaction ratio (the total allocated bandwidth over the total required bandwidth) of each channel with DAC and Oracle. We can see that when β is large enough (in this case 20), DAC with the estimated information achieves very similar results as Oracle.

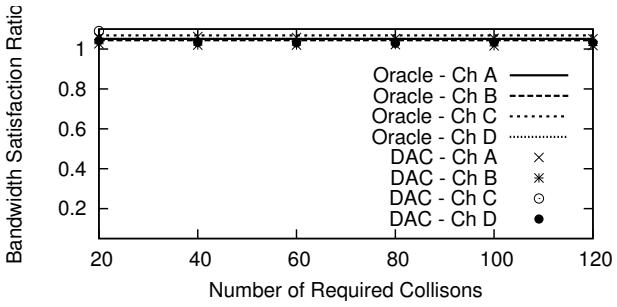

Figure 2.14: For large enough β, the performance of DAC is insensitive to the value of β.

2.4.3 Group II: Flexibility Evaluation DAC vs. AAO

We use two representative block scheduling algorithms as intra-channel streaming protocols and compare DAC with AAO. We vary three parameters to show that the performances of the two block scheduling algorithms depend on the flexibility of DAC and AAO. The three parameters are: the average resource index, the maximum number of neighbors and the streaming rate. By default, we simulate 2000 peers and 4 channels with a mesh structure (the parameters for beta distribution is (2,2), whose shape is shown in Fig 3.9). The resource indices with ISO for 4 channels are: 0.9, 1.3, 1.0 and 1.3. The default streaming rate is 300 Kbps. The simulation time for all simulations is 1000 seconds.

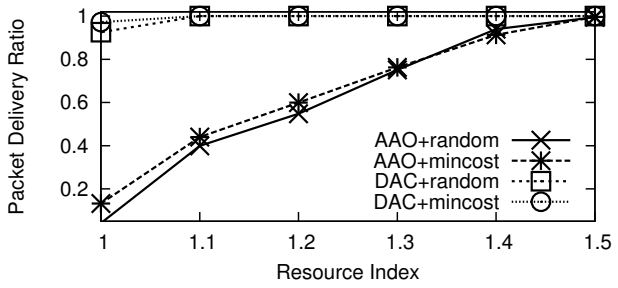

Figure 2.15: DAC provides good streaming quality for various resource indices; AAO requires more bandwidth to achieve similar performance.

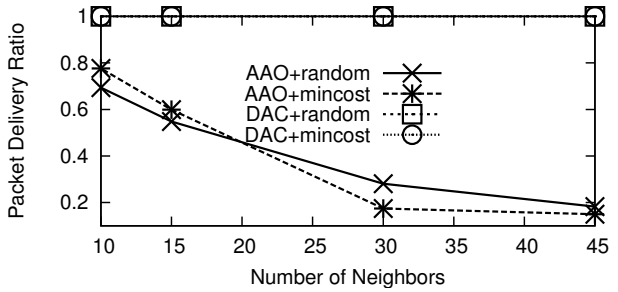

Figure 2.16: AAO streaming quality decreases as the number of neighbors increases due to inefficient bandwidth utilization; DAC always achieves good performance.

From Figs 2.15, 2.16, 2.17, we note that DAC always provides near optimal per-

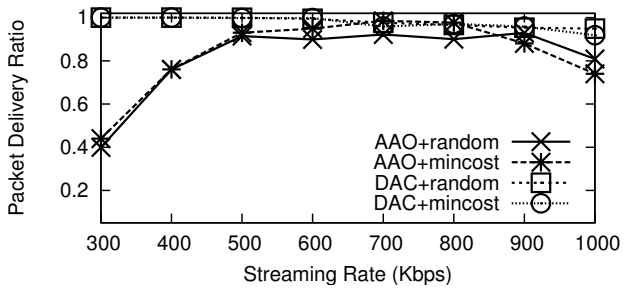

Figure 2.17: AAO's streaming quality fluctuates with different video streaming rates; DAC always achieves good performance.

formance with both streaming protocols. Compared with DAC, AAO suffers bad performance, due to its inflexible design. Fig 2.15 shows that AAO needs more bandwidth to provide good performance for both streaming protocols. Fig 2.16 shows that the performance of AAO decreases as the number of neighbor increasing. The reason is that AAO requires network coding to control the utilization of allocated bandwidth. Without network coding, the allocated bandwidth to each neighbor is poorly utilized, when the number of neighbors is large. Fig 2.17 shows that even with sufficient bandwidth (average index is 1.15), AAO's performance fluctuates with different streaming rates. From the comparison of AAO and DAC with the two intra-channel streaming protocols, we can conclude AAO is not as flexible as DAC.

2.4.4 Group III: Performance Evaluation of DAC vs. ISO

2.4.4.1 Systems with a large number of users

We simulate a system with a chain channel structure of 4 channels (parameters of beta distribution are (1,1)). Specifically, the resource index with ISO is 1.2, 1.0, 1.0, and 0.9 for channels A, B, C, and D, respectively. Fig 2.18 shows the packet delivery ratio of the system for DAC and ISO as the total number of users increases from 5000 to 20,000. Since the resource index of channel D with ISO is only 0.9, ISO

achieves a poor packet delivery ratio. We can see that DAC outperforms ISO across a wide range of system sizes, due to efficiently allocating bandwidth among different channels. Fig 2.19 shows the results with similar simulation setting, except that the system has a mesh channel structure (parameters for beta distribution are (2,2)). We can see that DAC outperforms ISO for a mesh channel structure.

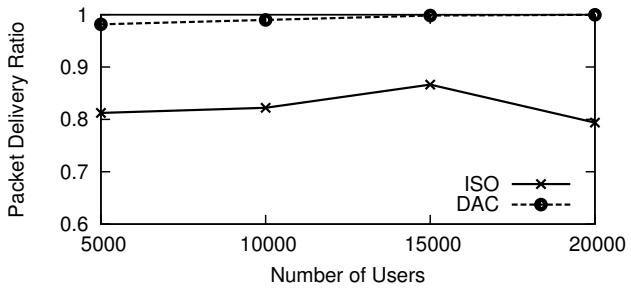

Figure 2.18: DAC outperforms ISO in systems with a chain channel structure when the number of peers increases from an intermediate scale to a large scale.

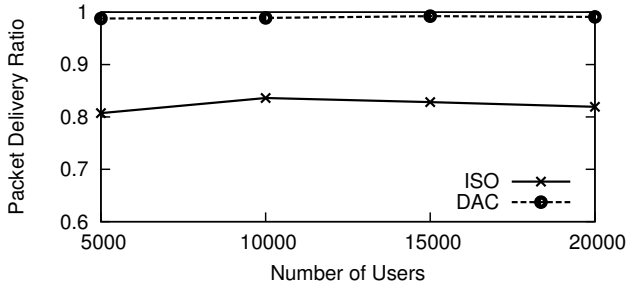

Figure 2.19: DAC outperforms ISO in systems with a mesh channel structure when the number of peers increases from an intermediate scale to a large scale.

2.4.4.2 Systems with a large number of channels

We simulate a system with a star channel structure (parameters of beta distribution are (0.8,0.8)). The number of channels varies from 2 to 32. The total bandwidth of the system is enough to support all channels, but different channels have different resource indices with ISO. Specifically, the resource index with ISO is 1.2 for half

of the channels and 0.9 for the other half of channels. The total number of users is 10,000. Again, from Fig. 2.20, we can see that DAC outperforms ISO in a wide range of channel numbers, due to efficiently allocating bandwidth among different channels.

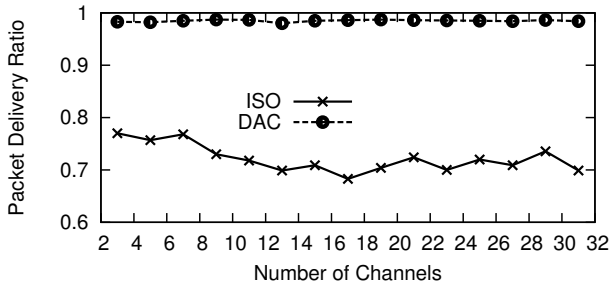

Figure 2.20: DAC outperforms ISO in systems with a star channel structure when the number of channels increases from small to large.

2.4.4.3 Systems with a dynamic number of users

We simulate a dynamic system with a mesh channel structure of 4 channels (parameters of beta distribution are (2,2)). For each channel set, the average user arrival rate is 3 users per second, and the average life time of a user is 15 minutes. The average number of users is 20,000. The total bandwidth of the system on average is enough to support all channels, and the resource index with ISO on average is 1.2, 1.0, 1.0, and 0.9 for channels A, B, C, and D, respectively. Fig 2.21 shows the packet delivery ratio of each channel for DAC and ISO. ISO achieves a good packet delivery ratio for channels A, B, and C, but not for channel D because channel D has insufficient bandwidth with ISO. We can see that DAC achieves a good packet delivery ratio for every channel.

Since peer dynamics are important features of P2P streaming systems, we use the following simulations to evaluate the performance of our DAC protocol. First, we increase the peers' arrival rate from 3 users per second to 6 users per second and then reduce the average life time of users to 8 minutes. We also change the time

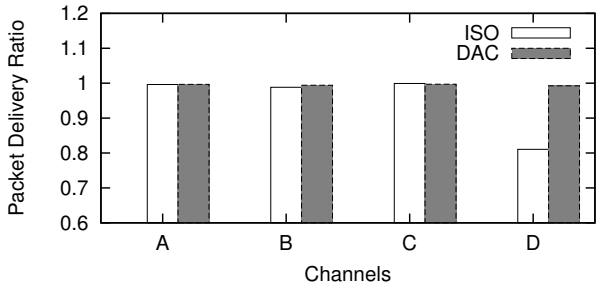

Figure 2.21: DAC outperforms ISO in dynamic systems for a large scale network.

interval Δt from 2 minutes to 4 minutes to show the impact of Δt. We only show the simulation results of channel D, since it is the channel that suffers the bandwidth deficit. Fig 2.22 shows the number of online peers watching channel D against the simulation time. From Fig 2.23, we can see that DAC can achieve a very good performance even in a very high dynamic situation. Fig 2.23 shows that from the beginning of the simulation to about 250 seconds, the average delivery ratio is not optimal, in that during every $\Delta t = 2$ minutes, there are about 720 peers joining the network (more than 20% population increase and DAC protocol should be paused). After 250 seconds, almost all peers have joined the network and the average life time is 8 minutes and therefore the average delivery ratio is close to 1. As we expected, increasing Δt might result in worse performance, as long as DAC cannot accurately estimate the system information required for bandwidth allocation. As previously mentioned (see Section 2.3.6), in this situation, DAC should be paused.

2.4.4.4 Trace-driven evaluation

We also evaluate our DAC protocol with a real trace collected from GridMedia, which provides the user's online life times. The trace was collected when GridMedia system helped CCTV to broadcast Spring Festival (Chinese New Year) Gala Show of Channel CCTV1 in 2006 through the global Internet, which served more than 1,800,000 users

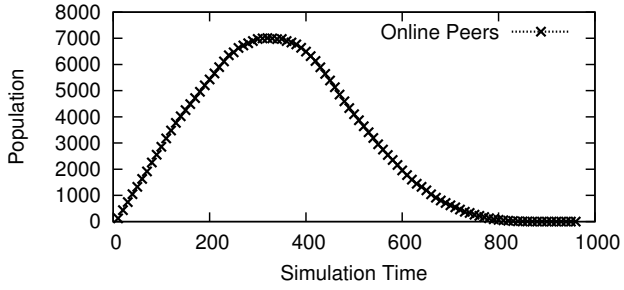

Figure 2.22: Number of peers watching channel D with arrival rate 6 user/second and average life time 8 minutes.

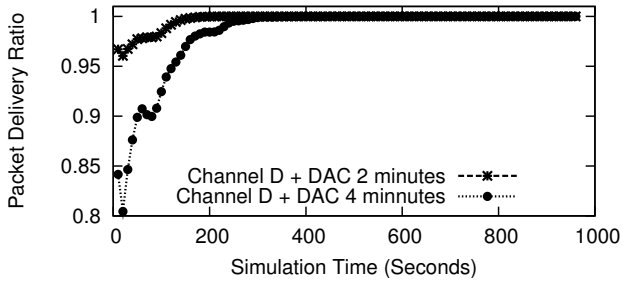

Figure 2.23: The average packet delivery ratio of channel D (calculated every 10 seconds), DAC execution interval is 2 minutes VS 4 minutes.

from tens of countries all over the world [102]. The CDF of user's online times is shown in Fig 2.24. Since the trace did not provide the user's arrival pattern, we still use the average arrival rate 6 users per second in this simulation. In addition, we simulate the same scenario as [102], where users join the system during the whole simulation.

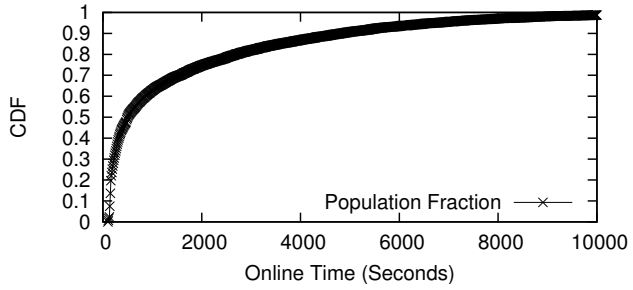

Figure 2.24: The CDF of users' life time from real trace.

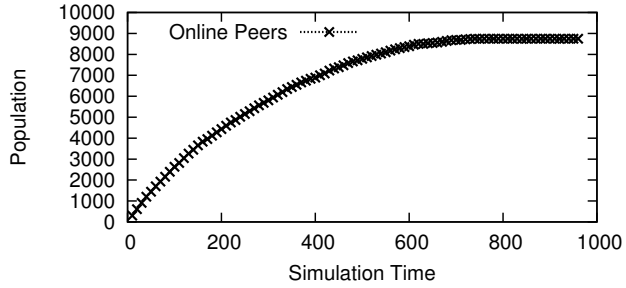

Figure 2.25: Number of peers watching channel D with arrival rate 6 user/second and average life time retrieved from the real trace.

Fig 2.25 shows the number of online peers watching channel D against the simulation time. From Fig 2.26, we can see that even the worst channel can achieve good a streaming quality with our DAC strategy, which is comparable to the simulation results in [102]. Moreover, Fig 2.26 shows that our DAC strategy improves the streaming quality up to 20%, compared with ISO.

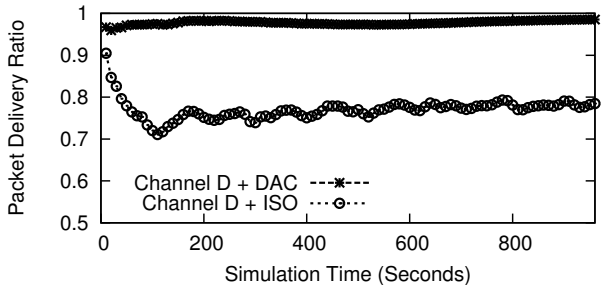

Figure 2.26: The average packet delivery ratio of channel D (calculated every 10 seconds).

2.4.4.5 Systems with insufficient bandwidth

We simulate a system with a chain channel structure of 4 channels (parameters of beta distribution are (1,1)). The total number of users is 20,000. But the total bandwidth of the system is insufficient to support all channels. In this case, different channels are assigned different priorities by changing R_c in the utility function (e.g., R_A is larger than R_C, which means that channel A has higher priority than channel C). Specifically, channels A and B are assigned the highest priority. Channel C has lower priority than A and B. Channel D is assigned the lowest priority. Fig 2.27 shows the packet delivery ratio of each channel measured every 10 seconds. We can see that channels A and B achieve the highest delivery ratio and channel D achieves the lowest delivery ratio.

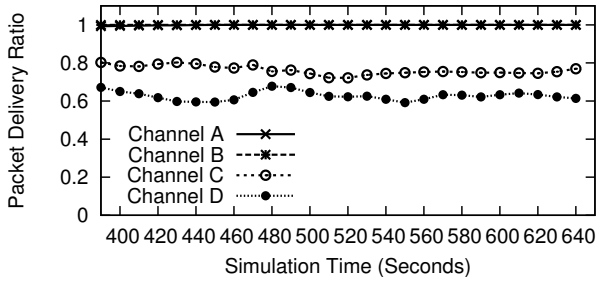

Figure 2.27: DAC provides a better packet delivery ratio to a channel with a high priority, when the total upload bandwidth is insufficient.

2.4.5 Group IV: Intra-channel streaming quality evaluation for DAC

In previous groups of simulations, we use the delivery ratio as the key metric to evaluate the performance of DAC. In this section, we evaluate the intra-channel streaming quality of DAC with metrics that are used in single-view system design [102]. 1) Packet arrival delay: it is the time elapsed between the packet sent by the source and finally received by the receiver after one or more hops; 2) Control packet rate: packets per second of control messages, which include management messages, buffermap exchange messages, and neighbor selection messages, etc.; and 3) 0.99-playback delay: 0.99 playback delay is defined as the minimum time interval between the instant that the peer starts to request the video and the instant that its delivery ratio reaches 0.99. It is used to capture the delay experienced by end users. Usually, peers buffer the video chunks for better streaming quality, but they have to wait for some time before playing the video, which will greatly influence user's watching experience and shorter delays imply better experiences. We simulate the single-view protocol proposed in [102] as a reference (REF for short), since we use it as the intra-channel streaming protocol for DAC.

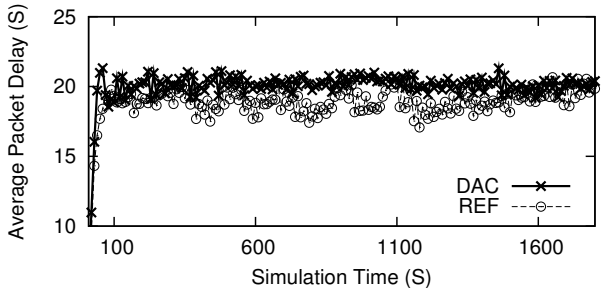

Figure 2.28: The average packet arrival delay of the 1,800 seconds simulations (calculated every 10 seconds).

We simulate a system with a chain channel structure of 4 channels and sufficient

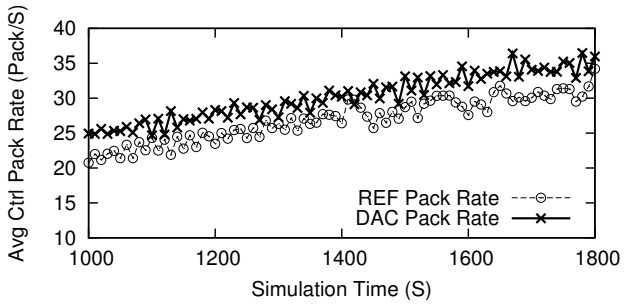

Figure 2.29: The average control packet rate of the 1,800 seconds simulations (calculated every 10 seconds).

bandwidth (parameters of beta distribution are (1,1)). The total number of users is 20,000. The peer joining/leaving is driven by the trace used in Section 2.4.4.4. We use the random data block scheduling algorithm. Fig. 2.28 shows the average packet arrival delay of all online peers during 1,800 seconds, which is similar to the REF result. From Fig. 2.29, we can see that the average control packet rate increases from about 21 packets/second to 35 packets/second, since the number of peers increases as shown in Fig 2.25. Moreover, compared with the REF result, the average control packet rate is only slightly higher, which provides reassurance that our sampling method does not introduce a large overhead. Fig. 2.30 shows that most of peers have a 25-second 0.99 playback delay in the worst channel, which is acceptable with pure pull-based method [100].

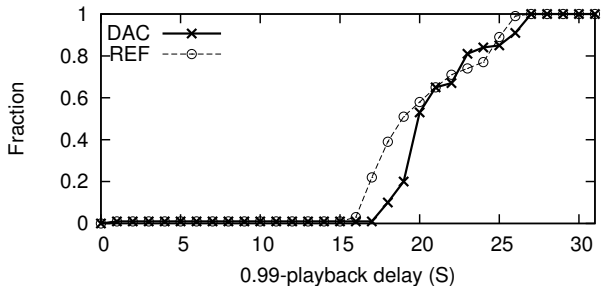

Figure 2.30: CDF of 0.99 playback delay of peers in the worst channel.

2.5 Chapter Summary

In this chapter, we propose a flexible, efficient and scalable protocol called DAC for multi-view P2P streaming systems using a divide-and-conquer strategy. To achieve flexibility and scalability, DAC solves the inter-channel competition problem at the channel level, compared with existing work AAO, which solves the problem at the peer level. Moreover, DAC integrates with the statistical sampling module to measure the system information used by DAC, and achieves reasonably good accuracy with affordable overheads. To meet the efficiency goal, DAC allocates the upload bandwidth to different channels according to their demands via our proposed utility based optimal resource allocation model. Our extensive packet level simulations show that DAC achieves its design goals.

Chapter 3

Exploring The Design Space Of Multi-Channel Peer-to-Peer Streaming Systems

3.1 Three Designs For Multi-Channel P2P Streaming System

Peer-to-peer (P2P) video streaming systems, including both live streaming and Video-On-Demand (VOD) applications, have been hugely successful in providing multimedia streaming services with hundreds of channels (e.g., UUSee claims to provide about 10,000 channels [77]). Other similar large-scale industry deployments including PPStream [67], CoolStreaming [104] and PPLive [66], support hundreds of channels with tens of thousands of concurrent users[1] [32]. All of these systems are referred to as multi-channel P2P video streaming systems.

Current measurement studies [32] [33] show that the resource distribution such

[1]We use the term *user* and *peer* inter-changeably in this chapter.

as upload bandwidth is unbalanced among different channels, which implies that some channels have satisfactory streaming qualities with surplus resources, while others suffer poor streaming qualities due to resource deficit. Allowing the channels with surplus bandwidth help those with deficit bandwidth is the common intuition behind several potential designs, since the upload bandwidth is the most precious resource that greatly influences the streaming qualities of all channels [102] [40]. In this chapter, cross-channel cooperation means sharing upload bandwidth among different channels.

In a multi-channel system with cross-channel cooperation, a user may subscribe to a variable number of channels[2], and simultaneously watch either all or some of the subscribed channels. It is realistic for a user to simultaneously watch multiple channels, since commercial P2P streaming systems allow their users to watch programs in customized manners (e.g., watching two channels using Picture-In-Picture). Note that, a user may not watch all of its subscribed channels. For example, Wu *et al.* [92] propose a View-Upload-Decoupling (VUD) approach for building multi-channel P2P streaming systems that requires a user to subscribe to other channels as a helper to alleviate the impact of channel switching, even though the user does not watch these subscribed channels.

There are three potential designs for multi-channel systems that allow users to watch/subscribe to a variable number of channels.

- Naive Bandwidth allocation Approach (*NBA*), where a user subscribes to only its watched channels, and allocates its upload bandwidth to its watched channels proportional to the channel streaming rates (e.g., if all channels have the same streaming rate, the user allocates its bandwidth equally to all watched

[2]Subscribing to a channel means that a peer participates in video dissemination for that channel, but may not watch that channel. If the peer does not watch that channel, it serves as a helper [92] for that channel.

channels.). Most of the current multi-channel systems use *NBA* due to its simplicity.

- Passive Channel-aware bandwidth allocation Approach (*PCA*), where a user subscribes to only its watched channels, and optimally allocates its bandwidth to its watched channels. The bandwidth allocation algorithm [88] for overlapped overlays and the protocol proposed in [81] are examples of *PCA* design.

- Active Channel-aware bandwidth allocation Approach (*ACA*), where a user subscribes to not only its watched channels, but also maybe to some other channels as a helper. A user optimally allocates its bandwidth to the watched channels and the subscribed but unwatched channels. Note that the main difference between *PCA* and *ACA* is that *ACA* requires a user to subscribe to some channels that it does not watch and to allocate its bandwidth to the unwatched channels. View-Upload-Decoupling (VUD) proposed in [92] is a special case of *ACA* design, since a peer is restricted to watch only one channel and might be selected by the system to join other channels as a helper.

Intuitively, *ACA* should perform better than *PCA*, since *ACA* can use all of its surplus bandwidth efficiently in a system. *PCA* should perform better than *NBA*, since *PCA* is aware of the bandwidth imbalance in a system. However, their implementation complexity also increases in the order of *NBA*, *PCA*, and *ACA* (refer to Section 3.3.4 for detailed implementation complexity discussions). Consequently, when designing multi-channel systems, we must decide which design should be used by considering the performance and complexity.

However, all the existing works focus on proposing and evaluating specific protocols instead of studying the intrinsic features of designing multi-channel systems. The goal of this chapter is to generalize and analyze the designs of multi-channel sys-

tems, and thus shed insights into choosing the proper design, in term of complexity and effectiveness. Specifically, we answer the following two questions: 1) what are the general characteristics of existing and potential designs? 2) under what circumstances, which design should be used to achieve the desired streaming quality with the lowest complexity?

The contributions of this chapter are as follows: 1) we identify three designs, namely *NBA*, *PCA* and *ACA*, for building multi-channel P2P streaming systems and develop simple models based on linear programming and network flow graphs for the three designs, which capture their main characteristics; 2) with established models, we further prove that finding optimal *ACA* design with overhead is NP-Complete and provide qualitative discussion of relative implementation complexities; 3) we derive closed-form results for a two-channel system; our results show that for this special case there is no need to use *ACA* design, and the *NBA* design can either only provide low quality streaming or consumes higher bandwidth to provide the same level of streaming quality as *PCA*; the channel structure (refer to the last paragraph of Section 3.4 and Section 3.5.2) greatly influences the performance; and 4) we conduct extensive numerical simulations to compare the three designs in general cases. Our results show that for general multi-channel P2P streaming systems, *PCA* can achieve the same performance as *ACA*, while for special applications, *ACA* is required.

The rest of this chapter is organized as follows. Section 3.2 briefly summarizes the related work. Section 3.3 describes our simple models based on network flow graphs and insights on the three designs. Section 3.4 discusses the homogenous two-channel systems. Section 3.5 describes the simulation settings and results. Finally, we conclude this chapter in Section 3.6.

3.2 Comparison With Existing Work

Most of the literature about P2P streaming systems focuses on improving the performance within a single channel (referred to as the single-channel P2P streaming systems). Tree-based overlay derived from IP multicast (e.g., Zigzag [76], [17]) is first used to build single-channel systems. However, the tree structure is not resilient to dynamics (e.g., peer joining/leaving the system randomly). Therefore, mesh-based overlays are widely used in commercial systems such as PPLive [66] and UUSee [77]. CoolStreaming [104] first introduces the data-driven design to P2P streaming systems, which has been proven to be powerful in real implementations. Generally speaking, all these designs aim to efficiently utilize peers' upload bandwidth for building scalable and robust single-channel P2P streaming systems.

Recently, P2P streaming systems where a user subscribes to more than one channel have emerged. Wu *et al.* [88] first investigate the case when a peer joins multiple overlays in a P2P live streaming system and propose an auction-based bandwidth allocation algorithm to improve the streaming quality for all channels. In our previous work [81], we propose a flexible protocol for multi-view P2P live streaming systems, based on the divide-and-conquer strategy, which solves the inter-channel competition and intra-channel streaming separately.

In terms of multi-channel systems, there are two closely related papers. In [91] the authors study the problem of provisioning the server bandwidth consumption in multi-channel systems. Wu *et al.* [92] propose the view-upload-decoupling approach to minimize the influence of channel churn among multiple channels. Moreover, in [93], the same authors establish queueing network models to analytically study the performance of multi-channel systems by considering channel churn, peer churn and bandwidth heterogeneity etc. Their analytical model differs from ours, due to following reasons. 1) They focus on the multi-channel system with the restriction that a

peer can watch exactly one channel, which is a special case of the *ACA*; we study more general cases. 2) they analyze the dynamic features for multi-channel systems, in which a specific approach is used. We focus on fundamental problems of whether a complex design should be used and which design is better.

In terms of theoretical analysis of P2P streaming, there are some studies on single-channel streaming systems, where there is exactly one channel in the system. Kumar *et al.* [40] study the performance limitations of a single-channel streaming system with a stochastic fluid model. Liu *et al.* [49] derive the performance bound of single-channel systems in terms of server load, streaming rate and tree depth. Massoulie *et al.* [58] develop a network flow based model to study the decentralized broadcasting problems and propose an optimal broadcasting algorithm. In [47], Liu *et al.* study the flash crowd problem in P2P live streaming systems. [105] and [13] focus on the chunk scheduling problem and propose optimal algorithms. They assume that the total upload bandwidth supply of all channels are sufficient to satisfy the total bandwidth demand to guarantee the algorithm convergence and the algorithm is evaluated with a small number of channels and channel combinations.

In our previous works [86] [83], we propose the framework for comparing multi-channel P2P streaming systems with linear programming models. In this chapter, we first extend the framework with detailed discussions and proofs and apply the framework for analyzing the three designs as well. Moreover, we discuss the model of *ACA* design with overhead and use extensive numerical simulations to compare the three designs in general scenarios.

3.3 Linear Programming Models, Network Flow Graphs and Insights For Multi-Channel P2P Streaming Designs

In Section 3.3.1, we introduce the linear programming models for the three designs with feasibility definitions. Section 3.3.2 uses network-flow graphs to shed insights of the three designs. Then, in Section 3.3.3, we prove that the model for *ACA* design with overhead is NP-Complete. Finally, we end this section with discussions of implementation complexities of the three designs in Section 3.3.4.

3.3.1 Linear programming models for the three designs

An important feature of a P2P system is user dynamics; that is, a user may randomly join or leave the system (referred to as peer churn), and change its watched channels (referred to as channel churn). In response to user dynamics, we divide the time axis into a series of short time intervals, and assume that during each interval the system is relatively stable.

The system with peers and their watched channels in each interval is defined as the system configuration for that time interval. Our models study various system configurations that occur in an interval.

Studying the system configurations in an interval is reasonable due to the following reasons. Recent P2P measurement studies [32, 42, 78] show that a P2P system is relatively stable over an interval of minutes, because most users have a lifetime longer than a minute, and at any time instant a significant percentage of users (e.g., $> 70\%$ on average reported in [78]) even have a lifetime on the order of hours. Moreover, we can use the queueing network models [93] to extend our model to capture peer

dynamics.

We model the upload bandwidth allocation problem for *NBA*, *PCA* and *ACA* with respect to a given system configuration, since the cross-channel upload bandwidth sharing is the key issue in designing multi-channel P2P streaming systems. *Furthermore, we are interested in comparing the three designs with the same design goal of maximizing the bandwidth obtained by each channel, which is the direct or indirect design goal in most scenarios.* It does not make any sense to compare designs with different design goals (e.g., the designs with different user utilities). Therefore, we define the *bandwidth satisfaction ratio* of a channel as the total obtained upload bandwidth of that channel over the the channel's total bandwidth demand (a formal definition will be introduced below). The goal of all the three designs is to maximize the aggregated bandwidth satisfaction ratio of all channels, in that the upload bandwidth influences the performance of a P2P streaming system [102] [40] [93].

We introduce the common notation used in Section 3.3.1 and Section 3.3.2 as follows.

- Let Θ be the set of all channels.
- Let $\theta \subseteq \Theta$ be a subset of channels.
- Let S_θ be the group of peers watching just channel set θ. That is, $S_\theta = \{m|\theta(m) = \theta\}$. Note that $S_{\theta_1} \cap S_{\theta_2} = \emptyset$ for $\theta_1 \neq \theta_2$. $\theta(m)$ denotes the channel set watched by peer m.
- Let u_m be the upload bandwidth of peer m.
- Let r_c be the streaming rate of channel $c \in \Theta$.
- Let x_c^θ denote the fraction of upload bandwidth that group S_θ allocates to channel $c \in \theta$.

- Let y_c^θ denote the fraction of upload bandwidth that group S_θ allocates to channel c not in θ. Note that $\sum_{c\in\theta} x_c^\theta + \sum_{c\notin\theta} y_c^\theta = 1$. Also note that for *NBA* and *PCA*, y_c^θ is always 0.

- Let γ_c denote the bandwidth satisfaction ratio of channel c, where γ_c is nonnegative and will be given for each design below.

- Let s_c be the upload bandwidth of the streaming server for channel c.

- Let U_θ be the total upload bandwidth supply of user set S_θ.

- Let D_c be the total upload bandwidth demand of channel c.

- Let N be the total number of peers.

- Let P_θ be the fraction of peers watching channel set θ.

3.3.1.1 Model for NBA

A peer in *NBA* may watch one or multiple channels, and it subscribes to only its watched channels. It allocates its upload bandwidth among its watched channels proportional to their streaming rates. Therefore, a peer m watching channel set θ, allocates its upload bandwidth u_m to channel $c \in \theta$ with the fraction $\frac{r_c}{\sum_{\forall c'\in\theta} r_{c'}}$. That is $x_c^\theta = r_c \times (\sum_{\forall c'\in\theta} r_{c'})^{-1}$

For each channel $c \in \Theta$, the total upload bandwidth demand is

$$D_c = \sum_{\forall\theta: c\in\theta} |S_\theta| r_c \tag{3.1}$$

The total upload bandwidth supply is

$$S_c = \sum_{\forall\theta: c\in\theta} x_c^\theta (\sum_{\forall m\in S_\theta} u_m) + s_c \tag{3.2}$$

The bandwidth satisfaction ratio γ_c for channel c is

$$\gamma_c = \frac{D_c}{S_c} \tag{3.3}$$

.

Definition 1 *Given a system configuration, the multi-channel P2P streaming system is defined as NBA feasible if* $\forall c \in \Theta$, $\gamma_c \geq 1$ holds.

3.3.1.2 Model for PCA

A peer in *PCA* may watch one or multiple channels, and it subscribes to only its watched channels. *PCA* is aware of bandwidth imbalance among different channels. Therefore, it optimally allocates the upload bandwidth of a peer in order to maximize the overall system streaming quality. That is, the goal of *PCA* is to find the optimal x_c^θ for the following optimization problem.

$$\max \sum_{\forall c \in \Theta} \gamma_c \tag{3.4}$$

subject to

$$\sum_{\forall \theta: c \in \theta} |S_\theta| r_c \leq \sum_{\forall \theta: c \in \theta} x_c^\theta (\sum_{\forall m \in S_\theta} u_m) + s_c, \forall c \in \Theta \tag{3.5}$$

$$\sum_{\forall c \in \theta} x_c^\theta = 1, \forall \theta \subseteq \Theta \tag{3.6}$$

$$x_c^\theta \geq 0, \forall c \in \Theta, \theta \subseteq \Theta \tag{3.7}$$

where $\gamma_c = (\sum_{\forall \theta: c \in \theta} x_c^\theta (\sum_{\forall m \in S_\theta} u_m) + s_c) \times (\sum_{\forall \theta: c \in \theta} |S_\theta| r_c)^{-1}$.

Definition 2 *Given a system configuration, the multi-channel P2P streaming system is defined as PCA feasible if the constraints (3.5) - (3.7) are satisfied simultaneously.*

3.3.1.3 Model for ACA

A peer in *ACA* may watch one or multiple channels. In addition to subscribing to the watched channels, a peer may also subscribe to one or multiple other unwatched channels, with the aim of contributing its surplus upload bandwidth to the channels with deficient upload bandwidth.

Note that, in order to forward packets of an unwatched channel, a peer must first download these packets, which in turn consumes the upload bandwidth of that channel. That is, while a peer contributes its bandwidth to an unwatched channel, it at the same time also consumes the bandwidth of the unwatched channel (called *overhead*). Therefore, an efficient *ACA* protocol should minimize its overhead. For example, the View-Upload-Decoupling (VUD) protocol proposed in [92] divides the video stream of a specific channel into multiple substreams (e.g. one substream contains packets with even sequence numbers and the other contains packets with odd sequence numbers), which greatly reduces the overhead due to partial downloading of the video stream. In this subsection, we assume that the overhead is zero in order to simplify the analysis. This implies that we consider the best performance of *ACA*. We will discuss the *ACA* with overhead in Section 3.3.3.

The goal of *ACA* is to find the optimal x_c^θ and y_c^θ for any c and θ for solving the following optimization problem.

$$\max \sum_{\forall c \in \Theta} \gamma_c \tag{3.8}$$

subject to

$$\begin{aligned}\sum_{\forall\theta:c\in\theta}|S_\theta|r_c &\leq \sum_{\forall\theta:c\in\theta}x_c^\theta(\sum_{\forall m\in S_\theta}u_m) \\ &+ \sum_{\forall\theta:c\notin\theta}y_c^\theta(\sum_{\forall m\in S_\theta}u_m) \\ &+ s_c, \forall c\in\Theta \end{aligned} \tag{3.9}$$

$$\sum_{\forall c:c\in\theta}x_c^\theta+\sum_{\forall c:c\notin\theta}y_c^\theta = 1, \forall\theta\subseteq\Theta \tag{3.10}$$

$$x_c^\theta, y_c^\theta \geq 0, \forall c\in\Theta, \theta\subseteq\Theta \tag{3.11}$$

where $\gamma_c =$

$$\frac{\sum_{\forall\theta:c\in\theta}x_c^\theta(\sum_{\forall m\in S_\theta}u_m)+\sum_{\forall\theta:c\notin\theta}y_c^\theta(\sum_{\forall m\in S_\theta}u_m)+s_c}{\sum_{\forall\theta:c\in\theta}|S_\theta|r_c}. \tag{3.12}$$

Definition 3 *Given a system configuration, the multi-channel P2P streaming system is defined as ACA feasible if the constraints (3.9) - (3.11) are satisfied simultaneously.*

In addition to the above three feasibility conditions, we also consider the following general feasibility condition.

Definition 4 *Given a system configuration, the system-wide feasibility for NBA, PCA and ACA is defined such that the following inequality holds*

$$\sum_{\forall c:c\in\Theta}\sum_{\forall\theta:c\in\theta}|S_\theta|r_c \leq \sum_{\forall m\in M}u_m+\sum_{\forall c:c\in\Theta}s_c \tag{3.13}$$

.

The system-wide feasibility condition is the necessary condition for all channels to stream the video at the source rate. Otherwise, none of the *NBA* feasibility condition, *PCA* feasibility condition or *ACA* feasibility condition can be achieved for the system. Note that the group of constraints (3.9) - (3.11) is equivalent to constraint (3.13), and thus we have the following theorem.

Theorem 2. *A system configuration is* ACA *feasible if and only if it is system-wide feasible.*

Proof: ($\Rightarrow$): According to Definition 3, if a system configuration is ACA feasible, then constraint (3.9) holds, which implies that for each channel c, the total bandwidth demand of that channel is less than or equal to the total bandwidth supply of channel c. Then, we do summation over all channels. Therefore, the left-hand side of (3.9) is $\sum_{\forall c:c\in\Theta}\sum_{\forall\theta:c\in\theta}|S_\theta|r_c$ and the right-hand side is $\sum_{\forall c:c\in\Theta}(\sum_{\forall\theta:c\in\theta}x_c^\theta(\sum_{\forall m\in S_\theta}u_m)+\sum_{\forall\theta:c\notin\theta}y_c^\theta(\sum_{\forall m\in S_\theta}u_m))+\sum_{\forall c:c\in\Theta}s_c$. By equation (3.10), $\sum_{\forall c:c\in\Theta}(\sum_{\forall\theta:c\in\theta}x_c^\theta(\sum_{\forall m\in S_\theta}u_m)+\sum_{\forall\theta:c\notin\theta}y_c^\theta(\sum_{\forall m\in S_\theta}u_m))=\sum_{\forall m\in M}u_m$, which implies that inequality (3.13) holds.
($\Leftarrow$): $\forall m\in M$ watching a channel set θ, the peer can allocate x_c^θ of its bandwidth to a channel c, $\forall c:c\in\theta$ and it can also allocate y_c^θ of its bandwidth to a channel c, $\forall c:c\notin\theta$. Furthermore, the relationship of x_c^θ and y_c^θ satisfies $\sum_{\forall c:c\in\theta}x_c^\theta+\sum_{\forall c:c\notin\theta}y_c^\theta=1$. Therefore, the first term of the right-hand side of inequality (3.13) is

$$\sum_{\forall m\in M}u_m(\sum_{\forall c:c\in\theta}x_c^\theta+\sum_{\forall c:c\notin\theta}y_c^\theta). \tag{3.14}$$

Rearranging term (3.14) based on each channel c, we get the new form of the first term of (3.13)

$$\sum_{\forall c:c\in\Theta}(\sum_{\forall\theta:c\in\theta}x_c^\theta(\sum_{\forall m\in S_\theta}u_m)+\sum_{\forall\theta:c\notin\theta}y_c^\theta(\sum_{\forall m\in S_\theta}u_m)) \tag{3.15}$$

Based on inequality (3.13) and (3.15), constraint (3.9) is satisfied by all channels in the system. Therefore, we can conclude that system-wide feasible condition guarantees the ACA feasible. □

We use the objective functions of maximizing the aggregated bandwidth satisfaction ratios to establish simple linear programming (LP) models for comparing the

three designs. These objective functions might not guarantee fair bandwidth allocation among different channels. However, the optimality of our LP models guarantees that there is at least one feasible solution for a specific design and fair allocations can be achieved by other non-linear objective functions. Therefore, our LP formulation serves well for establishing tractable models and comparing feasibilities of the three designs.

3.3.2 Network-flow graphs for the three designs

For better understanding of the three designs, we use generalized network flow graphs to represent the bandwidth allocation problems corresponding to the three designs. The network flow graphs can be constructed from the resource allocation graphs described below.

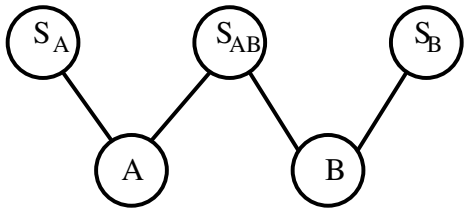

Figure 3.1: A resource allocation graph for a multi-channel P2P streaming system with two channels A and B.

We first consider the resource allocation graph and network flow graph for *PCA*. For each user set S_θ, *PCA* considers how to allocate the total upload bandwidth of all users watching just channel set θ to all channels $c \in \theta$. Intuitively, a user set S_θ provides its upload bandwidth to a set of channels and a channel c requests upload bandwidth from some user sets, therefore, we call a user set a *bandwidth supplier* and a channel a *bandwidth consumer*. The relationship between suppliers and consumers can be described by a bipartite resource allocation graph $G = (S, D, E)$, where vertex set S is the set of all suppliers (i.e., S contains S_θ for any $\theta \subseteq \Theta$), vertex set D is the

set of all consumers (i.e., D contains c for any $c \in \Theta$), and edge set E represents the supplier-consumer relationship (i.e., $e = (S_\theta, c) \in E$ iff $c \in \theta$). Fig 3.1 illustrates the bipartite graph with 3 suppliers and 2 consumers for a multi-channel P2P streaming system with 2 channels: A and B. For example, supplier S_{AB} allocates its upload bandwidth to consumers A and B.

Based on the resource allocation graphs, we can construct the network flow graph for PCA to visualize the model for PCA. We introduce two artificial vertices s and t to denote the source and sink of the network flow graph respectively. We add edges to connect the source s with all consumer vertices in D, whose capacities are the bandwidth demands of the correspondingly connected consumer vertices. The capacities of edges in the original resource allocation graph are set to $+\infty$. Then, we divide each supplier vertex into vertices S_θ and S'_θ connected by a single edge, whose capacity is determined as follows. 1) If $|\theta| = 1$, which implies that the users in S_θ watch a single channel, then the capacity $U_\theta = s_c + \sum_{\forall m \in \theta} u_m$ for $c \in \theta$. 2) If $|\theta| > 1$, then the capacity $U_\theta = \sum_{\forall m \in \theta} u_m$. Finally, we connect the vertices S'_θ to the sink t, with $+\infty$ edge capacities. Fig 3.2 shows the network flow graph with 3 suppliers and 2 consumers.

The network flow graph for ACA differs from the graph of PCA, since a user m who does not watch a channel, say c, is still able to help channel c. Therefore, we add new virtual edges to resource allocation graph to construct the network flow graph for ACA as follows. For any pair of a bandwidth supplier vertex S_θ and a bandwidth consumer vertex $c, c \in D$, there is a virtual edge connecting them. Then, we apply the same rules of constructing network flow graphs for PCA. An example is shown in Fig 3.3.

Finally, the network flow graph for NBA differs from that of PCA in the capacities of edges between the supplier and consumer vertices, which are set based on the

bandwidth allocation strategy of *NBA*, instead of $+\infty$. Fig 3.4 shows the network flow for the *NBA* design with 3 suppliers and 2 consumers, where the two channels have the same streaming rate.

With the network flow graphs of *NBA*, *PCA* and *ACA*, we can interpret the task of determining whether a specific design has a feasible solution, in the sense that there exists a bandwidth allocation strategy using which all channels' bandwidth demands are satisfied. As shown in Figs 3.2, 3.3 and 3.4, the capacities of edges connecting the source vertex s and the consumers denote the bandwidth demands. Therefore, if all these edges are saturated in the maximum flow of the graphs corresponding to *NBA*, *PCA* and *ACA* designs (the maximum flow is equal to the total bandwidth demand in the system), we say that the system is *NBA feasible*, *PCA feasible* and *ACA feasible*, respectively (Please refer to Definitions 1, 2 and 3). Obviously, the system-wide feasibility is a necessary condition for feasibilities of all three designs. Furthermore, from Fig 3.3, we can see that there is no edge capacity limit for the edge connecting a bandwidth consumer, and a bandwidth supplier and a bandwidth consumer can request bandwidth from any bandwidth supplier with the *ACA* design. Therefore, the system-wide feasible condition is also the sufficient condition for *ACA* feasibility, which is stated in Theorem 2 below. Note that the reason why system-wide feasibility implies *ACA* feasibility is that in our current *ACA* network flow graph, we do not consider the overhead caused by *ACA*, which will be discussed in the following subsection.

3.3.3 ACA model with overhead

As mentioned in Section 3.3.2, the *ACA* model in this chapter does not consider the overhead caused by cross-channel cooperation. In real P2P streaming systems, any *ACA* design will introduce different amount of overhead, since when a peer with

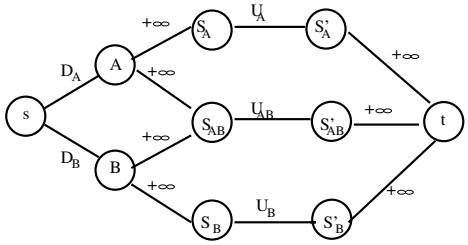

Figure 3.2: The network flow graph for 2-channel *PCA* model. The notations on edges denote the edge capacities.

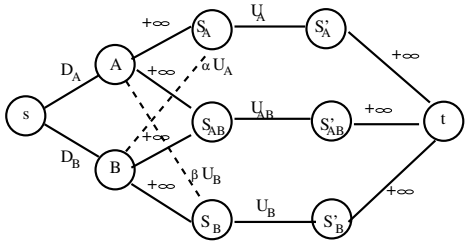

Figure 3.3: The network flow graph for 2-channel *ACA* model. The notations on edges denote the edge capacities. The dashed lines denote virtual edges.

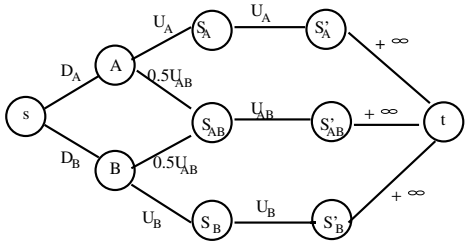

Figure 3.4: The network flow graph for 2-channel *NBA* model. The notations on edges denote the edge capacities. We assume that $r_A = r_B$.

surplus bandwidth intends to help peers watching its unwatched channels, it first needs to download the useful data (i.e. it increases the bandwidth demand of that channel). We use ACA-O to denote ACA with overhead. In this section, we show that omitting the overhead is necessary for achieving a tractable model of the ACA design and such a simplification does not change the relative order of the three designs in terms of performance.

Determining the feasibility of ACA-O could be NP-hard. Intuitively, a consumer determines from which supplier to request bandwidth and how much bandwidth should be requested from that supplier, where the number of choices of both decisions is exponentially large. Although searching over exponentially large space does not necessarily imply that the problem is NP-hard, we prove below that even a simplified ACA-O is NP-hard. Since with the ACA-O model, the peer with surplus bandwidth determines which peer should be helped and how much bandwidth should be allocated to that user while considering the overhead caused by such a cooperation. The simplified ACA-O in this chapter refers to the case that the peer with surplus bandwidth has the information about the bandwidth that can be allocated to different peers and the overhead (cost) for helping these peers. Therefore, the simplified ACA-O only needs to find out a set of peers that the peer with surplus bandwidth could help. A decision version of the simplified ACA-O is given below.

To illustrate the simplified ACA-O, we use Fig 3.3 as an example. Since ACA design causes overhead, it changes the edge capacity of edges between the source s and consumers. For example, in Fig 3.3, consumer A requests bandwidth from supplier S_B, which increases the edge capacity of edge (S, A). Therefore, for simplified ACA-O, we assume that for each consumer the amount of bandwidth requested from each supplier is set by some scheme, which is reflected by the edge capacity of edges between consumers and suppliers. For example, in Fig 3.3, the edge capacity of

(A, S_B) is some fixed number instead of $+\infty$. Then, determining the feasibility of the simplified ACA-O is to find out a feasible network flow in the flow graph with a set of saturated edges between consumers and suppliers, whose starting vertices cover all consumers. The decision version of this problem is shown below.

Simplified *ACA-O* Decision: Given a network flow graph of the simplified ACA-O, $G = (S, D, D^{'}, \{s, t\}, E)$, where S and D denote the consumers and suppliers respectively; $D^{'}$ denotes the mirror vertices of suppliers (e.g. $S_B^{'}$ in Fig 3.3); s and t denote the source and sink vertices; $S, D, D^{'}, \{s, t\}$ denotes the vertex set of the graph; E denotes the set of edges with capacities.

Question: Is there a network flow containing a set of saturated edges between S and D, whose starting vertices cover the vertex set S?

In order to prove that the above problem is NP-Complete, we introduce a known NP-Complete problem *Exact Weight Perfect Matching* (EWPM) of a bipartite graph [28], [106].

EWPM: An edge weight bipartite graph and a positive integer α.

Question: Does there exist a perfect matching M with $Weight(M) = \alpha$?

Theorem 3. *Simplified* ACA-O *Decision is* NP-Complete.

Proof: Reducing from Exact Weight Perfect Matching (EWPM) of a bipartite graph.

Step 1: Given a solution to Simplified ACA-O Decision, it can be verified in polynomial time, since the verification time is bounded by the number of edges in the solution. Therefore, Simplified ACA-O Decision is in class NP.

Step 2: Given an instance of EWPM, we can construct a network flow graph as follows. We assume that the bipartite graph can be represented by $G^{'} = (X, Y, E^{'})$. We plan to construct a network flow graph $G = (S, D, D^{'}\{s, t\}, E)$. First, we merge all vertices in Y into one vertex y and divide y into y_1, y_2 connected by an edge with capacity α. We set $S = X$ and $D = y_1$ and $D^{'} = y_2$ with edges $E^{'}$ connecting S

and D whose edge capacities are the weights of $E^{'}$. Then, we add s and t, where s connects all vertices in S with capacities $+\infty$ and t is connected to y_2 with the capacity $+\infty$. Finally, we place newly added edges in E. The construction process takes polynomial time. Therefore, if there is a polynomial time algorithm which solves Simplified *ACA-O*, EWPM can be solved polynomially.

Step 3: Given an instance of Simplified *ACA-O*, $G = (S, D, D^{'}\{s, t\}, E)$. We construct an instance of EWPM, $G^{'} = (X, Y, E^{'})$ as follows. We first set α to be the sum of capacities of edges connecting vertices in D and $D^{'}$. Then, we remove s and t and edges connecting to them. Next, we set $X = S$ and $D = Y$. The capacities of edges connecting vertices in S and D are set to be the weight of edges connecting X and Y. We remove $D^{'}$ and edges connecting to them. Since the number of suppliers is larger than or equal to the number of consumers, we might need to merge some vertices in Y as well as the connected edges to maintain even number of vertices to guarantee a perfect matching. Finally, we add a pair of vertices a and b into X and Y respectively. The weight of edge that connects a and b is the difference between *alpha* and sum of weights of all edges in $E^{'}$ except for (a, b). Since system-wide feasible condition holds, the weight of edge (a, b) is nonnegative.

Step 4: Combining all above steps, we prove that Simplified *ACA-O* is NP-Complete and that it is equivalent to EWPM in its complexity. □

To sum up, the *ACA* model without overhead used in studying the relative performance of the three designs is reasonable and will provide meaningful comparison results for the three designs.

3.3.4 Discussions of implementation complexity

As mentioned in Section 5.1, the implementation complexity of the three designs increases in the order of *NBA*, *PCA* and *ACA*. In this subsection, we briefly discuss

what are the main factors of implementation complexities for designing multi-channel P2P streaming systems with cross-channel bandwidth sharing and how they influence the implementation complexities of the three designs.

Below, we discuss the implementation complexities of the three designs. In general, the signaling overhead is an important factor in evaluating the implementation complexity (e.g. different types of signaling messages and whether the signaling process needs to be synchronized etc.). The second important factor is whether the design needs some optimization algorithms to achieve the design goal, which might include several aspects. For example, most of the optimization algorithms require the complete information about the system, such as peer population and bandwidth distributions, where some specific information gathering/estimation mechanism should be implemented. Furthermore, the characteristics of the algorithms also influence the implementation complexity. An algorithm whose convergence requires tight communication synchronization is much more complex than that requiring only loose communication synchronization.

The *NBA* design is not aware of the bandwidth imbalance of different channels and therefore does not require any extra system bandwidth information and there is no optimal bandwidth allocation algorithm either. By contrast, the *ACA* not only requires extra system bandwidth information for optimal bandwidth allocation, but also it needs some algorithm to find a proper/optimal way for peers to determine whether they should subscribe to unwatched channels and which channels should be subscribed, which results in the most complex design among the three. The implementation complexity of *PCA* design falls between *NBA* and *PCA*, since it requires extra system bandwidth information for optimal bandwidth allocation, which is similar to *ACA*, but does not need to determine whether and how peers to subscribe to unwatched channels. Note that the qualitative comparisons are based on the

assumption that the three designs are implemented with reasonable mechanisms. A poorly designed implementation of *PCA* is probably more complex than a well-designed implementation of *ACA*.

3.4 Two-channel P2P streaming systems

In this section, we compare the three designs in a P2P streaming system with two channels, using the closed-form feasibility discriminant.

3.4.1 The closed-form discriminant for homogenous two-channel system with PCA Design

The simplest multi-channel system is the two-channel system. To obtain a closed-form discriminant, we first assume that all peers have the same upload bandwidth and the streaming rates of the two channels are the same as well. This simplified system is referred to as homogenous two-channel system (HOMO-2 for short). In addition, to state the problem compactly and show the procedure of obtaining the results clearly, we represent the problem in the linear programming format.

Before presenting the results of HOMO-2, we first introduce a theorem which will be used for establishing the closed-form results for HOMO-2 and is a variant of Farkas's lemma [8]. To state the theorem compactly, we use the matrix notation to prove the theorem. Since constraints for *NBA*, *PCA* and *ACA* model can be rearranged into the form $ax \leq b$, the matrix notation of the constraints can be written in the following format:

$$\mathbf{Ax} \leq \mathbf{b} \tag{3.16}$$

$$\mathbf{x} \geq \mathbf{0}$$

where $\mathbf{A}$ denotes the coefficient matrix for the rearranged constraints and $\mathbf{b}$ denotes the righthand-side values of the constraints.

Theorem 4. *Given a matrix* $\mathbf{A}$ *of dimensions* $m \times n$ *and a vector* $\mathbf{b} \in \mathbf{R^m}$*, the feasibility set determined by the system of inequalities (3.16) is either non-empty or* $\exists \mathbf{p} \in \mathbf{R^m}$ *satisfies* $\mathbf{p} \geq \mathbf{0}, \mathbf{p^T b} < \mathbf{0}, \mathbf{p^T A} \geq \mathbf{0^T}$*, but not both.*

For the simplified system, there are two channels, Channel 1 and 2[3] with streaming rate r and three channel sets $\theta_1 = \{1\}$, $\theta_2 = \{2\}$, $\theta_3 = \{1, 2\}$. All peers have upload bandwidth u. Other notations are the same with Section 3.3. The following theorem shows the discriminant for *PCA* design.

Theorem 5. *For homogenous two-channel system with* PCA *design (*HPCA-2*), the feasibility of the bandwidth allocation problem represented by the network flow graph is determined by* $\Delta = \frac{s_1+s_2}{uN} + P_{\theta_1} + P_{\theta_2} + P_{\theta_3} - (P_{\theta_1} + P_{\theta_2} + 2P_{\theta_3})\frac{r}{u}$*. If* $\Delta < 0$*, the bandwidth allocation problem is infeasible, otherwise it is feasible.*

Proof: Following Theorem 4, we derive the primal and dual problem for *HPCA-2* as follows (we rearranged the constraints to be in the same format as those in Theorem 4):

Dual:

$$\max \mathbf{0^T x} \tag{3.17}$$

subject to

$$-P_{\theta_3} x_1^{\theta_3} \leq \frac{s_1}{uN} + P_{\theta_1} - (P_{\theta_1} + P_{\theta_3})\frac{r}{u} \tag{3.18}$$

$$P_{\theta_3} x_1^{\theta_3} \leq \frac{s_2}{uN} + P_{\theta_2} + P_{\theta_3} - (P_{\theta_2} + P_{\theta_3})\frac{r}{u} \tag{3.19}$$

$$x_1^{\theta_3} \geq 0$$

[3]We use 1 and 2 to represent channels instead of A and B in this and next sections.

Primal:

$$\min p_1(\frac{s_1}{uN} + P_{\theta_1} - (P_{\theta_1} + P_{\theta_3})\frac{r}{u}) + p_2(\frac{s}{u} + (P_{\theta_2} + P_{\theta_3})(1 - \frac{r}{u})) \tag{3.20}$$

subject to

$$-p_1 P_{\theta_3} + p_2 P_{\theta_3} \geq 0 \tag{3.21}$$

$$p_1, p_2 \geq 0$$

Based on $P_{\theta_3} \neq 0$ and the constraint (3.21), we conclude that $p_2 \geq p_1$. Therefore, the objective function (3.20) has an upper bound $p_2 \times \Delta$ and a lower bound $p_1 \times \Delta$. Because $p_1, p_2 \geq 0$, the sign of Δ determines the sign of the objective function (3.20). If $\Delta < 0$, the Primal problem is unbounded, since its upper bound is negative and any pair of $p_1, p_2 > 0$ is a certificate of infeasibility for *HPCA-2* based on Theorem 4. If $\Delta \geq 0$, we cannot find a pair $p_1, p_2 \geq 0$ that makes the primal objective function negative, since its lower bound is always non-negative. □

When $P_{\theta_3} = 0$, it becomes a two-channel system with two isolated channels, which is a special case of *NBA* design with no cross-channel resource sharing. $\Delta = \frac{s_1+s_2}{uN} + P_{\theta_1} + P_{\theta_2} + 0 - (P_{\theta_1} + P_{\theta_2} + 0)\frac{r}{u} = \frac{s_1+s_2}{uN} + P_{\theta_1} + P_{\theta_2} - (P_{\theta_1} + P_{\theta_2})\frac{r}{u}$. $\Delta \geq 0$ indicates that $s_1 + s_2 + Nu(P_{\theta_1} + P_{\theta_2}) \geq Nr(P_{\theta_1} + P_{\theta_2})$, which is exactly the same as the condition for system-wide feasibility. For this system, only if $u > r$ can it achieve the required streaming rate.

Corollary 1. *For* HPCA-2*, when $N \to \infty$ (i.e, the system size approaches infinity), there exists a critical point $P^\star$ for the fraction of peers watching both channels, where $P^\star = \frac{u-r}{r}$. If $P_{\theta_3} \leq P^\star$ bandwidth allocation problem is feasible, otherwise it is infeasible.*

Proof: For *HPCA-2*, the equation

$$P_{\theta_1} + P_{\theta_2} + P_{\theta_3} = 1 \tag{3.22}$$

holds. Substitute $P_{\theta_1} + P_{\theta_2} + P_{\theta_3}$ in Δ of Theorem 5 with equation (3.22). Thus $\Delta = \frac{s_1+s_2}{uN} + 1 - (1 + P_{\theta_3})\frac{r}{u}$. Then

$$\lim_{N\to\infty} \Delta = 1 - (1 + P_{\theta_3})\frac{r}{u} \tag{3.23}$$

Based on Theorem 5 and (3.23), if $P_{\theta_3} > \frac{u-r}{r}$, the bandwidth allocation problem is infeasible, which implies that the critical point $P^\star = \frac{u-r}{r}$. □

The intuition behind Corollary 1 is that in *HPCA-2*, multi-channel peers consume more bandwidth than single-channel peers, but their upload bandwidths are the same as single-channel peers'. Therefore, they are the cause for bandwidth deficit and the fraction of multi-channel peers that can be supported by the system is bounded. Particularly, Corollary 1 provides the insight that the ratio of peer's upload bandwidth over streaming rate is a key parameter for *HPCA-2* to determine whether the system is *PCA* feasible. Therefore in the following general cases, we should investigate the impact of peers' upload bandwidth and streaming rates for different channels on the three designs.

3.4.2 The closed-form discriminant for homogenous two-channel system with ACA Design

We can use a similar method to obtain the closed-form discriminant for Homogeneous 2 channels with ACA design (*HACA-2*). However, we can use Theorem 2 to obtain the discriminant directly.

Theorem 6. *The* HACA-2 *has the same discriminant as the* HPCA-2.

Proof: Based on the system-wide feasible condition, we obtain the following inequality *HOMO-2*:

$$Nr(P_{\theta_1} + P_{\theta_2} + 2P_{\theta_3}) \leq Nu(P_{\theta_1} + P_{\theta_2} + P_{\theta_3}) + s_1 + s_2 \tag{3.24}$$

Let $N \to \infty$, we can derive that if $P_{\theta_3} \leq \frac{u-r}{r}$, *HOMO-2* is system-wide feasible. Therefore, the condition for system-wide feasibility of *HOMO-2* is the same as the condition for *PCA* feasibility of *HPCA-2*. Based on Theorem 2, we have proven Theorem 6. □

3.4.3 The closed-form discriminant for homogenous two-channel system with NBA design

We compare the *NBA* and *PCA* design in *HOMO-2* (referred to as *HNBA-2*) to determine which design should be used.

For *NBA* design, we substitute $x_1^{\theta_3}$ and $x_2^{\theta_3}$ with $\frac{1}{2}$ to the constraints of *HNBA-2* and get the following conclusion. When $0 < \frac{r}{u} < \frac{1}{2}$, *HNBA-2* is always *NBA* feasible. When $\frac{1}{2} < \frac{r}{u} < 1$, if $P_{\theta_3} \leq \min(P_{\theta_1}, P_{\theta_2})\frac{2(u-r)}{2r-u}$, *HNBA-2* is *NBA* feasible.

Let $P_{\min}$ denote $\min(P_{\theta_1}, P_{\theta_2})$. We can determine whether to use *NBA* or *PCA* design by comparing $P_{\min}\frac{2(u-r)}{2r-u}$ and $\frac{u-r}{r}$. If $P_{\min}\frac{2(u-r)}{2r-u} < \frac{u-r}{r}$, when P_{θ_3} in the interval $(P_{\min}\frac{2(u-r)}{2r-u}, \frac{u-r}{r})$, *PCA* design should be used. If $P_{\min}\frac{2(u-r)}{2r-u} > \frac{u-r}{r}$, when *HOMO-2* is system-wide feasible, we should use *NBA* design. The precise conclusion is summarized as follows.

Conclusion for *NBA*: 1) If $\frac{1}{2} < \frac{r}{u} < 1$, and $P_{\min} < \frac{2r-u}{2r}$, when $P_{\min}\frac{2(u-r)}{2r-u} < P_{\theta_3} \leq \frac{u-r}{r}$, we should use *PCA* design and when $P_{\theta_3} \leq P_{\min}\frac{2(u-r)}{2r-u}$, we should use *NBA* design. 2) If $\frac{1}{2} < \frac{r}{u} < 1$, and $P_{\min} \geq \frac{2r-u}{2r}$, when $P_{\theta_3} \leq \frac{u-r}{r}$, we should use *NBA* design. 3) If $0 < \frac{r}{u} < \frac{1}{2}$, we should always use *NBA* design. We will visualize the

results for *NBA*, *PCA* and *ACA* in next subsection.

3.4.4 Discussion

We visualize the results of *HOMO-2* with Figs 3.5, 3.6 and 3.7. In these figures, for a given $\frac{r}{u}$, the feasible region of the three designs is represented by the area defined by populations of peers watching only channel A (P_{θ_1}) and channel B (P_{θ_2}). From Fig 3.5, we can conclude that if $\frac{r}{u} \leq 0.5$, *NBA* design will be good enough. Based on Fig 3.6 and Fig 3.7, the feasible region shrinks with the increase of $\frac{r}{u}$, in that the bandwidth demand in the system is close to the bandwidth supply.

Moreover, the results of *HOMO-2* have two important implications for designing real P2P streaming systems and comparing different designs. First, in a multi-channel P2P streaming system with well balanced resources allocated among different channels (i.e. the bandwidth of different peer sets can be roughly considered the same), the maximum achievable streaming rate is restricted by peers' upload bandwidth and *NBA* design can only support low quality videos or it requires more bandwidth than *PCA* to sustain the same streaming rate. However, based on measurement studies [33] [32], the resources allocated among different channels are highly unbalanced, which intuitively indicates that *NBA* design is not good enough for sustaining satisfactory quality of service. Therefore, choosing the proper design is urgent for real systems.

Second, the results of *HOMO-2* analysis show that the population and upload bandwidth of different user sets and the streaming rates of different channels greatly influence the feasible region of different designs. In fact, these factors determine the bandwidth supply and demand relationship in the system, which should be carefully considered in evaluating general cases in Section 3.5. A less obvious factor shown in *HOMO-2* that influences the design space is the channel-structure, which can be

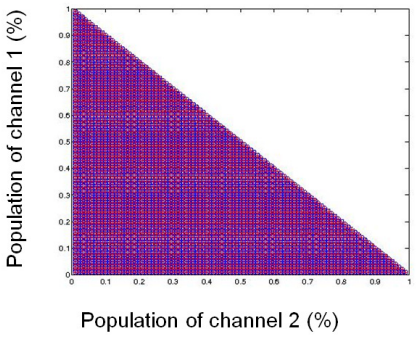

Figure 3.5: Feasible regions of the three designs when $\frac{r}{u} \leq 0.5$. All the three designs have the same feasible region.

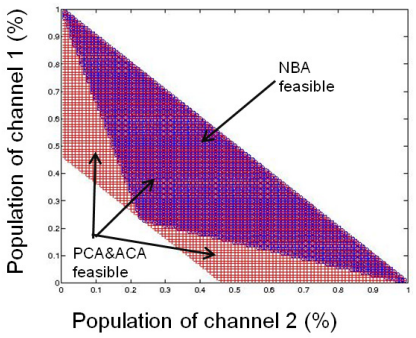

Figure 3.6: Feasible regions of the three designs when $\frac{r}{u} = 0.65$. PCA and ACA have larger feasible region, when $\frac{r}{u} > 0.65$.

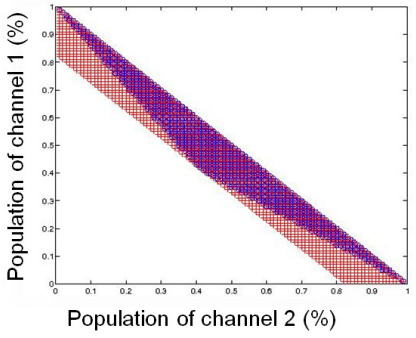

Figure 3.7: Feasible regions of the three designs when $\frac{r}{u} = 0.85$. When $\frac{r}{u}$ increases, the feasible regions of all three designs decreases.

informally defined as the channel sets in the system. For example, in *HOMO-2*, there are three channel sets: channel set $\{1\}$, set $\{2\}$ and set $\{1, 2\}$, which determine the way of cross-channel sharing. If a *HOMO-2* does not have channel set $\{1, 2\}$, channel 1 and 2 cannot share resources. In real systems, there might be a large number of channel sets corresponding to a variety of channel structures. Therefore, when comparing different designs for the general case, the channel structure factor should be given careful consideration.

3.5 Numerical Results

In this section, we investigate the characteristics of the three designs with extensive numerical simulations.

3.5.1 Experiments Setup

We develop a configurable simulator using C++ and integrate it with the *CPLEX* [21] optimization library to solve the bandwidth allocation problems of the three models. Therefore, we can compare the three designs for various peer population distributions, upload bandwidth distributions and channel structures by changing the following parameters.

As shown in the two-channel case, the channel streaming rate, the peer upload bandwidth and the peer population greatly influence the feasible region of the three designs. Therefore, the design space can be determined by the channel-structure, the peer population distribution and the peer bandwidth distribution. To explore the design space defined above, we study each design for multi-channel systems with the following two groups of parameters: 1) channel information parameters. 2) system information parameters.

The channel information parameters include the streaming rate r_c for channel c and the channel structure. We consider two types of channel streaming rates:

- homogeneous streaming rate, where all channels have the same streaming rate;
- heterogeneous streaming rates, where different channels have different streaming rates, corresponding to different video qualities;

The channel structure determines whether two or more channels overlap with each other. We use the following three types of channel structures to investigate the three designs, as illustrated in Fig 3.8:

- a chain structure where a user can view only the feeds from either a single camera or two consecutive cameras in a row of cameras;
- a mesh structure where every user watches a random number of channels;
- a star structure where there is one popular channel that every user watches;

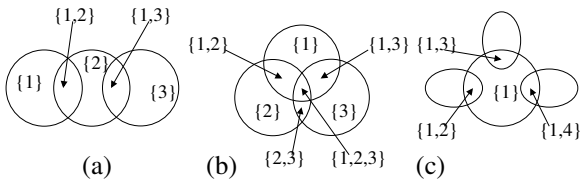

Figure 3.8: Three types of channel structures: a) chain, b) mesh, and c) star.

The above three channel structures cover a variety of general and special cases in real system. For example, the chain structure might be an application of camera monitoring systems used in traffic, zoo etc [81]; while the star structure might be a P2P streaming system with Picture-in-Picture (PIP) function [67]. For a given channel structure, the number of peers in each channel set is determined by the system information parameters below.

The system information parameters include the number of channels, the number of peers, the maximum number of simultaneously subscribed/watched channels, the bandwidth distribution of channel sets and the population distribution of the channel sets. We use the beta distribution with parameters y, z to control the bandwidth and population distribution. Beta distribution is a general type of statistical distribution, with the probability function $P(x) = \frac{(1-x)^{z-1}x^{y-1}}{B(y,z)}$, where $B(y, z)$ is the beta function defined as $B(y, z) = \frac{(y-1)!(z-1)!}{(y+z-1)!}$ [9]. The reason why we use beta distribution to control the bandwidth and population distributions is that it can generate distributions with various shapes representing different scenarios in real systems. For details, please refer to Section 3.5.2.

The population of each channel set is determined as follows: we first arrange the channel sets in lexicographical order and then assign a channel set a fraction f of the total number of peers N, which means the number of peers watching that channel set is $f * N$. We use the beta distribution to determine the fraction f. Fig 3.9 illustrates an example of assigning population distribution to a mesh channel structure with three channels and beta distribution parameters $(2, 2)$.

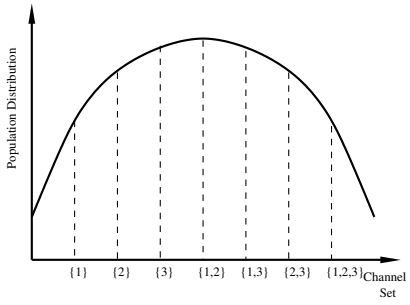

Figure 3.9: Population distribution of mesh structure with 3 channels, beta distribution with parameters (2,2).

Since the goal of these simulations is to compare the *relative* orders of the three designs, we set total upload bandwidth of each channel set as follows: for every group of simulations, we first calculate the total bandwidth demand based on the population

of each channel set and the streaming rate of each channel. Then, we set the total upload bandwidth supply U to be equal to the total bandwidth demand. That is, we only consider the case the system is feasible. Finally, we use a similar method of obtaining population fraction to assign each channel set a fraction f' of the total upload bandwidth U, generated by the bandwidth distribution function, which means that the upload bandwidth of that channel set is $f' * U$. Note that the beta distribution for controlling upload bandwidth differs from the one for controlling population. For example, Fig 3.10 shows how to assign bandwidth distribution for a 3-channel system with chain channel structure, where the user sets are $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}$ and the fractions of bandwidth of the user sets are determined by the beta distribution with parameters (1,1). *With this bandwidth setting, system-wide feasibility is guaranteed, which implies that ACA feasible condition always holds. Therefore, all the simulation results show the relative performance of the three designs.* For the reason why *ACA* feasible condition always holds, please refer to Sections 3.3.2 and 3.3.3.

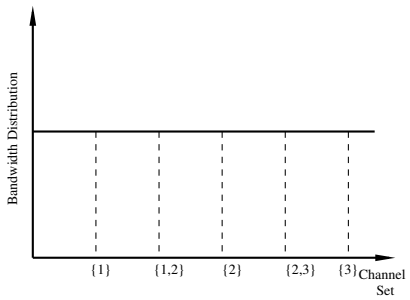

Figure 3.10: Bandwidth distribution of chain structure with 3 channels, beta distribution with parameters (1,1).

3.5.2 Simulation Parameters

The number of peers for all simulations is 100,000. Since both bandwidth and population distributions influence the design space of the three designs and the space determined by them is continuous, we try to choose as many representative cases as

possible to approximate the continuous space. For the beta distribution for controlling population distribution, the parameters y and z vary from $(1, 1)$ to $(10, 10)$ with the step size 0.5. Therefore, there are total of $19 * 19 = 361$ population distributions. Among these population distributions, we first select five representative populations to report the evaluation results for the three designs and then present the results of all distributions.

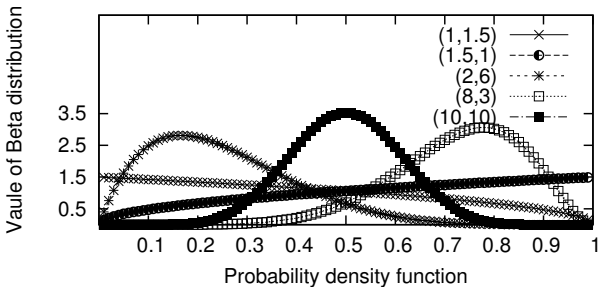

Figure 3.11: Five population distributions with their corresponding beta parameters.

The five population distributions are shown in Fig 3.11. The population distribution with parameters $(1.0, 1.5)$ [4], denotes that the majority of users are watching some specific channel sets with a single channel (the channel sets are arranged in lexicographical order) and there are a large number of channel sets of multiple channels with small number of users. By contrast, the population distribution with parameters $(1.5, 1.0)$ represents the opposite situation, where there a large number of users watching multiple channels. Both cases reflect the situation that the long tail channel popularity of current video streaming applications [98] [68], such as P2P-VoD and IPTV systems. Population distributions with parameters $(2, 6)$ and $(8, 3)$ represent the cases where there are some major events attracting most of the users (e.g. Olympic Games live broadcasting), and $(2, 6)$ denotes that most users watch channel sets with a single channel and $(8, 3)$ denotes that most users watch channel sets with

[4] we use the parameters (y, z) to represent a specific beta distribution hereinafter. For example, $(1.0, 1.5)$ denotes the beta distribution with parameters $(1.0, 1.5)$.

multiple channels. Finally, population distribution with parameters $(10, 10)$ denotes a normal population distribution. The other 356 distributions are variations of the five distributions and some examples are shown in Fig 3.12.

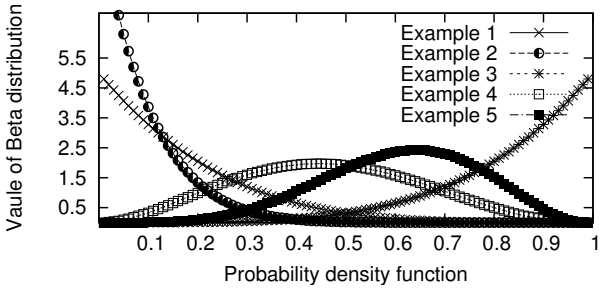

Figure 3.12: Examples of other beta distributions used for control bandwidth and population distributions.

Similar to controlling the populations of different channel sets, the bandwidth distribution of these channel sets are controlled by another beta distribution with parameters also varying from $(1, 1)$ to $(10, 10)$ with step size 0.5. The goal is to create different unbalanced bandwidth distributions among different channels covering the measurement studies [33] [32]. Therefore, for each specific population distribution, we evaluate a total number of $19 * 19 = 361$ bandwidth distributions. For all population distributions, we evaluate a total number of $361 * 361 = 130,321$ cases. The *size of feasible region for a specific design with a specific channel structure* is roughly defined as the number of feasible cases over the number of totally studied cases (i.e. 361 cases with different bandwidth distributions.). As an example, if the *PCA* design with a chain channel structure and a population distribution with parameters $(10, 10)$ has 30 feasible cases, the size of feasible solution space for this case is $\frac{30}{361} = 8.3\%$. The *size of feasible region for a specific design with all channel structures* is defined as the number of feasible cases over the total number of studied cases (i.e. $361 * 361 = 130,321$ cases with different bandwidth and population distributions.). If the *PCA* design with a chain channel structure has 20,000 feasible cases, then the overall size of feasible

region for this *PCA* with chain is $\frac{20{,}000}{130{,}321} = 15.3\%$.

For the chain channel structure, there are 10 channels and a peer can watch up to 2 consecutive channels. For the mesh channel structure, there are 10 channels and a peer can arbitrarily join/subscribe to up to 4 channels with maximum 200 channel sets. For the star channel structure, there are 20 channels and all peers watch a common channel and another channel. We simulate four streaming rates 300Kbps, 900Kbps, 1Mbps and 1.5Mbps in different simulation groups.

In the next subsections, we present the numerical results in two groups: 1) multi-channel systems with homogeneous streaming rate; 2) multi-channel systems with heterogeneous streaming rates for different channels.

3.5.3 Multi-Channel Systems With homogeneous Streaming Rate

We study the size of feasible solution spaces for three designs with homogenous streaming rate across all channels. Initially, we simulate the *NBA* and *PCA* designs with a low streaming rate (300Kbps), for all channel structures and for all bandwidth distributions and user population distributions. We summarize the results in Tables 3.1, 3.2, 3.3. The simulation results show that the feasible solution space (solution space for short) of *NBA* design with all channel structures is always empty, because for systems with unbalanced bandwidth, without the channel-aware bandwidth allocation strategy, *NBA* design can rarely lead to a bandwidth allocation that satisfies the bandwidth demands for all channels. From Table 3.1, we can see that the solution spaces of *PCA* design increases in the order of chain, star and mesh channel structure. The solution space of the *ACA* design is 100% for all channel structures, since we simulate the scenarios where system feasibility (refer to Definition 3.3.1.3) is always guaranteed.

Table 3.1: Relative feasible solution space size of PCA design for 300 Kbps streaming rate

Population Distribution	PCA mesh	PCA chain	PCA star
$(1.0, 1.5)$	98.6%	1.94%	22.2%
$(1.5, 1.0)$	99.4%	9.7%	23.3%
$(2.0, 6.0)$	98.9%	0.83%	20.2%
$(8.0, 3.0)$	99.7%	0.28%	31%
$(10, 10)$	98.9%	0.55%	46.5%
overall	99.1%	3.1%	35.2%

Table 3.2: Relative feasible solution space size of NBA design for 300 Kbps streaming rate

Population Distribution	NBA mesh	NBA chain	NBA star
$(1.0, 1.5)$	0%	0%	0%
$(1.5, 1.0)$	0%	0%	0%
$(2.0, 6.0)$	0%	0%	0%
$(8.0, 3.0)$	0%	0%	0%
$(10, 10)$	0%	0%	0%
overall	0%	0%	0%

Table 3.3: Relative feasible solution space size of ACA design for 300 Kbps streaming rate

Population Distribution	ACA mesh	ACA chain	ACA star
$(1.0, 1.5)$	100%	100%	100%
$(1.5, 1.0)$	100%	100%	100%
$(2.0, 6.0)$	100%	100%	100%
$(8.0, 3.0)$	100%	100%	100%
$(10, 10)$	100%	100%	100%
overall	100%	100%	100%

We then simulate the *NBA* and *PCA* designs with a much higher streaming rate (1Mbps) for all channel structures and for all bandwidth distributions and population distributions, which corresponds to the high definition videos (HD). The simulation results show that the solution space of *PCA* design does not change in HD scenarios. Therefore, we do not list the results. Similarly, the solution space of *NBA* design for all channel structures does not change with the increase of the streaming rate and is always empty. The solution space of the ACA design is 100% for all channel structures. Based on this group of simulations, we can conclude that for systems with homogenous streaming rate, the solution space is not affected by the streaming rate and the channel structure has a greater impact on the feasibilities of the three designs. The solution space size depends on the bandwidth imbalance among different channels and channel structures.

3.5.4 Multi-Channel Systems With Heterogeneous Streaming Rates

Many of the commercial multi-channel P2P streaming systems support heterogeneous streaming rates for different channels in order to provide different video qualities, such as high-definition videos and standard-definition videos. Therefore, in this subsection, we simulate multi-channel systems with heterogeneous streaming rates to investigate their impact on the solution space of the three designs. For all of the following simulations, the fractions of channels with streaming rate 300Kbps, 900Kbps and 1.5Mbps, are 40%, 30% and 30%, respectively.

The results are summarized in Tables 3.4, 3.5, 3.6. The solution space of the *NBA* design with all channel structures is still empty, due to the even higher bandwidth imbalance with heterogenous streaming rates. The streaming rate diversity greatly influences the solution space of the *PCA* design with chain and star channel structures,

Table 3.4: Relative feasible solution space size of PCA design for heterogenous streaming rates

Population Distribution	PCA mesh	PCA chain	PCA star
$(1.0, 1.5)$	96.1%	0%	0.55%
$(1.5, 1.0)$	97%	0%	0.28%
$(2.0, 6.0)$	97.5%	0%	0%
$(8.0, 3.0)$	93.9%	0%	0%
$(10, 10)$	97.5%	0%	0%
overall	96.7%	0.04%	0.02%

Table 3.5: Relative feasible solution space size of NBA design for heterogenous streaming rates

Population Distribution	NBA mesh	NBA chain	NBA star
$(1.0, 1.5)$	0%	0%	0%
$(1.5, 1.0)$	0%	0%	0%
$(2.0, 6.0)$	0%	0%	0%
$(8.0, 3.0)$	0%	0%	0%
$(10, 10)$	0%	0%	0%
overall	0%	0%	0%

Table 3.6: Relative feasible solution space size of ACA design for heterogenous streaming rates

Population Distribution	ACA mesh	ACA chain	ACA star
$(1.0, 1.5)$	100%	100%	100%
$(1.5, 1.0)$	100%	100%	100%
$(2.0, 6.0)$	100%	100%	100%
$(8.0, 3.0)$	100%	100%	100%
$(10, 10)$	100%	100%	100%
overall	100%	100%	100%

where the solution spaces of these two channel structures shrink to almost empty, as shown in Column 2 and 3 in Table 3.4. However, from Column 1 of Table 3.4, for the mesh channel structure, the solution space of the *PCA* is almost the same as that of the *ACA* design. Therefore, when building a multi-channel system with mesh channel structure, we can use the simple *PCA* design, since it can achieve a similar performance to that of *ACA*. The solution space change of *PCA* design with the increased bandwidth imbalance due to heterogenous streaming rates implies that *PCA* with mesh channel structure has the ability to balance bandwidth among different channel; whereas chain and star channel structures eliminate such ability. We investigate the possible reason below.

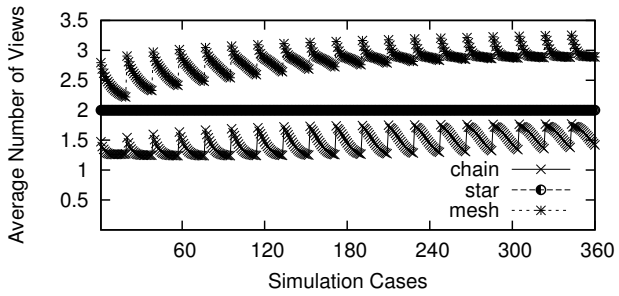

Figure 3.13: Average number of views for different channel structures of different simulations.

We define the average number of views for a specific channel structure as the sum of the number of channels watched by a peer across all peers divided by the total number of peers in the system. For example, assuming that the system has 2 peers, peer 1 watches one channel and peer 2 watches two channels. Therefore the average number of views in the system is $\frac{1+2}{2} = 1.5$. From the calculation, we can see that the average number of views depend on the channel structure and the population distribution. Fig 3.13 illustrates the average number of views for different channel structures against all simulated population distributions. From this figure, we can see that for the chain and star structures the average number of views is below 2. By contrast, the average

number of views for mesh structure is almost always greater than 2.5, which results in a larger solution space for *PCA*. Intuitively, the average number of views reflects the overlap among different overlays corresponding to different channels. Higher average number of views implies higher ability of balancing the bandwidth among channels with *PCA* design. The *ACA* design can also benefit from this result, since it can simply maintain the average number of views to be above some threshold instead of designing very complex schemes to maintain the helper group. For example, VUD [93] proposes a complex scheme to maintain the helper group.

3.6 Discussion and Chapter Summary

In this chapter, we focus on two fundamental problems in designing multi-channel P2P streaming systems: 1) what are the general characteristics of existing and potential designs; 2) and which design can be used to achieve the desired streaming quality with the lowest implementation complexity.

To answer the first question, we develop simple models based on linear programming and network flow graphs for *NBA*, *PCA* and *ACA* designs, which capture the main characteristics of cross-channel bandwidth allocation when designing multi-channel systems. We also prove that the *ACA* model with overhead is NP-Complete.

To answer the second question, we first study a special homogenous two-channel system and derive the closed-form results. Our results show that for this special case, there is no need to use the complex *ACA* design. The feasible solution space of *NBA* is much smaller than that of *PCA*. *NBA* can either support low quality videos or sustain the high streaming quality while consuming more bandwidth than *PCA*. Furthermore, the simple two-channel case implies that not only do the bandwidth and population distributions influence the feasible solution space, but the channel

structure of the system does as well.

Furthermore, we develop a C++ based simulator to numerically solve the cross-channel bandwidth allocation problems with various streaming rates, channel structures, bandwidth and population distributions. The extensive numerical results show that 1) *NBA* design can rarely achieve desired streaming quality in general cases; 2) for the mesh channel structure, which is the case for general multi-channel systems, the *PCA* design can achieve a similar performance as that of the *ACA* design even with heterogenous streaming rates, which indicates that we can build a general multi-channel system with a simpler design; 3) for special chain and star channel structures, which correspond to special P2P applications, *PCA* cannot achieve the desired streaming quality in most cases and therefore the more complex *ACA* design should be used. 4) for multi-channel systems, where the channels are isolated from one another (i.e. different channels do not overlap), *ACA* should always be used. In addition, based on simulations, we can use simple schemes instead of complicated membership management approaches in *ACA* design to provide desired streaming quality, which should maintain the average number of views in the systems above 2.5.

Due to simplicity reasons, our approaches of comparing the three designs have two limitations: 1) we do not study the system performance at transition states, which implies that our model lacks the ability of analyzing the performance under peer dynamics (e.g., channel switching, peer leaving); and 2) our models do not precisely model the implementation complexity of the three designs and our discussions are based on the intuition that *ACA* has the highest implementation complexity and *NBA* has the lowest implementation complexity. The latter will be worth studying in future work, since it will have broader impact on analyzing the design of distributed systems.

Chapter 4

Statistically Guaranteed Streaming Quality for P2P Live Streaming

4.1 Admission Control Problem in P2P Streaming Systems

Even though peer-to-peer (P2P) live streaming has been extensively studied [51], most of the literature focuses on how to provide *best-effort streaming quality* by efficiently using system bandwidth. That is, a P2P streaming system makes its best effort to provide good streaming quality by constructing an efficient P2P overlay architecture and running an efficient block scheduling algorithm; however, there is no guarantee about the provided streaming quality.

This chapter considers how to provide *guaranteed streaming quality* to a P2P live streaming system. Due to the dynamic nature of a P2P system, it is impossible to provide an absolute guarantee. Instead, we consider a statistical guarantee, which ensures that the streaming quality provided by a P2P system is statistically guaranteed. Statistically guaranteed streaming quality can greatly improve the satisfaction

of streaming users, compared to the best-effort streaming quality provided by the current P2P live streaming systems. In some cases, statistically guaranteed streaming quality is highly desired. For example, for a P2P live streaming system with some free channels and some paid channels, it is highly desirable that a paid channel provides a high streaming quality guarantee probability, whereas a free channel provides only best-effort streaming quality or a low streaming quality guarantee probability.

There are two different ways to provide statistically guaranteed streaming quality. First, at the individual peer level where the specific streaming quality provided to each peer is statistically guaranteed. Second, at the overall channel level where the average streaming quality provided to the whole channel is statistically guaranteed. The peer-level quality guarantee can provide a more accurate guarantee for each peer; however, it heavily depends not only on the overall system bandwidth but also on the underlying overlay construction method and block scheduling algorithm. On the other hand, the channel-level quality guarantee mainly depends on the overall system bandwidth. In addition, even though the channel-level quality guarantee cannot ensure the accurate streaming quality provided to each individual peer, it ensures the average streaming quality provided to all peers of a channel. In this chapter, we consider only the channel-level quality guarantee.

A fundamental problem in providing statistical channel-level quality guarantee is the statistical bandwidth guarantee problem, which is how to statistically guarantee that a channel has sufficient overall bandwidth for its streaming. We assume that the upload capacity of users is the only bottleneck for a P2P live streaming system. That is, the download capacity of a user is higher than the streaming rate, and bandwidth bottlenecks are located at the edges instead of the core of the Internet, which are reasonable assumptions [90] for the current Internet. With these assumptions, *the statistical bandwidth guarantee problem becomes how to guarantee that the probability*

for a channel to have sufficient overall upload bandwidth is higher than a threshold.

In order to achieve statistical bandwidth guarantee, we study a class of admission control algorithms, which admits or rejects a user based on the user information and the channel state. Another way to achieve statistical bandwidth guarantee is to drop users when a P2P system has insufficient overall bandwidth. However, dropping a user is usually considered more annoying to the user than rejecting a user, thus in this chapter, we consider only admission control algorithms. We are particularly interested in the user-behavior insensitivity of an admission control algorithm, which is whether the algorithm performance is insensitive to the fine statistics of user behaviors including both the distribution of user inter-arrival times and the distribution of user lifetimes. This is because we believe that user-behavior insensitivity is the key to designing an admission control algorithm that is robust and has predictable bandwidth guarantee in a dynamic and heterogeneous P2P system. We have the following observations from our results.

- *There is a tradeoff between the user blocking rate and user-behavior insensitivity when maintaining the same bandwidth guarantee.* Intuitively, this is because in order to reduce the user blocking rate, an algorithm uses more channel state information, which however makes the algorithm more sensitive to the statistics of user behaviors.

- *The statistical bandwidth guarantee achieved by an algorithm is more sensitive to the distribution change of user inter-arrival times than to that of user lifetimes.* Our simulation results show that the bandwidth guarantee probability obtained with a Poisson user arrival process can be used as the upper bound of the probability for general user arrival processes. The bandwidth guarantee probability obtained with an exponential user lifetime distribution can be used as a good estimate of the probability for general user lifetime distributions.

The rest of the chapter is organized as follow. Section 4.2 reviews the related work. Section 4.3 formulates the problem of statistical bandwidth guarantee. Section 4.4 proposes a class of admission control algorithms. Section 4.5 evaluates the performance of these admission control algorithms by simulation. Finally, Section 4.6 concludes the chapter.

4.2 Comparison With Existing Work

The proposed statistical streaming quality guarantee for a P2P live streaming system (referred to as P2P Quality of Service, or P2P QoS) is different from the traditional QoS for IP networking (referred to as IP QoS) [16, 11, 22]. IP QoS focuses on the bandwidth allocation in the Internet backbone, and it mainly considers the bandwidth mismatch between the edges and backbone of the Internet [22]. On the other hand, P2P QoS focuses on the upload-bandwidth allocation at the Internet edges, and it mainly considers the bandwidth asymmetry between the download and upload at the Internet edges. The current Internet has both asymmetric users (e.g. cable and Asymmetric Digital Subscriber Line (ADSL) users) and symmetric users (e.g Symmetric Digital Subscriber Line (SDSL) users), and we believe that the future Internet will remain as it is due to the heterogeneous nature of the Internet.

Admission control has been extensively studied in IP [36], ATM [65], wireless [6], and satellite [74] networks to provide bandwidth guarantee. However, existing admission control algorithms cannot be directly applied to P2P systems, since they are fundamentally different from previous types of networks, in that the total upload capacity of a P2P system is dynamic and dependent on the total number of users, which however is not the case for other types of networks.

Due to the dynamic nature of a P2P system, it is very challenging to provide

statistical service guarantee for a P2P system. There is very little related work on this topic. Bindal *et al.* [10] examine the factors that determine the statistical service guarantee in P2P file sharing applications, such as BitTorrent. They conclude that "*self-organizing P2P file distributions indeed need external help in order to provide QoS guarantees, but such guarantees are achievable with proper enhancements to the P2P network.*" Raghuveer *et al.* [69] consider how to ensure that each peer gets sufficient bandwidth with a high probability if the system has sufficient overall bandwidth. Kung *et al.* [41] and Xu *et al.* [94] use admission control to determine whether a peer should accept the request of another user to be a neighbor. Different from previous works, our work considers how to use admission control to ensure that the system has sufficient overall upload capacity with a high probability.

4.3 Problem Formulation

This section proposes a queueing model used to study the statistical bandwidth guarantee for a channel of a P2P live streaming system. The notation is summarized in Table 4.1.

Inspired by the stochastic fluid model by Kumar *et al.* [40], we model a channel of a P2P system by the queueing model shown in Figure 4.1, which captures two fundamental properties of P2P streaming, i.e., heterogeneous upload capabilities and peer churn. For example, CoolStreaming [42] conducted an experiment in 2006 with a streaming rate (denoted by r) of 768Kbps. Their results show that about 7% users can upload the stream at a rate higher than r, about 10% users can upload at a rate between r and $r/2$, about 20% users can upload at a rate between $r/2$ and $r/6$, and the remaining 63% users can upload at a rate even lower than $r/6$. This experiment clearly shows the different upload capabilities of different users.

Notation	Description
r	streaming rate
c^S	the upload capacity of the streaming server
λ^H	average arrival rate of super users
λ^L	average arrival rate of ordinary users
$1/\mu^H$	average life time of super users
$1/\mu^L$	average life time of ordinary users
c^H	upload bandwidth of a super user
c^L	upload bandwidth of an ordinary user
N^H	number of super users
N^L	number of ordinary users
C	total upload bandwidth of a system
R	total required bandwidth of a system
δ	required bandwidth guarantee probability

Table 4.1: Notation

Our model considers two classes of users (and can be extended to more classes). Class 1 contains a group of super users each capable of uploading at a high rate of c^H, and class 2 contains a group of ordinary users each capable of uploading at a low rate of c^L. We have $c^H > r > c^L$. A new user arrives at the system randomly with an average rate of λ^H and λ^L for a super user and an ordinary user, respectively. A new user may be admitted or rejected (also called blocked) by an admission control algorithm based on the upload bandwidth of the user and the current state of the system. If a user is admitted, it stays in the system for a random lifetime with average $1/\mu^H$ and $1/\mu^L$ for a super user and an ordinary user, respectively. Each class is modeled as a state-dependent processor-sharing (PS) queueing node [15] shown in Figure 4.1. The service rate of a queueing node depends on the current node state. For example, the service rate of the super user node (i.e., the top node in the figure) is $N^H\mu^H$, where N^H is the current number of super users.

We say that a system has sufficient upload bandwidth if $C \geq R$, where C and R denote the total upload bandwidth and the total required bandwidth, respectively. The total upload bandwidth C of the system is a function $f_C(\cdot)$ of the current system

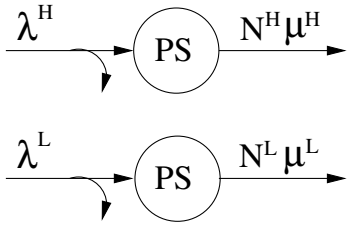

Figure 4.1: A state-dependent processor-sharing (PS) queueing model for a channel of a P2P live streaming system with two types of users: super users and ordinary users.

state as defined below

$$C = f_C\left(N^H, N^L\right) = N^H \times c^H + N^L \times c^L + c^S \tag{4.1}$$

where c^S is the upload capacity of the streaming server. The total required bandwidth R of the system is a function $f_R(\cdot)$ of the current system state as defined below

$$R = f_R\left(N^H, N^L\right) = \left(N^H + N^L\right) \times r \times \varepsilon \tag{4.2}$$

where $\varepsilon \geq 1.0$ indicates the control overhead and bandwidth inefficiency of the system, and depends on the underlying overlay architecture and block scheduling algorithm of the system. For example, packet-level simulation results [102] show that ε is about 1.15 for an overlay with a mesh-based overlay architecture and a random block scheduling algorithm.

Finally, *the statistical bandwidth guarantee problem is to determine whether a new user is admitted or rejected in order to guarantee* $\mathbb{P}[C \geq R] \geq \delta$, *where* δ *is the required bandwidth guarantee probability.*

We are interested in the following performance metrics when evaluating an admission control algorithm for achieving statistical bandwidth guarantee.

- *Implementation Difficulty*: How difficult is it to implement the algorithm?

- *Blocking Rate*: What is the average blocking rate for the algorithm to achieve a certain bandwidth guarantee probability δ?

- *Retry Robustness:* How robust is the algorithm in case that a rejected user repeatedly retries its admission request?

- *User-Behavior Insensitivity*: How insensitive is the algorithm to the fine statistics of user behaviors (i.e., user arrival process and user lifetime distribution)?

4.4 Admission Control Algorithms

In this section, we propose three admission control algorithms for achieving statistical bandwidth guarantee.

Inspired by insensitive load balancing work by Bonald *et al.* [12], we study the following three admission control algorithms for a channel.

1. *Static User Admission Control (SUAC)* admits all super users into the channel, and randomly admits an ordinary user with probability β_{SUAC}, where $\beta_{SUAC} \in [0, 1]$.

2. *Semi-Static User Admission Control (SSUAC)* admits all super users, and admits an ordinary user if the following condition is true, where $\beta_{SSUAC} \in [0, 1]$.

$$\frac{f_R\left(\mathbb{E}[N^H], N^L + 1\right)}{f_C\left(\mathbb{E}[N^H], N^L + 1\right)} \leq \beta_{SSUAC} \tag{4.3}$$

3. *Dynamic User Admission Control (DUAC)* admits all super users, and admits an ordinary user if the following condition is true, where $\beta_{DUAC} \in [0, 1]$.

$$\frac{f_R\left(N^H, N^L + 1\right)}{f_C\left(N^H, N^L + 1\right)} \leq \beta_{DUAC} \tag{4.4}$$

Since the upload bandwidth c^H of a super user is greater than the streaming rate r, all three algorithms always admit a super user. But they make different admission decisions for an ordinary user. SUAC is "static" in the sense that its admission decision for an ordinary user does not depend on the current system state (i.e. N^H and N^L), whereas DUAC is "dynamic" in that its admission decision depends on the current system state. SSUAC is "semi-static" since its admission decision depends only on the current state of ordinary users (i.e. N^L) but not on the current state of super users (i.e. N^H).

Below, we compare these three admission control algorithms according to the performance metrics described in Section 4.3.

- *Implementation Difficulty*: SUAC is the easiest to implement, since it does not need to measure anything. DUAC is the hardest to implement, since it is not trivial to accurately and quickly measure the current N^H and N^L. SSUAC is in the middle, since the average value of N^H can be obtained by using the history information and then it only needs to accurately and quickly measure the current N^L.

- *Blocking Rate*: Intuitively, since DUAC makes a dynamic decision based on the current channel state, it should achieve the lowest blocking rate for a given δ, whereas SUAC makes a static decision, it should achieve the highest blocking rate. The performance of SSUAC should fall somewhere in between. This is verified in the next section by numerical results.

- *Retry Robustness:* Both SSUAC and DUAC are robust in case of user retries, since for a given system state, no matter how many times a rejected ordinary user retries its admission request, it will always be rejected by both SSUAC and DUAC. However, with SUAC, a rejected ordinary user can keep retrying

its admission request until it is finally admitted. One possible solution for SUAC is to keep track of all recently rejected users (e.g. their IP addresses).

- *User-Behavior Insensitivity*: Intuitively, since DUAC is more dynamic than SUAC (i.e., more dependent on the channel state), DUAC is more sensitive to the fine statistics of user behaviors than SUAC. Specifically, we have the following insensitivity theorem.

We say that an admission control algorithm is *insensitive to the user lifetime distribution*, if the steady state distribution of a P2P live streaming system using this algorithm depends only on the average lifetime (i.e. $1/\mu^H$) of super users and that (i.e., $1/\mu^L$) of ordinary users, but does not depend on the lifetime distribution of super users and that of ordinary users. We have the following theorem.

Theorem 7. *Under the assumption that a super user arrives as a Poisson process and an ordinary user arrives as a Poisson process, the sufficient and necessary condition for an admission control algorithm to be insensitive to the lifetime distribution is that its admission decisions do not depend on the current number of super users (i.e., N^H).*

Proof: Let $a^H(N^H, N^L)$ and $a^L(N^H, N^L)$ denote the arrival rate of admitted super users and that of admitted ordinary users, respectively, when there are N^H supers users and N^L ordinary users.

According to the insensitivity theory of processor-sharing queueing networks developed by Bonald and Proutere [14, 12], the queueing model shown in Figure 4.1 is insensitive to the user lifetime distribution if and only if

$$a^H(N^H, N^L)a^L(N^H + 1, N^L) = a^H(N^H, N^L + 1)a^L(N^H, N^L)$$

Since every super user is admitted, we have $a^H(N^H, N^L) = \lambda^H$ for any N^H and N^L, and thus the sufficient and necessary condition for insensitivity becomes

$$a^L(N^H + 1, N^L) = a^L(N^H, N^L)$$

That is, the admission decision is independent of N^H, but can be dependent on N^L. □

It is then easy to see that both SUAC and SSUAC are insensitive to the user lifetime distribution under the Poisson user arrival assumption, but DUAC is sensitive to the user lifetime distribution. This implies that the bandwidth guarantee probability achieved by SUAC and SSUAC depends on the user lifetime distribution through the mean only.

We say that an admission control algorithm is *insensitive to the user arrival process*, if the steady state distribution of a P2P live streaming system using this algorithm depends only on the average arrival rate (i.e. λ^H) of super users and that (i.e., λ^L) of ordinary users, but does not depend on the inter-arrival time distribution of super users and that of ordinary users.

However, we do not have a theorem for the insensitivity to the user arrival process, and as shown in the next section, all three algorithms are sensitive to the user arrival process.

4.5 Numerical Results

In this section, we compare these three admission control algorithms with numerical results obtained for the queueing model shown in Figure 4.1. We study a P2P live streaming system with the following parameters. Streaming rate r is 3, upload rate c^H of a super user is 7, and upload rate c^L of an ordinary user is 1. According

to [40], these three values approximately reflect the actual settings of a typical P2P live streaming system (for example, when the rate unit is 100Kbps). The upload capacity c^S of the streaming server is 14. Limited by the memory space required to measure the state distribution $\mathbb{P}(N^H, N^L)$, we consider a small P2P system with an average of 25 super users and 50 ordinary users by setting $\lambda^H = 50$, $\lambda^L = 100$, and $\mu^H = \mu^L = 2$. If the time unit is 1 hour, this implies that a user stays in the system for an average of 0.5 hours. All results are obtained by simulating the system for 1,000,000 time units.

4.5.1 Blocking Rate of Ordinary Users

In this group of simulations, we study the blocking rate of each admission control algorithm in order to achieve a required bandwidth guarantee probability δ. We consider a system where both super users and ordinary users have a Poisson arrival process and an exponential lifetime distribution (other types of arrival processes and lifetime distributions are studied in Section 4.5.3). We simulate each algorithm with its parameter varying from 0 and 1, and then measure the blocking rate of ordinary users (i.e., the ratio of the number of rejected ordinary users to the total number of arrived ordinary users) and the bandwidth guarantee probability (i.e., the probability that the system has sufficient overall bandwidth). Finally, for each algorithm we find the blocking rate corresponding to a given bandwidth guarantee probability δ.

Figure 4.2 shows the blocking rate of each admission control algorithm for a bandwidth guarantee probability δ. We can see that for the same admission control algorithm, a higher blocking rate is required to achieve a higher bandwidth guarantee probability. We can also see that to achieve the same bandwidth guarantee probability, DUAC has the smallest blocking rate, followed by SSUAC and finally SUAC. For example, in order to achieve 99.9% bandwidth guarantee probability, the blocking

rate of DUAC is 19% (with β_{DUAC}=80%), the blocking rate of SSUAC is 41% (with β_{SSUAC}=48%), and the blocking rate of SUAC is 50% (with β_{SUAC}=50%).

In all the following simulations, we set β_{DUAC} = 80%, β_{SSUAC} = 48%, and β_{SUAC}=50%, so that all three algorithms can achieve the same bandwidth guarantee probability of 99.9% with a Poisson arrival process and an exponential lifetime distribution.

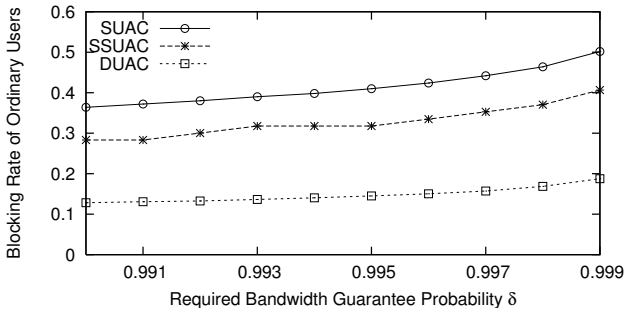

Figure 4.2: DUAC makes an admission decision based on the current channel state, and then has the smallest blocking rate among all three admission control algorithms in order to achieve a required bandwidth guarantee probability.

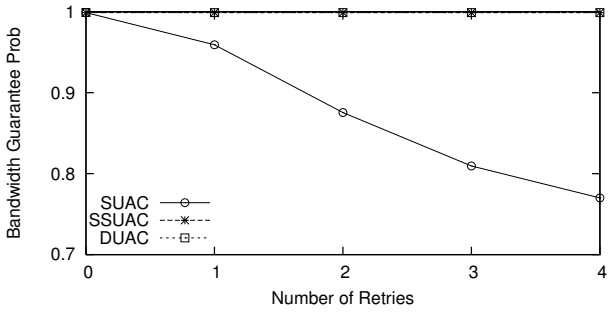

Figure 4.3: SUAC is not robust in case of user retries. That is, the bandwidth guarantee probability achieved by SUAC highly depends on how many times a rejected user retries its admission request.

4.5.2 Retry Robustness

Figure 4.3 shows the bandwidth guarantee probability achieved by each admission control algorithm when a rejected user retries its admission request. These results are

obtained for a system where both super users and ordinary users have a Poisson arrival process and an exponential lifetime distribution. We can see that the bandwidth guarantee probability achieved by both DUAC and SSUAC is always 99.9%, no matter how many times a rejected user retries its admission request. However, the bandwidth guarantee probability achieved by SUAC drops very quickly as a rejected user retries its admission request for more times.

4.5.3 User-Behavior Insensitivity

We first study the insensitivity to the user lifetime distribution. Motivated by the observation [78] that there are a small number of users who stay in the system for a very long time, we simulate a Pareto lifetime distribution. The lifetime of every user follows a Pareto distribution with a fixed mean $1/\mu^H = 1/\mu^L = 0.5$ and a shape parameter k varying from 1.52 to 1.52×8. The arrival process of a super user and an ordinary user is still a Poisson arrival process.

Figure 4.4 shows the state distribution $\mathbb{P}(N^H, N^L)$ of each admission control algorithm with $N^H = 10$ for $k = 1.52$ and 1.52×8. We can see that the state distribution of SUAC and SSUAC is insensitive to the user lifetime distribution, but the state distribution of DUAC is sensitive (although only slightly). This is consistent with Theorem 7. Figure 4.5 shows their bandwidth guarantee probabilities. We can see that the bandwidth guarantee probability achieved by SUAC and SSUAC does not depend on the lifetime distribution since they are insensitive to the lifetime distribution. We also observe that the bandwidth guarantee probability achieved by DUAC depends slightly on the lifetime distribution, and this is because the state distribution of DUAC is only slightly sensitive to the lifetime distribution.

Next, we study the insensitivity to the arrival process. Motivated by the observation [31] that the user arrival rates during different time intervals are different (e.g.,

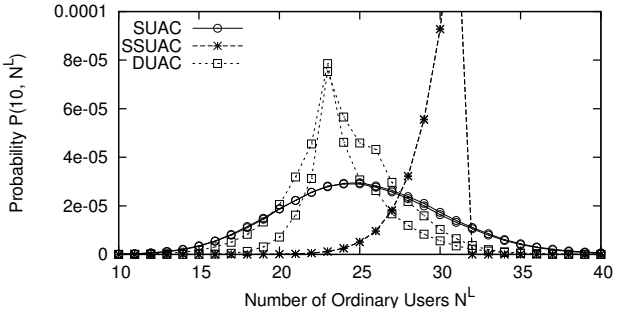

Figure 4.4: The state distribution of SUAC and SSUAC is insensitive to the lifetime distribution, but that of DUAC is sensitive (although only slightly). This is consistent with Theorem 7.

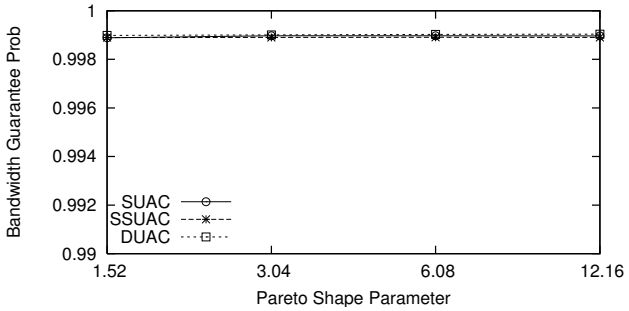

Figure 4.5: The bandwidth guarantee probability of SUAC and SSUAC does not depend on the lifetime distribution, and that of DUAC depends slightly on the lifetime distribution.

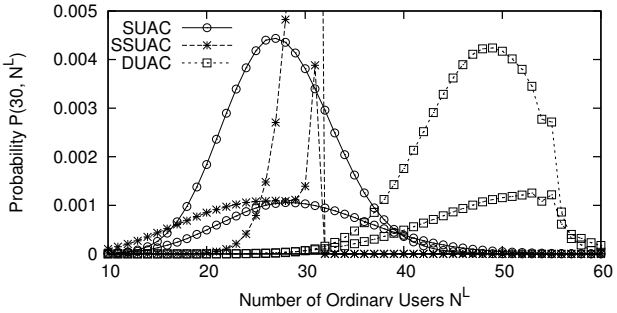

Figure 4.6: The state distribution of all three algorithms is sensitive to the user arrival processes.

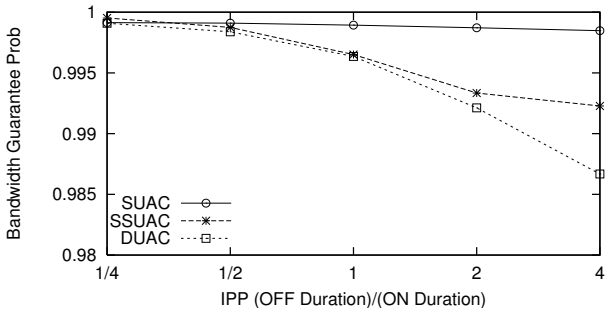

Figure 4.7: The bandwidth guarantee probability of SUAC slightly depends on the arrival processes, and that of SSUAC and DUAC highly depends on the arrival process.

higher arrival rate at the beginning of a program), we simulate an Interrupted Poisson Process (IPP). An IPP is an ON/OFF process, where both an ON period and an OFF period are exponentially distributed, and users arrive as a Poisson process only during an ON period. A super user arrives as an IPP with an average arrival rate of $\lambda^H = 50$, and an ordinary user arrives as an IPP with an average arrival rate of $\lambda^L = 100$. We set the average ON period to 1/16, which is much shorter than the average user lifetime (i.e., 1/2). We vary the average OFF period from 1/4 to 4 times of the average ON period, and we also adjust the user arrival rate in an ON period accordingly to maintain the same average user arrival rate. Both a super user and an ordinary user have an exponentially distributed lifetime.

Figure 4.6 shows the state distribution $\mathbb{P}(N^H, N^L)$ of each admission control algorithm with $N^H = 30$ when the average OFF period is 1/4 and 4 times of the average ON period. We can see that the state distribution of all three algorithms is sensitive to the user arrival process. Figure 4.7 shows their bandwidth guarantee probabilities. We can see that the overall bandwidth guarantee probability of SUAC slightly depends on the user arrival processes, and that of SSUAC and DUAC highly depends on the user arrival process. Note that, when the average OFF period is 1/4 times of the average ON period, an IPP is very similar to a Poisson process, and this is why

at that time, all three algorithms achieve 99.9% bandwidth guarantee probability.

Comparing Figure 4.2, Figure 4.4, and Figure 4.7, we can see that there is a tradeoff between the user blocking rate and user-behavior insensitivity. Intuitively, this is because in order to reduce the user blocking rate, an algorithm uses more channel state information, which however makes the algorithm more sensitive to the statistics of user behaviors. Comparing Figure 4.5 and Figure 4.7, we can see that the statistical bandwidth guarantee achieved by an algorithm is more sensitive to the distribution changes of user inter-arrival times than to the distribution changes of user lifetimes. The simulation results show that the bandwidth guarantee probability obtained with a Poisson user arrival process can be used as the upper bound of the probability for general user arrival processes. The bandwidth guarantee probability obtained with an exponential user lifetime distribution can be used as a good estimate of the probability for general user lifetime distributions.

4.6 Chapter Summary

In this chapter, we studied a class of admission control algorithms in order to provide statistically guaranteed streaming quality to a P2P live streaming system. In particular, we studied their insensitivity to the fine statistics of user behaviors. In the future, we plan to study admission control algorithms for a P2P live streaming system where a user can simultaneously watch multiple channels, and an admission control algorithm is used to decide whether to accept not only the request of a new user to join the system but also the request of an existing user to watch a new channel.

Chapter 5

On Providing Optimal Quality of Service in P2P Streaming Systems

5.1 Bandwidth Allocation and Block Scheduling in P2P Streaming Systems

Providing high quality of streaming service is an ultimate design goal for P2P streaming systems and numerous designs are proposed to achieve this goal. Earlier works on data swarming algorithms such as CoolStreaming [104] and GridMedia [29], show that data-driven approach is a very effective design for disseminating data among peers in an overlay network. Magharei *et al.* [55] suggest building mesh-based overlay for P2P streaming systems. All of these works have been used in commercial deployments (e.g., PPLive [66]). Recently, UUSee [77], one of the top three P2P streaming service providers in China, integrated network coding into its client software to improve system streaming quality [53].

Informally, the system-wide optimal streaming quality implies that all peers in the system are able to watch the subscribed channel at the source video rate. The system-

wide streaming quality mainly depends on how to use peer's upload bandwidth for video stream delivery and how to request useful data from neighboring peers. In this chapter, we study the problem of providing system-wide optimal streaming quality from the perspective of allocating upload bandwidth and requesting useful video data. Our goal is to propose generic methods of developing protocols for various applications and provide guidelines for system designers. We mainly focus on mesh-based overlays, since they are widely used in both academia and industry.

In P2P streaming systems, a peer can upload data to and download data from neighbors. Therefore, we distinguish it as a server peer and a client peer, respectively, when it takes different roles. Server peers allocate their corresponding upload bandwidth[1] to their neighboring peers for sending requested data to them, with the goal of minimizing the overlay congestion at these peers (e.g., allocating more bandwidth to a peer needing a large amount of data than the one needing a small amount of data.). This process is referred to as bandwidth allocation (BA). Simultaneously, client peers request missing data from neighboring peers to minimize the streaming cost (e.g., the total delay incurred by the requested data), which is referred to as content scheduling (CS)[2].

Existing designs usually solve the two problems separately. It means that a specific design only focuses on solving one of the two problems. For example, the network flow based optimal block scheduling algorithm [101] assumes that the bandwidth allocation problem has been solved and uses historical records to estimate the bandwidth allocation. Wu *et al.* [88] propose an optimal bandwidth allocation algorithm in co-

[1]In P2P literature, we assume that the bandwidth bottlenecks occur at network edges instead of the network core. Moreover, peers' upload bandwidth as the bottleneck is widely accepted by the P2P community, since the download bandwidth is much higher than the upload bandwidth in access networks.

[2]We use content scheduling to refer to the general process of requesting missing video data, because video data can be represented either as continuous substreams or discrete data blocks. If video stream is divided into blocks, we call the process as block scheduling.

existing overlays based on game theory, where network coding is used to simplify content scheduling. Figure 5.1 shows the interaction between *BA* and *CS* solutions. In practice, the algorithms for solving the two problems are carried out in an iterative manner. Algorithms solving *BA* take the solution of *CS* as an input and vice versa. Intuitively, optimal solutions to the two separate problem do not necessarily lead to system optimal solutions, because these solutions might not be Pareto Optimal (e.g., the optimal solution to *BA* might have a negative influence on *CS*'s performance). Therefore, to provide optimal quality of service for the whole P2P streaming system, we should carefully consider the interaction between *BA* and *CS*. Our contributions can be summarized as follows:

1) We first establish generic nonlinear optimization models for solving the two problems, which have good convergence properties with continuous and twice differentiable objective functions. Engineers can directly use these models to design specific protocols via replacing the generic objective functions with specific functions.

2) Instead of establishing complex joint optimization models, we analyze the interaction between *BA* and *CS* solutions with a two-player game theoretic model. Based on our analysis, if objectives of *BA* and *CS* are aligned, iteratively solving *BA* and *CS* can lead to Nash Equilibrium, which is system-wide optimal. If they are misaligned, the gap between system-wide suboptimal and optimal solutions can be arbitrarily large, where joint optimization should be used. For specific objective functions (e.g., affine), the gap is bounded, which can provide a lower bound for system-wide streaming quality.

3) We design and implement three groups of bandwidth allocation and block scheduling algorithms using a packet-level simulator, which demonstrate how to use our proposed models for designing new protocols. Moreover, extensive packet-level simulations confirm our analysis and help us gain a better understanding of the inter-

action between *BA* and *CS* solutions. With our results, engineers can easily choose proper designs for different application scenarios. Once a design is selected, the new algorithms can be implemented following our examples.

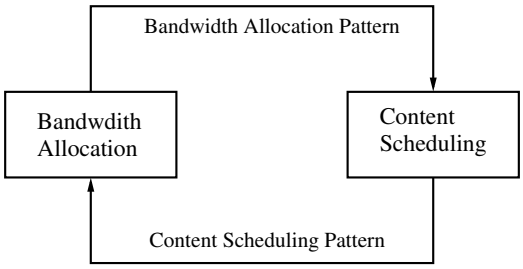

Figure 5.1: Interaction between BA and CS.

The rest of this chapter is organized as follows. Section 5.2 summarizes the related work in block scheduling and techniques used for studying efficiency loss between suboptimal and optimal solutions. Section 5.3 defines the optimization and game theory models with analytical results for studying interaction between *BA* and *CS*. We describe our simulation settings and results in Section 5.5. Finally, we conclude this chapter and discuss future works in Section 5.6.

5.2 Comparison With Existing Work

Bandwidth is essentially important for P2P streaming systems. Kumar *et al.* [40] establish a fluid model to study the impact of a peer's upload bandwidth and prove conditions for achieving universal streaming (i.e., the streaming rate achieved by the system is equal to the source video rate). Zhang *et al.* [102] study the bandwidth influence on data block scheduling algorithms via extensive packet-level simulations. They show that random block scheduling can achieve near-optimal streaming quality when the total upload bandwidth is at least 1.2 times of required bandwidth. Moreover, measurement studies and implementations [32] [42] also confirm that bandwidth

has a great impact on streaming quality for P2P streaming systems. Recently, researchers have proposed methods for improving streaming quality in multi-channel P2P streaming systems through optimally allocating bandwidth among different channels. Wu *et al.* [88] first study the bandwidth competition among coexisting overlays and propose an auction-based solution. The View-Upload Decoupling (VUD) [92] is a design that allows peers contributing upload bandwidth to unwatched channels. Our previous work [86] compares different bandwidth allocation schemes in multi-channel system via linear programming models. The above works are unaware of data blocks (content) and assume that bandwidth can be efficiently utilized by some content scheduling algorithms.

Compared with bandwidth, block scheduling algorithms are designed to fully utilize the bandwidth allocated among peers. Zhang *et al.* [101] propose an optimal block scheduling algorithm considering the priorities of missing data blocks based on a maximum flow model. Guo *et al.* [30] develop the AQCS block scheduling algorithm for P2P live streaming based on adaptive queueing models. Random Useful (RU) packet selection strategy [58] is proved to be rate optimal. All these algorithms assume that the end-to-end bandwidth allocation is known. For example, the block scheduling algorithm proposed in [101] uses the historical overlay link utilizations for future block scheduling. RU's proof assumes that the overlay link utilizations are known and fixed.

Network coding [79] has been theoretically proved to be useful for eliminating the data block diversities. Specifically, the coded data blocks can be considered as fluids and the theoretical optimal performance can be achieved [40] [58]. Tomozei *et al.* [23] apply random linear network coding to cost-efficient flow control in P2P systems and prove the rate optimal conditions. Liu *et al.* [53] integrated network coding to UUSee [77] and reported the large scale measurements. They conclude

that although network coding is very useful in improving streaming quality, it has weaknesses as well. For example, the client peer should send messages to stop the flows from server peers, which might possibly waste bandwidth due to redundant blocks. To sum up, network coding cannot perfectly substitute all block scheduling algorithms in all scenarios.

Game theoretic models have been used for analyzing the interactions among multiple parties. Liu *et al.* [52] analyze the interaction between overlay routing and traffic engineering with a two-player game model. Efficiency loss in selfish routing is well studied by Roughgarden *et al.* [70], where they establish bounds for routing with linear cost functions. Jiang *et al.* [38] propose a joint optimization model for solving the tussle between traffic engineering and content distribution. Similar to [38], DiPalantino *et al.* [26] study the cooperation between traffic engineering and content distribution as well, focusing on analysis of efficiency loss in non-cooperative scenarios. Although we borrow proof techniques from the above works, we establish new models for P2P streaming systems, which are different scenarios compared with routing.

5.3 Models for bandwidth allocation and content scheduling in P2P streaming

In this section, we describe the network model and the formal formulations of the bandwidth allocation and the content scheduling problems. The notation used in this chapter will be introduced in the following subsections and is summarized in Table 5.1.

We consider an overlay network represented by $G = (V, E)$, where V represents peers and E represents the neighboring relationship among peers. As mentioned in

Table 5.1: Key Notation Summary

Notation	Description
G	Overlay graph $G = (V, E)$. V set of nodes, E set of links
$NBR(i)$	neighbor set of peer i
K	number of substreams
C_i	upload capacity of peer i
d_i^k	peer i's demand for substream k
s_i^k	the amount of substream k held by peer i
r	streaming rate
π_{ij}	fraction of traffic flow sent from peer i to peer j
$\vec{\pi}$	bandwidth allocation pattern $\vec{\pi} = (\pi_{ij})$
a_{ij}^k	peer j's request of substream k from peer i
$\vec{a}$	content scheduling pattern $\vec{a} = (a_{ij}^k)$
e_{ij}	total amount of content requested by peer j from peer i
$f_{ij}(\cdot)$	the actual data flow sent from peer i to peer j, determined by $\vec{\pi}$ and $\vec{a}$
$S_{ij}(\cdot)$	the cost function used by bandwidth allocation
$D_{ij}(\cdot)$	the cost function used by content scheduling

Section 5.1, we consider a mesh-based overlay and therefore each peer i maintains a neighbor set $NBR(i)$, where $\forall j \in NBR(i)$, $(i,j) \in E$ and $i \in NBR(j)$. We use C_i to represent the upload capacity of peer i. There is a streaming server S with capacity C_s, which generates a video stream at rate r. The video stream is divided into a sequence of data blocks with sequence numbers $1, 2, 3 \ldots$ and data blocks are grouped into K substreams based on sequence numbers [29] [77] [42] (e.g., data blocks with sequence number k MOD $K = 1$ belongs to substream 1). Therefore, we establish flow-level models in this section for *analysis purpose only*, which can be revised into discrete packet-level models by replacing the continuous decision variables to discrete variables. When $K \to +\infty$, the continuous models approach the discrete models. In our implementations, we apply packet-level models for solving bandwidth allocation and content scheduling problems. Peers joining the same overlay network periodically exchange buffermaps to learn the content availability at neighboring peers, which is the basis for requesting data from neighbors. Let $d_i^k \geq 0$ denote peer i's demand for substream k, which is determined by the content availability of neighboring peers.

Given the streaming rate r and client peers' demand, server peers have the ability to control the end-to-end flow among neighboring client peers to sustain the streaming rate by solving problem BA. For each server peer i, the total flow out of i cannot exceed its upload capacity C_i. In a slight abuse of notation, we also use BA to denote *solutions* to the BA problem. For each $i \in V$, BA chooses a flow rate distribution for all $j \in NBR(i)$. We use π_{ij} to represent the fraction of flow that peer i allocates to peer j.

Definition 1. *A bandwidth allocation pattern is a vector* $\vec{\pi} = (\pi_{ij})$, $\forall i, j \in V$, *with* $\pi_{ij} \in [0,1]$ *and* $\sum_{j \in NBR(i)} \pi_{ij} = 1$*; that is the fraction of traffic flow sent from server peer* i *to client peer* j *with the consideration of neighbors' content demands from peer* i.

Suppose BA has determined the bandwidth allocation pattern $\vec{\pi}$. Then, each client peer must select server peers from which to request required contents. Specifically, client peer j will choose a rate distribution over its neighbor set $NBR(j)$ to satisfy its demand for each substream k, such that $\sum_{i \in NBR(j)} a_{ij}^k = d_j^k$, where a_{ij}^k denotes peer j's demand of substream k from peer i and d_j^k denotes peer j's total demand of substream k. We use $e_{ij} = \sum_{k \in K} a_{ij}^k$ to represent the total content flow demand between client peer j and server peer i. The content scheduling pattern can be defined as follows:

Definition 2. *A content scheduling pattern is a vector $\vec{a} = (a_{ij}^k)$, $\forall i, j \in V, k \in K$; that is the rate demand for substream k of peer j, which will be requested from its neighbors.*

With Definition 1 and Definition 2, the amount of traffic flow sent from server peer i to client peer j can be calculated with the following equation:

$$f_{ij}(\vec{\pi}, \vec{a}) = \min(\pi_{ij} \sum_{p \in NBR(i)} \sum_{k \in K} a_{ip}^k, \sum_{k \in K} a_{ij}^k) \tag{5.1}$$

Moreover, the total flow rate sent by peer i cannot exceed its upload capacity, which corresponds to the constraint $\sum_{j \in NBR(i)} f_{ij} \leq C_i$. Note that the system-wide optimal streaming quality can be achieved if and only if $\sum_{p \in NBR(j)} f_{pj} \geq r$, $\forall j \in V$,.

Based on the above discussions, we model the bandwidth allocation problem and content scheduling problem from the system perspective, in terms of minimizing social objectives (i.e., the overall cost incurred by peers in the system). We must emphasize that BA should be solved when a peer acts as a server and CS should be solved when it acts a client.

5.3.1 Bandwidth Allocation (BA) Model

According to the above discussions, f_{ij} is the traffic that is transmitted on overlay link (i, j) (from peer i to peer j). We assign a nonnegative, nondecreasing, convex, and differentiable cost function S_{ij} to represent the cost incurred when peer i transmits data flow f_{ij}. The cost can be interpreted as the congestion at peer i experienced by peer j [88] or the latency on overlay link (i, j) due to flow f_{ij} [81]. Then, BA tries to find an optimal bandwidth allocation pattern $\vec{\pi}$ over all possible rate allocations that minimizes the total cost of the whole overlay network. The constraints are: 1) The total flow sent by server peer i cannot exceed its upload capacity; 2) The total flow received by client peer j should satisfy its demand. With previously defined notations, the BA problem can be formulated as the following optimization problem: $\mathbf{BA}(\vec{\pi}|\vec{a})$:

$$\text{minimize} \sum_{\text{i}\in\text{V}} \sum_{\text{j}\in\text{NBR(i)}} \text{S}_{\text{ij}}(\text{f}_{\text{ij}}(\overrightarrow{\pi}, \overrightarrow{a})) \tag{5.2}$$

Subject to:

$$\sum_{j:j\in NBR(i)} f_{ij} \leq C_i, \forall i \in V$$

$$\sum_{i:i\in NBR(j)} f_{ij} \geq \sum_{k\in K} d_j^k, \forall j \in V$$

$$\sum_{j\in NBR(i)} \pi_{ij} = 1, \forall i, j \in V$$

$$\pi_{ij} \geq 0, \forall i, j \in V$$

5.3.2 Content Scheduling (CS) Model

Client peers learn the data availability at server peers via buffermap exchange. Then, they only decide which server peer should be selected to satisfy their demands on each substream k. They are unable to control the actual flow carried on overlay links, which

is determined by BA. For example, the min-cost content scheduling algorithm [101] maintains a historical record of overlay link utilization, which is used as the upload bandwidth estimate for future scheduling decisions. Random Useful (RU) [58] assumes that the overlay link utilization is known when proving rate optimality, where an estimation/measurement approach should be used for real implementations.

To formulate an abstract model, each overlay link associates with a price, which represents the streaming cost on that link (e.g., latency depending on the data flow through it [101]). In P2P live streaming systems, each substream k might have different priorities at the moment of content scheduling. For example, there may be fewer server peers holding substream k_1 than the the ones holding k_2. Therefore, substream k_1 should have higher priority than k_2, which corresponds to rarest-first scheduling approach [101]. Generally speaking, the priority of each substream is a constant value, which depends on the data availability at server peers and can be calculated before scheduling. At each scheduling round, the priorities can be considered as constants. Therefore, we can adjust the price (defined below) of each substream seen by peer j by its corresponding constant priority. To achieve a neat formulation, we do not include the priority below, which is considered in our simulations.

According to the above discussion, we establish the formal model for CS. Each overlay link (i, j) is assigned a nonnegative, nondecreasing, and continuous price function $p_{ij}(\star)$. We use $e_{ij} = \sum_{k \in K} a_{ij}^k$ to represent the total data flow requested by peer j through overlay link (i, j). Since BA and CS are carried out iteratively, client peers face a fixed bandwidth allocation pattern $\vec{\pi}$, when scheduling data requests. In practice, peer j maintains the historical values of π_{ij} as a bandwidth allocation estimate for next round scheduling. Therefore, the price of link (i, j) can be represented as $p_{ij}(e_{ij}, \pi_{ij})$, where π_{ij} is considered as a constant. Note that server peer's upload bandwidth is critical for streaming quality [40] [102] and the price function mainly

reflects the latency due to the bandwidth allocation pattern $\vec{\pi}$. From peer i's point of view, $\vec{\pi}$ is determined by BA, which is influenced by all other peers' (i.e., $\{V \setminus i\}$) CS decisions.

Let $D_{ij}(e_{ij}, \pi_{ij}) = \int_0^{e_{ij}} p_{ij}(t, \pi_{ij})dt$. Then, the content scheduling problem can be formulated as follows:

CS$(\vec{a}|\vec{\pi})$:

$$\text{minimize} \sum_{\text{j}\in\text{V}} \sum_{\text{i}\in\text{NBR(j)}} \text{D}_{\text{ij}}(\text{e}_{\text{ij}}, \pi_{\text{ij}}) \tag{5.3}$$

Subject to:

$$\sum_{i\in NBR(j)} a_{ij}^k = d_j^k, \forall j \in V, \forall k \in K \tag{5.4}$$

$$a_{ij}^k \leq s_i^k, \forall i, j \in V, \forall k \in K \tag{5.5}$$

$$d_j^k \leq \sum_{i\in NBR(j)} s_i^k, \forall j \in V, \forall k \in K \tag{5.6}$$

$$a_{ij}^k \geq 0, \forall i, j \in V, \forall k \in K \tag{5.7}$$

The objective (5.3) of CS is to minimize the streaming cost (e.g., requesting data from high capacity neighbors) when requesting data from neighboring peers. Constraints (5.5) and (5.6) guarantee the feasibility of CS: 1) Peer j cannot request data from peer i, which is not held by peer i; and 2) Peer j's total demand for substream k cannot exceed the total amount of substream k held by its neighbors. The two constraints are guaranteed via buffermap exchange in P2P streaming systems. In real systems, optimization problems (5.2) and (5.3) can be solved with distributed algorithms based on decoupling theory [63].

5.3.3 Characteristics of BA and CS

Peers make decisions on *BA* and *CS* independently and selfishly in P2P streaming systems [104] [29] [88] [81]. Each of their corresponding decision sets constitutes a best response for peers in the overlay network, which is in the style of Wardrop equilibrium [7]. We define best responses for peers and prove the existence of the best responses as follows.

According to the *CS* formulation (5.3), a peer chooses rate distribution $\vec{a}$ to minimize the price it faces to obtain video streams. Given a bandwidth allocation pattern $\vec{\pi}$ and fixing other users' decisions, a peer i chooses a rate distribution over its neighbor set to satisfy its demand defined by constraint (5.4) with the goal of minimizing the streaming cost. That is, for each $i \in NBR(j)$, with $e_{ij} = \sum_{k \in K} a_{ij}^k > 0$, the following inequality holds, where $\tilde{e_{ij}}$ denotes any feasible value:

$$p_{ij}(e_{ij}, \pi_{ij}) \leq p_{ij}(\tilde{e_{ij}}, \pi_{ij}) \tag{5.8}$$

Definition 3. *Best Content Scheduling Pattern: Given a bandwidth allocation $\vec{\pi}$, a content scheduling pattern $\vec{a}$ is the best content scheduling pattern for peers, if and only if $\forall j \in V$ and for each $i \in NBR(j)$, with $e_{ij} = \sum_{k \in K} a_{ij}^k > 0$, where e_{ij} can be arbitrarily close to 0, inequality (5.8) holds.*

With the Definition 3, we are ready to show the existence of equilibrium of *CS* when peers make their decisions independently and selfishly.

Proposition 1. *Given a bandwidth allocation pattern $\vec{\pi}$, a best content scheduling pattern exists. Then, $\vec{a}$ is the best content scheduling pattern if and only if it is the solution to the optimization problem **CS**$(\vec{a}|\vec{\pi})$ (5.3).*

Proof: Since $p_{ij}(\star) = D'_{ij}(\star)$ is nonnegative, nondecreasing, $D_{ij}(\star)$ is convex and differentiable. The conditions for convex programming problem are readily seen to be

equivalent to Definition 3 of best content scheduling pattern. Therefore, the existence of solution to problem (5.3) guarantees the existence of best content scheduling pattern. □

Similarly, we establish the best bandwidth allocation pattern for BA. The solution to optimization problem (5.2) is a rate distribution minimizing the overall congestion incurred by peers, which implies that the marginal cost of the chosen rate distribution is at least as good as any other rate distributions. Precisely, for each peer j, for all neighbors $i \in NBR(j)$, the marginal cost of the chosen rate distribution should be, where $\tilde{\pi_{ij}}$ denotes any feasible value:

$$\sum_{i \in NBR(j)} S'_{ij}(\pi_{ij}) \leq \sum_{i \in NBR(j)} S'_{ij}(\tilde{\pi_{ij}}) \tag{5.9}$$

It can be easily verified that for an overlay network $G = (V, E)$, the above inequality (5.9) is implied by the Karush-Kuhn-Tucker (KKT) conditions of optimization problem (5.2). We establish the relationship between optimal solution to (5.2) and inequality (5.9) with the proposition below.

Proposition 2. *Given a fixed content scheduling pattern $\vec{a}$, there exists a bandwidth allocation pattern $\vec{\pi}$ that is optimal to problem (5.2) and satisfies inequality (5.9). Furthermore, any bandwidth allocation $\vec{\pi}$ that satisfies inequality (5.9) is optimal to (5.2).*

Proof: For the first part of Proposition 2, the optimality of (5.2) implies the inequality (5.9), according to the property of optimal solution. For the second part, it follows the KKT conditions of (5.2), which are necessary and sufficient conditions for the optimality of (5.2). □

Definition 4. *Best Bandwidth Allocation Pattern: Given a fixed content scheduling pattern $\vec{a}$, a bandwidth allocation pattern $\vec{\pi}$ is the best bandwidth allocation pattern for peers, if and only if $\forall j \in V$, inequality (5.9) holds.*

In this subsection, we show the existences of optimal solutions to problems (5.2) and (5.3). We also formally define the best responses of users for solving *BA* and *CS*, which is the basis for studying the interaction between *BA* and *CS* with the two-player game defined in Section 5.3.4.

5.3.4 BA-CS Game and Equilibrium Concept

In current P2P streaming systems, *BA* and *CS* are carried out independently and we study the interaction between them based on the above formulations. They are coupled with each other through delivering video streams using peer's upload bandwidth. Both *BA* and *CS* optimize their own objectives with corresponding strategies. Therefore, we can model their interaction with a two-player non-cooperative game.

Definition 5. *BA-CS game consists of a tuple [N,A,U]. The player set* $N = \{BA, CS\}$. *The action set* $A_{BA} = \{\vec{\pi}\}$ *and* $A_{CS} = \{\vec{a}\}$, *where the feasible sets of* $\vec{\pi}$ *and* $\vec{a}$ *are determined by the constraints of optimization problems (5.2) and (5.3), respectively. The utility functions are* $U_{BA} = -S_{ij}(\star)$ *and* $U_{CS} = -D_{ij}(\star)$.

Figure 5.1 shows the interaction between *BA* and *CS*. Based on Section 5.3.3, both *BA* and *CS* play the best response strategy. That is, *BA* always minimizes its objective function (5.2) given the *CS*'s strategy $\vec{a}$ and *CS* always minimizes streaming cost incurred by users given *BA*'s strategy $\vec{\pi}$. This procedure can be formally described as follows, where *BA* and *CS* take turns to optimize their corresponding objectives considering the strategy of the other player as a constant. Therefore, for the $(k+1)$th iteration, equations (5.10) and (5.11) show their interaction

$$\overrightarrow{\pi}^{\mathbf{k+1}} = \mathbf{argmin}_{\overrightarrow{\pi}}\mathbf{BA}(\overrightarrow{a}^{\mathbf{k}}) \tag{5.10}$$

$$\overrightarrow{a}^{\mathbf{k+1}} = \mathbf{argmin}_{\overrightarrow{a}}\mathbf{CS}(\overrightarrow{\pi}^{\mathbf{k}}) \tag{5.11}$$

BA and *CS* are usually carried out in different time-scales. For example, *BA* problem might be solved in the order of seconds or minutes; while *CS* problem might be solved in the order of hundreds of milliseconds. Since peers make their decisions independently and only exchange data with their direct neighbors, we assume that each of the two players (i.e., *BA* and *CS*) has fully solved its own optimization problem before the other player starts, when discussing the equilibrium and performance loss below. We will remove this assumption in our simulations. Our simulation results show that they have good convergence properties with various parameter settings (e.g., system-wide bandwidth supplies), when *BA* and *CS* are carried out in the order of seconds and milliseconds, respectively.

Definition 6. ***Nash Equilibrium****: A strategy profile* $A^\star = \{A_{BA}(\vec{\pi^\star}), A_{CS}(\vec{a^\star})\}$ *is a Nash equilibrium if, for both players* BA *and* CS*,*

$$\mathbf{BA}(\overrightarrow{\pi^\star}|\overrightarrow{a^\star}) \leq \mathbf{BA}(\overrightarrow{\pi}|\overrightarrow{a^\star}) \tag{5.12}$$

$$\mathbf{CS}(\overrightarrow{a^\star}|\overrightarrow{\pi^\star}) \leq \mathbf{CS}(\overrightarrow{a}|\overrightarrow{\pi^\star}) \tag{5.13}$$

The following theorem shows that the Nash equilibrium of *BA*-*CS* game exists. Specifically, we show the equilibrium condition with general cost functions used by *BA* and *CS*, which are continuous, non-decreasing, and convex.

Theorem 8. *The BA-CS game has a Nash equilibrium, when the cost functions are continuous, non-decreasing, and convex.*

Proof: Based on the Nash theorem [59] [61], to prove the existence of an equilibrium, it is sufficient to show that the following two conditions hold: 1) for each player, its strategy space is non-empty, convex and compact; and 2) for each player, its utility function is quasi-concave. The constraints for BS's optimization problem (5.2) are affine inequalities. Therefore, it is a convex and compact set. The feasibility is guaranteed by the assumption that there is sufficient bandwidth to sustain the streaming rate of all peers, which implies that the BA's strategy space is non-empty. The cost function $S_{ij}(\star)$ is convex and we can verify that the objective function (5.2) is quasi-convex on $\vec{\pi}$, which implies the utility function defined in Definition 5 is quasi-concave. Similarly, CS's strategy space is defined by the set of constraints of optimization problem (5.3), which are also affine equalities and inequalities. Therefore, its strategy space is a convex and compact set. Since client peers request data based on the data availability obtained by buffermap exchange, CS's strategy space is non-empty. Based on the proof of Proposition 1, $D_{ij}(\star)$ is convex and non-decreasing and thus we can verify that the objective function (5.3) is quasi-convex. The utility of CS is quasi-concave. Therefore, the two conditions of Nash theorem are satisfied. □

Although Theorem 8 proves the existence of a Nash equilibrium of BA-CS game, the Nash equilibrium might not be unique and social optimal in general cases and we will discuss the performance loss with the defined game in the following section. Furthermore, from an algorithmic perspective, the existence of equilibrium does not necessarily guarantee that the procedure defined by (5.10) and (5.11) can find one. We use simulations to verify the convergence of the game in Section 5.5.

5.4 Interaction of BA-CS: A Game-Theoretic Analysis

In this section, we analyze the interaction between *BA-CS* based on the two-player game defined above. In particular, we first study a scenario, where the Nash equilibrium is unique and corresponds to the social optimal solution. Then, we use two examples to show the performance loss in the two-player game. Finally, we discuss the performance loss in general cases, in terms of different cost functions used by *BA* and *CS*.

5.4.1 Social Optimality under Same Objectives

We study a special case, where the *BA* and *CS* have the same objective. That is, they optimize the same objective function

$$BA = CS = \sum_{i \in V} \sum_{j \in NBR(i)} \Phi_{ij}(f_{ij}(\vec{\pi}, \vec{a})) \tag{5.14}$$

One application scenario of this case is using a push-based data delivery scheme in a relatively stable system, where server peers have the precise information about content demand at client peers and push fresh data to them. For example, in a tree-based overlay network for live streaming, parent peers push newly generated data to child peers. Random Useful (RU) [58] is another example in mesh-based overlay network, where the overlay link capacity is predetermined. In addition, the network-coding based data delivery scheme [53] is a good example of this case as well. However, the implementations of RU [43] shows that the original RU scheme generates a large volume of redundancy packets due to peer dynamics. Network coding based scheme [53] also generates redundancy packets in real implementations, in that server

peers continuously push data streams to client peers until they receive the explicit stop-faucet messages from client peers. That is why even though these approaches have theoretically optimal performances, there is still a need for other approaches.

Since *BA* and *CS* optimize the same objective function, the P2P streaming system can achieve a socially optimal operating point for data delivery. As a system designer, we are interested in the question whether the iterative two-player game defined in Section 5.3.4 can achieve the optimal point via alternating best-response decisions. To answer this question, we define the socially optimal point as the solution to the following optimization problem

BA-CS-Same($\vec{x}$):

$$\text{minimize} \sum_{\text{i}\in\text{V}} \sum_{\text{j}\in\text{NBR(i)}} \Phi_{\text{ij}}(\sum_{\text{k}\in\text{K}} \text{x}_{\text{ij}}^{\text{k}}) \tag{5.15}$$

Subject to:

$$\sum_{j\in NBR(i)} \sum_{k\in K} x_{ij}^k \leq C_i, \forall i \in V \tag{5.16}$$

$$\sum_{i\in NBR(j)} x_{ij}^k = d_j^k, \forall j \in V, \forall k \in K \tag{5.17}$$

$$x_{ij}^k \geq 0, \forall i, j \in V, \forall k \in K$$

where x_{ij}^k represents the rate of substream k, which is pushed from peer i to peer j. At the end of Section 5.4.1, we describe the relationship between x_{ij}^k and a_{ij}^k. Optimization problem (5.15) reflects the coordination between *BA* and *CS*, which allows the server peers sending the useful data to client peers in any way that minimizes the streaming cost. Therefore, this problem establishes a performance upper bound on P2P streaming. Since this problem can be considered as a special case of the general two-player game defined in Section 5.3.4, it has a Nash equilibrium as shown in Theorem 8.

In general, the existence of Nash equilibrium does not imply its uniqueness. If

there is no fresh data between peers i and j, peer i's decision on sending data to j can be arbitrary without changing its utility, because the actual flow between peer i and j is always 0. To avoid such a situation, we assume that there exist infinitesimal data flows between a peer i and its direct neighbors, which can be arbitrarily close to 0. In the following the theorem, we shown that the Nash equilibrium of the *BA-CS* game is unique and is exactly the social optimal solution to problem (5.15), when *BA* and *CS* optimize the same objective function.

Theorem 9. *Given* BA *and* CS *optimize the same continuous, non-decreasing, and convex objective function and there exists non-zero content demand between any peer pair, the Nash equilibrium of BA-CS game is unique and is a social optimal solution to problem (5.15).*

Proof Sketch: The existence of Nash equilibrium of this game is guaranteed by Theorem 8, because the problem (5.15) is a special case of the general problem. The uniqueness of the Nash equilibrium and optimal solution is guaranteed by convexity. The idea of proving that the unique equilibrium is optimal is to prove the equivalence of Nash equilibrium and the global optimum. □

Based on the above discussion, the flow $\sum_{k\in K} x_{ij}^k$ is actually the optimal flow on overlay link (i, j). Therefore, given an optimal flow vector $\vec{x}$, we can obtain the content scheduling pattern as $\vec{a} = \vec{x}$. The bandwidth allocation pattern is a vector $\vec{\pi}$, where $\pi_{ij} = \sum_{k\in K} x_{ij}^k / \sum_{j\in NBR(i)} \sum_{k\in K} x_{ij}^k$.

5.4.2 Examples of Performance Loss

The optimization problem (5.15) establishes a lower bound on performance loss of *BA* and *CS* in P2P streaming, which indicates that the BA-CS interaction does not sacrifice performance under the same objective. In this subsection, we use two simple

examples to show that the performance loss can be arbitrarily large, if the objective functions of BA and CS are misaligned.

5.4.2.1 Example 1: W-shaped topology with pull-based method

Figure 5.2 shows a W-shape topology consisting of 5 peers, where peers A, B and C are server peers with node capacity 1. The client peers D and E have one unit flow demand for a specific type of content, which can be served by the three server peers. Both peers D and E decide which server peer should be selected to request the data (i.e., pulling required data), based on the link latency functions defined below. Since peers A and C only serve a single peer, their only decision is to send the data once requested by the corresponding client peer. However, peer B has two choices, if D and E both request data from it. Suppose that B prefers D based on its cost functions on the two peers.

The latency function for the four overlay links are defined as follows: $D_{AD}(f_{AD}) = 0.9f_{AD}$, $D_{BD}(f_{BD}) = 0.1f_{BD}$, $D_{BE}(f_{BE}) = 0.1f_{BE}$ and $D_{CE}(f_{CE}) = 1000f_{CE}$. Further, assume $p_{ij} = D'_{ij}(f_{ij})$, to be the price for each overlay link. Therefore, $P_{\mathrm{AD}}(f_{AD}) = 0.9$, $P_{BD}(f_{BD}) = 0.1$, $P_{BE}(f_{BE}) = 0.1$ and $P_{CE}(f_{CE}) = 1000$. Client peers request data from peers that minimize the streaming cost (i.e., through the links with lowest price). Peer E has known that peer B prefers peer D based on historical information as stated in [101]. Therefore, E sends its request to C, even though the price of link CE is much higher than that of BE. Otherwise, E will suffer starvation in this round, in that B will send data to D.

Based on the above scenario, the overall streaming cost incurred by both D and E is $1 * 0.1 + 1 * 1000 = 1000.1$. However, if the server peer B aligns its objective function to minimize the latency incurred by client peers, the cost incurred by D and E is $1 * 0.9 + 1 * 0.1 = 1$, which implies that D requests data from A and E requests

data from B. We can see that the streaming cost with misaligned objective functions is about 1000-orders of magnitude higher than the cost of optimal solution. This example shows that the pull-based P2P streaming system with misaligned objective functions sacrifices the performance in terms of latency.

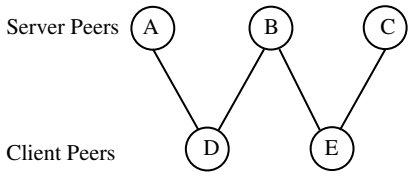

Figure 5.2: An example of performance loss: 1000-orders of magnitude higher than the cost of optimal flow.

5.4.2.2 Example 2: X-shaped topology with push-based method

Example 2 illustrates a toy system with four peers using push-based method, as shown in Figure 5.3. Peers A and B are sever peers with node capacity 1. Each of the client peers C and D has one unit flow demand, which will be pushed by server peers. The cost functions (congestion) of each overlay link seen by server peers are: $S_{AC}(f_{AC}) = f_{AC}$, $S_{AD}(f_{AD}) = \frac{1}{1-f_{AD}}$, $S_{BC}(f_{BC}) = \frac{1}{1-f_{BC}}$ and $S_{BD}(f_{BD}) = f_{BD}$. The latency functions of overlay links seen by client peers are: $D_{AC}(f_{AC}) = 1$, $D_{AD}(f_{AD}) = f_{AD}$, $D_{BC}(f_{BC}) = f_{BC}$ and $D_{BD}(f_{BD}) = 1$.

Server peers A and B schedule how to deliver the data flow by solving the optimization problem (5.2). The rates allocated over corresponding overlay links are: $\pi_{AC} = (\frac{1}{2})^{\frac{1}{3}}$, $\pi_{AD} = 1-(\frac{1}{2})^{\frac{1}{3}}$, $\pi_{BC} = 1-(\frac{1}{2})^{\frac{1}{3}}$ and $\pi_{BD} = (\frac{1}{2})^{\frac{1}{3}}$. However, the rates that have minimal overall latency incurred by client peers should be $\pi_{AC} = \frac{1}{2}$, $\pi_{AD} = \frac{1}{2}$, $\pi_{BC} = \frac{1}{2}$ and $\pi_{BD} = \frac{1}{2}$. The rates are determined by solving problem (5.3) with overlay link latency functions defined above.

From this example, we can see that the push-based data delivery approach is not necessarily better than the pull-based method, if the objective functions of BA and

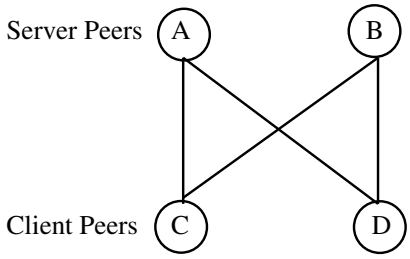

Figure 5.3: An example of performance loss using push-based method.

CS are misaligned. Based on our discussion so far, the performance loss are due to different shapes of cost functions and different types of costs modeled by these functions. In next subsection, we try to analyze the impacts of cost functions using the techniques borrowed from selfish routing [70].

5.4.3 General Prices and Performance Loss

In Section 5.4.1, we show the case that *BA* and *CS* achieve the social optimality, where the price function $p_{ij}(f_{ij}) = S'_{ij}(f_{ij})$. The optimal case implies that the prices correctly reflect the congestion at server peers seen by client peers, which are defined as Pigovian taxes [26]. As shown in above two examples, the performance loss might be arbitrarily large if prices incorrectly reflect Pigovian taxes. We try to establish bounds of performance loss for some special price functions in this subsection, before evaluating performance loss with general price functions by simulations.

Based on the social optimization problem (5.15) and Theorem 9, the video stream over P2P overlay networks can be considered as data flows sent among peers. Optimal *BA* and *CS* solutions aim to find out optimal data flows. In addition, the data flow can also be easily converted into *BA* and *CS* solutions, as shown in Section 5.4.1. From the flow perspective, we can consider analogues of the bounds of performance loss in selfish routing [70] and bounds of performance loss in P2P streaming systems, which might be ensured for special cases (i.e., special types of objective functions).

In this section, the cost function $S_{ij}(f_{ij}) = f_{ij}l_{ij}(f_{ij})$ with the price function $p_{ij}(f_{ij}) = l_{ij}(f_{ij})$, where $l_{ij}(f_{ij})$ is the latency function of overlay link (i, j). As defined in Sections 5.3.1 and 5.3.2, the objective functions of *BA* and *CS* are misaligned with this setting. In this case, *BA* allocates peer's upload bandwidth to minimize the total latency, $\sum_{i\in V}\sum_{j\in NBR(i)} f_{ij}l_{ij}(f_{ij})$. By contrast, each client peer selfishly schedules data requests based on the measured latency (or the historical record) without considering the impact on other peers. It implies that the *CS* problem minimizes $\sum_{j\in V}\sum_{i\in NBR(j)}\int_0^{f_{ij}} l_{ij}(y)dy$. In fact, this is a typical case studied in selfish routing and the bounds of performance loss are established in [70]. Our major contribution here is to identify the proper model and relate our model with selfish routing.

To obtain the bound of performance loss, we compare flows derived from the above setting with the socially optimal flow resulting from Section 5.4.1. Suppose the socially optimal problem (5.15) also minimizes the total latency $\sum_{i\in V}\sum_{j\in NBR(i)} f_{ij}l_{ij}(f_{ij})$. Let $\vec{f}^{OPT}$ represent the set of optimal flows, which can be realized by optimal bandwidth allocation patterns $\vec{\pi}^{OPT}$ and optimal content scheduling patterns $\vec{a}^{OPT}$. Therefore, the total latency is $\sum_{i\in V}\sum_{j\in NBR(i)} f_{ij}^{OPT}l_{ij}(f_{ij}^{OPT})$ and $f_{ij}^{OPT} = \pi_{ij}\sum_{j\in NBR(i)}\sum_{k\in K} a_{ij}^k$.

Compared with the socially optimal flow, there are two reference cases corresponding to the following two scenarios: 1) Client peers independently request substreams from best neighboring peers in terms of measured (or estimated) overlay link latency and server peers simply satisfy received requests (e.g., using FIFO or Round-Robin scheme to send requested data); and 2) Server peers have the global knowledge of both the types and amounts of contents requested by client peers and are able to make the best bandwidth allocation decisions $\vec{\pi}^{OPT}$ (e.g., by distributed cooperative message exchange or by centralized coordinations); whereas client peers make best responses to the bandwidth allocation, as defined in Section 5.3.4. The two scenar-

ios can represent practical protocols. For example, the min-cost block scheduling protocol [101] can be represented by the first scenario; and the second scenario is an idealized AQCS protocol [30], where peers are grouped into clusters and they cooperate with each other within a cluster and there are inter-cluster cooperations.

For the first scenario, the flows on overlay links are determined by client peers' requests, which can be model as the following optimization problem. The objective function is the same as the *CS* problem (5.3), $D_{ij}(f_{ij}(\vec{a})) = \int_0^f l_{ij}(y)dy$. Constraint (5.19) indicates the total flow through a server peer i cannot exceed its upload capacity and constraint (5.20) indicates that total amount of requested substream k should satisfy client peer j's demand of k. The solution to this problem is a set of equilibrium flows [26] and we use $\vec{f}^{EQ}$ to denote the solution. As discussed in Section 5.4.1, the bandwidth allocation pattern $\vec{\pi}^{EQ}$ and the content scheduling pattern $\vec{a}^{EQ}$ can be derived from the data flow.

$$\text{minimize} \sum_{\text{j}\in\text{V}} \sum_{\text{i}\in\text{NBR(j)}} \text{D}_{\text{ij}}(\text{f}_{\text{ij}}(\overrightarrow{a})) \tag{5.18}$$

Subject to:

$$\sum_{j\in NBR(i)} \sum_{k\in K} a_{ij}^k \leq C_i, \forall i \in V \tag{5.19}$$

$$\sum_{i\in NBR(j)} a_{ij}^k = d_j^k, \forall j \in V \tag{5.20}$$

$$a_{ij}^k \geq 0, \forall i, j \in V, \forall k \in K$$

We can derive the bound of cost resulting from the equilibrium flow $\vec{f}^{EQ}$, by relating it to selfish routing scenario. Server peers can be considered as physical links with capacity limits and client peers' requests can be considered as traffic to be routed through different physical links (i.e., requesting data from server peers). Therefore, the cost bound resulting from equilibrium flow has very similar mathematical proper-

ties to the selfish routing formulation. We can apply the techniques of analyzing the price of anarchy in selfish routing [70] to establish bounds of performance loss in our case. For example, if the link latency function, $l_{ij}(f_{ij})$, is affine, the total cost in equilibrium situation is no more than $\frac{4}{3}$ times of that in optimal situation. We can use the same techniques to relate the second scenario to routing scenarios. Once such relation is identified, the bounds of performance loss in P2P streaming system can be easily derived based on the well established mathematical analysis in the routing scenario. The streaming cost of some special functions are shown in Table 5.2 and we validate some bounds in simulations below, where $\bar{C}$ denotes average upload bandwidth and r_{max} denotes maximum achievable streaming rate. As an example, we show the proof of establishing the cost bound for linear functions, which has the form of $ax + b$. We use $COST(\vec{f}^{OPT}) = \sum_{i \in V} \sum_{j \in NBR(i)} f_{ij}^{OPT} l_{ij}(f_{ij}^{OPT})$ to denote the streaming cost resulted from optimal flow $\vec{f}^{OPT}$ and $COST(\vec{f}^{WE}) = \sum_{i \in V} \sum_{j \in NBR(i)} f_{ij}^{WE} l_{ij}(f_{ij}^{WE})$ to denote the streaming cost result flow at Nash Equilibrium defined in Section 5.3.4, where the overlay link cost function (e.g., peer perceived overlay link latency) $l_{ij} = af_{ij} + b$ and a, b are constants. Following the analysis procedure in [70], we can derive the following cost bound.

Theorem 10. *If peer perceived overlay link cost functions are linear, then* $\frac{COST(\vec{f}^{WE})}{COST(\vec{f}^{OPT})} \leq \frac{4}{3}$.

Proof: Since *BA* and *CS* determine video stream flows on overlay links, we consider P2P streaming as sending useful data flows from server peers to client peers. The game between *BA* and *CS* leads to an equilibrium as defined in Section 5.3.4. As stated in above theorem, $l_{ij} = af_{ij} + b$, the cost of per flow is $f_{ij}l_{ij} = af_{ij}^2 + bf_{ij}$ and the marginal cost is $2af_{ij} + b$. Based on [70], if $l_{ij}(f_{ij})$ is linear, then any feasible flow set $\vec{f}^{FEA}$ is optimal if and only if the flow set is at Nash equilibrium. With this statement, we can have the following property of flows at Nash equilibrium,

supposing that $l_{ij}(f_{ij})$ is linear and streaming rate is r: 1) the flow $\vec{f}^{WE}/2$ with streaming rate r is optimal for the P2P streaming system with streaming rate $r/2$; and 2) the marginal cost of increasing the flow at a server peer i with respect to $\vec{f}^{WE}/2$ equals the streaming cost of i with respect to $\vec{f}^{WE}$. The two results can be proved as follows.

With streaming rate r, a flow $\vec{f}$ is at equilibrium if and only if $\forall i \in V$, $af_{ij} + b \leq a\tilde{f}_{ij} + b$ and a flow $\vec{f}$ is optimal if and only if $\forall i \in V$, $2af_{ij} + b \leq 2a\tilde{f}_{ij} + b$, where $\tilde{f}_{ij}$ is an arbitrary flow through peer i to peer j. The latter means that the optimal flow should have the lowest marginal cost among all flows. To prove 1) above, we simply note that if f satisfies the equilibrium condition with streaming rate r, the flow $f/2$ satisfies the optimal condition with streaming rate $r/2$, since $\forall i \in V$, $af_{ij}+b \leq a\tilde{f}_{ij}+b$ implies $\forall i \in V$, $2a(f_{ij}/2) + b \leq 2a(\tilde{f}_{ij}/2) + b$. For 2), remember that the marginal cost per flow is $2af + b$ and thus the marginal cost of $f_{ij}/2$ for each peer equals to the streaming cost definition of peer i.

We can now prove Theorem 10 with two steps: in the first step, an optimal flow for streaming rate $r/2$ is supported by current P2P streaming system (which would be half of a Nash flow for streaming rate r based on 1)), and then we can augment the $f/2$ to f, which leads to the Nash flow for P2P streaming system with streaming r. Finally, we show that the cost of first half flow is at least $\frac{1}{4}COST(\vec{f}^{WE})$ and the second half flow cost is at least $\frac{1}{2}COST(\vec{f}^{WE})$. However, there is one problem of this two-step proof. That is, the augmentation in the second step may increase or decrease the amount of flow through any specific peer i. This problem has been solved by [70]. We apply similar technique below to prove that the cost of augmenting streaming rate r to $(1+\delta)r$ is at least $COST(\vec{f}^{OPT}) + \delta \sum_{\forall i \in V} \sum_{\forall j \in NBR(i)} \tilde{COST_{ij}}(f_{ij}^{OPT}) r_i$, where $\tilde{COST_{ij}}(f_{ij}^{OPT})$ denotes the minimum marginal cost of increasing flow between peer i and j with respect to flow f_{ij}^{OPT}).

For each overlay link, the cost function $l_{ij}f_{ij} = af_{ij}^2 + bf_{ij}$ is convex. Therefore, the inequality $l_{ij}(f_{ij})f_{ij} \geq l_{ij}(f_{ij}^{OPT})f_{ij}^{OPT} + (f_{ij} - f_{ij}^{OPT})\tilde{l}_{ij}(f_{ij}^{OPT})$, where f_{ij} is an arbitrary flow through link ij and $\tilde{l}_{ij}$ is the marginal cost of increasing flow on overlay link ij. With this inequality and $\tilde{COST}_{ij}$ is the minimum marginal cost, we can derive that the cost of sending flow with rate $(1+\delta)r$ satisfies the following inequality. $COST(\vec{f}) \geq \sum_{\forall i \in V}\sum_{\forall j \in NBR(i)} l_{ij}(f_{ij}^{OPT})f_{ij}^{OPT} + \sum_{\forall i \in V}\sum_{\forall j \in NBR(i)}(f_{ij} - f_{ij}^{OPT})\tilde{l}_{ij}(f_{ij}^{OPT}) \geq COST(\vec{f}^{OPT}) + \delta\sum_{\forall i \in V}\sum_{\forall j \in NBR(i)} \tilde{COST}_{ij}(f_{ij}^{OPT})r_i$. Finally, we can apply the above inequality and observations 1) and 2), we can show that $COST(\vec{f}) \geq \frac{3}{4}COST(\vec{f}^{WE})$, where $\vec{f}$ is arbitrary flow. Therefore, the cost of optimal flow with streaming rate r is at least $\frac{3}{4}$ times of flow at equilibrium.

□

Table 5.2: Bound of Content Scheduling Cost

Description	Typical Formula	Cost Bound
Linear	$ax + b$	$\frac{4}{3}$
Quadratic	$ax^2 + bx + c$	$\frac{3\sqrt{3}}{3\sqrt{3}-2}$
Cubic	$ax^3 + bx^2 + cx + d$	$\frac{4\sqrt[3]{4}}{4\sqrt[3]{4}-3}$
M/M/1 Delay	$\frac{1}{C-x}$	$\frac{1}{2}(1 + \sqrt{\frac{\bar{C}}{\bar{C}-r_{max}}})$

5.5 Simulations

In this section, we design and implement three groups of bandwidth allocation and block scheduling algorithms in a packet-level simulator [102]. As mentioned in Section 5.1, when the video stream is divided into discrete chunks, we refer to the process of requesting useful data blocks from neighboring peers as block scheduling instead of the general content scheduling. The terminology, block scheduling, is used herein after. We demonstrate the impact of cost functions in solving bandwidth allocation and block scheduling problems on system performance and how to design new

protocols with our established models. Complementary to the theoretical analysis, the simulation results allow us to obtain better understanding of the interaction between bandwidth allocation and block scheduling at the system level. Moreover, the simulation results and the theoretical analysis also provide guidelines for engineers who need to design protocols for applications with specific requirements to achieve optimal system performance (i.e., overall user perceived streaming quality). In the following subsections, we first describe the simulation setup and protocol implementation. Then, we present simulation results and related discussions.

5.5.1 Simulation Setup

We implement the following three groups of bandwidth allocation and block scheduling algorithms in C++ using an open-source, event-driven, packet-level simulator [102], where all streaming and control packets and node buffers are carefully simulated. It implements full functionalities of a P2P streaming system (e.g., overlay construction, data delivery, etc.) and provides reliable results for P2P streaming quality evaluation (e.g., VUD [92] uses this simulator for performance evaluation.).

5.5.1.1 OPT

Both the bandwidth allocation and block scheduling algorithms have exactly the same objective functions, where client peers periodically pull data with the goal of minimizing the streaming cost and server peers simply satisfy their requests if the requested blocks are available. The shape of the cost function is shown in Figure 5.4, which is used in [101] to design the min-cost block scheduling algorithm and obtained from the algorithm implementation [102]. We use OPT to refer to this case.

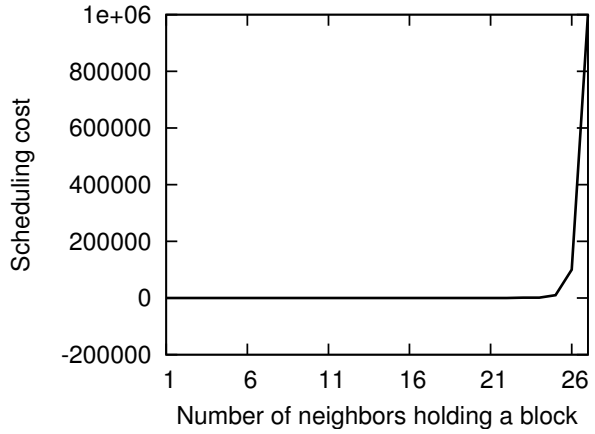

Figure 5.4: Cost function of block scheduling, which is used by OPT, MIN+PL and MIN+QUEUE. Note that the smallest cost is 1.

5.5.1.2 MIN+PL

The bandwidth allocation and block scheduling algorithms have different objective functions, where client peers periodically pull data with the goal of minimizing the streaming cost and server peers satisfy their requests with the goal of minimizing congestion. We use MIN+PL to refer to this case. The cost function of block scheduling is the same as OPT. The cost function of bandwidth allocation is shown in Figure 5.5, which is used to model the congestion cost in [38] and is given below:

$$S_i(f_i) = \begin{cases} f_i, & 0 \le f_i/C_i < 1/3 \\ 3f_i - 2/3C_i, & 1/3 \le f_i/C_i < 2/3 \\ 10f_i - 16/3C_i, & 2/3 \le f_i/C_i < 9/10 \\ 70f_i - 178/3C_i, & 9/10 \le f_i/C_i < 1 \\ 500f_i - 1468/3C_i, & 1 \le f_i/C_i < 11/10 \\ 5000f_i - 16318/3C_i, & 11/10 \le f_i/C_i < \infty \end{cases}$$

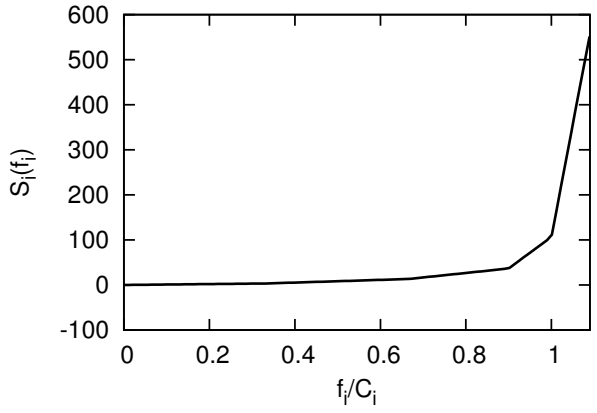

Figure 5.5: Piece-linear cost function of bandwidth allocation with $C_i = 10$, which is used by MIN+PL.

5.5.1.3 MIN+QUEUE

The bandwidth allocation and block scheduling algorithms have different objective functions, where client peers and server peers follow the same manner as MIN+PL to request and send data. We use MIN+QUEUE to refer to this case. The block scheduling cost function of MIN+QUEUE is the same as OPT and MIN+PL. The bandwidth allocation cost function of MIN+QUEUE is shown in Figure 5.6, which approximates the M/M/1 queueing delay and is given below:

$$S_i(f_i) = \frac{1}{C_i - f_i}, \quad f_i < C_i$$

As shown in Figures 5.4, 5.5 and 5.6, we intentionally choose the cost functions of bandwidth allocation and block scheduling to be very similar shapes. This allows us to show that even if the cost functions of MIN+PL and MIN+QUEUE are relatively well aligned, there are still performance gaps, compared with OPT. There are additional explanations of above cost functions: 1) The y-axis range of cost function in Figure 5.4 is obtained from algorithm implementation in [102] and the ranges of cost functions in Figure 5.5 and 5.6 are determined by peer i's upload bandwidth; 2) The

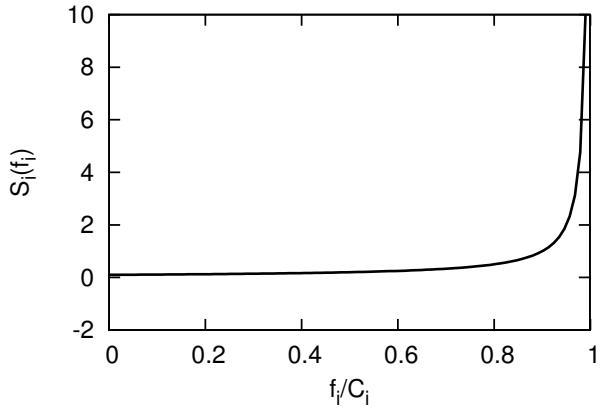

Figure 5.6: Queueing delay cost function of bandwidth allocation with $C_i = 10$, which is used by MIN+QUEUE.

range differences between the block scheduling cost function and the two bandwidth allocation cost functions do not influence the allocation results, in that block scheduling and bandwidth allocation are carried out independently; and 3) The function values of all these functions are *strictly greater* than 0.

To design scalable protocols, for all three cases, each peer solves local cost minimization problems and there is no centralized scheme to coordinate decisions among peers. Specifically, each peer maintains records of data sending rates and receiving rates of direct neighbors as estimates of their data demands and supplies, which can be converted to constraints of local *BA* and *CS* problems. In order to solve the local optimization problems, we integrate the NOMAD nonlinear optimization library into the simulator [5], which is open source and implemented in C++. It is a derivative-free solver for mixed variable nonlinear optimization and has superior convergence properties, which only requires the smoothness of objective functions. For a bandwidth allocation/block scheduling with 30 variables and 30 constraints, it converges within tens of milliseconds.

The underlying delay matrix is set the same as [102], which is a real-world node-

to-node latency matrix (2500 × 2500) with 79 ms average end-to-end delay [4]. In simulation results shown in this chapter, the raw streaming rate is 300Kbps (i.e., the 40-byte/packet header overhead is not included) and the maximum number of neighbors is 30 and the request window size is 20 seconds. In addition, the streaming server upload capacity is set to 600Kbps. To simulate bandwidth heterogeneity, there are three different types of peers, whose upload capacities are 1Mbps, 384Kbps and 128Kbps, respectively and download capacities are 3Mbps, 1.5Mbps, 768Kbps, respectively. We use the resource index [40] [102] to control the system upload bandwidth supplies via changing the fractions of the three types of peers. The resource index is defined as the ratio of the total upload bandwidth supply to the raw streaming rate (default value 300kbps) times the number of peers in the whole system, i.e., the ratio of bandwidth supply to the minimum bandwidth demand, which is a necessary condition for providing system-wide optimal streaming quality [40]. Peers request data blocks every 500 ms and allocate bandwidth every 5 seconds.

5.5.2 Simulation Results and Discussions

In this section, we show the simulation results with different number of peers and resource indices. We use the following metrics to evaluate the system performance: 1) Delivery ratio of a peer is defined as the number of received useful streaming blocks in a period over the number of packets that should be received in the same period; 2) Upload bandwidth utilization ratio of a peer is defined as the actually used upload bandwidth over the upload capacity of a peer at a monitoring time point; and 3) Block request cost of a peer is defined as the total cost of requested blocks during each block request interval, with the cost function shown in Figure 5.4. Lower cost implies better streaming quality. Note that we also collect results with other metrics of streaming quality (e.g., playback delay, control overhead and average packet delay,

etc.), which show similar trends as delivery ratio. Therefore, we omit these results in this chapter. In addition, each data point in the following figures is an average value of all online peers and is calculated every 10 seconds.

5.5.2.1 Simulation results

We first simulate a system with 500 peers, which join the system within 10 seconds after the beginning of simulation and stay in the network until the end of the simulation. Figures 5.7, 5.8 and 5.9 show results of OPT, MIN+PL and MIN+QUEUE with resource index 1.0. Since the resource index does not consider the control packets (e.g., buffermap exchange and block request packets), resource index 1.0 is insufficient to sustain the optimal streaming quality (i.e., delivery ratio is close to 1). Therefore, as shown in Figure 5.7, the delivery ratios of all three protocols are below 1, but OPT still maintains the highest delivery ratio among the three. From Figure 5.8, we can see that OPT fully utilizes peers' upload bandwidth (i.e., the utilization ratio is 1 almost all the time). By contrast, MIN+PL and MIN+QUEUE only have upload bandwidth utilization ratios above 0.8, even though the resource in insufficient for the whole system and the cost function shapes are very similar to the block scheduling cost function shape. Figure 5.9 shows that all the three protocols have similar average block request costs and even OPT suffers high scheduling cost, because total upload bandwidth is insufficient.

Figures 5.10, 5.11 and 5.12 show the results of a system with 500 peers and resource index 1.2. Compared with Figure 5.7, Figure 5.10 shows that OPT can achieve optimal delivery ratio, since there is sufficient upload bandwidth. However, the streaming quality of MIN+PL and MIN+QUEUE improves a little with sufficient upload bandwidth, in that block scheduling and bandwidth allocation have different objectives and the situations of counter examples shown in Sections 5.4.2.1

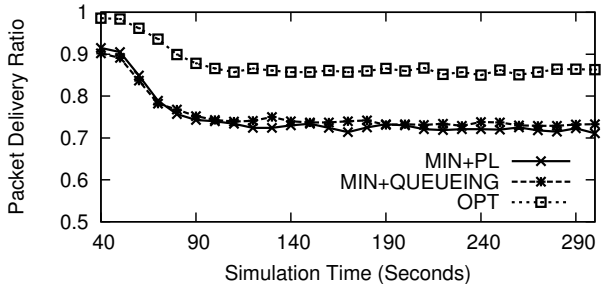

Figure 5.7: Average delivery ratio of a P2P streaming system with 500 peers and resource index 1.0.

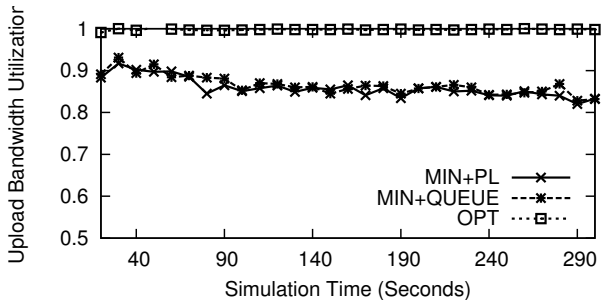

Figure 5.8: Average upload bandwidth utilization ratio of a P2P streaming system with 500 peers and resource index 1.0.

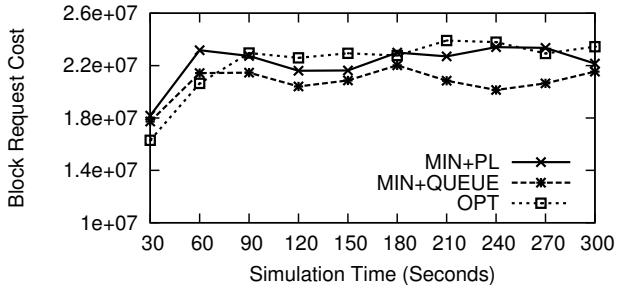

Figure 5.9: Average block request cost of a P2P streaming system with 500 peers and resource index 1.0.

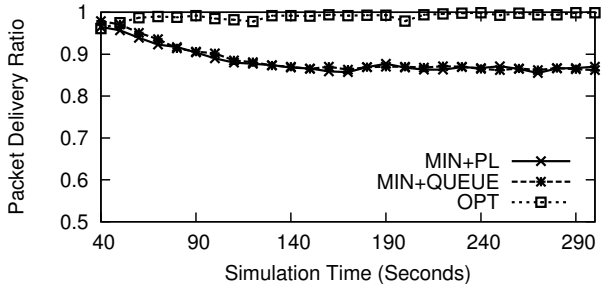

Figure 5.10: Average delivery ratio of a P2P streaming system with 500 peers and resource index 1.2.

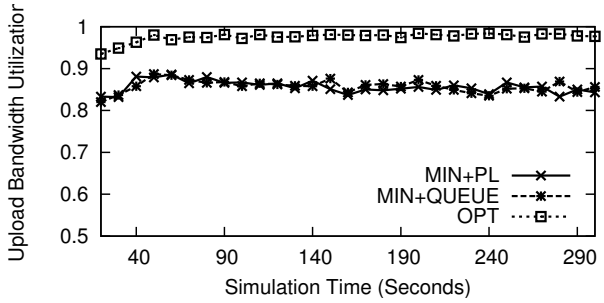

Figure 5.11: Average upload bandwidth utilization ratio of a P2P streaming system with 500 peers and resource index 1.2.

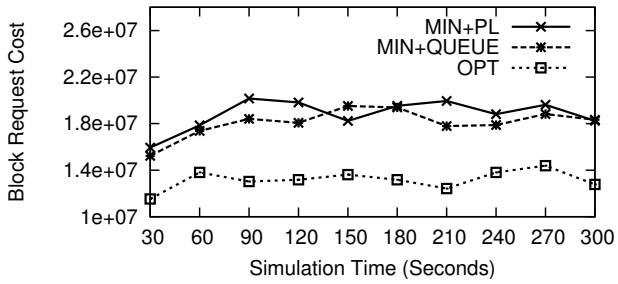

Figure 5.12: Average block request cost of a P2P streaming system with 500 peers and resource index 1.2.

and 5.4.2.2 occur frequently. The low upload bandwidth utilization ratios of MIN+PL and MIN+QUEUE in Figure 5.11 also confirm the former explanation. OPT's upload bandwidth utilization ratio decreases slightly (from 1 to about 0.98), because the control overhead is less than 0.1 of total upload bandwidth and the resource is more than enough. From Figure 5.12, we can see that OPT leads to lower block scheduling costs than MIN+PL and MIN+QUEUE, due to the sufficient bandwidth.

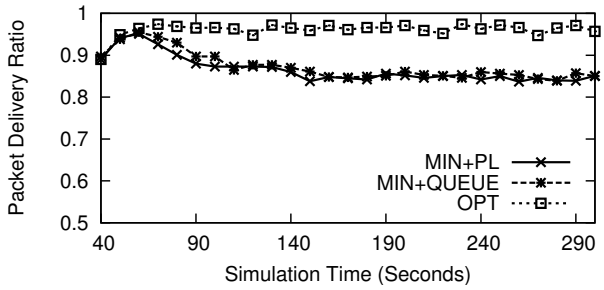

Figure 5.13: Average delivery ratio of a P2P streaming system with 5000 peers and resource index 1.2.

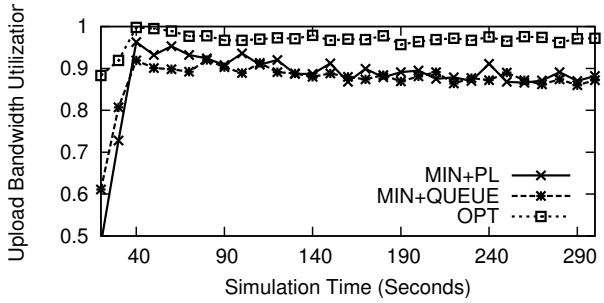

Figure 5.14: Average upload bandwidth utilization ratio of a P2P streaming system with 5000 peers and resource index 1.2.

We increase the number of peers from 500 to 5000 to determine whether the above trends are related to the system size. From Figures 5.13, 5.14 and 5.15, we can see that OPT still maintains near optimal delivery ratio, upload bandwidth utilization ratio and lower block request cost; while MIN+PL and MIN+QUEUE have similar results with the above smaller system. Note that the delivery ratio of OPT is slightly lower than 1, which is caused by other factors (such as exchange window size, data

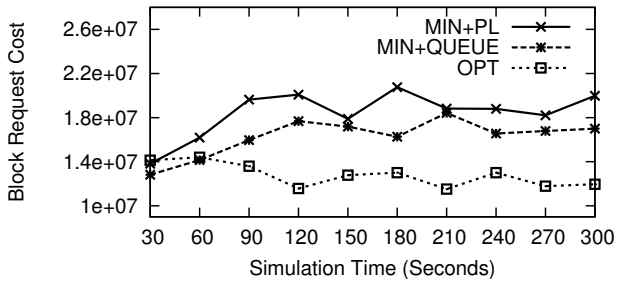

Figure 5.15: Average block request cost of a P2P streaming system with 5000 peers and resource index 1.2.

request interval, etc.) and is very similar to the optimal results of the same setting in [102].

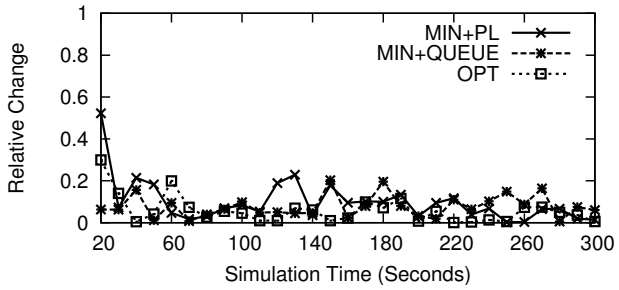

Figure 5.16: Scheduling cost changes of a P2P streaming system with 500 peers and resource index 1.0.

Furthermore, we study the convergence of block scheduling algorithms by calculating the relative changes of data request cost, which is defined as $|Cost_N - Cost_{N-1}|/Cost_{N-1}$. Figures 5.16, 5.17 and 5.18 show the changes and the changes fall below 10%, which shows good convergence property with the above simulation settings. Since convergence might be influenced by many factors (e.g., the frequency of running bandwidth allocation algorithms), we plan to investigate this further in future work.

Finally, we compare the bound obtained in Table 5.2 with our simulation results of 500 and 5,000 peers with resource index 1.2. The y-axis of Figure 5.19 is a ratio, defined as the average block scheduling cost of MIN+QUEUE over the average cost

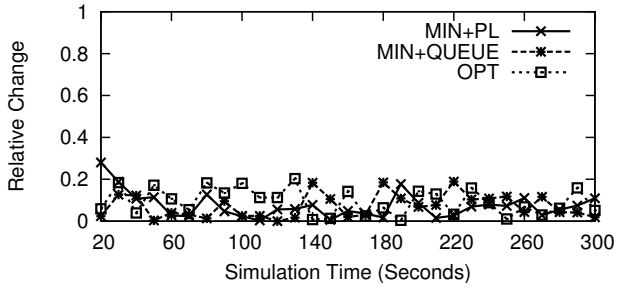

Figure 5.17: Scheduling cost changes of a P2P streaming system with 500 peers and resource index 1.2.

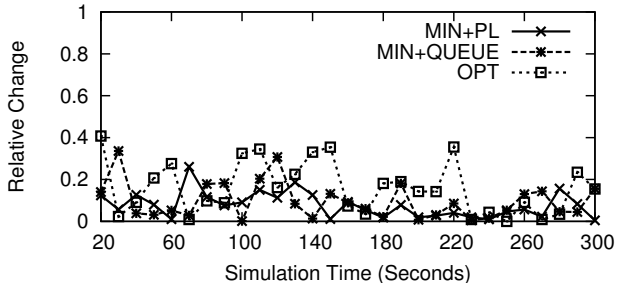

Figure 5.18: Scheduling cost changes of a P2P streaming system with 5000 peers and resource index 1.2.

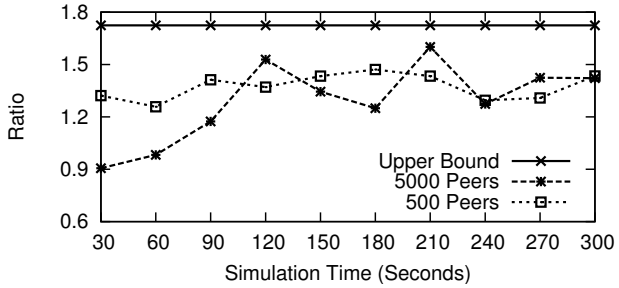

Figure 5.19: Simulation cost ratio vs Theoretical Bound.

of OPT. Based on simulation results, the maximum achievable streaming rate is close to 300Kbps. Based on definition of resource index, the average upload bandwidth is $1.2 * 300$Kbps. With the formula $\frac{1}{2}(1 + \sqrt{\frac{\bar{C}}{\bar{C}-r_{max}}})$, we obtain the upper bound ratio is 1.7247, which means that cost of MIN+QUEUE should not be higher than 1.7247 times OPT scheduling cost, as shown in Figure 5.19.

5.5.2.2 Discussion

Based on our packet-level simulations, we have the following observations: 1) When implementing bandwidth allocation and block scheduling algorithms, solving the corresponding optimization problems locally is usually a good approximation to solving them globally; 2) When the objective functions of bandwidth allocation and block scheduling are aligned, the system-wide performance is near-optimal; The near-optimal performance is due to the fact that the streaming quality might be slightly influenced by other parameters of a P2P streaming system [102] (e.g., the buffermap exchange interval), though the system-wide performance is theoretically optimal; 3) Even if the objective functions of solving the two problems are very similar, small differences can lead to large performance losses; and 4) In the case of misaligned objective functions, we need methods of coordinating bandwidth allocation and block scheduling. For example, Nash Bargaining Solution [61] is a good option. The study in [38] even provides distributed algorithms for Nash Bargaining in ISP networks, which are proved to achieve Pareto optimality. Another possible solution is to apply the multi-objective optimization, e.g., NOMAD [5] provides a bi-objective optimization solver. Intuitively, the implementation of such solutions will increase the control overhead and there should be a trade-off between control overhead and accuracy.

5.6 Chapter Summary

In this chapter, we study the problem of providing system-wide optimal quality of service in P2P streaming systems via properly controlling a peer's bandwidth allocation *BA* and content scheduling *CS*. Specifically, we propose generic nonlinear optimization models for solving bandwidth allocation and content scheduling problems. Then, we analyze the interaction between *BA* and *CS* with a two-player game theoretic model, which shows that the system-wide suboptimal performance is mainly due to the misaligned objectives of *BA* and *CS*. To validate our analysis, we design and implement three groups of protocols for solving the two problems using an event-driven, packet-level simulator. Based on the simulation results, even if the objective functions have very similar shapes, the system-wide streaming quality is still suboptimal and joint design (e.g., Nash Bargaining) should be used for achieving system-wide optimal streaming quality.

For future work, our models and analysis can be extended along two directions: 1) Analyzing the performance loss considering overlays spanning multiple ISPs; and 2) Analyzing the performance loss in multiple co-existing overlays. Both directions will add new constraints to the optimization models and introduce multiple parties into the game theoretic model. Furthermore, theoretically analyzing how system dynamics impact the performance loss is even more challenging.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we investigated fundamental issues in building multi-channel P2P streaming systems from a resource allocation perspective. To provide high streaming quality, a multi-channel P2P streaming system should optimally utilize peers' resources (e.g., upload bandwidth, low latency overlay links, etc.) and thus optimal resource allocation algorithms should be developed. Instead of seeking algorithms to specific applications, we studied a wide range of designs to provide design guidelines for P2P streaming systems, based on optimization and queueing models. Our work includes both theoretical analysis and simulations, which aims at solving the key resource allocation problems, namely bandwidth allocation and block scheduling.

When designing protocols for multi-channel P2P streaming systems, we should consider several important factors, including but not limited to efficiency, implementation complexity and control overhead. To choose a proper design, engineers must consider trade-offs among these factors, based on specific application scenarios. Our work mainly focused on establishing generic theoretical models for efficiently utilizing

peers' resources to provide optimal streaming quality. We further investigated implementation complexity and control overhead with packet-level simulations, where protocols for solving bandwidth allocation and block scheduling have been carefully implemented and can be directly migrated to practical P2P streaming systems. However, there is one weakness of our simulation studies. Only a small set of our traces is from real systems, since conducting Internet-scale measurements is very challenging and time-consuming. Therefore, we generated as many cases as possible to evaluate our proposed protocols.

Since bandwidth is the most precious resource in P2P streaming systems, a critical problem is to encourage peers to contribute their bandwidth as much as possible, which leads to the design efficient strategies for providing incentives to peers. This problem, which has attracted researchers from mathematics, economics and operations research, is beyond the scope of this dissertation. To sum up, our contributions addressed how to optimally utilize existing peers' resources instead of encouraging peers to contribute more resources.

6.2 Future Work

We anticipate future work in this topic to proceed in two directions: 1) There is scope for improving the performance of multi-channel P2P streaming systems; and 2) P2P streaming technology can be applied in other time-sensitive systems. We describe the two directions in detail.

The participants of P2P streaming system are Internet users and their behavior might greatly influence the streaming quality. For example, users have different interests and viewing habits (e.g., some users prefer to watch movies as opposed to other kinds of videos; many users watch videos at 9:00 PM; etc.). We can utilize these

facts to design systems for providing high quality of service. Recently, Niu *et al.* [60] propose a self-diagnostic P2P streaming system based on a machine learning framework, where the system estimates performance with historical data and the system resources can be more efficiently used. However, to fully understand user's behavior and interests is very challenging, because we need to retrieve such information from a huge amount of data, which requires cross-disciplinary expertise. Moreover, the social relationship among peers can also be utilized to design new generation P2P streaming systems, in terms of better performance and privacy protection. Isdal *et al.* [35] build a privacy-preserving P2P streaming system based on the trust among friends in social networks. Besides preserving privacy, other topics can be pursuied in the future, such as encouraging cooperations among friends, etc. Note that all future designs should consider the multi-channel P2P streaming systems, in that there are tens of thousands of videos for users to watch and all existing commercial systems support hundreds of channels.

The other direction of future work is to extend P2P streaming technology in other emerging Internet-scale technologies. Since the key design rationale of P2P streaming is to utilize participating peers' resources (e.g, upload bandwidth) for delay sensitive video delivery, the theoretical analysis and implementations can be extended to cloud computing, data center and large-scale network storage systems. There are some researches and prototypes for building peer-assisted Content Delivery Networks (CDN) [97] [34], which require peers to cache some contents for CDN servers. Liu *et al.* [48] propose a P2P storage cloud to provide high-definition video streaming, which constructs large-scale storage cloud purely based on P2P network and further extends the idea of building peer-assisted systems. These successful examples are only initial steps of extending P2P streaming to other large-scale networked systems. In data center networks, there are more resource allocation problems among servers

(e.g., power, bandwidth etc.), where we can find counterparts in P2P streaming.

P2P streaming is an amazing technology developed in the past few years, which has quickly attracted millions of users and has proved successful in sustaining high-definition video streaming with limited server bandwidth. Many new Internet applications and services (e.g., social network and service cloud) have similar characteristics as P2P streaming systems, such as large scale and cooperations among participants. Therefore, there are tremendous opportunities for both improving P2P streaming systems using these new applications and extending P2P streaming technologies for these new systems.

Bibliography

[1] Youtube: http://www.youtube.com.

[2] http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Online_Video_Rankings.

[3] ATT: http://www.att.com/u-verse.

[4] A Lightweight Approach to Network Positioning: http://www.cs.cornell.edu/People/egs/meridian/.

[5] M. A. Abramson, C. Audet, G. Couture, Jr. J. E. Dennis, and S. Le Digabel. http://www.gerad.ca/NOMAD/Project/Home.html.

[6] M. Ahmed. Call admission control in wireless networks: A comprehensive survey. *IEEE Communications Surveys and Tutorials 7*, 7(1):50–69, 2005.

[7] M. Beckmann, C.B. McGuire, and C.B. Winsten. *Studies in the economics of transportation*. Yale University Press for the Cowles Commission for Research in Economics, New Haven, 1959.

[8] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, February 1997.

[9] Beta-Distribution. http://mathworld.wolfram.com/BetaDistribution.html.

[10] R. Bindal and P. Cao. Can self-organizing P2P file distribution provide QoS guarantees. *ACM Operating Systems Review*, 40(3):22–30, July 2006.

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

[12] T. Bonald, M. Jonckheere, and A. Proutiére. Insensitive load balancing. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '04/Performance '04, pages 367–377, 2004.

[13] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. *SIGMETRICS Perform. Eval. Rev.*, 36(1):325–336, 2008.

[14] T. Bonald and A. Proutière. Insensitivity in processor-sharing networks. *Perform. Eval.*, 49(1/4):193–209, 2002.

[15] T. Bonald and A. Proutière. Insensitive bandwidth sharing in data networks. *Queueing Syst.*, 44(1):69–100, 2003.

[16] R. Braden, D. Clark, and S Shenker. Integrated services in the Internet architecture: an overview. *RFC1633*, June 1994.

[17] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in a cooperative environment. In *Proceedings of ACM SOSP*, Lake Bolton, NY, October 2003.

[18] M. Chen, S. Liu, S. Sengputa, M. Chiang, J. Li, and P. A. Chou. P2P streaming capacity under node degree bound. *Technical Report, Princeton University*, August 2009.

[19] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. In *in Proceedings of ACM Sigmetrics*, pages 1–12, 2000.

[20] B. Cohen. Bittorrent. `http://www.bittorrent.com/`.

[21] CPLEX. `http://www.ilog.com/products/cplex/`.

[22] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Qos's downfall: at the bottom, or not at all! In *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?*, RIPQoS '03, pages 109–114, 2003.

[23] T. Dan-Cristian and M. Laurent. Flow control for cost-efficient peer-to-peer streaming. In *Proceedings of IEEE INFOCOM*, San Diego, CA, March 2010.

[24] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, 1987.

[25] C. Diot, B. Neil, L. Bryan, and K. D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14:78–88, 2000.

[26] D. DiPalantino and R. Johari. Traffic engineering vs. content distribution: A game theoretic perspective. In *Proceedings of INFOCOM*, Rio de Janeiro, Brazil, 2009.

[27] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.

[28] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111, 2005.

[29] GridMedia. `http://www.gridmedia.com.cn/`.

[30] Y. Guo, C. Liang, and Y. Liu. Adaptive queue-based chunk scheduling for P2P live streaming. In *Proceedings of IFIP Networking*, 2008.

[31] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[32] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[33] X. Hei, Y. Liu, and K. Ross. Inferring network-wide quality in P2P Live streaming systems. *IEEE Journal on Selected Areas in Communications*, 25(9):1640–1654, December 2007.

[34] S. Ioannidis and P. Marbach. On the design of hybrid peer-to-peer systems. *Proceedings of SIGMETRICS*, 36(1):157–168, 2008.

[35] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with oneswarm. *Proceedings of SIGCOMM*, August 2010.

[36] S. Jamin, P. B. Danzig, S. J. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated service packet networks. *IEEE/ACM Trans. Netw.*, 5:56–70, February 1997.

[37] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, James W. O'Toole, Jr., M. Frans, and K. James. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th conference on Symposium on Operating System Design and Implementation*, pages 197–212, 2000.

[38] W. Jiang, R. Zhang-shen, J. Rexford, and M. Chiang. Cooperative content distribution and traffic engineering in an ISP network. In *Proceedings of Sigmetrics*, Seattle, WA, 2009.

[39] F. P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, pages 237–252, 1998.

[40] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for P2P streaming systems. In *Proceedings of IEEE INFOCOM*, pages 919–927, Anchorage, AK, May 2007.

[41] H. Kung and C. Wu. Differentiated admission for peer-to-peer systems: Incentivizing peers to contribute their resources. In *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, Berkely, CA, June 2003.

[42] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the new Coolstreaming: principles, measurements and performance implications. In *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.

[43] C. Liang, Y. Guo, and Y. Liu. Is random scheduling sufficient in P2P video streaming? In *Proceedings of ICDCS*, Beijing, China, June 2008.

[44] J. Liang and K. Nahrstedt. RandPeer: membership management for QoS sensitive peer-to-peer applications. In *Proceedings of IEEE INFOCOM*, pages 1–10, Spain, April 2006.

[45] J. Liang, B. Yu, Z. Yang, and K. Nahrstedt. A framework for future Internet-based TV broadcasting. In *Proceedings of IPTV Workshop*, Edinburgh, Scotland, May 2006.

[46] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng. AnySee: Peer-to-peer live streaming. In *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[47] F. Liu, B. Li, L. Zhong, and B. Li. Understanding the flash crowd in P2P live video streaming systems. In *Packet Video Workshop, 2009.*, pages 1–10, June 2009.

[48] F. Liu, S. Shen, B. Li, B. Li, H. Yin, and S. Li. Novasky: Cinematic-quality vod in a p2p storage cloud. *Proceedings of IEEE INFOCOM*, 2011.

[49] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance bounds for peer-assisted live streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):313–324, 2008.

[50] Y. Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 127–136, New York, NY, USA, 2007. ACM.

[51] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Journal of Peer-to-Peer Networking and Applications*, 1(1):18–28, March 2008.

[52] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the interaction between overlay routing and traffic engineering. In *in Proceedings of IEEE INFOCOM*, 2005.

[53] Z. Liu, C. Wu, B. Li, and S. Zhao. UUSee: Large-scale operational on-demand streaming with random network coding. In *Proceedings of IEEE INFOCOM*, 2010.

[54] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE Transactions on Networking*, 7(6):861–875, December 1999.

[55] N. Magharei and R. Rejaie. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *Proceedings of IEEE INFOCOM*, pages 1424–1432, 2007.

[56] N. Magharei and R. Rejaie. PRIME: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[57] L. Massoulie, E. Merrer, A. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks : Random walk methods. In *Proceedings of ACM PODC*, pages 123–132, Denver, Colorado, July 2006.

[58] L. Massoulié, A. Twigg, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. In *Proceedings of IEEE INFOCOM*, pages 1073–1081, 2007.

[59] J. F. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences of the United States of America*, 1950.

[60] D. Niu, B. Li, and S. Zhao. Self-diagnostic peer-assisted video streaming through a learning framework. In *Proceedings of ACM the international conference on Multimedia*, MM '10, pages 73–82, 2010.

[61] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1999.

[62] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, A. E. Mohr, and E. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of IPTPS*, pages 127–140, 2005.

[63] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal of Selected Aresa in Communications*, 24:1439–1451, 2006.

[64] D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Transactions on Automatic Control*, 52(12):2254–2269, December 2007.

[65] H. Perros and K Elsayed. Call admission control schemes: a review. *IEEE Communications Magazine*, 34(11):82–91, November 1996.

[66] PPLive. `http://www.pplive.com`.

[67] PPStream. `http://www.ppstream.com`.

[68] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu. Modeling channel popularity dynamics in a large IPTV system. In *Proceedings of SIGMETRICS '09*, pages 275–286, 2009.

[69] A. Raghuveer, Y. Dong, and D. Du. On providing reliability guarantees in live video streaming with collaborative clients. In *Proceedings of Multimedia Computing and Networking Conference (MMCN)*, San Jose, CA, January 2007.

[70] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of The ACM*, 49(2):236–259, 2002.

[71] S. Saroiu, K. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN) 2002*, San Jose, CA, USA, January 2002.

[72] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communication*, 13(7), September 1995.

[73] H. Shojania and B. Li. Nuclei: GPU-accelerated many-core network coding. In *Proceedings of IEEE INFOCOM*, pages 459–467, Rio de Janeiro, Brazil, April 2009.

[74] J. Siwko and I Rubin. Call admission control for capacity-varying networks. *Telecommnuication Systems*, 16(1):15–40, 2001.

[75] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proceedings of ACM IMC*, pages 377–390, Rio de Janeiro, Brazi, October 2006.

[76] D. A. Tran, K. A. Hua, and T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, March 2003.

[77] UUSee. `http://www.uusee.com`.

[78] F. Wang, J. Liu, and Y. Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of IEEE INFOCOM*, pages 1364–1372, Phoenix, AZ, April 2008.

[79] M. Wang and B. Li. Lava: A reality check of network coding in peer-to-peer live streaming. In *Proceedings of IEEE INFOCOM*, pages 1082–1090, Anchorage, AK, May 2007.

[80] M. Wang, L. Xu, and B. Ramamurthy. Channel-aware peer selection in multi-view peer-to-peer multimedia streaming. In *Proceedings of IEEE International Workshop on IP Multimedia Communications (IPMC) at ICCCN*, pages 1–6, 2008.

[81] M. Wang, L. Xu, and B. Ramamurthy. A flexible divide-and-conquer protocol for multi-view peer-to-peer live streaming. In *Proceedings of IEEE P2P*, pages 291–300, 2009.

[82] M. Wang, L. Xu, and B. Ramamurthy. Providing statistically guaranteed streaming quality for peer-to-peer live streaming. In *The 19th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 127–132, 2009.

[83] M. Wang, L. Xu, and B. Ramamurthy. Comparing multi-channel peer-to-peer video streaming system designs. In *Proceedings of IEEE LANMAN*, Long Branch, NJ, 2010.

[84] M. Wang, L. Xu, and B. Ramamurthy. Comparing multi-channel peer-to-peer video streaming system designs. *Technical Report, University of Nebraska-Lincoln*, November 2010.

[85] M. Wang, L. Xu, and B. Ramamurthy. Improving multi-view peer-to-peer live streaming systems with the divide-and-conquer strategy. *Technical Report, University of Nebraska-Lincoln*, August 2010.

[86] M. Wang, L. Xu, and B. Ramamurthy. Linear programming models for multi-channel P2P streaming systems. In *Proceedings of IEEE INFOCOM Mini-Conference*, 2010.

[87] M. Wang, L. Xu, and B. Ramamurthy. On providing optimal quality of service in p2p on providing optimal quality of service in p2p streaming systems. *The 31st International Conference on Distributed Computing Systems (Submitted)*, 2011.

[88] C. Wu and B. Li. Strategies of conflict in coexisting streaming overlays. In *Proceedings of IEEE INFOCOM*, pages 481–489, Anchorage, AK, May 2007.

[89] C. Wu, B. Li, and Z. Li. Dynamic bandwidth auctions in multi-overlay P2P streaming with network coding. *IEEE Transactions on Parallel and Distributed Systems*, 19(6):806–820, June 2008.

[90] C. Wu, B. Li, and S Zhao. Characterizing peer-to-peer streaming flows. *IEEE Journal on Selected Areas in Communications*, 25(9):1612–1626, December 2007.

[91] C. Wu, B. Li, and S. Zhao. Multi-channel live P2P streaming: Refocusing on servers. In *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.

[92] D. Wu, C. Liang, Y. Liu, and K. W. Ross. View-upload decoupling: A redesign of multi-channel p2p video systems. In *Proceedings of IEEE INFOCOM, Mini-Conference*, pages 1–6, Rio de Janeiro, Brazil, 2009.

[93] D. Wu, Y. Liu, and K. W. Ross. Queuing network models for multi-channel P2P live streaming systems. In *Proceedings of IEEE INFOCOM*, pages 73–81, Rio de Janeiro, Brazil, 2009.

[94] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *Proceedings of IEEE International Conferences on Distributed Computing Systems (ICDCS)*, Austria, July 2002.

[95] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nahrstedt, S. Jung, and R. Bajscy. TEEVE: the next generation architecture for tele-immersive environments. In *Proceedings of International Symposium on Multimedia*, Irvine, CA, December 2005.

[96] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. Viewcast: View dissemination and management for multi-party 3D tele-immersive environments. In *Proceedings of ACM Multimedia*, pages 882–891, Germany, September 2007.

[97] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with livesky. In *Proceedings of ACM International Conference on Multimedia*, pages 25–34, New York, USA, 2009.

[98] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. *SIGOPS Oper. Syst. Rev.*, 40(4):333–344, 2006.

[99] M. Zhang. `http://media.cs.tsinghua.edu.cn/~zhangm/`.

[100] M. Zhang, Y. Xiong, Q. Zhang, and S. Yang. On the optimal scheduling for media streaming in data-driven overlay networks. In *Proceedings of IEEE GLOBECOM*, pages 1–5, San Francisco, CA, 2006.

[101] M. Zhang, Y. Xiong, Q. Zhang, and S. Yang. Optimizing the throughput of data-driven peer-to-peer streaming. *IEEE Transactions on Parallel and Distributed Systems*, 20(1), 2009.

[102] M. Zhang, Q. Zhang, and S. Yang. Understanding the power of pull-based streaming protocol: Can we do better? *IEEE Journal on Selected Areas in Communications*, 25(8):1678–1694, 2007.

[103] X. Zhang and B. Li. On the market power of network coding in P2P content distribution systems. In *Proceedings of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[104] X. Zhang, J. Liu, B. Li, and T. Yum. DONet/CoolStreaming: A data-driven overlay network for live media streaming. In *Proceedings of IEEE INFOCOM*, Miami, FL, March 2005.

[105] Y. Zhou, D. Chiu, and J. Lui. A simple model for analyzing P2P streaming protocols. In *Proceedings of ICNP*, Beijing, China, October 2007.

[106] G. Zhu, X. Luo, and Y. Miao. Exact weight perfect matching of bipartite graph is NP-Complete. In *Proceedings of the World Congress on Engineering (WCE)*, volume 2, pages 1–6, London, UK, July 2008.