Design of Scalable On-Demand Video Streaming Systems Leveraging Video Viewing Patterns

Kyung-Wook Hwang

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2013

ABSTRACT

Design of Scalable On-Demand Video Streaming Systems Leveraging Video Viewing Patterns

Kyung-Wook Hwang

The explosive growth in on-demand access of video across all forms of delivery (Internet, traditional cable, IPTV, wireless) has renewed the interest in scalable delivery methods. Approaches using Content Delivery Networks (CDNs), Peer-to-Peer (P2P) approaches, and their combinations have been proposed as viable options to ease the load on servers and network links. However, there has been little focus on how to take advantage of user viewing patterns to understand their impact on existing mechanisms and to design new solutions that improve the streaming service quality.

In this dissertation, we leverage on the observation that users watch only a small portion of videos to understand the limits of existing designs and to optimize two scalable approaches — the content placement and P2P Video-on-Demand (VoD) streaming. Then, we present our novel scalable system called Joint-Family which enables adaptive bitrate streaming (ABR) in P2P VoD, supporting user viewing patterns.

We first provide evidence of such user viewing behavior from data collected from a nationally deployed VoD service. In contrast to using a simplistic popularity-based placement and traditionally proposed caching strategies (such as CDNs), we use a Mixed Integer Programming formulation to model the placement problem and employ an innovative approach that scales well. We have performed detailed simulations using actual traces of user viewing sessions (including stream control operations such as pause, fast-forward, and rewind). Our results show that the use of segment-based placement strategy yields substantial savings in both disk storage requirements at origin servers/VHOs as well as network bandwidth use. For example, compared to a simple caching scheme using full videos, our MIP-based placement using segments can achieve up to 71% reduction in peak link bandwidth usage.

Secondly, we note that the policies adopted in existing P2P VoD systems have not taken user viewing behavior — that users abandon videos — into account. We show that abandonment can result in increased interruptions and wasted resources. As a result, we reconsider the set of policies to use in the presence of abandonment. Our goal is to balance the conflicting needs of delivering videos without interruptions while minimizing wastage. We find that an Earliest-First chunk selection policy in conjunction with the Earliest-Deadline peer selection policy allows us to achieve high download rates. We take advantage of abandonment by converting peers to "partial seeds"; this increases capacity. We minimize wastage by using a playback lookahead window. We use analysis and simulation experiments using real-world traces to show the effectiveness of our approach.

Finally, we propose Joint-Family, a protocol that combines P2P and adaptive bitrate (ABR) streaming for VoD. While P2P for VoD and ABR have been proposed previously, they have not been studied together because they attempt to tackle problems with seemingly orthogonal goals. We motivate our approach through analysis that overcomes a misconception resulting from prior analytical work, and show that the popularity of a P2P swarm and seed staying time has a significant bearing on the achievable per-receiver download rate. Specifically, our analysis shows that popularity affects swarm efficiency when seeds stay "long enough". We also show that ABR in a P2P setting helps viewers achieve higher playback rates and/or fewer interruptions.

We develop the Joint-Family protocol based on the observations from our analysis. Peers in Joint-Family simultaneously participate in multiple swarms to exchange chunks of different bitrates. We adopt chunk, bitrate, and peer selection policies that minimize occurrence of interruptions while delivering high quality video and improving the efficiency of the system. Using traces from a large-scale commercial VoD service, we compare Joint-Family with existing approaches for P2P VoD and show that viewers in Joint-Family enjoy higher playback rates with minimal interruption, irrespective of video popularity.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to my advisors Vishal Misra and Dan Rubenstein who led me throughout the process of doing research, provided encouragement and support, and were so wonderful in helping me through difficult problems. I was truly fortunate to pursue the doctoral program under their guidance. I am deeply grateful to Dr. K. K. Ramakrishnan for his priceless advices and guide throughout many projects. I deeply thank my research collaborators, Vijay Gopalakrishnan, Rittwik Jana, and Seungjoon Lee for their valuable discussions and insightful feedback. I dedicate this thesis to my wife Joohyun whose love and understanding made me finish this thesis, and my parents who have always supported me in everything.

Chapter 1

Introduction

Video-on-Demand (VoD) has grown rapidly as the leading source of Internet traffic. Recent estimates [san, 2013] show that it accounts for about 62.0% of peak aggregate traffic (up from 29.5% in 2009) in North America. This increasing demand put significant pressure on content delivery systems requiring more costs of sustainable infrastructure such as disk storage and network bandwidth, in order to satisfy as many end users as possible. There has been renewed interest in scalable approaches for delivering videos. However, there has been little focus on how to take advantage of user viewing patterns to understand their impact on existing mechanisms and to derive new solutions that improve the streaming service quality.

In this dissertation, we leverage on the observation that users watch only a small portion of videos (we call this behavior *abandonment*) to understand the limits of existing designs and to optimize two scalable approaches — the content placement and Peer-to-Peer (P2P) VoD streaming. Then, we present our novel scalable system called Joint-Family which enables Adaptive Bitrate streaming (ABR) in P2P VoD, supporting user viewing patterns.

In the first part of the dissertation, we characterize user viewing behaviors using data collected from a real-world commercial VoD service. We provide proof that users watch only a small portion of videos (not just for short clips, but even with full-length movies). Based on this information, we explore how to take advantage of user viewing patterns to place content in provider networks to reduce their storage and network utilization. We use a optimization.

Moreover, we contend that the policies used in existing P2P VoD systems are not optimized because they have not taken user viewing patterns and viewers abandon prematurely. We show that the chunk selection and the peer selection policies are interdependent. They need to be addressed holistically and improved upon.

In the second part of the dissertation, we propose Joint-Family, a P2P streaming video delivery system that has peers participating in multiple swarms simultaneously, uses Earliest-First as the chunk selection policy, and Earliest-Deadline as the peer selection policy. We show analytically why our policies are suited for P2P VoD. Joint-Family with its seamless multiswarm approach enables P2P systems to support adaptive bitrate videos by leveraging resources across multiple swarms.

1.1 Leveraging Video Viewing Patterns for Scalable Streaming Systems

1.1.1 Optimal Video Segment Placement

To keep up with the explosively increasing demand on streaming videos, content providers have adopted a range of approaches including Content Distribution Networks (e.g., Akamai, Limelight), Peer-to-Peer based delivery (e.g., PPLive, PPStream, Zattoo) or hybrid combinations of the two (e.g., LiveSky [Yin *et al.*, 2009b]). Cable and Internet television (IPTV) providers use networks similar to content distribution networks for distributing and delivering on-demand content.

Despite the differences in the actual delivery mechanism, there is one common factor: Most of these approaches focus on content delivery and ignore storage. Instead they assume that a full copy of the content is stored in multiple, if not all, locations. These include the origin servers in case of content distribution networks (CDNs) or Video Hub Offices (VHOs) in the case of cable and IPTV providers.

Intelligent storage, however, can significantly reduce cost and improve delivery efficiency. It has been known for a while that content popularity is significantly skewed [Guo *et al.*, 2007a; Yu *et al.*, 2006; Yin *et al.*, 2009a]; as a result not every video needs to have an equal number of copies. More importantly, there is increasing evidence that users do not watch entire videos [Yu *et al.*, 2006; Yin *et al.*, 2009a; Li *et al.*, 2011]. Whether it is short clips, or full movies, users stop watching the video prematurely. Such behavior is consistent with a well established finding in psychology that people often make judgments about a subject without evaluating all aspects of the subject [Banaji and Hardin, 1996]. Further, with the ability to skip or replay portions of the video, users tend to skip portions of the video they watch [Gopalakrishnan *et al.*, 2011]. This, we argue, is crucial evidence against storing full videos in multiple locations.

Storage is cheap, so why bother? Despite the falling prices of storage, we believe that it is worthwhile to consider storing portions of the video in most locations. First, although the cost of *consumer* storage has fallen significantly, enterprise grade storage, as needed by a high performance and highly available streaming solution is many orders more expensive. Specifically, while it is possible to purchase a 2TB hard disk around 100 USD today, the cost of carrier grade storage servers with the same disk capacity can be higher by two orders of magnitude (e.g., 30K USD) [sun, 2010]. Most of the cost comes from the strict redundancy requirements and the sustained streaming rate needed for video delivery. Second, to cater to growing demand and to gain competitive advantage, providers have been *growing their library* catalogues at a rapid rate [com, 2010]. Third, *transferring* this ever increasing number of videos to all the locations and storing them there, especially when not all of the data will be viewed, is not just inefficient but also strains the distribution system. Finally, storing portions of the video opens up opportunities to *scale* the library significantly for the same amount of disk space.

In this dissertation, we seek to leverage users' viewing patterns to design a more efficient video placement strategy. Ideally, we want to identify small portions for each video that can satisfy a large number of requests. By replicating those portions in multiple places, we can maximize the number of requests that are satisfied locally with minimum disk usage.

Various studies have looked at the problem of content placement. For example, many approaches [Borst *et al.*, 2010; Baev *et al.*, 2008; Valancius *et al.*, 2009] use optimization-based techniques to guide placement. These, however, resort to heuristics to solve the problem or ignore important constraints such as link bandwidth. As a result, simple caches [Yin *et al.*, 2009b; Allen *et al.*, 2007] and cache replacement policies [Allen *et al.*, 2007;

Wu and Li, 2009] have been used within the CDN, cable network, or at the clients. Caching, however, can only be successful if the available cache is large enough to store the "working set". This working set is the set of unique videos requested over a time window. As we show in Section 3.2, the working set size at a location can be as large as 50% of the entire library. In this setting, smaller caches result in significant churn, rendering the cache ineffective.

Our design and analysis of Joint-Family makes the following essential contributions to the field of streamed VoD:

- We first analyze video access data from a nationally deployed VoD service and confirm that many requests indeed consume small fractions of videos (Section 3.2). We provide a detailed analysis of how viewers watch videos and describe how their usage changes across different parameters such as video size and popularity. Based on this finding, a natural approach is to break a video into multiple segments, which become units of storing and fetching content in our system. In this we study the advantages gained by two approaches: (a) splitting a video into a prefix and a suffix, and (b) splitting the video into fine-grained segments (e.g., chunks as in P2P systems).

- Instead of heuristics [Borst *et al.*, 2010; Baev *et al.*, 2008; Valancius *et al.*, 2009] or caching [Allen *et al.*, 2007; Wu and Li, 2009], we use a mixed integer program (MIP) formulation, whose solution guides how to place each segment across different locations (Section 4.1). The MIP formulation takes a projected demand for each segment, a disk constraint at each location, and a bandwidth constraint for each link, and computes a solution that minimizes the overall network utilization. While the formulation and the solution technique draw from our previous work [Applegate *et al.*, 2010], the problem of placing segments adds new challenges. For example, with chunks, the scale of the problem grows so large that even the approach in our earlier work [Applegate *et al.*, 2010] is not able to solve the MIP quickly. As a result, we try to cluster different chunks into groups and place these groups.

- We use detailed simulation experiments and compare our optimal placement approach against alternate schemes (Section 4.2). Our results show that, there is tremendous benefit to storing just portions of a video. For example, chunk-based MIP placement can achieve 71% reduction in peak link bandwidth usage over LRU caching with full-length videos and 52% reduction over MIP placement of full-length videos. Even simply utilizing prefixes in the MIP-based placement results in up to 62% reduction in peak link bandwidth usage.

1.1.2 User Abandonment and its Impact on P2P VoD

Peer-to-peer (P2P) systems have become part of the mainstream content distribution environment, improving the viewing experience by utilizing the upload capacity of the downloading nodes, thereby increasing overall upload capacity. Even traditional Content Distribution Network (CDN) providers such as Akamai [Maggs, 2012] are choosing to experiment with and deploy P2P-based delivery of video content.

and is being considered for commercial deployment VoD services [Maggs, 2012]. Much of this progress can be attributed to previous research identifying the set of policies that enable robust and scalable P2P delivery systems. In this dissertation, we consider the problem of viewers abandoning videos part way through the viewing of the video. While mostly overlooked so far, we show that *abandonment* (also called viewer engagement in other work [Dobrian *et al.*, 2011]) is a critical factor to consider since it directly affects the impact of various policies used for P2P VoD.

The two most important policies that determine P2P performance are the chunk- and peer-selection policies. File sharing systems have traditionally used a combination of Tit-for-Tat (TFT) as the peer selection and Rarest-first (RF) as the chunk selection as this combination offers the best tradeoff in terms of performance and fairness. Unfortunately, this combination does not work as well with streaming video, be it live or on-demand, since a video is generally consumed sequentially. Instead, an Earliest-First (EF) policy is a more natural chunk selection policy for video streaming. EF, however, is incompatible with TFT as peers in different points of playback have very little content of mutual interest to exchange with each other. As a result, there has been a lot of work to identify hybrid chunk selections strategies (e.g., EF+RF) [Zhou *et al.*, 2007; Fan *et al.*, 2010; Borghol *et al.*, 2010; Carlsson and Eager, 2007; Vlavianos *et al.*, 2006; Shah and Paris, 2007] that find a compromise between the need of streaming to get sequential data and TFT's need for diversity.

While the design of EF+RF implicitly assumes that viewers watch entire videos, recent studies [Yin *et al.*, 2009a; Li *et al.*, 2011] show that viewers of both short clips and full-length movies often watch only a small portion of a video and abandon the video part way through. Such abandonment may be attributed to how users find movies of interest (e.g., surfing for interesting content) or the possibility that a viewer loses interest in the content. Such viewer abandonment of videos has significant implications on the design of P2P policies. For example, peers using EF+RF may end up downloading rare chunks that they do not actually watch later due to abandonment. In this case, it would be more beneficial to use that upload capacity to deliver chunks that peer immediately need, to improve video playback experience and reduce unnecessary bandwidth consumption.

In this dissertation, we reconsider the set of chunk and peer selection policies to use in real-world P2P VoD systems with viewer abandonment (Section 5.1). We show that EF is a more appropriate chunk selection strategy in the presence of abandonment. Instead of using TFT, we introduce Earliest-Deadline (ED) as the peer selection strategy. In ED, a node picks peers with the earliest deadline among chunks when deciding which request to serve. Choosing ED not only gives us substantial performance improvement (as seen in our experiment results), but also allows us to break the inter-dependence between chunk- and peer selection that TFT introduces. While EF itself reduces wasted download compared to EF+RF, we introduce the notion of a playback lookahead window (PLA) to further limit the download rate, as is used in HTTP streaming [Akhshabi *et al.*, 2011; Rao *et al.*, 2011]. The window is sized such that when it is full, the user will not encounter interruptions. At the same time, users do not request any more data than needed to avoid "wasting" bandwidth in case they abandon the video.

Note that while the properties of churn and abandonment are similar in philosophy, there are still subtle differences. In particular, with abandonment, users do not necessarily leave the system. Instead they may stay connected and watch a different video after a short period of time, or even stay idle in the system. Instead of treating abandonment as a departure, we take advantage of the peer's staying connected by getting it to continue to participate in the swarm as a "partial seed" until the user watches the next video. We define a partial seed as a peer that does not have the entire video, but is not actively watching (and downloading) the video. These partial seeds continue to serve requests for the chunks they have downloaded already, and thus contribute to increasing the overall system capacity. We show using an analytical model accounting for abandonment and partial seeds (in Section 5.2) that the partial seed staying time significantly influences the swarm's performance. We evaluate our design using detailed simulations based on traces of user requests collected from a nationally deployed VoD service (Section 5.3).

In this dissertation, we make the following important contributions:

- Through trace-driven simulations, we show that existing P2P VoD systems do not perform well in presence of abandonment. They experience more interruptions compared to when users watch videos entirely. More importantly, we show that viewers that watch more of the video experience severe interruptions.

- In contrast to previous findings, our results show that with abandonment, a hybrid policy of EF+RF performs *worse* and results in longer interruption periods than EF. With EF, using ED instead of TFT brings significant performance enhancements.

- We develop a detailed analytical model that accounts for abandonment and partial seeds and show that "useful" download rate of EF effectively improves as partial seeds stay longer.

- We show that the combination of ED, EF, and partial seeds can significantly improve overall video playback performance, while PLA further reduces wasted bandwidth consumption.

1.2 Enabling Adaptive Bitrate Streaming in P2P VoD

The ever-increasing demand placed by streamed video traffic across both wired and wireless networks has been managed by two seemingly complementary approaches: adaptive bitrate (ABR) [ss, 2010; ado, 2010], and P2P delivery [joo, ; uus,]. ABR encodes a video at multiple bitrates, and maximizes the video bitrate within the available bandwidth, giving a higher fidelity video when possible, and dropping to lower quality rather than forcing an interruption of playback. P2P-based systems are a popular alternative to deliver on-demand videos as explained in the previous subsection. Intuitively, these two orthogonal approaches are not well-suited to work together, because viewers watching the video at differing rates are presumably unable to exchange video parts with one another. Hence, it appears that enabling ABR interferes with the ability for peers to share video parts with one another.

In this dissertation, we show that, contrary to current intuition, ABR and P2P can effectively be combined in a way that achieves the best of both worlds: P2P techniques improve upload capacity, and ABR enables the highest quality viewing at that capacity. Using a combination of analysis and simulations, we demonstrate the performance improvements offered by our novel design, which we call **Joint-Family**, and compare it to existing P2P video-on-demand (VoD) systems.

Our design and analysis of Joint-Family makes the following essential contributions to the field of streamed VoD:

- We identify a misleading generalization of the conclusion of previous fluid modeling efforts [Qiu and Srikant, 2004] that popularity of a stream does not have bearing on its effective download capacity. We also identify the conditions that contradict the conclusion of these models. We show that with ”sufficiently long” staying times, *content popularity in fact affects swarm efficiency*, yielding that more popular swarms have higher download rates than less popular swarms (Section 6.1.2).

- We analyze the effectiveness of caching previously watched videos and sharing them as a mechanism to extend seed staying time (Section 6.1.3). With caching we can also transfer underutilized capacity from one swarm to another and thereby improve global performance.

- We show how ABR, when combined with P2P, enables a swarm to efficiently adapt to the best encoding rate, without *a priori* knowledge of the video's popularity. (Section 6.1.4) We use our Markov model to show that ABR allows P2P swarms to migrate to the highest sustainable rate for that swarm: highly popular content will induce large swarms and have a high sustainable download capacity, whereas less popular content will have smaller swarms and a lower sustainable download capacity.

- Based on our analytical observations, we design a novel protocol called Joint-Family to deliver high-quality videos with minimal interruptions in a P2P system, using multi-swarm participation and ABR (Section 6.2).

A peer in Joint-Family caches and shares multiple ABR videos using storage space at the end system, and increases capacity of swarms (especially for unpopular videos) by supplementing it with (unused) peer capacity. Hence, the peer participates in multiple swarms concurrently, and shares different parts of the ABR video at multiple bitrates. To support this, we identify the right combination of chunk selection, peer selection, and bitrate adaptation policies that minimize interruptions. Our design makes Joint-Family immediately suitable for existing VoD infrastructures in which the provider owns the distribution infrastructure (e.g., CDNs [Maggs, 2012], IPTV [UVe,]). We also describe how the protocol can be applied in a decentralized setting by utilizing mechanisms that encourage sharing of content [Piatek *et al.*, 2010]. We conduct extensive performance evaluations of Joint-Family using traces from a nationally deployed VoD service (in Section 6.3), and show that ABR with P2P is indeed feasible. Compared to a generalized implementation of the state-of-the-art in P2P VoD, our instantiation of Joint-Family delivers high quality VoD streaming, even for unpopular videos, with minimal interruptions.

1.3 Overview of the Dissertation

This dissertation consists of two main parts. In Part I (Chapters 3, 4, and 5), we present user viewing patterns and our new approaches to video placement and P2P VoD that account for viewing patterns. Section 3.1 describes the real-world data set used in our study. We then analyze the data and present how videos are watched in Section 3.2. Section 4.1 presents the mixed integer program (MIP) that we use to determine the optimal content placement, and in Section 4.2 we evaluate the performance of our segment-based video placement through experiments. In Section 5.1, we re-evaluate key policies and design decisions for P2P VoD by taking viewer abandonment into account. We then analyze in Section 5.2 how abandonment affects the performance in P2P VoD. Section 5.3 evaluates our P2P VoD design that supports abandonment.

Part II (Chapter 6) presents our novel scalable system Joint-Family. Section 6.1 analytically shows how video popularity, the staying time of peers, and caching help increase system capacity. Also we analyze how adaptive bitrate can further improve the playback quality. Section 6.2 describes Joint-Family protocols, and Section 6.3 evaluates the performance of Joint-Family and compares it to state-of-the-art P2P approaches.

Chapter 2

Related Work

We discuss previous works on the design and analysis of video streaming systems and protocols. We first describe previous study related to user behavior or patterns in online video viewing (Section 2.1) since we seek to leverage users' viewing patterns to design more efficient video-on-demand (VoD) systems. We then present previous works on video caching and optimal video placement (Section 2.2) before we introduce our optimal video segment placement strategy in Chapter 4. We then review existing P2P VoD systems focusing on their main policies and design selections (Section 2.3), and we show in Chapter 5 that those existing schemes for P2P VoD do not perform well when users abandon a video part-way. Finally, we present the recent use of adaptive bitrate (ABR) video streaming which have been frequently adopted in practice (Section 2.4), and we show in Chapter 6 that Joint-Family's multi-swarm solution is a perfect match for the needs of ABR video distribution in P2P environment.

2.1 User Behavior in Online Video Watching

With more and more deployments of cable, IPTV, and P2P-based VoD systems cropping up, there has been a large amount of data to study different aspects of how users watch the videos. Yu et al. [Yu *et al.*, 2006] study the user behavior in PowerLive, a VoD system deployed in China. Huang et al. [Huang *et al.*, 2008] study different aspects the PPLive [ppl,] P2P VoD system. Yin et al [Yin *et al.*, 2009a] study user behavior using the data collected from VoD access to the 2008 Beijing Olympics. Gopalakrishnan et al. [Gopalakrishnan *et al.*, 2010] study how users use the DVD-like operations (e.g., skip, fast-forward, rewind) in a large-scale VoD system. Especially, Yin et al. [Yin *et al.*, 2009a] and Li et al.[Li *et al.*, 2011] present a measurement study on viewer abandonment in a large scale landline-based and mobile-based IPTV services provider, respectively. They both demonstrate that users often watch only a small portion of a video. In Chapter 3 we provide concrete evidence to viewer abandonment by measuring how often users abandon videos and how much of a video they watch, and we also study how user behavior correlates to different aspects such as video length and popularity.

Aalto et al. [Aalto *et al.*, 2011] analyze viewer abandonment in P2P VoD with limited simulation scenarios for model verification. Our work in Section 5.2 analyzes the contribution of "partial seeds" and performs practical evaluation with real traces to measure impact of abandonment in the real world.

2.2 Content Placement

Work related to our optimal video segment placement (Chapter 4) can be divided into two broad categories: work on full video and prefix caching, and work on optimal content placement. We describe each of these areas here.

Utilizing Video Prefixes Despite the lack of concrete evidence, the notion of taking advantage of prefixes is not new. Previous studies typically have focused on *caching* prefixes of videos, either for fast startup, or to reduce cache space. Sen et al. [Sen *et al.*, 1999] and Zhang et al. [Zhang *et al.*, 2000] propose caching an initial prefix of video to provide fast start-up, deal with jitter, and reduce server load. Park et al. [Park *et al.*, 2001] propose a caching scheme that stores a portion of the entire video, where the amount of video cached is determined by its popularity. Wang et al. [Wang *et al.*, 2002] attempt to minimize aggregate network bandwidth by analytically determining the optimal proxy prefix cache. Wu et al. [Wu *et al.*, 2001] propose partitioning videos into exponentially increasing segments and caching these segments. They also propose cache admission and replacement strategies that determine which segments to cache and which ones to replace. More recently, Huang et al. [Huang *et al.*, 2007] and Allen et al. [Allen *et al.*, 2007] study the use of P2P nodes for caching videos. The goal of both approaches is to reduce server load by using peers to serve the content. All these approaches rely on servers when content delivery from caches or peers is not possible. In contrast, our focus is on placing content among the backend servers and to take advantage of client viewing patterns among these backend servers.

Optimal Placement Approach Content placement and replication has, in general, been a topic of extensive research. Qiu et al. [Qiu *et al.*, 2001] consider the problem of positioning web server replicas to minimize the overall cost. Other work [Barbir *et al.*, 2003; Alzoubi *et al.*, 2008] focuses on how to direct user requests to the "best" among replicated servers (also known as *request routing*). Most existing work focuses on minimizing latency given constraints (e.g., server capacity), but do not consider network link capacity constraint. There have also been several efforts to address the problem of content placement using analytical framework. Valancius et al. [Valancius *et al.*, 2009] propose an LP-based heuristic to identify which videos and how many replicas to be placed at customer home gateways. Borst et al. [Borst *et al.*, 2010] focus on minimizing link bandwidth utilization assuming a tree structure with limited depth. Both proposals focus on a network structure that connects consumers to the server. In contrast, we consider placing content in the backbone networks with diverse disk and link bandwidth constraints. Baev et al. [Baev *et al.*, 2008] consider the *data placement problem* where the objective is to minimize the cost without taking bandwidth into account. They prove that their problem is NP-hard via a reduction to the uncapacitated facility location problem. Since the data placement problem is a special case, our problem is also NP-hard. Baev et al. also show that when object length is not the same, even deciding feasibility is NP-hard, and no approximation algorithm is possible unless P=NP. Our problem is strictly more complex, since we also consider link constraints. We also find provably-good solutions (e.g., within 1% of optimal) on instances arising from real-world large-scale systems dealing with diverse objects. Zhou and Xu [Zhou and Xu, 2002] consider the problem of minimizing the load imbalance among servers subject to disk space and server network bandwidth constraints. Their work only considers the egress link capacity from servers.

2.3 Abandonment and its Impact on P2P VoD

Our results in Chapter 5 show that viewer abandonment has a significant effect on the performance of existing P2P VoD streaming systems and that existing schemes for P2P VoD should be reconsidered to cope with more realistic demands. In this section, we study some of the key policies and design decisions for previous P2P VoD works.

Chunk Selection: To adapt BitTorrent for streaming systems (either live or VoD), a combination of the rarest first (RF) chunk selection policy and sequential chunk download (EF) has been exploited [Zhou *et al.*, 2007; Hwang *et al.*, 2008; Borghol *et al.*, 2010; Carlsson and Eager, 2007; Vlavianos *et al.*, 2006; Shah and Paris, 2007; Fan *et al.*, 2010]. The specific details of the proposed schemes vary from simple probabilistic hybrid models to using sophisticated network coding techniques. Previous literature claims that achieving balance between system utilization (by RF) and on-line playback (by EF) can substantially improve playback quality. However, we show in Chapter 5 that with viewers' abandonment, such a hybrid policy greatly degrades the playback performance. We further show that using EF only achieves better playback performance.

Peer Selection: BitTorrent's TFT is effective for file sharing with its inherent incentive mechanism to encourage a peer's contribution. However, several prior works [Huguenin *et al.*, 2010; Zhang *et al.*, 2011; Wen *et al.*, 2011; D'Acunto *et al.*, 2010; Shah and Paris, 2007; Yang *et al.*, 2010; Mol *et al.*, 2008; Guo *et al.*, 2007b] show that TFT is not suitable for streaming applications. This is primarily because chunk selection using RF is not suitable for streaming, and TFT without RF makes it difficult for new peers to contribute to older peers, thus preventing them from fully helping each other. Various peer selection approaches have been proposed for streaming. Shah et al. [Shah and Paris, 2007] modify TFT's optimistic unchoke, D'Acunto et al. [D'Acunto *et al.*, 2010] make peers act more altruistically, and Wen et al. [Wen *et al.*, 2011] group peers with similar playback points to help each other. To satisfy a viewer's uninterrupted playback experience, in Section 5.1.3 we replace TFT with the Earliest-Deadline (ED) policy, which ensures that each chunk is delivered to the viewer prior to its deadline.

Limiting Rate to Avoid Wastage: Popular VoD services such as YouTube (using Progressive Download) and Netflix (using Adaptive Bit Rate) have adopted approaches that limit the amount of video bytes delivered beyond the current playback point so as to limit wastage on network bandwidth and also reduce the load on the network for VoD streaming. However, most of these approaches have been applied for server-based environments such as HTTP streaming [Rao *et al.*, 2011], progressive download [Ghobadi *et al.*, 2012; Alcock and Nelson, 2011], and adaptive bitrate streaming [Akhshabi *et al.*, 2011]. A similar capability is desired for P2P VoD streaming. We show in Section 5.3.3 that our approach of having a limited 'look-ahead' window can also reduce wastage caused by abandonment in P2P systems, while not hurting the viewer's playback experience.

2.4 Adaptive Streaming in Multi-swarm P2P VoD

In terms of swarm participation, a peer viewing a video typically participates in the swarm only for as long as it is viewing that content. With abandonment, the work done in downloading the data is wasted if the peer does not view that portion or share it with others. A natural mechanism to improve system capacity and efficiency in such scenarios is to allow a peer to stay and also be in multiple swarms. Also, an important practical consideration is that most VoD providers (e.g., Netflix, Hulu) move users between higher and lower quality video chunks based on observed user download rate using Adaptive Bitrate (ABR) streaming.

In this section, we describe related work on multi-swarm P2P and adaptive streaming and show in Chapter 6 that our Joint-Family's multi-swarm protocol is eminently suitable for the distribution of ABR video.

Multiple Swarms in P2P: While most of the work has improved the performance in a single swarm, little effort has been put on multiple swarms to utilize idle upload/download bandwidth of peers by means of added capacity obtained between swarms. Wu et al. [Wu *et al.*, 2009] and Wang et al. [Wang *et al.*, 2010a] investigate the peer's bandwidth allocation to contribute across multiple swarms in live streaming but not in VoD. Zhou et al. [Zhou *et al.*, 2011] model inter-swarm data exchange in VoD, however, their implementation requires centralized schemes for estimating the demand and supply for each content piece. Wang et al. [Wang *et al.*, 2010b] focus on adjusting the peer's inter-swarm contribution based on the demand, which corresponds to one of the many aspects considered in our work.

Adaptive Streaming: Adaptive bitrate (ABR) streaming has been gaining popularity as a way to enable users to experience the highest quality of videos. ABR dynamically adapts to the user's network and playback condition. There are several flavors of ABR implementations (e.g., MPEG-DASH, Adobe Dynamic Streaming, Microsoft Smooth Streaming, Apple HTTP Streaming, Netflix [Stockhammer, 2011; ado, 2010; ss, 2010; app, 2011; Akhshabi *et al.*, 2011]). While ABR has been used for HTTP server based streaming, the use of P2P systems for ABR are not yet common. Roverso et al. [Roverso *et al.*, 2012] implement ABR in P2P systems for live media only. To the best of our knowledge, we are the first to investigate ABR for P2P VoD.

Scalable video coding (SVC) [Schwarz *et al.*, 2007] is yet another approach that enables end-systems to adapt network conditions. The SVC streams consist of a base layer and multiple enhancement layers where each enhancement layer improves the video quality. SVC can also be supported in P2P systems for VoD. [Castro *et al.*, 2003; Petrocco *et al.*, 2011; Rückert *et al.*, 2012; Eberhard *et al.*, 2012] take a multiple description or layered coding approach which causes interdependency of layers per chunk distribution in P2P and which is not directly applicable to ABR or our P2P work. Also, SVC has not been widely implemented due to the complexity of decoding on the end-systems, and the additional bandwidth requirements compared to ABR. ABR deployment has far outpaced other alternatives.

Part I

Leveraging Video Viewing Patterns for Scalable Streaming Systems

Chapter 3

User Behavior in Online Video Viewing

3.1 Description of Data Set

We first describe the data set used in our study. We then analyze the data and present how videos are watched in the next section.

3.1.1 Trace Data

We obtained trace data collected for 16 days in January of 2010 from a nationally deployed video-on-demand service. This trace includes over 13 million requests for both free and paid videos belonging to various classes including music videos and trailers, TV shows, and full length movies. A record in this anonymized trace data includes information such as the requested video ID (typically a hash of the content), its length, its price, the time of request, and the video server serving request (i.e., the Video Hub Office (VHO)). It also contains information about the set of DVD-like operations that the user performed while watching the video in a given session (i.e., timestamps when the user started watching the video and when they stopped watching it).

Note that a session duration may be longer than the video length because the user may have re-wound or paused the video. Further, the actual amount of video watched is quite

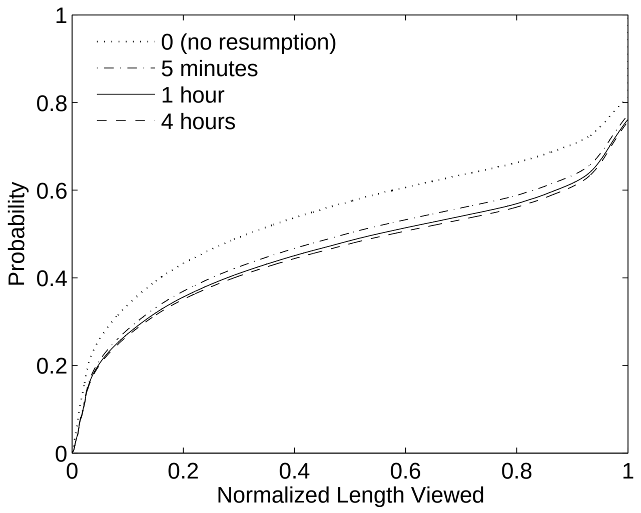


Figure 3.1: Cumulative distribution of the NLVs with different session resumption thresholds T_r.

different from the session duration or even the video length. To get an accurate estimate of how much of the video was actually watched, we included the time spent in fast-forward, rewind, pause, SKIP, and REPLAY. For example, suppose that a user spends the first 4 minutes fast-forwarding the video at 2x speed and then watches another minute before stopping the session. While the session length is 5 minutes, the amount of video consumed is 9 minutes. We measure the actual length of the video watched as a fraction of the video length and call it the *Normalized Length Viewed (NLV)*. In the above example, if the length of the video is 30 minutes, then NLV is 9/30=0.3.

3.1.2 Limitations of our data

Our data has certain limitations. Here we describe relevant limitations and how we address them.

3.1.2.1 Session Resumption

The VoD service allows users to watch a video partially and continue at a later time. However, our data did not clearly identify if a particular session by a user was a new one or

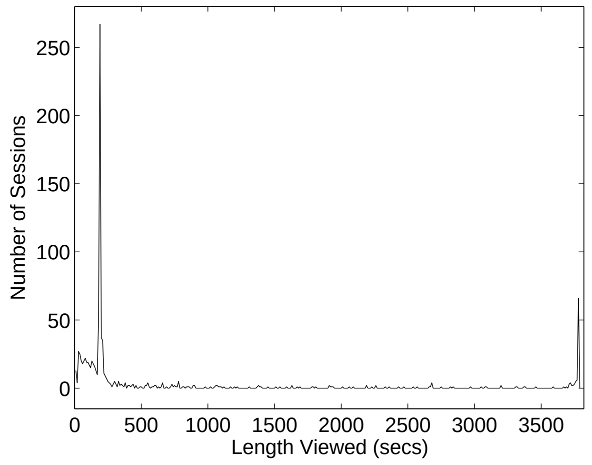


Figure 3.2: Typical signature of a video with free preview.

the resumption of a previous session. To overcome this, if a user requests the same video in two consecutive sessions that are less than T_r apart in time, then we assume that the later session resumes from where the previous session left off. Otherwise, we treat the later session as a new session watching from the beginning.

In Figure 3.1, we compare the NLVs when we take into account the session resumption with different thresholds T_r. $T_r = 0$ indicates that no session is resumed. First, we see that the NLVs under the session resumption are longer than those without resumption regardless of T_r. Next, we see that when we increase the threshold, we see that the change in NLVs decreases (e.g., T_r =1hr and T_r =4hr are almost identical). In the rest of the chapter, we use $T_r = 4$ since the number of consecutive sessions beyond this threshold was negligible in our trace data. Note that using this value is conservative and overestimates the access to latter part of videos.

3.1.2.2 Free Previews

The VoD service provides free short previews for some of the paid content. Users get to watch these previews before they decide if they want to pay for the full video. Unfortunately, our trace data does not have information that allows us to distinguish which videos have previews and for how long. Since these preview sessions are short, not identifying access to

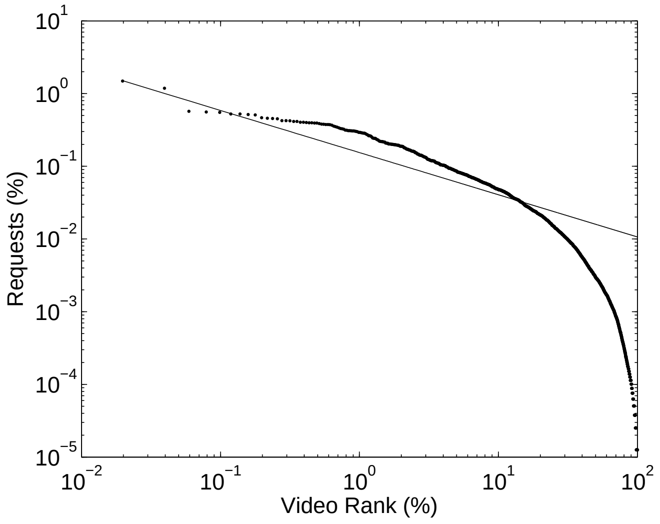


Figure 3.3: Fraction of viewer requests for all videos in the library (Zipf distribution). Rank 100% corresponds to the least popular video.

these previews can significantly skew our results.

To overcome this, we analyzed our data and observed that most paid videos with previews had a typical signature with a large proportion of the requests watching less than 5 minutes as shown in Figure 3.2. We use this signature to identify videos with free preview and exclude all sessions for these videos whose length is shorter than 5 minutes. Again, we believe this is a conservative threshold (most of the preview sessions are $1 \sim 3$ minutes) and underestimates the access to earlier part of videos.

3.2 Motivation for Placing Video Segments

By analyzing our data described in the previous section, we provide the motivation for placing segments (Chapter 4) rather than full videos. We study how much of a video users watch, and then investigate how this correlates to video properties such as video length and popularity.

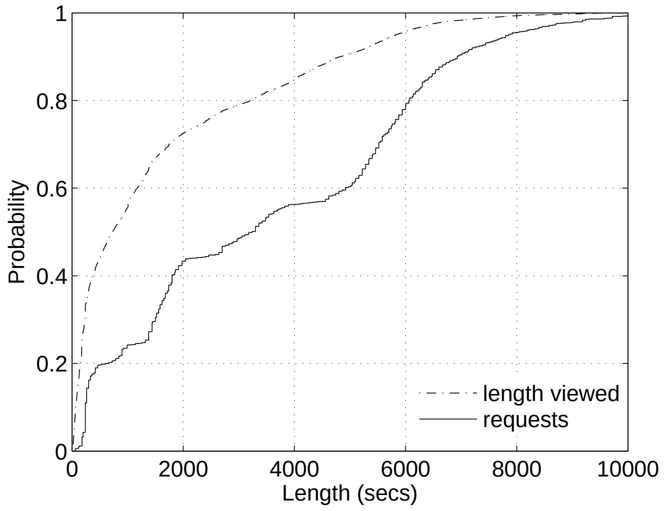


Figure 3.4: Cumulative distribution of requests to video of different sizes. The plot also shows the cumulative distribution of requests with different lengths viewed.

3.2.1 Most Requests Are Concentrated on Popular Videos

In Figure 3.3, we plot how many requests each video has during the 16 day period. The figure shows that most of viewer requests are concentrated on the limited number of very popular videos. The top 10% popular videos of the entire library have more than 75% of the entire requests. This viewing pattern strongly motivates our optimal video placement in Chapter 4, rather than replicating the entire content library at every VHO.

3.2.2 Most Users Watch Fraction of a Video

We first study the relation between the length of requested videos and the NLV for these viewing sessions. In Figure 3.4, we show the video length and the viewed length for all requests and plot the cumulative distribution. We observe that 43% of the requests are to videos that are 2000 secs (e.g., a 30 minute TV show) or less. By contrast, the amount of video watched by users is much shorter. Specifically, about 73% of the cases watched 2000 seconds or less of the requested video. This result indicates that most users do not watch videos fully.

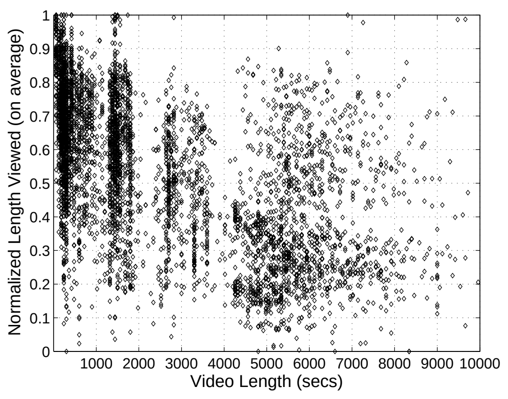


Figure 3.5: Scatterplot of averaged NLV for each video of different lengths. One point represents a single video.

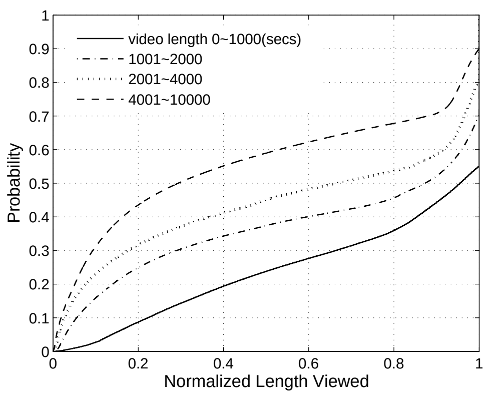


Figure 3.6: Cumulative distribution of NLV for different classes of video lengths.

3.2.3 Correlation between NLV and Video Properties

3.2.3.1 Longer videos tend to have smaller NLVs

Having established that viewers watch only a portion of the video, we examine the correlation between the length of a video and the normalized length viewed. In Figure 3.5, we show a scatter plot of the average NLVs for all requests to each video of different lengths. We also see some interesting trends here. First, we see that shorter videos tend to have larger NLVs. We also see four clusters of video lengths. This is not surprising because of the nature of the content in the video library: music videos and trailers, episodes of shows, documentaries, and movies.

In Figure 3.6, we plot the cumulative distributions of the four clusters of video lengths as identified in the previous figure. We clearly see that longer videos tends to have smaller NLV values. For the cluster of videos of the greatest length, around 55% of the requests stopped before reaching 40% of the video length. By contrast, for the group of shortest videos, only less than 20% of the requests stopped before viewing 40% of the video.

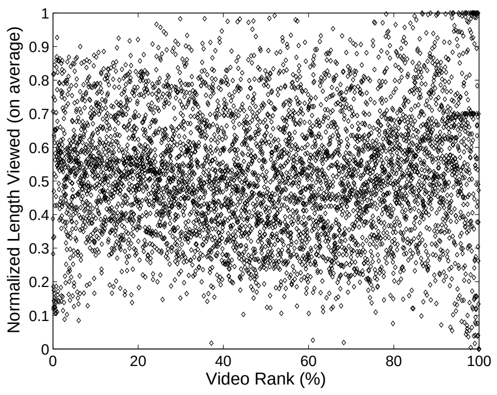


Figure 3.7: Scatterplot of NLV for videos of decreasing popularity. (Rank 100% corresponds to the least popular video.)

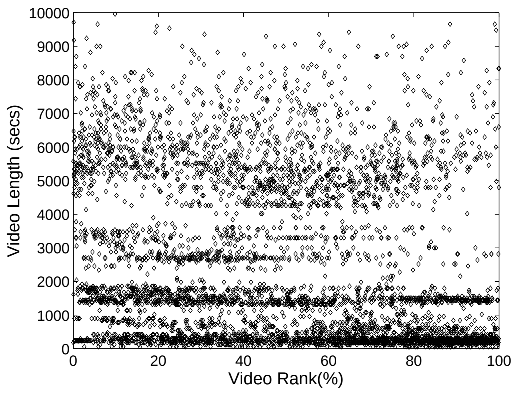


Figure 3.8: Scatterplot of video length for videos of decreasing popularity.

3.2.3.2 Weak correlation between popularity and NLV

We now study the relationship between the popularity of a video and how much of it is viewed. Conventional wisdom says that popular videos tend to be viewed fully. This belief also forms the basis for some of the prefix caching approaches [Park *et al.*, 2001]. For each video, we calculate the average NLV across all requests to the video and plot the result in Figure 3.7. Surprisingly, the figure shows that there is *little to no correlation* between the popularity of a video and how much of it is actually viewed.

We further investigate if there is any bias in popularity towards videos of a certain size. Figure 3.8 shows the relation between the length of videos and their popularity. As in Figure 3.5 we see four clusters of videos based on length. Also, within a given cluster, there are some indications of bias in popularity. For example, it seems that short videos are denser in the unpopular zone, while the density is higher for longer videos in the popular zone. However, there appears no strong bias in popularity across these clusters. Although not shown here, we also examined a subset of videos (e.g., longer than 4000 seconds), but did not observe any strong correlation between popularity and NLV.

This result indicates that just taking the popularity of a video and deciding how to store it may lead to an inefficient placement decision. We contend that there are benefits to be

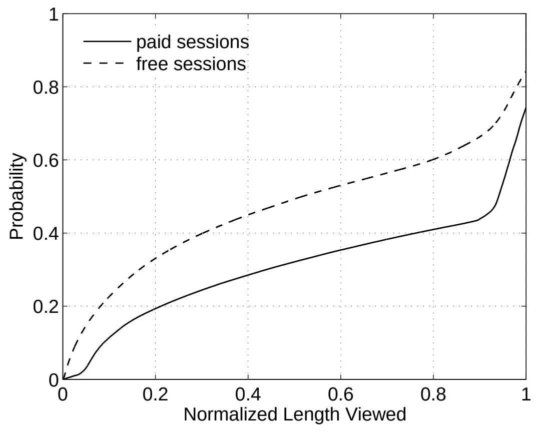


Figure 3.9: Cumulative distribution of the NLVs for free and priced videos.

gained by looking to store videos at a finer granularity.

3.2.3.3 Paid videos have longer NLV than free ones

We separately explore sessions for free and paid videos since it is generally assumed that viewers are more likely to watch longer when they pay for viewing. Figure 3.9 confirms that this assumption is indeed true, showing that paid sessions are significantly longer than free sessions. For example, about 45% of the requests for free videos stopped watching before 40% of the video while only 25% for paid videos stopped at that point.

3.2.3.4 Small difference between weekdays and weekends

We investigated the NLVs of each day's sessions from Monday to Sunday, and we observe that the NLV is the largest on a Saturday and the smallest on a Monday. However, the difference is modest. Specifically, in our results, about 51% of viewers on Monday watched 50% or less of the video, while 47% on Saturday watched the same amount.

3.2.3.5 NLV depending on genres

We also investigated the difference in NLV depending on the genres (e.g., kids, action) of videos and found that most genres show a similar NLV distribution (the result not displayed here). For example, in the case of paid videos, the majority of genres follow a curve similar to the one shown in Figure 3.9, although a few genres slightly deviate from the curve (e.g., adult genre with shorter viewing sessions even for paid videos).

Chapter 4

Optimal Video Segment Placement

4.1 MIP Formulation

In this section we present the mixed integer program (MIP) that we use to determine the optimal content placement. While we have presented this formulation in our earlier work [Applegate *et al.*, 2010], we describe it here for completeness. We also discuss new aspects of the problem when we consider placing portions of a video.

4.1.1 Problem Formulation

Given a request pattern for each video at each VHO over a time period, our goal is to find a video placement that minimizes the total network consumption while satisfying all user requests within the link bandwidth and disk capacity constraints. We first describe our input parameters and decision variables. Let V denote the set of VHO locations, L the set of directed links between these locations, W the set of videos in our video library, and M the set of distinct segments (e.g., chunks, or prefix and suffix) of video.[1] The set of time slices at which we enforce the link bandwidth constraints is T. We take the learning from [Applegate *et al.*, 2010] to identify the time slices to be 1-hour windows during the peak busy period on weekend days. Each VHO i has disk capacity D_i, and the size of

[1] In case a video is not broken into multiple pieces, $M = W$.

Parameters	Meaning
V	set of VHOs (vertices)
L	set of directed links
W	set of videos
M	set of distinct segments of all videos in W
T	set of time slices
D_i	disk capacity at $i \in V$ (in GB) reserved for fixed storage
s^m	size of video segment $m \in M$ (in GB)
P_{ij}	set of links on path used by $i \in V$ to serve requests at $j \in V$
B_l	capacity of link $l \in L$ (in Mbps)
r^m	bitrate of video segment $m \in M$ (in Mbps)
a_j^m	aggregate # of requests for video segment $m \in M$ at VHO $j \in V$
$f_j^m(t)$	# of requests for segment $m \in M$ at VHO $j \in V$ that are active at time $t \in T$
c_{ij}	cost of transferring one GB from $i \in V$ to $j \in V$
Decision variables	**Meaning**
y_i^m	binary variable indicating whether to store segment $m \in M$ at VHO $i \in V$
x_{ij}^m	fraction of requests for segment $m \in M$ at $j \in V$ served by $i \in V$

Table 4.1: Input parameters and decision variables used in the MIP

$m \in M$ is s^m. Instead of just having a few distinct video sizes (e.g., 4 in [Applegate *et al.*, 2010]), we allow a video to be of arbitrary size. For each pair of VHOs $i, j \in V$, we assume that there is a fixed directed path P_{ij} from i to j. Serving a request locally requires no links, so $P_{ii} = \emptyset$. The capacity of link $l \in L$ is B_l, while the bit rate of video segment $m \in M$ is r^m (both in Mbps). For each segment $m \in M$, VHO $j \in V$ receives a_j^m requests during the entire modeling period. Using the detailed DVD-like stream control traces (not just the distinct video requests by the user), we identify the exact length of the video that the user views, and thereby derive the counts for each portion of the video segment requested. At any given time slice $t \in T$, the number of concurrent streams is $f_j^m(t)$. We denote the cost of serving one GB of video from i to j by c_{ij}. This cost for a remote service is proportional to the number of hops between i and j (i.e., $|P_{ij}|$).

Our MIP model has two types of decision variables. For each $i \in V$ and each $m \in M$, y_i^m indicates whether we store m at i. y_i^m is a binary variable, because given m, i always stores it in its entirety or not (i.e., yes, if $y_i^m = 1$; no, if $y_i^m = 0$). When a request for m arrives at VHO j, it is served locally if the video is stored at j; otherwise, it must be fetched from some other VHO storing m. If there are multiple such VHOs, the variable x_{ij}^m tells what fraction of requests should be served from VHO i. Table 4.1 summarizes the symbols used and their meaning. The top section lists the input parameters, which our MIP treats as fixed.

Our MIP formulation is:

$$
\begin{aligned}
\min \quad & \sum_{m \in M} \sum_{i,j \in V} s^m a_j^m c_{ij} x_{ij}^m && (4.1)\\
\text{s.t.} \quad & \sum_{i \in V} x_{ij}^m = 1,\ \forall m \in M, j \in V && (4.2)\\
& x_{ij}^m \le y_i^m,\ \forall i, j \in V, m \in M && (4.3)\\
& \sum_{m \in M} s^m y_i^m \le D_i,\ \forall i \in V && (4.4)\\
& \sum_{m \in M} \sum_{\substack{i,j \in V:\\ l \in P_{ij}}} r^m f_j^m(t) x_{ij}^m \le B_l,\ \forall l \in L, t \in T && (4.5)\\
& x_{ij}^m \ge 0,\ \forall i, j \in V, m \in M && (4.6)\\
& y_i^m \in \{0, 1\},\ \forall i \in V, m \in M && (4.7)
\end{aligned}
$$

The objective expressed by (4.1) is to minimize the overall cost of byte transfer while serving *all* the requests for the entire period (Constraint (4.2)) without violating the disk capacity and link bandwidth capacity constraints (Constraints (4.4) and (4.5)). Constraint (4.3) captures the fact that location i can serve m only when it has a local copy.[2]

[2] Constraints (4.2) and (4.3) combine to ensure that every $m \in M$ is available in the system. That is, they imply $\sum_{i \in V} y_i^m \ge 1,\ \forall m \in M$.

4.1.2 Managing the Scale of the MIP

It is hard to find an optimal solution (due to NP-hardness) to the above MIP. We have presented a practical approach in our earlier work [Applegate *et al.*, 2010] that allows us to solve the MIP quickly. However, despite the advances presented in that work, the scale of the problem grows significantly when we attempt to place chunks. For instance, when an average video size is an hour, breaking each video into 10-second chunks increases the number of variables in the MIP formulation by a factor of 360. However, to take advantage of operations like 'skip' or 'replay' and eliminate the delivery of some chunks, we need to use small chunks.

To make the problem size tractable even with small chunks, we *cluster* chunks into groups. We then solve the MIP instance based on these groups and place the groups as dictated by the output of the MIP. As a result, all the chunks within a cluster are placed together. One approach we attempted is to retain the difference in popularity by clustering chunks of the same access frequency. This approach, however, has an important drawback in that it results in unbalanced cluster sizes — some clusters being very big (e.g., clusters with chunks having very few accesses) and some other clusters being very small. For example, with our data, we noticed that the size of the largest cluster was $\sim$130000 times that of the smallest cluster. The size of these large clusters makes it very hard to place them in any VHO, thereby defeating the very purpose of MIP-based placement.

We break this using a simple heuristic. We first sort all chunks based on the number of accesses during the training period in decreasing order. Then, we cluster groups of n chunks, so that the first cluster has the n most popular chunks, and the second cluster has the next n popular chunks, and so on. The downside of this approach is that it is an approximation and can potentially result in more copies of certain chunks than needed (because of the additive effect of the demand of each chunk). However, by limiting the size of a cluster to a small number, we can minimize this effect, while we still can solve the MIP quickly.

4.2 Experimental Evaluation

We evaluate the performance of chunk and prefix-based placement through simulation experiments using data from a nationally deployed VoD service. We make the following high level observations from our results:

- Both chunk- and prefix-based placement significantly reduce the amount of data transferred.
- We get most of our reduction from user abandoning videos.
- Despite the availability of stream control operations like skip, most of today's viewing is predominantly sequential.

In the rest of this section, we describe our experiment setup and provide details on each of these results.

4.2.1 Experiment Setup

We use a custom event-driven simulator that implements the network between the VHOs and performs trace-driven simulations. We assume that each VHO has a certain amount of disk, which is partitioned into two parts. One part stores the videos assigned to the VHO (i.e., not eligible for replacement), and the other part is used as a dynamic cache (i.e., LRU). To protect proprietary information, we use normalized disk sizes relative to the space needed to store the entire video library. For instance, when the total disk space across all VHOs is 2x the space needed to store the library, each VHO has disk space equaling ∼3% (2/59) of the total library space. This 3% includes the fixed portion of the disk as well as the dynamic cache. We use a small cache of 5% of the space at each VHO with our MIP-based placement schemes to accommodate for errors in demand estimation and to handle flash crowds.

Upon a user request, if the video segment is available locally (either in the pre-assigned portion or dynamic cache portion), the VHO simply handles the transfer locally without consuming network bandwidth. If the video segment is not available locally, the VHO fetches the segment from a remote VHO that stores this segment in its cache. We assume the existence of a directory that keeps track of what is stored in each VHO. With a dynamic cache, the set of video segments stored at a VHO constantly changes over time. For this, we assume a perfect directory that always finds the closest VHO with a local copy. Note that this is the best case scenario for those dynamic cache strategies (i.e., LRU). For our MIP-based strategy, the MIP solution guides from which VHO to fetch the video (i.e., x variables in the formulation in Section 4.1).

We use a network modeled from a deployed IPTV network for our experiments. This network has 59 nodes (i.e., VHOs) with 70+ bi-directional links of equal bandwidth connecting the different VHOs. We vary the bandwidth to understand the trade-offs between disk capacity and bandwidth. In a given experiment, we assume that all the VHOs have the same disk size. However, we vary the disk size value in different experiments to understand the performance tradeoff between disk and link bandwidth capacity. We set the streaming rate of all videos to 2 Mbps (Thus, one second of video needs about 256 KB of disk).

We use the data trace described in Section 3.2 to simulate video requests by users. The trace data has information about the stream control operations like skip, fast-forward, replay, etc. that are performed by users. In our experiments, we translate these stream control operations to accesses to specific segments of the video: In the case of chunk, it is the specific chunk requested, while with prefixes it is either the prefix or the suffix. When a user performs a skip, we treat the corresponding portion of the video as *not* accessed. However, we consider that all the segments are accessed during a fast-forward operation because the video is played out during the fast-forward operation albeit at a faster rate. Note that if a request from a user results in transferring data from a remote location, we *do not* stop the transfer prematurely even if the user does so. This allows us to cache the entire video segment and serve it locally when it is requested again.

Stream control operations like skip typically move the user about 30 seconds into the stream. To take advantage of these operations and not deliver chunks that are skipped, we need to make use of small chunks. However, as mentioned in Section 4.1, using small chunks significantly increases the time to solve the MIP instance. We addressed this by clustering a number of chunks. In our experiments, we use 10-second chunks and a cluster size of 100; we observed that this size gave us a good balance between the time taken to solve the MIP and the approximation introduced in the placement solution. Similarly for prefixes, we experimented with various prefix sizes and determined that a prefix size of 30% gave the best results, which is the prefix size used in our experiments.

In all our experiments, we are primarily interested in the amount of data stored and transferred among the VHOs; we do not focus on the transfer from a VHO to the end user. That said, we want to ensure that user experience is not affected in that our approach. Conceptually our approach is similar to what P2P systems do; however, their performance is affected because of lack of capacity at the serving peer. By taking available capacity into account as part of the MIP, we ensure that the solution *guarantees* sufficient capacity to serve each user request in time.

Comparison with LRU-based Caching: Previous work showed that LRU-based caching outperforms popularity based placement [Applegate *et al.*, 2010; Wu and Li, 2009]. As a result, we compare the performance of our placement scheme with LRU-based caches. To have an apples-apples comparison with the MIP-based placement, we assume that each VHO has a disk space equal to 3% of the library size. For LRU-cache based approach, we place one copy of every video at a random location and allow the dynamics of the requests to cache copies at the other VHOs. On average, a VHO uses around 50% of disk space for the dynamic cache. We also use 10-second chunks and 30% prefixes when experimenting with chunks and prefixes respectively. With the LRU-cache, we assume an idealized setting where each VHO knows the nearest location with a copy of the requested object.

4.2.2 Working Set Size

The working set at a VHO is defined as the number of unique videos that are requested within a time window. The significance of the working set is that it determines how effective caching is going to be. There will be a large number of cache evictions if the working set size is larger than the cache size. In Figure 4.1, we plot the working set size (in terms of bytes compared to the library) at each of the 59 VHOs during the peak hour on a Saturday. We consider three cases where we deal with (1) full videos, (2) prefixes and suffixes, and (3) 10-second chunks. For the prefix/suffix case, we set the prefix to the initial 30% of a video. (We elaborate further on determining prefix length in Section 4.2.7.)

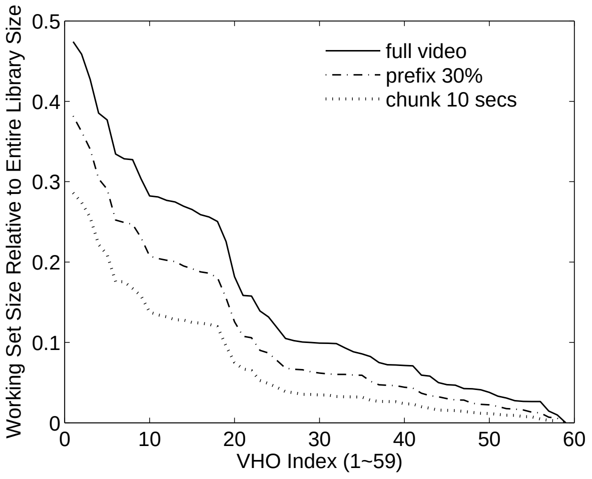


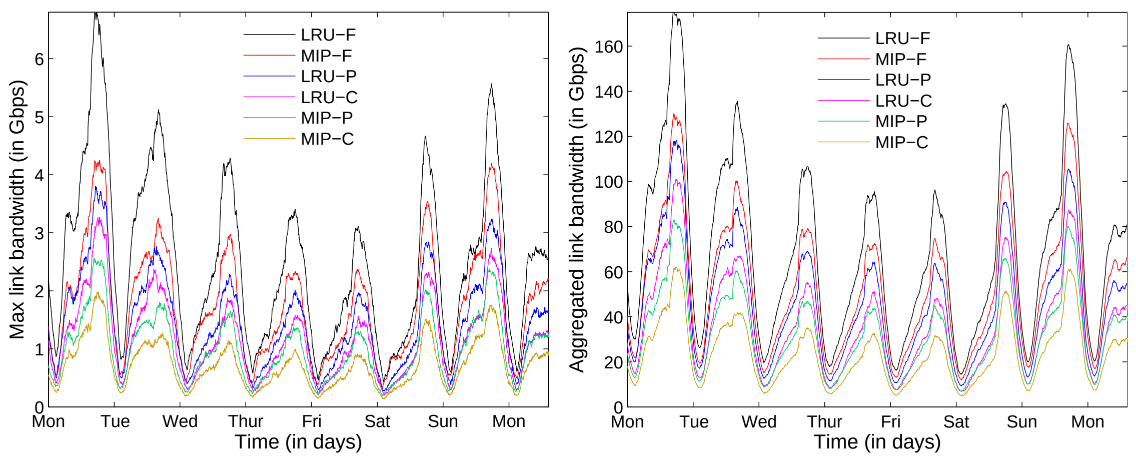
Figure 4.1: Fraction of working set size to the entire library size at each VHO.

We observe that the working set size at a VHO can be as high as about 48% of the library size. The top 20 VHOs have a working set of around 20% or more. With prefixes, the relative working set size can be as large as 39%, and the top 20 VHOs have a working set size of around 15% or larger. Although using chunks further reduces the relative working set size, but is still quite large (as high as 30%). This result shows that for caching to be successful, the cache space at each VHO has to be quite large.

4.2.3 Maximum and Aggregate Link Bandwidth

We quantify the benefit of placing segments of the video instead of the full video by comparing the peak and aggregate bandwidth needed for each of the approaches. Recall that our data was spread over 16 days. We use the first 9 days of data to predict the demand for videos (or segments) with the MIP-based placement, and to warm up the caches with LRU. At the end of 9 days, we solve the MIP formulation and place content accordingly. We then simulate for the last 7 days and measure the performance of both LRU and MIP-based placement. In [Applegate *et al.*, 2010] we addressed in detail the practical considerations for the algorithm design and parameter selection in demand estimation, time-varying demand, and placement update frequency.

Figure 4.2(a) shows the maximum link bandwidth used across all links in the network



(a) Maximum link bandwidth (b) Aggregate link bandwidth

Figure 4.2: Link bandwidth utilized across all links, computed every 5 minutes (the ordering of the curves is the same as in the legend).

measured in 5-minute windows over the last 7 days for full videos (MIP-F, LRU-F), 30% prefixes (MIP-P, LRU-P), and 10-second chunks (MIP-C, LRU-C). For the MIP-based schemes the peak bandwidth needed goes down from about 4.2 Gbps with MIP-F to 2.6 Gbps with MIP-P (a 38% reduction). MIP-C performs the best with a peak of only 2.0 Gbps. In Figure 4.2(b), we plot the total bytes transferred across all links in the network, averaged over 5-minute windows. MIP-F results in a maximum of 130 Gbps of aggregate traffic in the network while MIP-P has a maximum of 83 Gbps (36% reduction). At 61 Gbps, MIP-C again results in the least amount of data transferred.

Furthermore, we see that the MIP-based placement outperforms LRU-based caches. For example, MIP-F requires a maximum of 4.2 Gbps bandwidth, while LRU caching of full videos (LRU-F) requires 6.9 Gbps. We observe similar trends when we compare MIP-P and MIP-C with corresponding segment-based caching schemes LRU-P and LRU-C, respectively (e.g., 33% reduction from LRU-P to MIP-P). More interestingly, we see that LRU-C requires *more* bandwidth than even MIP-P. Between the worst case (LRU-F) and the best case (MIP-C), we achieve over a factor of 3 improvement in max. and aggregate network bandwidth.

These two results confirm our hypothesis: *Taking advantage of user behavior and placing*

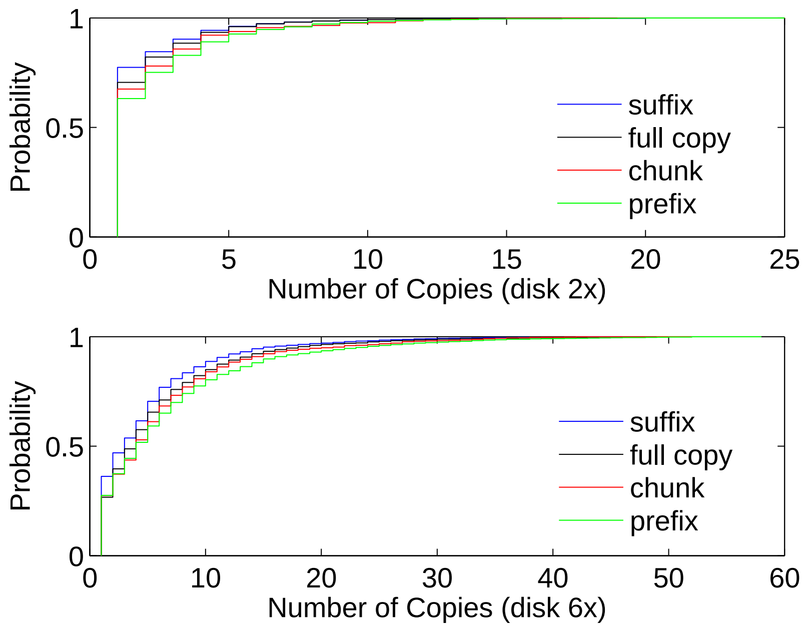


Figure 4.3: Cumulative distribution of the number of replicas created by the MIP based approach when the available disk is 2x and 6x library sizes.

segments provides significant benefit. First, by using prefixes and suffixes, we take advantage of the fact that people do not watch full videos. We get further benefits when using chunks because of two factors: (a) using chunks, we need not transfer unwatched portions of even prefixes or suffixes when users abandon a video, and (b) we can take advantage of fine-grained stream control such as SKIP operations to eliminate transfer of data within a prefix and a suffix.

4.2.4 Number of Replicas

In this subsection, we examine the number of copies for each *object* across all the VHOs, where an object is a full video in MIP-F or a video segment in MIP-P and MIP-C. In Figure 4.3, we show the cumulative distribution when the aggregate disk space in the system space in the system is 2x and 6x the library size. We observe that prefixes tend to be copied in more VHOs than suffixes or full videos. For instance, with the disk space of 2x, there is only one copy for 63% of prefixes, 71% of full videos, and 78% of suffixes. On average, there are 2.2

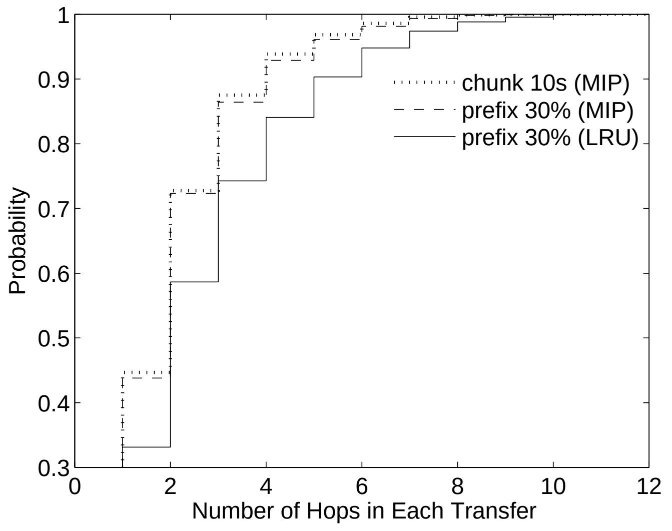


Figure 4.4: Cumulative distribution of the number of hops traversed when transferring content (segments) remotely.

copies for a prefix, 1.7 for a suffix, and 1.8 for a full video. We also observe that by flexibly placing fine-grained chunks, MIP-C replicates medium-popularity videos in more locations. In general, when videos are split into multiple segments, we can potentially "pack" the segments better into the available space. This result demonstrates that the MIP-based solution leverages the opportunity and intelligently replicates popular video segments in more locations.

More interestingly, these two results illustrate how the MIP solution adjusts to the available space by placing the objects better. When the total disk space is only 2x the library size, even the most popular object is not replicated in all 59 VHOs, and more than 60% of all objects have only one replica. When the disk space increases to 6x the library size, we see that the placement solution takes advantage of this extra space and places a copy of the popular objects in every location. We also see a greater spread in the number of copies of each object with 50% of the objects having anywhere from 2 to 15 replicas. This result shows an important aspect of VoD service and the need for more intelligent placement strategies such as our MIP based scheme: with videos, despite the skew in popularity, there is a large set that are requested enough number of times and cannot be ignored, and naive schemes (e.g., replicating top K videos in all locations) are unlikely to result in good performance [Applegate *et al.*, 2010].

4.2.5 Hops Traversed

In this subsection, we study the effectiveness of the placement in terms of the number of hops traversed to fetch video segments from remote VHOs, whenever a remote transfer is required. The disk space allocated to all the scenarios is 2x the library size. We plot the results in Figure 4.4. The figure shows that the MIP-based schemes show a similar hop count distribution and consistently result in smaller number of hops than the caching based approaches.

Specifically, the average of MIP-P is about 2.1 hops, where less than 10% of remote transfers take more than 4 hops. In contrast, the caching based approach (LRU-P) uses remote transfers longer than 4 hops in about 20% of the cases, and hence its average hop count is larger (2.8 hops). Recall that for the caching solutions we assume an idealized scenario where a perfect directory server can tell us the nearest location storing the object. Thus, the caching approaches have the most up-to-date view of which location is storing what content. This, along with the result in Figure 4.3, shows that the MIP-based solution does an effective job of placing objects close to where they are needed.

4.2.6 Disk and Bandwidth Tradeoff

We study the tradeoff between disk and link bandwidth capacity. Specifically, we vary the aggregate VHO disk size in the network between 2x the library size and 20x and measure the minimum amount of bandwidth required for each of those disk size. We present the result in Figure 4.5. We observe that in all cases, the amount of bandwidth required decreases as the total disk space in the system increases. However, by taking user behavior into account, segment-based approaches (MIP-P, MIP-C, LRU-P, and LRU-C) consistently require less bandwidth than storing full videos (MIP-F and LRU-F). This is true even when the disk space in the system is 20x the library size, where each VHO has enough space to store around one third of the entire library. We also observe that the MIP-based solutions outperform caching-based solutions, especially when the disk space is scarce. This shows

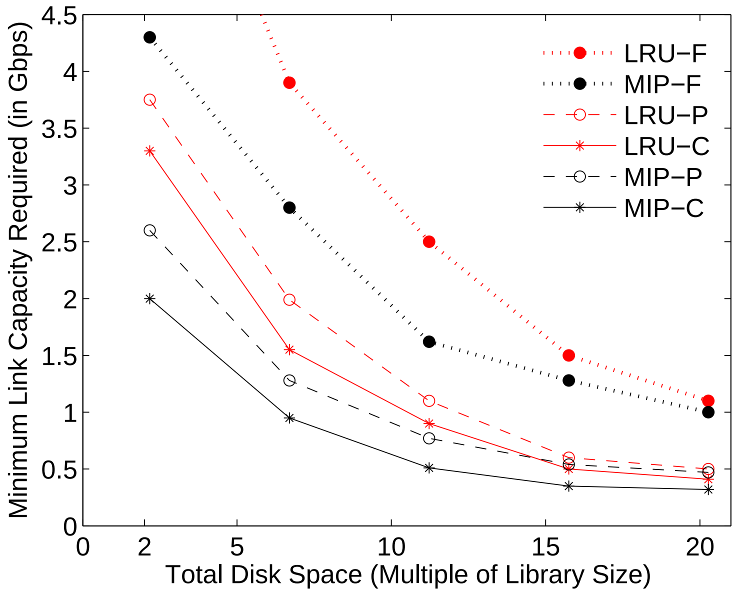


Figure 4.5: Feasibility region of Placement-based and Caching-based solutions (the ordering of the curves is the same as in the legend).

the importance of having storage space having at least the “working set size” with caching schemes.

4.2.7 Effect of Segment Size

We compare the effects of different chunk sizes in Table 4.2 by comparing the total bytes transferred from remote locations for MIP-P and LRU-P. As the table shows, the total bytes transferred initially decreases with increasing prefix size, but eventually starts increasing. The least amount of data is transferred when the prefix size is 30%. We also experimented with per-video prefix values. However, our results showed very modest improvements compared to a fixed prefix size.

Prefix Size	10%	20%	30%	40%	50%	100%
LRU	1689	1497	1451	1496	1565	2203
MIP	1425	1274	1220	1254	1284	1832

Table 4.2: Total bytes transferred from remote locations for different prefix sizes (in TB)

We understand the effect of chunk size by calculating the total bytes the system serves (both locally and remotely) for different segment sizes (that is, $\sum_{j,m} a_j^m s^m$ using the notation in Section 4.1.) We report the values in Table 4.3. We observe that with 10-second chunks, the system needs to serve only 48% of total bytes served when using full videos (3563 vs. 7483TB). We also find that the amount of saving due to SKIP operations is 101TB for the 10-second chunks (about 3% of the total reduction of 3920TB). We observe that the difference between 1-minute chunks and 10-second chunks is modest. This result indicates that the majority of gain with the chunk-based placement comes from being able to stop video transfer shortly after a user aborts a session. We postulate that this is because of limitations of existing VoD interfaces and that this may change when DVD-like navigation becomes possible.

	full	prefix	chunk	
	video	30%	10sec	1min
total	7483	4875	3563	3664

Table 4.3: Total served bytes for different segment sizes (in TB)

4.2.8 Cache Dynamics

Since in all approaches, VHOs make use of caches (small for MIP and large for the LRU caching scheme), we looked at how placement affects the performance of these caches. We study the number of remote transfers (which are equivalent of cache misses), the number of cache evictions (replacements), and the number of failed cache insertions. Instead of counting the number of occurrences, we study these factors in terms of TB of data to account for the heterogeneity in object sizes. Table 4.4 shows the results of this study.

	LRU-F	LRU-P	LRU-C	MIP-F	MIP-P	MIP-C
Remote Transfer	2115	1451	1147	1839	1220	921
Cache Replacement	177	365	1104	34	109	918
Failed Insertions	1841	1081	0	1805	1107	0

Table 4.4: Cache dynamics for the different schemes over the last 7 days (in TB).

As expected, we see in Table 4.4 that the MIP-based, segment placement schemes result in the least amount of data transferred. More interesting are the results of cache replacements and failed insertions. A failed insertion occurs when there is insufficient space in the cache and none of the existing objects can be replaced because they are all being used. It is of particular significance with videos because it means that all the work done in transferring this large object will have to be repeated again. Table 4.4 shows that despite having 1/10th the cache size, MIP-P experiences 70% less replacements and approximately the same amount of failed insertions compared to LRU-P. This shows that the cache in MIP-P is being used effectively; objects in the cache are used and hence cannot be replaced. MIP-C and LRU-C do not have failed insertions because all objects are small and of the same size. However, this also results in a lot of cache churn.

Chapter 5

User Abandonment and its Impact on P2P VoD

5.1 Abandonment and P2P VoD

Recent studies [Li *et al.*, 2011; Yin *et al.*, 2009a] have shown that users abandon videos before viewing them in their entirety. Our results (in Section 5.3) will show that this abandonment has a significant effect on the performance of P2P VoD streaming systems. In this section, we re-evaluate some of the key policies and design decisions for P2P VoD by taking abandonment into account. Using the right chunk selection policy and peer selection policy, we balance the need to download the video fast enough to minimize interruptions and startup delay while also minimizing wastage of network resources.

5.1.1 Abandonment in Trace Data

Using traces from a large scale VoD service provider, we demonstrate that user abandonment indeed occurs in Chapter 3. We show that the video watching duration distribution depends on various aspects such as video length and popularity. In this chapter, we also use trace data from the same commercial VoD service, but the data covers a different period (fifteen day period in 2010) and has millions of requests. The trace includes information about

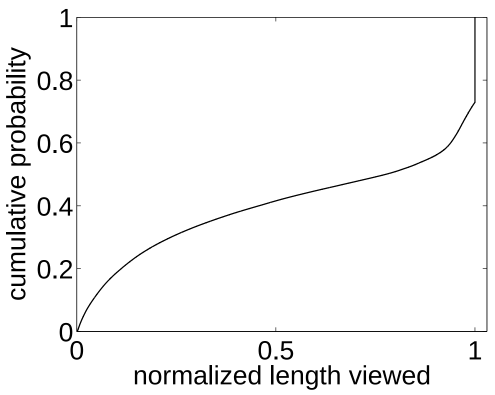


Figure 5.1: Cumulative distribution of normalized length viewed of all requests in trace data

users' interactive operations (e.g, FastForward, Skip, Rewind, etc.), and we calculate the viewed length of each user taking into account of these specific operations (more details described in Section 5.3.1).

In Figure 5.1, we show the extent of user abandonment by plotting the cumulative distribution (CDF) of normalized length viewed (NLV). To compare abandonment for videos of different lengths we divide each viewed length by the original duration of the video. The figure indicates that only 26% of the sessions consumed the corresponding videos fully. Thus, the assumption that each user views the entire video and therefore downloads the entire video file (as assumed by many existing P2P works), is not appropriate when considering real-world user behaviors.

5.1.2 Chunk Selection Policy

The chunk selection policy determines the order in which a peer downloads chunks. File sharing systems have traditionally used Rarest-First (RF) as the chunk selection policy [Cohen, 2003; Legout *et al.*, 2006; Bharambe *et al.*, 2006b]. RF has several desirable properties including distributing rare chunks among peers and allowing the seed to quickly offload chunks. RF however is inherently unsuitable for streaming systems which require chunks to arrive in order. Instead, a chunk selection policy that attempts to get chunks close to playback is a more natural fit for video streaming. In fact, a significant amount of previous work [Zhou *et al.*, 2007; Fan *et al.*, 2010; Borghol *et al.*, 2010; Carlsson and Eager, 2007; Vlavianos *et al.*, 2006; Shah and Paris, 2007] has shown that the combination of EF and RF (EF+RF) incorporates the strengths of each policy and results in the best playback continuity with P2P VoD.

However, none of these consider the effect of abandonment by users. We observe that propagating rare chunks, usually from the latter half of a video, is counter-productive and *wasteful* when abandonment is taken into consideration. Instead, we could use that bandwidth to transfer chunks that are needed immediately. As a result, we adopt EF as our chunk selection strategy. This allows us to download chunks in order and minimize the possibility of interruption. It also allows us to control the rate at which chunks are downloaded to minimize wastage. Our experimental results in Section 5.3.5 confirm that EF outperforms EF+RF in the presence of abandonment.

Note that we use EF despite previous work reports that the use of EF can lead to "throughput collapse" when peers possess a similar collection of chunks [Fan *et al.*, 2010]. We argue that this throughput collapse is a side-effect of using EF with Tit-for-Tat as the peer selection policy.

5.1.3 Peer Selection Policy

The peer selection policy determines the subset of requests that are served by a peer upon receiving requests. Unlike chunk selection where multiple options are used in practice, most systems use TFT as the peer selection policy. TFT works by forcing peers to upload data in order to download content [Cohen, 2003; Bharambe *et al.*, 2006b]. This allows peers to disseminate content quickly to high bandwidth peers and eliminates free-riding. However, it requires that peers have content to exchange with each other. This, however, has the unfortunate side effect of introducing interdependency between chunk- and peer selection policies. Since peers have to upload some data in order to be able to download (setting aside considerations of optimistic unchoking used by TFT), peers need to have a diverse set of chunks. This is not an issue with RF as it is designed to create such diversity. With EF, however, peers at different points of their playback will not have content of mutual interest to exchange with each other. In addition, there is growing realization that there are inefficiencies due to TFT [Piatek *et al.*, 2010; Huguenin *et al.*, 2010; D'Acunto *et al.*, 2010; Yang *et al.*, 2010; Wen *et al.*, 2011] in streaming systems, particularly with regard to playback interruptions.

For this reason, we complement EF by choosing the peer with the "Earliest-deadline". To satisfy a viewer's uninterrupted playback experience, each chunk must be delivered to the viewer prior to its deadline. In our *Earliest-Deadline* (ED) peer selection scheme, a requesting peer specifies a chunk and its deadline with each request. Then, a potential provider (seed or peer) receiving chunk requests from multiple connected peers during a certain interval chooses to serve the peer with the earliest deadline (with ties broken at random).

Unlike TFT, a peer does not choke another in ED. This brings up new protocol aspects that we need to address. First, as a provider, a peer may receive upload requests from all of its connected peers. In order to ensure that the per-chunk upload speed does not become too small, we limit the number of concurrent uploads from a peer to other peers. Second, as it is not choked, a downloading peer can have a large number of parallel downloads. Since all the downloads may share a single downstream link to the peer, that link may become the bottleneck. If the number of parallel downloads becomes large, the per-chunk download rate decreases. This results in longer start-up delays and frequent interruptions. To address this, each peer adjusts its maximum number of parallel downloads dynamically, based on the availability of its download bandwidth. Peers can increase the number of parallel downloads until they use up their download capacity. They stop adding streams when any additional download has the potential to decrease the speed of the ongoing downloads.

By serving peers with the most urgent need, ED focuses on 'fairness' of each peer's streaming performance. While this notion of fairness is certainly a 'qualitative' one, we show in Section 5.3.3 that ED performs substantially better than TFT with respect to quantitative metrics such as interruption time.

5.1.4 Handling Free Riding

Free riders in P2P systems can significantly impact the overall system performance and introduce unfairness. TFT was designed specifically to prevent such free riding. However, as stated earlier, TFT introduces dependency on chunk selection that is incompatible with P2P streaming. Hybrid chunk selection policies seek a middle ground. There have even been efforts to change seed policies [Carlsson *et al.*, 2009] or to cluster peers at similar playback points and introduce diversity within these clusters [Huguenin *et al.*, 2010]. Unfortunately, abandonment significantly diminishes the effectiveness of such approaches, as we will see in Section 5.3.

Our approach in this chapter is to deviate from TFT and instead use ED. While ED does not guarantee against free riders, ED offers better performance in terms of streaming and quality-of-experience (QoE) compared to TFT. The decoupling of peer selection from the chunk selection policy allows us to overcome inefficiencies due to TFT [Piatek *et al.*, 2010; Huguenin *et al.*, 2010] in deployments that do not worry about free riding (e.g., managed content delivery [Maggs, 2012]). In scenarios where eliminating free-riding is still important, we can adopt approaches like Contracts [Piatek *et al.*, 2010] or iPASS [Liang *et al.*, 2010], appropriately modified, for P2P VoD, to incentivize peers to share content. We are exploring these strategies as part of our ongoing work.

5.1.5 Using "Partial" Seeds

Using ED as the peer selection policy allows us to eliminate the artificial bottlenecks that arise from using TFT. It also allows peers to make progress by favoring chunks with the smallest deadline. However, it does not eliminate the fact that seeds can still be overloaded, and thus become the bottleneck for some peers. We overcome this by taking advantage of the content that peers have already downloaded. We take advantage of the fact that with abandonment, there is a period between consecutive videos (or when the user is performing other activities) that the node remains connected to the system even though it is not actively viewing a video.

Consequently, we assume that the abandoning peer becomes a *partial seed* and continues to stay in the system and shares the portion it has already downloaded. This is akin to a seed with the entire video, except that this node only has the partial video. Partial seeds can offload serving the initial parts of the video (that are presumably requested more frequently), allowing the seed to serve the later but rare portions to the few users that remain to watch the video fully. Our analysis in Section 5.2 shows that by having partial seeds stay longer in the system, we can improve performance significantly.

5.1.6 Reducing Wastage by Limiting Playback Lookahead

The primary objective of the chunk selection policies is to sustain a sufficient rate so that a viewer does not experience interruptions. Our combination of EF and ED with partial seeds allows us to achieve a rate comparable, if not better, than TFT (Figure 5.16). However, unlike the ED+EF policy, TFT with RF-based chunk selection (e.g., the hybrid policy EF+RF) generates wastage by unnecessarily propagating later chunks that are not likely to be watched when a viewer abandons the video (Section 5.3.5). We investigate which combination of chunk selection and peer selection policies generates the smallest amount of the wastage of network bandwidth.

In order to further limit the wastage with ED+EF, we adopt a playback lookahead window (PLA) that is measured in number of chunks. The PLA restricts excessive downloads of chunks beyond the current playback point. Specifically, when a peer has downloaded a predefined number (equal to the PLA window) of consecutive chunks ahead of its current playback point, it stops requesting further downloads until the playback progresses and the window 'opens' up. The window also moves forward as the playback progresses. By adjusting the PLA window size, ED+EF greatly reduces the bandwidth wastage without hurting playback continuity.

We note that this method of limiting the delivered rate has been widely used for server-based video streaming protocols such as HTTP streaming [Rao *et al.*, 2011], progressive download [Ghobadi *et al.*, 2012], and adaptive bitrate streaming [Akhshabi *et al.*, 2011]. Our results show that P2P systems can also achieve substantial reduction in wastage by implementing a rate limiting capability in the form of a playback lookahead.

5.2 Abandonment Analysis

In this section we analyze how abandonment affects swarm population and useful download rates for EF and RF schemes. Aalto et al. [Aalto *et al.*, 2011] analyze a viewer's aban-

Parameter	Definition
B	File byte size
M	Number of chunks of file
U	Max. upload connections
D	Max. download connections
C	Throughput per connection
λ	Leecher arrival rate
$1/\mu_1$	Normal seed staying time
$1/\mu_2$	Partial seed staying time
$x(t)$	Number of leechers at time t
$y(t)$	Number of normal seeds at time t
$z(t)$	Number of partial seeds at time t
β	Fraction of leechers converted to normal seeds
ρ	Fraction of the file a partial seed has on average
δ	Prob. that a partial seed has chunks available for a leecher
γ	Leecher download rate
$\bar{\gamma}$	Leecher *useful* download rate

Table 5.1: Parameters and Definition for Analysis

donment in P2P VoD as a stochastic queueing model, where leechers abort and leave the swarm instantly. Thus, they ignore the concept of partial seeds. Our analysis builds on the work by Qiu et al. [Qiu and Srikant, 2004] and Parvez et al. [Parvez *et al.*, 2008] but also takes abandonment into account.

We consider a single swarm with leechers, normal seeds, and partial seeds, where we denote their count at time t by $x(t), y(t), z(t)$ respectively (or x, y, z for simplicity). A leecher can have D concurrent download connections and U simultaneous upload connections, and each connection has a throughput of C. Leechers enter the system at rate λ and attempt to play back a video of B byte size that has M chunks. The leecher views the video fully and becomes a normal seed with probability β, while with probability $1-\beta$, it abandons its video and becomes a partial seed. We consider a demand-driven system where $xD > (x+y+z)U$

(upload capacity constrained). We summarize the notations in Table 5.1.

5.2.1 Swarm Population

RF swarm size: The total system upload capacity that is useful to a random leecher (i.e., system goodput) is $(x + y + \delta z)UC$ where δ indicates the fraction of partial seeds that can upload a desired chunk for a leecher. With RF, we assume partial seeds can always upload to leechers (i.e., $\delta = 1$), whereas for EF, δ depends on the distribution of chunks at a partial seed.

For the system to become the steady state, the rate of data loss due to the departures of normal seeds and partial seeds should be equal to the system goodput [Benbadis *et al.*, 2008] as follows,

$$\{\beta + (1-\beta)\rho\}B\lambda = (\bar{x} + \bar{y} + \delta\bar{z})UC \tag{5.1}$$

where $\beta B\lambda$ is the loss rate due to normal seed departures and $(1-\beta)\rho B\lambda$ is the one due to partial seed departures. From Equation (5.1) with $\delta = 1$, the total swarm population of RF is

$$\bar{x} + \bar{y} + \bar{z} = \frac{\{\beta + (1-\beta)\rho\}B\lambda}{UC} \tag{5.2}$$

which is independent of the normal or partial seed staying time.

Since our model assumes that the system is upload capacity constrained, for the model validity we need to compute a condition on the staying times of partial seeds and normal seeds. Using Little's law, $\bar{y}$ and $\bar{z}$ in the steady state are

$$\bar{y} = \beta\frac{\lambda}{\mu_1} \tag{5.3}$$

$$\bar{z} = (1-\beta)\frac{\lambda}{\mu_2} \tag{5.4}$$

From Equations (5.2), (5.3), and (5.4), $\bar{x}$ is

$$\bar{x} = \lambda\{\frac{\{\beta + (1-\beta)\rho\}B}{UC} - \frac{\beta}{\mu_1} - \frac{1-\beta}{\mu_2}\} \tag{5.5}$$

Therefore, to satisfy $\bar{x} > 0$, the model requires the following condition on μ_1 and μ_2:

$$\frac{\beta}{\mu_1} + \frac{1-\beta}{\mu_2} < \frac{\{\beta + (1-\beta)\rho\}B}{UC} \tag{5.6}$$

which indicates very large partial or normal seed staying time can violate our assumption of the upload capacity constraint.

EF swarm size: Following [Parvez *et al.*, 2008], from the viewpoint of any given peer A, there are *younger* peers who arrived after A and *older* peers who arrived before A. With EF A can only download from its older peers (with more chunks) and can only provide content to younger peers. An uploader that receives more than U requests chooses to serve U requests at random and rejects the rest.

Consider a peer that has been in the system for time t_m. The probability that the peer is successful in obtaining a download connection for its next desired chunk is defined as: $p(t_m) = \frac{\tilde{U}(t_m)}{\tilde{D}(t_m)}$, where $\tilde{U}(t_m)$ and $\tilde{D}(t_m)$ are the connection supply and demand at time t_m. A peer of age t_m requests a download connection from older peers ($t > t_m$).The total number of possible upload connections available for this peer is $\tilde{U}(t_m) = (x + y + \delta z - \lambda t_m)U$. For computing $\tilde{D}(t_m)$, we first note that the total number of download requests in the system is xD and that peers with more chunks (including seeds and partial seeds) receive higher demand. In [Parvez *et al.*, 2008] $\tilde{D}(t_m)$ is indirectly calculated by finding the total number of download requests handled by peers younger than t_m and subtracting it from xD, and is approximated as $\tilde{D}(t_m) = \frac{xD}{\alpha}$ where α depends on the system parameters. However, we approximate α as a constant, and its range is $[1.09, 1.25]$ for typical scenarios in [Parvez *et al.*, 2008] (we will provide the upper bound on α to achieve the steady state system in Inequality (5.11)).

Using Little's law, we can derive the average downloading rate: $\gamma_{EF} = \frac{1}{T}\int_0^T Dp(t)Cdt = \alpha UC(y/x + \delta z/x + 1/2)$, where T is session duration. Similar to Equation (5.1), the rate of data loss due to leaving seeds should equal the system's goodput in the steady state as follows,

$$\{\beta + (1-\beta)\rho\}B\lambda = (\frac{\bar{x}}{2} + \bar{y} + \delta\bar{z})\alpha UC \tag{5.7}$$

where the right hand side comes from $\gamma_{EF}x$. $\bar{y}$ and $\bar{z}$ are independent of chunk selection schemes (i.e., the same as Equation (5.3) and (5.4), respectively). From Equations (5.3), (5.4), and (5.7) we obtain $\bar{x}$:

$$\bar{x} \;=\; 2\lambda\{\frac{\{\beta + (1-\beta)\rho\}B}{\alpha UC} - \frac{\beta}{\mu_1} - \frac{(1-\beta)\delta}{\mu_2}\} \tag{5.8}$$

Since we assume $\bar{x} > 0$, similar to Inequality (5.6) we have the following condition on μ_1 and μ_2 for the EF scheme:

$$\frac{\beta}{\mu_1} + \frac{(1-\beta)\delta}{\mu_2} < \frac{\{\beta + (1-\beta)\rho\}B}{\alpha UC} \tag{5.9}$$

The total EF swarm population is

$$\bar{x} + \bar{y} + \bar{z} = \lambda\{\frac{2\{\beta + (1-\beta)\rho\}B}{\alpha UC} - \frac{\beta}{\mu_1} - \frac{(2\delta - 1)(1-\beta)}{\mu_2}\} \tag{5.10}$$

Unlike with RF policy, the swarm size can increase depending on how long the partial seeds and the normal seeds reside in the system.

Note that we should limit α such that $(\bar{x} + \bar{y} + \bar{z})UC \geq \{\beta + (1-\beta)\rho\}B\lambda$ since the left hand side indicates the aggregate upload bandwidth of all peers in the system and should be equal to or larger than the system's goodput $(\frac{\bar{x}}{2} + \bar{y} + \delta\bar{z})\alpha UC$ in Equation (5.7). Therefore, using Equations (5.10), α has the following upper bound:

$$\alpha < \frac{2\{\beta + (1-\beta)\rho\}B\mu_1\mu_2}{\{\beta + (1-\beta)\rho\}B\mu_1\mu_2 + \{(2\delta - 1)(1-\beta)\mu_1 + \beta\mu_2\}UC} \tag{5.11}$$

5.2.2 Useful Download Rates

We define that a chunk download is useful only if the downloader has already downloaded all other sequentially earlier chunks. We now compare the useful download rates to show that EF with abandonment provides quicker download compared to the RF policy with abandonment.

RF without abandonment: We can get $p(t) = \frac{\tilde{U}(t)}{\tilde{D}(t)} = \frac{(x+y)U}{xD}$ with RF without abandonment. The download rate is

$$\gamma_{RF} = \frac{1}{T}\int_0^T Dp(t)Cdt = UC(1 + \frac{y}{x})$$

RF with abandonment: Similarly, $p(t) = \frac{(x+y+z)U}{xD}$ (assuming $\delta = 1$ for RF), and

$$\gamma_{RF} = \frac{1}{T}\int_0^T Dp(t)Cdt = UC(1 + \frac{y}{x} + \frac{z}{x})$$

The useful download rate for RF with abandonment is obtained by scaling γ_{RF} with the probability of useful chunks that have been downloaded up to time t. This scaling factor is shown to be $\frac{1}{M-k+1}$ [Fan *et al.*, 2010]. Therefore:

$$\bar{\gamma}_{RF} = \gamma_{RF}\frac{1}{M-k+1}$$

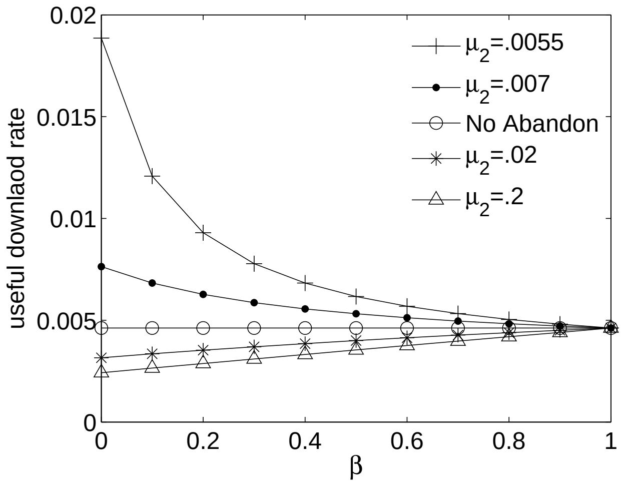


Figure 5.2: Useful download rate of EF vs β ($\mu_1 = 0.01$)

where k is the number of chunks that have been downloaded so far for a file with a total number of M chunks.

EF without abandonment: In EF, since the chunks are sequentially downloaded, all chunks downloaded are useful.

$$\bar{\gamma}_{EF} = \gamma_{EF} = \alpha UC(\frac{y}{x} + \frac{1}{2}) \tag{5.12}$$

EF with abandonment: Similarly,

$$\bar{\gamma}_{EF} = \gamma_{EF} = \alpha UC(\frac{y}{x} + \frac{\delta z}{x} + \frac{1}{2}) \tag{5.13}$$

Figure 5.2 shows the impact of partial seeds on the useful download rate for EF. Using Equations (5.12) and (5.13), we apply $\alpha = 1.2, \rho = 0.5, 1/\mu_1 = 100, M = 100, U = 4$, and $C = 0.001$ assuming the unit file size $B = 1$. We also set $\delta = 0.5$ assuming the distribution of the number of downloaded chunks at each partial seed is uniformly distributed, and thus on average only half of the partial seeds have chunks at or beyond the point defined by the desired chunk. Note that by Equation (5.9) $\mu_2 > 0.0048$ should be satisfied for the system to remain under the assumption of upload capacity constraint. When partial seeds stay for a short period of time (e.g., $\mu_2 \geq 0.02$), the useful download rate with abandonment is smaller than the case without. As partial seeds continue to reside for extended periods of time (potentially longer than the normal seeds), the useful download rate with abandonment is seen to be significantly higher than without. Thus, in a system with abandonment, a partial seed's staying time significantly influences the useful download rate. In case of $\mu_2 = 0.0055$ or 0.007, the useful download rate decreases with increasing β since normal seed staying time is relatively shorter than partial seed staying time (note $\mu_1 = 0.01$). Comparing EF and RF, as M becomes larger, EF's useful download rate is much higher than for RF (not shown) confirming earlier results.

5.3 Experimental Evaluation

5.3.1 Data Set

To reflect realistic viewing patterns of a large population of users, we collected trace data from a nationally deployed Video-on-Demand service serving millions of customers. While the data collected is available for a period of several years (during which the service has grown to serve several million customers), we examine a "heavy-viewing" period of 15 consecutive days in 2010. We focus on the trace from a single large metropolitan area, totalling approximately 1 million requests. The trace data contains information for each viewing session: an anonymized user ID, user request time, video ID, video length, and the duration viewed. To ensure user privacy, all user data is kept anonymous and was analyzed in aggregate, without the ability to identify each user. The trace also has information about the set of DVD-like operations that the user performed while watching the video in a given session.

We use the duration viewed in the trace as session duration (time elapsed since the user request time) in the simulation, at which point the peer abandons the video. Note that the final playback point of the video in the simulation may be shorter than the session duration (due to startup delay, interruptions, etc.). We view this difference as an indicator of the performance of the system (smaller the better).

To obtain representative results, we repeat the experiment independently with 8 different popular videos that have different lengths across a wide range, from 30 to 150 minutes. They show different abandonment patterns as in Figure 5.3. They also show a clear daily pattern in their request volume. For example, Figure 5.4 shows the number of concurrent sessions

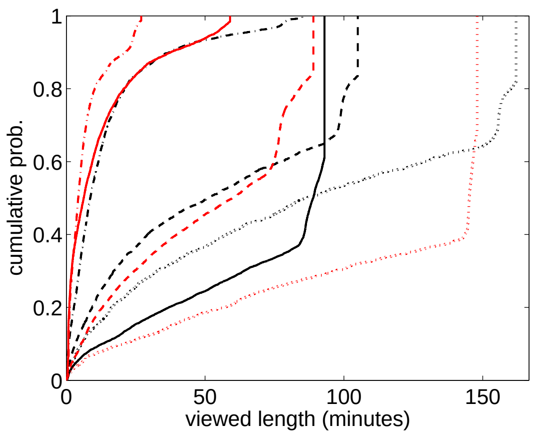


Figure 5.3: CDF of viewed length for 8 videos

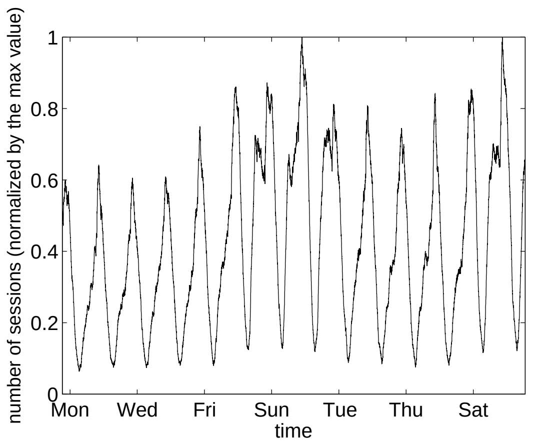


Figure 5.4: Number of concurrent sessions for a random video in the real trace.

(i.e., swarm size) of one of the 8 videos. Note that to protect proprietary information, the Y-axis is normalized by the peak value. Although the absolute request volume varies, the other 7 videos also follow very similar daily patterns. We report the average results obtained from these 8 videos in this section.

5.3.2 Experiment Setup and Assumptions

To evaluate our approach, we use a discrete event-driven BitTorrent simulator [Bharambe *et al.*, 2006a]. It implements peer activity (e.g., joins, leaves, setting up connections with other peers, chunk exchange, etc.) as well as many of the policy mechanisms associated with BitTorrent (RF, TFT, and so on). However, the original simulator was developed to simulate P2P file sharing. Therefore, we enhance the original P2P file sharing simulator for video streaming, so that each peer waits till a playback buffer fills up and then starts playing back as the download progresses. In addition to original TFT, we also experiment with ED, EF, and EF+RF to compare the following set of schemes:

- **TFT+EF:** using EF chunk selection with original TFT
- **TFT+[EF+RF]:** using hybrid of EF and RF with TFT
- **ED+EF:** using EF with ED peer selection

Parameter	Default
Number of initial seeds	1
Upload bandwidth of an initial seed	3 Mbps
Peer download/upload bandwidth	5 Mbps/ 1 Mbps
Max. concurrent initial seed uploads	15
Max. concurrent peer uploads	5
Video bitrate	1 Mbps
Chunk size	10 seconds
Peer arrival rate (synthetic trace only)	$\lambda = 0.05$ (Poisson arrival)
Startup buffer (synthetic trace only)	10 second video
Video length (synthetic trace only)	30 minutes (180 chunks)
Number of sample peers (synthetic trace only)	2000 peers

Table 5.2: Simulation parameters and their default values.

We further enhance EF+RF to reduce the startup delay as follows. Instead of simply selecting EF or RF based on a probability [Vlavianos *et al.*, 2006; Zhou *et al.*, 2007; Fan *et al.*, 2010; Carlsson and Eager, 2007], a peer in the enhanced scheme initially uses EF only, but switches to EF+RF only if it has enough chunks in the playback sequence. In our experiments, if there are 5 or more chunks, a peer uses EF with probability of 0.7 and RF with probability of 0.3.

We use one initial seed that has the complete video to serve to other requesting peers and stays in the swarm throughout the simulation. In our streaming model, we assume that a peer can play back a chunk while it is being downloaded (subject to the startup delay and the appropriate portion being available). However, the peer cannot share the chunk with other peers until it is completely downloaded. We assume that a peer downloads only one chunk at a time from a given uploader. All peers follow the same chunk- and peer-selection policies. The tracker behavior remains unchanged from current BitTorrent systems.

For the trace driven simulation, peers make requests for a video at the time instants specified in the trace. Peers that download the entire video convert to normal seeds while those that abandon the video part-way become partial seeds. Since the trace does not tell

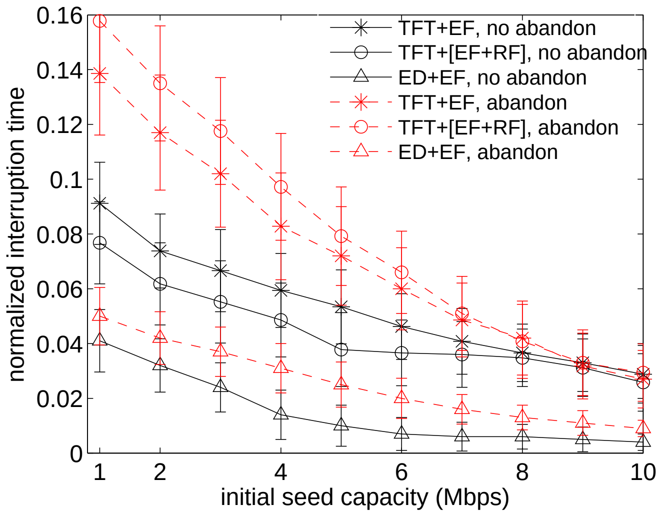


Figure 5.5: Comparing NITs of different combination of chunk- and peer- selection policies when user abandonment exists and when it does not (with 95% confidence interval).

us when nodes depart, we model the staying time of normal seeds and partial seeds as an exponential distribution with the average of $1/\mu_1$ and $1/\mu_2$ seconds, respectively. After this time, normal or partial seeds also permanently leave the system.

In all our experiments, each chunk is equal to 10 seconds of playback. Also, we allow a startup buffer b for each peer, and define startup delay as the time taken for a peer to download the first b seconds of the video. We use $b = 10$ (i.e., 1 chunk) as the default. We assume that the video playout rate is 1Mbps for all videos. We summarize the different parameters used in the simulations and their default values in Table 5.2.

We use playback interruption time as our main metric. However, since the viewed length by a user varies widely, instead of just measuring total interruption time of each view, we normalize it by the viewed length, which we call the *normalized interruption time (NIT)*. In addition to interruption time, we also measure the wastage of system-wise bandwidth due to abandonment. We define *wastage* as the fraction of bytes downloaded in the swarm, but not viewed. That is, wastage $= 1 - \frac{\sum_{i \in P} V_i}{\sum_{i \in P} D_i}$, where P is the set of all peers, and D_i and V_i are the total bytes that peer i downloaded and viewed, respectively. Note that there is no wastage when every viewer watches the video fully. We will show that a hybrid of EF+RF causes substantial wastage compared to the EF-only case, in Section 5.3.5.

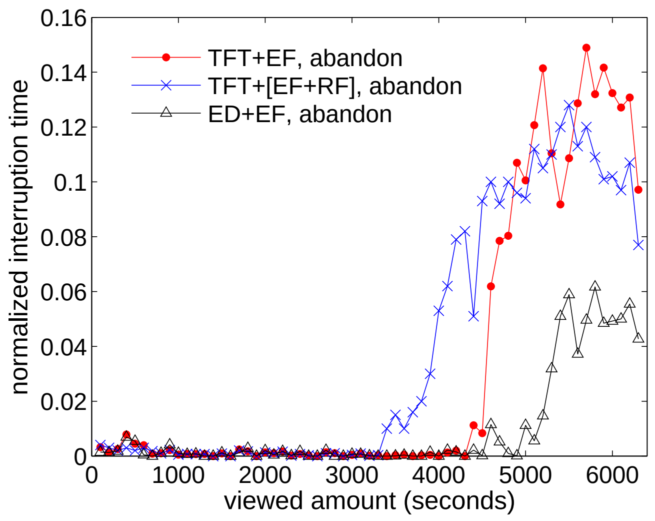


Figure 5.6: Average NITs of each peer group divided based on viewed length. The initial seed capacity is 3Mbps.

5.3.3 Impact of Abandonment

We first investigate how user abandonment affects the performance of different combinations of chunk selection and peer selection policies. We vary the initial seed capacity in Figure 5.5 and record the resulting NIT. Note that when we vary the capacity of the initial seed, we accordingly adjust the maximum number of its concurrent uploads allowed (e.g., 10 concurrent uploads with 2Mbps upload capacity, 20 with 4Mbps, etc.). Figure 5.5 shows that all three schemes (TFT+EF, TFT+[EF+RF], ED+EF) have larger NITs in presence of abandonment than without abandonment. This indicates that while the absolute time of interruption might be smaller with abandonment, the proportional impact of interruption is larger with abandonment. Proportion is more important because if a viewer is interrupted for longer, there is the further likelihood that he/she may abandon the video earlier [Li *et al.*, 2011; Dobrian *et al.*, 2011]. Clearly, a higher seed capacity benefits all the schemes. Also importantly, many existing works have suggested the desirability of using a EF+RF hybrid scheme for P2P VoD. However, we observe that with TFT, EF+RF hybrid actually causes larger NITs compared to EF when user abandonment exists. This is because with abandonment, chunks closer to the end of a video are viewed rarely. Exchanging rare

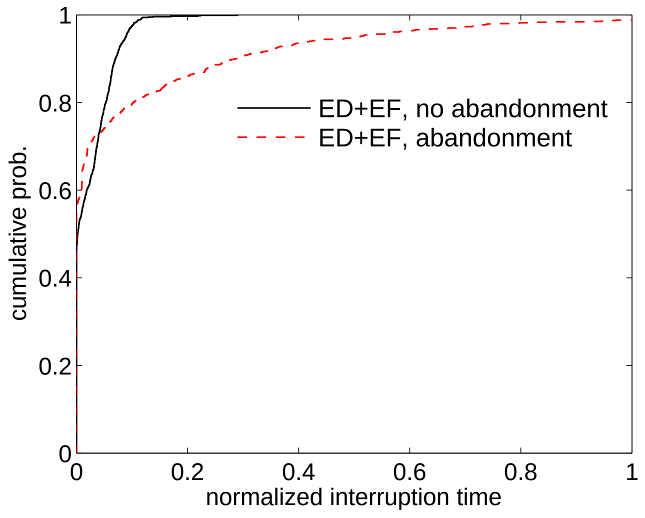


Figure 5.7: Cumulative distribution of NITs when user abandonment exists and when it does not, respectively.

chunks (typically later parts of the video) which are not watched results in inefficient use of resources. In Figure 5.5, we observe that our proposed ED+EF combination has the smallest NITs.

This shows the importance of accounting for abandonment; something that earlier works have overlooked. This also shows that serving peers with most urgent chunks helps improve overall user experience. We also measured the startup delay for each approach. We do not observe a significant difference between different approaches whether there is abandonment or not; this is not surprising as they all use EF at startup.

In Figure 5.6, we use one of the longer videos (105 minutes), group peers based on how much they watched, and plot the average NIT for each group. Specifically, we divide the video into 100 second bins, and group the viewers into these bins based on how much they watch (i.e., the first group includes peers who watched 0–100 seconds of the video, the next group watched 101–200 seconds of the video, etc.). The initial seed upload capacity is 3Mbps. We observe that peers who watch for a long period have larger NITs than peers who watch for a shorter interval. We also note that TFT+[EF+RF] reduces NITs compared to TFT+EF for peers who watch the video longer than 4800 seconds. However, for most of peers who watch less than 4800 seconds, TFT+[EF+RF] causes more interruption. As

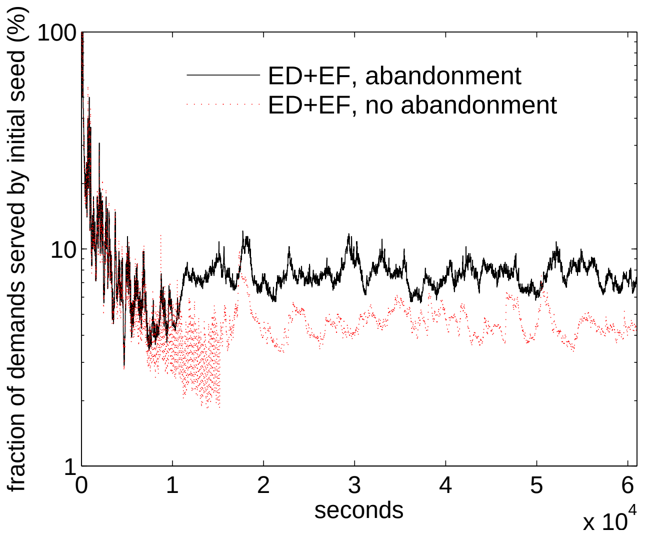


Figure 5.8: The fraction of demand (in terms of bytes) satisfied by the initial seed over time.

a result, TFT+[EF+RF] results in larger overall NITs than TFT+EF in the presence of abandonment, just as we saw in Figure 5.5. As before, the use of ED+EF results in consistently lower interruption than the other two policies.

To understand the cause for these results and their relationship to abandonment, we first compare NITs of each view as CDFs in Figure 5.7 between the case when abandonment exists and when abandonment does not exist. The initial seed capacity is 3Mbps, and all peers use ED+EF. We observe that with abandonment some views have very long NITs, such as NIT ≥ 1. Therefore, we now focus on the peers who have NITs larger than 1 to understand how abandonment makes their playback performance worse.

Specifically, we conduct an in-depth study step by step using a simple synthetic trace for a better understanding. While in the real trace peer arrival patterns are fixed, with the synthetic trace we have full control over peer arrivals. By adjusting arrival rates and patterns, we are able to more clearly explain the impact of abandonment by presenting distinct trends on the results with less variance. Based on the findings from the synthetic trace, we will also compare and validate our observations with the real trace results.

For the synthetic trace experiments, we model the peer arrival and abandonment patterns as random processes. We assume that peer arrival follows a Poisson process with rate

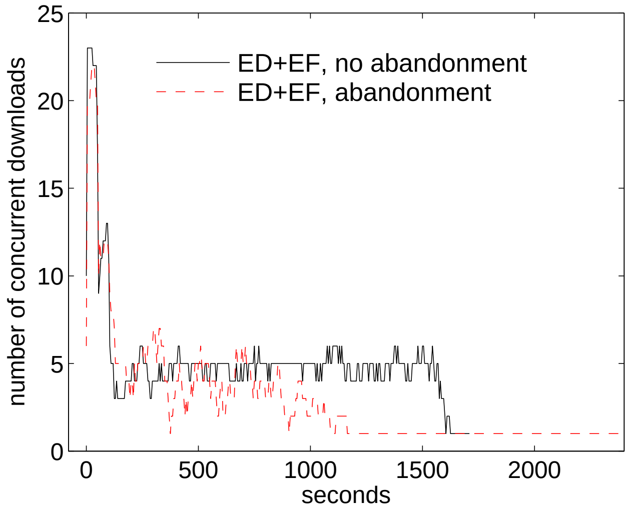
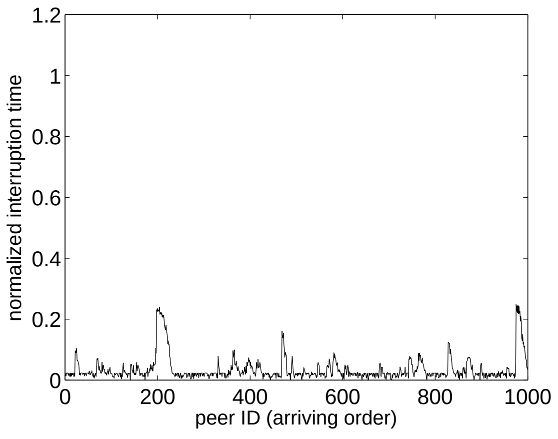


Figure 5.9: Number of concurrent downloads of a peer over time. The solid curve ends at 1610 seconds.

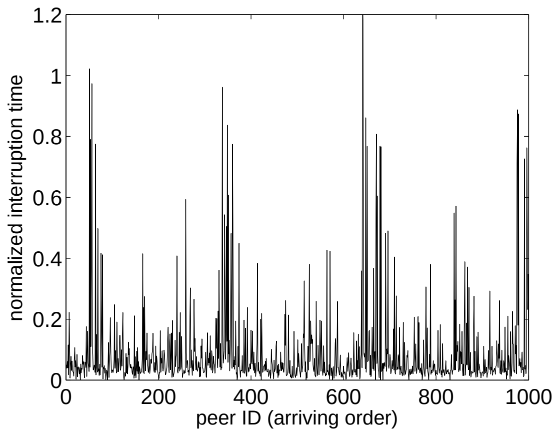
$\lambda = 0.05$. We use a 30 minute video, and each arriving peer watches uniformly between 3 and 30 minutes and then abandons. We measure NITs of 2000 consecutive peers who arrive in the system after the system reached a steady state (where the swarm size becomes stable). Also, to compare NITs more precisely, we remove startup buffer at each peer so that startup delay is also considered as an interruption and all contributions of delay are now included in the NIT. Unless otherwise stated, the synthetic trace experiments use the same experiment parameters as the real trace experiments in Table 5.2.

First from the synthetic trace results, we compare the load on the initial seed between with and without abandonment in Figure 5.8. With abandonment, the load as a byte fraction of requests served by the initial seed is larger than without abandonment. This is because peers leave early with abandonment causing loss of upload capacity available. This result indicates that abandonment imposes more critical role on the initial seed. Note that the initial large drops till the first 10^4 seconds for the both curves indicate that the swarm size is initially not yet stable but is growing till it becomes stable.

Then, we monitor the number of concurrent chunk downloads at each peer over time who has NIT ≥ 1 and observe that those peers have a similar trend to that presented in Figure 5.9. We see that when the peer joins the swarm, it initially has many providers, 22–23



(a) NITs of ED+EF with no abandonment



(b) NITs of ED+EF with abandonment

Figure 5.10: NITs of 1000 peers sorted in their arriving order

at max. However, with abandonment the number of concurrent downloads goes down, and after about 1100 seconds, the number becomes only 1 which is the initial seed and never increases until the peer abandons the video. On the other hand, without abandonment, although the peer loses lots of download connections in a similar manner, it manages to maintain about 4–6 parallel downloads. Also, the downloads end at 1610 second without abandonment, which means that the download finishes earlier than the actual playback.

This trend indicates that with abandonment older peers (i.e., those who arrived earlier than this peer) have all left at about the 1100 second mark, and therefore this peer loses all its possible uploaders other than the initial seed. We note that this trend is strongly related to our EF chunk selection policy since younger peers cannot help older peers with EF. However, we will show that although using EF+RF may alleviate this issue, EF+RF results in more peers having interruptions than EF only, with abandonment.

More importantly, losing older peers seen in Figure 5.9 occurs more severely with abandonment because viewers watch different length of the video. If a peer watches for a longer period than its older peers, that peer would be more likely to lose its potential uploaders early. In Figure 5.10(a) and 5.10(b) we show NITs of each peer in an arriving order, when abandonment does and does not exist, respectively. While a similar set of peers experience larger NITs in both cases, we observe that the magnitude is much larger in the case of

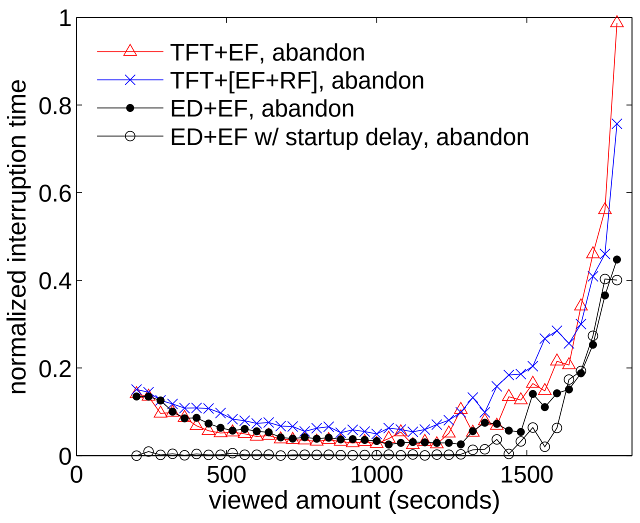


Figure 5.11: Average NITs of each peer group divided based on viewed length. The synthetic trace used.

abandonment.

In Figure 5.11, we divide peers into different groups based on their viewed length, and plot the average NITs of each group for the synthetic trace, similarly to Figure 5.6 with the real trace. Each group has a 40 second range of viewed length. We now clearly observe that NITs grow superlinearly as a peer watches the video for a longer time. We also note that TFT+[EF+RF] reduces NITs compared to TFT+EF for peers who watch for a very long time (more than 1640 seconds). This is because, by using RF, older peers have a chance to download from younger peers as well. However, for most of peers who watch for short durations, RF causes more interruption by exchanging chunks closer to the end of the video; but those chunks are rarely viewed. Peers who watch for a very short time, even smaller than 500 seconds, have slightly larger NITs than peers who watch around 500–1200 seconds. This is because, as stated earlier in this section, we do not consider startup delay for synthetic trace experiments. To confirm this, we also plot the results when peers have a startup buffer of 10 seconds of video just like the experiment with the real trace, and we see that NITs for peers who watched less than about 1300 seconds of the video with ED+EF is almost 0.

Comparing Figures 5.11 and 5.6, although NITs in the real world do not consistently and

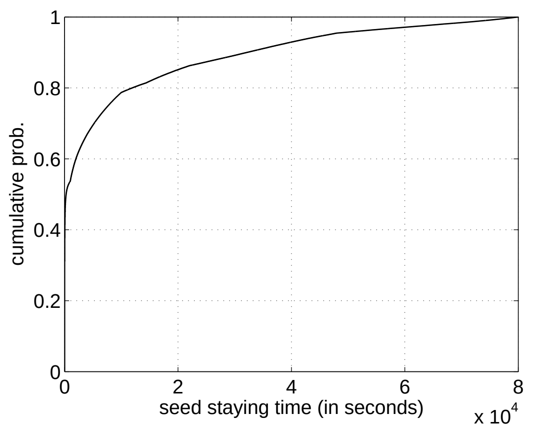


Figure 5.12: Seed staying time from trace

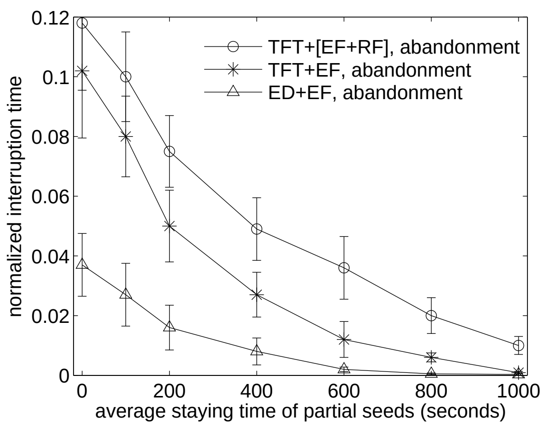


Figure 5.13: NITs of three different schemes in presence of abandonment as a function of partial seed staying time.

smoothly grow with viewed length, but rather fluctuate, peers with longer views generally suffer more interruptions than peers with shorter views. Furthermore, we observe exactly the same performance relationship among the three different policy combinations.

5.3.4 Utilizing Partial Seeds

We also investigate the effect of utilizing partial seeds. To understand the potential of seeds staying on in practice, we present results from our request traces collected at the set-top boxes of viewers. Specifically, for each customer, we calculate the distribution of the time between the completion of one video and the start of the next video request. Based on this, we determine how long each video would be available in a customer's set-top box (Figure 5.12). We observe that in over 45% of the occurrences, there is at least 1000 seconds of time the seed (whether it is a partial or normal seed) can continue to stay and serve an existing swarm before the user starts viewing another video.

In Figure 5.13, we plot NITs of different chunk- and peer- selection strategies as a function of the average staying time of partial seeds ($1/\mu_2$ in Section 5.2) when the initial seed capacity is 3Mbps with a maximum of 15 concurrent uploads. Note that we also make normal seeds who have downloaded the entire video stay on as long as the partial seeds,

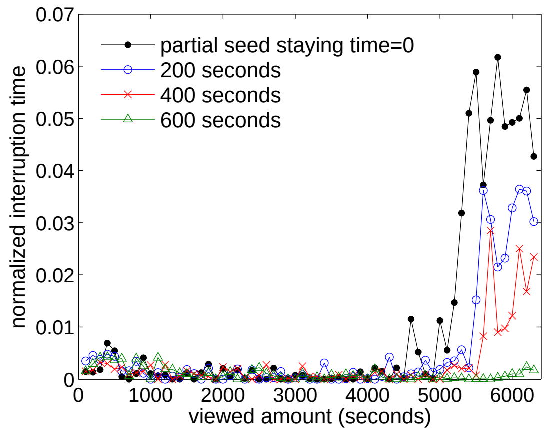


Figure 5.14: Average NITs of each peer group divided based on viewed length ($\mu_1 = \mu_2$). ED+EF used. Real trace used.

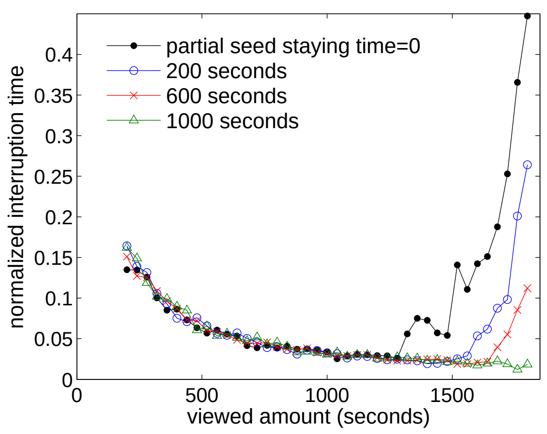


Figure 5.15: Average NITs of each peer group divided based on viewed length ($\mu_1 = \mu_2$). ED+EF used. Synthetic trace.

i.e., $\mu_1 = \mu_2$. In the presence of peer abandonment, having peers staying on as partial seeds benefits all of the strategies. Not surprisingly, the benefit increases as the staying time of partial seeds increases.

As shown in Figure 5.14, as the staying time increases, NIT decreases significantly, especially for the viewers with larger viewed lengths. For verification, we also repeat the partial seed experiments with the synthetic trace used in Section 5.3.3, and NITs of peers with long views gradually decrease as the partial seeds stay longer, as shown in Figure 5.15.

5.3.5 Minimizing Wastage

We measure the bandwidth wastage caused by abandonment for the three different policy combinations, and also investigate how utilizing partial seeds impacts wastage. First, we investigate how the peer selection policies, TFT and ED impact bandwidth wastage. When comparing the download rates of TFT+EF and ED+EF, we see that the average download rates of TFT+EF are higher than ED+EF (e.g., 1.57 Mbps vs. 1.20 Mbps) with abandonment but with no partial seed staying ($1/\mu_2 = 0$). However, the distribution of download rates is quite revealing. Figure 5.16 shows that more than 80% of peers achieve download rates higher than the video streaming rate (1Mbps) with ED+EF. On the other

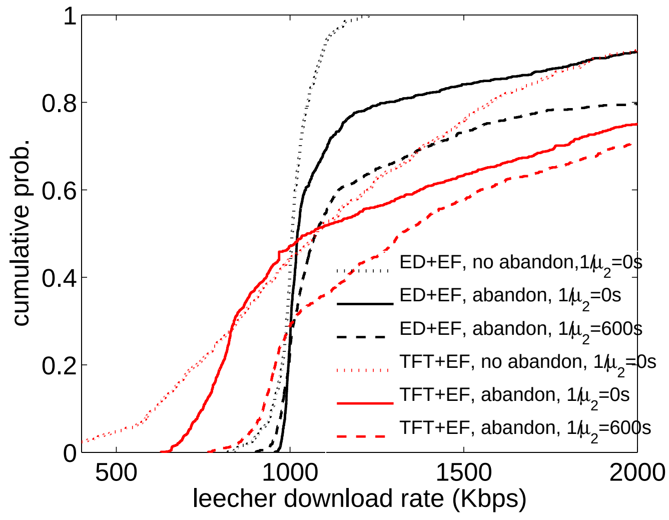
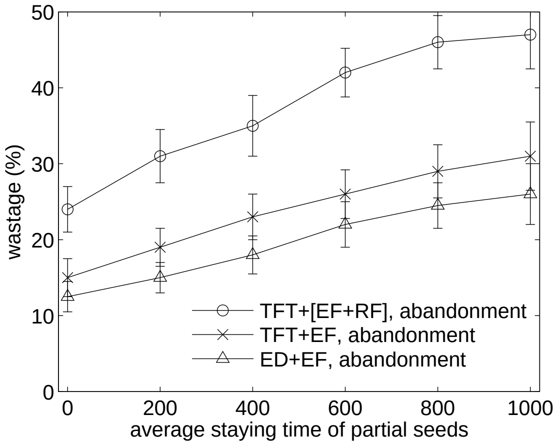


Figure 5.16: CDFs of leecher download rate ($\mu_1 = \mu_2$). $1/\mu_2 = 0$ indicates partial seeds leave immediately after abandoning.

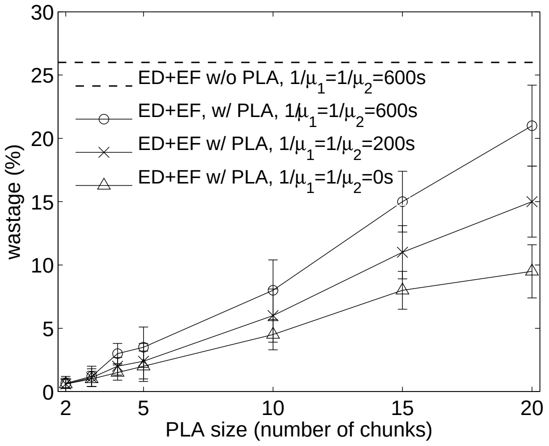
hand, TFT+EF has higher variability; some peers get high rates, while many fall below 1Mbps because they struggle to get unchoked, which is a key deficiency of such a policy for streaming video.

Consequently, we have more wastage with TFT schemes as seen in Figure 5.17(a). TFT+[EF+RF] causes more wastage than TFT+EF due to exchanging rare chunks by RF. ED+EF results in the least amount of wastage. Also, we see that as partial seeds stay longer in the swarm, wastage grows for all the schemes, since download rates of both TFT and ED schemes increase by having partial seeds contribute, as shown in Figure 5.16 (with $1/\mu_2 = 600$ secs).

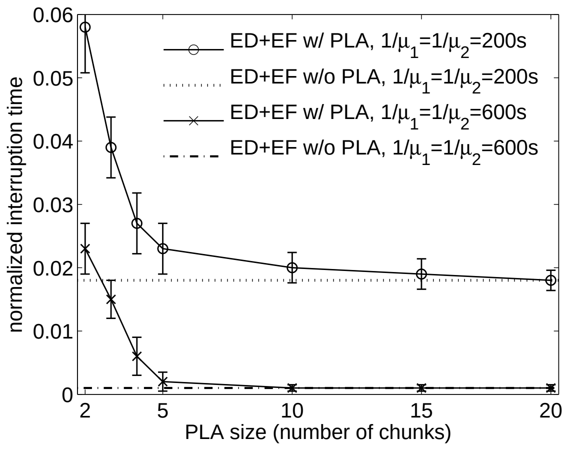
We performed experiments by varying the size of the playback look-ahead (PLA) window (described in Section 5.1.6) to achieve a balance between bandwidth wastage (Figure 5.17(b)) and playback quality (Figure 5.17(c)). We observe that with a modest PLA window size (e.g., 5 chunks), ED+EF can achieve almost the same level of playback performance but achieve much lower wastage – by almost 85% (reducing from wastage 26% to 4%). In contrast, a smaller PLA values cause more playback interruptions.



(a) Wastage and partial seed staying time



(b) Wastage and PLA



(c) NITs and PLA

Figure 5.17: Wastage of different approaches, varying partial seed staying time and PLA window size, with 95% confidence interval.

Part II

Enabling Adaptive Bitrate Streaming in P2P VoD

Chapter 6

JOINT-FAMILY: Adaptive Streaming in Multi-swarm P2P VoD

6.1 Analysis of P2P Systems for VoD and Adaptive Streaming

We analytically show how video popularity, the staying time of a peer in a swarm, and caching help increase system capacity. Further, we show how adaptive bitrate techniques can significantly improve the playback experience even for unpopular content.

6.1.1 Assumptions

The notations used in our model are summarized in Table 6.1. We use the leecher arrival rate λ for a video as its popularity (i.e., if arrival rate of video i is larger than that of video j, then i is more popular than j). A leecher's download (streaming) rate can be faster than the video playback rate for potentially fewer playback interruptions. We assume that each leecher watches a video till the end, and thus seeds have the entire video. However, all our

Parameter	Definition
B	Bit size of streaming video
u	Upload capacity of each peer (leecher or seed)
λ	Leecher arrival rate (Poisson arrival)
$1/\gamma$	Average seed staying time of exponential distribution
x	Number of leechers
y	Number of seeds
r	Playback rate of video
c	Number of videos each peer can locally cache

Table 6.1: Parameters and Definition

experiments in Section 6.3 also account for viewers' premature abandonment based on real traces. Similar to other P2P studies [Parvez *et al.*, 2008; Fan *et al.*, 2010] and based on the wireline subscriber statistics [poi, 2013], we assume that upload capacity u is the limiting factor (the download capacity per peer is much larger). u is identical for every peer. We investigate the impact of heterogeneous peers in Section 6.3.7.

6.1.2 Popularity, Download Rate, and Seed Staying Time

The fluid model based analysis by Qui and Srikant [Qiu and Srikant, 2004] suggested the download performance of files is relatively independent of their popularity. They explain that the supply and the demand placed by leechers are always offset regardless of video popularity. There has been subsequent work [Parvez *et al.*, 2008; Lehrieder *et al.*, 2012] based on their model to explain performance on live or on-demand streaming. In contrast to these models, we first show that more popular a video, the higher the download rate as long as the following conditions hold:

- request arrivals are stochastic, and
- after completing the download, each peer stays on to serve the video (as a seed) sufficiently long compared to the average download time.

The main reason for the difference between the results is that fluid models assume deterministic arrivals of requests, which likely holds when a video is highly popular (i.e., when the request arrival rate goes to infinity). However, when the request arrivals are stochastic — as seen in practice — a version of Feller's paradox takes place and Palm calculus [pal,] can explain what the fluid model misses. Intuitively, if we plot the intervals between request arrivals and observe the download rate at any random instant, our observation is likely to fall into a "larger" interval. In these large intervals, the download rates of leechers monotonically increases, since we assume the seeds stay for a sufficiently long time and many active leechers transition to being seeds. This simultaneously increases the supply as well as reduces the demand for download capacity. Feller's paradox explains why these longer intervals have a greater effect on the time averaged download times, and in our case the effect is beneficial.

Our analysis uses a continuous time Markov chain, where we define $x, y \geq 0$ to be the respective numbers of leechers and seeds in a swarm. Our model is motivated by a two-dimensional (2D) model by Veciana and Yang [Veciana and Yang, 2003] (which only presents recursive relationship). In our analysis, we first fix y and derive a conditional expectation using a variant of M/M/∞ queue. We then derive simple formulas for the expected number of leechers and download time. While our analysis could be equally applicable for file sharing scenarios, we focus on video streaming only.

Given y seeds, consider a Markov chain, where each state corresponds to the number of leechers (x). Then, the transition rate from state i to $i+1$ is: $q_{i,i+1} = \lambda$ for $i \geq 0$, where λ is the request arrival rate. For the transition down from i to $i-1$, we assume a "perfect cascade" as used in Fan et al. [Fan *et al.*, 2010], where all leechers except the latest arrival can always upload to other leechers. Then, $q_{i,i-1} = (\eta(i-1) + y)u/B$ for $i \geq 1$, where η corresponds to the efficiency parameter for data transfer from leechers [Veciana and Yang, 2003]. This parameter is experimentally shown to be close to 1 for most practical cases [Qiu and Srikant, 2004; Parvez *et al.*, 2008], and we also use $\eta = 1$ in the rest of the chapter. Then we obtain the following recursive equation for the steady-state probability of state $i \geq 1$:

$$\pi_i = \frac{\rho^i}{\prod_{k=1}^{i}(k+y-1)}\pi_0 \tag{6.1}$$

where $\rho = \lambda B/u$. From $\sum_{i=0}^{\infty} \pi_i = 1$, we derive π_0 as follows:

$$\begin{aligned}
\pi_0 &= \frac{1}{1+\sum_{i=1}^{\infty}\frac{\rho^i}{\prod_{k=1}^{i}(k+\theta)}} &(6.2)\\
&= \frac{1}{1+\sum_{i=1}^{\infty}\frac{\rho^i\theta!}{(k+\theta)!}} &(6.3)\\
&= \frac{1}{1+\sum_{i=1}^{\infty}\frac{\rho^{i+\theta}\rho^{-\theta}\theta!}{(k+\theta)!}} &(6.4)\\
&= \frac{1}{1+\frac{\theta!}{\rho^\theta}(e^\rho-\sum_{i=0}^{\theta}\frac{\rho^i}{i!})} &(6.5)\\
&= \frac{1}{\frac{\theta!}{\rho^\theta}(e^\rho-\sum_{i=0}^{\theta-1}\frac{\rho^i}{i!})} &(6.6)
\end{aligned}$$

where $\theta = y - 1$. Note that we assume that θ is integer.

Recall that the steady state probability is under the condition for a particular y. Using Equations (6.1) and (6.6) we can obtain the conditional expectation as follows:

$$\begin{aligned}
E[X|Y=y] &= \sum_{i=1}^{\infty} i\pi_i &(6.7)\\
&= \pi_0\sum_{i=1}^{\infty}\frac{i\rho^i\theta!}{(i+\theta)!} &(6.8)\\
&= \pi_0\theta!\sum_{i=1}^{\infty}(\frac{(i+\theta)\rho^i}{(i+\theta)!}-\frac{\theta\rho^i}{(i+\theta)!}) &(6.9)\\
&= \pi_0\theta!(\sum_{i=1}^{\infty}\frac{\rho^{i+\theta-1}\rho^{1-\theta}}{(i+\theta-1)!}-\theta\sum_{i=1}^{\infty}\frac{\rho^{i+\theta}\rho^{-\theta}}{(i+\theta)!}) &(6.10)\\
&= \pi_0\theta!(\frac{1}{\rho^{\theta-1}}(e^\rho-\sum_{i=0}^{\theta-1}\frac{\rho^i}{i!})-\frac{\theta}{\rho^\theta}(e^\rho-\sum_{i=0}^{\theta}\frac{\rho^i}{i!})) &(6.11)
\end{aligned}$$

where $\theta = y - 1$. By substituting π_0 (Equation (6.6)),

$$\begin{aligned}
E[X|Y=y] &= \rho-\theta\frac{e^\rho-\sum_{i=0}^{\theta}\frac{\rho^i}{i!}}{e^\rho-\sum_{i=0}^{\theta-1}\frac{\rho^i}{i!}} &(6.12)\\
&= \rho-\theta-\frac{\frac{\rho^\theta}{(\theta-1)!}}{\sum_{i=\theta}^{\infty}\frac{\rho^i}{i!}} &(6.13)\\
&= \rho-\theta+\frac{\rho^\theta}{e^\rho(\Gamma(\theta)-\Gamma(\theta,\rho))} &(6.14)
\end{aligned}$$

where $\Gamma(\theta)$ is the gamma function ($\Gamma(\theta) = (\theta-1)!$) and $\Gamma(\theta,\rho)$ is the upper incomplete gamma function ($\Gamma(\theta,\rho) = (\theta-1)!e^{-\rho}\sum_{i=0}^{\theta-1}\frac{\rho^i}{i!}$).

Now, let us consider the distribution for the number of seeds (y). A seed arrival is equivalent to a leecher completing download of the video. As a result, at steady-state, the leecher arrival rate λ is the same as the seed arrival rate. On the other hand, a seed leaves the swarm at the rate of γ (i.e., determined by the staying time). This forms the standard M/M/∞ queueing system, where the up-transition rate is λ and the down-transition rate is γy. Thus, $P[Y = y] = e^{-\sigma}\frac{\sigma^y}{y!}$, where $\sigma = \lambda/\gamma$. By combining this with (6.14), we get:

$$E[X] = \sum_{y=0}^{\infty}\left(\rho - y + 1 + \frac{e^{-\rho}\rho^{(y-1)}}{\Gamma(y-1) - \Gamma(y-1,\rho)}\right)\frac{e^{-\sigma}\sigma^y}{y!} \tag{6.15}$$

From Little's Law, the average download time is $E[T] = \frac{E[X]}{\lambda}$.

Evaluation: We numerically evaluate (6.15) and demonstrate the relationship between video popularity and download performance. We also validate our model with experiments using a discrete event-driven P2P VoD simulator (see Section 6.3.1 for detail). In our experiments, we consider a 1800-second video of r=625Kbps, resulting in B=1125Mbits. We use u=312.5Kbps. We also simulate state transitions using the 2D Markov model [Veciana and Yang, 2003] for comparing results with our analysis. Specifically, we start at state $(x = 0, y = 0)$ and simulate transitions according to the transition rates until we reach a steady state (where the change on both x and y becomes very small). After reaching a steady state, we record the time between arrival and conversion to a seed for each of next 3000 leechers and compute the downloading rate. We use a similar warm-up strategy for our event-driven experiments.

In Figure 6.1, we investigate the average download rate of leechers for different λ with $\frac{1}{\gamma}$ =1 hour. We compare four cases: simulated transition on the 2D Markov model [Veciana and Yang, 2003] (2D-MC), numerical results from our analysis model (MOD), and two simulation results with one using stochastic arrivals (SIM-Stochastic) and the other using periodic arrivals (SIM-Periodic). Note that SIM-Periodic is to understand the impact of the assumption used in previous fluid models [Qiu and Srikant, 2004; Parvez *et al.*, 2008]. First, the figure shows that our model closely matches 2D-MC and SIM-Stochastic. We observe a clear trend in which the average download rate increases as λ (i.e., popularity) increases. In contrast, we see that the trend with SIM-Periodic is distinct from the other cases, where the increase in download rate seems slowing down with increasing λ. This result indicates

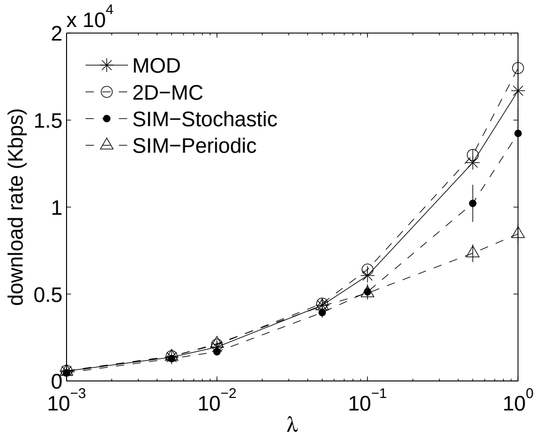


Figure 6.1: Download rate as a function of λ with $1/\gamma = 3600$ secs. (X axis in log scale)

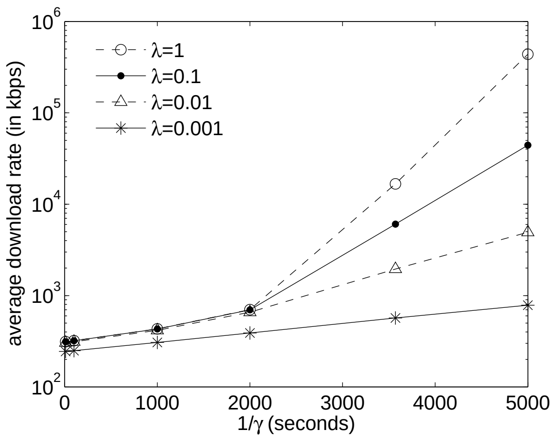


Figure 6.2: Download rate as a function of seed staying time $1/\gamma$. (Y axis in log scale)

that the leecher arrival pattern also plays a critical role in the download performance, and the assumption of periodic arrivals in the fluid models [Qiu and Srikant, 2004; Parvez *et al.*, 2008] can lead to incorrect conclusions in practical scenarios.

We next investigate the effect of seed staying time on download performance. In Figure 6.2, we plot the average download rate from our analytical model when we vary the seed staying time (X axis) and arrival rate (different lines). When the seed staying time is smaller than 2000 seconds, the download rate changes little with different popularity (λ), just like in [Qiu and Srikant, 2004; Parvez *et al.*, 2008]. However, as the seed staying time is sufficiently large, the download rate varies significantly as λ varies, showing video popularity affects download performance only under a long seed staying. When the video size is larger and the corresponding download time increases, the seed staying time is also required to be longer accordingly for the same observation (figures not shown here).

6.1.3 Caching to Increase Staying Time and Download Rate

As seen in Section 6.1.2, a necessary condition where popularity and download rate are correlated is for peers to stay as seeds for a sufficiently long period, compared to their download time. One way to increase seed staying time of a video is for a peer to cache the video and act as a seed serving other viewers of the same video even after the peer has moved on to viewing another video. However, with multiple videos in cache, a peer would need to split its upload capacity between those multiple videos, and thus it is not immediately clear whether caching multiple videos would improve the performance.

To analyze the benefit of caching, we first assume that each video is the same size of B bytes, and a peer can store a maximum of c videos. Note that our analysis in Section 6.1.2 corresponds to $c = 1$. One can envisage a variety of policies on how to split the upload capacity between multiple videos, depending on whether a peer is actively watching a video or not. To make the analysis tractable, we use a simple policy where a leecher watching a video serves only the video that it is watching. When not actively watching, a peer equally splits its upload capacity between c videos in its cache. We remove this assumption in our protocol design and experiments.

Using our Markov chain based analysis, but also considering a cache of size c, the down-transition rate from state i to $i-1$ would be:

$$q^c_{i,i-1} = (i + y/c - 1)u/B \tag{6.16}$$

for $i \geq 1$. Note that the benefit of caching from this analysis actually serves as a lower bound, as the transition rate $q^c_{i,i-1}$ assumes that all c videos are always requested. In particular, if a cached video is not requested, in practice a peer would allocate its upload capacity to the other videos being requested, resulting in a higher transition (service) rate than modeled here.

While a peer's upload capacity is split into c videos, a video stays longer in its cache for larger c. The cache replacement policy plays a role in determining how long a video would stay in the cache. In our analysis, we make a simplifying assumption that a peer uses FIFO (First-In First-Out) replacement. However, in our experiments, we also compare FIFO with LFU (Least Frequently Used). With FIFO, the time a peer stays as a seed, S, for each video is hypoexponentially distributed with the average $E[S] = c/\gamma$. The distribution for the number of seeds in the system still holds for $c > 1$ as:

$$P[Y = y] = e^{-\sigma_c}\sigma_c^y/y! \tag{6.17}$$

where $\sigma_c = c\lambda/\gamma$. From (6.16) and (6.17), we can obtain the average download time T by

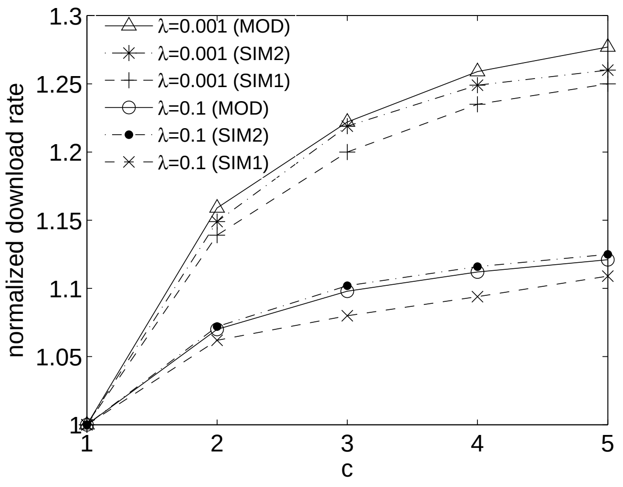


Figure 6.3: Caching multiple videos: each download rate is divided by the download rate when $c = 1$. ($1/\gamma = 3600$ secs)

following similar derivation as in Section 6.1.2. In our numerical evaluation of $E[X]$ with $c > 1$, we substitute y in Equation (6.14) with an integer value $\lfloor y/c \rfloor$ instead of y/c for simplicity. Note that this simplification underestimates the download rate in the presence of caching and thus provides a lower bound of the benefit from caching.

Evaluation: We validate our caching analysis using the simulator as in Section 6.1.2 with a synthetic trace. Figure 6.3 plots the normalized average download rate from our analysis and from the simulation for different cache size c. For SIM1, we simulate exactly the policy described for deriving Equation (6.16) for validation. Also, SIM2 shows the results without our assumption so that a leecher actively watching a video also uploads all other videos in its cache. We first observe that the analysis (MOD) and simulation results match well. Also, caching is more beneficial with small λ (i.e., less popular videos). Secondly, we see the diminishing returns as c grows since our small u which is the bottleneck quickly becomes more utilized (thus, we omit the results with $c > 5$). Finally, we show that the download rate in SIM2 only improves as we remove our assumption. In Section 6.3.5 we also explore different cache replacement schemes such as LFU using real-world traces.

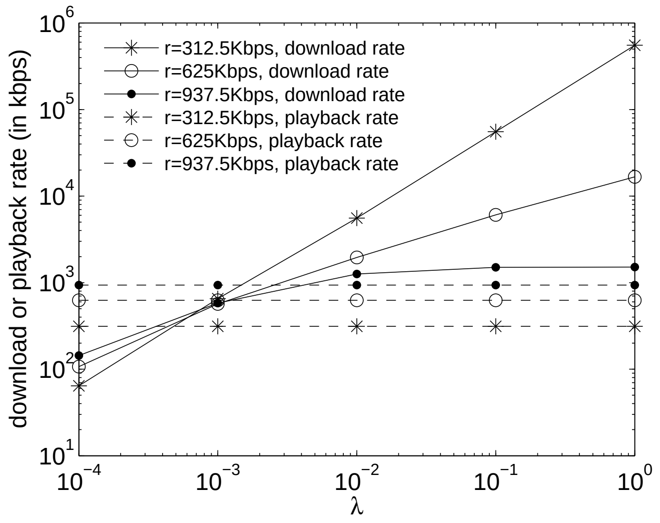


Figure 6.4: Download/playback rate vs. arrival rate with different chunk bitrates

6.1.4 Adaptive Bitrate Analysis

We showed in Section 6.1.2 that more popular videos result in higher download rates only with seeds staying long enough. When the download rate is (unnecessarily) much higher than the video playback rate, we now leverage the abundant capacity to improve the video quality through ABR. Using the example of a single video at different bitrates via our model, we show in Figure 6.4 that as the video popularity varies, the achievable average download rate varies quite significantly. We choose 3 different bitrates (312.5 – 937.5 Kbps) with the corresponding horizontal lines. When the video is unpopular, the peer download rate can be smaller than the playback rate, especially for the higher bitrates, likely resulting in playback interruptions. When the video is more popular ($\lambda = 0.01$ or higher), the download rate is higher than the playback rate, especially for the lower bitrates (e.g., 312.5Kbps).,

We make the following observations: First, using a single bitrate for all videos is suboptimal. If the bitrate is set too high, streaming an unpopular video would result in significant amount of playback interruption. If the bitrate is too low (with the goal of minimizing interruptions), viewers of popular videos would be unnecessarily restricted to low bitrates – i.e., poor streaming quality. To overcome this, one might consider predicting the video popularity and using the highest bitrate sustainable for that popularity. However, that is challenging, since we have to deal with prediction error and popularity changes. With ABR, the system can potentially adapt to the currently available bandwidth of a video, which does not require the popularity information, and thus the bitrate adaptively becomes large for popular videos and small for unpopular videos.

We now show that by using ABR with P2P VoD, we can deliver higher video quality to a viewer of more popular videos which can sustain higher bitrate. Suppose we have m playback bitrates: $R = \{r_1, r_2, \ldots, r_m\}$, where $r_i < r_{i+1}$. In our analysis, we assume an idealized rate adapting scheme, where a leecher only increases the video bitrate to reach the highest bitrate it can sustain. Specifically, each leecher starts with r_1 and increases the bitrate from r_i to r_{i+1} if it has at least s_u seconds of video chunks at rate r_i buffered ahead of its playback point (also explained in Section 6.2.3). If a leecher is not able to go to a higher bitrate, then it stays at the current bitrate until the streaming finishes. Also, we assume that all leechers for a given video go through the same set of "transition points" in a steady state. In other words, all leechers download B_1 bytes at r_1 before switching up to r_2 and receive B_2 bytes at r_2 before transitioning to r_3, and so on.

Our goal is to find an equilibrium point $(B_1, B_2, \ldots, B_m)$, and then calculate the corresponding download rate: $\frac{\sum_{i=1}^{m} B_i}{\sum_{i=1}^{m} B_i/r_i}$. To determine an equilibrium point, we use the following steps. Suppose we have an estimate of $\tilde{B} = (\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_m)$. We consider m independent Markov chains, one for each bitrate as described in Section 6.1.2. Each state is the number of leechers downloading at the corresponding bitrate. We assume that a seed for a video splits its capacity across multiple bitrates, such that it serves chunks of r_k in proportion to $\tilde{B}_k$. That is, the down-transition rate for the Markov chain corresponding to chunks of r_k is:

$$q^k_{i,i-1} = (i + f_k y - 1)u/\tilde{B}_k, \tag{6.18}$$

where $f_k = \frac{\tilde{B}_k}{\sum_j \tilde{B}_j}$. Then, following the analysis for each Markov chain in Section 6.1.2 (Equation (6.15)), we can derive the average download time ($\tilde{T}_k$) and the corresponding download rate ($\tilde{d}_k$). However, since the bitrate switch happens only after s_u seconds of chunks at r_k are buffered, we can calculate the corresponding time as $T'_k = s_u r_k/(\tilde{d}_k - r_k)$, which we expect to match $\tilde{T}_k$ in an equilibrium point. In our evaluation, we calculate $B'_k = T'_k \tilde{d}_k$ and numerically find an estimate $\tilde{B}$ that minimizes the Euclidean distance from

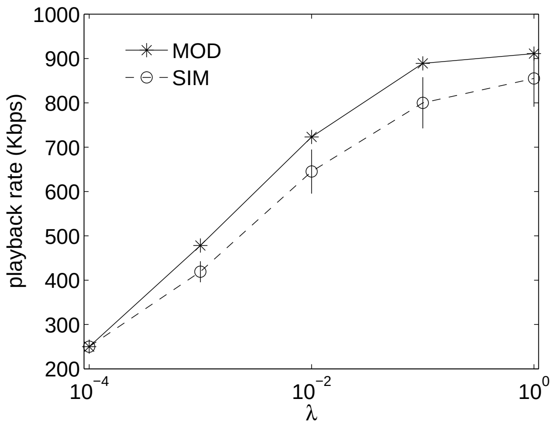


Figure 6.5: Validation of ABR analysis using simulation

$B' = (B'_1, \ldots, B'_m)$.

Evaluation: We can employ a variety of methods to find the equilibrium point minimizing the Euclidean distance (e.g., gradient descent) between $\tilde{B}$ and B'. However, to minimize the error arising from the particular method we use, we evaluate an entire space (using small fixed increment on $\tilde{B}$ values) and report the point with the minimum distance. We use a 1800 second video with 4 bitrates {250, 500, 750, 1000} Kbps, and set $s_u = 50$. In the simulation, peers have to switch down to lower bitrates if the size of buffered chunks becomes smaller than s_d, and we use $s_d = 10$ (see Section 6.2.3 for detail). Figure 6.5 shows the average playback rates obtained from both our model and simulator as video popularity varies. Considering that, unlike the model, peers in simulation may go down to lower bitrates and peers transfer data chunk-by-chunk (each 10 second chunk) instead of bit-by-bit, the two results match reasonably (especially in the variation with popularity), and demonstrate that with ABR in a P2P system, we can achieve a higher playback rate for a more popular video.

6.2 JOINT-FAMILY Design

We take the learnings from our analytical results in Section 6.1 to design a P2P protocol that supports the delivery of high quality video using ABR. To the best of our knowledge,

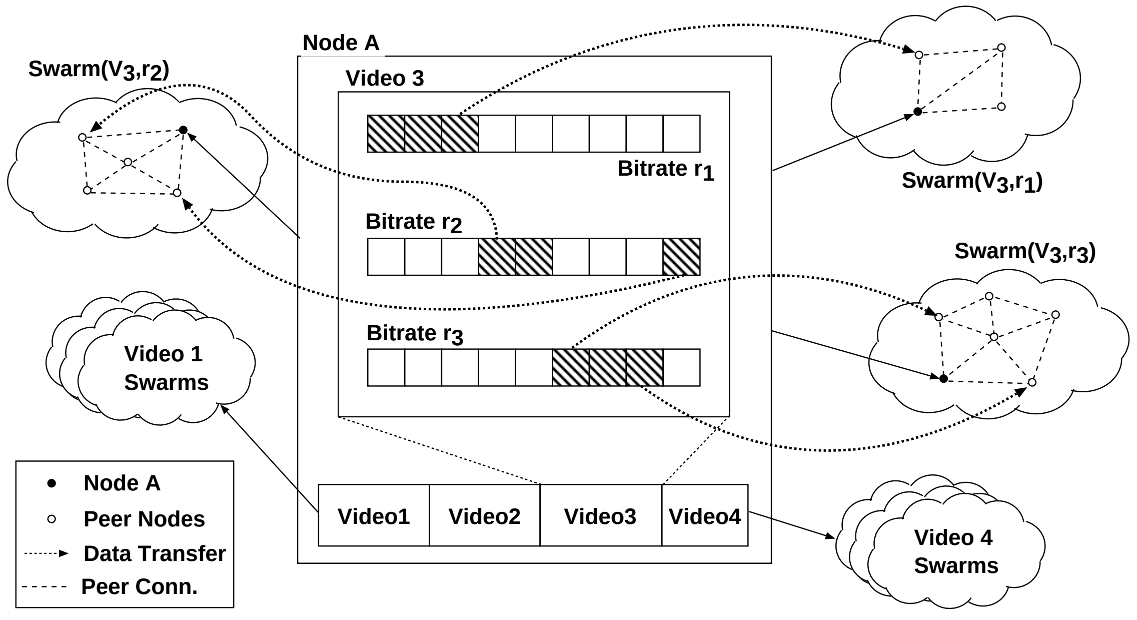


Figure 6.6: A peer in Joint-Family participates in multiple swarms.

Joint-Family is the first practical P2P VoD system that incorporates ABR. Joint-Family enables peers to enjoy the highest possible playback rate based on the available system capacity.

6.2.1 Overview

Most P2P systems maintain a notion of a "swarm" per video. Peers watching this video participate in the swarm and exchange chunks with other peers. With ABR, this delineation of a swarm per video becomes unclear since the same video has different set of files, one at each rate. A natural extension, and one that we use, is to assign a different swarm for each rate of the video. This change alone, however, is not sufficient. Peers today participate in one swarm only. Each time they attempt to change rates due to the ABR rate adaption, they would have to leave one swarm and join the swarm of the next rate. Leaving one swarm and joining another is inefficient as it is heavyweight process and also introduces a lot of churn in the system. Instead, a peer in Joint-Family joins the different swarms of each video concurrently and maintains active connections. The peer then sends out requests to the appropriate swarm as it downloads and uploads chunks of different bitrates as a result of bitrate adjustment.

Once we have the support for multiple swarms of a given video, the same primitive can be extended to support participation in multiple swarms of different videos. This allows a peer to serve cached chunks of videos it has already viewed, which as shown in Section 6.1.3 and 6.1.4 has a beneficial effect on the overall download performance and playback rate for ABR videos. Figure 6.6 illustrates the typical multi-swarm participation of peer A. A has 4 videos in its cache. The figure focuses on Video3 and shows that, as a result of rate adaptation, A has chunks in each of the 3 rates of Video3. A simultaneously participates in the swarms associated with each of these rates (solid dot in each swarm). The figure also shows A concurrently uploading chunks at different rates to peers (unfilled dots) in the corresponding swarms. These peers will also be participating in multiple swarms, but may not necessarily be connected to A in all of these other swarms. The same process is repeated for the other videos in A's cache (e.g., Video1).

6.2.2 Protocol Mechanisms for Multi-Swarm P2P

While multi-swarm participation is conceptually straightforward, realizing it in P2P systems requires a detailed understanding of inter-dependencies between protocol components and careful protocol re-design.

Connection management: We term all connections that node A has to peers in swarms of the video it is currently watching as *selfish*. Connections to swarms of cached videos are termed *altruistic*. The peer on the other end of a selfish connection, B, can be either a leecher or a seed. In the latter case, this is an *altruistic* connection for B. However, a connection cannot be altruistic for both endpoints. In typical P2P systems, a node can have connections to a maximum of n peers to avoid depleting local resources (e.g., by having too many TCP connections). For example, a typical BitTorrent peer can have as many as 80 connections. It contacts a tracker for more peers if the number of connections goes below 40. When a peer participates in multiple swarms for multiple videos, there is an inherent tension between the number of selfish connections and altruistic ones.[1] Specifically, if the

[1]We do not differentiate swarms for a single video since a peer can always switch between different bitrates.

peer uses its entire quota for selfish connections, caching is rendered useless. Conversely, even with sufficient connections, a leeching peer can suffer from starvation if the majority of its connections are altruistic.

Our solution with multiple swarms is to partition the number of connections for different swarms. We define a parameter α_l, such that the number of altruistic connections for a leecher is at most $n\alpha_l$. In Joint-Family, a leecher needs to re-classify the connections regularly and ensure that the number of altruistic connections is below the threshold. In the experiments, we use α_l=0.5. However, by definition, a peer who does not actively watch any video cannot have a selfish connection. For those peers, α_l=1 is used, allowing all connections to be altruistic.

Another aspect in a multi-swarm P2P system is to choose which peers to serve. In BitTorrent-like P2P VoD systems, the peer selection behavior changes depending on whether a peer is leeching or not. Specifically, a leecher unchokes those peers that sent the leecher the most chunks, while a seed unchokes those peers that can download the fastest. In Joint-Family, a peer can simultaneously be a leecher (for the video it is currently watching) and a seed (for other videos in its cache). As a result, if the BitTorrent policy is strictly followed, a leecher has no incentive to use upload capacity for altruistic connections. This is because the leecher is more likely to be unchoked when it uses all its upload bandwidth for bilaterally selfish connections. We present more detailed protocol mechanisms related to peer selection in Section 6.2.4.

Caching and sharing multiple videos: We have shown in Section 6.1.2 that increasing seed staying time in a swarm increases the capacity of the swarm. Our approach to increase staying time is, as modeled in Section 6.1.3, to cache videos previously watched and share them with other peers. Sharing multiple videos simultaneously is currently not possible in VoD systems as peers move from one swarm to another as they change videos. However, our primitive of participating in multiple swarms allows a peer in Joint-Family to cache and share multiple videos in parallel. We assume that each peer can store at most c different videos in its local cache regardless of the length of the video (we recognize videos can be of different lengths, and ABR or premature abandonment can also cause a difference in size). When the cache is full, a peer can choose the video to be deleted based on well-known cache replacement policies. In Section 6.3.5, experimental results on the benefits of caching are presented.

6.2.3 Chunk Selection and Rate Adaptation

Chunk selection: In Joint-Family, we use the Earliest-First (EF) chunk selection policy. There are a number of advantages we get from using EF. As shown in Section 5.3, EF allows for a fast startup, potentially fewer and shorter interruptions, and smaller wastage of downloaded chunks when users abandon viewing a video. Also, with buffer-based bitrate adaption schemes for ABR, having more sequential chunks in the playback buffer is more likely to help the peer move up to a higher playback rate quickly. Moreover, ABR complicates using Rarest-First (RF), as the rarest chunk at the time of download may not match the right bitrate at the time of playback. While we do not address it in this paper, EF is also amenable to DVD-like operations. Further, the performance degradation by using EF is highly dependent on the number of seeds in the swarm as shown in Section 5.3.4, and our caching mechanism increases the number of seeds and helps avoid the "missing piece syndrome" [Zhou *et al.*, 2011].

Rate selection: Having identified the chunk to download, the peer needs to decide which of the video rates to download. As is frequently adopted in practice [Akhshabi *et al.*, 2011], we have designed Joint-Family to use hysteresis when making a change in the bitrate for a video, so that the quality does not change too frequently, thereby providing the user a better quality-of-experience (QoE). However, our algorithm is parameterized and can easily get rid of the hysteresis if needed. Note that unlike several rate adaptation schemes used in server-client based ABR [De Cicco *et al.*, 2011; Liu *et al.*, 2011; Tian and Liu, 2012], we do not estimate the bandwidth between an uploader and a downloader. Unlike server-client schemes, in P2P a peer generally downloads chunks simultaneously from multiple different uploaders, and also peers who can upload to that downloader keep changing frequently depending on their chunk availability.

Instead of checking the bandwidth for each connection, a leecher uses a simple bitrate adaptation scheme based on its buffer status. Once the peer's buffer goes above (below) a certain threshold, it triggers the peer to adopt a bitrate increase (decrease). We supplement

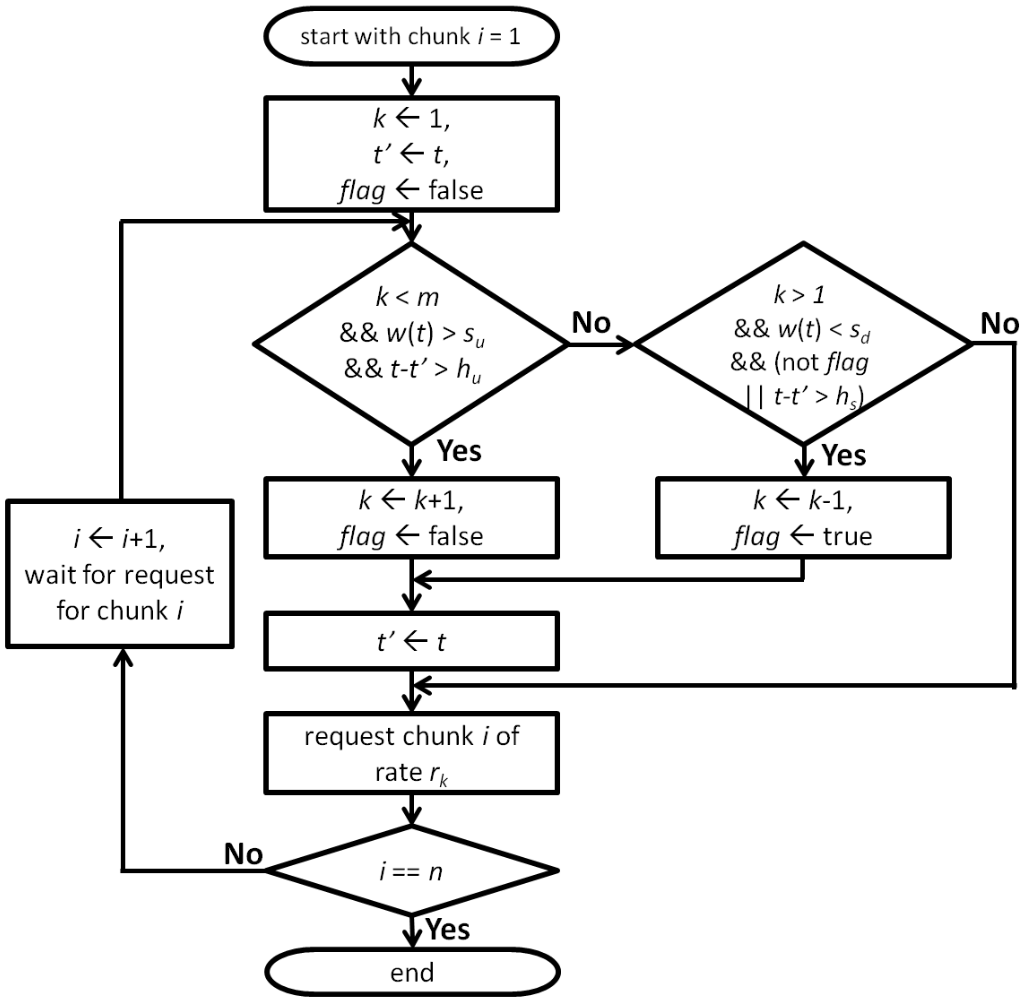

start with chunk i = 1
$k \leftarrow 1$, $t' \leftarrow t$, $flag \leftarrow$ false
$k < m$ && $w(t) > s_u$ && $t-t' > h_u$
No
$k > 1$ && $w(t) < s_d$ && (not $flag$ || $t-t' > h_s$)
No
Yes
$k \leftarrow k+1$, $flag \leftarrow$ false
Yes
$k \leftarrow k-1$, $flag \leftarrow$ true
$i \leftarrow i+1$, wait for request for chunk i
$t' \leftarrow t$
request chunk i of rate r_k
No
$i == n$
Yes
end

Figure 6.7: Flow chart for bitrate adaptation

this with hold-down timers to avoid rapid bitrate fluctuations. Specifically, a peer increases the bitrate if the following two conditions hold: its buffer has more than s_u seconds of chunks to play back (i.e., sequential chunks), and the last bitrate change was more than h_u seconds ago. Contrarily, a peer decreases the bitrate if its buffer has less than s_d seconds of chunks, and the last downward rate change was more than h_d seconds ago.

The specific adaptation logic used is shown in a flow chart in Figure 6.7, where a viewer's buffer size (in seconds) at time t is $w(t)$, the number of the video chunks is n, and m different bitrates $\{r_1, ..., r_m\}$ are provided. Note that h_d is applied only when a viewer has to switch down to lower bitrates sequentially.

A possible improvement in bitrate selection could be to also consider chunk availability at a rate. For example, for a particular portion of a video, if more peers have the chunk at bitrate r_i than at r_j, a leecher might prefer the chunk at r_i. We briefly explored this direction, but found that without careful design, peers can end up being stuck at lower bitrates even when there is capacity. This is because other peers may have downloaded lower bitrate chunks at a time when the swarm could only support that low rate. A sophisticated bitrate selection scheme that takes both chunk availability and video playback quality into account is still an open area of research.

6.2.4 Earliest-Deadline (ED) Peer Selection

We complement the Earliest-First (EF) chunk selection policy in Joint-Family with an enhanced peer selection strategy of choosing the peer with the "earliest-deadline", as we introduced in Section 5.1.3. This replaces BitTorrent's Tit-for-Tat (TFT) peer selection policy which favors clients with higher upload bandwidth and more chunks, but leads to frequent playback interruptions in streaming applications. To satisfy a viewer's uninterrupted playback experience, Joint-Family ensures each chunk to be delivered to the viewer prior to its deadline using the ED policy (more details in Section 5.1.3).

6.3 Performance Evaluation

We evaluate the performance of Joint-Family and compare it to a generalized version of state-of-the-art P2P approaches, using trace-driven simulations. We first show that the changes proposed in Joint-Family result in significant improvements in terms of the video playback rate and the number of interruptions and thereby improve a viewer's quality-of-experience (QoE). We then show how each of the design policies in Joint-Family contributes to improving system performance.

6.3.1 Experiment Setup

To evaluate Joint-Family, the BitTorrent simulator [Bharambe *et al.*, 2006b] is used with the following major modifications: (1) video streaming support (e.g., playback buffer), (2)

Parameter	Default value
Number of initial servers	5
Upload bandwidth of each server	25 Mbps
Peer upload/download bandwidth	625 Kbps/2 Mbps
Non-ABR video bitrate	625 Kbps
ABR video bitrates	250,500,750,1000 Kbps
Max. concurrent uploads per server	30
Max. concurrent uploads per peer	5
Chunk size	10 secs
Startup buffer size per peer	10 secs

Table 6.2: Simulation parameters

bitrate adaption (Section 6.2.3), (3) multi-swarm participation (Sec. 6.2.2), (4) different chunk (Sec. 6.2.3) and peer selection policies (Sec. 6.2.4). Note that for the hybrid chunk selection (EF+RF), a peer initially uses EF, but switches to EF+RF once enough chunks are in its playback buffer. This helps achieve lower startup delay and playback continuity by providing the slack needed to deal with possible future reductions in the download rate. In our experiments, a peer uses EF with probability 0.7 and RF with 0.3, once there are 5 or more chunks in its buffer.

To reflect realistic viewing patterns of a large population of users, trace data from a nationally deployed VoD service is used. The data covers a two week period with millions of requests. The trace contains information including the anonymized user ID, request time, video ID, video length, and the duration viewed for each session. For the experiments, our 14-day trace is split into seven 2-day trace segments. We use these trace segments to get 7 different simulation runs and report the average results and 95% confidence intervals.

We summarize the different parameters used in the simulations in Table 6.2. We use 5 servers, each with 25Mbps uplink capacity to host all the videos and behave like seeds. We assume continuous network connectivity of each joining peer until the end of an experiment so that the peer helps other peers as a seed for previously viewed videos. While the playback rate for non-ABR videos is set to 625Kbps, we use 4 quality levels for ABR videos: 250, 500, 750 and 1000Kbps (the average being 625Kbps). Like most P2P systems, each video is broken into chunks of 10 seconds of playback, and a peer can play back a chunk while it is being downloaded (subject to the startup delay and the appropriate portion being available). For ABR, hold-down time for bitrate switch-up (h_u) and switch-down (h_d) are set to 30 and 10 seconds, respectively. Also, for the buffer size parameters of switch-up and switch-down, we use $s_u = 50$ and $s_d = 20$ seconds. We chose these bitrate switch parameters based on our experiments (not shown here) where the parameters achieved the largest average playback rate with fairly small playback interruptions. We use **playback rate** and **interruption time** as the metrics to evaluate video playback performance. While the former gives information about the quality of video viewed by the user, the latter captures the aggregate disruptions experienced by the viewer.

6.3.2 Joint-Family vs. Server-based ABR

To first understand the benefit of using P2P for ABR video delivery, we compare Joint-Family with the traditional server-based ABR scheme. Viewers in the server-based ABR do not share their downloaded content. Joint-Family uses a cache size of $c = 5$. The same buffer-based rate adaptation is applied in both schemes. We look at the average viewers' playback bitrate and interruption time in Table 6.3 as the server bandwidth increases. We assume a single server, with the maximum number of concurrent uploads allowed for each 25 Mbps of server upload bandwidth being 30, as in Table 6.2 (e.g., 125 Mbps server bandwidth allows $30 * 5 = 150$ concurrent uploads). Joint-Family requires only 125 Mbps server bandwidth to achieve about the same performance as a server-based ABR with 2 Gbps server bandwidth. Note that the improvement in the playback rate of Joint-Family with larger server bandwidths reaches a point of diminishing returns because the highest ABR video bitrate is limited to 1000 Kbps.

6.3.3 Joint-Family vs. State-of-the-art P2P

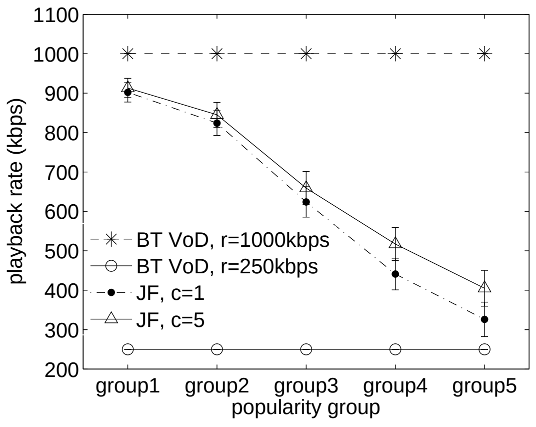
For the performance comparison of Joint-Family, instead of comparing with specific existing implementations, we use a generalized implementation that incorporates the state-of-the-art in P2P VoD. The generalized implementation (henceforth BT VoD) uses the hybrid policy

Server bandwidth	125 Mbps	500 Mbps	1 Gbps	2 Gbps
Server-based	261 Kbps	334 Kbps	501 Kbps	723 Kbps
ABR	195 seconds	67 secs	16 secs	2 secs
Joint-Family	748 Kbps	881 Kbps	940 Kbps	975 Kbps
(c=5)	4 seconds	0 sec	0 sec	0 sec

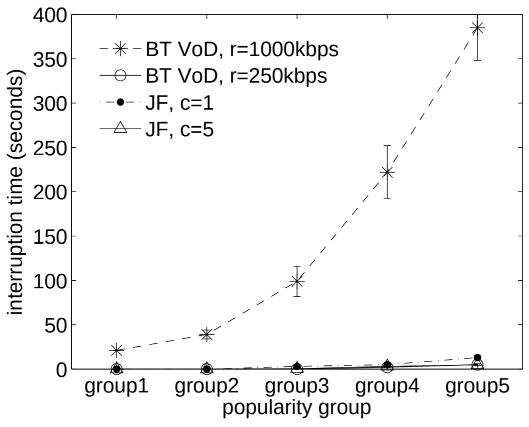
Table 6.3: Playback rates and interruption time with server-based ABR scheme and Joint-Family

(EF+RF) for chunk selection and TFT for peer selection. Since existing P2P systems only support a single rate, we experiment with two fixed rates: 1000Kbps and 250Kbps to represent the two extremes (high quality and no interruption). Note that 250Kbps is the maximum bitrate for BT VoD that achieved no interruption for all viewers. Further, these systems only allow participation in one swarm (equivalent to $c = 1$ in Joint-Family). We use two scenarios for Joint-Family: $c = 1$ and $c = 5$. Joint-Family uses ABR with {250, 500, 750, 1000}Kbps bitrates and all the improvements suggested in this chapter. The goal here is to show the total benefits from using Joint-Family. To understand the dependency between popularity and playback performance, results are presented for 5 different groups, where each group has 100 videos for corresponding popularity. For example, Group 1 consists of the 100 most popular videos, while Group 5 has the 100 least popular videos.

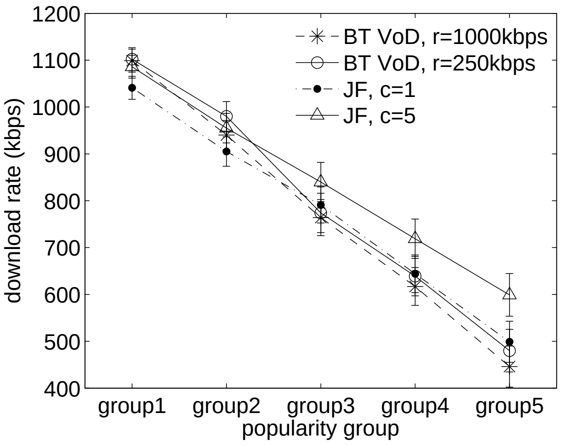
First, Figure 6.8(a) shows the average playback rate experienced by peers. With BT VoD, the playback rate is constant across all videos, since just a single rate is used. Joint-Family, on the other hand, has the ability to adapt the playback rate to the available capacity for that video. Consequently, popular videos experience a high playback rate (as shown in Section 6.1.4). Interestingly, the average playback rate of the least popular videos is also much higher with Joint-Family (by ∼100Kbps) than the lowest possible rate. Similar to our analysis in Section 6.1.3, we also consistently see the benefit of caching and participating in multiple video swarms (e.g., $c = 5$ vs. $c = 1$). To understand whether the playback rate is sustained with minimal interruptions, we plot the average interruption time in Figure 6.8(b). Although the playback rate of BT VoD with 1000Kbps is always higher than Joint-Family, it causes significant interruption times. It is particularly bad for less popular videos where



(a) Playback Rate



(b) Interruption time



(c) Download rate

Figure 6.8: Comparison between Joint-Family and the state-of-the-art P2P system (group 1: the set of most popular videos).

the interruption can range from 100 to almost 400 seconds. In contrast, BT VoD with 250Kbps and Joint-Family result in comparably negligible interruptions; Joint-Family with $c = 5$ essentially performs as well as BT VoD at 250Kbps while still achieving significantly higher playback rates. To understand the reason for Joint-Family's improvement, we plot the average download rate achieved by each alternative in Figure 6.8(c). The different approaches achieve mostly similar download rate (although Joint-Family with $c = 5$ achieves higher throughput for unpopular videos) that decreases with decreasing popularity.

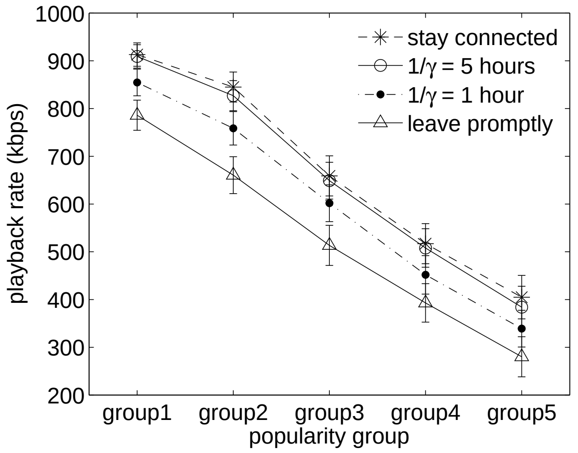


Figure 6.9: Effect of seed staying time ($1/\gamma$) for JF, $c = 5$

The combination of these results illustrates why it is important to adapt: If we pick too high a quality (e.g., 1000Kbps bitrate), users of less popular videos experience frequent interruption since the achievable download rate may be lower than the playback rate. Contrarily, if we pick a very low playback rate (250Kbps), interruptions may be minimized, but quality of popular videos is unnecessarily sacrificed. By dynamically adapting to the available capacity (as seen by the achieved playback rate for the different popularity groups), Joint-Family is able to achieve a nice balance between quality and interruptions. Moreover, we see that by caching more videos ($c = 5$), Joint-Family exploits the increased capacity and is hence able to deliver higher quality video at almost no interruptions across all types of videos.

We now study the effect of seed staying time in Joint-Family with $c = 5$. For viewers who are not currently watching any video, we vary their average staying time $1/\gamma$. In Figure 6.9, the 'leave promptly' curve indicates that all viewers leave the VoD network right after they finish watching, while the 'stay connected' curve (identical to 'JF, c=5' in Figure 6.8(a)) indicates that they stay connected till the end of each simulation. We first see that the playback results have a similar trend in that more popular swarms still achieve higher bitrates. Secondly, the improvement in playback rates with longer staying times reduces (e.g., '$1/\gamma = 5$ hours' and 'stay connected' are almost identical). This is because, unlike our analysis, viewers' arrivals do not strictly follow a Poisson process but

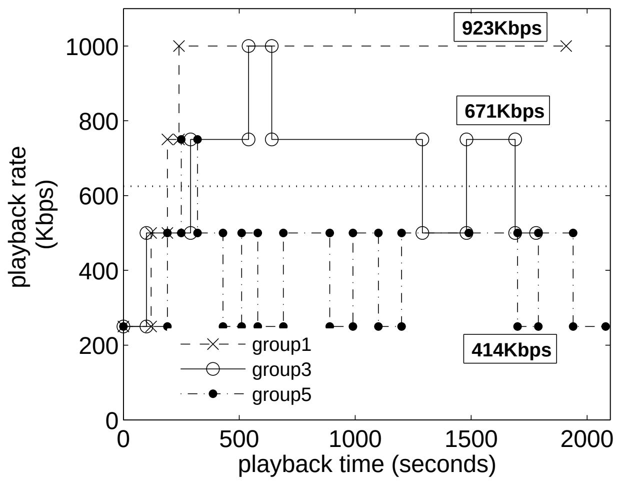


Figure 6.10: Rate adaptation with ABR

instead show significant diurnal patterns with busy periods (e.g., 8∼12 PM in Figure 6.12) and other periods that are less busy. Further, even after viewers leave, they can still come back to the network (e.g., to watch other videos) and have their previously viewed videos available for sharing. The interruption time (not shown here) is negligibly small for all cases.

6.3.4 Performance Improvement with ABR

We take a closer look at how a peer's playback experience evolves over a video streaming session. As Joint-Family adapts using ABR according to the available capacity for that video based on its popularity, we select a sample user from each popularity group and plot the playback rate over time as well as its overall average when $c = 5$. For the clarity of presentation, in Figure 6.10, we only show 3 groups. For the popular videos (Group 1), the video quickly ramps up to 1000 Kbps and stays at the rate to achieve an average playback rate of 923 Kbps, which is similar to the total average for Group 1 (as seen in Figure 6.8(a)). The Group 3 user also briefly goes up to 1000 Kbps before settling back down to 750 Kbps for the most part. In both these cases, the average playback is higher than the bitrate of non-ABR case (625 Kbps). Finally, since there is no sufficient capacity for the unpopular videos to support a high rate, the Group 5 user oscillates between 500 and 250 Kbps to achieve an average of 414 Kbps (while the group average is 410 Kbps). While this average

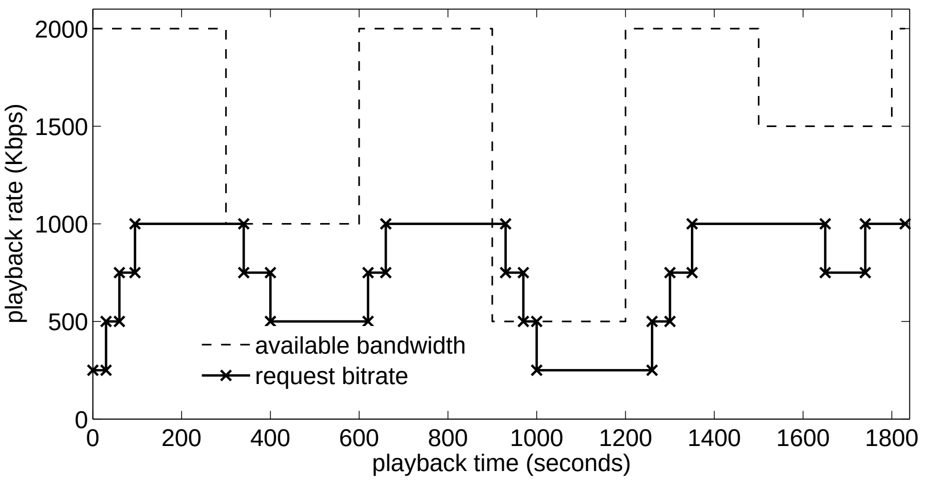


Figure 6.11: Rate adaptation when the link bandwidth available at a sample peer changes over time

is lower than 625 Kbps, the total interruption time for Group 5 with ABR was only 4.7 seconds compared to 20.6 seconds for the group without ABR.

We thus see that Joint-Family works harmoniously with ABR, enabling peers to dynamically adapt to the available system capacity among the servers and peers in the system for a particular quality/rate for the video. As seen in our ABR model in Section 6.1.4, by allowing peers to participate in multiple swarms, peers viewing a popular video are naturally able to take advantage of the higher bitrate chunks that become available because of the increased system capacity for such popular content.

Impact of Changes in Link Bandwidth Available: We also observe how the playback rate adapts as the link bandwidth available at a peer changes over its streaming session. We pick a sample peer from Group 1 who views a 30 minute video entirely, and we change the down-link bandwidth of the peer between 500 Kbps and 2 Mbps. In Figure 6.11 although we see a little delay between the link bandwidth change and the peer's bitrate adaptation, our simple buffer-based rate adaptation scheme results in the very similar transition to the bandwidth change. We also observe a few playback interruptions of the peer, about 8 seconds in total, during 1000∼1250 second period. The peer almost entirely relied on the server for 250 Kbps bitrate chunks during this period since most of the other peers in the

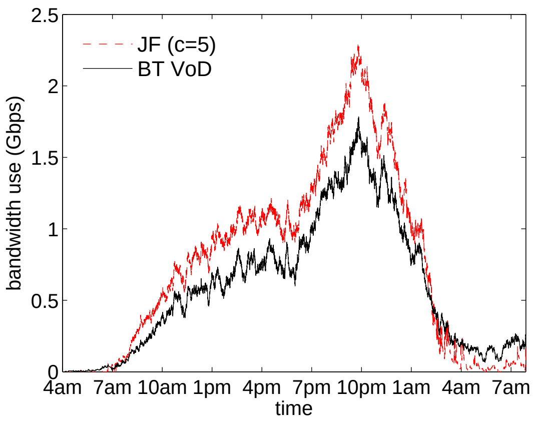


Figure 6.12: Aggregate upload bandwidth for Joint-Family and BT VoD

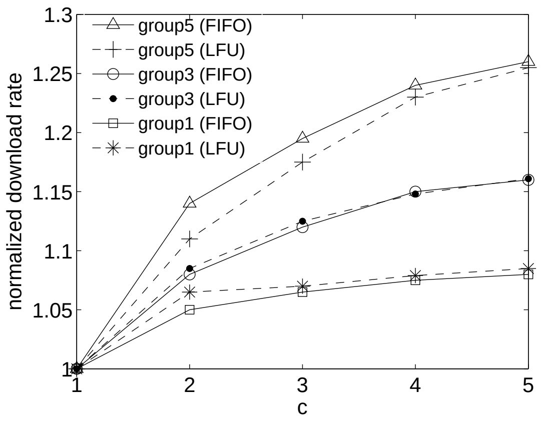


Figure 6.13: Video popularity and the effect of cache size (c)

same video swarm did not have 250 Kbps chunks available for this sample peer.

6.3.5 Effect of Multiple Swarms

To understand the underlying reason for the improved performance of Joint-Family, we examine the overall system utilization. We periodically sample the upload bandwidth aggregated across all peers (excluding servers) and report the time series for Joint-Family (with $c = 5$) and BT VoD. In this experiment we do not use ABR to remove the performance impact by rate adaption. Figure 6.12 shows Joint-Family effectively increases the system utilization compared to BT VoD. Specifically, at the peak viewing period, the aggregate upload bandwidth by BT VoD is 1.8 Gbps while 2.3Gbps with Joint-Family (an increase of 27%). By being in multiple swarms, peers in Joint-Family can use their upload capacity as long as they receive a chunk request from *any* of the swarms, thus improving overall upload capacity and playback experience.

Caching and video popularity: We turn our attention to increasing system capacity so that we can increase the video download rate through caching. We run Joint-Family with a constant bitrate of 625 Kbps and experiment with both LFU cache replacement (popular in the literature as a replacement policy for video caches) and FIFO (used in our analysis). Figure 6.13 shows the variation of the download rate as the cache size increases from 1 to 5 videos. We again pick 3 groups of videos with different popularity: Group 1, 3, and 5. The Y-axis shows the average download rate of each group normalized by the rate achieved when $c = 1$. Very similar to our analytical model in Section 6.1.3, we observe that: (a) caching consistently improves the download rate of peers across videos of all popularity levels, (b) the benefit from caching reduces as we increase the amount of caching, (c) unpopular videos see more benefit with caching than popular videos (>25% improvement compared to about 8%), and (d) the specific cache replacement mechanism does not play a significant role (in this limited size of the number of cache entries). Note that while the normalized download rates for popular videos improve less than unpopular videos, the absolute value for the download rate is much higher (1049 vs. 597 Kbps).

6.3.6 Impact of Chunk and Peer Selection Policies

We evaluate the contribution of the chunk selection and peer selection policies in Joint-Family. To perform this experiment, we started with BT VoD and first replaced the hybrid chunk selection policy with EF (TFT+EF, using the terminology of peer selection + chunk selection policies). We then replaced TFT peer selection with ED (ED+EF). A single bitrate, 625Kbps is used. Similar to Figure 6.8(c), the download rates for different policies are comparable and thus omitted here. Figure 6.14 shows that BT VoD experiences much longer interruptions compared to TFT+EF. Specifically, for the least popular group, the interruption time reduces by 33% when TFT+EF is used instead of BT VoD. This is because EF prioritizes chunks closest to the current playback point. In contrast, even though the download rate of BT VoD is similar to that of TFT+EF (not shown here), the partial use of RF in BT VoD results in many downloaded chunks that are not immediately useful. Next, when replacing TFT by ED (i.e., ED+EF), we consistently get further reductions in interruption times. In particular, the interruption time of ED+EF goes down by an additional 44% compared to TFT+EF. This can be attributed to the "fairness" aspect of ED, where we prioritize peers that really need the chunk soon, as opposed to TFT where peers unchoke other peers based on their upload rates.

To show this property, in Figure 6.15, we plot the cumulative distribution of the interruption time for each interruption the user experiences. We see that with ED, more than

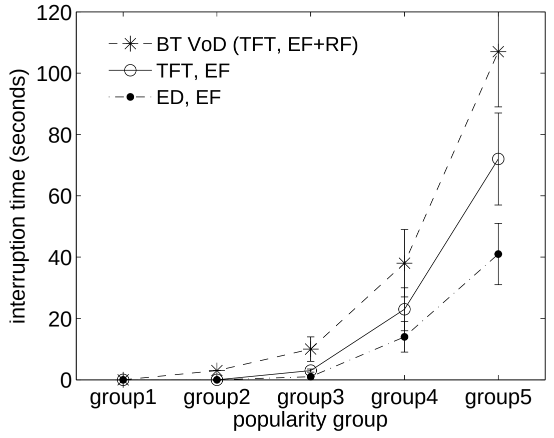


Figure 6.14: Interruption time with different chunk- and peer-selection policies

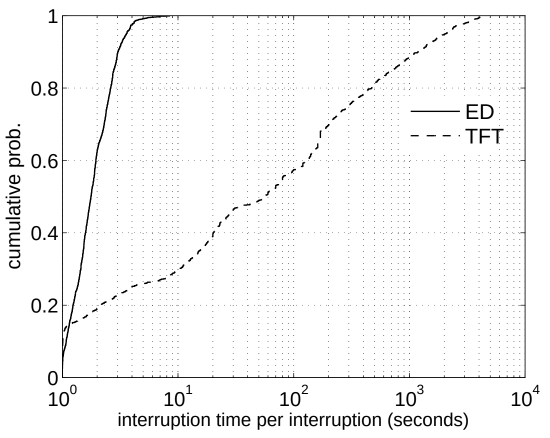


Figure 6.15: CDF of interruption time for each interruption

Policy	# of interruptions	Std. Dev
TFT	82.8	250.1
ED	112.1	220.5

Table 6.4: Interruptions count with ED and TFT

98% of interruptions last for less than 3 seconds, while 60% last even shorter (1 sec or less). The maximum interruption time is less than 10 seconds. On the other hand, with TFT there are much fewer short interruptions, while the majority of interruptions are long (40% last for $>$ 120 seconds). This result demonstrates inherent unsuitability of TFT with streaming video. We also examine the number of interruptions in Table 6.4. ED experiences more interruptions than TFT. However, since the duration of each interruption is significantly shorter, the overall total interruption time due to ED is very small. Additionally, the fact that 60% of interruptions last for 1 second or less suggests that the deadline we are using is extremely aggressive.

6.3.7 Effect of Heterogeneous Peers

In practice, the upload and download bandwidth of peers can vary, depending on network technology and pricing plans chosen by users. We examine the impact of varying the up-

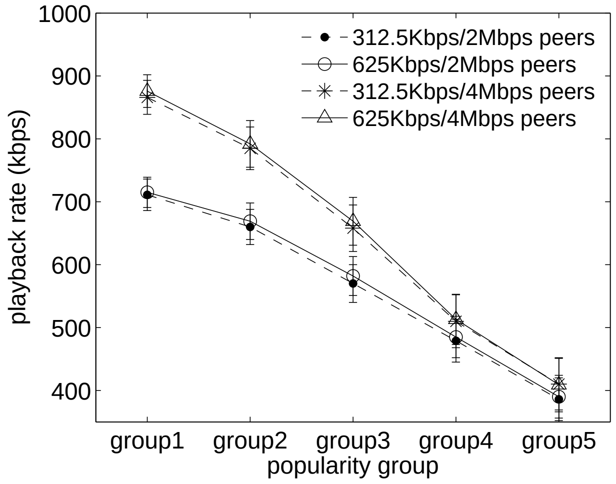


Figure 6.16: Playback rate of Joint-Family for heterogeneous peers

link and downlink bandwidth of peers. Instead of using homogeneous 625Kbps/2Mbps (up/down) bandwidth peers, we also consider the heterogeneous environment where peers have different link bandwidth. We choose 4 bandwidth combinations: 312.5K/2Mbps, 625K/2Mbps, 312.5K/4Mbps, and 625K/4Mbps, and each arriving peer has one of those bandwidth chosen uniformly at random. Figure 6.16 shows the playback rate for the corresponding peers. The benefit is predominantly seen for popular videos. Peers with higher downlink bandwidth see a greater improvement in the playback rate for their popular videos than when their uplink bandwidth changes. The higher downlink bandwidth allows the system (initially by the servers) to populate the environment (peers) with higher quality chunks (even if the uplink bandwidth is halved from 625 to 312.5 Kbps) which is then effectively shared among the peers viewing the popular video over time due to the increased system capacity for the popular video.

Part III

Conclusions and Future Work

Chapter 7

Conclusions

In order to overcome explosively increasing demands on online video streaming and to improve the streaming quality and performance, in this dissertation we investigated users' viewing patterns and developed novel scalable schemes based on the patterns that take into account a variety of practical considerations such as viewer abandonment, peers participating in multiple swarms, and adaptive bitrate streaming.

We first considered the problem of content placement in a backbone network. We took advantage of the belief that (a) users do not watch videos fully, but rather stop after skimming through the initial portion, and (b) users skip over portions of data. Using data from a nationally deployed VoD service, we showed that the belief is indeed true. We then developed a placement approach based on a Mixed Integer Program (MIP) that places segments of each video (chunks, or prefixes and suffixes) across locations in the backbone. Our MIP formulation requires the projected demand for each segment in order to compute the placement. We showed that we can use the popularity of TV shows and TV series to predict the demand for videos.

Using detailed simulations we showed that placing segments significantly outperforms alternate schemes while chunk-based placement yields best results. Importantly, MIP-based placement consistently resulted in lower bandwidth usage than LRU caching, even in an idealized setting for LRU caching. This is despite the fact that caching strategies can adapt quickly to changes in viewer request dynamics. We also showed that by just using prefixes we can obtain system performance close to what can be achieved by splitting the video into finer grained chunks. Therefore, our results demonstrate that our MIP-based placement approach is able to successfully take advantage of user behavior for placing content optimally.

Second, we demonstrated that user abandonment of videos can impact P2P VoD streaming performance significantly, by revisiting peer and chunk selection policies used in P2P VoD. Prior designs for P2P systems have been driven, to a great extent, by concerns over free riding (peer selection using Tit-for-Tat (TFT)) and downloading entire contents (chunk selection using Rarest-First (RF)). These concerns are overly constraining for a video streaming system with abandonment.

In all the schemes we considered in this dissertation, abandonment caused larger interruptions (NITs), particularly with peers watching longer as they are isolated with no other peers to upload from. With abandonment, distributing rare chunks (by RF or EF+RF hybrid) becomes wasteful and performs worse than Earliest-First (EF), as peers are likely to abandon before consuming the downloaded rare chunks. Through analysis and trace-driven simulations we showed that our scheme that combines Earliest-Deadline (ED) for peer selection, EF for chunk selection, and the use of partial seeds outperformed existing well-known schemes by significantly improving overall video playback performance and reducing wasted bandwidth consumption. Additionally, we further reduced wastage by peers having a playback lookahead window.

Finally, we presented a holistic redesign of P2P VoD called Joint-Family that for the first time supports the delivery of adaptive bitrate videos. We showed through analysis that only with sufficiently long staying times, the available download capacity in P2P VoD depends on the popularity of the content, and used this to guide our protocol design. Joint-Family achieved much better performance than other strategies as demonstrated by our simulations that used traces from a commercially deployed VoD service. By choosing ED as peer selection and EF as chunk selection policy, we dramatically improved the viewer's quality-of-experience (QoE) by minimizing interruptions. Joint-Family allowed peers to smoothly adapt their quality and achieve a high playback rate for popular content. Not only that, even for unpopular content, Joint-Family achieved almost 40% higher playback rate (350 Kbps) than an existing P2P VoD system with a fixed bitrate (250 Kbps). And it did this while reducing the total interruption time by a factor of 4 compared to the fixed bitrate P2P VoD approach. Joint-Family achieved this by leveraging resources across swarms that are potentially wasted by other schemes, and increased system utilization by 30% at peak viewing periods.

7.1 Future Work

There are several challenges and open questions for the directions of future research arising from our work in this dissertation which need to be pursued.

The results from our optimal video segment placement show that, while taking advantage of stream control functions like SKIP helps reduce the data transfers, the bulk of our reduction comes from users abandoning videos. This indicates that despite the availability of stream control operations, large parts of a video are watched sequentially. We postulate that when existing VoD interfaces start providing a more DVD-like navigational capability, we are likely to see skips playing an even bigger role. Similarly, although we observed that the performance difference between 1-minute chunks and 10-second chunks is modest, this may change with DVD-like navigation, and we may need to carefully choose the chunk size. Also, future work includes building a large-scale VoD system that takes advantage of our placement design.

Secondly, while TFT for P2P VoD systems introduces dependencies on chunk selection policies that are incompatible with P2P streaming, we showed that our approach in Joint-Family of using ED instead of TFT helped decouple the strict dependencies and offered better performance in terms of streaming and QoE compared to TFT. However, unlike TFT, ED does not guarantee against free riding which can significantly affect the overall P2P system performance and introduce unfairness. Although there are several deployments that do not worry about free riding such as managed content delivery [Maggs, 2012], for scenarios where eliminating free-riding is still important, we plan to investigate strategies to eliminate free-riding. We can adopt approaches like Contracts [Piatek *et al.*, 2010] or iPASS [Liang *et al.*, 2010], appropriately modified, for P2P VoD.

Contracts was designed for live streaming and hence relies on promoting users close to the source as the main incentive. While this incentive is not very useful for P2P VoD, we can leverage the other aspects of Contracts, i.e., exchanging receipts, using the tracker for verification and preventing collusion. Peers in Joint-Family can exchange similar receipts for contributing upload capacity. When requesting content, peers have to show proof that they have shared data with other peers in the form of receipts. Note that using receipts not only allows us to move from pair-wise exchange mechanisms towards one that allows a peer to carry credit for work done in sharing one video to fetching a different video. We are exploring these strategies as part of our future work.

Third, it would be possible to improve the bitrate adaptation policy of Joint-Family, by also considering chunk availability at each rate in addition to each peer's playback buffer status. For instance, for a particular portion of a video, if more peers have the chunk at bitrate r_i than at r_j, a leecher might prefer the chunk at r_i. We briefly explored this direction, but found that without careful design, peers can end up being stuck at lower bitrates even when there is capacity. This is because other peers may have downloaded lower bitrate chunks at a time when the swarm could only support that low rate. A sophisticated bitrate selection scheme that takes both chunk availability and video playback quality into account is an open area of research.

Another potential topic is unfairness between unpopular and popular videos in P2P [Hei *et al.*, 2007; Liu *et al.*, 2009]. Although Joint-Family greatly improved the streaming performance for unpopular videos offering almost no playback interruption, the average bitrate played back was still smaller compared to the one with popular videos. One possible approach to abate the unfairness is to be more in favor of viewers of less popular videos, and it could be divided into server-side and peer-side approaches. Both the server and peers should be able to keep tracking of video popularity, e.g., by simply recording the number of requests per video. Then, based on video popularity, the server can preferably upload to peers who request less popular videos. Similarly, a peer who has multiple videos in its local cache can also preferably upload less popular videos in its cache. Another simple peer-side strategy would be to ensure that peers keep less popular videos longer in their cache than more popular videos when their cached videos have to be evicted.

Finally, it would be beneficial to implement and deploy a functional prototype for Joint-

Family (i.e., one that goes beyond simulator-based experiments). Such prototype would enable us to refine our design. We can leverage existing open-source BitTorrent-like P2P VoD clients [BTC, 2013] to manage the low-level P2P operations, and focus on developing Joint-Family specific mechanisms.

Part IV

Bibliography

Bibliography

[Aalto *et al.*, 2011] Samuli Aalto, Pasi Lassila, Petri Savolainen, and Sasu Tarkoma. How Impatience Affects the Performance and Scalability of P2P Video-on-Demand Systems. In *SIGMETRICS MAMA*, June 2011.

[ado, 2010] Adobe HTTP Dynamic Streaming. `http://www.adobe.com/products/hds-dynamic-streaming.html`, 2010.

[Akhshabi *et al.*, 2011] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. In *ACM MMSys*, 2011.

[Alcock and Nelson, 2011] Shane Alcock and Richard Nelson. Application Flow Control in YouTube Video Streams. *ACM SIGCOMM Comput. Commun. Rev.*, 41(2):24–30, April 2011.

[Allen *et al.*, 2007] Matthew S. Allen, Ben Y. Zhao, and Rich Wolski. Deploying video-on-demand services on cable networks. In *Proceedings of IEEE ICDCS*, Toronto, Canada, June 2007.

[Alzoubi *et al.*, 2008] Hussein A. Alzoubi, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van der Merwe. Anycast CDNs revisited. In *Proceeding of WWW*, 2008.

[app, 2011] Apple HTTP Live Streaming. `http://developer.apple.com/resources/http-streaming/`, April 2011.

[Applegate *et al.*, 2010] David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and K. K. Ramakrishnan. Optimal content placement for a large-scale vod system. In *Proceedings of ACM CoNEXT*, CoNEXT. ACM, 2010.

[Baev *et al.*, 2008] Ivan D. Baev, Rajmohan Rajaraman, and Chaitanya Swamy. Approximation algorithms for data placement problems. *SIAM J. Computing*, 38(4):1411–1429, 2008.

[Banaji and Hardin, 1996] Mahzarin R. Banaji and Curtis D. Hardin. Automatic stereotyping. *Psychological Science*, 7(3):136–141, 1996.

[Barbir *et al.*, 2003] A. Barbir, B. Cain, F. Douglis, M. Green, M. Hofmann, R. Nair, D. Potter, and O. Spatscheck. Known Content Network (CN) Request-Routing Mechanisms. RFC 3568, July 2003.

[Benbadis *et al.*, 2008] Farid Benbadis, Fabien Mathieu, Nidhi Hegde, and Diego Perino. Playing with the Bandwidth Conservation Law. In *IEEE P2P '08*, Aachen, Germany, 2008.

[Bharambe *et al.*, 2006a] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. `http://www.research.microsoft.com/projects/btsim`, 2006.

[Bharambe *et al.*, 2006b] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Analyzing and Improving a BitTorrent Networks Performance Mechanisms. In *INFOCOM'06*, 2006.

[Borghol *et al.*, 2010] Youmna Borghol, Sebastien Ardon, Niklas Carlsson, and Anirban Mahanti. Toward Efficient On-Demand Streaming with BitTorrent. In *IFIP NETWORKING*, 2010.

[Borst *et al.*, 2010] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *Proceeding of IEEE INFOCOM'10*, pages 1478–1486, San Diego, California, USA, 2010.

[BTC, 2013] Comparison of BitTorrent clients. `http://en.wikipedia.org/wiki/BitTorrent_client`, 2013.

[Carlsson and Eager, 2007] Niklas Carlsson and Derek L. Eager. Peer-assisted on-demand streaming of stored media using BitTorrent-like protocols. In *Proceedings of IFIP NETWORKING*, NETWORKING'07, pages 570–581, Atlanta, GA, USA, 2007. Springer-Verlag.

[Carlsson *et al.*, 2009] Niklas Carlsson, Derek L. Eager, and Anirban Mahanti. Peer-assisted On-demand Video Streaming with Selfish Peers. In *Proceedings of IFIP NETWORKING'09*, Aachen, Germany, 2009.

[Castro *et al.*, 2003] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *ACM SOSP*, pages 298–313, 2003.

[Cohen, 2003] Bram Cohen. Incentives Build Robustness in BitTorrent. Technical report, bittorrent.org, 2003.

[com, 2010] Comcast VoD Choices Hits 25K. `http://www.lightreading.com/document.asp?doc_id=191777&site=lr_cable`, May 2010.

[D'Acunto *et al.*, 2010] Lucia D'Acunto, Nazareno Andrade, Johan Pouwelse, and Henk Sips. Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-Like VoD Systems. In *ISM*, pages 89–96, 2010.

[De Cicco *et al.*, 2011] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Feedback Control for Adaptive Live Video Streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 145–156, San Jose, CA, USA, 2011. ACM.

[Dobrian *et al.*, 2011] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *ACM SIGCOMM*, 2011.

[Eberhard *et al.*, 2012] Michael Eberhard, Amit Kumar, Licio Mapelli, Andi Palo, Riccardo Petrocco, and Mikko Uitto. NextSharePC: An Open-Source BitTorrent-based P2P Client

Supporting SVC. In Carsten Griwodz, editor, *Proceedings of the ACM Multimedia Systems Conference*, Chapel Hill, North Carolina, U.S.A., Feb. 2012. ACM.

[Fan *et al.*, 2010] Bin Fan, David G. Andersen, Michael Kaminsky, and Konstantina Papagiannaki. Balancing Throughput, Robustness, and In-Order Delivery in P2P VoD. In *Proc. CoNEXT*, December 2010.

[Ghobadi *et al.*, 2012] Monia Ghobadi, Yuchung Cheng, Ankur Jain, and Matt Mathis. Trickle: Rate Limiting YouTube Video Streaming. In *Proceedings of the USENIX Annual Technical Conference*, 2012.

[Gopalakrishnan *et al.*, 2010] Vijay Gopalakrishnan, Rittwik Jana, Ralph Knag, K. K. Ramakrishnan, Deborah Swayne, and Vinay Vaishampayan. Characterizing interactive behavior in a large-scale operational iptv environment. In *Proceedings of IEEE INFOCOM*, April 2010.

[Gopalakrishnan *et al.*, 2011] Vijay Gopalakrishnan, Rittwik Jana, K. K. Ramakrishnan, Deborah F. Swayne, and Vinay A. Vaishampayan. Understanding couch potatoes: measurement and modeling of interactive usage of IPTV at large scale. In *Proceedings of ACM Internet Measurement Conference (IMC'11)*, 2011.

[Guo *et al.*, 2007a] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. Does internet media traffic really follow zipf-like distribution? In *Proceedings of ACM SIGMETRICS*, 2007.

[Guo *et al.*, 2007b] Yang Guo, Saurabh Mathur, Kumar Ramaswamy, Shengchao Yu, and Bankim Patel. PONDER: Performance Aware P2P Video-on-Demand Service. In *GLOBECOM*, pages 225–230. IEEE, 2007.

[Hei *et al.*, 2007] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9, 2007.

[Huang *et al.*, 2007] Cheng Huang, Jin Li, and Keith W. Ross. Can internet video-on-demand be profitable? In *Proceedings of ACM SIGCOMM'07*, pages 133–144, Kyoto, Japan, 2007.

[Huang *et al.*, 2008] Yan Huang, Tom Z. J. Fu, Dah ming Chiu, John C. S. Lui, and Cheng Huang. Challenges, design and analysis of a large-scale p2p vod system. In *ACM SIGCOMM*, 2008.

[Huguenin *et al.*, 2010] Kevin Huguenin, Anne-Marie Kermarrec, Vivek Rai, and Maarten Van Steen. Designing a Tit-for-Tat Based Peer-to-Peer Video-on-Demand System. In *NOSSDAV*, 2010.

[Hwang *et al.*, 2008] Kyung-Wook Hwang, Vishal Misra, and Dan Rubenstein. Stored Media Streaming in Bittorrent-like P2P Networks. In *Columbia University Computer Science Technical Reports*, CUCS-024-08, 2008.

[joo,] Joost. `http://www.joost.com`.

[Legout *et al.*, 2006] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms Are Enough. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 203–216, Rio de Janeriro, Brazil, 2006. ACM.

[Lehrieder *et al.*, 2012] Frank Lehrieder, György Dán, Tobias Hossfeld, Simon Oechsner, and Vlad Singeorzan. Caching for BitTorrent-like P2P Systems: a Simple Fluid Model and its Implications. *IEEE/ACM Trans. Netw.*, 20(4), August 2012.

[Li *et al.*, 2011] Yuheng Li, Yiping Zhang, and Ruixi Yuan. Measurement and Analysis of a Large Scale Commercial Mobile Internet TV System. In *Proceedings of ACM IMC*, IMC '11, 2011.

[Liang *et al.*, 2010] Chao Liang, Zhenghua Fu, Yong Liu, and Chai Wah Wu. Incentivized Peer-Assisted Streaming for On-Demand Services. *IEEE Trans. Parallel Distrib. Syst.*, 21(9):1354–1367, September 2010.

[Liu *et al.*, 2009] Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. Why Are Peers Less Stable in Unpopular P2P Streaming Channels? In *Proceedings of the 8th Interna-*

tional IFIP-TC 6 Networking Conference, NETWORKING '09, pages 274–286, Aachen, Germany, 2009. Springer-Verlag.

[Liu *et al.*, 2011] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate Adaptation for Adaptive HTTP Streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, San Jose, CA, USA, 2011.

[Maggs, 2012] Bruce Maggs. A First Look at a Commercial Hybrid Content Delivery System. In *Keynote presentation at 15th IEEE Global Internet Symposium*, Orlando, FL, March 2012.

[Mol *et al.*, 2008] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems. In *Multimedia Computing and Networking 2008*, volume 6818, San Jose, USA, January 2008. SPIE Vol. 6818.

[pal,] Palm calculus. `http://en.wikipedia.org/wiki/Palm_calculus`.

[Park *et al.*, 2001] Seong-Ho Park, Eun-Ji Lim, and Ki-Dong Chung. Popularity-based partial caching for vod systems using a proxy server. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, 2001.

[Parvez *et al.*, 2008] Nadim Parvez, Carey Williamson, Anirban Mahanti, and Niklas Carlsson. Analysis of BitTorrent-Like Protocols for On-Demand Stored Media Streaming. In *SIGMETRICS*, 2008.

[Petrocco *et al.*, 2011] Riccardo Petrocco, Michael Eberhard, Johan A. Pouwelse, and Dick H. J. Epema. Deftpack: A Robust Piece-Picking Algorithm for Scalable Video Coding in P2P Systems. In *ISM*, 2011.

[Piatek *et al.*, 2010] Michael Piatek, Arvind Krishnamurthy, Arun Venkataramani, Richard Yang, David Zhang, and Alexander Jaffe. Contracts: Practical Contribution Incentives for P2P Live Streaming. In *NSDI*, 2010.

[poi, 2013] Global Broadband Statistics, 4Q 2012, Point Topic. `http://point-topic.com/free-analysis-type/subscriber-numbers/`, April 2013.

[ppl,] Pplive. `http://www.pplive.com`.

[Qiu and Srikant, 2004] Dongyu Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *ACM SIGCOMM 2004*, pages 367–378, Portland, Oregon, USA, 2004.

[Qiu *et al.*, 2001] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the Placement of Web Server Replicas. In *Proceeding of IEEE INFOCOM*, Anchorage, AK, April 2001.

[Rao *et al.*, 2011] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous. Network Characteristics of Video Streaming Traffic. In *CoNEXT*, 2011.

[Roverso *et al.*, 2012] Roberto Roverso, Sameh El-Ansary, and Seif Haridi. SmoothCache: HTTP-Live Streaming Goes Peer-to-Peer. In *IFIP NETWORKING*, Prague, Czech Republic, May 2012.

[Rückert *et al.*, 2012] Julius Rückert, Osama Abboud, Thomas Zinner, Ralf Steinmetz, and David Hausheer. Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics. In *IFIP Networking*, 2012.

[san, 2013] Sandvine Global Internet Phenomena Report (1H 2013). `http://www.sandvine.com/downloads/documents/Phenomena_1H_2013/Sandvine_Global_Internet_Phenomena_Report_1H_2013.pdf`, June 2013.

[Schwarz *et al.*, 2007] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. In *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, volume 17, pages 1103–1120, 2007.

[Sen *et al.*, 1999] Shubabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of IEEE INFOCOM*, pages 1310–1319, 1999.

[Shah and Paris, 2007] Purvi Shah and Jehan-Francois Paris. Peer-to-Peer Multimedia Streaming Using BitTorrent. In *Proceedings of the 26th IEEE International Performance Computing and Communications Conference, IPCCC 2007*, pages 340–347, New Orleans, Louisiana, USA, April 2007. IEEE Computer Society.

[ss, 2010] Microsoft Smooth Streaming. `http://go.microsoft.com/?linkid=9682896`, June 2010.

[Stockhammer, 2011] Thomas Stockhammer. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 133–144, San Jose, CA, USA, 2011.

[sun, 2010] Sun storage 6180 array. `http://www.oracle.com/us/products/servers-storage/storage/disk-storage/047193.html`, 2010.

[Tian and Liu, 2012] Guibin Tian and Yong Liu. Towards Agile Smooth Video Adaptation in Dynamic HTTP Streaming. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 109–120, Nice, France, 2012. ACM.

[uus,] UUSee. `http://www.uusee.com`.

[UVe,] AT&T U-verse. `http://www.att.com/shop/u-verse.html`.

[Valancius *et al.*, 2009] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. In *Proceedings of ACM CoNEXT*, 2009.

[Veciana and Yang, 2003] Gustavo De Veciana and Xiangying Yang. Fairness, Incentives and Performance in Peer-to-Peer Networks. In *the Forty-first Annual Allerton Conference on Communication, Control and Computing*, 2003.

[Vlavianos *et al.*, 2006] Aggelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *9th IEEE Global Internet Symposium 2006*, April 2006.

[Wang *et al.*, 2002] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley. Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In *Proceedings of IEEE INFOCOM*, 2002.

[Wang *et al.*, 2010a] Miao Wang, Lisong Xu, and Byrav Ramamurthy. Linear Programming Models for Multi-Channel P2P Streaming Systems. In *INFOCOM'10*, San Diego, California, USA, 2010.

[Wang *et al.*, 2010b] Zhi Wang, Chuan Wu, Lifeng Sun, and Shiqiang Yang. Strategies of Collaboration in Multi-Channel P2P VoD Streaming. In *GLOBECOM*, Miami, Florida, 2010. IEEE.

[Wen *et al.*, 2011] Zheng Wen, Nianwang Liu, Kwan L. Yeung, and Zhibin Lei. Closest Playback-Point First: A New Peer Selection Algorithm for P2P VoD Systems. In *GLOBECOM*. IEEE, 2011.

[Wu and Li, 2009] Jiahua Wu and Baochun Li. Keep Cache Replacement Simple in Peer-Assisted VoD Systems. In *Proceedings of IEEE INFOCOM*, pages 2591–2595, Rio De Janeiro, Brazil, April 2009.

[Wu *et al.*, 2001] Kun-Lung Wu, Philip S. Yu, and Joel L. Wolf. Segment-based Proxy Caching of Multimedia Streams. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 36–44, Hong Kong, 2001. ACM.

[Wu *et al.*, 2009] Di Wu, Yong Liu, and Keith W. Ross. Queuing Network Models for Multi-Channel P2P Live Streaming Systems. In *INFOCOM 2009*, pages 73 –81, April 2009.

[Yang *et al.*, 2010] Yan Yang, Alix L. H.Chow, Leana Golubchik, and Danielle Bragg. Improving QoS in BitTorrent-like VoD systems. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 2061–2069, Piscataway, NJ, USA, 2010. IEEE Press.

[Yin *et al.*, 2009a] Hao Yin, Xuening Liu, Feng Qiu, Ning Xia, Chuang Lin, Hui Zhang, Vyas Sekar, and Geyong Min. Inside the Bird's Nest: Measurements of Large-Scale

Live VoD from the 2008 Olympics. In *Proceedings of the ACM SIGCOMM Internet measurement conference*, 2009.

[Yin *et al.*, 2009b] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *Proceedings of ACM Multimedia*, pages 25–34, 2009.

[Yu *et al.*, 2006] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings ACM SIGOPS/EuroSys*, 2006.

[Zhang *et al.*, 2000] Zhi-Li Zhang, Yuewei Wang, David H. C. Du, and Dongli Su. Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks. *IEEE/ACM Transactions on Networking*, 8(4), August 2000.

[Zhang *et al.*, 2011] Honggang Zhang, Sudarshan Vasudevan, Ran Li, and Don Towsley. A Case for Coalitions in Data Swarming Systems. In *2011 IEEE International Conference on Network Protocols (ICNP)*, Oct. 2011.

[Zhou and Xu, 2002] Xiaobo Zhou and Cheng-Zhong Xu. Optimal Video Replication and Placement on a Cluster of Video-on-Demand Servers. In *Proceedings of the 2002 International Conference on Parallel Processing*, ICPP '02, pages 547–, Washington, DC, USA, 2002. IEEE Computer Society.

[Zhou *et al.*, 2007] Yipeng Zhou, Dah Ming Chiu, and J.C.S. Lui. A Simple Model for Analyzing P2P Streaming Protocols. In *2007 IEEE International Conference on Network Protocols (ICNP)*, pages 226 –235, Oct. 2007.

[Zhou *et al.*, 2011] Xia Zhou, Stratis Ioannidis, and Laurent Massoulie. On the Stability and Optimality of Universal Swarms. In *SIGMETRICS Perform. Eval. Rev.*, volume 39, pages 301–312, New York, NY, USA, 2011. ACM.