Joint-Family: Adaptive BitRate Video-on-Demand Streaming over Peer-to-Peer Networks with Realistic Abandonment Patterns.

Kyung-Wook Hwang[a], Vijay Gopalakrishnan[a], Rittwik Jana[a], Seungjoon Lee[b], Vishal Misra[c], K. K. Ramakrishnan[d], Dan Rubenstein[c]

[a]*AT&T Labs – Research, Bedminster, NJ 07921, United States*
[b]*Two Sigma Investments, New York, NY 10013, United States*
[c]*Dept. of Computer Science, Columbia University, New York, NY 10025, United States*
[d]*Dept. of Computer Science and Engineering, University of California, Riverside, CA 92521, United States*

Abstract

Previous studies of peer-to-peer (P2P) video-on-demand (VoD) are performed separately from studies utilizing adaptive bit rate video since the techniques seemingly tackle orthogonal goals. Additionally, previous policies used by P2P VoD do not account for viewer abandonment of video during download and playback. Through analysis, we show that the popularity of a P2P swarm and seed staying time significantly affects the achievable per-receiver download rate. Specifically, we identify conditions under which popularity affects swarm efficiency, contradicting a typical misconception in previous work, and we show that abandonment under these previous policies significantly increases playback interruptions. In light of these observations, we propose Joint-Family, a protocol that supports HTTP-based adaptive bitrate streaming for on-demand videos using P2P techniques. Joint-Family accounts for user video viewing behavior, such as abandonment, to improve the quality of experience for the viewer. Peers in Joint-Family simultaneously participate in multiple swarms to exchange chunks of different bitrates. Joint-Family takes advantage of abandonment by converting peers to "partial seeds"; this increases system capacity. Joint-Family adopts chunk, bitrate, and peer selection policies that minimize occurrence of interruptions in the presence of abandonment 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.

Keywords: Peer-to-peer video-on-demand, HTTP adaptive streaming, viewer engagement, abandonment

1. Introduction

The ever-increasing demand placed by streaming video traffic across both wired and wireless networks has been managed by two seemingly complementary approaches: HTTP-based adaptive bitrate (ABR) [1, 2], and peer-to-peer (P2P) delivery [3–5]. ABR encodes a video at multiple bitrates, and maximizes the video bitrate within the available bandwidth, delivering a higher fidelity video over HTTP when possible, and dropping to lower quality rather than causing an interruption of playback of the excessively high bitrate video. MPEG-DASH [6] is an ISO/IEC standard for ABR. While layered coding such as scalable video coding (SVC) [7] is another approach for adaptive streaming, SVC has had difficulty in being implemented in the real world due to coding complexity and bitrate overhead (discussed more in Section 5). We focus on ABR in this work.

P2P-based systems are a popular alternative to deliver on-demand video, improving the viewing experience by utilizing the uplink capacity of the downloading peer nodes, thereby increasing overall system upload capacity. Even traditional Content Distribution Network (CDN) providers such as Akamai are experimenting with and deploying P2P-based delivery of video content [8].

Intuitively, P2P and ABR seem poorly suited to work together, because peer viewers watching the video at differing rates are presumably unable to exchange video segments (chunks) with one another. Thus, intuition suggests that enabling ABR reduces the peers' ability to share video chunks with one another. We show in this paper that, contrary to current intuition, ABR and P2P effectively combine to leverage both of their strengths: P2P techniques improve upload capacity, and ABR enables the highest quality viewing at that capacity while minimizing interruptions.

In addition to the ABR and P2P combination, we focus on a user viewing pattern called **viewer abandonment**

Email addresses: kwhwang@research.att.com (Kyung-Wook Hwang), gvijay@research.att.com (Vijay Gopalakrishnan), rjana@research.att.com (Rittwik Jana), Seungjoon.Lee@twosigma.com (Seungjoon Lee), misra@cs.columbia.edu (Vishal Misra), kk@cs.ucr.edu (K. K. Ramakrishnan), danr@cs.columbia.edu (Dan Rubenstein)

since it substantially impacts the performance of video streaming, especially on P2P-based systems. This viewing pattern indicates that viewers abandon a video part way through the viewing of the video. Such abandonment may be attributed to user behavior (e.g., surfing for interesting content) or due to loss of interest in the currently viewed content.

While mostly overlooked so far, we show that abandonment (also called viewer engagement in other work [9, 10]) is a critical factor to consider since it directly affects the impact of various policies used for P2P Video-on-Demand (VoD). P2P file sharing systems have traditionally used a combination of Tit-for-Tat (TFT) for peer selection and Rarest-first (RF) for chunk selection. Unfortunately, this combination does not work as well with streaming video, 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 not ideally compatible with TFT as peers at different points in the playback have very little content to exchange with each other. As a result, prior work to identify hybrid of EF and RF (EF+RF) [11–15] explore the compromise between the need of streaming to get sequential data and TFT's need for diversity.

However, we show that, due to abandonment, peers using EF+RF will download rare chunks that they do not actually watch later. In this case, it would be more beneficial to use that upload capacity to deliver chunks that have to be played soon, to improve video playback experience and reduce unnecessary bandwidth consumption.

Here, we present a novel system called **Joint-Family** that combines P2P and ABR to provide high-quality streaming[1] VoD in the presence of viewer abandonment. Joint-Family's design is based on our analysis that uses a Markov model, where we identify the relationship between video popularity, seed staying time and download rate. We show that when seeds stay "sufficiently long", *content popularity affects swarm efficiency* (Section 2.2). This implies that swarms of popular videos can have higher download rates than less popular ones if seeds stay long enough. This is in contrast to existing fluid modeling results [16] which claim independence between popularity and download capacity. Hence, we identify the conditions under which prior results are contradicted. We then analyze the effectiveness of caching previously viewed videos and sharing them, as a mechanism to extend seed staying time (Section 2.3). Caching also enables transfer of underutilized capacity from one swarm to another, improving global performance. Finally we show how ABR, when combined with P2P, enables a swarm to efficiently adapt to the best video rate without *a priori* knowledge of the video's popularity (Section 2.4). Our Markov model shows that ABR allows P2P swarms to migrate to the highest sustainable rate for that swarm: highly popular content will enable large swarms and have a high, sustained download capacity, whereas less popular content will have smaller swarms and a lower sustainable capacity.

Based on our analytical observations, we design Joint-Family to deliver high-quality videos with minimal playback interruptions in a P2P system, using multi-swarm participation and ABR (in Section 3). 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 combination of chunk selection, peer selection, and bitrate adaptation policies that minimize interruptions with viewer abandonment. We show that EF is a more appropriate chunk selection strategy than hybrid of EF and RF when abandonment exists. 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 they request when deciding which request to serve. Choosing ED not only substantially improves performance (as seen in our experiment results), but also breaks the inter-dependence between chunk- and peer selection that TFT introduces. Because in practice, abandonment of a video does not imply departure from the P2P system, peers can be classified as **partial seeds**, who do not have the entire video, and are not actively watching (and downloading) the video, yet are still connected. Our system permits partial seeds to continue to serve requests for the chunks they have downloaded already, and thus contribute to increasing the overall system capacity.

Our design makes Joint-Family immediately suitable for existing VoD infrastructures in which the provider owns the distribution infrastructure (e.g., CDNs [8], IPTV [17]). We also describe how the protocol can be applied in a decentralized setting by utilizing mechanisms that encourage sharing of content [18]. We conduct extensive performance evaluations of Joint-Family using traces from a nationally deployed VoD service (in Section 4), and show that ABR with P2P is indeed feasible, even when abandonment occurs. 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.

2. Analysis of P2P Systems for VoD

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 (ABR) techniques can significantly improve the playback experience even for unpopular content.

[1] As opposed to download-and-play

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

Table 1: Parameters and Definition

2.1. Assumptions

The notations used in our model are summarized in Table 1. We use the leech (i.e., downloader) 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 leech's download (streaming) rate can be faster than the video playback rate for potentially fewer playback interruptions. We assume that each leech watches a video till the end, and thus seeds have the entire video. However, all our experiments in Section 4 also account for viewers' premature abandonment based on real traces. Similar to other P2P studies [13, 19] and based on the wireline subscriber statistics [20], 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 4.9.

2.2. Popularity, Download Rate, and Seed Staying Time

The fluid model based analysis by Qui and Srikant [16] suggests that the download performance of files is relatively independent of their popularity. They explain that the supply and the demand placed by leeches are always offset regardless of video popularity. There has been subsequent work [19, 21] 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.

Fluid models assume deterministic arrivals of requests, which likely holds when a video is highly popular (i.e., the request arrival rate going to infinity). However, when the request arrivals are stochastic — as seen in practice — a version of Feller's paradox takes place, and Palm calculus [22] 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 leeches monotonically increases, since we assume the seeds stay for a sufficiently long time and many active leeches 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 leeches and seeds in a swarm. Our model is motivated by a two-dimensional (2D) model by Veciana and Yang [23] (which only presents recursive relationship but no explicit formula). 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 leeches and download time.

Given y seeds, consider a Markov chain, where each state corresponds to the number of leeches (x). 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. [13], where all leeches except the latest arrival can always upload to other leeches. 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 leeches [23]. This parameter is experimentally shown to be close to 1 for most practical cases [16, 19], and we use $\eta = 1$ in the rest of the paper. Then we obtain the following recursive equation for the steady-state probability of state $i \geq 1$:

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

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

$$\pi_0 = \frac{1}{\frac{(y-1)!}{\rho^{(y-1)}}(e^{\rho} - \sum_{i=0}^{y-2}\frac{\rho^i}{i!})} \tag{2}$$

Recall that the steady state probability is under the condition for a particular y. Using Equations (1) and (2) we can obtain the conditional expectation as follows:

$$E[X|Y=y] = \rho - y + 1 + \frac{e^{-\rho}\rho^{(y-1)}}{\Gamma(y-1) - \Gamma(y-1,\rho)} \tag{3}$$

where $\Gamma(y)$ is the gamma function ($\Gamma(y) = (y-1)!$) and $\Gamma(y,\rho)$ is the upper incomplete gamma function ($\Gamma(y,\rho) = (y-1)!e^{-\rho}\sum_{i=0}^{y-1}\frac{\rho^i}{i!}$). See Appendix Appendix A for the detailed derivation.

Now, let us consider the distribution for the number of seeds (y). A seed arrival is equivalent to a leech completing download of the video. As a result, at steady-state, the leech 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

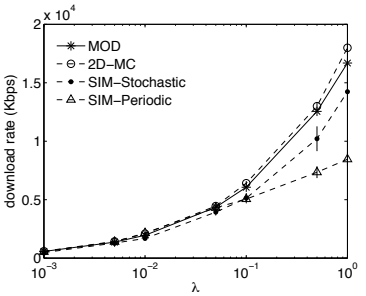


Figure 1: Download rate as a function of λ with $1/\gamma = 3600$ secs. (X axis in log scale)

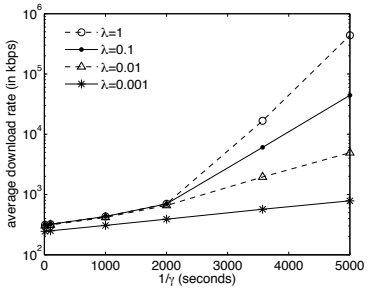


Figure 2: Download rate as a function of seed staying time $1/\gamma$. (Y axis in log scale)

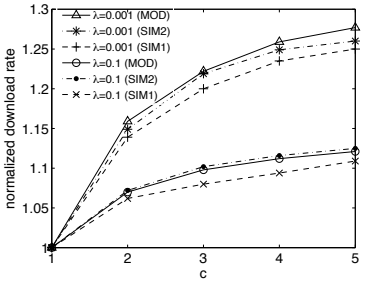


Figure 3: Caching multiple videos: each download rate is divided by the download rate when $c = 1$. ($1/\gamma = 3600$)

this with (3), we get:

$$E[X] = \sum_{y=0}^{\infty} \left(\rho - y + 1 + \frac{e^{-\rho}\rho^{(y-1)}}{\Gamma(y-1) - \Gamma(y-1,\rho)} \right) \frac{e^{-\sigma}\sigma^{y}}{y!} \tag{4}$$

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

Evaluation: We numerically evaluate (4) 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 4.2 for detail). In our experiments, we consider an 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 [23] 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 leeches and compute the downloading rate. We use a similar warm-up strategy for our event-driven experiments.

Fig. 1 shows the average download rate of leeches for different λ with $\frac{1}{\gamma}$ =1 hour. We compare four cases: simulated transition on the 2D Markov model [23] (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 [16, 19]. 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, 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 that the leech arrival pattern also plays a critical role in the download performance, and the assumption of periodic arrivals in the fluid models [16, 19] can lead to incorrect conclusions in practical scenarios.

We next investigate the effect of seed staying time on download performance. In Fig. 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 [16, 19]. 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).

2.3. Caching to Increase Staying Time and Download Rate

As seen in Section 2.2, 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 performance.

To analyze the benefit of caching, we 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 2.2 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 leech 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:

$$q^{c}_{i,i-1} = (i + y/c - 1)u/B \tag{5}$$

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:

$$P[Y = y] = e^{-\sigma_c} \frac{\sigma_c^y}{y!} \tag{6}$$

where $\sigma_c = c\lambda/\gamma$. From (5) and (6), we can obtain the average download time T by following the similar derivation as in Section 2.2. In our numerical evaluation of $E[X]$ with $c > 1$, we substitute y in Equation (3) 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 2.2 with a synthetic trace. Fig. 3 plots the normalized average download rate from our analysis and from the simulation for different cache size c. In SIM1, we simulate exactly the policy described in deriving Equation (5) for precise validation.In SIM2, we show the results without our assumption so that a leech actively watching a video also uploads all other videos in its cache. We first observe that the analysis (MOD) and simulation results (SIM1, SIM2) match well. Also, caching is more beneficial with small λ (i.e., less popular videos). We then see the diminishing returns as c grows since our small u which is the bottleneck quickly becomes more utilized (thus, we omit the results for $c > 5$). Finally, we show that the download rate in SIM2 only improves as we remove our assumption. In Section 4.8 we explore different cache replacement schemes such as LFU using real-world traces.

2.4. Adaptive Bitrate Analysis

We showed in Section 2.2 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 Fig. 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).

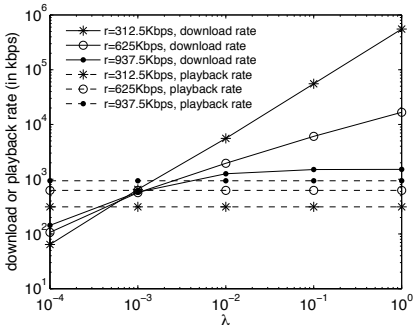


Figure 4: Download/playback rate vs. arrival rate with different chunk bitrates

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. But 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 leech only increases the video bitrate to reach the highest bitrate it can sustain. Specifically, each leech 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 3.3). If a leech 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 leeches for a given video go through the same set of "transition points" in a steady state. In other words, all leeches 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 correspond-

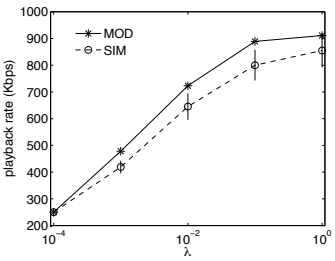


Figure 5: Validation of ABR analysis using simulation

ing 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 2.2. Each state is the number of leeches 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:

$$q^k_{i,i-1} = (i + f_k y - 1)u/\tilde{B}_k, \tag{7}$$

where $f_k = \frac{\tilde{B}_k}{\sum_j \tilde{B}_j}$. Then, following the analysis for each Markov chain in Section 2.2 (Equation (4)), 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 $B' = (B'_1, \ldots, B'_m)$.

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 an 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 3.3 for detail). Fig. 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. In Section 4.6 and 4.7 we confirm this by the experiments with larger trace with multiple videos of different popularity.

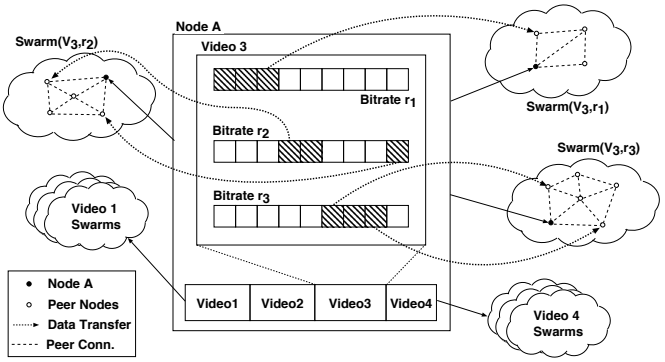


Figure 6: A peer in Joint-Family participates in multiple swarms.

3. JOINT-FAMILY Design

Recent studies [10, 24, 25] have shown that users abandon videos before viewing them in their entirety. Our results show that 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. We take the learnings from our analysis in Section 2 to design a P2P protocol that supports the delivery of high quality video using HTTP adaptive bitrate (ABR) schemes. To the best of our knowledge, Joint-Family is the first practical P2P VoD system that incorporates ABR.

3.1. 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 2.3 and 2.4 has a beneficial effect on the overall download performance and playback rate for ABR videos. Fig. 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.

3.2. 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 leech or a seed. In the latter case, the connection is altruistic 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). When a peer participates in multiple swarms for multiple videos, there is an inherent tension between the number of selfish connections and altruistic ones.[2] Specifically, if the 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 leech is at most $n\alpha_l$. In Joint-Family, a leech 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.

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 leech unchokes those peers that sent the leech the most chunks, while a seed unchokes those peers that can download the fastest. In Joint-Family, a peer can simultaneously be a leech (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 leech has no incentive to use upload capacity for altruistic connections. This is because the leech 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 3.4.

Caching and sharing multiple videos: As shown in Section 2.2, increasing seed staying time in a swarm increases the capacity of the swarm. Our approach to increase staying time is, as modeled in Section 2.3, 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 4.8, experimental results on the benefits of caching are presented.

3.3. Chunk Selection and Rate Adaptation

Chunk selection: The chunk selection policy determines the order in which a peer viewing a video downloads chunks of that video. While Rarest-First (RF) has been the de-facto standard chunk selection policy for file sharing systems, RF is inherently unsuitable for streaming systems which desire chunks to arrive in order [13]. ABR further complicates this, as the rarest chunk at the time of download may not match the right bitrate at the time of playback. Instead, Earliest-First (EF) that attempts to get chunks close to playback is a more natural fit for streaming. In fact, a significant amount of previous work [11–15] 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 by EF+RF schemes, 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. In Joint-Family, we use EF chunk selection, and we show in Section 4.3 that EF outperforms EF+RF when abandonment exists. Also, with buffer-based rate 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. While we do not address it in this paper, EF is also amenable to DVD-like operations. 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 [13]. We argue that this throughput collapse is a side-effect of using EF with Tit-for-Tat peer selection policy. Further, the performance degradation highly depends on the number of seeds in the

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

swarm, and our caching mechanism helps avoid the "missing piece syndrome" [26].

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 [27, 28], we have designed Joint-Family to use hysteresis when making a change in the bitrate, so that the quality does not change too frequently, thereby providing the user a better quality-of-experience (QoE). A leech uses a simple rate 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 this with hold-down timers to avoid rapid bitrate fluctuations. Specifically, a peer increases the bitrate if 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 secs ago. Contrarily, a peer decreases the bitrate if (1) its buffer has less than s_d secs of chunks, and (2) the last downward rate change was more than h_d secs ago. In Section 2.4 we validated that the simulation result using this scheme closely follows the ABR model's result (see Figure 5).

A possible improvement in rate 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 leech 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 rate selection scheme that takes both chunk availability and video playback quality into account is still an open area of research.

3.4. Earliest-deadline (ED) Peer Selection

The peer selection policy determines the subset of requests that a peer serves upon receiving requests. While most P2P systems use tit-for-tat (TFT) as the peer selection policy, TFT requires that peers have content to exchange with each other and works best when peers have a diverse set of chunks. This aspect creates an implicit interdependency between the chunk selection and peer selection policies. Specifically, TFT works well with RF, as RF is designed to create such chunk diversity. However, there is growing realization that there are inefficiencies due to TFT [18, 29] in streaming systems, particularly with regard to interruptions.

With EF, however, peers at different points of their playback will not have content of mutual interest to exchange with each other. 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 leech) receiving requests from multiple connected peers during a certain interval chooses to serve the peer with the earliest deadline (with ties broken at random). 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 4.3 that ED performs substantially better than TFT with respect to quantitative metric such as interruption time.

Switching to ED, we also consider the following aspects.

Choking peers: Unlike TFT, a peer does not choke another in ED. This brings up new protocol aspects to be addressed. First, as a provider, a peer may receive upload requests from all of its connected peers. To ensure that the per-chunk upload rate does not become too small, we limit the number of concurrent uploads from a peer. Second, when it is not choked, a downloading peer can have a large number of parallel downloads, and the per-chunk download rate can greatly decrease, resulting in longer start-up delays and frequent interruptions. In many cases, all the downloads may share a single downstream bottleneck link to that peer. To address this, each peer adjusts the 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 an additional download has the potential to decrease the speed of the ongoing downloads.

Handling free riding: Free riders in P2P systems can significantly impact the overall system performance and introduces 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. While using ED instead of TFT does not protect against free riders, ED offers better performance in terms of streaming and QoE compared to TFT. The decoupling of peer selection from chunk selection allows us to overcome inefficiencies due to TFT [18, 29] in deployments that do not worry about free riding (e.g., managed content delivery [8]). In scenarios where eliminating free-riding is of concern, we can use the mechanisms proposed in Contracts [18], modified appropriately for P2P VoD, to incentivize peers to share content.

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.

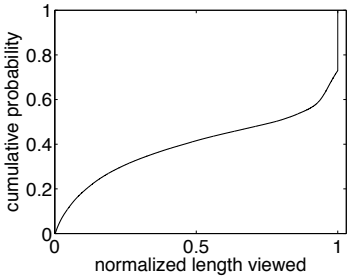


Figure 7: Cumulative distribution of normalized length viewed of all requests in trace data

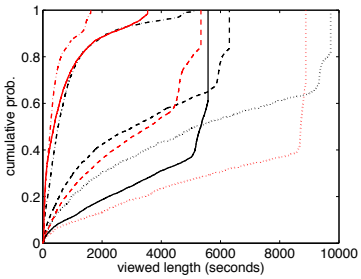


Figure 8: Cumulative distribution of viewed length for 8 videos used in Sec. 4.3 and Sec. 4.4

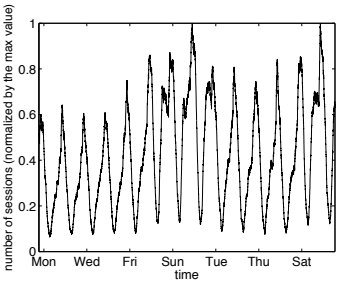


Figure 9: Number of concurrent sessions for a random video in the real trace.

3.5. 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. This time period measured in our real trace will be presented in Figure 17.

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 experiments in Section 4.4 shows that by having partial seeds stay longer in the system, the performance can be improved significantly.

4. 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 interruption time. We then show how each of the design policies contributes to improving system performance.

4.1. Data Set with User Abandonment

To reflect realistic viewing patterns of a large population of users, trace from a nationally deployed VoD service is used. The trace covers a two week period with millions of requests. The data contains information including the anonymized user ID, request time, video ID, video length, and the duration viewed for each session.

Parameter	Default value
Number of initial seeds (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 seconds
Startup buffer size per peer	10 seconds

Table 2: Simulation parameters

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). In Fig. 7, we show the extent of user abandonment by plotting the cumulative distribution (CDF) of normalized length viewed. To compare abandonment for movies 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.

To first understand the impacts of abandonment (Sec. 4.3 and Sec. 4.4), 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 Fig. 8. They also show a clear daily pattern in their request volume. For example, Fig. 9 shows the number of concurrent sessions (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 the 8 videos.

Then, with abandonment existing, we evaluate our system with ABR video delivery. For the experiments (Sec. 4.5 $\sim$ 4.9), we include playback results from all requested videos in the entire trace as well as the 8 videos. 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.

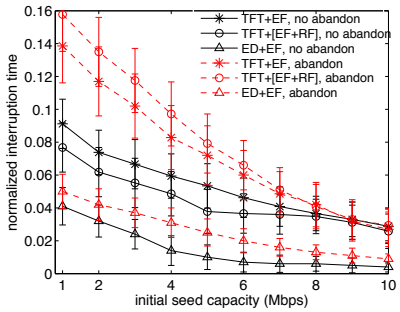


Figure 10: Comparing NITs of different combination of chunk- and peer- selection policies on Joint-Family when user abandonment exists and when it does not (with 95% confidence interval).

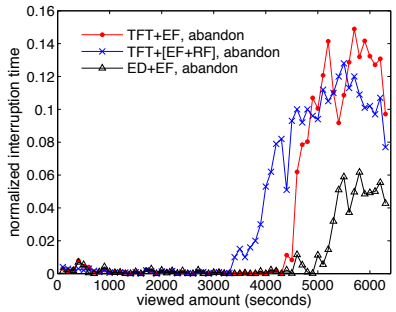


Figure 11: Average NITs of each peer group divided based on viewed length. The server (initial seed) capacity is 3Mbps.

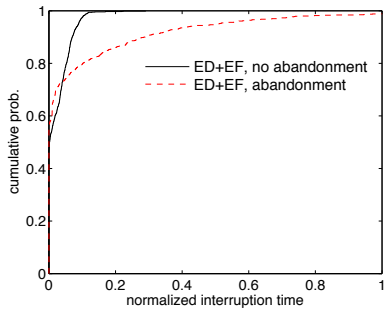


Figure 12: Cumulative distribution of NITs on Joint-Family when user abandonment exists and when it does not, respectively. (3Mbps initial seed capacity)

4.2. Experiment Setup

To evaluate Joint-Family, the BitTorrent simulator [30] is used with the following major modifications: (1) video streaming support (e.g., playback buffer), (2) bitrate adaption (Section 3.3), (3) multi-swarm participation (Sec. 3.2), (4) different chunk (Sec. 3.3) and peer selection policies (Sec. 3.4), and (5) partial seeds (Sec. 3.5). 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.

We summarize the different parameters used in the simulations in Table 2. We use 5 servers, each with 25Mbps uplink capacity to host all the videos. 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. Peers that download the entire video convert to *normal seeds* while those that abandon the video part-way become *partial seeds*. For the trace driven simulation, peers make requests for a video at the time instants specified in the trace.

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 $s_d = 20$ secs. 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 for performance evaluation.While the former gives information about the quality of video viewed by the user, the latter captures the aggregate disruptions experienced by the user.

4.3. Impact of Abandonment

We first investigate how user abandonment affects the performance of different combinations of chunk selection and peer selection policies with Joint-Family. 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 this sub-section, we test each single video per experiment and make the following adjustment on the simulation parameters of Joint-Family: we use (1) only one initial seed (server), (2) 1Mbps non-ABR bitrate, and (3) 5Mbps/1Mbps peer download/upload bandwidth. We vary the server capacity in Fig. 10 and accordingly adjust the maximum number of its concurrent uploads allowed (e.g., 10 concurrent uploads with 2Mbps upload capacity, 20 with 4Mbps, etc.).

Fig. 10 shows that all three schemes (TFT+EF, TFT+[EF+RF], ED+EF, using the terminology of peer selection + chunk selection policies) have larger NITs with abandonment existing 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 [9, 24]. Clearly, a higher server 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 actually causes larger NITs compared to EF when abandonment exists, because with abandonment, chunks closer to the end of a video are viewed rarely. Exchanging rare chunks (typically later parts of the video) which are not watched results in inefficient use of resources. We observe that our proposed ED+EF combination of Joint-Family has the smallest NITs. This shows

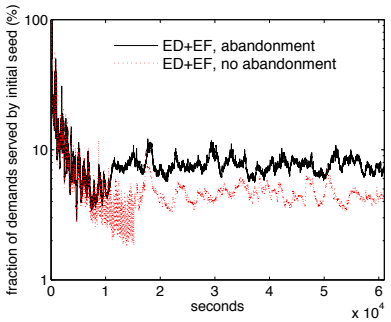


Figure 13: The fraction of demand (in terms of bytes) satisfied by Joint-Family server (initial seed) over time.

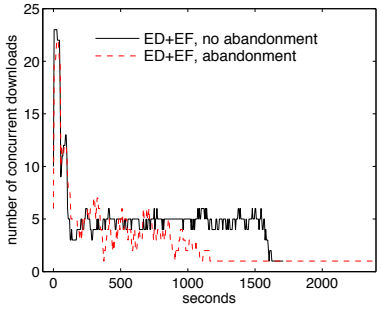


Figure 14: Number of concurrent downloads of a peer in Joint-Family over time. The solid curve ends at 1610 seconds.

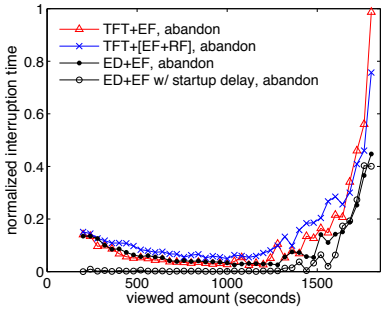


Figure 15: Average NITs of each peer group divided based on viewed length. The synthetic trace used.

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 measure the startup delay of Joint-Family, but there is no significant difference between different approaches whether there is abandonment or not; this is not surprising as they all use EF at startup.

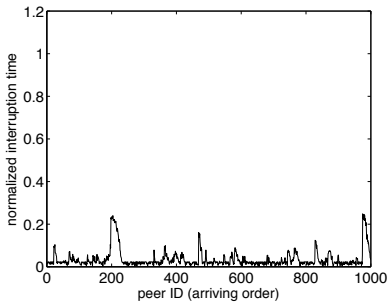
In Fig. 11, we use one of the longer videos (105 minutes), group peers based on how much they watched, and plot the average NIT for each group. Specifically, we divide the video into 100 second bins, and group the viewers into these bins based on how much they watch (i.e., the first group includes peers who watched 0–100 seconds of the video, the next group watched 101–200 seconds of the video, etc.). The server (initial seed) upload capacity in Joint-Family 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 a result, TFT+[EF+RF] results in larger overall NITs than TFT+EF in the presence of abandonment, just as we saw in Fig. 10. As before, the use of ED+EF on Joint-Family 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 Fig. 12 between the case when abandonment exists and when abandonment does not exist. The server capacity in Joint-Family 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.

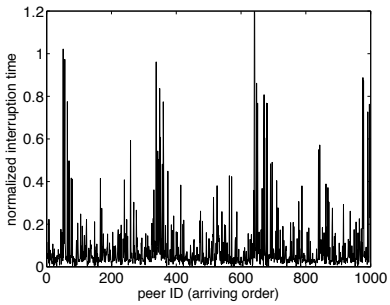
For the synthetic trace experiments, we model the peer arrival and abandonment patterns as random processes. We assume that peer arrival follows a Poisson process with rate $\lambda = 0.05$. We use a 30 minute video, and each arriving peer watches uniformly between 3 and 30 minutes and then abandons. We measure NITs of 2000 consecutive peers who arrive in Joint-Family after Joint-Family 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 2.

First from the synthetic trace results, we compare the load on the Joint-Family server between with and without abandonment in Fig. 13. With abandonment, the load as a byte fraction of requests served by the server 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 server. 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 Fig. 14. When the peer joins the swarm in Joint-Family, it initially has many providers, 22–23 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 server 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 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



(a) NITs of ED+EF with no abandonment



(b) NITs of ED+EF with abandonment

Figure 16: NITs of 1000 peers in Joint-Family sorted in their arriving order

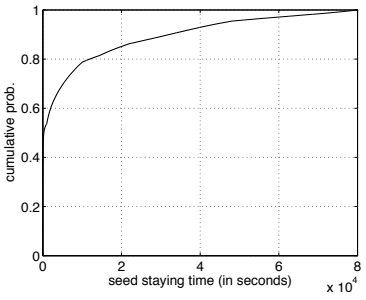


Figure 17: Cumulative distribution of seed staying time from real trace

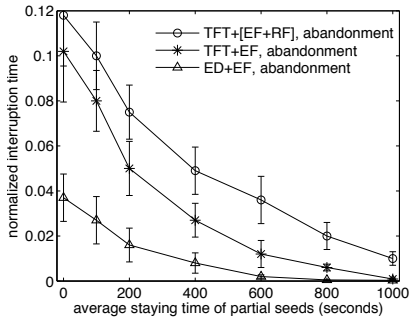


Figure 18: NITs of three different schemes in presence of abandonment as a function of partial seed staying time.

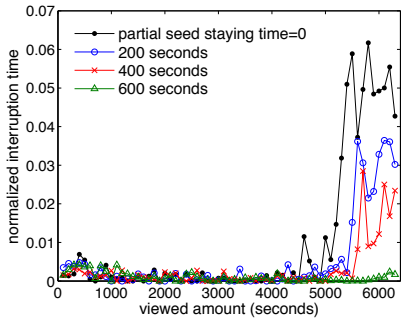


Figure 19: Average NITs of each peer group divided based on viewed length. ED+EF used. Real trace used.

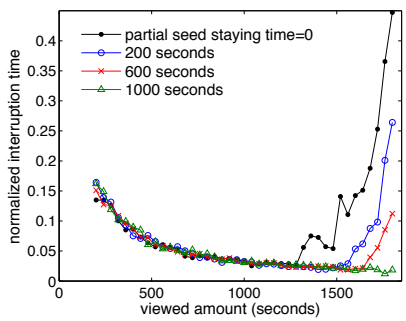


Figure 20: Average NITs of each peer group divided based on viewed length. ED+EF used. Synthetic trace used.

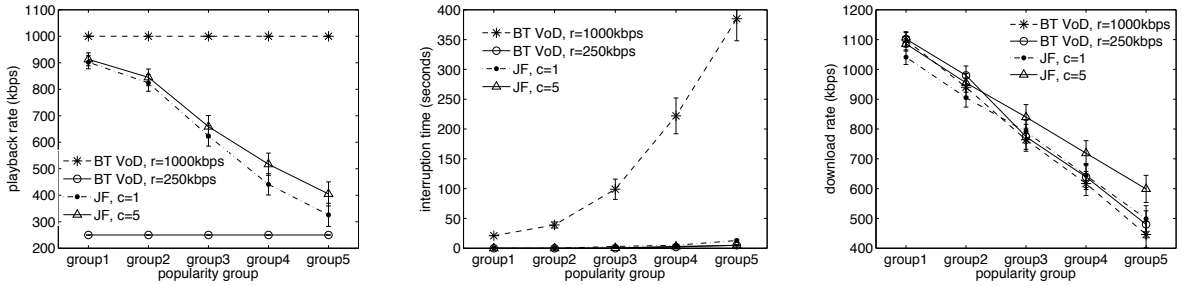
all left at about the 1100 second mark, and thus this peer loses all its possible uploaders other than the server. We note that this trend is strongly related to Joint-Family's EF chunk selection policy since younger peers cannot help older peers with EF. However, we will show that although using EF+RF may alleviate this issue, EF+RF results in more peers having interruptions than EF only, with abandonment.

More importantly, losing older peers seen in Fig. 14 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 Fig. 16(a) and 16(b) we show NITs of each peer in an arriving order, with and without abandonment, respectively. While a similar set of peers experience larger NITs in both cases, the magnitude is much larger in the case of abandonment.

In Fig. 15, we divide peers in Joint-Family into different groups based on their viewed length, and plot the average NITs of each group for the synthetic trace, similarly to Fig. 11 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 periods, 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 Fig. 15 and Fig. 11, although NITs in the real world do not consistently and 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.

4.4. Utilizing Partial Seeds

We investigate the effect of utilizing partial seeds in Joint-Family. To understand the potential of seeds staying on in practice, we present our real traces for all videos collected at the set-top boxes of viewers. Specifically, for each user, 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 viewer's set-top box. Fig. 17 shows 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.



(a) Playback Rate (b) Interruption time (c) Download rate

Figure 21: Comparison between Joint-Family and the state-of-the-art P2P system in presence of abandonment (group 1: the set of most popular videos).

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

Table 3: Playback rates and interruption time with server-based ABR scheme and Joint-Family in presence of abandonment.

Fig. 18 plots NITs of different chunk- and peer- selection strategies as a function of the average staying time of partial seeds when the server capacity in Joint-Family is 3Mbps with a maximum of 15 concurrent uploads. With abandonment, having partial seeds benefits all of the strategies. Here, partial seeds include normal seeds who have downloaded the entire video. Note that the staying time of partial seeds is an exponential distribution, and after this time, partial seeds permanently leave the system. Not surprisingly, the benefit increases as the staying time of partial seeds increases as shown in Fig. 19. 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 4.3, and NITs of peers with long views in Fig. 20 gradually decrease as the partial seeds stay longer.

4.5. Comparison against Server-based ABR

Now we focus on understanding the benefit of using P2P for ABR video delivery when abandonment exists. Starting from this subsection we experiment Joint-Family's ABR videos with allowing each peer in Joint-Family to join multiple swarms. The same trace with viewer abandonment shown in Fig. 7 is also used. We first 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 $c = 5$ (i.e., each peer can store maximum 5 different videos in its local cache regardless of the length of the video). The same buffer-based rate adaptation is applied in both schemes. We look at the average viewers' playback bitrate and interruption time in Table 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 2 (e.g., 125Mbps server bandwidth allows $30 * 5 = 150$ concurrent uploads). Joint-Family requires only 125Mbps server bandwidth to achieve about the same performance as a server-based ABR with 2Gbps 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 1000Kbps.

4.6. Comparison against State-of-the-art P2P VoD

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 (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 was 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 and all the improvements suggested in this paper. 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. To obtain realistic performance results, every experiment is run in presence of viewer abandonment.

First, Fig. 21(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 2.4). 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 2.3, we

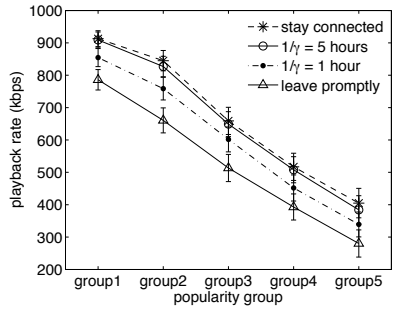


Figure 22: Effect of seed staying time $1/\gamma$ with abandonment (c=5)

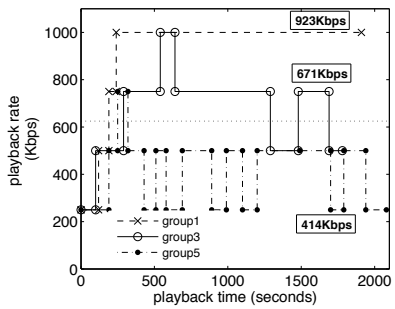


Figure 23: Rate adaptation with ABR in presence of abandonment (c=5)

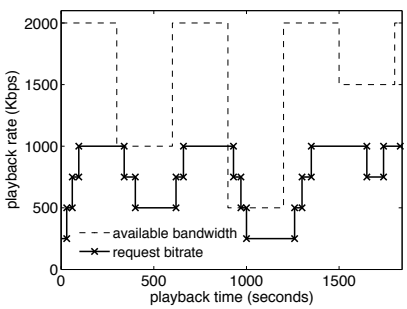


Figure 24: Impact of available bandwidth on ABR

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 Fig. 21(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 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 Fig. 21(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.

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 $c = 5$ when abandonment exists. For users not currently viewing videos, we vary their average staying time $1/\gamma$. In Fig. 22, 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 Fig. 21(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, peer arrivals do not strictly follow a Poisson process but instead show significant diurnal patterns with peak hours (e.g., 8∼12PM in Fig. 25) and off-peak hours. 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.

4.7. 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 realistic experiments we keep taking abandonment into account. For the clarity of presentation, in Fig. 23, we only show 3 groups. For the popular videos (Group 1), the video quickly ramps up to 1000Kbps and stays at the rate to achieve an average playback rate of 923Kbps, which is similar to the total average for Group 1 (as seen in Fig. 21(a)). The Group 3 user also briefly goes up to 1000Kbps before settling back down to 750Kbps for the most part. In the both cases, the average playback is higher than the bitrate of non-ABR case (625Kbps). 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 414Kbps (while the group average is 410Kbps). While this average is lower than 625Kbps, the total interruption time for Group 5 with ABR was only 4.7 secs compared to 20.6 secs for the group without ABR. Fig. 24 plots playback rate transition when a peer's available link bandwidth changes over time. We select a sample user from Group 1 and vary its downlink bandwidth. For most of the time the playback rate tracks the available bandwidth changes.

Thus Joint-Family works harmoniously with ABR, enabling peers to dynamically adapt to the available system capacity among the servers and peers for a particular quality/rate for the video. As seen in our ABR model in Section 2.4, by allowing peers to participate in multiple

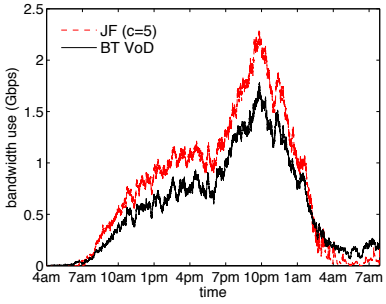


Figure 25: Aggregate upload bandwidth for Joint-Family and BT VoD in presence of abandonment

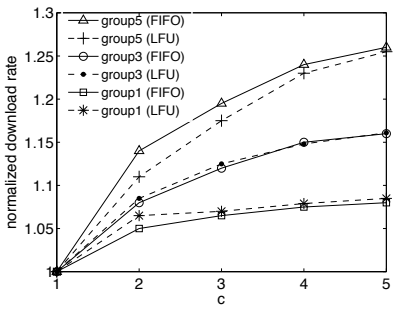


Figure 26: Video popularity and the effect of cache size (c)

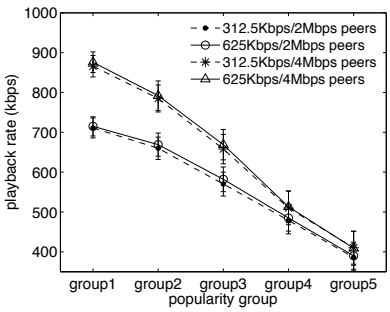


Figure 27: Playback rate of Joint-Family for heterogeneous peers with abandonment

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.

4.8. 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) when abandonment exists, 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. Fig. 25 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 1.8Gbps while 2.3Gbps with Joint-Family (an increase of 27%). By being in multiple swarms, peers in Joint-Family can still upload 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 625Kbps and experiment with both LFU cache replacement (popular in the literature for video caches) and FIFO (used in our analysis). Fig. 26 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 $c = 1$. Similar to our analysis in Section 2.3, we observe that: (a) caching consistently improves the download rate 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 (26% improvement vs. 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).

4.9. 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 uplink/downlink bandwidth of peers. 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. Fig. 27 shows the playback rate for the corresponding peers when abandonment exists. 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.

5. Related Work

We broadly classify related work into the following subsections namely, abandonment, adaptive streaming, multiple swarms, and chunk and peer selection.

Abandonment: Hwang et al.[25], Li et al.[24], and Shafiq et al. [10] present measurement studies on viewer's behavior with abandonment in large scale landline-based or mobile-based IPTV services providers. They demonstrate that users often watch only a small portion of a video. We take into account this effect of abandonment and show that existing schemes for P2P VoD should be reconsidered to cope with more realistic demands. Aalto et al. [31] analyze abandonment in P2P VoD with limited simulation scenarios for model verification. Our work analyzes the contribution of "partial seeds" and performs practical evaluation with real traces to measure impact of abandonment in the real world.

Adaptive Streaming: HTTP adaptive bitrate streaming (ABR) 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 con-

dition. There are several flavors of ABR implementations (e.g., MPEG-DASH [32], YouTube [1], Microsoft Smooth Streaming, Netflix, Adobe Dynamic Streaming [28], Apple Live Streaming). For analysis Yin et al. [33] present a control-theoretic model of ABR with different QoE objectives. While ABR has been used for HTTP server based streaming, the use of P2P systems for ABR is not yet common. Roverso et al. [34, 35] implement ABR in P2P systems for live media only. Lin and Shen [36] study ABR in a P2P-assisted cloud VoD system, while their work is limited to a single swarm and peers in the swarm have to rely on cloud resources.

Scalable video coding (SVC) [7] is yet another approach that enables end-systems to adapt network conditions. [37–39] 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 computational complexity of decoding on end-systems, and the additional bandwidth requirements and overhead compared to ABR. ABR can also easily work with existing network infrastructure such as firewalls and CDNs, and ABR deployment has far outpaced SVC and other alternatives.

Multiple Swarms: 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. [40] and Wang et al. [41] investigate the peer's bandwidth allocation to contribute across multiple swarms in live streaming but not in VoD. Zhou et al. [26] 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. [42] 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.

Chunk and Peer Selection: To adapt BitTorrent for streaming systems (either live or VoD), a combination of rarest first (RF) chunk selection and sequential chunk download (EF) has been exploited [11–15]. Existing schemes vary from a simple probabilistic hybrid model to using sophisticated network coding techniques. Previous work claims that achieving balance between system utilization (by RF) and on-line playback (by EF) can substantially improve playback quality. However, we show that it is a side-effect of using BitTorrent's Tit-for-Tat (TFT) together as peer selection policy and further show that using EF only achieves better playback performance with our Earliest-Deadline (ED) peer selection.

A number of prior works [15, 18, 29, 43–46] show that TFT is not suitable for streaming applications. This is primarily because RF chunk selection is not suitable for streaming, and TFT without RF makes it difficult for new peers to contribute to older peers. Various peer selection approaches have been proposed for streaming. Shah et al. [15] modify TFT's optimistic unchoke policy, D'Acunto et al. [44] make peers act more altruistically, and Wen et al. [43] group peers with similar playback points to help each other. To satisfy a viewer's uninterrupted playback, we replace TFT with ED policy, which ensures that each chunk is delivered to the viewer prior to its deadline.

6. Conclusion

We have presented a holistic redesign of P2P VoD, Joint-Family, which for the first time supports the delivery of adaptive bitrate video in the presence of user abandonment of videos. We show through analysis that, unlike previously known results, it is only with sufficiently long staying times that the available download capacity in P2P VoD depends on the popularity of the content. Our analysis also shows that adaptive bitrate in P2P VoD systems can achieve a higher playback rate for a more popular video. We have demonstrated that user abandonment can impact P2P VoD streaming performance significantly. Abandonment causes larger interruptions (NITs), particularly for peers that watch a video longer, as they are isolated, with no other peers to upload from. We use this analysis and the trace-driven simulation results to guide our design of the Joint-Family protocol.

Joint-Family built upon achieves much better performance than existing strategies as demonstrated by our simulations using traces from a commercial VoD service. By choosing Earliest-Deadline as the peer selection policy and Earliest-First as the chunk selection policy, Joint-Family dramatically improves the viewer's QoE by minimizing interruptions. Joint-Family allows peers to smoothly adapt their quality and achieve a much high playback rate for popular content. Not just that, even for unpopular content, Joint-Family achieves almost 40% higher playback rate than the state-of-the-art P2P VoD approaches which in fact use a fixed bitrate for the video, while reducing the total interruption time by a factor of 4. Joint-Family leverages resources across swarms that are potentially wasted by other schemes and increases system utilization by 30% at peak viewing periods.

Appendix A. Derivation for Equations (2) and (3)

By the balance equations of our 1D M/M/∞ queueing model, we obtain π_i as Equation (1). Using constraint $\sum_{i=0}^{\infty}\pi_i = 1$, we derive π_0 as follows,

$$\pi_0 = \frac{1}{1+\sum_{i=1}^{\infty}\frac{\rho^i}{\prod_{k=1}^{i}(k+\theta)}} \tag{A.1}$$

$$= \frac{1}{1+\sum_{i=1}^{\infty}\frac{\rho^i\theta!}{(k+\theta)!}} \tag{A.2}$$

$$= \frac{1}{1+\frac{\theta!}{\rho^\theta}(e^\rho-\sum_{i=0}^{\theta}\frac{\rho^i}{i!})} \tag{A.3}$$

$$= \frac{1}{\frac{\theta!}{\rho^\theta}(e^\rho-\sum_{i=0}^{\theta-1}\frac{\rho^i}{i!})} \tag{A.4}$$

where $\theta = y - 1$. Note that we assume that θ is integer for Equation (A.2).

Using π_i and π_0, we derive

$$E[X|Y=y] = \sum_{i=1}^{\infty} i\pi_i \tag{A.5}$$

$$= \pi_0 \sum_{i=1}^{\infty} \frac{i\rho^i \theta!}{(i+\theta)!} \tag{A.6}$$

$$= \pi_0 \theta! \left(\sum_{i=1}^{\infty} \frac{\rho^{i+\theta-1}\rho^{1-\theta}}{(i+\theta-1)!} - \theta \sum_{i=1}^{\infty} \frac{\rho^{i+\theta}\rho^{-\theta}}{(i+\theta)!}\right)$$

$$= \pi_0 \theta! \left(\frac{1}{\rho^{\theta-1}}\left(e^{\rho} - \sum_{i=0}^{\theta-1} \frac{\rho^i}{i!}\right) - \frac{\theta}{\rho^{\theta}}\left(e^{\rho} - \sum_{i=0}^{\theta} \frac{\rho^i}{i!}\right)\right)$$

where $\theta = y - 1$. By substituting π_0 (Eq. (A.4)),

$$E[X|Y=y]$$

$$= \rho - \theta \frac{e^{\rho} - \sum_{i=0}^{\theta} \frac{\rho^i}{i!}}{e^{\rho} - \sum_{i=0}^{\theta-1} \frac{\rho^i}{i!}} \tag{A.7}$$

$$= \rho - \theta - \frac{\frac{\rho^{\theta}}{(\theta-1)!}}{\sum_{i=\theta}^{\infty} \frac{\rho^i}{i!}} \tag{A.8}$$

$$= \rho - \theta + \frac{\rho^{\theta}}{e^{\rho}(\Gamma(\theta) - \Gamma(\theta, \rho))} \tag{A.9}$$

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

References

[1] J. Roettg, Don't Touch That Dial: How YouTube is Bringing Adaptive Streaming to Mobile, TVs, http://gigaom.com/2013/03/13/youtube-adaptive-streaming-mobile-tv (2013).

[2] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z.-L. Zhang, Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery, in: IEEE INFOCOM, 2012.

[3] Joost, http://www.joost.com.

[4] UUSee, http://www.uusee.com.

[5] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, B. Li, LiveSky: Enhancing CDN with P2P, ACM Trans. Multimedia Comput. Commun. Appl. 6 (3).

[6] ISO/IEC, Information technology Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats, in: ISO/IEC 23009-1:2012(E), 2012.

[7] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H. 264/AVC standard, IEEE Trans. Circuits and Systems for Video Technology 17 (9).

[8] B. Maggs, A first look at a commercial hybrid content delivery system, in: Keynote presentation at IEEE Global Internet Symposium, 2012.

[9] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, H. Zhang, Understanding the impact of video quality on user engagement, in: ACM SIGCOMM, 2011.

[10] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, J. Wang, Understanding the impact of network dynamics on mobile video user engagement, in: ACM SIGMETRICS '14, 2014.

[11] A. Vlavianos, M. Iliofotou, M. Faloutsos, BiToS: Enhancing BitTorrent for Supporting Streaming Applications, in: 9th IEEE Global Internet Symposium, 2006.

[12] Y. Zhou, D. M. Chiu, J. Lui, A Simple Model for Analyzing P2P Streaming Protocols, in: IEEE ICNP '07, 2007.

[13] B. Fan, D. G. Andersen, M. Kaminsky, K. Papagiannaki, Balancing Throughput, Robustness, and In-Order Delivery in P2P VoD, in: Proc. CoNEXT, 2010.

[14] Y. Borghol, S. Ardon, N. Carlsson, A. Mahanti, Toward Efficient On-Demand Streaming with BitTorrent, in: NETWORKING, 2010.

[15] P. Shah, J. francois Paris, Peer-to-Peer Multimedia Streaming Using BitTorrent, in: IEEE IPCCC 2007, 2007.

[16] D. Qiu, R. Srikant, Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks, in: SIGCOMM, 2004.

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

[18] M. Piatek, A. Krishnamurthy, A. Venkataramani, R. Yang, D. Zhang, A. Jaffe, Contracts: Practical Contribution Incentives for P2P Live Streaming, in: NSDI, 2010.

[19] N. Parvez, C. Williamson, A. Mahanti, N. Carlsson, Analysis of Bittorrent-like Protocols for On-Demand Stored Media Streaming, in: SIGMETRICS, 2008.

[20] Global Broadband Statistics, 1Q 2013, Point Topic., http://point-topic.com/free-analysis-type/subscriber-numbers/ (June 2013).

[21] F. Lehrieder, G. Dán, T. Hossfeld, S. Oechsner, V. Singeorzan, Caching for BitTorrent-like P2P Systems: a Simple Fluid Model and its Implications, IEEE/ACM Trans. Netw. 20 (4).

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

[23] G. De Veciana, X. Yang, Fairness, Incentives and Performance in Peer-to-Peer Networks, in: Seeds, 2003.

[24] Y. Li, Y. Zhang, R. Yuan, Measurement and analysis of a large scale commercial mobile internet TV system, in: ACM IMC, 2011.

[25] K.-W. Hwang, D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, V. Misra, K. K. Ramakrishnan, D. Swayne, Leveraging Video Viewing Patterns for Optimal Content Placement, in: Networking, 2012.

[26] X. Zhou, S. Ioannidis, L. Masosulie, On the Stability and Optimality of Universal Swarms, in: SIGMETRICS, 2011.

[27] G. Tian, Y. Liu, Towards Agile Smooth Video Adaptation in Dynamic HTTP Streaming, in: ACM CONEXT, 2012.

[28] S. Akhshabi, A. C. Begen, C. Dovrolis, An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP, in: ACM MMSys, 2011.

[29] K. Huguenin, A.-M. Kermarrec, V. Rai, M. Van Steen, Designing a Tit-for-Tat Based Peer-to-Peer Video-on-Demand System, in: NOSSDAV, 2010.

[30] A. R. Bharambe, C. Herley, V. N. Padmanabhan, Analyzing and Improving a BitTorrent Networks Performance Mechanisms, in: INFOCOM, 2006.

[31] S. Aalto, P. Lassila, P. Savolainen, S. Tarkoma, How Impatience Affects the Performance and Scalability of P2P Video-on-Demand Systems, in: SIGMETRICS MAMA, 2011.

[32] T. Stockhammer, Dynamic Adaptive Streaming over HTTP: Standards and Design Principles, in: ACM MMSys, 2011.

[33] X. Yin, A. Jindal, V. Sekar, B. Sinopoli, A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP, SIGCOMM '15, ACM, 2015.

[34] R. Roverso, S. El-Ansary, S. Haridi, SmoothCache: HTTP-Live Streaming Goes Peer-to-Peer, in: IFIP Networking, 2012.

[35] R. Roverso, R. Reale, S. El-Ansary, S. Haridi, SmoothCache 2.0: CDN-quality Adaptive HTTP Live Streaming on Peer-to-peer Overlays, in: ACM MMSys, 2015.

[36] Y. Lin, H. Shen, AutoTune: Game-based Adaptive Bitrate Streaming in P2P-Assisted Cloud-Based VoD Systems, IEEE P2P, 2015.

[37] J. Rückert, O. Abboud, T. Zinner, R. Steinmetz, D. Hausheer, Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics, in: IFIP Networking, 2012.

[38] M. Eberhard, A. Palo, A. Kumar, R. Petrocco, L. Mapelli, M. Uitto, NextSharePC: An Open-source BitTorrent-based P2P Client Supporting SVC, in: ACM MMSys '12, 2012.

[39] S. Medjiah, T. Ahmed, R. Boutaba, Avoiding Quality Bottlenecks in P2P Adaptive Streaming, Vol. 32 of IEEE Journal on Selected Areas in Communications, 2014.
[40] D. Wu, Y. Liu, K. Ross, Queuing Network Models for Multi-Channel P2P Live Streaming Systems, in: INFOCOM, 2009.
[41] M. Wang, L. Xu, B. Ramamurthy, Linear Programming Models for Multi-Channel P2P Streaming Systems, INFOCOM'10, 2010.
[42] Z. Wang, C. Wu, L. Sun, S. Yang, Strategies of Collaboration in Multi-Channel P2P VoD Streaming, in: GLOBECOM, 2010.
[43] Z. Wen, N. Liu, K. L. Yeung, Z. Lei, Closest Playback-Point First: A New Peer Selection Algorithm for P2P VoD Systems, in: GLOBECOM, IEEE, 2011.
[44] L. D'Acunto, N. Andrade, J. Pouwelse, H. Sips, Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-Like VoD Systems, ISM '10, 2010, pp. 89–96.
[45] H. Zhang, S. Vasudevan, R. Li, D. Towsley, A Case for Coalitions in Data Swarming Systems, in: ICNP, 2011.
[46] Y. Yang, A. L. H.Chow, L. Golubchik, D. Bragg, Improving QoS in BitTorrent-like VoD systems, in: IEEE INFOCOM, 2010.

Kyung-Wook Hwang received his B.S. degree in Electronics Engineering from Sogang University, South Korea. He received his M.S. and Ph.D. degrees in Electrical Engineering from Columbia University, USA, focusing on multimedia networks and P2P networks with emphasis on modeling and performance analysis. He is currently a senior member of technical staff at AT&T Research. His research interests are in the areas of network management and traffic analysis on mobile networks. He is a member of the IEEE.

Vijay Gopalakrishnan received the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, MD, USA, in 2003 and 2006, respectively. He has been with AT&T since 2006 and has worked on innovative solutions in the space of network management, content delivery, and the mobile Web. He is currently a Director with the Network and Service Quality Management Center, AT&T Labs-Research, Bedminster, NJ, USA, leading a team of researchers focused on systems challenges in the architecture, protocols and management of networks.

Rittwik Jana is a Lead Inventive Scientist at AT&T Labs Research. His research interests span Internet technologies, networked video streaming, cellular networks and systems, intelligent service composition of VNFs. Rittwik holds a Bachelor of Engineering in Electrical and Electronics from the University of Adelaide, Australia, and a Ph.D. in Telecommunications Engineering from the Australian National University, Canberra, Australia.

Seungjoon Lee is currently with Two Sigma Investments, LLC. Before joining Two Sigma in 2014, he was with AT&T Research, NJ for more than 8 years. He received his Ph.D in Computer Science from University of Maryland, College Park in 2006. He also received Bachelor's and Master's degrees in Computer Science from Seoul National University, Seoul, Korea, in 1996 and 2000. His research interests include large scale systems, network management, content distribution, cloud computing, and mobile computing.

Vishal Misra (IEEE Fellow, 2016) is Professor in the Computer Science Department, with a joint appointment in the Electrical Engineering Department of Columbia University. His research emphasis is on mathematical modeling of networking systems, bridging the gap between practice and analysis. He served as the Vice-Chair of the Computer Science Department at Columbia University from 2009 to 2011, and in 2011 he spun out Infinio, a company in the area of datacenter storage. In the past he also cofounded CricInfo, which was subsequently acquired by ESPN. He received his undergraduate degree from IIT Bombay and MS and PhD degrees from the school of engineering at University of Massachusetts at Amherst, which gave him a distinguished alumnus award in 2014.He is also the recipient of the NSF Career, DoE Career, IBM and Google Faculty awards.

K. K. Ramakrishnan (IEEE Fellow, 2005) is a Professor in the Computer Science and Engineering Department of the University of California, Riverside. From 1994 until 2013, he was with AT&T, most recently a Distinguished Member of Technical Staff at AT&T Labs-Research, Florham Park, NJ. Prior to 1994, he was a Technical Director and Consulting Engineer in Networking at

Digital Equipment Corporation. Between 2000 and 2002, he was at TeraOptic Networks, Inc., as Founder and Vice President. Dr. Ramakrishnan is also an AT&T Fellow, recognized in 2006 for his work on congestion control, traffic management and VPN services, and for "fundamental contributions on communication networks with a lasting impact on AT&T and the industry". He received an AT&T Technology Medal in 2013 for his work on Mobile Video Delivery Strategy and Optimization. His work on the "DECbit" congestion avoidance protocol received the ACM Sigcomm Test of Time Paper Award in 2006. He has published more than 200 papers and has 151 patents issued in his name. He has been on the editorial board of several journals and has served as the TPC Chair and General Chair for several networking conferences. He received his MS from the Indian Institute of Science (1978), MS (1981) and Ph.D. (1983) in Computer Science from the University of Maryland, College Park, USA.

Dan Rubenstein is an Associate Professor in the Department of Computer Science at Columbia University. He received a B.S. degree in mathematics from M.I.T., an M.A. in math from UCLA, and a PhD in computer science from University of Massachusetts, Amherst. His research interests are in network technologies, applications, and performance analysis. He is an editor for IEEE/ACM Transactions on Networking, was program chair of IFIP Networking 2010 and ACM Sigmetrics 2011, and has received an NSF CAREER Award, IBM Faculty Award, the Best Student Paper award from the ACM SIGMETRICS 2000 conference, and Paper awards from the IEEE ICNP 2003 Conference, ACM CoNext 2008 Conference, and IEEE Communications 2011. He spent 2011 at Google.