Joint-Family: Enabling Adaptive Bitrate Streaming in Peer-to-Peer Video-on-Demand

Kyung-Wook Hwang*, Vijay Gopalakrishnan†, Rittwik Jana†,
Seungjoon Lee†, Vishal Misra*, K.K. Ramakrishnan†, Dan Rubenstein*
*Columbia University, kwhwang@ee.columbia.edu, {misra, danr}@cs.columbia.edu
†AT&T Labs – Research, {gvijay, rjana, slee, kkrama}@research.att.com

***Abstract*—We propose Joint-Family, a protocol that combines peer-to-peer (P2P) and adaptive bitrate (ABR) streaming for video-on-demand (VoD). While P2P for VoD and ABR have been proposed previously, they have not been studied together because they attempt to tackle problems with seemingly orthogonal goals. We motivate our approach through analysis that overcomes a misconception resulting from prior analytical work, and show that the popularity of a P2P swarm and seed staying time has a significant bearing on the achievable per-receiver download rate. Specifically, our analysis shows that popularity affects swarm efficiency when seeds stay "long enough". We also show that ABR in a P2P setting helps viewers achieve higher playback rates and/or fewer interruptions. We develop the Joint-Family protocol based on the observations from our analysis. Peers in Joint-Family simultaneously participate in multiple swarms to exchange chunks of different bitrates. We adopt chunk, bitrate, and peer selection policies that minimize occurrence of interruptions while delivering high quality video and improving the efficiency of the system. Using traces from a large-scale commercial VoD service, we compare Joint-Family with existing approaches for P2P VoD and show that viewers in Joint-Family enjoy higher playback rates with minimal interruption, irrespective of video popularity.**

I. Introduction

The ever-increasing demand placed by streamed video traffic across both wired and wireless networks has been managed by two seemingly complementary approaches: adaptive bitrate (ABR) [1, 2], and peer-to-peer (P2P) delivery [3, 4]. ABR encodes a video at multiple bitrates, and maximizes the video bitrate within the available bandwidth, giving a higher fidelity video when possible, and dropping to lower quality rather than forcing an interruption of playback. P2P-based systems are a popular alternative to deliver on-demand video, improving the viewing experience by utilizing the upload capacity of the downloading nodes, thereby increasing overall upload capacity. Even traditional Content Distribution Network (CDN) providers such as Akamai [5] are choosing to experiment with and deploy P2P-based delivery of video content.

Intuitively, P2P and ABR do not seem well-suited to work together, because viewers watching the video at differing rates are presumably unable to exchange video parts with one another. Hence it appears that enabling ABR interferes with the ability for peers to share video parts with one another. We show in this paper that, contrary to current intuition, ABR and P2P can be effectively combined in a way that leverages their strengths: P2P techniques improve upload capacity, and ABR enables the highest quality viewing at that capacity while minimizing interruptions.

In this paper, we present a novel system called **Joint-Family** that combines P2P and ABR to provide high-quality streaming[1] Video-on-Demand (VoD). The basis for Joint-Family comes from our analysis based on a Markov model, where we identify the relationship between video popularity, seed staying time and downloading rate. We show that when seeds stay "sufficiently long", *content popularity affects swarm efficiency* (Section II-B). 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 [6] which claim independence between popularity and download capacity. Hence, we identify the conditions under which existing results are contradicted. We then analyze the effectiveness of caching previously watched videos and sharing them as a mechanism to extend seed staying time (Section II-C). With caching we can also transfer underutilized capacity from one swarm to another and thereby improve 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 II-D). We use our Markov model to show that ABR allows P2P swarms to migrate to the highest sustainable rate for that swarm: highly popular content will induce large swarms and have a high sustainable download capacity, whereas less popular content will have smaller swarms and a lower sustainable download capacity.

Based on our analytical observations, we design a novel protocol called Joint-Family to deliver high-quality videos with minimal playback interruptions in a P2P system, using multi-swarm participation and ABR (in Section III). A peer in Joint-Family caches and shares multiple ABR videos using storage space at the end system, and increases capacity of swarms (especially for unpopular videos) by supplementing it with (unused) peer capacity. Hence, the peer participates in multiple swarms concurrently, and shares different parts of the ABR video at multiple bitrates. To support this, we identify the right combination of chunk selection, peer selection, and bitrate adaptation policies that minimize interruptions. Our design makes Joint-Family immediately suitable for existing VoD infrastructures in which the provider owns the distribution infrastructure (e.g., CDNs [5], IPTV [7]). We also describe how the protocol can be applied in a decentralized setting by utilizing mechanisms that encourage sharing of content [8]. We conduct extensive performance evaluations of Joint-Family using traces from a nationally deployed VoD service (in Section IV), and show that ABR with P2P is indeed feasible. Compared to a generalized implementation of the state-of-the-art in P2P VoD, our instantiation of Joint-Family delivers high quality VoD streaming, even for unpopular videos, with minimal interruptions.

II. 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 ABR can significantly improve the playback experience even for unpopular content.

[1]As opposed to download-and-play

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

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

TABLE I. PARAMETERS AND DEFINITION

A. Assumptions

The notations used in our model are summarized in Table I. We use the leecher arrival rate λ for a video as its popularity (i.e., if arrival rate of video i is larger than that of video j, then i is more popular than j). A leecher's download (streaming) rate can be faster than the video playback rate for potentially fewer playback interruptions. We assume that each leecher watches a video till the end, and thus seeds have the entire video. However, all our experiments in Section IV also account for viewers' premature abandonment based on real traces. Similar to other P2P studies [9, 10] and based on the wireline subscriber statistics [11], 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 IV-G.

B. Popularity, Download Rate, and Seed Staying Time

The fluid model based analysis by Qui and Srikant [6] suggests that the download performance of files is relatively independent of their popularity. They explain that the supply and the demand placed by leechers are always offset regardless of video popularity. There has been subsequent work [9, 12] 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 [13] can explain what the fluid model misses. Intuitively, if we plot the intervals between request arrivals and observe the download rate at any random instant, our observation is likely to fall into a "larger" interval. In these large intervals, the download rates of leechers monotonically increases, since we assume the seeds stay for a sufficiently long time and many active leechers transition to being seeds. This simultaneously increases the supply as well as reduces the demand for download capacity. Feller's paradox explains why these longer intervals have a greater effect on the time averaged download times, and in our case the effect is beneficial.

Our analysis uses a continuous time Markov chain, where we define $x, y \geq 0$ to be the respective numbers of leechers and seeds in a swarm. Our model is motivated by a two-dimensional (2D) model by Veciana and Yang [14] (which only presents recursive relationship). In our analysis, we first fix y and derive a conditional expectation using a variant of M/M/∞ queue. We then derive simple formulas for the expected number of leechers and download time. While our analysis could be equally applicable for file sharing scenarios, we focus on video streaming only.

Given y seeds, consider a Markov chain, where each state corresponds to the number of leechers (x). Then, the transition rate from state i to $i+1$ is: $q_{i,i+1} = \lambda$ for $i \geq 0$, where λ is the request arrival rate. For the transition down from i to $i-1$, we assume a "perfect cascade" as used in Fan et al. [10], where all leechers except the latest arrival can always upload to other leechers. Then, $q_{i,i-1} = (\eta(i-1)+y)u/B$ for $i \geq 1$, where η corresponds to the efficiency parameter for data transfer from leechers [14]. This parameter is experimentally shown to be close to 1 for most practical cases [6, 9], and we also 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:

$$E[X|Y=y] = \sum_{i=0}^{\infty} i\pi_i = \rho - y + 1 + \frac{e^{-\rho}\rho^{(y-1)}}{\Gamma(y-1) - \Gamma(y-1,\rho)} \tag{3}$$

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

Now, let us consider the distribution for the number of seeds (y). A seed arrival is equivalent to a leecher completing download of the video. As a result, at steady-state, the leecher arrival rate λ is the same as the seed arrival rate. On the other hand, a seed leaves the swarm at the rate of γ (i.e., determined by the staying time). This forms the standard M/M/∞ queueing system, where the up-transition rate is λ and the down-transition rate is γy. Thus, $P[Y=y] = e^{-\sigma}\frac{\sigma^y}{y!}$, where $\sigma = \lambda/\gamma$. By combining this with (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 IV-A for detail). In our experiments, we consider a 1800-second video of r=625Kbps, resulting in B=1125Mbits. We use u=312.5Kbps. We also simulate state transitions using the 2D Markov model [14] for comparing results with our analysis. Specifically, we start at state ($x = 0, y = 0$) and simulate transitions according to the transition rates until we reach a steady state (where the change on both x and y becomes very small). After reaching a steady state, we record the time between arrival and conversion to a seed for each of next 3000 leechers and compute the downloading rate. We use a similar

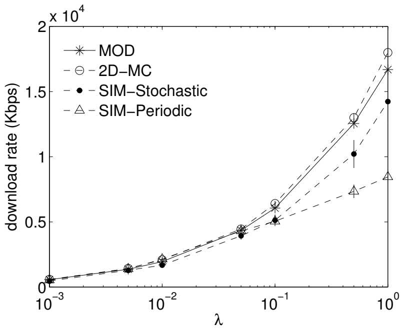


Fig. 1. Download rate as a function of λ with $1/\gamma = 3600$ secs. (X axis in log scale)

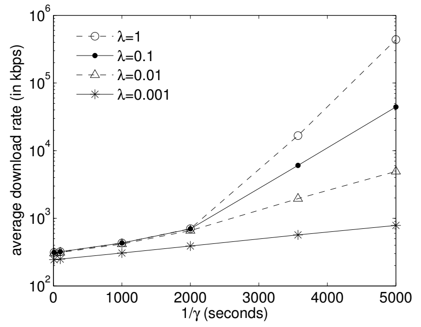


Fig. 2. Download rate as a function of seed staying time $1/\gamma$. (Y axis in log scale)

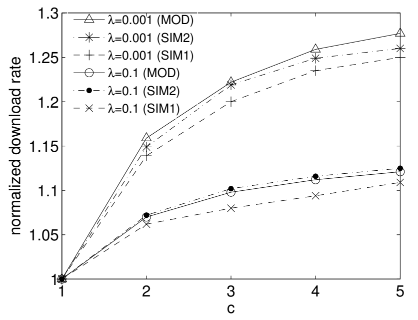


Fig. 3. Caching multiple videos: each download rate is divided by the download rate when $c = 1$. ($1/\gamma = 3600$ secs)

warm-up strategy for our event-driven experiments.

Figure 1 shows the average download rate of leechers for different λ with $\frac{1}{\gamma}$ =1 hour. We compare four cases: simulated transition on the 2D Markov model [14] (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 [6, 9]. 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 leecher arrival pattern also plays a critical role in the download performance, and the assumption of periodic arrivals in the fluid models [6, 9] can lead to incorrect conclusions in practical scenarios.

We next investigate the effect of seed staying time on download performance. In Figure 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 [6, 9]. 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).

C. Caching to Increase Staying Time and Download Rate

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

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

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

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:

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

where $\sigma_c = c\lambda/\gamma$. From (5) and (6), we can obtain the average download time T by following similar derivation as in Section II-B. 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 II-B with a synthetic trace. Figure 3 plots the normalized average download rate from our analysis and from the simulation for different cache size c. For SIM1, we simulate exactly the policy described for deriving Equation (5) for validation. Also, SIM2 shows the results without our assumption so that a leecher actively watching a video also uploads all other videos in its cache. We first observe that the analysis (MOD) and simulation results match well. Also, caching is more beneficial with small λ (i.e., less popular videos). We then see the diminishing returns as c grows since our small u which is the bottleneck quickly becomes more

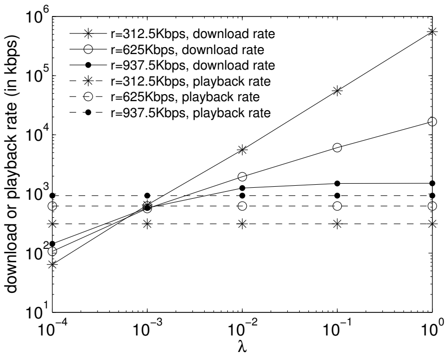


Fig. 4. Download/playback rate vs. arrival rate with different chunk bitrates

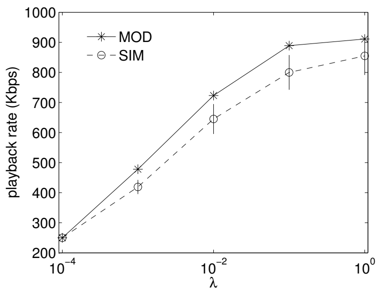


Fig. 5. Validation of ABR analysis using simulation

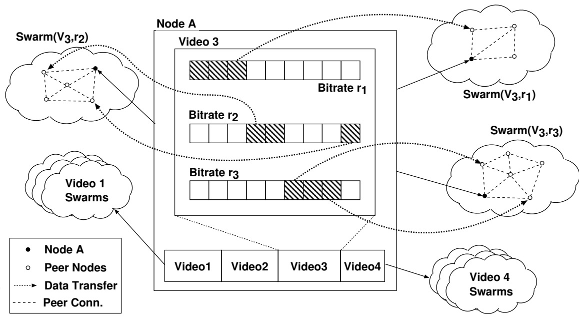


Fig. 6. A peer in Joint-Family participates in multiple swarms.

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 IV-E we explore different cache replacement schemes such as LFU using real-world traces.

D. Adaptive Bitrate Analysis

We showed in Section II-B that more popular videos result in higher download rates only with seeds staying long enough. When the download rate is (unnecessarily) much higher than the video playback rate, we now leverage the abundant capacity to improve the video quality through ABR. Using the example of a single video at different bitrates via our model, we show in Figure 4 that as the video popularity varies, the achievable average download rate varies quite significantly. We choose 3 different bitrates (312.5 – 937.5 Kbps) with the corresponding horizontal lines. When the video is unpopular, the peer download rate can be smaller than the playback rate, especially for the higher bitrates, likely resulting in playback interruptions. When the video is more popular ($\lambda = 0.01$ or higher), the download rate is higher than the playback rate, especially for the lower bitrates (e.g., 312.5Kbps).

We make the following observations: First, using a single bitrate for all videos is suboptimal. If the bitrate is set too high, streaming an unpopular video would result in significant amount of playback interruption. If the bitrate is too low (with the goal of minimizing interruptions), viewers of popular videos would be unnecessarily restricted to low bitrates – i.e., poor streaming quality. To overcome this, one might consider predicting the video popularity and using the highest bitrate sustainable for that popularity. 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 leecher only increases the video bitrate to reach the highest bitrate it can sustain. Specifically, each leecher starts with r_1 and increases the bitrate from r_i to r_{i+1} if it has at least s_u seconds of video chunks at rate r_i buffered ahead of its playback point (also explained in Section III-C). If a leecher is not able to go to a higher bitrate, then it stays at the current bitrate until the streaming finishes. Also, we assume that all leechers for a given video go through the same set of "transition points" in a steady state. In other words, all leechers download B_1 bytes at r_1 before switching up to r_2 and receive B_2 bytes at r_2 before transitioning to r_3, and so on.

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

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

where $f_k = \frac{\tilde{B}_k}{\sum_j \tilde{B}_j}$. Then, following the analysis for each Markov chain in Section II-B (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 a 1800 second video with 4 bitrates {250, 500, 750, 1000} Kbps, and set $s_u = 50$. In the simulation, peers have to switch down to lower bitrates if the size of buffered chunks becomes smaller than s_d, and we use $s_d = 10$ (see Section III-C for detail). Figure 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.

III. Joint-Family Design

We take the learnings from our analysis in Section II to design a P2P protocol that supports the delivery of high quality video using ABR. To the best of our knowledge, Joint-Family is the first practical P2P VoD system that incorporates ABR.

A. 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 II-C and II-D has a beneficial effect on the overall download performance and playback rate for ABR videos. Figure 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.

B. Protocol Mechanisms for Multi-Swarm P2P

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

Connection management: We term all connections that node A has to peers in swarms of the video it is currently watching as *selfish*. Connections to swarms of cached videos are termed *altruistic*. The peer on the other end of a selfish connection, B, can be either a leecher or a seed. In the latter case, 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.

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

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

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

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

C. Chunk Selection and Rate Adaptation

Chunk selection: The chunk selection policy determines the order in which a peer watching 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 [10]. ABR further complicates this, as the rarest chunk at the time of download may not match the right bitrate at the time of playback.

In Joint-Family, we use Earliest-First (EF) chunk selection. EF allows for a fast startup, potentially fewer and shorter interruptions, and smaller wastage of downloaded chunks when users abandon viewing a video. Also, with buffer-based 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 [10]. 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 swarm, and our caching mechanism helps avoid the "missing piece syndrome" [15].

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 [1, 16], 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 leecher 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.

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

D. 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 inter-dependency 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 [8, 17] 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 leecher) 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 IV-F 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 [8, 17] in deployments that do not worry about free riding (e.g., managed content delivery [5]). In scenarios where eliminating free-riding is of concern, we can use the mechanisms proposed in Contracts [8], 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.

IV. 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.

A. Experiment Setup

To evaluate Joint-Family, the BitTorrent simulator [18] is used with the following major modifications: (1) video streaming support (e.g., playback buffer), (2) bitrate adaption (Section III-C), (3) multi-swarm participation (Sec. III-B), (4) different chunk (Sec. III-C) and peer selection policies (Sec.

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

TABLE II. SIMULATION PARAMETERS

III-D). Note that for the hybrid chunk selection (EF+RF), a peer initially uses EF, but switches to EF+RF once enough chunks are in its playback buffer. This helps achieve lower startup delay and playback continuity by providing the slack needed to deal with possible future reductions in the download rate. In our experiments, a peer uses EF with probability 0.7 and RF with 0.3, once there are 5 or more chunks in its buffer.

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

We summarize the different parameters used in the simulations in Table II. We use 5 servers, each with 25Mbps uplink capacity to host all the videos and behave like seeds. We assume continuous network connectivity of each joining peer until the end of an experiment so that the peer helps other peers as a seed for previously viewed videos. While the playback rate for non-ABR videos is set to 625Kbps, we use 4 quality levels for ABR videos: 250, 500, 750 and 1000Kbps (the average being 625Kbps). Like most P2P systems, each video is broken into chunks of 10 seconds of playback, and a peer can play back a chunk while it is being downloaded (subject to the startup delay and the appropriate portion being available). For ABR, hold-down time for bitrate switch-up (h_u) and switch-down (h_d) are set to 30 and 10 seconds, respectively. Also, for the buffer size parameters of switch-up and switch-down, we use $s_u = 50$ and $s_d = 20$ 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 to evaluate video playback performance. While the former gives information about the quality of video viewed by the user, the latter captures the aggregate disruptions experienced by the viewer.

B. *Joint-Family vs. Server-based ABR*

To first understand the benefit of using P2P for ABR video delivery, we compare Joint-Family with the traditional server-based ABR scheme. Viewers in the server-based ABR do not share their downloaded content. Joint-Family uses a cache size of $c = 5$. The same buffer-based rate adaptation is applied in both schemes. We look at the average viewers' playback bitrate and interruption time in Table III 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 II (e.g., 125Mbps server bandwidth allows $30*5 = 150$ concurrent uploads). Joint-Family requires only 125Mbps server bandwidth

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

TABLE III. PLAYBACK RATES AND INTERRUPTION TIME WITH SERVER-BASED ABR SCHEME AND JOINT-FAMILY

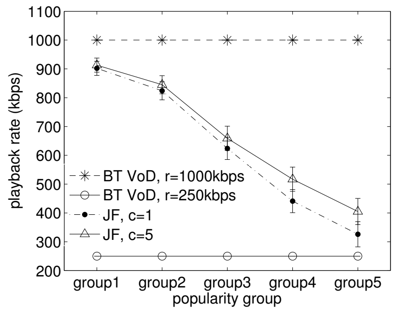
to achieve about the same performance as a server-based ABR with 2Gbps server bandwidth. Note that the improvement in the playback rate of Joint-Family with larger server bandwidths reaches a point of diminishing returns because the highest ABR video bitrate is limited to 1000Kbps.

C. *Joint-Family vs. State-of-the-art P2P*

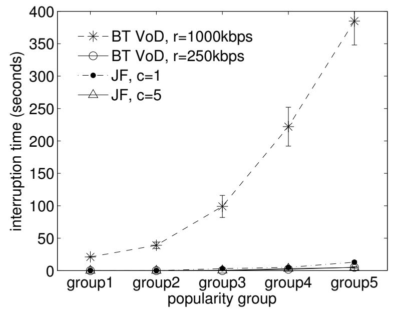
For the performance comparison of Joint-Family, instead of comparing with specific existing implementations, we use a generalized implementation that incorporates the state-of-the-art in P2P VoD. The generalized implementation (henceforth BT VoD) uses the hybrid policy (EF+RF) for chunk selection and TFT for peer selection. Since existing P2P systems only support a single rate, we experiment with two fixed rates: 1000Kbps and 250Kbps to represent the two extremes (high quality and no interruption). Note that 250Kbps is the maximum bitrate for BT VoD that achieved no interruption for all viewers. Further, these systems only allow participation in one swarm (equivalent to $c = 1$ in Joint-Family). We use two scenarios for Joint-Family: $c = 1$ and $c = 5$. Joint-Family uses ABR 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.

First, Figure 7(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 II-D). Interestingly, the average playback rate of the least popular videos is also much higher with Joint-Family (by $\sim$100Kbps) than the lowest possible rate. Similar to our analysis in Section II-C, we also consistently see the benefit of caching and participating in multiple video swarms (e.g., $c = 5$ vs. $c = 1$). To understand whether the playback rate is sustained with minimal interruptions, we plot the average interruption time in Figure 7(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 Figure 7(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.

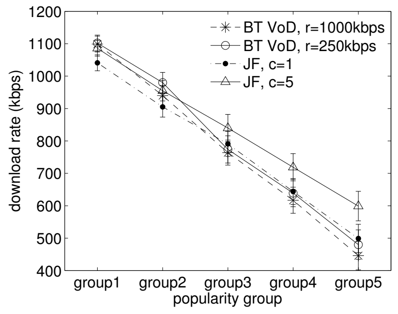
The combination of these results illustrates why it is important to adapt: If we pick too high a quality (e.g.,



(a) Playback Rate



(b) Interruption time



(c) Download rate

Fig. 7. Comparison between Joint-Family and the state-of-the-art P2P system (group 1: the set of most popular videos).

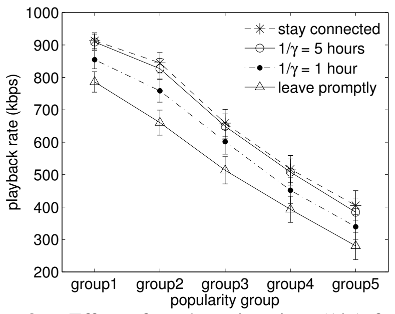


Fig. 8. Effect of seed staying time ($1/\gamma$) for JF, c=5

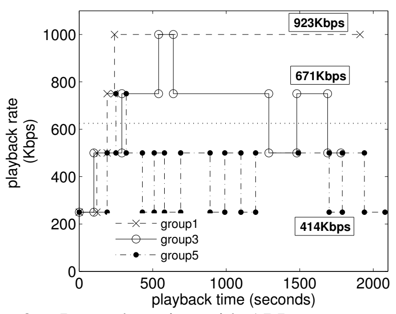


Fig. 9. Rate adaptation with ABR

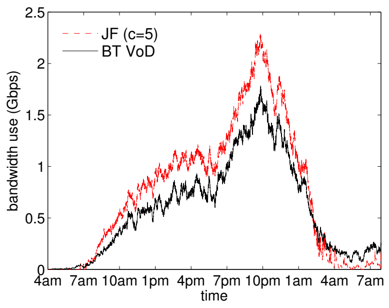


Fig. 10. Aggregate upload bandwidth for Joint-Family and BT VoD

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

We now study the effect of seed staying time in Joint-Family with $c = 5$. For viewers who are not currently watching any video, we vary their average staying time $1/\gamma$. In Figure 8, the 'leave promptly' curve indicates that all viewers leave the VoD network right after they finish watching, while the 'stay connected' curve (identical to 'JF, c=5' in Figure 7(a)) indicates that they stay connected till the end of each simulation. We first see that the playback results have a similar trend in that more popular swarms still achieve higher bitrates. Secondly, the improvement in playback rates with longer staying times reduces (e.g., '$1/\gamma = 5$ hours' and 'stay connected' are almost identical). This is because, unlike our analysis, viewers' arrivals do not strictly follow a Poisson process but instead show significant diurnal patterns with peak hours (e.g., 8$\sim$12PM in Figure 10) 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.

D. Performance Improvement with ABR

We take a closer look at how a peer's playback experience evolves over a video streaming session. As Joint-Family adapts using ABR according to the available capacity for that video based on its popularity, we select a sample user from each popularity group and plot the playback rate over time as well as its overall average when $c = 5$. For the clarity of presentation, in Figure 9, we only show 3 groups. For the popular videos (Group 1), the video quickly ramps up to 1000 Kbps and stays at the rate to achieve an average playback rate of 923 Kbps, which is similar to the total average for Group 1 (as seen in Figure 7(a)). The Group 3 user also briefly goes up to 1000 Kbps before settling back down to 750 Kbps for the most part. In both these cases, the average playback is higher than the bitrate of non-ABR case (625 Kbps). Finally, since there is no sufficient capacity for the unpopular videos to support a high rate, the Group 5 user oscillates between 500 and 250 Kbps to achieve an average of 414 Kbps (while the group average is 410 Kbps). While this average is lower than 625 Kbps, the total interruption time for Group 5 with ABR was only 4.7 secs compared to 20.6 secs for the group without ABR.

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 II-D, by allowing peers to participate in multiple swarms, peers viewing a popular video are naturally able to take advantage of the higher bitrate chunks that become available because of the increased system capacity for such popular content.

E. Effect of Multiple Swarms

To understand the underlying reason for the improved performance of Joint-Family, we examine the overall system utilization. We periodically sample the upload bandwidth aggregated across all peers (excluding servers) and report the time series for Joint-Family (with $c = 5$) and BT VoD. In this experiment we do not use ABR to remove the performance impact by rate adaption. Figure 10 shows Joint-Family effec-

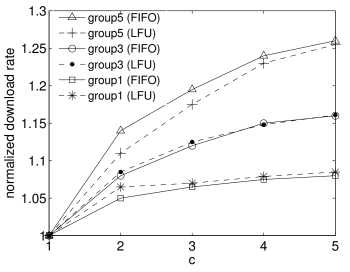


Fig. 11. Video popularity and the effect of cache size (c)

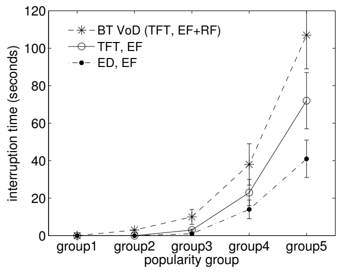


Fig. 12. Interruption time with different chunk- and peer-selection policies

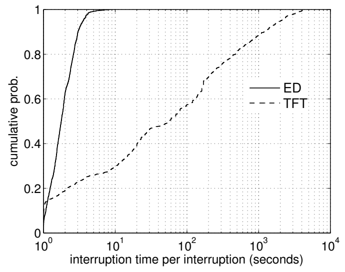


Fig. 13. CDF of interruption time for each interruption

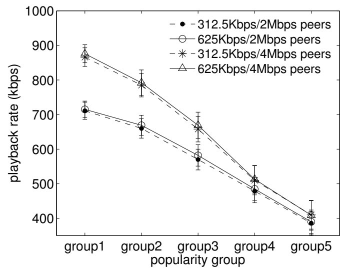


Fig. 14. Playback rate of Joint-Family for heterogeneous peers

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

TABLE IV. Interruptions count with ED and TFT

tively increases the system utilization compared to BT VoD. Specifically, at the peak viewing period, the aggregate upload bandwidth by BT VoD is 1.8 Gbps while 2.3Gbps with Joint-Family (an increase of 27%). By being in multiple swarms, peers in Joint-Family can use their upload capacity as long as they receive a chunk request from *any* of the swarms, thus improving overall upload capacity and playback experience.

Caching and video popularity: We turn our attention to increasing system capacity so that we can increase the video download rate through caching. We run Joint-Family with a constant bitrate of 625Kbps and experiment with both LFU cache replacement (popular in the literature for video caches) and FIFO (used in our analysis). Figure 11 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 II-C, 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 (about 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).

F. Impact of Chunk and Peer Selection Policies

We evaluate the contribution of the chunk selection and peer selection policies in Joint-Family. To perform this experiment, we started with BT VoD and first replaced the hybrid chunk selection policy with EF (TFT+EF, using the terminology of peer selection + chunk selection policies). We then replaced TFT peer selection with ED (ED+EF). A single bitrate, 625Kbps is used. Similar to Figure 7(c), the download rates for different policies are comparable and thus omitted here. Figure 12 shows that BT VoD experiences much longer interruptions compared to TFT+EF. Specifically, for the least popular group, the interruption time reduces by 33% when TFT+EF is used instead of BT VoD. This is because EF prioritizes chunks closest to the current playback point. In contrast, even though the download rate of BT VoD is similar to that of TFT+EF (not shown here), the partial use of RF in BT VoD results in many downloaded chunks that are not immediately useful. Next, when replacing TFT by ED (i.e., ED+EF), we consistently get further reductions in interruption times. In particular, the interruption time of ED+EF goes down by an additional 44% compared to TFT+EF. This can be attributed to the "fairness" aspect of ED, where we prioritize peers that really need the chunk soon, as opposed to TFT where peers unchoke other peers based on their upload rates.

To show this property, in Figure 13, we plot the cumulative distribution of the interruption time for each interruption the user experiences. We see that with ED, more than 98% of interruptions last for less than 3 seconds, while 60% last even shorter (1 sec or less). The maximum interruption time is less than 10 secs. On the other hand, with TFT there are much fewer short interruptions, while the majority of interruptions are long (40% last for $>$ 120 secs). This result demonstrates inherent unsuitability of TFT with streaming video. We also examine the number of interruptions in Table IV. ED experiences more interruptions than TFT. However, since the duration of each interruption is significantly shorter, the overall total interruption time due to ED is very small. Additionally, the fact that 60% of interruptions last for 1 second or less suggests that the deadline we are using is extremely aggressive.

G. 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 and 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. Figure 14 shows the playback rate for the corresponding peers. The benefit is predominantly seen for popular videos. Peers with higher downlink bandwidth see a greater improvement in the playback rate for their popular videos than when their uplink bandwidth changes. The higher downlink bandwidth allows the system (initially by the servers) to populate the environment (peers) with higher quality chunks (even if the uplink bandwidth is halved from 625 to 312.5 Kbps) which is then effectively shared among the peers viewing the popular video over time due to the increased system capacity for the popular video.

V. Related Work

Adaptive Streaming: 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 condition. There are several flavors of ABR implementations (e.g., MPEG-DASH, Adobe Dynamic Streaming, Microsoft Smooth Streaming, Apple HTTP Streaming [1, 2, 19, 20]). While ABR has been used for HTTP server based streaming, the use of P2P systems for

ABR are not yet common. Roverso et al. [21] implement ABR in P2P systems for live media only. To the best of our knowledge, we are the first to investigate ABR for P2P VoD. Scalable video coding (SVC) is yet another approach that enables end-systems to adapt network conditions. [22–24] take a multiple description or layered coding approach which causes interdependency of layers per chunk distribution in P2P and which is not directly applicable to ABR or our P2P work. Also, SVC has not been widely implemented due to the complexity of decoding on the end-systems, and the additional bandwidth requirements compared to ABR. ABR deployment has far outpaced other alternatives.

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. [25] and Wang et al. [26] investigate the peer's bandwidth allocation to contribute across multiple swarms in live streaming but not in VoD. Zhou et al. [15] 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. [27] 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 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 [10, 28, 29]. 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 TFT together as peer selection policy and further show that using EF only achieves better playback performance with our ED peer selection.

Peer selection: BitTorrent's TFT is effective for file sharing with its incentive mechanism to encourage a peer's contribution. However, a number of prior works [8, 17, 29–31] 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. [29] modify TFT's optimistic unchoke policy, D'Acunto et al. [31] make peers act more altruistically, and Wen et al. [30] group peers with similar playback points to help each other. To satisfy a viewer's uninterrupted playback experience, we replace TFT with the Earliest-Deadline (ED) policy, which ensures that each chunk is delivered to the viewer prior to its deadline.

VI. Conclusion

We have presented a holistic redesign of P2P VoD called Joint-Family that for the first time supports the delivery of adaptive bitrate videos. We show through analysis that only with sufficiently long staying times, the available download capacity in P2P VoD depends on the popularity of the content, and use this to guide our protocol design. Joint-Family achieves much better performance than other strategies as demonstrated by our simulations that use traces from a commercial VoD service. By choosing Earliest-Deadline as peer selection and Earliest-First as chunk selection policy, we dramatically improve the viewer's QoE by minimizing interruptions. Joint-Family allows peers to smoothly adapt their quality and achieve a high playback rate for popular content. Not only that, even for unpopular content, Joint-Family achieves almost 40% higher playback rate than an existing P2P VoD system with a fixed bitrate (250Kbps), 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.

References

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

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

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

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

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

[6] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks," in *SIGCOMM 2004*, 2004.

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

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

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

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

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

[12] F. Lehrieder, G. Dán, T. Hossfeld, S. Oechsner, and V. Singeorzan, "Caching for BitTorrent-like P2P Systems: a Simple Fluid Model and its Implications," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, Aug. 2012.

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

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

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

[16] G. Tian and Y. Liu, "Towards Agile Smooth Video Adaptation in Dynamic HTTP Streaming," in *ACM CONEXT*, 2012.

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

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

[19] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proceedings of ACM MMSys*, 2011.

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

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

[22] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *ACM SOSP*, 2003, pp. 298–313.

[23] R. Petrocco, M. Eberhard, J. A. Pouwelse, and D. H. J. Epema, "Deftpack: A Robust Piece-Picking Algorithm for Scalable Video Coding in P2P Systems," in *ISM*, 2011.

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

[25] D. Wu, Y. Liu, and K. Ross, "Queuing Network Models for Multi-Channel P2P Live Streaming Systems," in *INFOCOM 2009*, april 2009.

[26] M. Wang, L. Xu, and B. Ramamurthy, "Linear Programming Models for Multi-Channel P2P Streaming Systems," ser. INFOCOM'10, 2010.

[27] Z. Wang, C. Wu, L. Sun, and S. Yang, "Strategies of Collaboration in Multi-Channel P2P VoD Streaming," in *GLOBECOM*, 2010, pp. 1–5.

[28] Y. Borghol, S. Ardon, N. Carlsson, and A. Mahanti, "Toward Efficient On-Demand Streaming with BitTorrent," in *NETWORKING*, 2010.

[29] P. Shah and J. francois Paris, "Peer-to-Peer Multimedia Streaming Using BitTorrent," in *IEEE IPCCC 2007*, 2007.

[30] Z. Wen, N. Liu, K. L. Yeung, and Z. Lei, "Closest Playback-Point First: A New Peer Selection Algorithm for P2P VoD Systems," in *GLOBECOM*. IEEE, 2011.

[31] L. D'Acunto, N. Andrade, J. Pouwelse, and H. Sips, "Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-Like VoD Systems," ser. ISM '10, Washington, DC, USA, 2010, pp. 89–96.