Abandonment and its Impact on P2P VoD Streaming

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

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

I. Introduction

Peer-to-peer (P2P) systems have become part of the mainstream content distribution environment, and are being considered for commercial deployment of video-on-demand (VoD) services [17]. Much of this progress can be attributed to previous research identifying the set of policies that enable robust and scalable P2P delivery systems. In this paper, we consider the problem of viewers abandoning videos part way through the viewing of the video. While mostly overlooked so far, we show that *abandonment* (also called viewer engagement in other work [10]) is a critical factor to consider since it directly affects the impact of various policies used for P2P VoD.

The two most important policies that determine P2P performance are the chunk- and peer-selection policies. File sharing systems have traditionally used a combination of Tit-for-Tat (TFT) as the peer selection and Rarest-first (RF) as the chunk selection as this combination offers the best tradeoff in terms of performance and fairness. Unfortunately, this combination does not work as well with streaming video, be it live or on-demand, since a video is generally consumed sequentially. Instead, an Earliest-First (EF) policy is a more natural chunk selection policy for video streaming. EF, however, is not compatible well with TFT as peers in different points of playback have very little content of mutual interest to exchange with each other. As a result, there has been a lot of work to identify hybrid chunk selections strategies (e.g., EF+RF) [6, 7, 11, 22, 23, 28] that find a compromise between the need of streaming to get sequential data and TFT's need for diversity.

While the design of EF+RF implicitly assumes that viewers watch entire videos, recent studies [14, 15] show that viewers of both short clips and full-length movies often watch only a small portion of a video and abandon the video part way through. Such abandonment may be attributed to how users find movies of interest (e.g., surfing for interesting content) or the possibility that a viewer loses interest in the content. Such viewer abandonment of videos has significant implications on the design of P2P policies. For example, peers using EF+RF may end up downloading rare chunks that they do not actually watch later due to abandonment. In this case, it would be more beneficial to use that upload capacity to deliver chunks that peer immediately need, to improve video playback experience and reduce unnecessary bandwidth consumption.

In this paper, we reconsider the set of chunk and peer selection policies to use in real-world P2P VoD systems with viewer abandonment (Section II). We show that EF is a more appropriate chunk selection strategy in the presence of abandonment. Instead of using TFT, we introduce Earliest-Deadline (ED) as the peer selection strategy. In ED, a node picks peers with the earliest deadline among chunks when deciding which request to serve. Choosing ED not only gives us substantial performance improvement (as seen in our experiment results), but also allows us to break the inter-dependence between chunk- and peer selection that TFT introduces. While EF itself reduces wasted download compared to EF+RF, we introduce the notion of a playback lookahead window (PLA) to further limit the download rate, as is used in HTTP streaming [2, 21]. The window is sized such that when it is full, the user will not encounter interruptions. At the same time, users do not request any more data than needed to avoid "wasting" bandwidth in case they abandon the video.

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

In this paper we make the following important contributions:

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

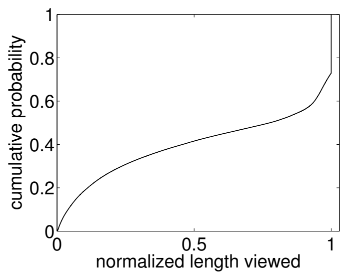


Fig. 1: Cumulative distribution of normalized length viewed of all requests in trace data

- In contrast to previous findings, our results show that with abandonment, a hybrid policy of EF+RF performs *worse* and results in longer interruption periods than EF. With EF, using ED instead of TFT brings significant performance enhancements.
- We develop a detailed analytical model that accounts for abandonment and partial seeds and show that "useful" download rate of EF effectively improves as partial seeds stay longer.
- We show that the combination of ED, EF, and partial seeds can significantly improve overall video playback performance, while PLA further reduces wasted bandwidth consumption.

II. Abandonment and P2P VoD

Recent studies [14, 15, 26] 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. Using the right chunk selection policy and peer selection policy, we balance the need to download the video fast enough to minimize interruptions and startup delay while also minimizing wastage of network resources.

A. Abandonment in Trace Data

Using traces from a large scale VoD service provider, Hwang et al. [14] demonstrate that user abandonment indeed occurs. They show that the video watching duration distribution depends on various aspects such as video length and popularity. In this paper, we also use trace data from a nationally deployed VoD service. Our data covers a two week period in 2010 and has millions of requests. The trace includes information about users' interactive operations (e.g, FastForward, Skip, Rewind, etc.), and we calculate the viewed length of each user taking into account these specific operations (more details described in Section IV-A). In Figure 1, 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. Thus, the assumption that each user views the entire video and therefore downloads the entire video file (as assumed by many existing P2P works), is not appropriate when considering real-world user behaviors.

B. Chunk Selection Policy

The chunk selection policy determines the order in which a peer downloads chunks. File sharing systems have traditionally used Rarest-First (RF) as the chunk selection policy. RF has several desirable properties including distributing rare chunks among peers and allowing the seed to quickly offload chunks. RF however is inherently unsuitable for streaming systems which require chunks to arrive in order. Instead, a chunk selection policy that attempts to get chunks close to playback is a more natural fit for video streaming. In fact, a significant amount of previous work [6, 7, 11, 22, 23, 28] has shown that the combination of EF and RF (EF+RF) incorporates the strengths of each policy and results in the best playback continuity with P2P VoD.

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

C. Peer Selection Policy

The peer selection policy determines the subset of requests that are served by a peer upon receiving requests. Unlike chunk selection where multiple options are used in practice, most systems use TFT as the peer selection policy. TFT works by forcing peers to upload data in order to download content. This allows peers to disseminate content quickly to high bandwidth peers and eliminates free-riding. However, it requires that peers have content to exchange with each other. This, however, has the unfortunate side effect of introducing interdependency between chunk- and peer selection policies. Since peers have to upload some data in order to be able to download (setting aside considerations of optimistic unchoking used by TFT), peers need to have a diverse set of chunks. This is not an issue with RF as it is designed to create such diversity. With EF, however, peers at different points of their playback will not have content of mutual interest to exchange with each other. In addition, there is growing realization that there are inefficiencies due to TFT [13, 19] in streaming systems.

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

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

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

D. Handling Free Riding

Free riders in P2P systems can significantly impact the overall system performance and introduce unfairness. TFT was designed specifically to prevent such free riding. However, as stated earlier, TFT introduces dependency on chunk selection that is incompatible with P2P streaming. Hybrid chunk selection policies seek a middle ground. There have even been efforts to change seed policies [8] or to cluster peers at similar playback points and introduce diversity within these clusters [13]. Unfortunately, abandonment significantly diminishes the effectiveness of such approaches, as seen in Section IV.

Our approach in this paper is to deviate from TFT and instead use ED. While ED does not guarantee against free riders, ED offers better performance in terms of streaming and quality-of-experience compared to TFT. The decoupling from the chunk selection policy allows us to overcome inefficiencies due to TFT [13, 19] in deployments that do not worry about free riding (e.g., managed content delivery [17]). In scenarios where eliminating free-riding is still important, we can adopt approaches like Contracts [19] or iPASS [16], appropriately modified, for P2P VoD. We are exploring these strategies as part of our ongoing work.

E. Using "Partial" Seeds

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

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

F. Reducing Wastage by Limiting Playback Lookahead

The primary objective of the chunk selection policies is to sustain a sufficient rate so that a viewer does not experience interruptions. Our combination of EF and ED with partial seeds allows us to achieve a rate comparable, if not better, than TFT (Figure 16). However, unlike the ED+EF policy, TFT with RF-based chunk selection (e.g., the hybrid policy EF+RF) generates wastage by unnecessarily propagating later chunks that are not likely to be watched when a viewer abandons the video (Section IV-E).

In order to further limit the wastage with ED+EF, we adopt a playback lookahead window (PLA) that is measured in number of chunks. The PLA restricts excessive downloads of chunks beyond the current playback point. Specifically, when a peer has downloaded a predefined number (equal to the PLA window) of consecutive chunks ahead of its current playback point, it stops requesting further downloads until the playback progresses and the window 'opens' up. The window also moves forward as the playback progresses. By adjusting the PLA window size, ED+EF greatly reduces the bandwidth wastage without hurting playback continuity. We note that this method of limiting the delivered rate has been widely used for server-based video streaming protocols such as HTTP streaming [21], progressive download [12], and adaptive bitrate streaming [2]. Our results show that P2P systems can also achieve substantial reduction in wastage by implementing a rate limiting capability in the form of a playback lookahead.

III. Abandonment Analysis

In this section we analyze how abandonment affects swarm population and useful download rates for EF and RF schemes. Aalto et al. [1] analyze a viewer's abandonment in P2P VoD as a stochastic queueing model, where leechers abort and leave the swarm instantly. Thus, they ignore the concept of partial seeds. Our analysis builds on the work by Qiu et al. [20] and Parvez et al. [18] but also takes abandonment into account.

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

A. Swarm Population

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

For the system to become the steady state, the rate of data loss due to the departures of normal seeds and partial seeds

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

TABLE I: Parameters and Definition for Analysis

should be equal to the system goodput [4] as follows,

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

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

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

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

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

$$\bar{y} = \beta\frac{\lambda}{\mu_1} \tag{3}$$

$$\bar{z} = (1-\beta)\frac{\lambda}{\mu_2} \tag{4}$$

From Equations (2), (3), and (4), $\bar{x}$ is

$$\bar{x} = \lambda\{\frac{\{\beta + (1-\beta)\rho\}B}{UC} - \frac{\beta}{\mu_1} - \frac{1-\beta}{\mu_2}\} \tag{5}$$

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

$$\frac{\beta}{\mu_1} + \frac{1-\beta}{\mu_2} < \frac{\{\beta + (1-\beta)\rho\}B}{UC} \tag{6}$$

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

EF swarm size: Following [18], from the viewpoint of any given peer A, there are *younger* peers who arrived after A and *older* peers who arrived before A. With EF A can only download from its older peers (with more chunks) and can only provide content to younger peers. An uploader that receives more than U requests chooses to serve U requests at random and rejects the rest.

Consider a peer that has been in the system for time t_m. The probability that the peer is successful in obtaining a download connection for its next desired chunk is defined as: $p(t_m) = \frac{\tilde{U}(t_m)}{\tilde{D}(t_m)}$, where $\tilde{U}(t_m)$ and $\tilde{D}(t_m)$ are the connection supply and demand at time t_m. A peer of age t_m requests a download connection from older peers ($t > t_m$).The total number of possible upload connections available for this peer is $\tilde{U}(t_m) = (x + y + \delta z - \lambda t_m)U$. For computing $\tilde{D}(t_m)$, we first note that the total number of download requests in the system is xD and that peers with more chunks (including seeds and partial seeds) receive higher demand. In [18] $\tilde{D}(t_m)$ is indirectly calculated by finding the total number of download requests handled by peers younger than t_m and subtracting it from xD, and is approximated as $\tilde{D}(t_m) = \frac{xD}{\alpha}$ where α depends on the system parameters. However, we approximate α as a constant, and its range is $[1.09, 1.25]$ for typical scenarios in [18] (we will provide the upper bound on α to achieve the steady state system in Inequality (11)).

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

$$\{\beta + (1-\beta)\rho\}B\lambda = (\frac{\bar{x}}{2} + \bar{y} + \delta\bar{z})\alpha UC \tag{7}$$

where the right hand side comes from $\gamma_{EF}x$. $\bar{y}$ and $\bar{z}$ are independent of chunk selection schemes (i.e., the same as Equation (3) and (4), respectively). From Equations (3), (4), and (7) we obtain $\bar{x}$:

$$\bar{x} = 2\lambda\{\frac{\{\beta + (1-\beta)\rho\}B}{\alpha UC} - \frac{\beta}{\mu_1} - \frac{(1-\beta)\delta}{\mu_2}\} \tag{8}$$

Since we assume $\bar{x} > 0$, similar to Inequality (6) we have the following condition on μ_1 and μ_2 for the EF scheme:

$$\frac{\beta}{\mu_1} + \frac{(1-\beta)\delta}{\mu_2} < \frac{\{\beta + (1-\beta)\rho\}B}{\alpha UC} \tag{9}$$

The total EF swarm population is

$$\bar{x} + \bar{y} + \bar{z} = \lambda\{\frac{2\{\beta + (1-\beta)\rho\}B}{\alpha UC} - \frac{\beta}{\mu_1} - \frac{(2\delta-1)(1-\beta)}{\mu_2}\} \tag{10}$$

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

Note that we should limit α such that $(\bar{x}+\bar{y}+\bar{z})UC \geq \{\beta + (1-\beta)\rho\}B\lambda$ since the left hand side indicates the aggregate upload bandwidth of all peers in the system and should be equal to or larger than the system's goodput $(\frac{\bar{x}}{2} + \bar{y} + \delta\bar{z})\alpha UC$ in Equation (7). Therefore, using Equations (10), α has the following upper bound:

$$\alpha < \frac{2\{\beta + (1-\beta)\rho\}B\mu_1\mu_2}{\{\beta + (1-\beta)\rho\}B\mu_1\mu_2 + \{(2\delta-1)(1-\beta)\mu_1 + \beta\mu_2\}UC} \tag{11}$$

B. Useful Download Rates

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

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

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

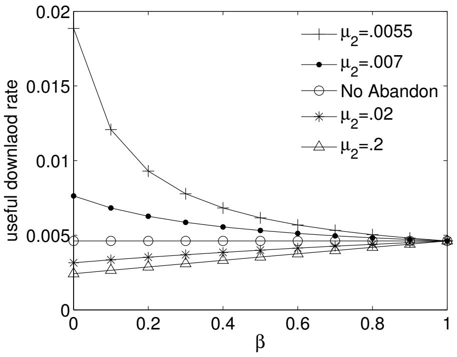


Fig. 2: Useful download rate of EF vs. β ($\mu_1 = 0.01$)

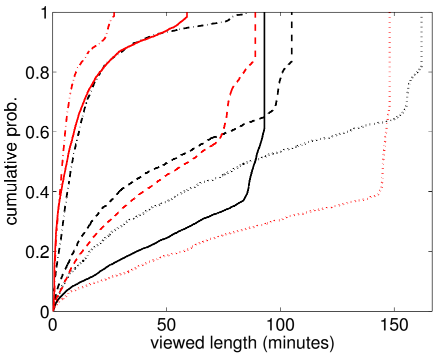


Fig. 3: CDF of viewed length for 8 videos

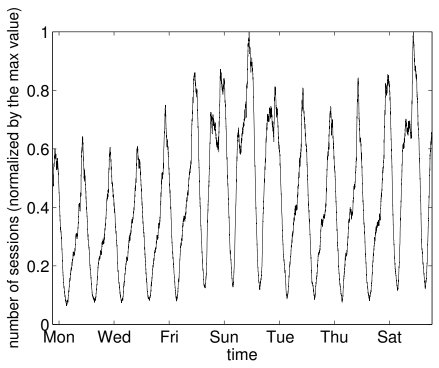


Fig. 4: Number of concurrent sessions for a random video in the real trace.

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

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

The useful download rate for RF with abandonment is obtained by scaling γ_{RF} with the probability of useful chunks that have been downloaded up to time t. This scaling factor is shown to be $\frac{1}{M-k+1}$ [11]. Therefore:

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

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

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

$$\bar{\gamma}_{EF} = \gamma_{EF} = \alpha UC(\frac{y}{x} + \frac{1}{2}) \quad (12)$$

EF with abandonment: Similarly,

$$\bar{\gamma}_{EF} = \gamma_{EF} = \alpha UC(\frac{y}{x} + \frac{\delta z}{x} + \frac{1}{2}) \quad (13)$$

Figure 2 shows the impact of partial seeds on the useful download rate for EF. Using Equations (12) and (13), we apply $\alpha = 1.2, \rho = 0.5, 1/\mu_1 = 100, M = 100, U = 4$, and $C = 0.001$ assuming the unit file size $B = 1$. We also set $\delta = 0.5$ assuming the distribution of the number of downloaded chunks at each partial seed is uniformly distributed, and thus on average only half of the partial seeds have chunks at or beyond the point defined by the desired chunk. Note that by Equation (9) $\mu_2 > 0.0048$ should be satisfied for the system to remain under the assumption of upload capacity constraint. When partial seeds stay for a short period of time (e.g., $\mu_2 \geq 0.02$), the useful download rate with abandonment is smaller than the case without. As partial seeds continue to reside for extended periods of time (potentially longer than the normal seeds), the useful download rate with abandonment is seen to be significantly higher than without. Thus, in a system with abandonment, a partial seed's staying time significantly influences the useful download rate. In case of $\mu_2 = 0.0055$ or 0.007, the useful download rate decreases with increasing β since normal seed staying time is relatively shorter than partial seed staying time (note $\mu_1 = 0.01$). Comparing EF and RF, as M becomes larger, EF's useful download rate is much higher than for RF (not shown) confirming earlier results.

IV. Evaluation

A. Data Set

To reflect realistic viewing patterns of a large population of users, we collected trace data from a nationally deployed Video-on-Demand service serving millions of customers. We examine a "heavy-viewing" period of 14 consecutive days in 2010. We focus on the trace from a single large metropolitan area, totalling approximately 1 million requests. The trace data contains information for each viewing session: an anonymized user ID, user request time, video ID, video length, and the duration viewed. To ensure user privacy, all user data is kept anonymous and was analyzed in aggregate, without the ability to identify each user. The trace also has information about the set of DVD-like operations that the user performed while watching the video in a given session. We use the duration viewed in the trace as session duration (time elapsed since the user request time) in the simulation, at which point the peer abandons the video. Note that the final playback point of the video in the simulation may be shorter than the session duration (due to startup delay, interruptions, etc.). We view this difference as an indicator of the performance of the system (smaller the better).

To obtain representative results, we repeat the experiment independently with 8 different popular videos that have different lengths across a wide range, from 30 to 150 minutes. They show different abandonment patterns as in Figure 3. They also show a clear daily pattern in their request volume. For example, Figure 4 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 these 8 videos in this section.

B. Experiment Setup and Assumptions

To evaluate our approach, we use a discrete event-driven BitTorrent simulator [5]. We enhance the original P2P file sharing simulator for video streaming, so that each peer waits till a playback buffer fills up and then starts playing back as the download progresses. In addition to original TFT, we also experiment with ED, EF, and EF+RF to compare the following set of schemes:

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

Parameter	Default
Number of initial seeds	1
Upload bandwidth of an initial seed	3 Mbps
Peer download/upload bandwidth	5 Mbps/ 1 Mbps
Max. concurrent uploads (seed)	15
Max. concurrent uploads (peer)	5
Video bitrate	1 Mbps
Chunk size	10 seconds
Peer arrival rate (synthetic trace only)	$\lambda = 0.05$ (Poisson arrival)

TABLE II: Simulation parameters and their default values.

We further enhance EF+RF to reduce the startup delay as follows. Instead of simply selecting EF or RF based on a probability [7, 11, 23, 28], a peer in the enhanced scheme initially uses EF only, but switches to EF+RF only if it has enough chunks in the playback sequence. In our experiments, if there are 5 or more chunks, a peer uses EF with probability of 0.7 and RF with probability of 0.3.

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

For the trace driven simulation, peers make requests for a video at the time instants specified in the trace. Peers that download the entire video convert to normal seeds while those that abandon the video part-way become partial seeds. Since the trace does not tell us when nodes depart, we model the staying time of normal seeds and partial seeds as an exponential distribution with the average of $1/\mu_1$ and $1/\mu_2$ seconds, respectively. After this time, normal or partial seeds also permanently leave the system.

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

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

C. Impact of Abandonment

We first investigate how user abandonment affects the performance of different combinations of chunk selection and peer selection policies. We vary the initial seed capacity in Figure 5 and record the resulting NIT. Note that when we vary the capacity of the initial seed, we accordingly adjust the maximum number of its concurrent uploads allowed (e.g., 10 concurrent uploads with 2Mbps upload capacity, 20 with 4Mbps, etc.). Figure 5 shows that all three schemes (TFT+EF, TFT+[EF+RF], ED+EF) have larger NITs in presence of abandonment than without abandonment. This indicates that while the absolute time of interruption might be smaller with abandonment, the proportional impact of interruption is larger with abandonment. Proportion is more important because if a viewer is interrupted for longer, there is the further likelihood that he/she may abandon the video earlier [10, 15]. Clearly, a higher seed capacity benefits all the schemes. Also importantly, many existing works have suggested the desirability of using a EF+RF hybrid scheme for P2P VoD. However, we observe that with TFT, EF+RF hybrid actually causes larger NITs compared to EF when user abandonment exists. This is because with abandonment, chunks closer to the end of a video are viewed rarely. Exchanging rare 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 has the smallest NITs. This shows the importance of accounting for abandonment; something that earlier works have overlooked. This also shows that serving peers with most urgent chunks helps improve overall user experience. We also measured the startup delay for each approach. We do not observe a significant difference between different approaches

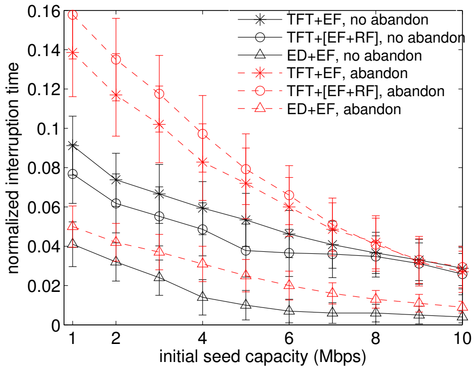


Fig. 5: Comparing NITs of different combination of chunk- and peer- selection policies when user abandonment exists and when it does not (with 95% confidence interval).

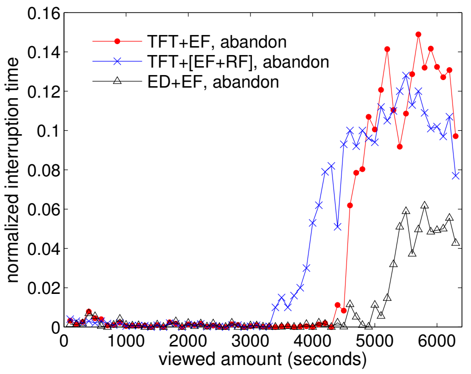


Fig. 6: Average NITs of each peer group divided based on viewed length. The initial seed capacity is 3Mbps.

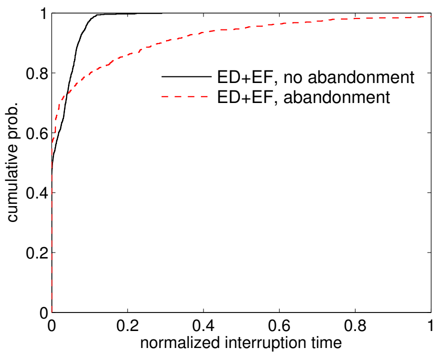


Fig. 7: Cumulative distribution of NITs when user abandonment exists and when it does not, respectively.

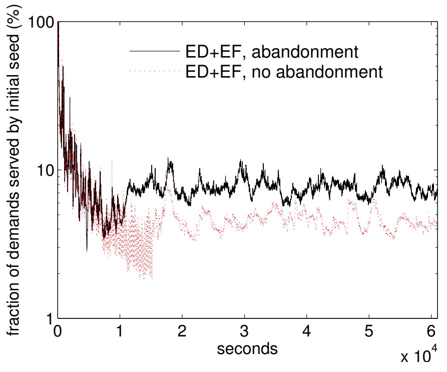


Fig. 8: The fraction of demand (in terms of bytes) satisfied by the initial seed over time.

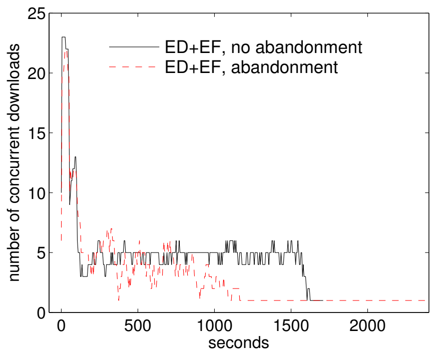


Fig. 9: Number of concurrent downloads of a peer over time. The solid curve ends at 1610 seconds.

whether there is abandonment or not; this is not surprising as they all use EF at startup.

In Figure 6, we use one of the longer videos (105 minutes), group peers based on how much they watched, and plot the average NIT for each group. Specifically, we divide the video into 100 second bins, and group the viewers into these bins based on how much they watch (i.e., the first group includes peers who watched 0–100 seconds of the video, the next group watched 101–200 seconds of the video, etc.). The initial seed upload capacity is 3Mbps. We observe that peers who watch for a long period have larger NITs than peers who watch for a shorter interval. We also note that TFT+[EF+RF] reduces NITs compared to TFT+EF for peers who watch the video longer than 4800 seconds. However, for most of peers who watch less than 4800 seconds, TFT+[EF+RF] causes more interruption. As a result, TFT+[EF+RF] results in larger overall NITs than TFT+EF in the presence of abandonment, just as we saw in Figure 5. As before, the use of ED+EF results in consistently lower interruption than the other two policies.

To understand the cause for these results and their relationship to abandonment, we first compare NITs of each view as CDFs in Figure 7 between the case when abandonment exists and when abandonment does not exist. The initial seed capacity is 3Mbps, and all peers use ED+EF. We observe that with abandonment some views have very long NITs, such as NIT ≥ 1. Therefore, we now focus on the peers who have NITs larger than 1 to understand how abandonment makes their playback performance worse.

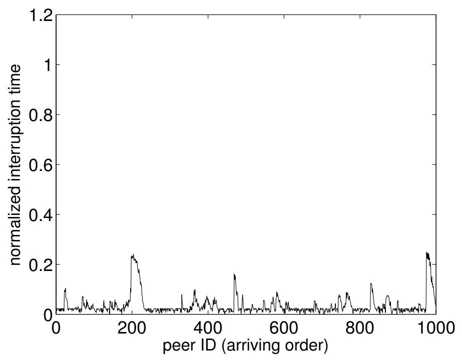
Specifically, we conduct an in-depth study step by step using a simple synthetic trace for a better understanding. While in the real trace peer arrival patterns are fixed, with the synthetic trace we have full control over peer arrivals. By adjusting arrival rates and patterns, we are able to more clearly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

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

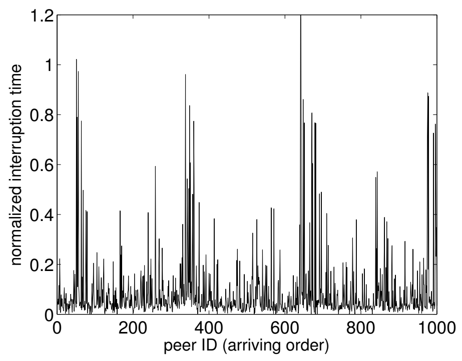
First from the synthetic trace results, we compare the load on the initial seed between with and without abandonment in Figure 8. With abandonment, the load as a byte fraction of requests served by the initial seed is larger than without abandonment. This is because peers leave early with abandonment causing loss of upload capacity available. This result indicates that abandonment imposes more critical role on the initial seed. Note that the initial large drops till the first 10^4 seconds for the both curves indicate that the swarm size is initially not yet stable but is growing till it becomes stable.

Then, we monitor the number of concurrent chunk downloads at each peer over time who has NIT ≥ 1 and observe that those peers have a similar trend to that presented in Figure 9. We see that when the peer joins the swarm, 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 initial seed and never increases until the peer abandons the video. On the other hand, without abandonment, although the peer loses lots of download connections in a similar manner, it manages to maintain about 4–6 parallel downloads. Also, the downloads end at 1610 second without abandonment, which means that the download finishes earlier than the actual playback. This trend indicates that with abandonment older peers (i.e., those who arrived earlier than this peer) have all left at about the 1100 second mark, and therefore this peer loses all its possible uploaders other than the initial seed. We note that this trend is strongly related to our EF chunk selection policy since younger peers cannot help older peers with EF. However, we will show that although using EF+RF may alleviate this issue, EF+RF results in more peers having interruptions than EF only, with abandonment.

More importantly, losing older peers seen in Figure 9 occurs more severely with abandonment because viewers watch different length of the video. If a peer watches for a longer period than its older peers, that peer would be more likely to lose its potential uploaders early. In Figure 10(a) and 10(b) we show NITs of each peer in an arriving order, when



(a) NITs of ED+EF with no abandonment



(b) NITs of ED+EF with abandonment

Fig. 10: NITs of 1000 peers sorted in their arriving order

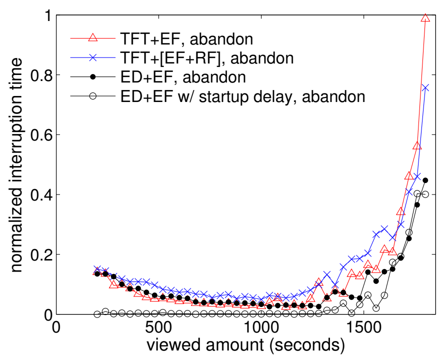


Fig. 11: Average NITs of each peer group divided based on viewed length. The synthetic trace used.

abandonment does and does not exist, respectively. While a similar set of peers experience larger NITs in both cases, we observe that the magnitude is much larger in the case of abandonment.

In Figure 11, we divide peers into different groups based on their viewed length, and plot the average NITs of each group for the synthetic trace, similarly to Figure 6 with the real trace. Each group has a 40 second range of viewed length. We now clearly observe that NITs grow superlinearly as a peer watches the video for a longer time. We also note that TFT+[EF+RF] reduces NITs compared to TFT+EF for peers who watch for a very long time (more than 1640 seconds). This is because, by using RF, older peers have a chance to download from younger peers as well. However, for most of peers who watch for short durations, RF causes more interruption by exchanging chunks closer to the end of the video; but those chunks are rarely viewed. Peers who watch for a very short time, even smaller than 500 seconds, have slightly larger NITs than peers who watch around 500–1200 seconds. This is because, as stated earlier in this section, we do not consider startup delay for synthetic trace experiments. To confirm this, we also plot the results when peers have a startup buffer of 10 seconds of video just like the experiment with the real trace, and we see that NITs for peers who watched less than about 1300 seconds of the video with ED+EF is almost 0. Comparing Figures 11 and 6, 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.

D. Utilizing Partial Seeds

We also investigate the effect of utilizing partial seeds. To understand the potential of seeds staying on in practice, we present results from our request traces collected at the set-top boxes of viewers. Specifically, for each customer, we calculate the distribution of the time between the completion of one video and the start of the next video request. Based on this, we determine how long each video would be available in a customer's set-top box (Figure 12). We observe that in over 45% of the occurrences, there is at least 1000 seconds of time the seed (whether it is a partial or normal seed) can continue to stay and serve an existing swarm before the user starts viewing another video.

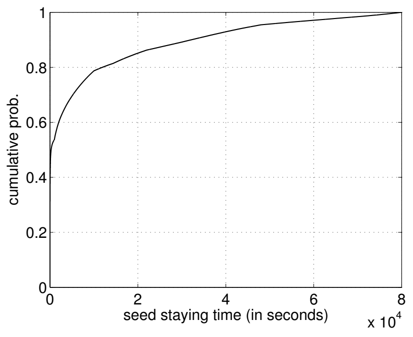


Fig. 12: Seed staying time from trace

In Figure 13, we plot NITs of different chunk- and peer-selection strategies as a function of the average staying time of partial seeds ($1/\mu_2$ in Section III) when the initial seed capacity is 3Mbps with a maximum of 15 concurrent uploads. Note that we also make normal seeds who have downloaded the entire video stay on as long as the partial seeds, i.e., $\mu_1 = \mu_2$. In the presence of peer abandonment, having peers staying on as partial seeds benefits all of the strategies. Not surprisingly, the benefit increases as the staying time of partial seeds increases. As shown in Figure 14, as the staying time increases, NIT decreases significantly, especially for the viewers with larger viewed lengths. For verification, we also repeat the partial seed experiments with the synthetic trace used in Section IV-C, and NITs of peers with long views gradually decrease as the partial seeds stay longer, as shown in Figure 15.

E. Minimizing Wastage

We measure the bandwidth wastage caused by abandonment for the three different policy combinations, and also investigate how utilizing partial seeds impacts wastage. First, we investigate how the peer selection policies, TFT and ED impact wastage. When comparing the download rates of TFT+EF and ED+EF, we see that the average download rates of TFT+EF are higher than ED+EF (e.g., 1.57 vs. 1.20 Mbps with abandonment but no partial seed staying). However, the distribution of download rates is quite revealing. Figure 16 shows that more than 80% of peers achieve download rates higher than the video streaming rate (1Mbps) with ED+EF. TFT+EF on the other hand has higher variability; some peers get high rates, while many fall below 1Mbps because they struggle to

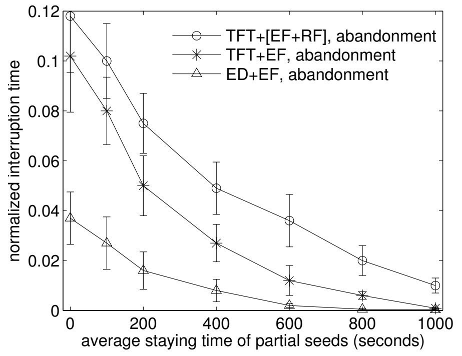


Fig. 13: NITs of three different schemes in presence of abandonment as a function of partial seed staying time.

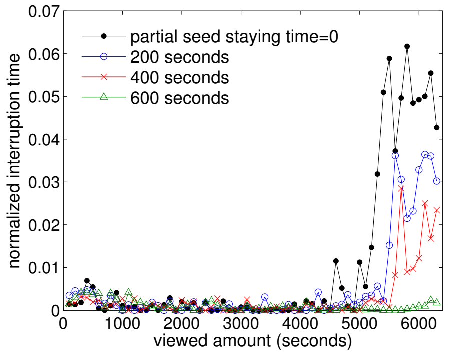


Fig. 14: Average NITs of each peer group divided based on viewed length ($\mu_1 = \mu_2$). ED+EF used. Real trace used.

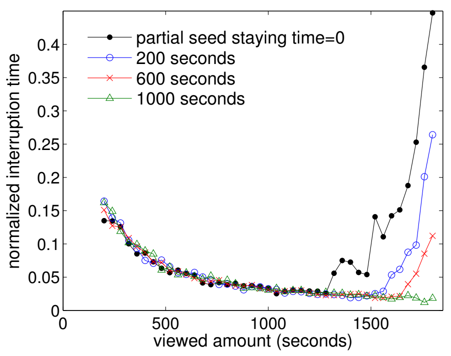


Fig. 15: Average NITs of each peer group divided based on viewed length ($\mu_1 = \mu_2$). ED+EF used. Synthetic trace.

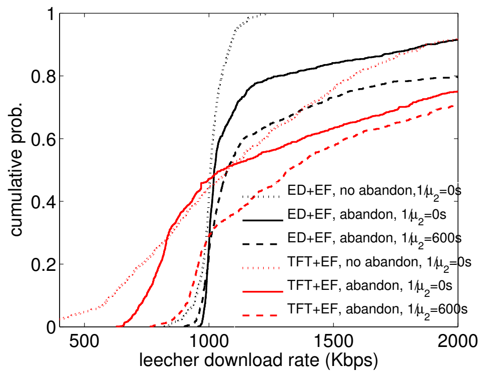


Fig. 16: CDFs of leecher download rate ($\mu_1 = \mu_2$). $1/\mu_2 = 0$ indicates partial seeds leave immediately after abandoning.

get unchoked, which is a key deficiency of such a policy for streaming video. Consequently, we have more wastage with TFT schemes as seen in Figure 17(a). TFT+[EF+RF] causes more wastage than TFT+EF due to exchanging rare chunks by RF. ED+EF results in the least amount of wastage. Also, we see that as partial seeds stay longer in the swarm, wastage grows for all the schemes, since download rates of both TFT and ED schemes increase by having partial seeds contribute, as shown in Figure 16 (with $1/\mu_2 = 600$ secs).

We performed experiments by varying the size of the playback look-ahead (PLA) window (described in Section II-F) to achieve a balance between bandwidth wastage (Figure 17(b)) and playback quality (Figure 17(c)). We observe that with a modest PLA window size (e.g., 5 chunks), ED+EF can achieve almost the same level of playback performance but achieve much lower wastage - by almost 85% (reducing from wastage 26% to 4%). In contrast, a smaller PLA values cause more playback interruptions.

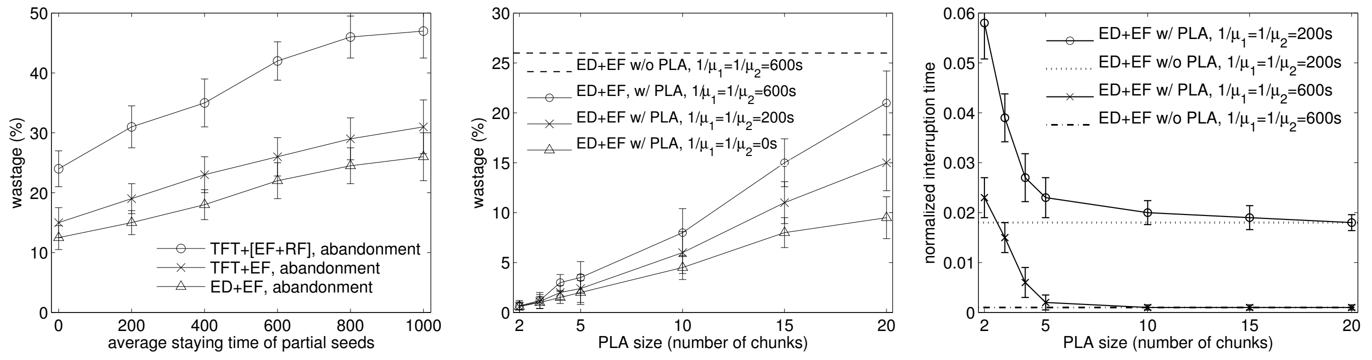
V. Related Work

Abandonment: Hwang et al. [14] and Li et al. [15] present a measurement study on viewer's behavior with abandonment in a large scale landline-based and mobile-based IPTV services provider, respectively. 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. [1] 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.

Chunk selection: To adapt BitTorrent for streaming systems (either live or VoD), a combination of the rarest first (RF) chunk selection policy and sequential chunk download (EF) has been exploited [6, 7, 11, 22, 23, 28]. The specific details of the proposed schemes vary from simple probabilistic hybrid models to using sophisticated network coding techniques. Previous literature claims that achieving balance between system utilization (by RF) and on-line playback (by EF) can substantially improve playback quality. However, we show that with viewers' abandonment, such a hybrid policy greatly degrades the playback performance. We further show that using EF only achieves better playback performance.

Peer selection: BitTorrent's TFT is effective for file sharing with its inherent incentive mechanism to encourage a peer's contribution. However, several prior works [9, 13, 19, 22, 24, 25, 27]show that TFT is not suitable for streaming applications. This is primarily because chunk selection using RF is not suitable for streaming, and TFT without RF makes it difficult for new peers to contribute to older peers, thus preventing them from fully helping each other. Various peer selection approaches have been proposed for streaming. Shah et al. [22] modify TFT's optimistic unchoke, D'Acunto et al. [9] make peers act more altruistically, and Wen et al. [24] 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.

Limiting Rate to Avoid Wastage: Popular VoD services such as YouTube (using Progressive Download) and Netflix (using Adaptive Bit Rate) have adopted approaches that limit the amount of video bytes delivered beyond the current playback point so as to limit wastage and also reduce the load on the network for VoD streaming. However, most of these approaches have been applied for server-based environments such as HTTP streaming [21], progressive download [3, 12], and adaptive bitrate streaming [2]. A similar capability is desired for P2P VoD streaming. We show that our approach of having a limited 'look-ahead' window can also reduce wastage caused by abandonment in P2P systems, while not hurting the viewer's playback experience.



(a) Wastage and partial seed staying time (b) Wastage and PLA (c) NITs and PLA

Fig. 17: Wastage of different approaches, varying partial seed staying time and PLA window size, with 95% confidence interval.

VI. Conclusion

In this paper, we demonstrated that user abandonment of videos can impact P2P VoD streaming performance significantly. In all the schemes we considered in this paper, abandonment caused larger interruptions (NITs), particularly with peers watching longer as they are isolated with no other peers to upload from. With abandonment, distributing rare chunks (by RF or EF+RF hybrid) becomes wasteful and performs worse than EF, as peers are likely to abandon before consuming the downloaded rare chunks. Through analysis and trace-driven simulations we show that our design that combines ED for peer selection, EF for chunk selection, and the use of partial seeds outperforms existing well-known schemes by significantly improving overall video playback performance and reducing wasted bandwidth consumption. Additionally, we further reduce wastage by peers having a playback lookahead window. As part of our future work, we plan to investigate strategies to eliminate free-riding.

Acknowledgement

We would like to thank our shepherd, Fabien Mathieu, as well as the anonymous reviewers for their valuable comments that helped to improve the paper significantly.

References

[1] S. Aalto, P. Lassila, P. Savolainen, and S. Tarkoma. How Impatience Affects the Performance and Scalability of P2P Video-on-Demand Systems. In *SIGMETRICS MAMA '11*, June 2011.

[2] S. Akhshabi, A. C. Begen, and C. Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. In *ACM MMSys*, 2012.

[3] S. Alcock and R. Nelson. Application Flow Control in YouTube Video Streams. *SIGCOMM Comput. Commun. Rev.*, 41(2), Apr. 2011.

[4] F. Benbadis, F. Mathieu, N. Hegde, and D. Perino. Playing with the Bandwidth Conservation Law. In *P2P '08*, 2008.

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

[6] Y. Borghol, S. Ardon, N. Carlsson, and A. Mahanti. Toward Efficient On-Demand Streaming with BitTorrent. In *IFIP Networking*, 2010.

[7] N. Carlsson and D. L. Eager. Peer-Assisted On-Demand Streaming of Stored Media Using BitTorrent-like Protocols. In *Networking*, 2007.

[8] N. Carlsson, D. L. Eager, and A. Mahanti. Peer-assisted On-demand Video Streaming with Selfish Peers. In *IFIP Networking*, 2009.

[9] L. D'Acunto, N. Andrade, J. Pouwelse, and H. Sips. Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-Like VoD Systems. In *ISM*, 2010.

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

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

[12] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis. Trickle: Rate Limiting YouTube Video Streaming. In *USENIX*, 2012.

[13] 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.

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

[15] Y. Li, Y. Zhang, and R. Yuan. Measurement and Analysis of a Large Scale Commercial Mobile Internet TV System. In *IMC*, 2011.

[16] C. Liang, Z. Fu, Y. Liu, and C. W. Wu. Incentivized Peer-Assisted Streaming for On-Demand Services. *IEEE Trans. Parallel Distrib. Syst.*, 21(9):1354–1367, Sept. 2010.

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

[18] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson. Analysis of BitTorrent-Like Protocols for On-Demand Stored Media Streaming. In *SIGMETRICS*, 2008.

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

[20] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In *SIGCOMM '04*, 2004.

[21] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network Characteristics of Video Streaming Traffic. In *CoNEXT*, 2011.

[22] P. Shah and J. francois Paris. Peer-to-Peer Multimedia Streaming Using BitTorrent. In *IPCCC*, 2007.

[23] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *9th IEEE Global Internet Symposium*, April 2006.

[24] 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.

[25] Y. Yang, A. L. H.Chow, L. Golubchik, and D. Bragg. Improving QoS in BitTorrent-like VoD systems. In *INFOCOM*. IEEE Press, 2010.

[26] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min. Inside the Bird's Nest: Measurements of Large-Scale Live VoD from the 2008 Olympics. In *ACM IMC*, 2009.

[27] H. Zhang, S. Vasudevan, R. Li, and D. Towsley. A Case for Coalitions in Data Swarming Systems. In *ICNP*, 2011.

[28] Y. Zhou, D. M. Chiu, and J. Lui. A Simple Model for Analyzing P2P Streaming Protocols. In *ICNP*, 2007.