Noname manuscript No.
(will be inserted by the editor)

The effect of incomplete sessions on peer-to-peer video on demand

Miklós Máté, Rolland Vida, and Attila Mihály

the date of receipt and acceptance should be inserted later

Abstract Video on demand systems are going to be important services of the future Internet. To achieve scalability and fault tolerance, these systems should rely on distributed video delivery schemes, using peer-to-peer (P2P) networks built among the clients. However, generic P2P data sharing protocols cannot guarantee timely arrival of the video segments. Therefore, they must be adapted to video streaming by restricting P2P delivery into a download window. In this paper we investigate the effect of the frequently occurring phenomenon of incomplete sessions on the efficiency of stored video streaming via P2P data distribution. Through theoretical considerations and a simulation study, we've found that with a P2P download window the number of superfluous downloads can be limited, while retaining the efficiency of the P2P distribution.

Keywords incomplete sessions, P2P, video on demand

1 Introduction

The currently popular online video on demand (VoD) services, like Netflix and Hulu, or the aggregators of user-generated content, like YouTube, are just the first step towards the truly interactive multimedia experience of the future. However, a wide range of content, easy navigation, and competitive pricing are not enough to replace conventional television, as network operators see these services as a huge threat against their deployed network capacity, and try to discourage their customers via pricing.

VoD systems have to use a scalable and resilient architecture in order to overcome the network limitations and reach the masses. A single content server is both fragile and expensive: it represents a single point of failure, it needs very high capacity, and it generates huge amounts of network traffic, as it cannot be close enough to all possible clients. To overcome these limitations the content library is usually (partially) replicated at several sites, thus dividing the load, and increasing fault tolerance. These replica servers can also optimize the overall network load, because they are installed in the vicinity of the clients [7].

Distributing videos is also possible via decentralized peer-to-peer (P2P) systems [1]. These schemes let the clients share the data they have already downloaded; thus, the required network capacity at the content server does not increase linearly with the number of clients. P2P systems can even function without any central content server, and they do not require such a strict management as in case of caching. Peers don't have to be always connected, but join the network on a voluntary basis; thus, the fault tolerance of P2P solutions is usually higher.

P2P systems come in wide variety, from distributed content indexing to distributed data storage. They can be structured (tree or mesh based), or unstructured, the latter being more fault tolerant, but less efficient. Some of the most popular P2P file sharing systems include Gnutella2 [12] and BitTorrent [3]. P2P-assisted commercial VoD systems have also been deployed, e.g. Joost and PPLive.

It is known that users often do not watch the entire videos, and several video caching system designs consider the effect of incomplete sessions. However, to our

M. Máté and R. Vida
Budapest University of Technology and Economics
E-mail: {mate,vida}@tmit.bme.hu

Attila Mihály
TrafficLab, Ericsson Research
E-mail: Attila.Mihaly@ericsson.com

knowledge we are the first to examine the effect of this phenomenon on stored video streaming via P2P. In this paper we analyse the effects of windowing BitTorrent to adapt it to video streaming. We show that windowing can guarantee timely arrival with high probability, and the reduced number of active sessions increases the efficiency of P2P distribution.

The rest of the paper is organized as follows. In section 2 we summarise the related work on P2P VoD and the incomplete sessions. In sections 3, 4, 5, and 6 we analyse how windowing affects the efficiency of BitTorrent. We present results of our simulation study in sections 7, and 8. Finally, we draw the conclusions in section 9.

2 Related Work

2.1 P2P Video on Demand

Several different P2P architectures have been proposed for video delivery, most of them trying to offload central servers. Tree-based solutions [6, 5] build forking forwarding chains, with the server in the root. However, the presence of a central server in most designs is important, as the peers are considered unreliable. They enter and leave the P2P swarm at will, but the service quality must be kept above a threshold.

The most popular mesh-based P2P protocol applied for video streaming is BitTorrent [3], because it is very popular as a file sharing system, and it is relatively simple, but efficient and flexible. It divides the data into small pieces, called chunks, and those chunks are exchanged by the participating peers. Those having all the pieces are called seeders, while the others are called leechers. The incentive to upload is ensured through the peer selection mechanism, which is based on a tit-for-tat scheme; a node prefers uploading to peers that have been willing and efficient to upload themselves in the past, this scheme being called unchoking. To achieve high throughput, BitTorrent employs the rarest-first chunk selection scheme, which eliminates the bottlenecks by equilibrating the availability of the chunks.

However, BitTorrent is not suitable for streaming video without modifications, exactly because of this rarest-first scheme. If the chunks are video segments, they have to arrive more-or-less in playback order, or either the users have to suffer from very high startup delay (they must wait until most of the video is downloaded), or several segments have to be downloaded from the central server to fill the gaps. Several improvements have been proposed to address this limitation, usually by adding a sliding window ahead of the playback position, and preferring to download chunks within that window (BASS [4], BiToS [13], Toast [2]).

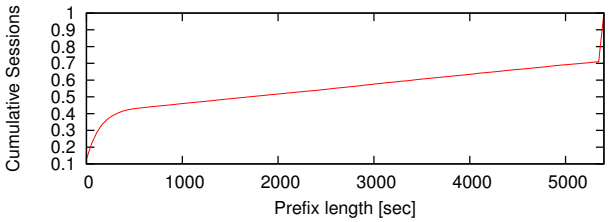
Numerous papers analysed the efficiency of P2P video distribution systems analytically as well. There are two research tracks that are important for us. In [9] the mean of the achievable throughput and the starting latency for in-order and rarest-first segment selection schemes are analyzed with a fluid model, based on the one presented in [10] for the unmodified BitTorrent protocol. There are analyses on live video streaming systems as well, where determining the optimal buffer size [14] and the priorities for downloading each segment in the buffer [17] are the important questions. Live streaming is fundamentally different from stored video streaming (e.g., the playback position is the same for all clients, none of them has any segments outside the playback buffer, and the missed segments are simply skipped), but their methodology is applicable.

2.2 Incomplete Sessions

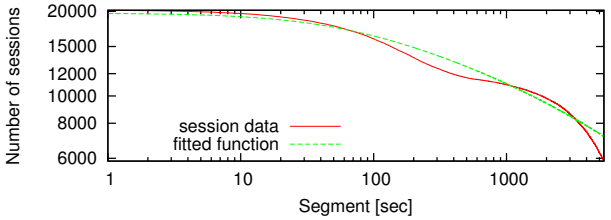
The authors of [11] conducted measurements on two corporate video servers, and published a statistical analysis of client behaviour, regarding video popularity distribution, the evolution of these popularities, the temporal characteristics of the requests (e.g., inter-arrival time, diurnal request intensity change), and the session lengths. They've found that the clients terminate most of the video sessions very early, and watch the entire video only in very few cases. They call the distribution of the session lengths *prefix distribution*, and give a characterization using three distributions. There is a given ratio of completed sessions, the distribution of the sessions shorter than 5 minutes is exponential, and in the range in between those the prefix has a uniform distribution. According to their studies, the prefix distribution highly depends on the length of the video.

Based on their measurement results and statistical models, they have created a video server workload generator tool, called *MediSyn*, which generates synthetic video request patterns that obey all statistical properties they measured. We used this tool to generate the prefix of the video requests for our simulations.

The browsing nature of client accesses was also observed in [15]. They used measurements taken on public media servers, and reached similar conclusions to [11]: most of the sessions terminate very early, and only a few of them are completed. They also examined the connection between the popularity of the videos and the prefix distribution, and they found that the mean prefix size does not depend on the video popularity, but its distribution does: less popular videos generally have a larger percentage of completed sessions.



(a)



(b)

Fig. 1 Prefix distribution used in our simulations: (1a) cumulative distribution function; (1b) popularity distribution of the segments, and a k-transformed Zipf distribution fitted on it

An explanation for the prefix phenomenon is that the clients are browsing through the content. They cannot decide if they want to watch a video based only on its title or description, so they start watching, but change their minds after the first few minutes. The late terminations might be attributed to boredom, lack of time or lack of patience ([11] examined servers with corporate training videos). In public VoD systems the reasons for incomplete sessions are similar, but pay-per-view schemes might increase this effect further. In such services the clients are allowed to watch the first ~ 10 minutes of the videos for free, where they have to decide whether they want to pay for the rest or not.

A further analysis on incomplete sessions was published in [16]. In that paper the k-transformed Zipf distribution was proposed to model the popularity distribution of video segments, which they call *internal popularity*. The k-transformation was initially developed by the authors of MediSyn in [11] to ease the fitting of the Zipf-Mandelbrot law [8] on the popularity curve of the videos. An online method is also presented in [16] to determine the parameters of the internal popularity distribution, which is then used to perform per-segment caching decisions.

The video request list that we generated for our simulations consists of requests for a single video with constant arrival rate and has the same prefix distribution as the default settings of MediSyn. We could not use MediSyn directly, as it only has a fixed set of built-in video popularity evolution functions, and cannot generate constant arrival rate.

As the fitted curve in Fig. 1b shows, the segment popularity distribution generated with MediSyn is not exactly a k-transformed Zipf distribution. This does not invalidate the analysis done in [16]; however, in that paper the authors admit that not every video follows strictly that distribution, and the distribution of MediSyn might be wrong as well. We decided to use the model of MediSyn, because it is well-defined, and the two are very close.

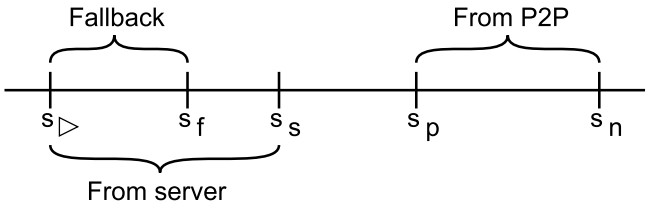


Fig. 2 The download windows used by the clients

3 Windowing BitTorrent

We examine a P2P-assisted VoD system that serves stored media files, where the P2P distribution network offloads a central video server. The P2P download of the clients is limited by a sliding window placed ahead of the playback position, as it was proposed in several papers already [13,2]. In those schemes the window was used to increase the timeliness of segment arrival; in addition, we also use it to avoid downloading segments that are not going to be played, because of the video being stopped by the user. For this reason, we also examined, if it is advantageous to place the P2P window at a fixed distance from the playback position, preventing it from advancing too fast, and downloading too much. In this section we formally analyze the efficiency of this kind of windowing.

Our analysis focuses on one video, for which the requests arrive according to a Poisson process with λ intensity. The video is composed of N equally sized segments, which are the atomic transfer units, i.e., a client can download a segment from just one other node at a time, and segment downloads cannot be interrupted. This is a simplification over the real BitTorrent protocol, where the chunks are the unit of bookkeeping, and it is possible to download almost any byte range of the content (the addressing is <chunk, offset, length>). Also, BitTorrent allows downloading the same part from multiple sources at the same time, which is used heavily in the endgame.

The segments are numbered from 1 to N in the playback order. The playback position is at segment $s_\triangleright$, as shown in Fig. 2. The P2P download window is in the $[s_p, s_n)$ range, the window size is $W = s_n - s_p$. Segments preceding s_s are downloaded from the server, if their transfer hasn't been started, because we have to

provide continuous video experience. This is called the *fallback window*. Naturally, $s_s \geq s_\triangleright$ and $s_p \geq s_\triangleright$ are required.

In the P2P window each segment can be either Full, if it is downloaded or is under download, or Empty. The number of Full and Empty segments are F and E respectively, and of course $W = F + E$. The playback of one segment is called a round or timeframe; in each round the playback position and the P2P window advance. The playback window advances 1 segment in each round, and the P2P window advances $a(t)$ segments, where time t is the playback position.

The window can also be viewed as a shift register: the first segment is removed (the *exit segment*), the rest is shifted with one position, and an Empty segment is shifted in at the Wth position. If the exit segment is Empty, it is called a *miss*. Such missing segments must be downloaded from the server once they enter the fallback window to provide continuous video experience.

The time unit is the playback time of one segment. The clients have uniform uplink capacity, they can upload at most U segments at a time, and it takes at least T timeframes to transfer one segment. The total download speed may also be limited to at most D parallel downloads, but in the theoretical study we had to assume $D = \inf$ for simplicity. In a real system the constraint is $D > T$, because the download capacity must be higher than the video bitrate.

According to [10], the generic model for the evolution of the number of leecher (x) and seeder (y) peers in a BitTorrent-based P2P system is

$$\begin{aligned} \mathrm{d}x/\mathrm{d}t &= \lambda - \theta x - \gamma x \\ \mathrm{d}y/\mathrm{d}t &= \gamma x - \mu y, \end{aligned} \tag{1}$$

where λ is the rate of arrival of new clients, θ is the rate of early departures (these leave the system), γ is the completion rate (these keep on uploading), and $1/\mu$ is the average seeding time. If we solve (1) for $\mathrm{d}x/\mathrm{d}t = \mathrm{d}y/\mathrm{d}t = 0$, we obtain

$$\overline{x} = \frac{\lambda}{\theta + \gamma} \qquad \overline{y} = \frac{\lambda}{\mu}. \tag{2}$$

Analyses in [9,10] focus on determining γ, because that is the key parameter that governs the efficiency, while the others are external parameters describing the behaviour of the users.

In the usual BitTorrent terminology the seeders have the entire video, but in our case the clients abort downloading the video as the playback is stopped, while continuing the uploading. Therefore, in our terminology the clients that are both downloading and uploading are "leechers", and the ones that are only uploading are "seeders", irrespectively of the amount they have.

Observe that y is independent of the prefix, as it only depends on the arrival and departure intensity, while being insensitive to the length of the video and the transfer speeds. The consequence of this is that the swarm might be better seeded, since there are fewer leechers due to the shorter sessions, but the number of seeders is the same; they are not able though to provide segments to all leechers.

4 Download Initiation Probability

In this section we show how the probability of finding a P2P uploader for a segment depends on the system parameters. This train of thought is an adaptation of the model shown in [9] to the prefixed case.

The incomplete sessions are described with a $\phi(i)$ function, which is defined as the number of accesses for the ith segment (as seen, e.g., in Fig. 1b) divided by the total number of sessions. It is not a probability measure, as it is not normalized to 1. If we assume that all clients start playing the video at the first segment, and no jumps are allowed, then $\phi(0) = 1$ and $\phi(i)$ is a monotonically decreasing function, but we don't limit this analysis to such constraints.

In reality the segments have finite length; thus, $\phi[t]$ should be a discrete-time function, but due to technical considerations we have to treat it as a continuous function in this section.

As mentioned earlier, each client can upload to U other clients simultaneously, and there are approx. $x+y$ clients in total who can upload. The leechers have ongoing download connections, which already occupy upload slots, and of course they want to start new downloads in each round; thus, they have a time dependent download demand $d(t)$. There are $\lambda \mathrm{d}t$ clients in an infinitesimal time interval around time t in the playback without prefix, and $\lambda\phi(t)\mathrm{d}t$ with prefix.

The number of leechers older than t' is

$$\lambda\psi(t') = \lambda \int_{t'}^{N} \phi(t)\mathrm{d}t, \tag{3}$$

their demand is

$$\lambda\Xi(t') = \lambda \int_{t'}^{N} d(t)\phi(t)\mathrm{d}t. \tag{4}$$

The number of leechers is $\lambda\psi(0)$, the total demand is $\lambda\Xi(0)$. Since $\phi(t)$ is a known function, $\psi(t)$ is also known, but $\Xi(t)$ contains the yet unknown evolution of the demand $d(t)$.

According to [9] the probability that a client can start a new download can be calculated as

$$p(t) = \min\left\{\frac{\text{relevant supply}}{\text{concurrent demand}}, 1\right\}, \tag{5}$$

where the relevant supply are given by the peers having segments the client wants, and the concurrent demand is generated by the peers who also want to download from those peers. Note that without the clamping high supply or low demand could cause $p > 1$.

Due to the windowing the clients download the segments of the video more-or-less in order; thus, a client at time $t' = s_\triangleright$ is only interested in the upload of the leechers that are older than itself. The number of such clients is $\lambda\psi(t')$. Due to the prefix, not all seeders have segments beyond t'. The number of seeders that do have such segments is $y\phi(t')$. Using these we get

$$\text{supply relevant for client in } t' = U(y\phi(t') + \lambda\psi(t')). \quad (6)$$

According to [9] the concurrent demand can be calculated indirectly from the demand served by clients younger than t'. The number of clients in a small $\mathrm{d}t$ time interval at $t < t'$ is $\lambda\phi(t)\mathrm{d}t$, their demand is $d(t)\lambda\phi(t)\mathrm{d}t$ (note that in [9] they have $\lambda D\mathrm{d}t$, because in their simplified model the clients can initiate D new connections in every round, and transfers take one timeframe). This demand is distributed uniformly to the $y\phi(t) + \lambda\psi(t)$ potential uploaders a-priori. One potential uploader receives

$$\frac{d(t)\lambda\phi(t)\mathrm{d}t}{y\phi(t) + \lambda\psi(t)} \quad (7)$$

demand from these clients, and there are $\lambda(\psi(t) - \psi(t'))$ of them in $[t, t']$. The total demand that can be ignored is thus

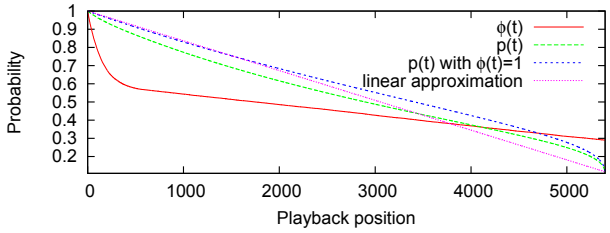
$$\delta(t') = \int_0^{t'} \frac{\lambda^2 d(t)\phi(t)(\psi(t) - \psi(t'))}{y\phi(t) + \lambda\psi(t)}\mathrm{d}t. \quad (8)$$

From (6) and (8) the download initiation probability is

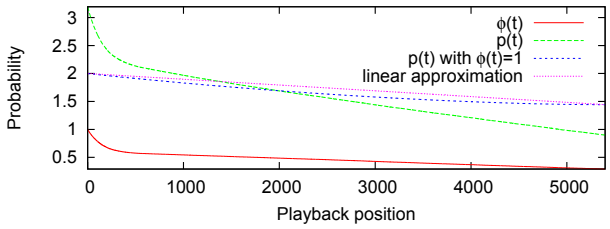
$$p(t') = \frac{U(y\phi(t') + \lambda\psi(t'))}{\lambda\Xi(0) - \delta(t')} \quad (9)$$

The nominator only depends on the playback position, and the known constants U, y, λ, and $\phi(t)$. The denominator, however, depends on Ξ, which is the integral of $d(t)$, for which we will develop difference equations in the next section. An interesting property of (9) is that in the equilibrium, where $y = \lambda/\mu$ according to (2), we can completely eliminate λ from the formula.

To analyse the dependency of $p(t')$ on $\phi(t)$, we computed (9) numerically for the segment popularity of MediSyn and the unprefixed case with $U = 1$ and $d(t) = D = 1$. The (unclamped) results are shown in Fig. 3 for some μ values. At short seed durations the two are very close, but as $1/\mu$ gets more realistic they begin to differ. In Fig. 3b this doesn't seem important, as $p > 1$ in both cases, but $D > 1$ moves them into the $p < 1$ range. At even longer seed durations the shape of $p(t')$ becomes identical to $\phi(t)$.



(a)



(b)

Fig. 3 Numeric evaluation of (9) for (3a) $1/\mu = 1$s; (3b) $1/\mu = 5400\text{s} = N$ mean seed duration

If $\phi(t) = 1$ and $d(t) = D$ constant, (9) can be solved analytically. The formula is too complicated to be useful, but it is quite simple at the beginning

$$\phi = 1 \to p(t' = 0) = \frac{U}{D}\frac{1 + \mu N}{\mu N} \quad (10)$$

and the end

$$\phi = 1 \to p(t' = N) = \frac{U}{D}\frac{1}{\ln(1 + \mu N)} \quad (11)$$

of the video. The linear approximation of $p(t')$ using these endpoints is also shown in Fig. 3. In this case $p(t)$ is constant for realistically long seed durations (several times longer than the length of the video); thus, the linear approximation gives the exact solution. Although (10) and (11) are different, if μ is small we can use the formula $\ln(1 + x) \sim x$, and they both become

$$p \sim U/(D\mu N). \quad (12)$$

(9) cannot be further simplified analytically to find its dependency on $d(t)$, even in the simpler $\phi(t) = 1$ case. In the next section we will use constant $\phi(t)$ in the numeric analysis to avoid introducing errors.

5 Windowing Algorithms

In this section we analyse how the windowing affects the system performance. Unlike the previous section here we use discrete time to describe the system.

We are mainly interested in the miss probability, and the number of Empty segments $E[t]$, the demand $d[t]$, the position of the P2P window $\pi[t]$.

The demand of a client is the total number of upload slots it intends to allocate for itself in a round. It

already allocated slots for the ongoing downloads, and in each round it wants to start downloading all remaining Empty segments in the P2P window; thus the total demand is

$$d[t] = E[t] + \text{ongoing}[t]. \tag{13}$$

The formula for the change of the demand is

$$\begin{aligned}\Delta d[t]=&\text{newly started} - \text{ended since last round} + \\ &+\Delta E[t].\end{aligned} \tag{14}$$

As substituting this difference equation into (9) is not solvable, we will analyse it numerically by assuming constant p.

We examine two P2P window placement schemes. In the *Progressive* scheme the P2P window steps over every Full segment in the beginning of the window; thus, $a[t] > 1$, and $a[t] = 0$ are possible, but $s_p \geq s_\triangleright$ is required. The *Streaminglike* scheme places the P2P window at a constant distance from the playback position; thus, $a[t] = 1$ in every round. This one is expected to download fewer unneeded segments than the Progressive scheme.

We examine two segment selection schemes within the P2P window. The *Random* scheme randomly picks segments for download in the P2P window. According to [9] the random segment selection scheme is a good approximation of the rarest-first scheme. The *Linear* scheme tries to download segments in-order, starting from the first Empty one, and the process stops when no uploader was found for a segment, or the end of the P2P window is reached. Together with the two window placement schemes they allow 4 combinations, which we analyse in the following sections.

The position of the P2P window $\pi[t]$ is updated in every round. In the Streaminglike case $a = 1$, and the window maintains a constant distance from the playback position. In the Progressive case this only happens, if $s_p = s_\triangleright$, and $a[t] \leq 1$, otherwise the window position is

$$\pi[t] = \sum_{i=1}^{t} a[i]. \tag{15}$$

5.1 Progressive Mode, Random Segment Selection

In this scheme the P2P window advances $a[t]$ segments in the tth round, where $a[t]$ equals the number of Full segments at the beginning of the window. Of course, the P2P window can't stop for long, even if $a \sim 0$, because it must be kept ahead of the playback position, potentially resulting in missed segments. We will denote the number of rounds the kth segment of the window has spent in the P2P window with $g[k,t]$ ($k = \{1 \dots W\}$), which depends on the advance of the recent rounds.

In each round $a[t]$ Empty segments are shifted in at the end of the P2P window, and pE Empty segments are converted into Full on average. The download speed is thus pE, which is faster than the playback speed, if $p > 1/E$. It is compared to the linear case in Fig. 6.

The exact probability of a segment miss is rather complicated, we will approximate it with $(1-p[t])^{g[1]}$. Using these we can construct the evolution of the number of Empty segments as

$$\Delta E[t] = a[t] - p[t]E[t] - (1-p[t])^{g[1,t]}, \tag{16}$$

and the evolution of the demand of a client as

$$\begin{aligned}\Delta d[t]=&\text{starts}[t] - \text{stops}[t] + \Delta E[t] \\ =&p[t]E[t] - p[t-T]E[t-T] + \\ &+(a[t] - p[t]E[t] - (1-p[t])^{g[1,t]}) \\ =&a[t] - p[t-T]E[t-T] - (1-p[t])^{g[1,t]}.\end{aligned} \tag{17}$$

Generic formulas for $g[k,t]$ and $a[t]$ are hard to find, especially if we consider that $p[t]$ depends on the playback position. Instead, we will develop their values for the steady-state, where E is constant or changes very slowly. In this case pE is also constant, the P2P window advances $a(p)$ segments in each round, resulting in constant g.

If $a(p)$ is constant, segments spend approximately $\lceil W/a(p) \rceil$ rounds in the P2P window; therefore, the segment in the kth position is in the window since

$$g[k] = \left\lceil \frac{W-k+1}{a(p)} \right\rceil \tag{18}$$

rounds, in other words it had $g(k)$ opportunities to become Full. We get the number of Empty segments

$$\overline{E} = \frac{a(p) - (1-p)^{g[1]}}{p}, \tag{19}$$

by solving (16) for $\Delta E = 0$.

The number of Full segments at the beginning of the window is $f(p)$. The probabilities for the possible values are

$$\begin{aligned}\mathbf{P}(f(p)=0)=&(1-p)^{g[1]} \\ \mathbf{P}(f(p)=W)=&\prod_{j=1}^{W}(1-(1-p)^{g[j]}) \\ \mathbf{P}(f(p)=i-1)=&(1-p)^{g[i]}\prod_{j=1}^{i-1}(1-(1-p)^{g[j]}),\end{aligned} \tag{20}$$

where segments $[1, i-1]$ are Full and segment i is the first Empty. This is a probability measure, as its sum is 1.

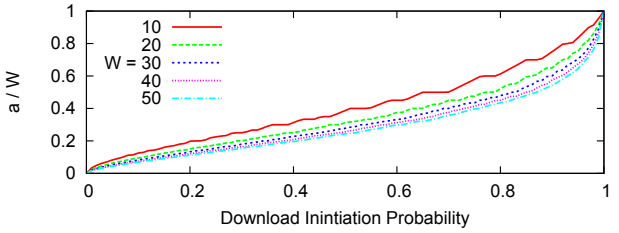


Fig. 4 Advance speed of the p2p window in the equilibrium of the progressive random scheme as a function of the window size W and the download initiation probability p

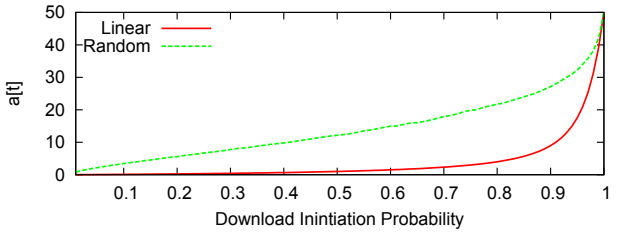


Fig. 5 Advance speed of the Progressive P2P window with different operation methods for $W = 50$

The expected advance is thus

$$a(p)=\mathbf{E}\{f(p)\} = \\ =\sum_{i=2}^{W}(i-1)(1-p)^{g[i]}\prod_{j=1}^{i-1}(1-(1-p)^{g[j]}) + \\ +W\prod_{j=1}^{W}(1-(1-p)^{g[j]}), \qquad (21)$$

if there is no need to correct it to $a = 1$ to keep the P2P window ahead of the playback position. Solving this is quite complicated, as $a(p)$ is embedded in $g[j]$. We calculated $a(p)$ from (21) numerically for some window sizes. The result of this is shown in Fig. 4; the jumps in the curves are caused by the rounding in $g[j]$. A comparison of this advance speed to the Progressive Linear case is shown in Fig. 5 for $W = 50$.

5.2 Streaminglike Mode, Random Segment Selection

The difference between this and the Progressive mode is that $a[t] = 1$ constant. The effect of this change is that the P2P window can be full of Full segments, because it cannot advance faster than the playback. The evolution of the number of Empty segments in this case is

$$\Delta E = 1 - pE - (1-p)^W, \qquad (22)$$

as $g[k] = W - k + 1$ independently of p or the playback position. The number of Empty segments in the P2P window in steady-state is shown in Fig. 7 for constant p compared to the Linear case. Though the download is faster than the playback even at small p according to Fig. 6, the decreasing number of Empty segments

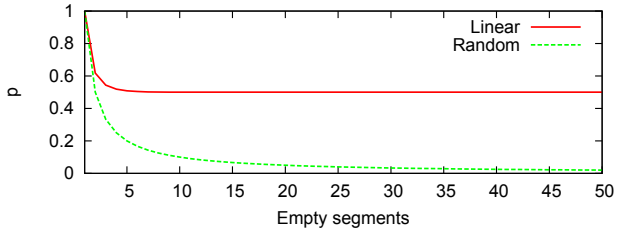


Fig. 6 Download initiation probability p required for faster download than the playback speed

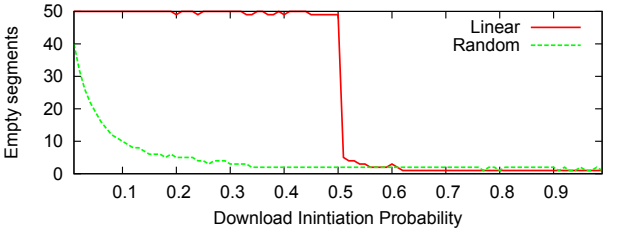


Fig. 7 Number of Empty segments in the P2P window in the Streaminglike case in steady-state

limits the download speed, and the window cannot be fully converted below a certain p.

5.3 Streaminglike Mode, Linear Segment Selection

We start the Linear segment selection with the Streaminglike case, because this is the more general one. At the beginning of a round there might be some full segments at the beginning of the window, and the remaining $E = W - F$ are Empty. The probability that $i - 1$ segments are converted in the round is

$$P_i = \mathbf{P}(i\text{th is the first Empty}) = p^{i-1}(1-p). \qquad (23)$$

Converting all of them is a special case, it has p^E probability. This is a probability measure, as its sum is 1. The expected number of newly started downloads (Leap) is

$$L=\mathbf{E}\{p_i\} - 1 = \sum_{i=1}^{E} ip^{i-1}(1-p) + (E+1)p^E - 1 \\ =\frac{1-p^E}{1-p} - Ep^{E-1} + (E+1)p^E - 1 \sim p\frac{1-p^E}{1-p}, \qquad (24)$$

where the derivative of the formula for the sum of Geometric series was used. The last step contains a simplification that introduces negligible error, while greatly simplifying the formula. If $L < 1$, then the miss probability is $(1 - p)$, but miss-free operation is possible, if $L \geq 1$. (24) yields a degree $E + 1$ polynomial, if we want to compute the required p for faster download than playback ($L = 1$); thus, it has no universal solution formula. Numerical results are shown in Fig. 6 compared to the random case.

The number of Empty segments is decreased with L in each round, one new Empty segment is shifted in

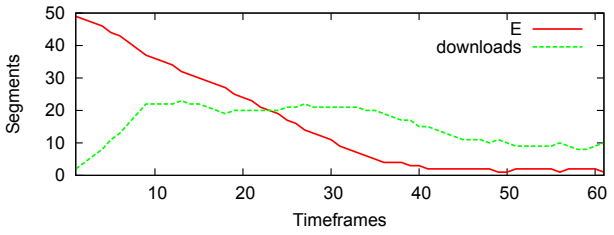


Fig. 8 The evolution of the number of Empty segments in the P2P window and the number of ongoing downloads in the Streaminglike case at $p = 0.7$

as the window moves, and the exit segment might be Empty as well; thus, the evolution of E is

$$\Delta E[t] = 1 - L[t] - \text{exit}[t]. \tag{25}$$

The state of the exit segment is hard to handle analytically. If $E < W$, then it is surely Full, but if $E = W$, then two cases are possible: if $L \geq 1$, then it was converted to F, otherwise it is Empty with probability $1-p$. Additionally, there is a $0 \leq E \leq W$ constraint. We evaluated the evolution of the number of Empty segments numerically for constant p. Fig. 7 shows the number of Empty segments in the P2P window in the steady-state. Unlike the Random scheme, there is a phase change at $p = 0.5$.

The evolution of the demand of a client is

$$\begin{aligned}\Delta d[t] &= \text{starts}[t] - \text{stops}[t] + \Delta E[t] \\ &= L[t] - L[t-T] + \Delta E[t] \\ &= 1 - \text{exit}[t] - L[t-T].\end{aligned} \tag{26}$$

The constraints for this are $d \geq 0$ and the ongoing downloads should be $d - E < D$. As the numerical results for $E[t]$ showed, the steady state is either $E = W$ or $E = 1$ depending on p; the demands in these cases are $W \leq d \leq W + T$ and $d = T$ respectively, as at most T ongoing downloads are possible in both cases. At high p the Linear and the Random scheme is indistinguishable: in both cases the P2P window is full, and it doesn't matter how the one new Empty segment is selected. The download process is shown in Fig. 8 for the "high p" phase: E decreases monotonously, while the number of downloads initially increases, then decreases to $\sim T$; d is the sum of these two.

5.4 Progressive mode, Linear segment selection

This case is the same as the Streaminglike case, but here $E = W$ in each round, as $a[t] = L[t]$, which is compared to the Random case in Fig. 5. The implication on $d[t]$ is that only $W \leq d \leq W + T$ is possible, because the "high p" phase never occurs.

6 Further Effects

In this section we describe two effects that arise in real systems, but can't be covered by the models presented in the previous sections.

6.1 Prebuffering

Before the playback can begin, the first few segments have to be downloaded, this is called prebuffering. The size of the buffer should be $B \geq T$, because once the buffer is filled, the playback starts, and the first segment after the buffer has to be downloaded before the playback arrives. As the download speed is $D \geq T$, the question is whether B should be smaller or bigger than D. To minimize the number of extra downloaded segments in case the session is aborted early, $B = T$ is a good choice, but it may happen that the downlink and the storage at the clients are cheap, in which case the extra downloads increase the number of potential uploaders.

If $D > B$, and in the prebuffering phase the client utilizes its entire downlink capacity, the P2P download window is not empty when the playback starts (unless $D = kB$, where $k \in \{2, 3, 4 \ldots\}$). Depending on p this means that in the Streaminglike case either the P2P window gets filled faster, or there is a little time before the window settles at $E = W$. In the Progressive case the P2P window starts farther from the beginning of the video than its initial placement, which might lead to more extra download if the session is aborted.

6.2 Overloading

The advance of the P2P window is $a(p[t])$. If the playback is interrupted, the download of the Full segments ahead of it occupied the network unnecessarily. The number of such segments is approx.

$$s[t] = (1 - p_{\text{miss}}(t))(s_p - s_\triangleright) + (W - E[t]), \tag{27}$$

if the playback stopped at time t. The expected value of $s[t]$ is

$$S = \sum_{t=1}^{N-1} S[t](\phi[t] - \phi[t+1]). \tag{28}$$

The segment popularity function $\phi[t]$ given in [11, 15] are based on playback control events, but due to prebuffering and overloading the popularity curve of played and downloaded segments differ significantly. Instead of building an analytic model for this phenomenon we decided to conduct a simulation study instead.

7 Simulation Setup

Our analysis in the previous sections used a simplified model of the VoD clients, and focused only on the P2P downloads. As we have seen, the windowing and the prebuffering make the system nonlinear, and hard to analyze analytically; thus, we used stream-level simulations to further investigate the system in realistic scenarios. To do this, first we need to define the operation of the client in detail.

7.1 System Details

As already mentioned, the P2P download window can miss some segments, either because of the download initiation probability p being low or in the Progressive case the window moving too fast. These segments have to be downloaded from a different content source to provide continuous video experience to the user. This content source might be a central server storing all videos or a distributed network of video caches.

A general model of the fallback mechanism has two stages, as shown in Fig. 2. Empty segments in the server window (range $(s_\triangleright, s_s]$) can be downloaded from the server if there is sufficient downlink capacity. The segments in the fallback window (range $(s_\triangleright, s_f]$) are too close to the playback position; if an Empty segment enters this range, its download must be initiated without delay. The size of the fallback window must be $s_f - s_\triangleright \geq T + 1$ segments to guarantee timely arrival, and not much larger to minimize overload, but $s_s - s_\triangleright$ is arbitrary. The server window might even be absent, we didn't implement it for our simulations.

We implemented the downlink limitation of the clients as a hard limit on maximum number of concurrent downloads D. The two most important consequence of the limited downlink are that the download demand is 0, if the limit is reached, and fallback is temporarily not possible, which may lead to buffer underrun.

There are two possible schemes to ensure fallback with limited download capacity. One is to avoid such situations by implementing a complex resource management algorithm that detects the missed segments and reserves downlink slots in advance for the fallback. The other is to abort P2P downloads to free downlink capacity for the fallback. We chose the latter scheme, because of its simplicity, but it is important to note that it breaks the assumption that segment transfers are atomic.

In our model the server can upload any segment in T timeframes. The P2P clients are usually very unreliable, which we model with a probabilistic upload time: uniform distribution in range $[T, \zeta T]$, where $\zeta \geq 1$; all presented results have $\zeta = 2$. Due to the unreliability of the P2P uploads we decided to prebuffer exclusively from the server to minimize the startup delay.

7.2 Simulation Parameters

We used a synthetic list of requests for the video arriving with $\lambda = 1/172.8$ requests per second (this is a very popular video). These requests were distributed among a large client pool with $U = 1$, $T = 8$. The mean of the seed duration was set to $1/\mu = 5\text{days} = 432000$ seconds, because Set-top-boxes usually have large storage, and they can seed the video until it is overwritten by newly watched content. The number of seeders is thus $y = \lambda/\mu = 2500$. In the simulations all clients seed the video; thus, $\theta = 0$ in (2).

The video itself is assumed to be 90 minutes long with 10 second segments; thus, $N = 540$. In a real system the segments (the atomic transfer units) would be < 1 second long. The number of leecher peers, x, depends on λ and the length of the video: with incomplete sessions $x \sim 15$, with complete sessions $x \sim 32$.

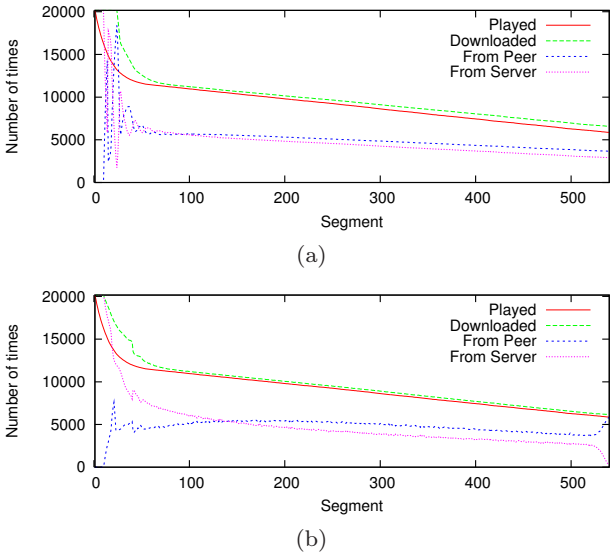
8 Simulation Results

In Fig. 9 the number of times the segments were played is compared to the number of times they were downloaded from the two sources. In both cases the downlink limitation D is tuned to show the turning point, where the P2P and the server downloads are almost equal ($D = 14$ for Linear, $D = 11$ for Random). The P2P download peak is clearly visible in Fig. 9b, but it is not that apparent that the lines are much more smooth in Fig. 9a than Fig. 9b. In the Linear case $D \pm 1$ zeroes out either the server or the P2P downloads except for the beginning; the shift is gradual in case of Random segment selection.

The Streaminglike scheme, shown in Fig. 10 is almost identical to the Progressive. At the same settings there are slightly fewer P2P downloads, and, unlike the Progressive case, the steepness of the curve of the total download amount cannot differ from the watched popularity. Both Fig. 9 and Fig. 10 show that the greatest difference between the played and the downloaded amount is in the beginning of the video, but not all of that can be attributed to the prebuffering.

9 Conclusions

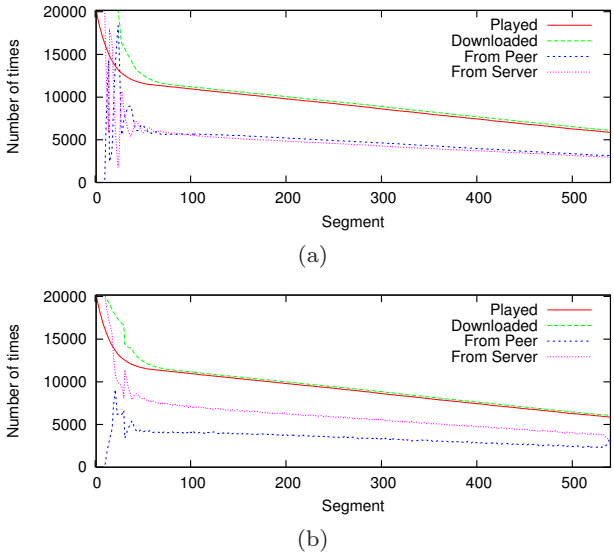
In this paper we analysed the effect of incomplete sessions on on-demand stored video streaming assisted by



(a)

(b)

Fig. 9 Number of segments watched and downloaded in the Progressive scheme; (9a) Linear; (9b) Random



(a)

(b)

Fig. 10 Number of segments watched and downloaded in the Streaminglike scheme; (10a) Linear; (10b) Random

peer-to-peer distribution. We found that the use of a modified version of the BitTorrent protocol with a download window added to increase the timeliness of the arrival of video segments provides good performance.

In the theoretical study we developed models for several possible operation methods of the system, and found that in a poorly seeded P2P swarm the there is a big difference in performance for the different methods, which disappears if there are enough seeders and the downlink speed is adequate.

Our simulation study confirmed these findings, and provided further insight. The efficiency of the P2P distribution heavily depends on the downlink capacity, and a download window at a fixed distance from the playback position is as efficient as a freely moving window.

References

1. Cha, M., Rodriguez, P., Moon, S., Crowcroft, J.: On next-generation telco-managed P2P TV architectures. In: Proc. of IPTPS'08. Tampa Bay, FL, USA (2008)
2. Choe, Y.R., Schuff, D.L., Dyaberi, J.M., Pai, V.S.: Improving vod server efficiency with bittorrent. In: Proc. of MULTIMEDIA '07, pp. 117–126. ACM, Augsburg, Germany (2007)
3. Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. of 1st Workshop on Economics of Peer-to-Peer Systems (2003)
4. Dana, C., Li, D., Harrison, D., Chuah, C.N.: BASS: BitTorrent assisted streaming system for video-on-demand. In: Proc. of MMSP'05, pp. 1–4. Shanghai, PRC (2005)
5. Do, T., Hua, K., Tantaoui, M.: P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proc. of ICC'04, pp. 1467–1472. Paris, France (2004)
6. Guo, Y., Suh, K., Kurose, J., Towsley, D.: P2cast: peer-to-peer patching scheme for vod service. In: Proc, of WWW'03, pp. 301–309. Budapest, Hungary (2003)
7. Liu, J., Xu, J.: Proxy caching for media streaming over the internet. IEEE Communications Magazine **42**(8), 88–94 (2004)
8. Mandelbrot, B.: Information Theory and Psycholinguistics. Penguin Books (1968)
9. Parvez, N., Williamson, C., Mahanti, A., Carlsson, N.: Analysis of BitTorrent-like protocols for on-demand stored media streaming. SIGMETRICS Perform. Eval. Rev. **36**(1), 301–312 (2008)
10. Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. In: Proc. of SIGCOMM '04, pp. 367–378. Portland, Oregon, USA (2004)
11. Tang, W., Fu, Y., Cherkasova, L., Vahdat, A.: Modeling and generating realistic streaming media server workloads. Computer Networks **51**(1), 336–356 (2007)
12. Vapa, M., Auvinen, A., Ivanchenko, Y., Kotilainen, N., Vuori, J.: Optimal resource discovery paths of Gnutella2. In: Proc. of AINA'08, vol. 0, pp. 546–553. Los Alamitos, CA, USA (2008)
13. Vlavianos, A., Iliofotou, M., Faloutsos, M.: BiToS: Enhancing BitTorrent for supporting streaming applications. In: Proc. of INFOCOM'06, pp. 1–6. Barcelona, Spain (2006)
14. Ying, L., Srikant, R., Shakkottai, S.: The asymptotic behavior of minimum buffer size requirements in large p2p streaming networks. In: Proc. of ITA2010, pp. 1–6. San Diego CA, USA (2010)
15. Yu, H., Zheng, D., Zhao, B.Y., Zheng, W.: Understanding user behavior in large-scale video-on-demand systems. In: Proc. of EuroSys '06, pp. 333–344. ACM, Leuven, Belgium (2006)
16. Yu, J., Chou, C., Yang, Z., Du, X., Wang, T.: A dynamic caching algorithm based on internal popularity distribution of streaming media. Multimedia Systems **12**(2), 135–149 (2006)
17. Zhao, B., Lui, J., Chiu, D.M.: Exploring the optimal chunk selection policy for data-driven p2p streaming systems. In: Proc. of IEEE P2P'09, pp. 271–280. Seattle, WA, USA (2009)