Acta Universitatis Sapientiae
Electrical and Mechanical Engineering, 7 (2015) 36-61

Assessing the Efficiency of a P2P Video Client

Miklós MÁTÉ, Rolland VIDA

Department of Telecommunications and Media Informatics,
Budapest University of Technology and Economics, Budapest,
e-mail: {mate; vida}@tmit.bme.hu

Manuscript received February 11, 2016; revised May 5, 2016

Abstract: Video on Demand systems are increasingly popular sources of entertainment, rapidly replacing conventional television. To meet the high bandwidth and fault tolerance requirements, most VoD content providers are utilizing distributed data delivery technologies, like Peer-to-Peer schemes. The client programs of such systems usually restrict the P2P scheme into a sliding window to ensure timely arrival of the video segments. In this paper we analyze the performance of a generic P2P VoD client. We develop analytical connections between the state descriptor variables, and provide conditions for the P2P scheme to be self-sufficient. We also examine the effect of limited downlink capacity.

Keywords: Peer-to-Peer, Video-on-Demand, Steady-state, Self-sustainability.

1. Introduction

The traffic transferred through the Internet is growing exponentially, mostly due to the increasing popularity of video content, and the demand for higher video quality increases the bandwidth-demand of the individual streams as well. Content providers have always used distributed server architectures to decrease network load and combat the latency issues by moving the content close to the consumers, but the increasing complexity of creating and maintaining such systems motivates the search for entirely new solutions. A scheme that offers great performance, error-resilience, and requires low maintenance is to utilize the clients as content sources for other clients. The downside of such Peer-to-

Peer (P2P) schemes is that the access networks are typically not optimized for uplink traffic.

The performance of Video-on-Demand (VoD) systems that use P2P technology is heavily influenced by the efficiency of the clients. Therefore, it is important to analyze the performance impact of the details of the P2P technology used in the clients. P2P VoD clients have to provide timely arrival of the video segments, or the playback cannot proceed smoothly; practically, this means that the general P2P schemes must be modified for use in a VoD system. Even though the performance of the general P2P scheme is known, these modifications can alter its properties to great extent.

This model of the P2P clients is based on the BitTorrent protocol, but is not specific to it. The clients use a sliding window of W segments, called P2P window, which is placed somewhere ahead of the playback position, and the P2P segment retrieval scheme is restricted to operate only within the P2P window. We consider two window placement schemes, and two segment selection schemes within the P2P window.

The most important contributions of this paper are the analytical results for the connections between the state descriptor parameters, and the criteria for self-sustained operation of the P2P scheme.

The rest of the paper is organized as follows. In the next section we give an overview of the related work in this field. In Section 3 we describe the model of the P2P client in detail. In Section 4 we construct and analyze the state-space description of the P2P download window. The sections after that each deal with a specific property of the system: the number of Full segments in Section 5, the number of new segment downloads in Section 6, the number of Empty segments in Section 7, and the advance speed of the window in Section 8. Section 9 gives formulas for the probability of missing a segment, and criteria for avoiding that entirely, if possible. Section 10 presents our findings about the performance of the client when the downlink capacity is limited. Finally, Section 11 draws the conclusions.

2. Related work

Peer-to-peer networks gained high popularity in the last decade. They can provide decentralized content indexing, data distribution, or both of them. Perhaps the best known P2P protocol is BitTorrent [4]. Its popularity can be attributed to its relatively simple design, efficient and flexible operation, and its open specification. It is also extremely resilient against network and node failures, which certainly helped its adoption as a distribution method of large files. It divides the content into small pieces, called chunks, which are exchanged among the clients interested in the content: each participant is a

downloader and an uploader at the same time. Those who have all pieces, and therefore are not downloading anymore, are called *seeders*, while the others are the *leechers*.

The incentive to upload is raised through the peer selection scheme of BitTorrent. It uses the tit-for-tat scheme, which is the best known solution to the repeated prisoner's dilemma [1]. In this scheme the clients prefer uploading to other clients who were willing to upload recently, punish denial with denial (this is called the choking mechanism), but are quick to forgive (optimistic unchoke). It is possible to free-ride in BitTorrent, and to build false reputation [6], but there are also some proposed optimizations to avoid that [14]. To maximize the throughput by eliminating bottlenecks, BitTorrent clients try to download the rarest chunks first. This scheme efficiently equalizes the availability of the chunks within the content.

BitTorrent has been thoroughly analyzed in the research community, several measurement studies have been conducted [9], analytical models have been constructed to explain its scalability [10], and these models were then validated by additional measurements [5].

However, despite the numerous advantages of BitTorrent, it is not directly suitable for video delivery, because the rarest-first chunk selection policy prevents the playback until a significant portion of the video is downloaded. To alleviate this issue the chunk selection scheme of BitTorrent must be modified to provide more-or-less in-order download policy. The usual approach is to introduce a download window, and only allow BitTorrent downloads within the window [11]. Some proposals let the client select chunks from outside the window with a given probability [12], and others use an exponentially weighted priority instead of a window [11], which may increase the efficiency of data sharing, but they also increase the chance of a buffer underrun. For this simulation study a closed download window seemed optimal, because it provides the strongest guarantee for timely arrival, and it is easier to examine analytically.

Numerous papers have analyzed the efficiency of P2P video distribution systems analytically. There are two research tracks that are important for us. In [8] 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. In [3] there is a detailed analysis on the remaining server load and the self-sustainability of the P2P system; however, that analysis assumes strictly linear segment retrieval.

There is analysis on live video streaming systems as well, where determining the optimal buffer size [13] and the priorities for downloading each segment in the buffer [15] 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.

3. Model of the P2P client

3.1. The P2P window

In this paper the only part of the client application that will be in focus is the P2P download window. The size of the window is always W segments, and the segments in the window are numbered $1 \ldots W$, where segment 1 is the closest to the playback position. In this model the P2P scheme only works inside the P2P window.

In this paper we only consider the case, where the download window is somewhere in the middle of the video. In the beginning the startup process is hard to model analytically, and it's quite different from the rest of the download process. Similarly, at the end of the video the P2P window stops, and the remaining segments are downloaded with an endgame protocol, which can be different from the main downloading mechanism.

When a client attempts to start downloading a segment of the video, it succeeds with probability p, which depends on the global state of the P2P swarm. Failing to start a download can happen for various reasons. It can happen that all the clients holding that segment are occupied, or maybe none of the other clients hold that segment yet. In the latter case, the clients revert to downloading the segment from the server, once it leaves the P2P window, and enters the fallback window, but that mechanism is outside of the scope of this study.

We assume that p is constant, which means that the P2P swarm is in equilibrium. This happens for example, when the new clients enter the system with constant intensity λ arrivals per second, and the seeders leave the system with constant intensity μ departures per second. We start with this known p, and derive formulas for the state variables of the system.

In this model the P2P window is thought of as a stochastic process, representing the statistical ensemble of several individual video downloads. This means that the variables that describe the system are random variables, and their distribution evolves with time throughout the download process. In this study the state of the individual clients will be irrelevant, and only their average behavior will be analyzed.

The downlink capacity of the VoD clients may not necessarily be infinite. In this model the downlink capacity of the clients is at most D ongoing downloads

at the same time. Throughout most of this paper $D=\infty$ will be assumed, except where explicitly stated otherwise.

3.2. State variables

Within the P2P window the segments can be in two different states. The *Full* segments are the ones, which are being downloaded, or already downloaded. The rest of the segments are *Empty*, and the goal of the download process is to convert as many Empty segments into Full as possible, before they leave the P2P window. The number of Full segments in the window is *F*, the number of Empty segments is *E*.

Since the P2P window must remain ahead of the playback position, it might happen that an Empty segment has to be shifted out. That event is a *Miss*, and the probability of missing a segment is *M*. The main goal of this study is to determine the *p* required for Miss-free operation, if it can be achieved. *Fig. 1* shows an example of the P2P window with a missed segment.

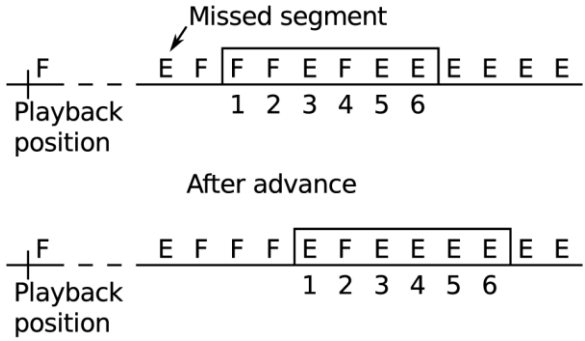

Figure 1: Illustration of the P2P window with *W*=6, and an example of Progressive window advancement

The download process works in rounds. In each round the algorithm scans the P2P window, and tries to start as many new downloads as it can. The number of newly started downloads in a round is *S*.

The number of Full segments at the beginning of the P2P window is *a*, and the advance speed of the P2P window is *A*. The connection between these two will be explained shortly.

These state variables are all non-negative integer valued random numbers. In most cases we will only consider their expected values, because their distributions are difficult to determine. The notation $\overline{X}$ will be used for the expected value of *X*, calculated as the ensemble average. Obviously, these expected values are not necessarily integer.

Time is measured in *timeframes*, one timeframe is the time it takes to play one segment of the video. Downloading a segment from a peer takes T timeframes, which is a constant in this paper.

The client initiates new downloads in rounds: it periodically runs the segment selection, and the window positioning algorithms for each timeframe.

If $T > 1$, there are usually some ongoing segment downloads when the round starts. The number of ongoing downloads is $O \leq D$.

```
function LINEAR SELECTION
    for i = 1...W do
        segment = segments[window_begin + i]
        if EMPTY(segment) then
            if CAN START DOWNLOADING(segment) then
                segment.state = "Full"
            else
                return
            end if
        end if
    end for
end function
function RANDOM SELECTION
    for i =SHUFFLE(1...W) do
        segment = segments[window_begin + i]
        if EMPTY(segment) then
            if CAN START DOWNLOADING(segment) then
                segment.state = "Full"
            else
                no action
            end if
        end if
    end for
end function
```

Figure 2: Pseudocode of the segment selection schemes

The pseudocode of the periodic window management of the P2P client is shown in *Fig. 3*. There are two points in the code marked as **A**, and **B**. The state variables that are absolute quantities will be indexed with the point, where the quantity is measured; thus, E_A is the number of Empty segments after the window was advanced, and E_B is the number of Empty segments after new downloads have been started. These indexed variables are *E*, *F*, and *O*, and *a*, while *S*, *M*, and *A* describe changes that happen during transitions between point A and point B.

```
function MANAGE P2P WINDOW
    if Progressive then
        while segments[window_begin].state is "Full" do
            window_begin++                                  ▷ Natural advance
        end while
    end if
    if window_begin < playback_position + guard_distance then
        window_begin = playback_position + guard_distance   ▷ Coerced advance
    end if
    Point A: after window advance
    Start new downloads
    Point B: before window advance
end function
```

Figure 3: Pseudocode of the P2P window management, which the client calls periodically for each timeframe

3.3. Segment selection schemes

In this paper we examine two segment selection schemes within the P2P window.

Definition 1 Linear selection: the segments are selected strictly in-order. The round ends, if all of the Empty segments in the window were started, or one segment download failed to start.

It follows from the above definition that $F = a$ with Linear selection.

Definition 2 Random selection: the Empty segments of the P2P window are tried in random order. The round ends, if starting each of them was attempted once.

The pseudocodes for the two segment selection schemes are shown in *Fig. 2*.

It is obvious that Random selection is more efficient than Linear, because it scans all Empty segments in each round, but the latter is important for performance comparison, and it has a few interesting properties as well.

3.4. Window positioning schemes

Naturally, the P2P window has to be ahead of the playback position at all times. It's also obvious that there should be a gap of at least *T* segments between them, because downloading a segment takes *T* timeframes. These constraints don't fully determine the positioning of the P2P window, however. We will examine two different schemes for positioning the P2P window: the Streaminglike scheme, and the Progressive scheme.

Definition 3 Streaminglike: maintain a constant distance; thus, $A = 1$.

Definition 4 Progressive: advance, until the first segment in the window is Empty; thus,

$$A = \begin{cases} 1 & \text{if } a_B < 1 \\ a_B & \text{if } a_B \geq 1 \end{cases}$$

The reason for the two cases with the Progressive scheme is that the playback time t_m, where the state of the window is examined, is far from the beginning of the video, and mostly constant a_B can be assumed. This suggests that if a_B is usually less than one, the playback position has caught up with the P2P window regardless of the initial distance between them. In this case $A=1$ has to be maintained in order to keep the window ahead of the playback position. On the other hand, if $a_B > 1$, the P2P window is well ahead of the playback position in t_m.

The Streaminglike scheme borrows its name from the live streaming systems, where the playback position and the P2P window (buffering in this case) are tied together. This is the most conservative window advancement scheme. The Progressive scheme pushes the P2P window forward as fast as possible, without jumping over Empty segments. *Fig. 1* shows an example for Progressive window advancement.

It follows from the definitions that these two window positioning schemes are indistinguishable, when $a_B < 1$. The following lemma also shows a trivial consequence of the definitions.

Lemma 1 With Linear segment selection and Progressive window placement $a_B = S$ and $E_A = W$.

Proof The Progressive advance scheme leaps over all the Full segments at the beginning of the window. The Linear segment selection starts the new downloads at the beginning of the window. Therefore, every round begins with $F_A = 0$, and $E_A = W$. From this starting condition $S = a_B$ follows naturally. □

4. System dynamics and the steady-state

4.1. State-space description of the system

The dynamics of the P2P window of the average of the clients is best described with a difference equation of the state variable $\overline{E}$. The number of Empty segments is updated in every round in accordance to the segment selection scheme and the window advancement scheme.

Theorem 1 The change in the number of Empty segments in a round on average is

$$\Delta\overline{E} = \overline{A} - \overline{S} - \overline{M}. \quad (1)$$

Proof In each round the number Empty segments is changed by the following three factors:

- Some of them are converted to Full
- When the window advances, new Empty segments are shifted in
- Miss occurs, if the segment shifted out is Empty

When combining these three we get $\Delta E = A - S - M$ for a single client. Calculating the expected value of this yields equation (1), because the mean of a sum is the sum of the means. □

Equation (1) is a first order, linear, time-invariant difference equation for the variable $\overline{E}$. On the right hand side $\overline{S}$ is a function of $\overline{E}$, and $\overline{A} - \overline{M}$ is the drive, because they are independent of $\overline{E}$ (at least in the Streaminglike scheme).

4.2. Steady state

Definition 5 Steady-state: a constant response of the system for constant input.

The sufficient condition for the steady-state to exist is the asymptotic stability of the system.

Definition 6 Asymptotic stability: a system is asymptotically stable, if its output is $y(t) \to 0$, when its input is $s(t) = 0$ (undriven response).

Note that usually the steady-state of a stochastic system is defined such that the distribution of the system variable converges to a certain distribution at time $t \to \infty$. In this work we could only examine the ensemble average in most cases; thus, we had to relax the requirements here.

In the following sections most of the theorems are going to present results for the steady-state of equation (1), but first the existence of the steady-state has to be proven. The steady-state does not necessarily exist for a dynamic system. Theoretically it is quite possible that the output keeps oscillating in the absence of an input signal, if there is no energy loss in the system.

Theorem 2 The system defined in equation (1) is asymptotically stable for both Linear and Random segment selection.

Proof The undriven response is obtained for $\overline{A} = 0$, in which case $\overline{M} = 0$ follows naturally. For both segment selection schemes $E > \overline{S}(E, p)$ if $E > 0$. Obviously, $S = 0$ if $E = 0$. Thus, from any $E_0 > 0$ starting point $E_0 + \sum \Delta E = E_0 - \sum \overline{S}$ reaches $E = 0$ eventually. □

In the steady-state the following connections between the parameters are true.

Lemma 2 With all segment selection and window placement schemes $\overline{S} = \overline{A} - \overline{M}$ in the steady-state.

Proof In the steady state $\Delta E = 0$; thus, we get the statement of the lemma from equation (1) for the system dynamics. □

Lemma 3 In the steady-state $O_A = ST$ with all segment selection and window placement schemes.

Proof In the steady state in every round S new downloads are started, and they all last for T timeframes. □

Lemma 4 In the steady-state

$$\overline{M} = \begin{cases} 0 & \text{if } \overline{a}_B > 1 \\ 1 - \overline{a}_B & \text{if } \overline{a}_B < 1 \end{cases} \tag{2}$$

with all segment selection and window placement schemes.

Proof If there is at least one Full segment at the beginning of the window, there is no miss, when the window advances. If $\overline{a}_B < 1$, it means that there is a Full segment at the beginning of the window with probability $\overline{a}_B$. In such cases $A = 1$ with both window placement schemes (see their definitions in Section 3.4); thus, a segment is missed with probability $1 - \overline{a}_B$. □

5. The number of full segments at the beginning of the window

Using the lemmas from the previous section we can prove a surprising statement that is a refined, stronger version of the statement of lemma 2 for the Progressive scheme.

Theorem 3 With the Progressive window positioning scheme $\overline{S} = \overline{a}_B$ in the steady-state.

Proof For Linear segment selection this is trivial, and it has already been part of lemma 1.

With Random segment selection we know from lemma 2 that $\overline{S} = \overline{A} - \overline{M}$.

If $\overline{a}_B > 1$, then $A = \overline{a}_B$ from the definition of the Progressive scheme, and $\overline{M} = 0$ from lemma 4; thus, $\overline{S} = \overline{a}_B - 0$.

If $\overline{a}_B < 1$, then we have $A = 1$ from the definition of the Progressive scheme, and $\overline{M} = 1 - \overline{a}_B$ from lemma 4; thus, $\overline{S} = 1 - (1 - \overline{a}_B)$. Note that this also proves $\overline{S} = \overline{a}_B$ for Linear selection; thus, using lemma 1 was not necessary. □

This theorem describes a strange phenomenon. The Random segment selection scheme starts downloads all over the P2P window, yet the number of

newly started downloads equals the number of Full segments at the beginning of the window. Note that only the average value of these two quantities equal: they have entirely different probability distributions, as *Fig. 4* shows. Their domain can also be different, if $D < W$, because $a \in \{0 \ldots W\}$, and $S \in \{0 \ldots \min(D,W)\}$.

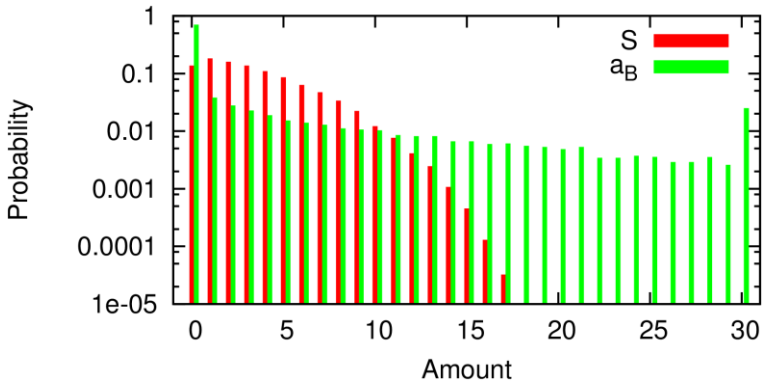

Figure 4: Comparison of the distributions of a_B, and S for the Progressive Random scheme (p=0.3, W=30, D=∞)

6. Number of new segment downloads initiated in a round

The next two theorems show how the number of new downloads started in a round can be calculated, if the number of Empty segments in the P2P window is known.

Theorem 4 With Random segment selection the number of new downloads started in a round is

$$\overline{S} = p\overline{E}_A \tag{3}$$

Proof If the number of Empty segments in the window is E_A in a round at point A, then the number of segments started follows Binomial distribution B(E_A, p), and pE_A downloads are started on average. However, E_A is also a random variable. Calculating average S for the average E_A yields

$$\overline{S} = \sum_{i=1}^{W} pi\mathbf{P}(E_A = i) = p\sum_{i=1}^{W} i\mathbf{P}(E_A = i) = p\overline{E}_A \tag{4}$$

which had to be proven. □

Theorem 5 With Linear segment selection the number of new downloads started in a round is

$$\overline{S} = p\frac{1-p^{\overline{E}_A}}{1-p} \tag{5}$$

Proof With Linear segment selection the round starts with F_A Full segments at the beginning of the window, and the remaining W - F_A segments are Empty. The probability that i segments are started in a round can be calculated from

$$P_i = \mathbf{P}(i\text{th is the first Empty}) = p^{i-1}(1-p) \tag{6}$$

Starting all of the Empty segments happens with probability p^{E_A}. The expected number of newly started downloads is

$$\begin{aligned}\overline{S} =& \mathbf{E}\{P_i\} + (E_A+1)p^{E_A} - 1 = \sum_{i=1}^{E_A} ip^{i-1}(1-p) + (E_A+1)p^{E_A} - 1 = \\ =& \frac{1-p^{E_A}}{1-p} - E_A p^{E_A-1} + (E_A+1)p^{E_A} - 1 = p\frac{1-p^{E_A}}{1-p},\end{aligned} \tag{7}$$

where the derivative of the formula for the sum of Geometric series was used. The generalization for the average $\overline{E}_A$ is done the same way as for the Random scheme. □

These results are applicable for the system at any time, not just the steady-state, but they depend on $\overline{E}_A$, which is unfortunately only known in the steady-state, as the next section shows.

7. Number of empty segments with the streaminglike scheme

The Streaminglike scheme always advances the window with A = 1 segments. The main concern in this case is the number of Empty segments in the window in the steady-state, because that will be the key to determine the necessary conditions for $\overline{M} = 0$.

7.1. Linear segment selection

If the window advancement scheme is less aggressive than the Progressive scheme, there may be some Full segments at the beginning of the window. The remaining E_A = W - F_A segments are all Empty, because the download process terminates at the first unsuccessful attempt.

Theorem 6 With Linear segment selection the expected value of the number of Empty segments in the steady-state is

$$\overline{E}_A = \begin{cases} \frac{\log(2p-1)-\log(p)}{\log(p)} & \text{if } p > 0.5 \\ W & \text{if } p \leq 0.5 \end{cases} \tag{8}$$

Proof We have seen in lemma 2 that $\overline{S} = \overline{A} - \overline{M}$ in the steady-state. With the Streaminglike scheme $A = 1$; thus,

$$\overline{S} = 1 - \overline{M} \tag{9}$$

If $\overline{M} = 0$, then $\overline{S} = 1$. Solving equation (5) for $\overline{S} = 1$ yields the first part of the formula in the statement. That formula yields $E = \infty$ for $p < 0.5$; thus, we also got the threshold p required for $\overline{M} = 0$.

If $\overline{M} > 0$, then we know from lemma 4 that $\overline{M} = 1 - \overline{a}_B$. We also know from the definition of the Linear selection that $F_B = a_B$. Using these we get that

$$\overline{S} = 1 - \overline{M} = \overline{a}_B = \overline{F}_B = W - \overline{E}_B = W - \overline{E}_A + \overline{S} \tag{10}$$

Thus, we got $\overline{E}_A = W$. □

Fig. 5 shows that this scheme results in a two-phase behavior: the segments in the P2P window are either all Empty, or all Full, depending on p. The results in the supercritical phase are indistinguishable from the Random segment selection, but in its subcritical phase the Linear scheme performs much worse.

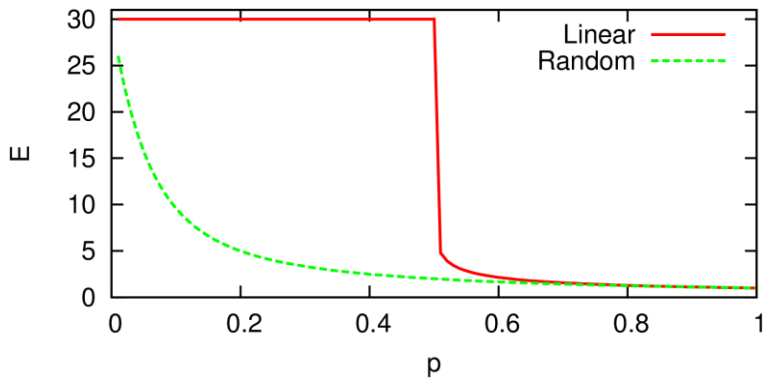

Figure 5: Number of Empty segments in the P2P window in the steady-state of the Streaminglike scheme; W=30

7.2. Random segment selection

The distribution of the number of Empty segments in the window is difficult to calculate, but getting their average number is quite easy.

Theorem 7 With Random segment selection the expected value of the number of Empty segments in the steady-state is

$$\overline{E}_A = \frac{1-(1-p)^W}{p} \tag{11}$$

Proof In the Streaminglike scheme $A = 1$; thus, we know that the ith segment of the P2P window has spent $W - i$ rounds in the window. Therefore, it is Empty, if it wasn't started in any of those rounds. The probability of a segment being empty is thus

$$\mathbf{P}(s_i = E_A) = (1-p)^{W-i} \tag{12}$$

independently of the other segments. The expected value of the number of Empty segments is the sum of the individual Empty probabilities

$$\overline{E}_A = \sum_{i=1}^{W} P(s_i = E) = \sum_{i=1}^{W} (1-p)^{W-i} = \frac{1-(1-p)^W}{p} \tag{13}$$

which had to be proved. □

In this case there is no phase change. As *Fig. 5* shows, the number of Empty segments gradually decreases as p increases. The minimum of both curves is at $\overline{E}_A = 1$, because after the advance there is always one Empty segment – the one at the end of the window.

8. Advance speed in progressive scheme

The P2P scheme is always self-sufficient, if there are no missed segments. In the Progressive scheme $\overline{a}_B > 1$ is a sufficient condition for this, because it means that the distance between the P2P window and the playback position keeps expanding. By the time it reaches the steady state, the probability that starting the download of the first segment fails so many times that a coerced advance becomes necessary is negligible.

In this section we calculate the expected advance speed of the P2P window, and the download initiation probability p_c required for $\overline{a}_B > 1$.

8.1 Linear segment selection

Theorem 8 With Linear segment selection the advance speed of the P2P window is

$$\overline{A} = \begin{cases} \overline{S} & \text{if } \overline{S} > 1 \\ 1 & \text{if } \overline{S} < 1 \end{cases} \tag{14}$$

Proof If we substitute $S = a_B$ from lemma 1 into the definition of the Progressive scheme, we get the equation of the statement. □

To calculate the download initiation probability p required for $S \geq 1$, and thus $M = 0$, equation (5) has to be evaluated for $S = 1$ and $E_A = W$.

$$1 = p\frac{1-p^W}{1-p} \quad \rightarrow \quad 0 = p^{W+1} - 2p + 1 \tag{15}$$

This is a polynomial of degree W+1, for which no general solution formula exists. We solved this equation numerically, and compared the results to the Random scheme in *Fig. 6*. As expected, $p = 0.5$ is the threshold value for reasonable window sizes ($W > 10$), which is significantly higher than the threshold value of the Random scheme.

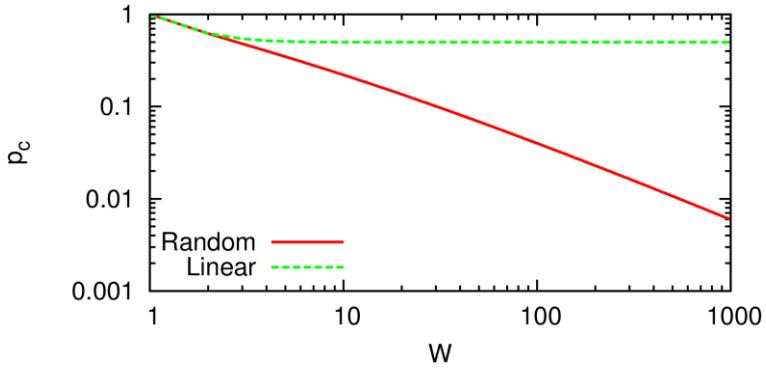

Figure 6: Comparison of the p_c threshold for $a > 1$, for Random and Linear segment selection schemes

8.2. Random segment selection

Determining the advance speed for Random segment selection is significantly harder than for Linear segment selection. The probability of a segment being Empty depends on the state of all the other segments, and the distribution of the advance speed. In turn, the distribution of the advance speed depends on the Empty probabilities of all the segments. We found that determining even the average advance speed requires multivariate numerical

optimization. Instead of finding an exact solution, we decided to construct an approximation of $\overline{a}_B$ that ignores this inter-dependency, yet gives usable results.

If we ignore the inter-dependency between *A* and *E*, and assume near constant $\overline{A}$ in at least the previous $W / \overline{A}$ rounds, the segment in the *k*th position has been in the window for $(W-k+1)/\overline{A}$ rounds, including the current one (in point B). In other words, it had exactly that many opportunities to become Full so far. The first one is special, because it must have been definitely left Empty in the last round, otherwise the window would have advanced at least one segment more. The number of opportunities each segment had so far is thus

$$g(k) = \begin{cases} \frac{W-k+1}{\overline{A}} & \text{if } k > 1 \\ 1 & \text{if } k = 1. \end{cases} \tag{16}$$

Our first idea was to take formula (11) for $\overline{E}_A$ in the Streaminglike case, adapt it to $A \neq 1$ using $g(k)$, and calculate $\overline{S} = p\overline{E}_A$, as shown in Theorem 4, but that was a failure. The results were completely different from the simulation results.

The approach that gave usable results was to construct the probability distribution of the number of Full segments at the beginning of the P2P window. Here we assume that $\overline{A} = \overline{a}_B > 1$*; t*he distribution of the number of Full segments at the beginning of the window is thus

$$\begin{aligned} \mathbf{P}(f(p) = 0) &= (1-p)^{g(1)} \\ \mathbf{P}(f(p) = W) &= \prod_{j=1}^{W} (1-(1-p)^{g(j)}) \\ \mathbf{P}(f(p) = i-1) &= (1-p)^{g(i)} \prod_{j=1}^{i-1} (1-(1-p)^{g(j)}), \end{aligned} \tag{17}$$

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

$$\begin{aligned} \overline{A} &= \overline{a}_B = \mathbf{E}\{f(p)\} = \\ &= \sum_{i=2}^{W} (i-1)(1-p)^{g(i)} \prod_{j=1}^{i-1} (1-(1-p)^{g(j)}) + W \prod_{j=1}^{W} (1-(1-p)^{g(j)}). \end{aligned} \tag{18}$$

This cannot be solved analytically, because $g(k)$ in the exponents also contain *A*. We solved equation (18) numerically by fine-tuning the *A* substituted

into $g(k)$ until equation (18) returned the same A, for the given p, and W. When comparing the results with simulation results, we found that the probability distribution in equation (17) is completely different from the real distribution, as demonstrated in *Fig. 7*, but $\bar{a}_B$ from equation (18) always underestimates the real one, and it's usually about 95-98% of it. *Fig. 8* shows an example of this result for $W = 10$.

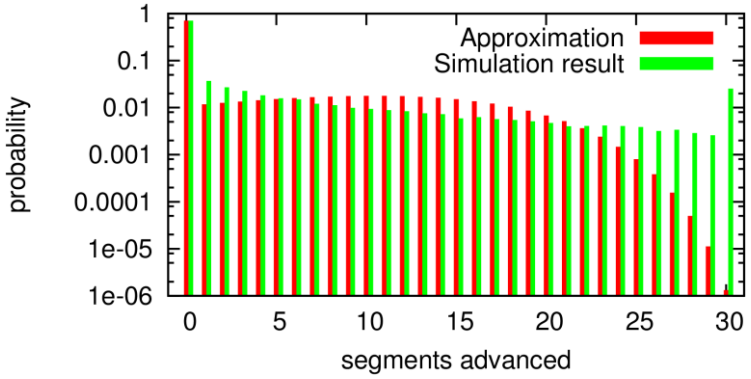

Figure 7: Comparison of the real distribution of a, and the distribution predicted by the approximation (p=0.3, W=30, D=∞, a_{approx}=3.185, $a_{measured}$=3.265)

The above approximation method assumes that $\bar{A} = \bar{a}_B > 1$. As *Fig. 8* shows, for $a_B < 1$ this method indeed looses its precision considerably. We tried to correct this by using the same formula, but substituting $A = 1$ into $g(k)$; however, it made the results worse. Accurate results for $a_B < 1$ would have been useful to calculate the miss probability, but at least this approximation technique is good enough to provide a threshold p for miss-free operation.

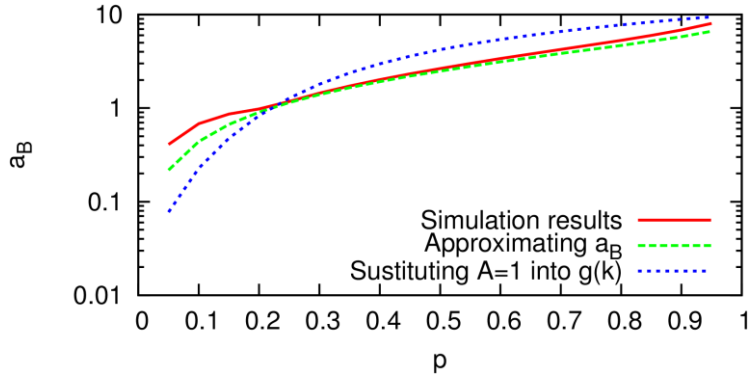

Figure 8: Comparison of the simulation result, the approximations of a_B with the assumptions A=a_B, and A=1 (W=10, D=∞)

Solving equation (18) numerically for $\overline{a}_B = 1$ yields a threshold value of p for the P2P scheme to be self-sufficient. As *Fig. 9* shows, this result overestimates the real threshold value obtained from simulations, because the approximation underestimates $\overline{a}_B$. This threshold is shown in *Fig. 6*, compared to the threshold value for the linear scheme. As expected, the Progressive scheme is much more efficient than the Streaminglike scheme.

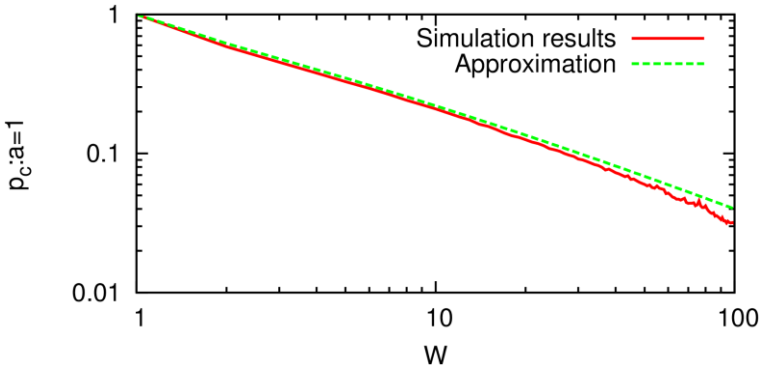

Figure 9: Comparison of the approximation and the simulation results for the p_c threshold for a>1 for the Progressive Random scheme

9. The probability of missing a segment

At the beginning of the download process the transient processes behave quite erratically, which prevents determining analytically the number of segments missed. In the steady-state, however, the miss probability is easy to calculate. The theorems in this section are specializations of lemma 4.

9.1. Linear segment selection

With this segment selection scheme the miss probability is the same for both window placement schemes under normal circumstances.

Theorem 9 With Linear segment selection the miss probability in the steady-state is

$$\overline{M} = \begin{cases} 1 - \overline{S} & \text{if } p < 0.5 \\ 0 & \text{if } p \geq 0.5. \end{cases} \tag{19}$$

With the Progressive scheme the threshold is higher for tiny P2P windows (see *Fig. 6*), but that can be safely ignored.

Proof For the Streaminglike case we have already seen this in the proof of theorem 6. If $p < 0.5$, then $\overline{M} = 1 - \overline{S}$. If $p \geq 0.5$, then $\overline{S} = 1$, and $\overline{M} = 0$.

In the Progressive case theorem 8 showed that the window is well ahead of the playback position, if $\overline{S} \geq 1$, for which the necessary condition is $p \geq 0.5$ for reasonable P2P window sizes (see *Fig. 6*); in this case $\overline{M} = 0$. If $\overline{S} < 1$, the system behaves exactly like the Streaminglike scheme. □

For small *p* one would think that the window is completely Empty, and the first segment is converted to Full with probability *p*; thus, $\overline{M} = 1 - p$. This is only true for a single client; though, because the average behavior of several clients also includes the uncertainty of a_A.

The exact value of $\overline{M}$ in the steady-state can be computed from equation (5), by substituting equation (8).

The advantage of the Linear segment selection scheme is thus the ability to run the system without missed segments regardless of the window positioning scheme, even though the required *p* is rather high.

9.2. Random Segment Selection

With Random selection the window placement scheme can heavily influence the miss probability.

Theorem 10 In the steady-state of the Streaminglike Random case the number of segments missed is

$$\overline{M} = (1 - p)^W \tag{20}$$

Proof The probability of missing a segment equals the probability of the first segment in the window being Empty. That segment has spent *W* rounds in the window; thus, it is still Empty only if all of the starting attempts failed, which happens with probability $(1 - p)^W$.

The same result can be obtained by using theorem 4, lemma 2, theorem 7, and knowing that *A*=1 with the Streaminglike scheme. The miss probability is thus

$$\overline{M} = A - \overline{S} = 1 - \overline{S} = 1 - p\overline{E} = 1 - p\frac{1 - (1 - p)^W}{p} = (1 - p)^W \tag{21}$$

as expected. □

Theorem 11 In the steady-state of the Progressive Random case the number of segments missed is

$$\overline{M} = \begin{cases} 1 - \overline{a}_B & \text{if } \overline{a}_B < 1 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

Proof This is basically lemma 4 with theorem 3 substituted. □

In the Progressive Random case the numerical result for $\overline{a}_B$ can be used to estimate p_c, and it can also give a crude estimate of $\overline{M}$.

Although the Random segment selection scheme cannot guarantee that the system is self-sufficient in all cases, the probability of a missed segment can be kept minimal by choosing a sufficiently large P2P window. *Fig. 10* shows a comparison of the Miss probability for all four combinations. The Random selection scheme is much more efficient than the Linear selection for low *p* values, but when the download initiation probability is high enough, the performance of the two segment selection schemes is identical.

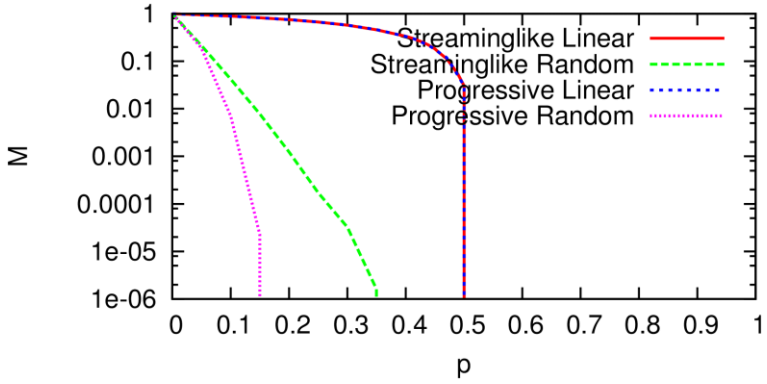

Figure 10: Miss probability with the four combinations of the schemes; *W*=30

10. Limited downlink capacity

In the previous sections $D = \infty$ was always assumed, but in real P2P clients that is not always true. In the literature it is generally assumed that the performance of a P2P system is limited by either the uplink speed of the clients, or their downlink speed [13, 8]. In this section we show that these two kinds of limitation can occur at the same time.

We initially discovered the phenomenon discussed in this section, when we were conducting our simulation study of a combined caching-P2P VoD system [7]. In that simulation study we used our own network simulator, *cdnsim*, which we also used to obtain some of the results presented in this section.

The remainder of the results of this section was obtained with a much simpler simulator, which we called *sim.pl*. We originally created it to validate the theoretical results presented in the previous sections, and later extended it to support limited downlink capacity. It basically starts with an array of "E", runs the algorithm of *Fig. 2*, with a predefined constant download initiation probability, and stops when all elements became "F".

The analysis presented here focuses mainly on the Progressive Random scheme, because, being the most efficient combination, it is the one that is affected the most by the limited downlink. Most of the observations are also applicable to the other scheme combinations, but the numeric values are different.

The downlink capacity D of the clients cannot be arbitrarily small, and there is a finite size, above which it is not limiting. The following lemmas establish the valid interval for the downlink capacity.

Lemma 5 The downlink capacity of the clients must be $D \geq T$.

Proof A timeframe is the time it takes to play a video segment, and it takes *T* timeframes to download a segment. Therefore, the client needs to be able to download at least *T* segments in parallel to keep up with the playback speed.

Lemma 6 If $p = 1$, the downlink limits the efficiency of the P2P downloading, when $D < WT$ in the Progressive scheme.

Proof If $p = 1$, all of the Empty segments in the P2P window are filled in each round; thus, $A = W$, and the next round is started with $E_A = W$ independent of the segment selection scheme. Therefore, $S = W$, and $O = WT$.

Lemma 7 If $p = 1$, the downlink limits the efficiency of the P2P downloading, when $D < W + T$ in the Streaminglike scheme.

Proof If $p = 1$, the window is filled in the first timeframe; thus, $S = W$. In the subsequent rounds there is only one Empty segment in the window, because *A*=1, and the segment is converted into Full as soon as it is shifted into the window; therefore, *S*=1. The number of ongoing downloads is maximal in the *T*th round: this is the last one, where the initial downloads are still ongoing. In this round $O = W + T$.

Note that the Streaminglike scheme only needs at most *D*=*T*, if the startup phase is not considered, because $S \leq 1$ in the steady-state with both segment selection schemes. If $T < D < W + T$, then it takes more time to reach the steady-state, but in the steady-state the system is not limited by the downlink capacity.

As explained in section 3.1, the download initiation probability *p* in the previous sections was a supply/demand ratio that represents the global state of the P2P swarm. In this section

$$q = p * \mathbf{P}(O < D) \tag{23}$$

will be the *real* download initiation probability, which may be smaller than p due to the limited downlink. It is obvious that q changes even within a round, because O is increased with each new successful download attempt.

In summary, to determine whether D limits S, we need to know the distribution of O, and to compute that we need the distribution of S. The distribution of S obviously depends on the distribution of E_A, and O; thus, constructing an exact formula for q seems hopeless. For this reason, there will only be simulation results in the remainder of this section.

Fig. 11 shows the download initiation probability in *cdnsim*. The oscillation in the startup process makes it difficult to see, but q starts at around 0.13, and it decreases to around 0.08 due to the limited downlink capacity. At the end of the download process the P2P window cannot go further; thus, with the decreasing number of remaining Empty segments the downlink capacity of the client is no longer a limiting factor. This causes the $q=p$ spike at the end of the curve.

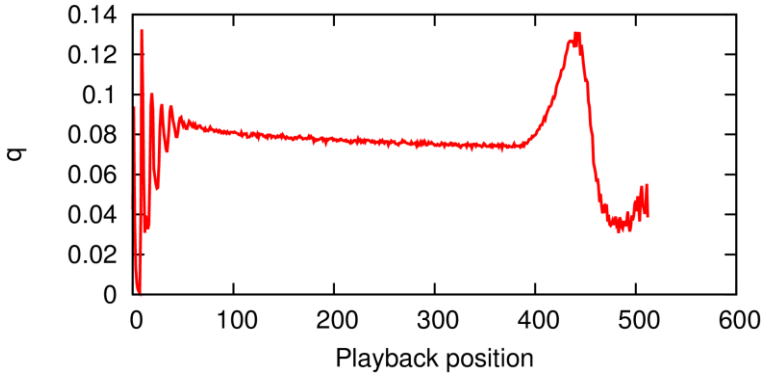

Figure 11: Download initiation probability during the playback process in cdnsim with limited downlink; *T*=8, *W*=30, *D*=16

A similar pattern can be observed in *Fig. 12*, but this time p is known precisely, because for *sim.pl* it is an input parameter. This figure also shows $p\mathrm{P}(O_B < D)$ for comparison, and in this parameter configuration the two almost perfectly match. With other parameter settings the difference between them can be larger, because the downlink can get filled anywhere between point A and point B during the round.

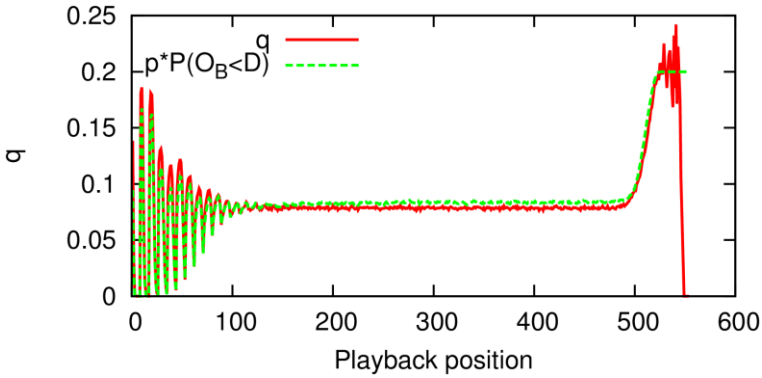

Figure 12: Download initiation probability during the playback process in sim.pl with limited downlink; p=0.2, T=9, W=30, D=10

To further verify that this phenomenon is indeed caused by the limited downlink, and not by the finite P2P window size, *Fig. 13* offers some more insight. The q/p ratio is 1 for small p values, but after that it decreases rapidly. With higher downlink capacity it only starts to decrease later, but the steepness of the slope remains almost the same. The q/p ratio also depends on W, of course: with bigger P2P window the downlink limitation becomes more severe.

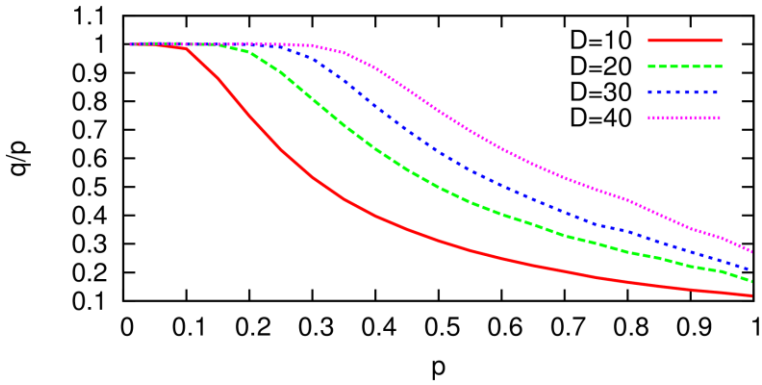

Figure 13: The q/p ratio for various downlink capacities in sim.pl; T=9, W=30

There is yet another curious phenomenon, which arises with limited downlink capacity. Fig. 14 shows the distance between the playback position and the start of the P2P window. With the Progressive scheme it is naturally not constant, but the peak at the end is quite remarkable. When q increases at the end of the video, the speed of the P2P window follows, and thus creates a peak on this diagram as well. According to our simulation results, the window position doesn't always have a peak, when q does.

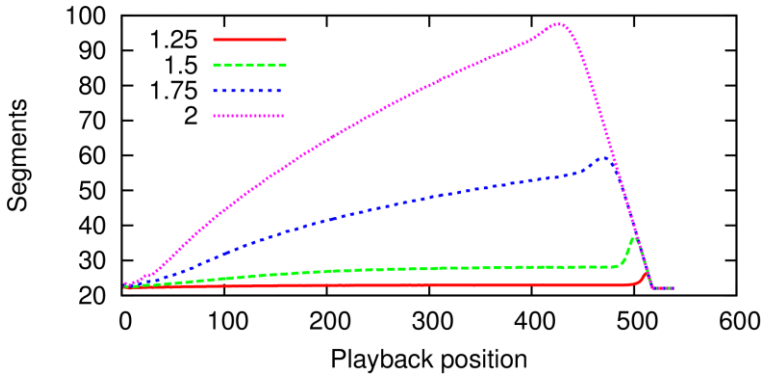

Figure 14: Distance between the playback position and the beginning of the P2P Window with the Progressive Random scheme in cdnsim, for some downlink speeds in range [1.25…2] relative to the video bitrate

This pattern only arises for the average behavior of several clients; though. When observing a single client, the relative position of the P2P window is completely different, as *Fig. 15* demonstrates.

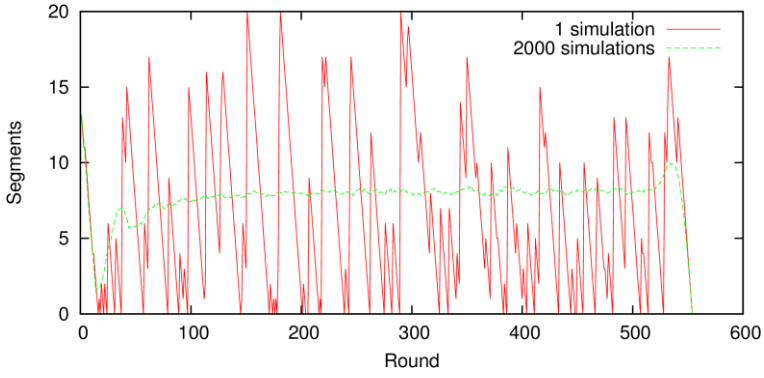

Figure 15: Distance between the playback position and the beginning of the P2P Window; comparing the behavior of a single client and the ensemble average

This description of the limited downlink has the advantage that the effect of $D < \infty$ is contained entirely in the download initiation probability; thus, the results of the previous sections still hold, if q is used in place of p in the formulas.

11. Conclusions

In this paper we examined analytically the performance of a generic P2P VoD client that uses closed P2P windowing. In this model the clients can only initiate the download of the video segments that are within the P2P window; the segments of the window can be selected in linear or random order. We examined two methods of positioning the P2P window: the Streaminglike scheme maintains a fixed distance between the window and the playback position, while the Progressive scheme pushes the window forward as fast as possible.

We described the evolution of the state of the P2P window with a linear difference equation, and showed that it can reach a steady-state. We developed formulas for the state descriptor quantities, depending on the segment selection scheme, the window positioning scheme, and the segment download initiation success probability. Perhaps the most important results of this analysis are the criteria for the P2P system to be self-sufficient, but we also discovered an interesting connection between the advance speed of the P2P window and the number of downloads initiated in a round, when using the Progressive window placement scheme. Finally, we examined the effect of limiting the downlink speed of the clients, and found that, unlike the popular assumption, the uplink and the downlink capacities limit the efficiency of the content sharing at the same time.

References

[1] Axelrod, R., "The evolution of cooperation". *Basic Books* (1984).

[2] Carlsson, N., Eager, D. L., "Peer-assisted on-demand streaming of stored media using BitTorrent-like protocols". In: *Proc. of NETWORKING'07*, pp. 570–581. Springer-Verlag, Atlanta, GA, USA (2007).

[3] Ciullo, D., Martina, V., Garetto, M., Leonardi, E., Torrisi, G. L., "Stochastic analysis of self-sustainability in peer-assisted vod systems". In: *Proc. of INFOCOM'12*. IEEE, Orlando, FL, USA (2012).

[4] Cohen, B., "Incentives build robustness in BitTorrent". In: *Proc. of 1st Workshop on Economics of Peer-to-Peer Systems* (2003).

[5] Erman, D., Saavedra, D., Sánchez González, J., Popescu, A., "Validating bittorrent models". *Telecommunication Systems 39*, 103–116 (2008).

[6] Locher, T., Moor, P., Schmid, S., Wattenhofer, R., "Free riding in bittorrent is cheap". In: *Proc. of HotNets-V*. Irvine, CA, USA (2006).

[7] Máté, M., Vida, R., Mihály, A., Császár, A., "Offloading video servers: P2p and/or caches?" In: *Proc. of NETWORKS2012*, pp. 400–405. Rome, Italy (2012).

[8] Parvez, N., Williamson, C., Mahanti, A., Carlsson, N., "Analysis of BitTorrent-like protocols for on-demand stored media streaming". *SIGMETRICS Perform. Eval. Rev.*, ACM 36(1), 301–312 (2008).

[9] Pouwelse, J. A., Garbacki, P., Epema, D. H. J., Sips, H. J., "The bittorrent p2p file-sharing system: Measurements and analysis". In: *Proc. of IPTPS'05*. Ithaca, NY, USA (2005).

[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] Shah, P., Pâris, J. F., "Peer-to-peer multimedia streaming using bittorrent". In: *Proc. of IPCCC'07*. New Orleans, LA, USA (2007).
[12] Vlavianos, A., Iliofotou, M., Faloutsos, M., "BiToS: Enhancing BitTorrent for supporting streaming applications". In: *Proc. of INFOCOM'06*, pp. 1–6. IEEE, Barcelona, Spain (2006).
[13] 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).
[14] Zghaibeh, M., Harmantzis, F., "A lottery-based pricing scheme for peer-to-peer networks". *Telecommunication Systems 37*, 217–230 (2008).
[15] 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).