University of Würzburg
Institute of Computer Science
Research Report Series

Investigation of chunk selection strategies in peer-assisted video-on-demand systems

Tobias Hoßfeld, Simon Oechsner, Frank Lehrieder,
Christian Bergner, Phuoc Tran-Gia

Report No. 446 August 2008

University of Würzburg, Institute of Computer Science,
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{hossfeld,oechsner,lehrieder,bergner,trangia}@informatik.
uni-wuerzburg.de

Investigation of chunk selection strategies in peer-assisted video-on-demand systems

Tobias Hoßfeld, Simon Oechsner, Frank Lehrieder, Christian Bergner, Phuoc Tran-Gia
University of Würzburg, Institute of Computer Science,
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{hossfeld,oechsner,lehrieder,bergner,
trangia}@informatik.uni-wuerzburg.de

Abstract

The applications for peer-to-peer (P2P) architectures recently have evolved from straightforward file download to more demanding services like Video-on-Demand (VoD). Here, not only the known advantages of P2P overlays are essential, but also a timely delivery of video frames for their play back. Since the video is watched during the download process, this poses a major challenge, which has to be mastered to guarantee a high service quality to the end user. One mechanism influencing the performance of a P2P content distribution network is the chunk selection strategy in systems using multi-source download. This paper aims at systematically investigating the impact of the chunk selection strategies on the system behavior and its self-organization capabilities, as well as the user perceived quality. In particular, we answer if just an adaptation of this strategy suffices to make such systems viable for VoD services.

1 Introduction

In the last few years, the performance of mainly user-initiated peer-to-peer (P2P) file sharing networks has led to the application of their mechanisms for efficient and also legal content distribution by content owners. Software releases and patches as well as movies are currently distributed with support from dedicated P2P networks. This is due to the fact that P2P networks offer several advantages over traditional content servers. They harness users' resources in a self-organizing and scalable fashion, withstanding even flash-crowd effects, i.e., the sudden arrival of a large number of downloaders. Since this is a common scenario for new and popular content, such robustness makes these systems attractive for content providers.

Most efficient P2P content distribution networks (CDNs) implement the principle of multi-source downloads. Each participating client or peer is able to download parts of the complete content from different sources in parallel, leading to a faster creation of new sources for these smaller parts, commonly called chunks or pieces. The basic mechanisms governing this download process are the cooperation strategies between the peers, namely the chunk and the peer selection process.

One of the most recent applications of P2P content distribution are P2P-based Video-on-Demand (VoD) systems. A VoD service offers its users a selection of movies to watch. Once the user chooses a video, it is streamed to his end system and starts playing back after a short buffering phase. This application has much higher requirements than simple file-sharing or file

downloading, since the video transmitted is watched during the download process. The quality of the video service experienced by the end user, also called Quality of Experience (QoE), is much more sensitive to performance problems than the quality of a download service. Inefficient use of resources by the overlay may cause parts of the video to be received too late or even never at the end user. This leads to fragments in consecutive frames of the movie or even missing pictures.

The aim of this work is to systematically investigate common chunk selection strategies known from file sharing systems. The question we want to answer in this paper is whether it is sufficient to adapt the chunk selection strategy in an exemplary P2P-based VoD architecture to guarantee a sufficient QoE, or whether also the peer selection process or other mechanisms have to be adequately implemented. To this end, we give a background on cooperation strategies in CDNs and review related work in Section 2. We then present the model used in the simulative performance evaluation comparing different chunk selection strategies in Section 3. In Section 4, the results from this evaluation are discussed. The lessons learned from this performance study are summarized in Section 5, where we will also point out our current and future work which aims at a QoE controlled, peer-assisted VoD system.

2 Background: Cooperation Strategies in CDNs

P2P systems are widely used as content distribution networks (CDNs) in today's Internet. Users (peers) do not receive the whole content from one server but interchange already received parts of the content among each other. This reduces the load and the bandwidth requirements on the server considerably. Furthermore, P2P networks are much more scalable as the upload capacity increases with the number of users. Though, proper cooperation strategies are necessary to permit fast content distribution while preventing selfish peers from cheating.

Different types of content can be distributed by CDNs. In particular, P2P networks are used for file sharing, live streaming, and video on demand. In the following, we review related work in these areas.

2.1 File Sharing

BitTorrent [1] is a popular CDN for simple file distribution. It splits a file into several chunks and every chunk into blocks. A new peer joining the network contacts the tracker which informs it about other peers exchanging the specific file. The peer requests blocks from the other peers and downloads them when the senders have free upload capacity. In contrast to server based file distribution, the content is downloaded from multiple sources at the same time (multi-source download).

For cooperation, basically two mechanisms are decisive: chunk selection and peer selection. Chunk selection (also called piece selection) mechanisms decide which chunk will be requested by the peer. To prevent situations where some chunks are only sparsly distributed, BitTorrent uses the least-shard first (LSF) policy, i.e., rare chunks are preferentially exchanged. The peer selection algorithms select the peers where download requests are placed. To avoid that selfish peers do not share their content, the chunk exchange is based on a tit-for-tat policy in BitTorrent [2].

2.2 Live Streaming

Live streaming applications can also profit from the use of CDNs. However, they impose further constraints on the mechanisms because all users watching a news broadcast or a popular sports event are typically interested in the same piece of the stream at the same time. Therefore, delays of one or more minutes seem unacceptable and chunk selection strategies like least-shared first inappropriate.

Two popular examples are CoolStreaming and CoopNet. CoolStreaming [3, 4] uses an unstructured overlay network. In contrast, CoopNet is build on tree-based network structure [5]. In [6], Tewari and Kleinrock develop an analytical model for P2P live streaming based on BitTorrent.

2.3 Video on Demand

In contrast to simple file sharing, the VoD users want to watch the video before finishing the download of the entire file. Therefore, special cooperation strategies are necessary to maximize the number of video frames the clients recieve before their playout time. Compared to live streaming, VoD differs mainly in two aspects. Firstly, peers start downloading the video at very different times, i.e., while one reaches the end of the video, others just start watching. Secondly, VCR functions such as fast forward are expected by the user.

BASS [7] implements a hybrid VoD system using BitTorrent clients and dedicated video servers. Peers interchange chunks of the video file using the BitTorrent mechanism explained above. In addition, they download chunks which they cannot recieve in time via BitTorrent from a central server. Choe et al. pursue a similar approach in [8]. Erman [9] investigated the necessary extensions to BitTorrent in order to make it capable of delivering VoD. He concludes that the piece and peer selection algorithms are the crucial components to be adapted. Regarding peer selection, [10] proposes that peers interested in the same part of the video preferentially exchange chunks. Piece selection algorithms are addressed, amongst others, in [11, 12, 13, 14]. These studies develop new piece selection strategies, which are more suitable to VoD, and compare them to strategies like least-shared first or earliest chunk first. These approaches are very similar as all proposed piece selection algorithms stochastically choose the next chunk to download whereby chunks with a near playout deadline have higher probatilities to be elected. Janardhan and Schulzrinne design a P2P architecture for set-top boxes based on an IP network [15] including admission control and locality aware content fetching. However, that paper does not mention simulation experiments or quantitative results.

In contrast to most existing work, we do not propose a complete architecture for P2P based VoD. However, we systematically investigate the necessary mechanisms. In this paper, we focus on chunk selection strategies. We study their performance under various load conditions by simulation experiments and show that under some circumstances chunk selection mechanisms alone are not capable of providing a proper VoD service. As a consequence, further adjustments have to be made to the cooperation mechanisms, e.g. to the peer selection algorithm.

3 Modeling a P2P-based VoD System

The scenario we chose for evaluating different chunk selection strategies consists of servers and peers, forming an overlay network to distribute a video. The servers are provided to support the network by adding resources, while peers are the client machines of end users that want to watch the video. These clients are set-top boxes in our scenario, which can be parametrized by the overlay service provider and offer enough storage capacity for several videos. They are assumed to be connected via the broadband internet connection of the user, reserving a part of the total bandwidth of the users internet access. The offered video service is a VoD service as described in the previous section. In the following, we describe how we modeled this system on a level abstract enough for simulation.

3.1 Video model

In the simulations we consider two videos in low and high resolution for different load scenarios that are distributed to the peers. As video we use the movie "Seven Years in Tibet" of duration $d_v = 2{:}09{:}54\,\mathrm{h}$. The video in high resolution ('*high-res*') is encoded with the variable bit rate H.264 video codec using the primary High Profile (HiP) with 720×576 pixel; as audio codec the MPEG-4 AAC with 24 kHz in stereo is used. The low resolution video ('*low-res*') is scaled down to reduce the required video bit rate and, thus, the system load.

Both videos contain 194,859 frames that are played back with a frame rate of 25 $^{\text{frames}}/_{\text{s}}$. The total size of the smaller video is 122.08 MB, with the larger video having a size of 489.77 MB, resulting in an average bitrate of 131.39 and 527.11 $^{\text{kbit}}/_{\text{s}}$, respectively. Each frame is characterized by its size and type, i.e., being an I-, P- or B-frame. The videos are separated into chunks of size 1 MB, with the low-res video consisting of 123 and the high-res one of 490 chunks. Each chunk in turn consists of 16 blocks of size 64 kB. The chunks and blocks are not adapted to the video structure, i.e., to frame or Group of Picture (GoP) sizes, but have a fixed size to enable an efficient multi-source download process.

As a consequence, the average number of frames and GoPs contained in a single block or chunk differs for both videos, and therefore the average playback time of a block or chunk. It has to be noted that we have chosen the upload bandwidth of the providing peers and servers such that the chunk upload time is larger than the average chunk playback time. This allows to investigate the scalability of the chunk selection strategies in overload or extreme flash crowd situations. The parameters of both videos are summarized in Table 1.

When playing back the video, the current playback position is updated at block ends. The time it takes to play back a block is calculated from the number of frames it contains and the frame rate of the video. Only successfully received blocks on transport layer are given to the application layer and, thus, can be played back. This means for each block a *deadline* in time exists when it has to be available for playback. Otherwise, the entire block and the frames within are canceled.

In contrast, changes in the quality of the video, i.e., whether frames are available for playout or not, are updated at the playback time of the last and first I-frame before/after the currently considered block end for quality degradation and increase, respectively. This is due to the forward- and backward-referencing B- and P- frames in a GoP. Thus, we can determine how much video

Table 1: Overview of the parameters for the low-res and the high-res video

Resolution	*high-res*	*low-res*
Size [MB]	489.77	122.08
Length [min]	129.91	129.91
Avg. bit rate [kbps]	527.11	131.39
Avg. chunk playback time [sec]	15.91	63.85
Chunk upload time [sec]	87.38	87.38
Avg. number of GoPs per chunk	3.62	14.52

data was available for each part of the video on a GoP level.

A peer starts watching a video after the first of the following events: receiving either the first 5 consecutive blocks or after 1 minute has passed after requesting the video stream.

3.2 Peer model

The peer join process is modeled by a Poisson process with rate $\lambda = 0.5\,{}^{1}/_{\min}$. A peer stays online for the duration of watching the video plus an exponentially distributed seeding time t_s with a mean value of 60 minutes. To support the network and to initially offer the content, we assume 20 VoD servers are in the network which stay online during the whole simulation. Except this behaviour and the fact that they have the complete video and act as seeders from the start, these servers behave just like regular peers.

Each peer joins the system with no video data locally available and receives knowledge about all other peers from the tracker. Also, every peer already online is informed immediately about the new peer. Each peer places a download request in the form of a buffermap request at every other peer at all times, with the exception of remote peers it is currently downloading from. However, the number of download slots is limited to 10 , i.e., peers can download content from at most 10 peers in parallel.

All download requests at one peer wait to be served in an upload queue with FIFO processing discipline. Each peer has a single upload slot. When an upload is finished, the peer with the first download request in the queue is contacted to check whether the request is still valid (in case the peer has finished downloading the video), whether the remote peer (*downloader*) has spare download capacity and whether the local peer (*uploader*) has chunks the remote peer still needs to download. To this end, a current buffermap of the uploader is transmitted to the downloader. If not all of these criteria are met, the request is canceled and the downloader sends a new request to be inserted in the upload queue if it is no seeder. However, if the uploader has content the downloader wishes to download, one of the chunk selection strategies described in Section 3.4 is applied to the set of chunks available for upload, and finally the blocks of the selected chunk are uploaded in order. After the upload is finished, the downloader again places a download request in the local peer's upload queue. The complete signaling message flow between two peers is depicted in Figure 1.

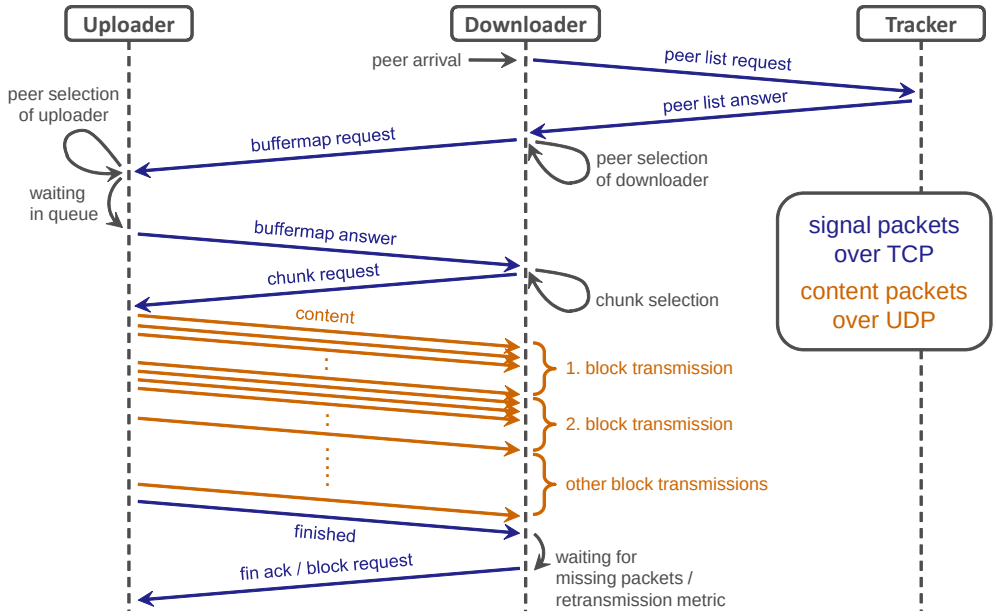


Figure 1: Signaling message flow

3.3 Network model

Data transfer and signaling message exchange is modeled on IP packet level. Chunks are transmitted as blocks, which in turn are separated into fragments of the size of payloads of IP packets. Each IP packet payload has 28 bytes added in total for the IP and UDP headers. In contrast, signaling messages are transferred as one IP packet of size $100\,\mathrm{B} + 40\,\mathrm{B}$ IP and TCP header.

The transmission bottleneck of the network is assumed to be the access network. Therefore, we do not simulate the entire physical network with its routers and links, but model the access network of a peer by using a download and an upload network queue for each peer. A peer receives packets from the download queue with the download access speed of 1024 $^{\mathrm{kbit}}/_{\mathrm{s}}$, while the upload queue sends packets to the network with an upload capacity of 96 $^{\mathrm{kbit}}/_{\mathrm{s}}$. With this speed, the packets also arrive at the download queue of the remote peer in a connection. Each queue can hold 10 IP packets, with the exception of signaling packets which are assumed to be small enough in any case. We also assume that signaling uses the TCP protocol so that it can be conducted reliably. Packet loss occurs when data packets are inserted into a full queue, with the last packet being dropped.

Since we only consider peers with one upload slot in this scenario, we do not need to use bandwidth sharing for uploading data.

3.4 Chunk selection strategies

In this section, we present the strategies used to select the chunks to download from the set of eligible chunks of an uploader, i.e., chunks the downloader still needs and that the uploader already has stored. As the blocks and chunks of the videos have a playout deadline – if received

later they are dropped on application layer – a required chunk means that the peer has yet not downloaded this chunk and that the chunk is not expired due to this deadline. This deadline of blocks and chunks distinguishes these strategies for VoD services from file sharing systems.

3.4.1 Random selection

The first selection strategy randomly chooses one of the eligible chunks for download, labeled with 'rand' in the figures in Section 4. While easy to implement and not requiring any knowledge about the file to be shared or the distribution of chunks, this strategy causes problems especially in scenarios with selfish peers, i.e., peers that go offline immediately when finishing a file download, or in networks with a high churn rate [16].

When selecting chunks at random, at some point in time one chunk will be less distributed than the rest. This is due to the fact that only chunks that have been downloaded can be shared, therefore chunks that are distributed earlier are spread much better than chunks distributed later. Then, for many peers this least-shared chunk will be the last chunk they are missing for a complete copy of the file, due to its low availability. When they finally get this last chunk, they will leave the system if they are selfish peers, or they will have gone offline before being able to download it under high churn conditions. Therefore, the number of copies of this chunk does not increase, in extreme cases leading even to the loss of this chunk in the overlay if the few available copies go offline. This problem is called chunk starvation and leads to high download times or failure to download a file.

3.4.2 Least-Shared First

To counter the chunk starvation, a strategy called Least-Shared First (LSF) was integrated e.g., in the BitTorrent [2] system. It selects the chunk with the least number of copies in the network from the eligible chunks of an uploader. If there are several chunks which are least shared, then the first one regarding their deadlines is chosen. Ideally, this prevents any chunk from being shared less than the rest and keeps the availability of all parts of a file roughly equal. However, this method requires global knowledge about the distribution of all chunks in the whole network. In overlays with a central component, such as a tracker, this might be possible, in fully distributed systems heuristics or locally available information have to be used, leading to less-than-optimal results.

In our simulation, we assume that we have global knowledge via the tracker, and correctly select the least shared chunk from the available ones for download. There are two variants for this strategy. The first one does not consider the playback position of the downloading peer, but includes all chunks of the file in its computation. Hence, chunks no longer useful for the video playout at a user may also be downloaded. This is to make sure that the whole file is well distributed, so that peers joining the network at a later stage have enough download options. This variant is referred to as least-shared first with requesting expired chunks and labeled with 'lsfwrx' in Section 4. The second variant, labeled with 'lsf', only considers chunks that have not yet been played back and that can be received before their playback deadline. These are the only chunks the local peer does profit from downloading, thus this strategy focuses more on the benefit of the local user instead of the whole overlay.

3.4.3 In Order

A simple strategy that captures the linear importance of chunks for playout is to simply download all chunks in consecutive order. Since it can not be guaranteed that all chunks are available, the exact strategy is to select the available chunk closest to the downloading peer's playback deadline that this peer is still missing. This strategy is labeled with 'ord' in the figures in the next section.

3.4.4 Hyperbola

This strategy tries to balance the urgency for downloading chunks near their playback deadline with the need to stabilize the distribution of all chunks in the overlay. It ranks chunks according to the following function:

$$R(c_i) = \frac{\alpha}{A^X} + \frac{\delta}{D^Y} \, ,$$

where $R(c_i)$ is the rating value of chunk c_i; A is the relative availability of c_i, i.e. the number of sources of chunk c_i divided by the total number of peers and servers in the system; and D is the temporal distance of the playback deadline of c_i to the current playback position of the video at the downloading peer. This function returns higher values for chunks that are close to their deadline as well as for chunks that are rare in the network. The relative importance of these factors can be tuned with the parameters α, δ, X, and Y. In our simulations, we use $\alpha = 0.5$ and $\delta = 1 - \alpha = 0.5$, with $X = Y = 0.5$. This strategy is labeled with 'hyp' in Section 4.

The reason why we want to prioritize chunks close to their playback time is that the peers are watching the video while downloading. Chunks that should already have played back are of no worth for the local user. Therefore, a VoD application does not have the luxury of being able to download the chunks of a file in arbitrary order, such as in file-sharing. For a high quality of the shown video, as many chunks as possible have to be available for display. As a consequence, some form of order has to be imposed on the download process if the user does not accept large buffering times.

3.5 System Capabilities

After introducing our model, we can determine the capabilities of our system. The number of sources N_{src} is given by the sum of the number of servers N_{srv}, seeders N_s and leechers N_l:

$$N_{src} = N_{srv} + N_s + \eta \cdot N_l \tag{1}$$

Thereby, leecher can not be considered as mature sources since they have little fragments to share. That is why the contribution of N_l is alleviated by η. This additional factor was first introduced in [17] and indicates the effectiveness of the file sharing. η takes values in $[0, 1]$, where $\eta = 0$ means leechers do never act as source and $\eta = 1$ implies they behave like seeders or servers. According to [17], η can be determined by

$$\eta \approx 1 - \left(\frac{\log N}{N} \right)^k \tag{2}$$

where N is the total number of participants in the system and k indicates the number of connections of a downloader. The last one is given by $k = \min\{x - 1, K\}$, where x is the number of uploaders in the system and K is the maximum number of uploaders to which a peer can connect. In our two scenarios, $k = 10$ and $N = N_{srv} + N_s + N_l = N_{srv} + \lambda(t_s + d_v) = 110$, whereby $\eta \approx 1$. However, for our calculations, we always assume $\eta = 0.9$.

Now, the maximum deliverable video bit rate r_v^{max} is given by the sum of all upload bandwidths bw_u divided by the number of leechers.

$$r_v^{max} = \frac{bw_u \cdot N_{src}}{N_l} \tag{3}$$

Therewith, the system load ρ can be defined as quotient of the bit rate of the shared video used in our system and r_v^{max}.

$$\rho = \frac{r_v}{r_v^{max}} \tag{4}$$

Using Little's theorem, N_s and N_l are given by $\lambda \cdot t_s$ and $\lambda \cdot t_l$, where λ specifies the peer arrival rate and t_s and t_l indicate the seeding/leeching time, respectively. Furthermore, t_l corresponds to the video duration d_v. Hence, the equations 3 and 4 can be transformed into:

$$r_v^{max} = \frac{bw_u \cdot \big(N_{srv} + \lambda \cdot (t_s + \eta \cdot d_v)\big)}{\lambda \cdot d_v} \tag{5}$$

$$\rho = \frac{r_v \cdot \lambda \cdot d_v}{bw_u \cdot \big(N_{srv} + \lambda \cdot (t_s + \eta \cdot d_v)\big)} \tag{6}$$

Applied to our two scenarios high-res and low-res, we achieve a system load of 328.83 % and 81.96 %, respectively.

4 Numerical Results

In this section, we present the numerical results of the simulation runs for the different chunk selection strategies. The parameters introduced in the modeling section result in an overload scenario as the available upload capacity in the entire system is not sufficient to serve all peers such that they receive all blocks of the video stream in time. Thus, some blocks get lost and the video quality decreases accordingly. The resolution of the video, i.e. low-res or high-res, determines the actual system load ρ which is roughly 82 % and 329 %, respectively. The overload scenario allows us to investigate clearly the temporal evolution and the dynamics of the system, in other words, the self-organization of the P2P-based CDN (Section 4.1); in a low or medium loaded system the influence of the chunk selection strategies cannot be seen such clearly as most users will obtain an acceptable quality.

We evaluate the performance of the VoD service from a global point of view by describing the chunk availability and the number of uploads for the individual chunks. We show that due to the deadline of blocks, these performance values for a VoD service differ significantly from file sharing. After that, we consider the performance from the user's point of view. In Section 4.2, we compare the number of blocks played back for the different chunk selection strategies. Subsequently, the peers obtaining video qualities which behave unlike all expectations are analyzed

in Section 4.3. Finally, we compare the peers' upload link utilization to show the weaknesses of the introduced chunk selection strategies.

4.1 Temporal Evolution and System Dynamics

In the following, we consider for each strategy the normalized chunk availability over time and the number of uploads per chunk. The normalized availability A_i of the chunk i is the ratio of the number of servers and peers holding a copy of this chunk and the total number of peers and servers in the system. In our simulations, we investigate the system offering the high-res video for 24 hours, starting with the initially available 20 VoD servers. Thus, in the beginning the normalized chunk availability is 100 % before the first peer enters the system and requests the video. On average, we have around $\lambda(t_s + d_v) = 90$ peers in the system with the peer arrrival rate $\lambda = 0.5\,{}^{1}/_{\min}$, the seeding time $t_s = 60\,\min$, and the video duration $d_v = 129.91\,\min$. Thus, the minimum normalized chunk availability is around 18 %. It is depicted as a black curve in the according figures.

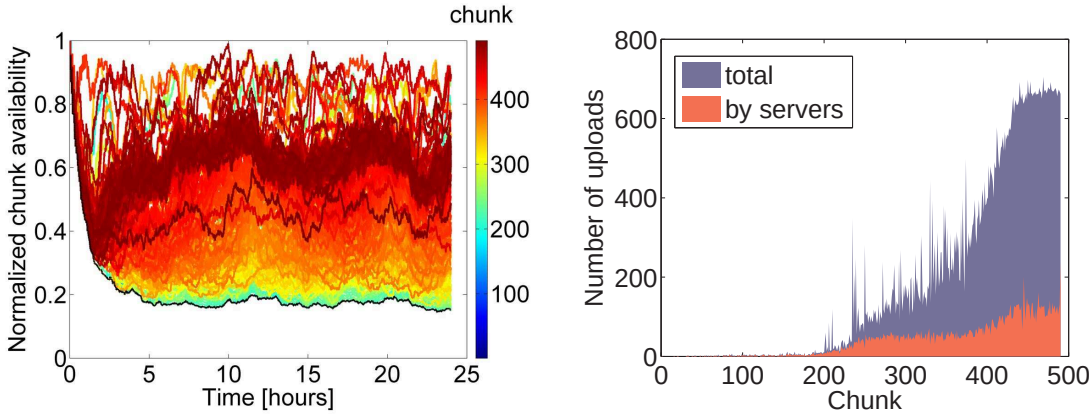
The number of uploads counts the chunks uploaded by peers and servers during the simulation time. These uploads do not necessarily have to be successful and fail if either the uploading peer or the downloading peer goes offline. Consequently, the successful playback of chunks is not considered by this metric.

4.1.1 Random Chunk Selection

Figure 2 shows the performance results for the random chunk selection strategy. The normalized chunk availability over time is depicted for each chunk of the high-res video in Figure 2(a). The chunk id and, hence, its position in the video is encoded by the color of the individual curve for this chunk.

Although the normalized chunk availability A_i varies strongly for the individual chunk indices i from 0.2 to 0.9, which also occurs in file sharing systems, there is a clear correlation between the playback position i and A_i. The availability A_i clearly depends on the position of the chunk in the video. The later the chunk is in the video the higher is its availability. This can also be seen by the number of uploaded chunks (cf. Figure 2(b)). We additionally distinguish between the total number of uploaded chunks, i.e. either by peers or by servers, and the number of chunks only uploaded by servers. This differentiation gives us further insight into the system behavior as the servers share all chunks of the video.

The first 200 chunks are rarely uploaded and thus are less available. This is caused by the high congestion in the system and the lack of peer selection or admission control mechanisms. In our simulations, the peers are served in a first-come-first-serve manner and are waiting in an unlimited queue at the uploading sources, i.e. at peers and servers. The peer is enqueued at the end of this queue, when it enters the system or after being served. During the waiting time, however, the chunk deadlines expire and when being served, one of the remaining useful chunks not yet downloaded is randomly selected for download. The later during the video playout a peer is served the higher is the probability to select a late chunk for downloading, for which the peer will act as source afterwards. This results in this increasing number of uploads for late chunks and the uneven distribution of chunk availability. Taking a closer look at the number of uploads



(a) Chunk availability (b) Number of chunk uploads

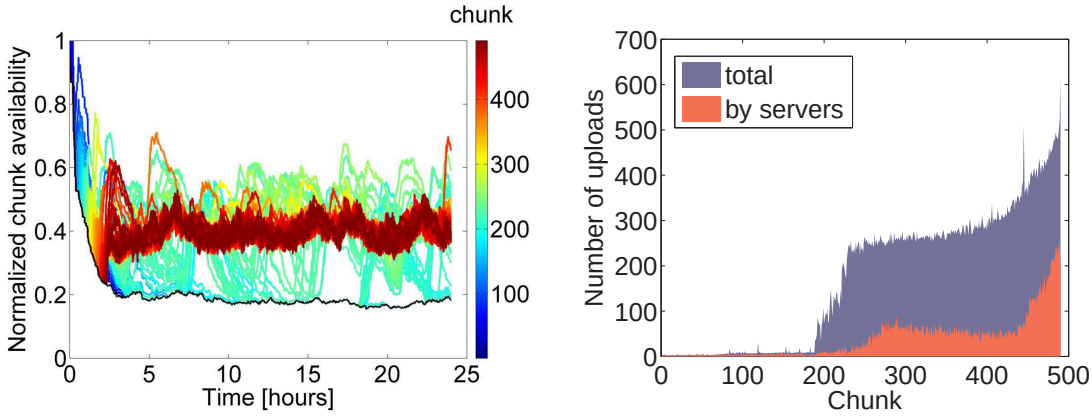
Figure 2: Random chunk selection (’rand’) for high-res video

for the last chunks, we recognize a flattening. This is due to the fact that the system only serves about 700 participants in our 24h consideration window and thus, all peers should have received these last video parts.

4.1.2 Least-Shared First Chunk Selection

The LSF strategy aims at downloading the least-shared chunks of the remaining required, not expired chunks. For this reason, the variance of the chunk availability is decreased in contrast to the random strategy, cf. Figure 3(a). However, again, late chunks in the video are more often uploaded (cf. Figure 3(b)). In turn, the reason for this is the high congestion like stated above for the random chunk selection. We additionally investigated the waiting time of a peer at an uploader, which here averages 60 minutes corresponding to about 230 chunks. Thus, the chunks 1 to 200 are rarely uploaded after which most peers are served for their first times. Since they enqueued at all participants in the system at the same time (their arrival), they are supplied several times in quick succession building their first *serving boom*. Being supplied, they again queue up in the uploaders’ waiting queues. The video takes about 130 minutes. Consequently, many peers are served in a second serving boom at chunk 460 corresponding to the playback time 120 minutes. These two serving booms explain the risings in Figure 3(b) at the corresponding chunk ids. In spite of these serving booms, we observe that the LSF chunk selection leads to a more even distribution of the late chunks than the random selection. This is the main effect of always choosing the rarest chunk first.

Taking a closer look at the region of the first boom, we recognize a shift of this boom between server and peer uploads. This effect is caused by shorter waiting times at uploading peers than at servers. In turn, these shorter waiting times are caused by unsuccessful requests at peers which will thus save the serving time and which will never occur at uploading servers. In an analogous way, the second serving boom by peers is shifted earlier even more compared to the server one and takes place at chunk 350 to 450. However, this effect is straightened by the under-proportional number of uploads by servers at this region. From there on, the second serving

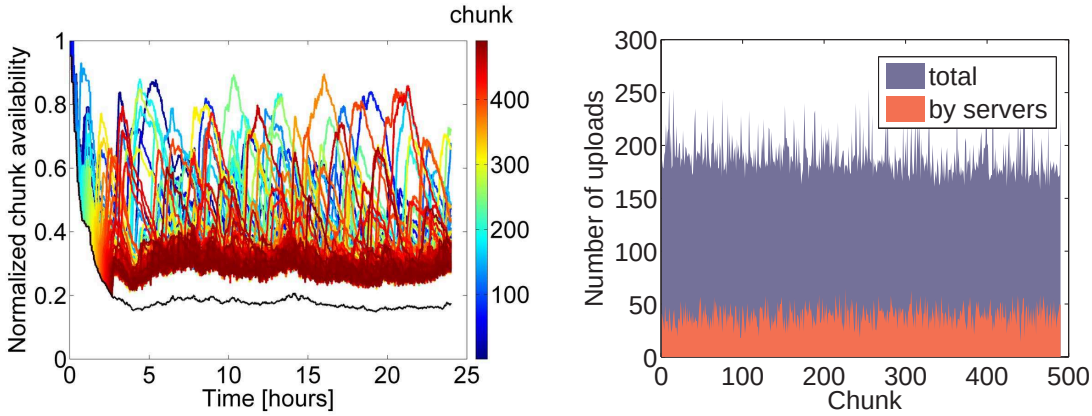


(a) Chunk availability (b) Chunk uploads

Figure 3: Least shared first chunk selection (’lsf’) for high-res video

boom by servers starts which leads to an over-proportional number of uploads by servers at the last chunks. Overall, we observe that the ’lsf’ strategy is capable of handling the high distinctive serving booms by allocating them reasonable among all chunks. This results in an almost even chunk availability distribution at least for the second half.

If we apply the ’lsfwrx’ variant, i.e. peers also download already expired chunks, then this leads to an even chunk distribution with nearly the same chunk availability and number of uploads for all chunks (cf. Figure 4(a) and 4(b)). In the temporal evolution we recognize that after an initial phase of about 3 hours no chunk availability falls down to the minimum but always remains above it (Figure 4(a)). This is the main goal of choosing the rarest chunk first – counteracting the chunk starvation of individual chunks, cf. [16].



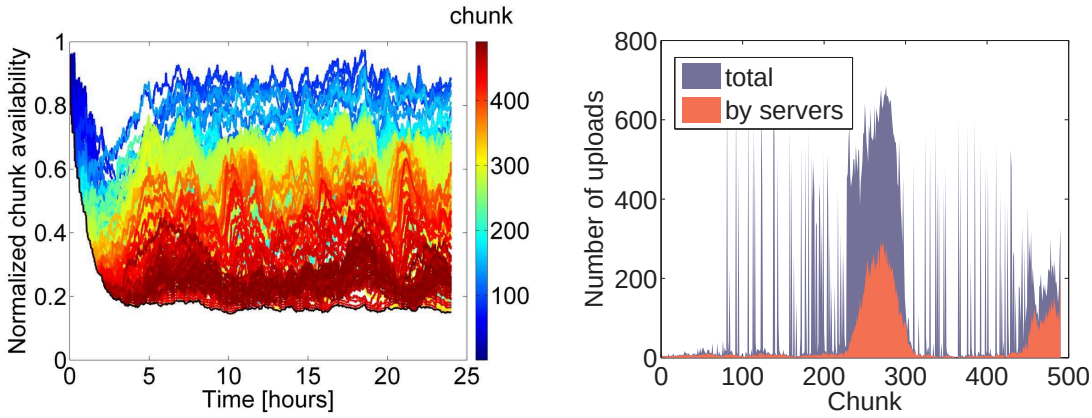
(a) Chunk availability (b) Chunk uploads

Figure 4: Least shared first chunk selection with requesting expired chunks (’lsfwrx’) for high-res video

In Figure 4(a), we observe individual chunk availabilities boom within no time whereupon they slowly decrease back to the mean availability. This behavior can be explained by the following. A given peer P has up to simulation time T no chunk available and just finished receiving chunk C. Until T, P was not able to serve any other participant, but as from now on, P can share C. Whether or not an arbitrary downloader at P needs C and irrespective of its availability, P always serves C until P receives another chunk to share. From then on, P will share the chunk of its available ones which is the rarest in the system. Thus, the availability of C normalizes slowly to the mean chunk availability of 0.3.

4.1.3 In Order Chunk Selection

The in order chunk selection tries to download the chunks sequentially, such that a peer always downloads the next required chunk before its deadline expires. Figure 5 shows the results for this strategy. As newly arriving peers see large waiting queues due to the overload, they request mainly chunks between 200 and 300 from the servers (cp. first serving boom of 'lsf' and cf. Figure 5(b)). Then, they act as source for these chunks resulting in a high availability for these chunks. When a peer is served by a server for the second time, some chunks from the end of the video are requested building the second serving boom. Nevertheless, the probability of being served outside these two regions is greater zero. Therefore, there are several peaks beginning from chunk 100 till chunk 400, as a peer P was downloading this chunk from a server and acts as seed for this chunk for the remaining peers. All peers which arrive after P, will potentially download the chunks from P, if they do not have to wait too long at P.



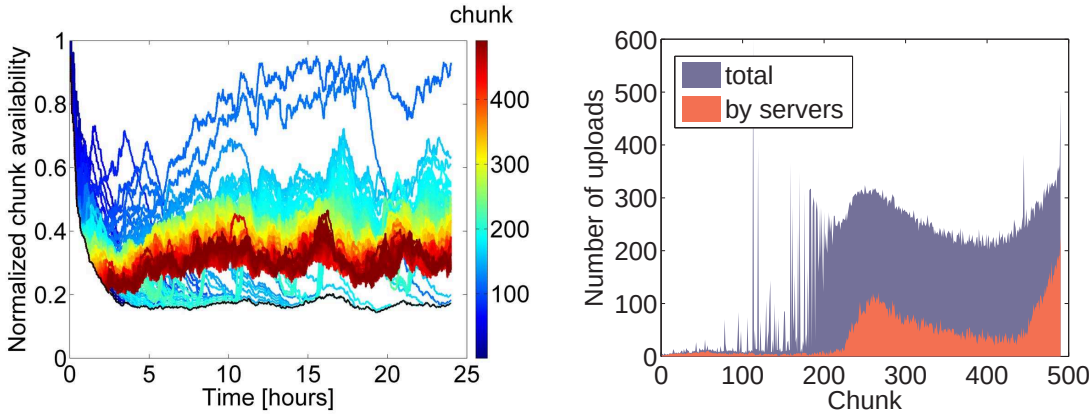
(a) Chunk availability

(b) Chunk uploads

Figure 5: In order chunk selection ('ord') for high-res video

4.1.4 Hyperbola Chunk Selection

The hyperbola chunk selection strategy tries to combine the advantages of the least-shared first strategy as well as the in order strategy. It is intended to obtain a robust system which takes into account the timeliness of block delivery. The availability of chunks and the number of chunk



(a) Chunk availability (b) Chunk uploads

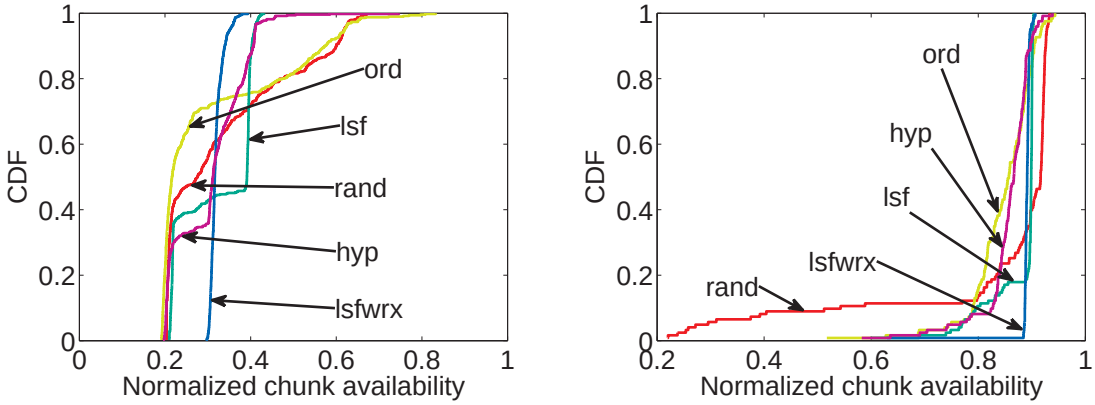
Figure 6: Hyperbola chunk selection (’hyp’) for high-res video

uploads look like a superposition of the corresponding curves for ’lsf’ and in order selection (cf. Figure 6(a) and 6(b)). As for ’lsf’, the chunks from 200 to 500 are evenly distributed and the shifted serving booms of peers and servers are observable as well. The individual shape of the hyperbola curve can be influenced by the shaping parameters α, β, X, and Y of the applied ranking function. As for the in order selection, some of the early chunks in the video are more often shared due to the deadline of the chunks when being served.

4.1.5 Comparison of Chunk Availabilities

Figure 7 shows a comparison of the normalized chunk availability of all chunk selection strategies. For a better overview, we omitted the temporal evolution and computed the temporal average of every chunk availability. These are plotted as cdf for each strategy for the high and low resolution scenario in Figure 7(a) and 7(b), respectively. There are two major goals to stabilize a P2P file-sharing system concerning its chunk availabilities. They should be as high as possible as well as having a small variance to prevent chunk starvation.

As expected and described earlier, the ’lsfwrx’ strategy in Figure 7(a) shows a nearly deterministic chunk availability of 30 %. By contrast, the ’lsf’ variant disperses almost completely into two availabilities of 20 % and 40 % representing the first and the second half of chunk IDs, respectively. Only in between, there is a small ascent which is caused by the higher number of uploads of the last chunks in the video. The random and in order chunk selection strategies behave quite similar regarding their high variances. They only differ in their prioritizing chunk numbers, where ’ord’ has a high availability in the mid range and the last chunk IDs. By contrast, ’rand’ prioritizes increasingly from the middle until the end. Furthermore, the mean chunk availability of ’rand’ is higher than the one of ’ord’ which is due to the fact that ’rand’ has a more uniform distribution of high chunk availabilities than ’ord’. Finally, the hyperbola strategy combining the ’ord’ with the ’lsf’ strategy behaves like expected by composing the particular chunk availabilities. Compared to ’ord’, the variance of ’hyp’ is lower because of the ’lsf’ influence and builds a uniform rising distribution for the high availabilities. However, the low ones



(a) High-res video (b) Low-res video

Figure 7: Normalized chunk availability CDF

still form a deterministic probability of 20 % because of the heavy overload.

For the low-res scenario, the differences between the normalized chunk availabilities are smaller (cf. Figure 7(b)). Like with the high-res video, the availability of 'lsfwrx' is nearly deterministic with a value of 90 %. The strategies 'lsf', 'hyp' and 'ord' behave similary having a small variance and a mean availability of 80 to 90 %. The only difference is that values of in order chunk selection are more uniformly distributed in contrast to the least shared first strategy which splits its availabilities into a uniform and deterministic part. The unifrom part is caused by the high load which leads to long waiting queues at uploaders and thus to a lower availability of chunks in the beginning. As expected, the cdf of 'hyp' again lays between its original strategies. Only 'rand' has a very high variance which is caused by constantly increasing the priority of chunks from the beginning to the end of the video. Hence, chunks in the beginning are hardly downloaded and shared.

For a stable P2P file-sharing system, the chunk availability plays a major role. Of course, these values should be as high as possible but they also should have a small variance to prevent the chunk starvation. This combined objective is best solved by the 'lsfwrx' variant in both load scenarios. All other strategies have the same problem of having low availabilities for several chunks. However, since 'rand' and 'ord' have the highest variances, they poll worst in this consideration.

4.2 Blocks Received on Time

In the previous section, the performance of the system was described with respect to robustness and availability of chunks. To investigate the system behavior, we mainly considered the overload scenario with the high-res video. However, this investigation does not contain any relation to the user perceived quality when watching the video. Therefore, we focus now on the comparison of the blocks played back and also the block loss rate.

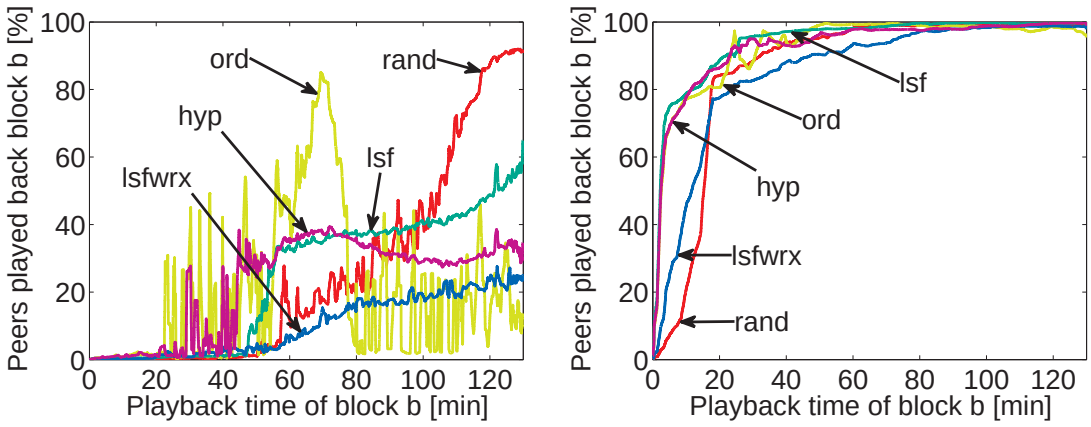
Figure 8 compares the percentage of blocks successfully played back for the different chunk

selection strategies for the high-res and the low-res video. For each block of the video we determined in the simulation whether a peer was able to download and to play back the block before its deadline expired or not. In Figure 8, the playback time of the blocks is on the x-axis while on the y-axis the percentage of peers is given which were able to play back the block. For the sake of readability, we used a moving average over the last 50 blocks.

Figure 8(a) shows the percentage of peers who successfully played back the individual blocks for the high-res video. The shapes of the curves fit to the number of uploads illustrated in the previous section, as a peer only requests a chunk if it is not expired. Only the 'lsfwrx' variant breaks ranks. This strategy gives an equal distribuiton for all chunks, resulting in the approximately same number of uploads for all chunks (cf. Figure 4(b)). However, since the deadlines of chunks are not taken into account, most blocks from the beginning of the video are received too late and cannot be played back (cf. Figure 8(a)). From the comparison we see that the curve for the hyperbola strategy is the superposition of the in order and the 'lsf' curve.

Using 'rand', it should be additionally pointed out that only 90 % of peers successfully played back the last 15 minutes. Although, the corresponding chunks have been uploaded to all peers (cf. Figure 2(b)). This is caused by transmission aborts of leaving peers or due to blocks being transmitted too late.

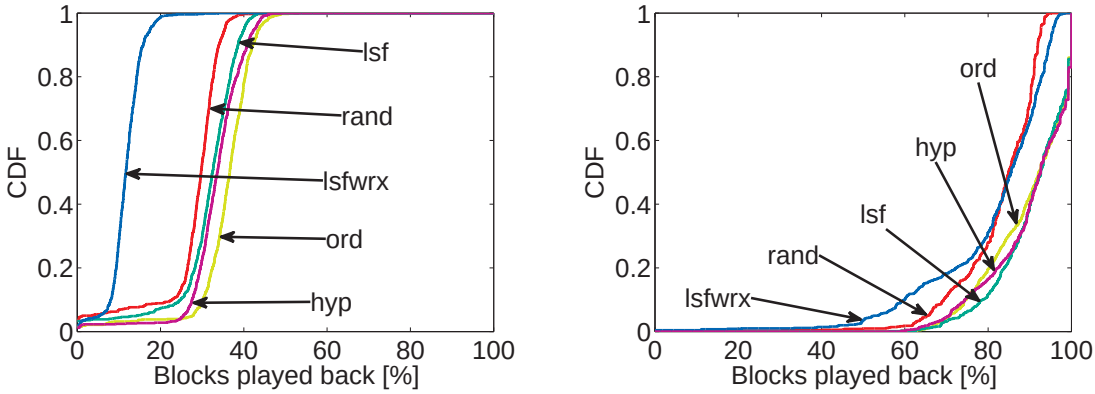
In Figure 8(b), we consider the distribution of the low-res video. In this case, the system is still heavily loaded. However, due to the self-organization capabilities of the P2P-based CDN, the system copes with this high load and to ensure a good video quality after some time. In particular, for all strategies the percentage of peers playing back the blocks after 20 minutes is above 80 %. Before that, the 'rand' and the 'lsfwrx' strategy lead to significantly worse performance results while the performance for 'hyp', 'ord', and 'lsf' are significantly better. Nevertheless, it has to be noted that in all cases it requires a few minutes until the peer receives an acceptable quality. This is, however, not acceptable for the end-user, as he will probably not use or drop the service if the system returns a bad video quality in the beginning.



(a) High-res video

(b) Low-res video

Figure 8: Comparison of blocks successfully played back for the different chunk selection strategies



(a) High-res video (b) Low-res video

Figure 9: CDF of blocks successfully played back

The reason for this can be seen in Figure 9 in which the cumulative distribution function (CDF) of blocks successfully played back is depicted. If a block is not played back on application layer, it is referred to as lost here. In both load scenarios the results show that the number of blocks successfully played back is lowest for 'lsfwrx' and 'rand' which will lead to the worst QoE for the end-user. By contrast, the values for the strategies 'ord', 'lsf' and 'hyp' behave similary heightening the number of blocks successfully played back and thus providing a better video quality.

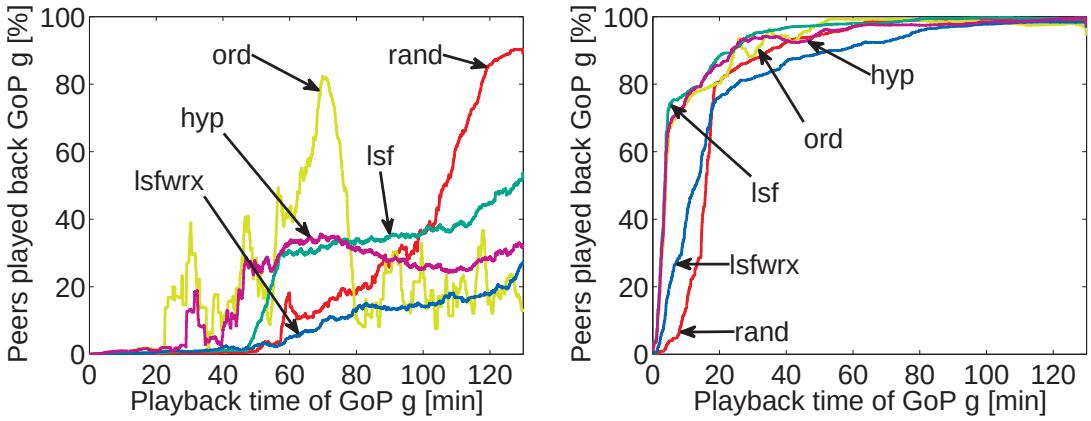
4.3 Video Qualities Obtained by Peers

In the former section, we only compared the data ready transfered in time neglecting which data parts could not be played back due to missing referenced video parts. However, in this section, we take a look at the video qualities obtained by peers. Since we use fixed chunk sizes in our model, block and chunk boundaries do not match GoP boundaries. But we determine the received video quality on a GoP level already introduced in Section 3.1. Hence, a GoP g can be played back if all chunks and blocks containing data of g are available in time. Thus, the QoE of a user is not only affected on block availabilities, but also on the structural relationship of chunks, blocks and GoPs.

Figure 10(a) and Figure 10(b) show the percentage of peers playing back the individual GoPs for the high-res and low-res video, respectively. Again, for the sake of readability, we used a moving average over the last 50 GoPs. Comparing these figures with Figure 8, we hardly see a difference. Hence, the individual behaviors viewed in the graphs are caused by the same reasons.

But regarding the average received video quality, we will recognize a huge difference. We computed the timely averaged video quality based on GoPs successfully played back for every peer. These are combined in a cdf and plotted in Figure 11 for every chunk selection strategy.

According to the observations above, the high-res scenario depicted in Figure 11(a) should show the same behavior as illustrated in Figure 9(a). Based on the order of the depicted curves,

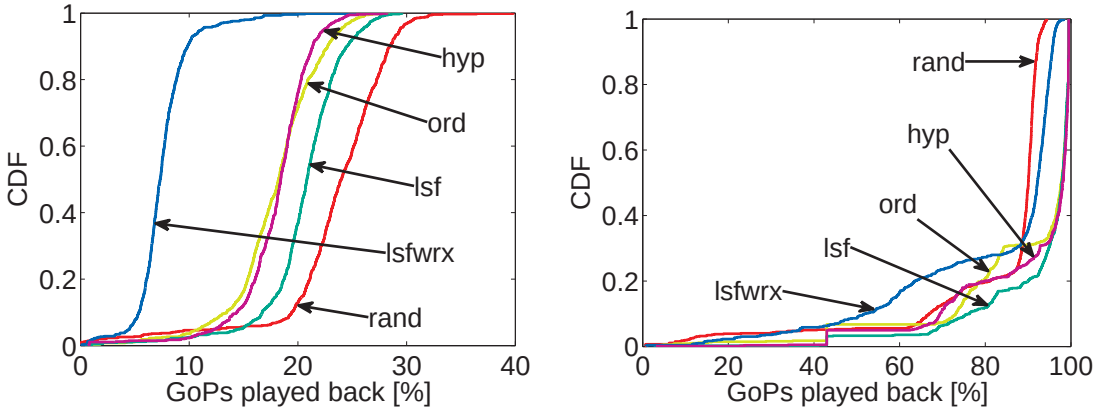


(a) High-res video (b) Low-res video

Figure 10: Comparison of GoPs played back for the different chunk selection strategies

this applies to every strategy but the random one. At first sight, this surprises since 'rand' beats every other introduced chunk selection strategy according to the average received video quality.

To explain this effect, we have to take a closer look at the relationship between chunks and GoPs. Like denoted in Table 1, a chunk of the high-res video contains about 3.62 GoPs. Imagine three successive chunks A, B and C each containing 3.62 GoPs whereby all three chunks are available but B. According to the GoP boundaries, A and C can each play back at least two GoPs, which would lead to an average QoE of $\frac{2+2}{3 \cdot 3.62} = 36.8\,\%$. If now additionaly B is available, the last and first truncated GoP of A and C respectively can be played back too. This will result in a over-proportional gain of $100\,\%$ and the QoE will amount now $\frac{2+4+2}{3 \cdot 3.62} = 73.7\,\%$. Hence, the timeliness of chunk delivery plays a major role to the QoE of users.



(a) High-res video (b) Low-res video

Figure 11: CDF of GoPs successfully played back

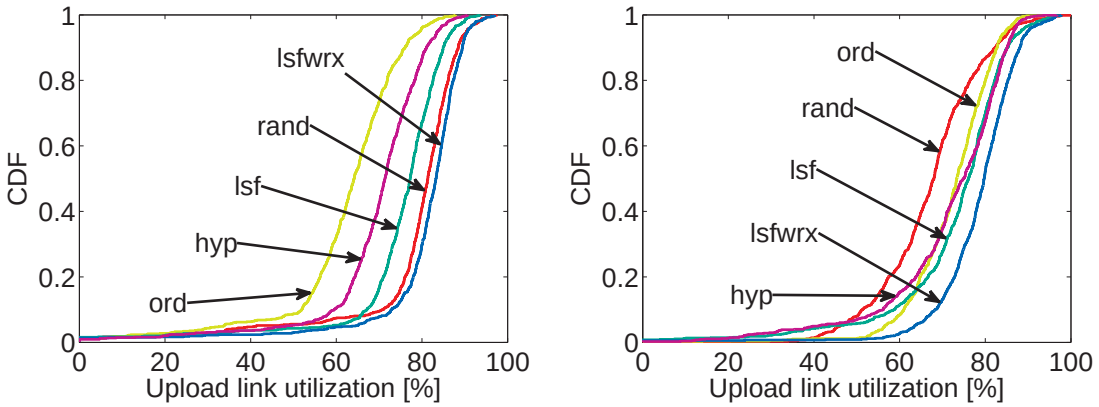
Applying this consideration to Figure 10(a), we recognize the long-lasting quality of 'rand' at the end of the video. No other chunk selection strategy persists in quality in such a way without fall-offs. All other strategies heavily vary in their quality traces which lowers their QoE due to the bad timeliness of chunks. That is why they are excelled by the random chunk selection strategy in the average received video quality.

For smaller video resolutions like the low-res one, the timeliness of chunks does not fall into account to such an extent. Here, the number of GoPs per chunk amounts 14.52 (cp. Table 1) whereby the truncated GoPs at the chunk boundaries carry no weight (cf. Figure 11(b)). Since the different curves lay close together and each one forms a single simulation run, no further clear statement can be given for the low-res scenario.

The QoE of a user does not only consist of its received video quality, but also of its waiting time starting by choosing the video and lasting until it starts to play back. The given maximum of 1 minute did not suffice for any of the presented chunk selection strategies for neither the high-res nor the low-res scenario. But regarding the obtained qualities by peers in Figure 10, it is noticeable that 'lsf', 'ord' and 'hyp' excel 'rand' and 'lsfwrx' in this discipline.

4.4 Uplink Utilization of Peers

Finally, we want to investigate the sharing behavior of peers. Figure 12 shows the CDF upload link utilization of the peers in the system for the different strategies and load scenarios. We can see that in both Figures 12(a) and 12(b) the uplink utilization is beyond 100 %, although the system load is high. Since no peer selection mechanism is applied the following situation might happen. A peer P offers two chunks A and B, one chunk A is shared by many others, while for the second chunk B there are only a few sources available. If a peer now is served by P and selects chunk A, although this chunk could be easily uploaded by another peer, then system resources are wasted and the entire content distribution process is harmed. From this we can conclude that a proper cooperation strategy for VoD has to take into account the chunk selection as well as the peer selection to provide a good service quality.



(a) High-res video (b) Low-res video

Figure 12: Upload link utilization

5 Lessons Learned and Future Work

In this section, we want to sum up the most important issues we experienced in evaluating a P2P VoD system. We will provide insight into the specific problems occurring in this context, and how we plan to address them. One of the first observations made during the simulations is that the behavior of cooperation strategies that are well investigated for file-sharing networks changes significantly when applied to a VoD system. An example would be the pattern of the chunk availability, even for the random strategy, that evolves due to the play back deadline moving forward in the video during the download. Therefore, these mechanisms have to be re-evaluated for VoD systems instead of simply using older results.

Another observation is that an efficient peer selection process is needed for a high-quality VoD service. A strategy simply aiming at increasing the number of potential sources, such as the one used in this paper, leads to long upload queues and waiting times of the peers. Also, peers slow each other down by using upload resources inefficiently from the complete system's point of view. To illustrate this, envision a peer that offers just the first chunk of the video, and one seeder having the complete file. If the peer is pushed back in the upload queue of the seeder, other peers will download chunk 1 from the seeder, since they also need it first. However, this means that the seeder will not be able to spread copies of the other chunks.

In case of high load or even overload, the video quality is too low for most of the users to keep them satisfied with the service. This can be remedied by different approaches. To lower the needed upload capacity of the system, user admission control to the system can be established, or the video quality can be lowered. Alternatively, the upload capacity can be increased by adding more servers to the system, or by offering incentives to users to make more of their resources available for seeding the video. Most of these mechanisms have a negative effect on either the users' satisfaction, or on the providers cost.

In our system, we ignored the possibility to retransmit blocks, i.e., when an uploading peer goes offline, the chunks it was currently uploading are lost to the downloaders. This means that possibly several GoPs can not be played back, and the resources up-/downloading the successfully transmitted blocks are wasted from the system's perspective, since these blocks cannot be shared. Therefore, we see the need for the retransmission of single blocks instead of only whole chunks.

The fact that, using fixed chunk sizes, block and chunk boundaries do not match GoP boundaries has a large impact on the QoE of a user. GoPs that are separated by a chunk boundary can only be played back if *both* chunks are downloaded in time. For large videos, where only a few GoPs are included in one chunk, this has a larger effect than for smaller videos. Moreover, statistics about lost and played out blocks do not capture the actual quality of the video. Due to the heterogeneous video structure, the actual number and types of frames that could be played back have to be determined for each video in detail. Simple mapping functions between QoS and QoE are needed here. An exception is the user's waiting time until the video starts playing, which is a QoE measure that can be easily metered.

As a consequence, we have identified the fields where we invest our future work on this topic. As a first step, we will extend our evaluation on peer selection mechanisms to cover the complete cooperation strategy of a peer. To be able to investigate user behavior and satisfaction, we also want to create a mapping between QoS parameters and end user QoE. Then, we can include

scenarios where, e.g., a user stops watching a video because its quality was too low for a certain time. This has already been done for VoIP in [18].

Also, we want to extend the video model used in this work to integrate more complex video structures, namely layered video codecs. This would allow for an investigation into quality control based on overlay performance monitoring and will be done in the G-Lab Project [19]. An additional aspect fully neglected in this work is the effect of the generated overlay traffic on the physical network. Strategies to adapt the overlay structure to ease the physical network load and to increase the efficiency of the service while lowering the costs for ISPs are investigated in this context in the SmoothIT Project [20].

Acknowledgements

This work has been performed partially in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259).

References

[1] BitTorrent, Inc., "Home Page." `http://www.bittorrent.com/`.

[2] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer Systems*, May 2003.

[3] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. of INFOCOM 2005*, vol. 3, pp. 2102–2111, 13–17 March 2005.

[4] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *Proc. of INFOCOM 2008*, pp. 1031–1039, 13–18 April 2008.

[5] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. of the 12th NOSSDAV*, 2002.

[6] S. Tewari and L. Kleinrock, "Analytical model for bittorrent-based live video streaming," in *Proc. of CCNC 2007*, pp. 976–980, 2007.

[7] C. Dana, D. Li, D. Harrison, and C.-N. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *Proc. IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4, Oct. 2005.

[8] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving vod server efficiency with bittorrent," in *Proc. of MULTIMEDIA '07*, pp. 117–126, ACM, 2007.

[9] D. Erman, "Extending bittorrent for streaming applications," in *4th Euro-FGI Workshop on "New Trends in Modelling, Quantitative Methods and Measurements"*, 2007.

[10] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. R. Rodriguez, "Is high-quality vod feasible using p2p swarming?," in *Proc. of WWW '07*, pp. 903–912, ACM, 2007.

[11] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *Proc. of INFOCOM 2006*, pp. 1–6, 2006.

[12] P. Garbacki, D. H. J. Epema, J. Pouwelse, and M. van Steen, "Offloading servers with collaborative video on demand," in *Proc. of IPTPS'08*, February 2008.

[13] J. Lv, X. Cheng, Q. Jiang, J. Ye, T. Zhang, I. Lin, and L. Wang, "Livebt: Providing video-on-demand streaming service over bittorrent systems," in *Proc. of PDCAT '07*, pp. 501–508, 2007.

[14] C. Huang, J. Li, and K. Ross, "Peer-assisted vod: Making internet video distribution cheap," in *Proceedings of Sixth Internaltional Workshop on Peer-to-Peer Systems*, 2007.

[15] V. Janardhan and H. Schulzrinne, "Peer assisted vod for set-top box based ip network," in *Proc. of P2P-TV '07*, pp. 335–339, ACM, 2007.

[16] D. Schlosser, T. Hoßfeld, and K. Tutschku, "Comparison of robust cooperation strategies for p2p content distribution networks with multiple source download," in *Proc. of P2P*, 2006.

[17] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like p2p networks," in *SIGCOMM Comput. Commun. Rev.*, vol. 34, (New York, NY, USA), pp. 367–378, ACM, Aug-Sep 2004.

[18] T. Hoßfeld, D. Hock, P. Tran-Gia, K. Tutschku, and M. Fiedler, "Testing the IQX hypothesis for exponential interdependency between qos and qoe of voice codecs iLBC and g.711," in *Proc. of 18th ITC Specialist Seminar on Quality of Experience*, 2008.

[19] The GLab project, "www.german-lab.de," 2008.

[20] The SmoothIT project, "www.smoothit.org," 2008.