Comparison of Robust Cooperation Strategies for P2P Content Distribution Networks with Multiple Source Download

Daniel Schlosser[†], Tobias Hoßfeld[†], Kurt Tutschku[†,‡]
[†]Institute of Computer Science, University of Würzburg, Würzburg, Germany.
{schlosser,hossfeld,tutschku}@informatik.uni-wuerzburg.de
[‡]Faculty of Computer Science, TU Chemnitz, Chemnitz, Germany.
tutschku@informatik.tu-chemnitz.de

Abstract

The performance of Peer-to-Peer (P2P) content distribution networks depends highly on the coordination of the peers. Sophisticated cooperation strategies, such as the multiple source download, are the foundation for efficient file exchange. The detailed performance of the strategies are determined by the peer characteristics and the peer behaviour, such as the number of parallel upload connections, the selfishness, or the altruistic re-distribution of data.

The purpose of this work is to evaluate and investigate different cooperation strategies for multiple source download and select the best one for a scenario for even leeching peers, i.e. peers which depart as soon as they have finished their download. The question arises whether the cooperation strategy can smoothen the overall performance degradation caused by a selfish peer behaviour. As performance indicator the evolution of the numbers of copies of a chunk and the experienced download times of files is applied. The considered scenarios comprise best-case (altruistic peers) and worst-case scenarios (selfish peers). We further propose a new cooperation strategy to improve the file transfer even when mainly selfish peers are present, the CygPriM (cyclic priority masking) strategy. The strategy allows an efficient P2P based content distribution using ordered chunk delivery with only local information available at a peer.

1 Introduction

The mechanisms of P2P content distribution networks can be distinguished in two major categories: *resource mediation* mechanisms, which are functions for searching and locating resources and *resource access control* mechanisms, i.e. function for exchanging files or parts of it.

The *resource access control* is typically implemented using a *cooperation strategy* among the peers. An efficient and robust way of cooperative content delivery is the *multiple source download (MSD)*, which means that the recipient peer orders and downloads the desired data from many providing peers instead from a single one. The efficiency of MSD was demonstrated by the success of the P2P files sharing platforms eDonkey and BitTorrent and was scientifically researched for example in [1, 2].

The coordination of peers in a P2P file sharing network is no simple task. An inappropriate coordination of the peers may decrease the performance of the strategy, i.e. it may increase the overall download time of a file. A particular problem in P2P file sharing networks is the so-called ”last chunk” or ”starving chunk” problem [3]. Here, a single chunk of a file may not spread in the file sharing network as the other chunks do. Hence, a shortage of providers for this chunk may arise. As a result, the remaining providing peers may be overloaded and the file exchange is delayed.

There are many attempts to overcome this problem like the least-shared-first chunk selection of BitTorrent [4] or Avalanche network coding [5]. In this paper we propose a new cooperation strategy, which has an efficient chunk diffusion behaviour and unlike other strategies it is based only on existing local information available at the peer.

In order to compare the different approaches of today’s P2P content distribution platforms, we propose a unified system model. We focus on the cooperation strategies and therefore do not address features like grouping of peers or incentive mechanisms. We use the evolution of the number of copies of a chunk and the experienced download times of files as a measure of the performance of the distribution strategy, because it reflects the key performance characteristic from the user’s point of view.

The paper is organized as follows. Section 2 formulates the problem in detail. Other approaches to fight the ”last chunk” problem are discussed in Section 3. Section 4 describes the three chunk distribution strategies considered in this work. Section 5 outlines the proposed system model, its parameters, and the considered distribution scenarios. Section 6 compares the performance of the considered cooperation strategies. Finally, Section 7 concludes this work and gives a brief outlook on future work.

2 Problem Formulation

P2P content distribution mechanisms which apply MSD split files into chunks and blocks which are subparts of chunks. For the eDonkey application for example, the chunk size is typically 9.5 MB and the block size is 180 kB. A downloading peer requests blocks from serving peers, i.e. sources of that file, and might download from these sources in parallel. As soon as a peer has downloaded a complete chunk, it becomes a source for the file, i.e. it can redistribute the already received chunks. As a result, MSD does not rely on a single source and can therefore avoid bottlenecks and overcome churn. From user perspective a MSD mechanism can be evaluated in terms of the time needed to download a file and from global point of view in terms of availability, i.e. the number of peers sharing a certain chunk.

The coordination of the peers to enable the MSD is realized by the *cooperation strategy*. Its task is to decide *a)* which peers requesting for blocks are served by an uploading peer using a priority function and *b)* which is the next chunk to download by a downloading peer, the so-called chunk selection strategy.

The performance of the P2P file-sharing application is determined by the implementation of this cooperation strategy, the *peer characteristics*, i.e. the available capacity resources for exchanging files (upload and download bandwidth, maximum number of inbound and outbound connections), and the *user behaviour*. The latter one includes the file request pattern, the churn behaviour, and the willingness to participate in the network, i.e. selfish or altruistic peers. As an extreme case of selfish peers we consider *leechers* which immediately leave the system after finishing the download of a file. In such a case, the leaving users reduce the availability of chunks. As a result, in the worst-case a specific chunk may get rarely in the system. Figure 1 depicts such a behaviour. It shows the spreading of chunks, i.e. the number of sharing peers for each chunk i of the file, over time in a leeching scenario. Most of the chunks are spreading throughout the system, cf. label 'popular chunks' in Figure 1. One of the chunks denoted as 'starving chunk' will not spread due to the leeching behaviour of peers, as the downloading peer disappears from the system as soon as it has downloaded this chunk. The only remaining sources of this chunk are the initial seeds. As a result, unfinished peers which seek the final, last chunk have to wait until they receive the chunk from one of the initial sources. This leads to large download times. This problem is called in this paper the *last chunk problem*.

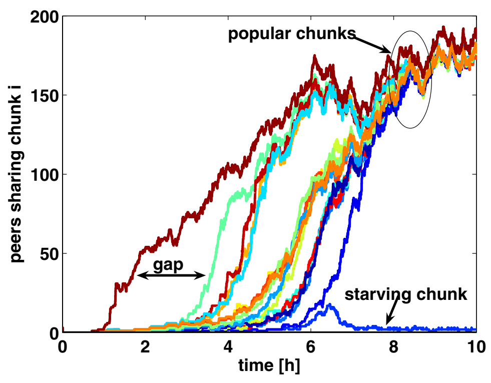

Figure 1. Illustration of the last chunk problem (least-shared-first strategy with four upload connections in the leeching scenario)

The question arises whether a cooperation strategy can leverage the selfish behaviour of the peers and avoids the last chunk problem. Within this work, different cooperation strategies are evaluated with respect to the last chunk problem by their download performance and their spreading behaviour of the chunks.

3 Related Work

One of the possibilities to overcome the last chunk problem are incentive mechanisms which are based on altruistic behaviour of peers. Examples are credit point systems as used in eDonkey or tit-for-tat strategies like in BitTorrent. However, in this work we try a different approach by changing the cooperation strategy in such a way that all chunks of a file are more or less homogeneously distributed to challenge the last chunk problem and to maximize availability and download speed.

Only few scientific works investigate the impact of different cooperation strategies to deal with the problem formulated above. Massoulie et al. [6] propose a coupon replication system and consider scenarios comparable to the ones presented here, cf. Section 5.2. However, churn is not taken into account. Their result suggests that the performance of file-swarming systems does not depend critically on the user behaviour or on the cooperation strategy.

In [7], the authors focus on a random and a rarest-first chunk selection strategy. However, the authors do not investigate the churn behaviour of peers and different numbers of simultaneously served peers.

In previous works, we studied mobile P2P file-sharing networks based on an eDonkey and proposed an enhanced architecture concept [8] leading to a higher performance [9]. In [10], the size of the chunks and blocks was optimized for a mobile environment to minimize the required download times. In this work we use the mobile P2P file-sharing scenario and its parameter, cf. Section 5.2, as a case study to see how the cooperation strategy can further improve the performance. The mobile P2P file-sharing scenario is of particular interest to investigate the robustness of a system, due to the increased churn behaviour of peers and the poor connectivity of the peers which may increase the selfishness of peers.

4 Cooperation Strategies

The cooperation strategy describes the selection of the next peer being served as well as the choice which chunk should be transferred, i.e. the priority function and the chunk selection strategy. Both decisions have direct impact onto the last chunk problem. Therefore three possible strategies are described and their influence on the chunk dissemination is discussed: the eDonkey-like random chunk strategy, the BitTorrent-like least-shared-first strategy, and the proposed *cyclic priority masking (CygPriM)* strategy. Figure 2 shows the first two steps of the distribution process of a file for the three different cooperation strategies. It shows three initial sources $(S1, S2, S3)$ which share all chunks of a file. In the example the file consists of two chunks. Furthermore there are ten leeching peers $(P0, \cdots, P9)$ who want to download the file. The first step of the download process is equal for all strategies: peer P0 downloads chunk 1 from S1, peer P1 chunk 2 from S2, and peer P9 chunk 1 from S3. After the first round of transferring chunks, the dissemination behaves different for each strategy. The strategies will be explained the following.

4.1 Random Chunk Strategy

Applying the random chunk strategy, like the one used by eDonkey, a downloading peer issues a request to a sharing peer. The sharing peer queues this request in a first-come-first-serve (FCFS) manner. As soon as the downloading peer is served, it chooses a random chunk which it has not downloaded yet. In the example of Figure 2 (case 'random') peer P0 selects its missing part and departures after downloading from the network. In addition, peer P2 and peer P4 choose chunk 1 randomly and independently and download it from S1 and S3, respectively.

The *random chunk strategy* relies on the random selection of required chunks. The randomization avoids that all downloading peers select the same chunk. Thus, the simultaneously downloading peers get different chunks and can therefore exchange these different chunks in the further distribution process. This fosters the cooperation among peers.

As will be shown in Section 6.2.1 this strategy performs well as long as peers are altruistic, i.e. as long as peers are willing to share after they have completed their download.

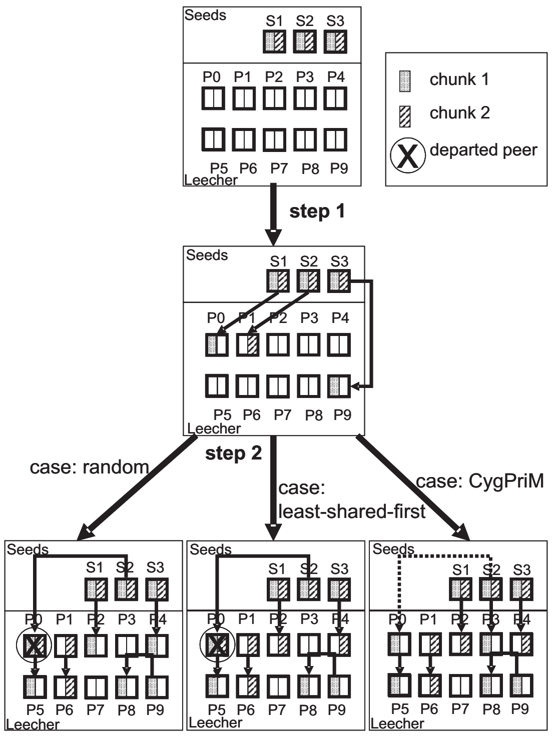

Figure 2. Example of chunk exchange for the cooperation strategies in the first two steps

However, if most of the peers are leeching, the random selection can not guarantee an even distribution of the chunks. This leads to the situation of one chunk being less shared than the others and the last chunk problem occurs.

4.2 Least-Shared-First Strategy

The least-shared-first strategy also uses the same priority function like the random chunk strategy, i.e. requests are served in a first-come-first-serve manner. However, the chunk selection differs. Peers choose as next chunk to be downloaded the one which is *least-shared* in the P2P network. This means that this chunk has the smallest number of sharing peers, compared to the number of possible sources for other chunks. If there are several chunks fulfilling the least-shared criteria one of these is chosen randomly. In order to know the least-shared chunk, it is necessary that every peer is aware of the numbers of peers sharing a specific chunk. Figure 2 (case 'least-shared-first') shows the selection for the given example. Peers choose the least-shared chunk not yet being downloaded at the moment of the download.

A peer using this strategy selects the required chunk which has the lowest number of providing peers. This mechanism results typically in a evenly spread number of sharing peers for all chunks of the file. However, there are cases in which this is not true. As it will be shown in Section 6 this strategy is very efficient as long as the chosen chunk is the least-shared one at the end of the download of this chunk. However, the decision which chunk is the currently least-shared one is done at the beginning of the download. Thus, another chunk can get the least-shared one which undermines the homogeneous chunk dissemination.

4.3 CygPriM Strategy

A sharing peer should take care of the homogoneous chunk dissemination in the network to avoid the last chunk problem. If there is only local information available, the sharing peer should deliver chunks in an ordered way.

The basic idea is to distribute the file in upload rounds. In each upload round the mechanisms tries to distribute a sequence of all chunks to requesting peers, as long as requests for these chunks are available. If no request is available for one of the chunks in the sequence, this chunk is skipped and the next chunk of the sequence is chosen to be distributed. After the complete sequence is processed, a new upload round starts. In order to prevent the downloading peer from selecting any other chunk, we propose the following chunk selection strategy: The uploading peer offers only this one chunk. If a peer accepts this offer, or no peer wants the chunk, then the next chunk from the cycle is chosen. We call this strategy *CygPriM* (case 'CygPriM' in Figure 2) which stands for cyclic priority masking. In Figure 2 the example shows, that all seeds share the opposite chunk in the second transfer phase, as they did in the first. For example seed S2 has transferred chunk 2 in step 1, thus S2 will distribute chunk 1 in step 2. Peer P0 would be served by seed S2 in the second step. However, peer P0 has been masked because it already has chunk 1. The next peer wanting to download chunk 1 (in the example: peer P3) is served instead. As it will be shown in Section 6.2.3 this strategy is only a little bit slower than the least-shared-first strategy. But it neither needs additional signaling traffic nor has the last chunk problem.

5 Simulation Description

5.1 System and Queueing Model

In this section the system model is introduced. We assume a hybrid P2P file sharing architecture. That means, the resource mediation is offered by a central entity and the resource access control is distributed among the peers.

In our model we assume that a peer, which is interested in a file, requests all available sources at an index server. Therefore each peer knows all sources, which are connected to the network at the moment of the request. New sources will be discovered by a periodical source request messages of a downloading peer, which are sent every ten minutes. However, the signaling traffic is neglected in the system model, i.e., it has no impact on the available resources and requires no transmission time.

The resource access control in the model is represented by the queueing model, which is assumed to be equal for all the peers in the network. In order to share a file it is divided into chunks. In [10] it was shown, that further transmission fragmentation of chunks into blocks is not necessary and can even worsen the download time. Thus the model assumes that peers try to transfer complete chunks. Nevertheless chunk fragmentation is possible due to a connection loss between two peers caused by the churn behaviour. In this case the received data is stored and the chunk will be finalized without any retransmission of already received data.

The model assumes that every time a peer receives a new source, it sends a download request containing an identifier for all required chunks. If the peer addressed by the request has none of the required chunks, the request is neglected. Otherwise the serving peer queues the request in the waiting queue Q_W. This queue is ordered to some priority function, as described in Section 4. We assume all peers to use the same priority function. Requests in Q_W are served by a set of at most Q_A upload slots . Requesting peers being served in parallel share the uploading bandwidth of the providing peer. The limitation to Q_A upload slots guarantees a minimum amount of bandwidth per P2P connection, which is at least $\frac{1}{Q_A}$ of the providing peers upload bandwidth. However, if a downloading peer is not able handle the offered bandwidth due to restrictions of his own download bandwidth, the surplus is equally divided among the other peer connections. After a download is completed, the providing peer verifies if there are any other chunks requested, which he actually shares. In this case the request is re-queued. Otherwise the request is discarded.

A major influence factor in the system modell is the number of parallel uploads. Reducing the number of parallel uploads to one, which means no parallel uploads at all, could possibly enhance the diffusion. This is reasonable by considering a scenario in which only one source exists at time t_0 which provides a file consisting of one chunk. All peers are assumed to have the same upload bandwidth, which allows them to upload one chunk within Δt using the complete bandwidth. Thus the number of peers sharing the file is $2^{\Delta t}$ in the case of one upload and $k^{\frac{\Delta t}{k}}$ in the case of k parallel uploads. It holds $k^{\frac{\Delta t}{k}} \leq 2^{\Delta t}$ for $k > 0$, i.e. an outbound degree of one performs best in theory. However, selfish behaviour and churn might lead to other results in practice, which will be discussed in Section 6.

5.2 Scenario Description

The performance of the cooperation strategies is evaluated for different scenarios considering the user behaviour and the peer characteristics. In all scenarios we assume churn. The duration of an ON and an OFF period is exponentially distributed with a mean of 1 hour, respectively.

The selfishness of peers is investigated in a worst-case scenario, the *leeching scenario*, and a best-case scenario, the *diffusion scenario*. In the diffusion scenario all peers finishing the file transfer will serve as uploading peers during the rest of the simulation. From the diffusion scenario it can be concluded, whether a strategy uses the available resources efficiently or not. Against this, a peer finishing the download will depart from the net shortly after in the second scenario, which is called the *leeching scenario*.The leeching scenario will demonstrate if a strategy can deal with uncooperative peer behaviour.

Another influence factor on the system performance is the peer characteristic. In all scenarios peers are assumed to have the same bandwidth characteristics. We consider the mobile P2P file-sharing scenario, as motivated in Section 3, cf. [9, 10]. The peers have GPRS access with an upload bandwidth of 12 kbps and a download bandwidth of 48 kbps. The maximum number of outbound connections, i.e. parallel uploads of a peers, might strongly impact the system and is varied between one and four connections.

For the numerical evaluation we simulated ten runs for each scenario and each cooperation strategy. The considered network consists of 1000 peers. These peers are interested in one file which is provided by three initial seeds. The file has a size of 8 MB and consists of 17 chunks with a maximal chunk size of 480 kB. One of the peers which yet has not downloaded the file starts requesting chunks of the file. The time between two consecutive file requests follows an exponential distribution with a mean of 80 seconds. The simulation stops after all peers have successfully downloaded the file.

6 Numerical Results

Next we investigate the cooperation strategies in a diffusion and in a leeching scenario.

6.1 Performance in the Diffusion Scenario

The system performance in the diffusion scenario is expected to be rather good for all cooperation strategies, since all peers stay in the network and offer their chunks to be downloaded. Thus, the number of available sources should grow rapidly and provide short download times.

In Figure 3, the complementary cumulative distribution function (CCDF) of the required time for downloading the file is given using a single upload connection. The least-shared-first strategy and the random chunk strategy show a slightly better performance than the CygPriM strategy. However, the differences are marginal. The 95% confidence intervals are also given in Figure 3 and are small enough. For the sake of clarity, we and will no longer plot confidence intervals in the following figures.

Figure 4 shows the CCDF of the download time with at most four parallel upload connections. In this case, the achieved performance of the random chunk strategy is slightly worse. However, as expected the download times show the same behaviour and lie in the same order of magnitude. As a first result, it can be stated that in the *diffusion scenario* the cooperation strategy and the peer characteristics in terms of outbound degree have no significant impact on the download time. An explanation for this phenomenon is that the peers exhibit an altruistic behaviour and are willing to contribute to the content distribution network.

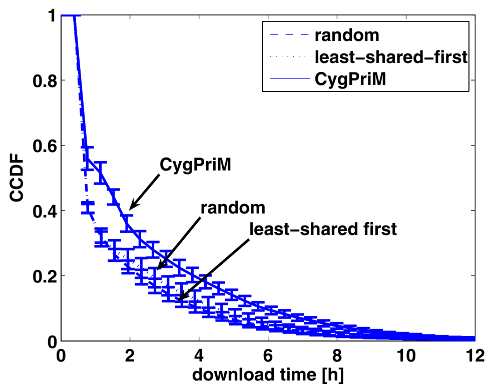

Figure 3. Download time using one upload connection in diffusion scenario

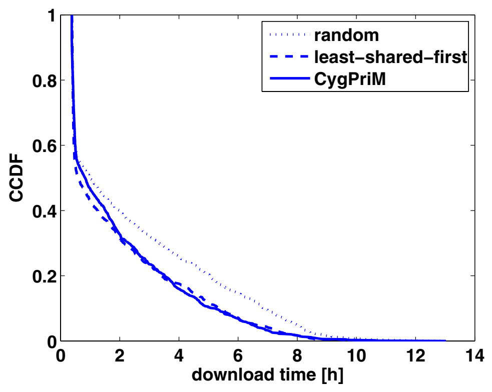

Figure 4. Download time using four upload connections in diffusion scenario

6.2 Performance in the Leeching Scenario

The last chunk problem is assumed to be mainly caused by the selfish behaviour of peers. We therefore continue to investigate the impact of the peer behaviour to analyze the last chunk problem. In the leeching scenario, a peer leaves the system immediately after having downloaded the complete file. In this context, the cooperation strategy is expected to heavily affect the chunk dissemination. Thus, we examine the evolution of the number of peers sharing a chunk for the different cooperation strategies in detail. The performance of the different strategies is investigated for the *leeching scenario* in the following.

6.2.1 Random Chunk Strategy

The first strategy analyzed closely is the random selection strategy, which is used in the eDonkey system. Figure 5 depicts the number of peers sharing a chunk at a specific time using at most one uplink connection. Every chunk of the file is represented by one line in Figure 5. It is evident that the number of peers sharing a chunk does not rise equally. Every chunk that reaches a high popularity, has a massive increase of sharing peers at the beginning of its distribution. After the first peer downloads a chunk, it shares this chunk to the community. The other peers can now discover this new source and may enqueue themselves in the waiting list of the new source. A peer that downloads from this new source can not select which chunk it wants to download. It has to download this only available chunk. Thus, this chunk will be more often downloaded than the other ones.

After the first chunk was distributed among the requesting peers, a peer downloads another chunk and this chunk spreads in the network like the first one. Later this leads to a situation in which nearly all chunks are often shared.

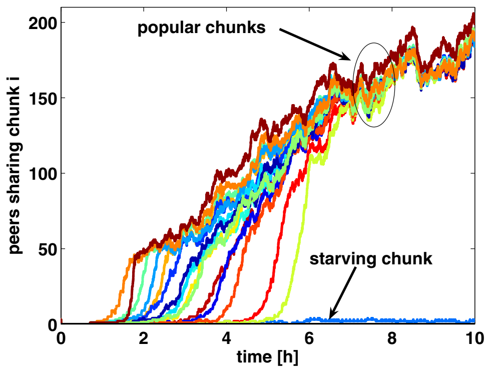

Figure 5. Number of peers sharing a chunk for the leeching scenario using the random strategy with no parallel data connections[1]

This is the point where the leeching behaviour harms the system. If a peer finally downloads this 'starving chunk' it has the 'popular chunks' already, or will get them very fast because of the high number of peers sharing the popular chunks. In both situations it will leave the network shortly after the 'starving chunk' was downloaded. Later on, the queues of the initial sources will contain so many peers that every peer reaching the top position will most likely have the other chunks. As a result, one chunk is shared only by a few peers and required by many peers which forces them to wait for this final chunk to be transferred. In Figure 5 this least shared chunk is referred to as 'starving chunk'.

The random strategy using four parallel upload slots shows the same development of shared chunks and therefore is not shown here; all results can be found in technical report [11]. However, the upload bandwidth of a peer is now equally split among the served peers which leads to a slower reproduction of chunks. Hence, some leeching peers that already have the rare chunk are forced to stay in the net to complete the file. Thus the upload capacity for the rare chunk is a little higher and the experienced download time is a little shorter for the four parallel uploads (cf. Figure 9) than for one upload (cf. Figure 8) in the leeching scenario. However, the last chunk problem occurs for both upload variants.

6.2.2 Least-Shared-First Strategy

The least-shared-first strategy tries to overcome the last chunk problem by favoring rare chunks. In Figure 6 the number of sharing peers is visualized for each chunk using at most one upload connection of a serving peer. In the beginning the different chunks spread equally within the network. But they seem to disseminate more slowly. The number of sharing peers stays much closer together than for the same scenario using random strategy, cf. Figure 5. This is caused by the fact, that after the first peer has downloaded a chunk from the providing peers, no other peer requests this chunk until the other chunks are no longer less shared. Nevertheless, if one chunk is much more frequent than the others, e.g. it was randomly chosen more often than other least-shared chunks (cf. Section 4.2), then the high number of sharing peers will keep this chunk more popular than the others. An example for this is depicted by the 'very popular chunk' in Figure 6. However the least-shared-first strategy with one uplink connection can stabilize the number of sharing peers for all the chunks and therefore improves the download speed.

Getting back to the initial "last chunk" problem description, Figure 1 shows the number of peers sharing a part for the variant with at most four parallel uploads. At the beginning there is only one chunk which is often shared. The

[1]A similar behaviour is observed with four upload connections.

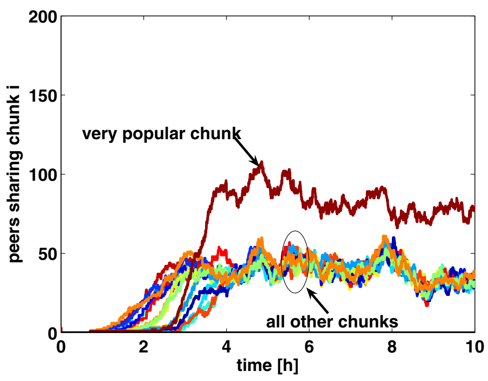

Figure 6. Chunk dissemination in the leeching scenario using the least-shared-first strategy with one upload connection

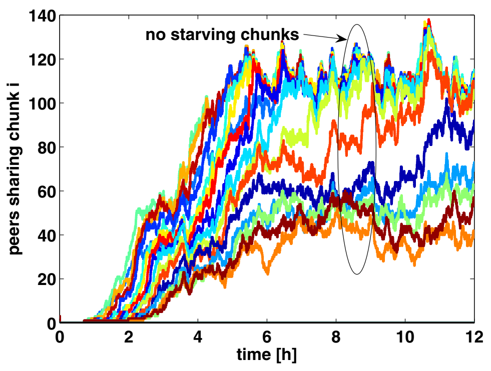

Figure 7. Number of peers sharing a chunk for the leeching scenario using the CygPriM strategy variant with one upload connection[2]

slow reproduction rate in combination with churn leads to the situation that it takes a long time before some more peers have a complete copy of a specific chunk. A chunk that is copied several times becomes independent of the peers churn behaviour. Thus such a chunk will start to be reproduced much faster. In Figure 1 this is shown by the graphs of the different chunks, which rise fast and have gaps between the graphs. Thus the same problem as with the random selection arises. Figure 1 depicts the one causing the last chunk problem as 'starving chunk'. It is sparely shared while the others are heavily and equally distributed.

6.2.3 CycPriM Strategy

Figure 7 shows the number of peers sharing one chunk for the CygPriM strategy using one upload connection for a single simulation run in the leeching scenario. The different chunks spread fast and at the beginning they stay relative close together. The number of peers sharing one chunk is very dynamic, which is caused by the local decision of peers. A peer determines independently of other peers the next chunk to be distributed according to its own, local chunk upload sequence. Unpopular chunks get very popular and vice versa. The most popular chunk is three times more often shared than the most unpopular one. The system is stable within these parameters. During the complete simulation no chunk is starving.

The chunk dissemination of the variant with four parallel upload connections is similar than with one upload and therefore not shown here; all results can be found in technical report [11]. The slower reproduction rate arranges for a less dynamic development of the number of peers sharing a chunk. As a result, this variant has a slightly improved download time.

6.2.4 Consequences on the File Download Time

The consequence of the chunk dissemination on the file download time is shown in this section for the three cooperation strategies. Figure 8 shows the CCDFs of the download time for the three strategies using one upload connection. The least-shared-first strategy as well as the CygPriM strategy provide a much faster file transmission than the random strategy. However, the least-shared-first strategy shows a slightly better performance.

The resulting CCDFs of the download time achieved with the three strategies using four upload connections are plotted in Figure 9. In the case of four upload connection the download times of the least-shared-first strategy is in the same order of magnitude like the one of the random strategy. Both graphs show a slow decay, while the CygPriM strategy still provides a fast file distribution. Neither the random strategy nor the least-shared-first strategy can prevent the last chunk problem, in particular in the case of four peers served in parallel.

These results demonstrate that the use of a specific cooperation strategy has a big impact of the download performance in scenarios with a high number of leeching peers. The random strategy shows weaknesses for the file distribution in the leeching scenario no matter what peer characteristic is assumed. Distributing a file with the least-shared-first strategy in the leeching scenario can be the fastest way. But therefore it is necessary to use only a single upload. Again, the CygPriM strategy has proven to be robust against leeching peers and changes of the number of upload connections. Summarizing, the CygPriM strategy provides short download times in all considered cases.

[2]A similar behaviour is observed with four upload connections.

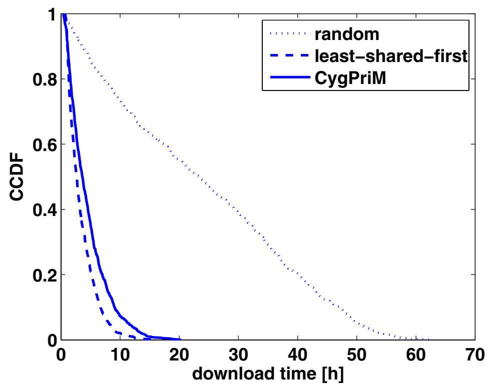

Figure 8. CCDFs of the download time using one upload connection in leeching scenario

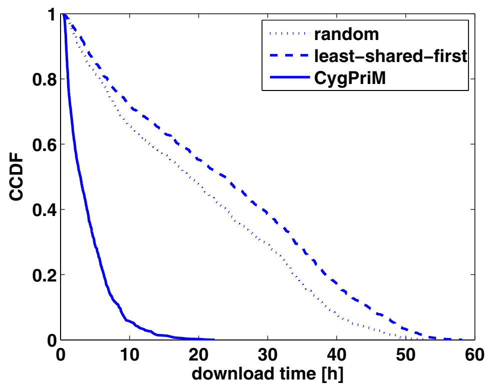

Figure 9. CCDFs of the download time using four upload connections in leeching scenario

7 Conclusions and Outlook

This paper analyses the eDonkey-like random chunk strategy, the BitTorrent-like least-shared-first strategy and the proposed CygPriM strategy for chunk dissemination in P2P content distribution networks using MSD. As performance indicator the user perceived download time and the availability in terms of the number of sharing peers of a chunk is chosen. It is shown, that in cases where most of the peers are selfish, i.e. show a leeching behaviour, the performance of a file distribution P2P network can be significantly improved with an appropriate cooperation strategy. The detailed chunk distribution analysis demonstrates, that the random chunk strategy suffers from the last chunk problem. Thus, this strategy can not guarantee a fast file distribution with leeching peers. The least-shared first strategy overcomes these problems and supports fast file distribution in most cases. However, it is necessary to prevent changes to the peer characteristic, i.e. restrict the number of parallel uploads. The proposed CygPriM strategy is robust against leeching as well as against changes of peer characteristics.

Future work comprises the influence of incentive mechanisms [4] on the system performance. Incentive mechanisms might use other peer selection strategies in order to promote sharing peers and suspend peers that do not upload data.

References

[1] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Proceedings of PAM 2004*, Antibes Juan-les-Pins, France, April 2004.

[2] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of the SIGCOMM '04*, New York, NY, USA, 2004.

[3] A. Al Hamra and P. A. Felber. Design choices for content distribution in P2P networks. *ACM SIGCOMM Computer Communication Review, volume 35, issue 5*, October 2005.

[4] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[5] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom 2005*, Miami, FL, March 2005.

[6] L. Massoulie and M. Vojnovic. Coupon replication systems. In *Proceeding of SIGMETRICS '05*, Banff, Alberta, Canada, June 2005.

[7] P. Felber and E. W. Biersack. Cooperative content distribution: Scalability through self-organization. In *Procceedings of Self-star Properties in Complex Information Systems*, May 2005.

[8] J. Oberender, F.-U. Andersen, H. de Meer, I. Dedinski, T. Hoßfeld, C. Kappler, A. Mäder, and K. Tutschku. Enabling mobile peer-to-peer networking. In *Mobile and Wireless Systems, LNCS 3427*, Germany, January 2005.

[9] T. Hoßfeld, K. Tutschku, F.-U. Andersen, H. de Meer, and J. Oberender. Simulative performance evaluation of a mobile peer-to-peer file-sharing system. In *Procceedings of NGI2005*, Rome, Italy, April 2005.

[10] T. Hoßfeld, K. Tutschku, and D. Schlosser. Influence of the size of swapping entities in mobile P2P file-sharing networks. In *Peer-to-Peer-Systeme und -Anwendungen*, GI/ITG-Workshop in conjunction mit KiVS 2005, Kaiserslautern, Germany, March 2005.

[11] Daniel Schlosser, Tobias Hoßfeld, and Kurt Tutschku. Comparison of robust cooperation strategies for P2P content distribution networks with multiple source download. Technical Report 385, University of Würzburg, May 2006.